[포팅 메뉴얼]

대전B209 [우린 약하지 않아]

1. 프로젝트 기술 스택, 사용 툴

Backend

- JDK 17.0.7
- Spring boot 3.0.2
- MySQL 8.0.33, MySQL workbench 8.0.33
- Nginx 1.18.0
- Jenkins
- AWS EC2
- AWS S3
- Redis 7.0.12

Frontend

- React 18.2.0
- Typesciprt 4.9.5
- Recoil 0.7.7
- Node.js 18.17.1

2. 개발 환경 setting

2-1. Backend

- **ec2 접속**
  - SSH
  - Host Name: i9b209.p.ssafy.io
  - 발급 받은 key입력

- **DB setting**

  1. **mysql 설치하기**

```
sudo apt-get update
sudo apt-get install mysql-server

//mysql 접속
sudo mysql -u root //초기에는 root로 -p없이 로그인
```

  2. **mysql 접속 후**

```
//사용할 database 만들기
create database cartel;

//database 만들어 졌는지 확인하기
show databases;
```

3. **사용할 user 생성하기**

```
//db바꾸기
use mysql;

//사용자에게 권한 주기 '%'->모든 권한 주기
create user '사용자 이름'@'%' identified by '비밀번호';
grant all on 'db이름'.* to '권한 줄 사용자 이름'@'%';

//권한 잘 주어졌는지 확인
show grants for 'cartel'@'%';
```

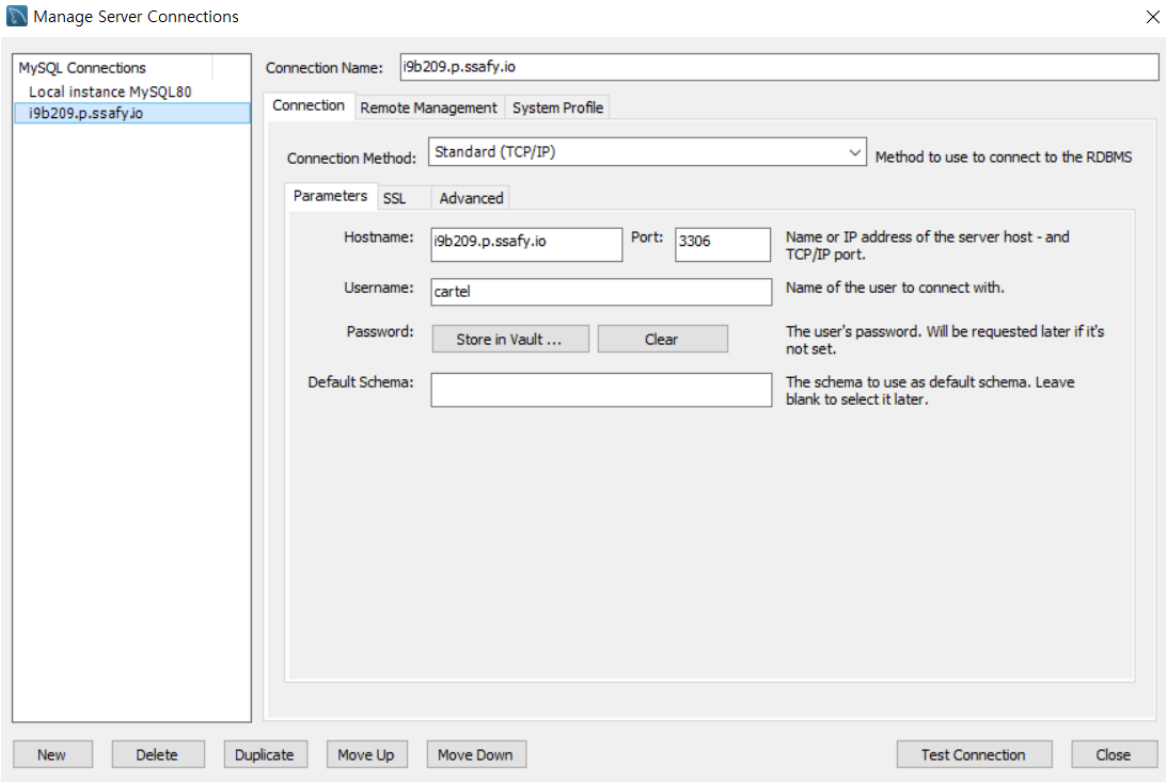
4. **workbench**

- **로컬 workbench에서 ec2 DB에 접속하기 위해 추가적으로 설정해줘야 하는 부분**
  → 127.0.0.1로 설정 되어있는 bind-address값을 0.0.0.0으로 수정해서 외부 접속 허용해준다.(i 눌러서 수정해주기→ ctrl+c (insert 종료) → :wq+enter (저장))

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

5. `해킹 위험 때문에 포트를 변경하기`

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

#해당 설정 파일로 가서 포트번호를 바꾸기
sudo ufw allow 6033 # 바꿀 포트 허용

#껐다 켜기
sudo service mysql stop
sudo service mysql start
```



2-2. Frontend

- 필수 라이브러리 설치

```
cd frontend
npm i
npm start
```

3. Openvidu 설치하기

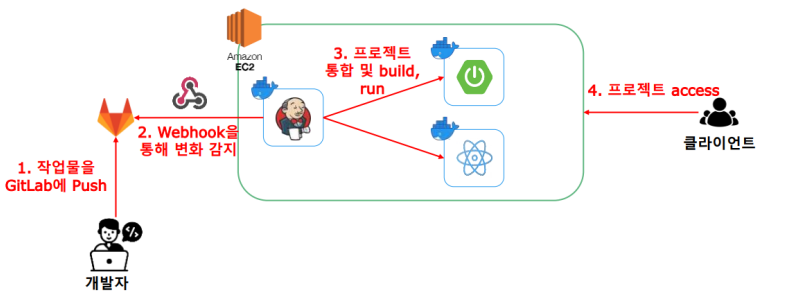
▼ openvidu docs

> On premises - OpenVidu Docs
>
> https://docs.openvidu.io/en/stable/deployment/ce/on-premises/

- **EC2에 openvidu 설치하기**
  openvidu 설치 전에 nginx 포트랑 겹침/ openvidu 설치후 nginx 설치하기 !

```
sudo su
cd /opt

# /opt 위치에 openvidu platfrom 설치!
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash

# openvidu 설치 경로로 이동
cd openvidu
# .env 파일 수정
vip .env

## 수정할 부분!
# (i): 수정하기
# DOMAIN_OR_PUBLIC_IP= 서버도메인
# OPENVIDU_SECRET= 사용할 비밀번호
# CERTIFICATE_TYPE=letsencrypt //인증서 받았으면 letsencrypt로 수정해주기
# LETSENCRYPT_EMAIL=user@example.com //이메일 작성
# (ctrl+c): 작성완료
# (:wq): 저장, 나가기

## 일단 처음에는 80,443 포트 쓰고 ngninx 설치할때 port번호 수정!
# HTTP_PORT=80
# HTTPS_PORT=443```bash


./openvidu start

## 실행 성공하면 나오는 openvidu 서버에 들어가기!
```

4. 배포 환경 Setting



- **EC2에 Docker 설치하기**
  1. docker 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/nul
```

2. docker engine과 plugin 설치

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

sudo apt install docker-compose

#정상 설치 되엇는지 확인
sudo docker -v
sudo docker compose version
```

- **Jenkins 설치하기**

docker-compose를 사용해서 jenkins container를 실행시키기

1. jenkins container를 실행 시킬 docker-compose 만들기

```
sudo vim docker-compose.yml

## 아래 작성 된 사항은 jenkins 기본이미지로 jdk 11버전을 지원해주기 때문에jdk 17이상을 설치 해야 한다면
 ## –> image: jenkins/jenkins:jdk17 이렇게 수정해 주면 됨
## jenkins 기본 포트는 8080인데 9090포트 사용하도록 지정해줌

# (i) : docker - compose 파일에 필요한 스펙을 작성 // 아래 캡쳐 화면 작성하기
# (ctrl+c) : 작성 완료
# (:wq) : 저장, 나오기
```

```
version: '3'

services:
        jenkins:
                image: jenkins/jenkins:jdk17
                container_name: jenkins
                volumes:
                        - /var/run/docker.sock:/var/run/docker.sock
                        - /jenkins:/var/jenkins_home
                ports:
                        - "9090:8080"
                user: root
```

```
sudo docker-compose up -d

##정상적으로 jenkins container가 실행 되고 있는지 확인
sudo docker ps
##정상적으로 실행 되고 있지 않다면 해당 container가 어떤 상태인지 확인
sudo docker ps -a
```

- jenkins 컨테이너 내부에 접속해서 docker 설치

```
sudo docker exec -it jenkins bin/bash

## root~~ 나오면
## jenkins 컨테이너 내부에 접속 완료

##docker 설치
apt-get update
apt-get install ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
```

- jenkins, gitlab → spring boot 배포하기

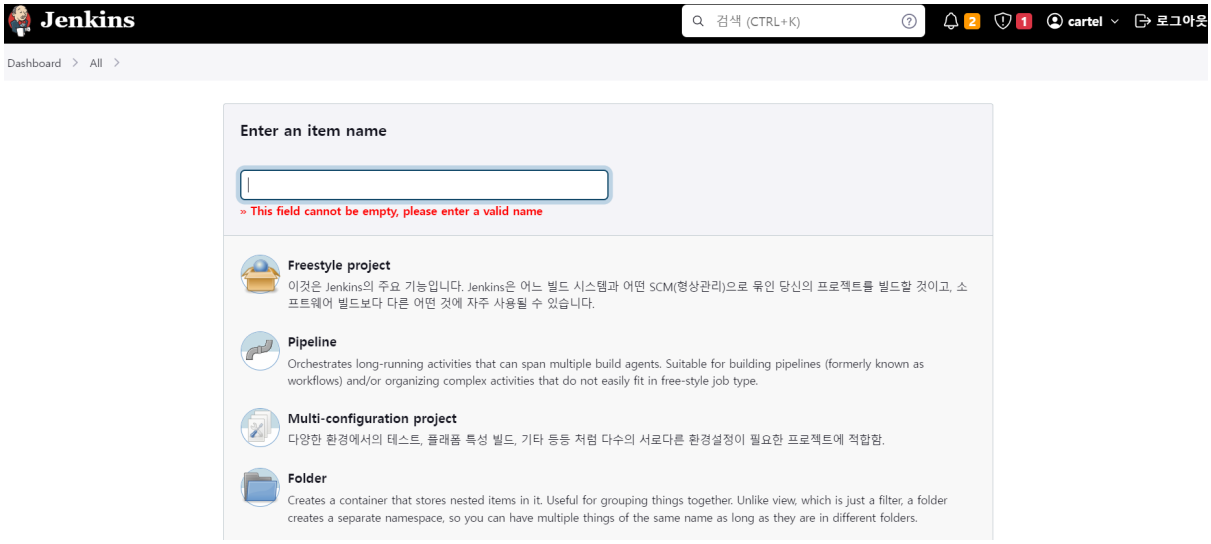
> jenkins : http://도메인:9090포트로 접속

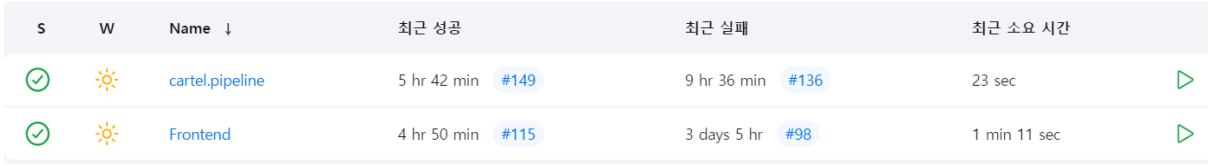
1. 설치 참고 파일 보면서 Jenkins 세팅하기!!!

jenkins 계정

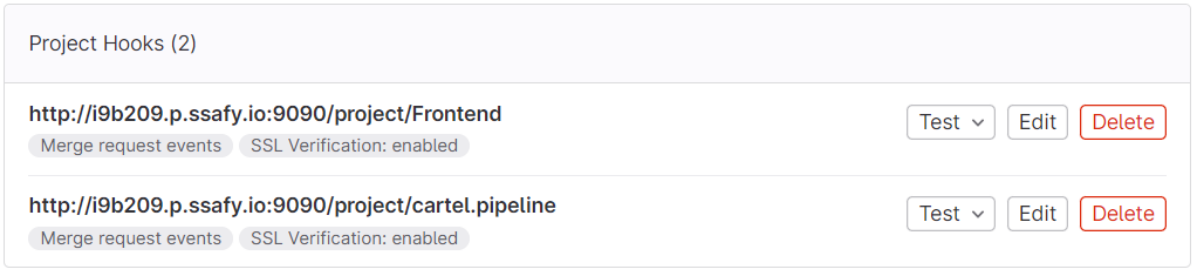
tkadpahfrnspzkfmxpf (삼에몰구네카르텔)
dudehdrudehxoals(영동경도태민)209

- jenkins project 생성





- webhook 생성



2. Jenkins pipeline SCM 작성하기

- Jenkinsfile : pipeline 작성
  - spring boot 폴더에 Jenkinsfile 만들어서 pipeline작성하기

1. Springboot build
   - .jar 파일 생성!!
2. Docker image build
   - docker 이미지 생성
3. Deploy 배포!

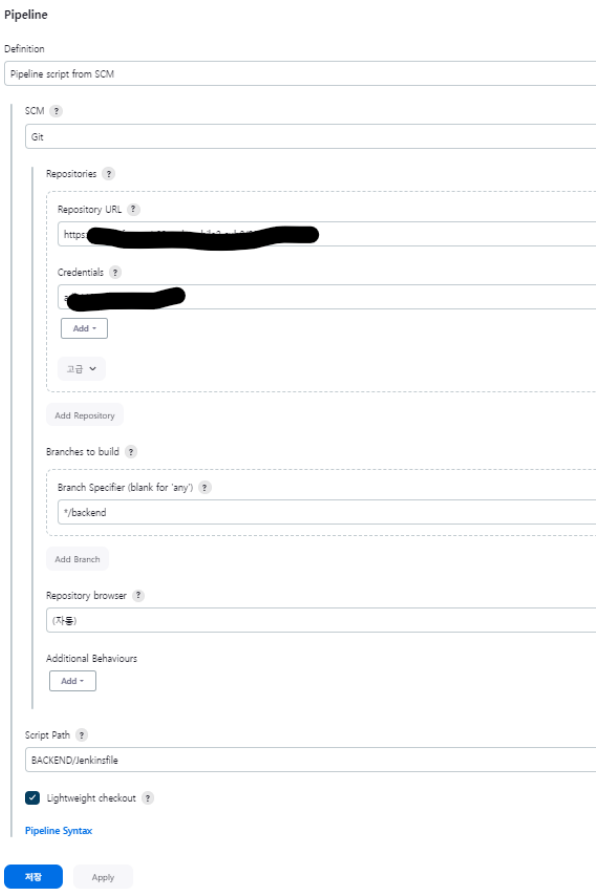
```
pipeline {
    agent any

    stages {
        stage('Springboot build') {
            steps {
                dir('BACKEND'){
                    sh '''
                    echo 'springboot build'
                    chmod +x gradlew
                    ./gradlew clean build //스프링 부트 빌드, .jar파일 생성
                    '''
                }
            }
        }
        stage('Dockerimage build') {
            steps {
                dir('BACKEND'){
                    sh '''
                    echo 'Dockerimage build'
                    docker build -t docker-springboot:0.0.1 .//도커 이미지파일 빌드
                    '''
                }
            }
        }
        stage('Deploy') {
            steps {
                dir('BACKEND'){
                    sh '''
                    echo 'Deploy' //배포
                        // 스프링 구동

                    //최초 처음에 터미널에서 run한번 해줘야해 그 다음부터는 빌드할때마다 자동으로 멈추고 재실행 반복해줌
                    //docker run -d -p 8080:8080 --name springboot docker-springboot:0.0.1

                    docker stop springboot
                    docker rm springboot
                    docker run -d -p 8080:8080 --name springboot docker-springboot:0.0.1
                    '''
                }
            }
        }
    }
}
```

- Docker file 작성
  - spring boot 폴더에 Dockerfile 만들기

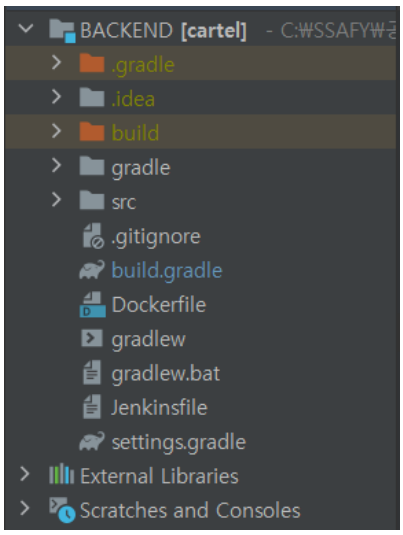
```
FROM openjdk:17-jdk-slim //본인 jdk로 설정!
VOLUME /tmp
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

- pipeline설정 (연결 깃, Jenkinsfile 경로 설정 해주기)

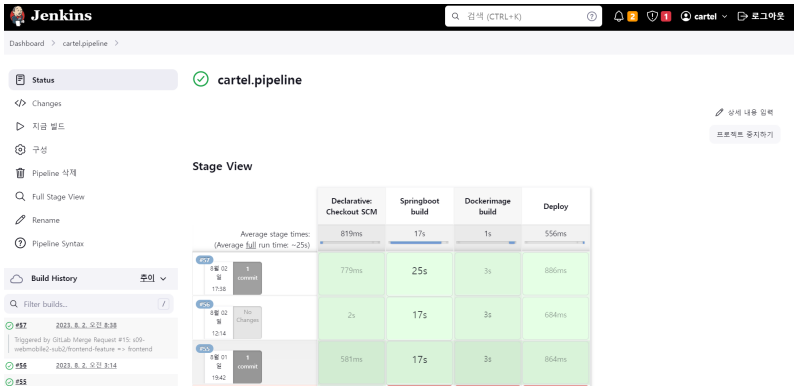
  - 젠킨스 계정 → 구성 → pipeline
  지워진 부분 - 1. 연결 깃 주소 2. 본인 이메일



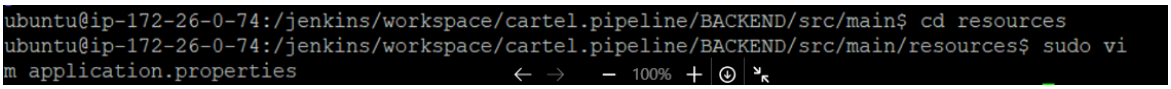
  - 파일 구성(Jenkinsfile, Dockerfile 위치!)



- 초록색으로 뜨면 성공



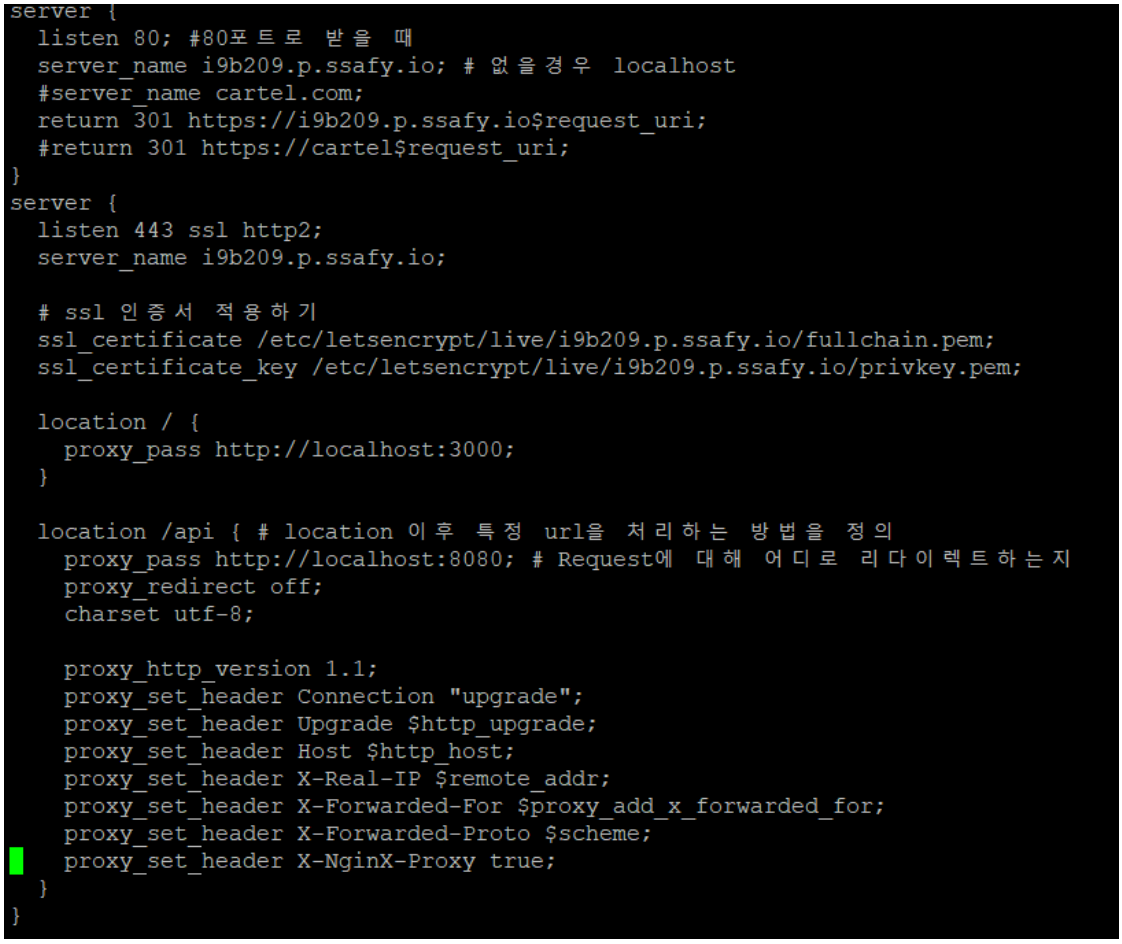
application.properties 파일 서버에 올려주기



- **nginx 설치 & SSL 인증서**

```
#nginx 설치
sudo apt install nginx

#nginx 상태 확인
sudo systemctl status nginx

#let's Encrypt설치
sudo apt-get install letsencrypt

#cerbot 설치
sudo apt-get install certbot python3-certbot-nginx

#certbot동작
sudo certbot --nginx

#방화벽 설정
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
```

```
#nginx 환경설정
sudo vim /etc/nginx/sitex-available/nginx.conf
```



```
server {
  listen 80; #80포트로 받을 때
  server_name i9b209.p.ssafy.io; # 없을경우 localhost
  #server_name cartel.com;
  return 301 https://i9b209.p.ssafy.io$request_uri;
  #return 301 https://cartel$request_uri;
}
server {
  listen 443 ssl http2;
  server_name i9b209.p.ssafy.io;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/i9b209.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/i9b209.p.ssafy.io/privkey.pem;

  location / {
    proxy_pass http://localhost:3000;
  }

  location /api { # location 이후 특정 url을 처리하는 방법을 정의
    proxy_pass http://localhost:8080; # Request에 대해 어디로 리다이렉트하는지
    proxy_redirect off;
    charset utf-8;

    proxy_http_version 1.1;
    proxy_set_header Connection "upgrade";
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-NginX-Proxy true;
  }
}
```

```
#sites-enabled에 심볼릭 링크 생성
sudo ln -s /etc/nginx/sites-available/nginx.conf /etc/nginx/sites-enabled
```

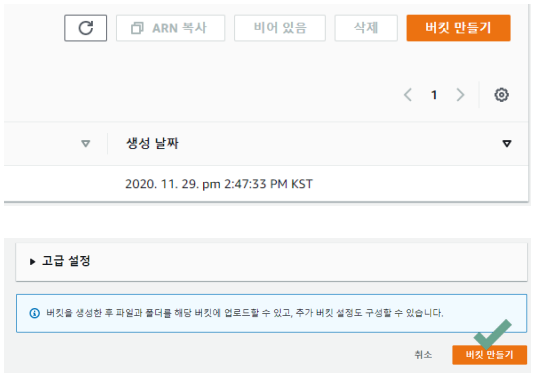
5. Redis

- 회원가입, 비밀번호 찾기 이메일로 인증 번호 전송시에 key-value 값으로 데이터 관리

```
# redis 설치

sudo apt update
sudo apt install redis-server

# conf파일에서 포트 번호 변경해주기
sudo vi /etc/redis/redis.conf
```

6. AWS S3

1. S3 Storage Service 진입
2. Bucket 만들기

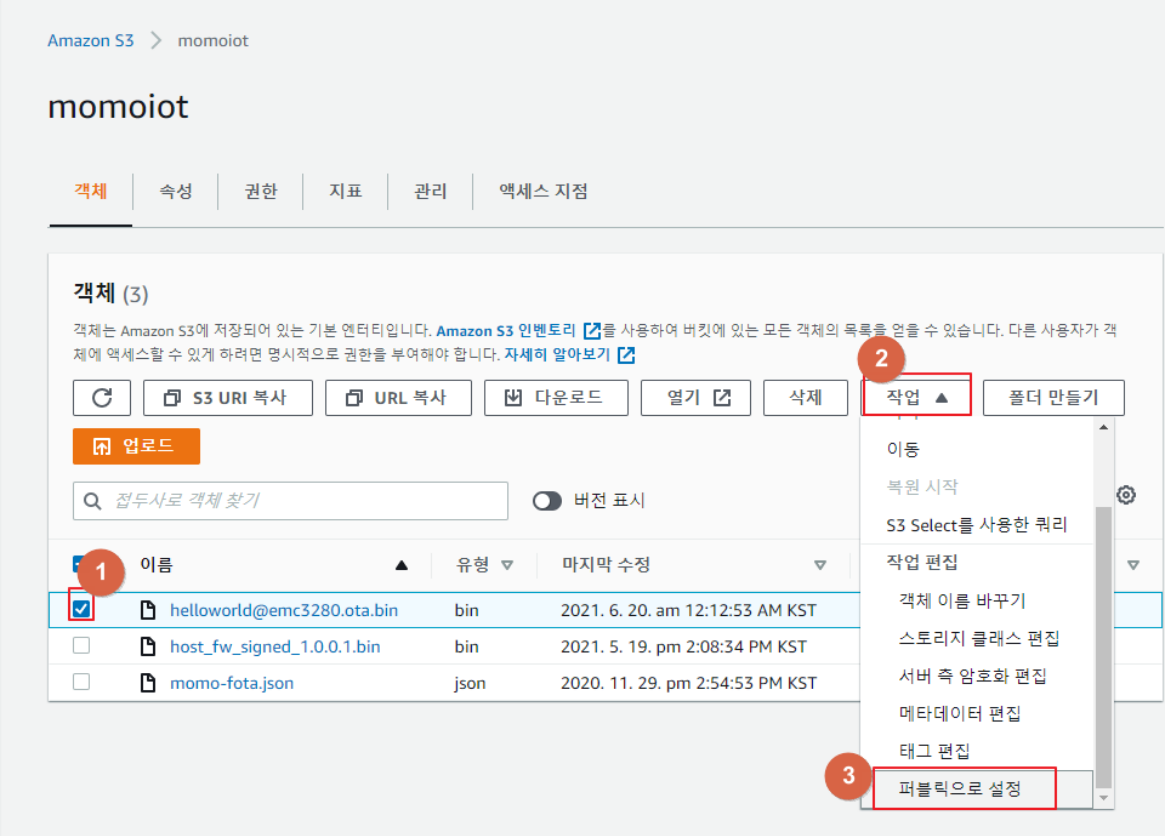


3. Bucket에 File 업로드

4. Bucket과 객체의 Access 권한 설정

   주의사항 : Public Access 차단을 비활성화 한 상태에서는 리소스의 URL을 안다면 누군든지 버킷의 리소스에 액세스가 가능하게 된다.

   이로 인해, AWS로 부터 과도한 과금이 청구 될 수 있으니, 시험을 완료한 후에는 반드시 다시 활성화를 해놓도록 한다.



스프링 연동하기

- build.gradle에 의존성 추가

```
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
```

- application.properties 설정 정보 추가

```
# aws S3
cloud.aws.s3.bucket=cartel-img // 사용할 버킷 이름
cloud.aws.credentials.access-key= <발급받은 accessKey>
cloud.aws.credentials.secret-key= <발급받은 secretKey>
cloud.aws.region.static=ap-northeast-2 // 내 리전 고정
cloud.aws.region.auto=false // 리전 자동 설정 X
cloud.aws.stack.auto=false // ec2에서 spring cloud 프로젝트 실행되지 않게 설정
```

- 스프링 설정 추가
  - S3Config.java

```
@Configuration
public class S3Config {
   @Value("${cloud.aws.credentials.access-key}")
   private String accessKey;
   @Value("${cloud.aws.credentials.secret-key}")
   private String secretKey;
   @Value("${cloud.aws.region.static}")
   private String region;

   @Bean
   public AmazonS3Client amazonS3Client() {
      BasicAWSCredentialsawsCredentials= new BasicAWSCredentials(accessKey, secretKey);
      return (AmazonS3Client)AmazonS3ClientBuilder.standard()
         .withRegion(region)
         .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
```

```
         .build();
   }
}
```

- Upload Controller 생성

```
@RestController
@RequestMapping("/upload")
@RequiredArgsConstructor
public class FileUploadController {

   private final AmazonS3Client amazonS3Client;

   @Value("${cloud.aws.s3.bucket}")
   private String bucket;

   @PostMapping
   public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFilefile) {
      try {
         StringfileName=file.getOriginalFilename();
         StringfileUrl= "https://" + bucket + "/test" +fileName;
         ObjectMetadatametadata= new ObjectMetadata();
metadata.setContentType(file.getContentType());
metadata.setContentLength(file.getSize());
         amazonS3Client.putObject(bucket,fileName,file.getInputStream(),metadata);
         return ResponseEntity.ok(fileUrl);
      } catch (IOExceptione) {
e.printStackTrace();
         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
      }
   }
}
```