



포팅 매뉴얼

Backend 사용 기술

사용 기술	버전
Java	1.8
AWS(Server)	Ubuntu 20.04.4 LTS
Amazon S3	
mysql	8.0.29
Spring boot	2.5.12
Swagger	3.0.0

Frontend 사용 기술

Project	Version
React	18.0.0
Redux	4.2.0
React-redux	8.0.2
Redux-saga	1.1.3
Openvidu-browser	2.21.0
Node.js	16.13.1
npm	8.1.2
Material UI	5.5.2

환경 변수 및 설정 파일

1. AWS 설정 파일 목록
 - a. /home/ubuntu/.bashrc
 - b. /opt/openvidu/.env
 - c. Mysql 계정 설정은 [MySQL Docker] 에 정리
2. Dockerfile
 - a. Gitlab 프로젝트에서 back-end/the_record/Dockerfile
 - b. Gitlab 프로젝트에서 front-end/the-record/Dockerfile
3. CI/CD
 - a. Gitlab 프로젝트 루트 경로의 Jenkinsfile.groovy
4. Back-end 프로젝트 설정 파일
 - a. back-end/the_record/src/main/resources의 .yaml 확장자 파일 전체
 - b. back-end/the_record/build.gradle

AWS Server 접속 및 패키지 관리

서버 접속 및 기본 설정

```
# SSH (Secure Shell) 로 접속 (리눅스의 경우 sudo로 실행)
ssh -i cert.pem ubuntu@k6b204.p.ssafy.io
# 인증 키 오류 발생 시
chmod 600 [Key 파일 이름]
# Permission denied (권한 오류) 발생 시
sudo ssh -i cert.pem ubuntu@k6b204.p.ssafy.io

# update & upgrade
sudo apt update && sudo apt upgrade -y

# docker 설치
# 이전버전 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

# apt update
sudo apt-get update

# 필요한 패키지 설치
sudo apt-get install \
    ca-certificates \
```

```

curl \
gnupg \
lsb-release

# docker GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

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

# apt update
sudo apt-get update

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

# net-tools 설치
sudo apt install net-tools

```

[Mysql] Docker

참고 블로그 : <https://wooono.tistory.com/169>

```

# mysql image 받기
sudo docker pull mysql

# docker image list 보기
sudo docker images

# docker 실행중인 목록
sudo docker ps -al

# Mysql Container 생성 및 db 볼륨 마운트
sudo docker run -d -p 3306:3306 --name mysql_record \
-e MYSQL_ROOT_PASSWORD=<MySQL 비밀번호> \
-v ~/dockerMount/mysqlVol/:/docker-entrypoint-initdb.d/ \
--restart="always" mysql

# docker image 내부 접속
sudo docker exec -it mysql_record bash

# Mysql 접속
mysql -u root -p

# 특정 사용자 생성
create user 'record'@'%' identified by '<MySQL 비밀번호>';
# 특정 사용자에게 권한 부여 (%:모든 곳에서 접속 허용)
grant all privileges on *.* to record@'%';

-----

# docker image 내부 접속
sudo docker exec -it <image_name> bash

```

- 도커 컨테이너 생성 시 포트
 - 3306:3306
 - **host 포트 : 컨테이너 포트**

[Ubuntu] 방화벽 설정

참고 블로그 : <https://milkye.tistory.com/343>

UFW 설치

```

# update & upgrade
sudo apt-get update && sudo apt-get upgrade -y

# ufw 설치
sudo apt-get install ufw

```

UFW 셋팅

```
> sudo ufw allow 22
> sudo ufw enable
> sudo ufw allow 80
> sudo ufw allow 443
> sudo ufw allow 3306
> sudo ufw status
```

- sudo ufw allow 22 : 22번 포트를 허용한다
- sudo ufw enable : 방화벽을 활성화 한다. (*sudo ufw allow 22을 하지 않고 활성화 할경우 ssh가 차단됨)
- sudo ufw allow 80,sudo ufw allow 443 : 80번 포트와 443번 포트를 활성화 한다
- sudo ufw status : 방화벽 상태를 확인한다

22	ALLOW	Anywhere
80	ALLOW	Anywhere
443	ALLOW	Anywhere
3306	ALLOW	Anywhere
8080	ALLOW	Anywhere
9080	ALLOW	Anywhere
8888	ALLOW	Anywhere
9091	ALLOW	Anywhere
22 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
3306 (v6)	ALLOW	Anywhere (v6)
8080 (v6)	ALLOW	Anywhere (v6)
9080 (v6)	ALLOW	Anywhere (v6)
8888 (v6)	ALLOW	Anywhere (v6)
9091 (v6)	ALLOW	Anywhere (v6)

*sudo ufw allow 3306 : 3306포트를 허용해야 Mysql 접속 가능

[Ubuntu] 설정 파일

- /home/ubuntu/.bashrc 파일 (환경 변수)

```
sudo vim ~/.bashrc
# 아래의 내용을 환경변수에 추가

export profile=prod
export MYSQL_DATABASE_URL=<mysql URL>
export MYSQL_DATABASE_USERNAME=<MYSQL 계정명>
export MYSQL_DATABASE_PASSWORD=<MYSQL 계정 비밀번호>
export OPENVIDU_SECRET=<OPENVIDU 비밀번호>
export OPENVIDU_URL=<OPENVIDU URL>
export RECORD_SECRET=<SSL 인증키 비밀번호>
export GMAIL_ID=<SMTP를 사용할 GMAIL 계정명>
export GMAIL_PW=<SMTP를 사용할 GMAIL 계정 비밀번호>
export S3_BUCKET=the-record.bucket
export S3_ACCESSKEY=<Amazon S3 Access KEY>
export S3_SECRET_KEY=<Amazon S3 Secret KEY>

# 저장하고 나온 후, 환경변수 적용
source ~/.bashrc
```

- docker-compose.yml

```
version: '3'

services:
  # 인증서 발급시 1회성으로 nginx를 실행해야 함.
  # nginx:
  #   image: nginx:1.15-alpine
  #   restart: unless-stopped
  #   volumes:
```

```

# - ./data/nginx/default.conf:/etc/nginx/conf.d/default.conf
# - ./data/certbot/conf:/etc/letsencrypt
# - ./data/certbot/www:/var/www/certbot
# ports:
# - "80:80"
# - "443:443"
# command: "/bin/sh -c 'while ;; do sleep 6h & wait $$!}; nginx -s reload; done & nginx -g \"daemon off;\""
certbot: # nginx와 마찬가지로 인증서 발급 후에는 필요 X
image: certbot/certbot
restart: unless-stopped
volumes:
- ./data/certbot/conf:/etc/letsencrypt
- ./data/certbot/www:/var/www/certbot
entrypoint: "/bin/sh -c 'trap exit TERM; while ;; do certbot renew; sleep 12h & wait $$!}; done;'"
# 인증서 발급용 코드 끝
jenkins:
image: 'jenkinsci/blueocean'
restart: unless-stopped
environment:
- profile=prod
- MYSQL_DATABASE_URL=<mysql URL>
- MYSQL_DATABASE_USERNAME=<MYSQL 계정명>
- MYSQL_DATABASE_PASSWORD=<MYSQL 계정 비밀번호>
- OPENVIDU_SECRET=<OPENVIDU 비밀번호>
- OPENVIDU_URL=<OPENVIDU URL>
- RECORD_SECRET=<SSL 인증키 비밀번호>
- GMAIL_ID=<SMTP를 사용할 GMAIL 계정명>
- GMAIL_PW=<SMTP를 사용할 GMAIL 계정 비밀번호>
- S3_BUCKET=the-record.bucket
- S3_ACCESSKEY=<Amazon S3 Access KEY>
- S3_SECRET_KEY=<Amazon S3 Secret KEY>
user: root
privileged: true
ports:
- '9080:8080'
volumes:
- ./dockerMount/jenkins/data/jenkins:/var/jenkins_home
- /var/run/docker.sock:/var/run/docker.sock
- $HOME:/home
- ./data/certbot:/certbot
- ./data/record:/record
container_name: 'jenkins'
#openvidu:
#image: openvidu/openvidu-server-kms:2.21.0
#restart: unless-stopped
#environment:
#- OPENVIDU_URL=<OPENVIDU URL>
#- OPENVIDU_SECRET=<OPENVIDU 비밀번호>
#- OPENVIDU_RECORDING=true
#- OPENVIDU_RECORDING_PATH=/opt/openvidu/recordings
#- DOMAIN_OR_PUBLIC_IP=<사용할 도메인 명>
#volumes:
#- /var/run/docker.sock:/var/run/docker.sock
#- ./dockerMount/openvidu/opt/openvidu/recordings:/opt/openvidu/recordings
#ports:
#- '4443:4443'

```

[Ubuntu] ssl 설정

참고 링크 : <https://github.com/wmnd/nginx-certbot>

<https://velog.io/@hytenic/SSL-Lets-Encrypt-Certbot-docker를-이용하여-SSL-인증서-발급받기>

인증서 발급용 sh파일

```

#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
    echo 'Error: docker-compose is not installed.' >&2
    exit 1
fi

domains=the-record.co.kr
rsa_key_size=4096
data_path="./data/certbot"
email="ssafy0601@gmail.com" # Adding a valid address is strongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data_path" ]; then
    read -p "Existing data found for $domains. Continue and replace existing certificate? (y/N) " decision
    if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
        exit
    fi
fi

```

```

fi
fi

if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ ! -e "$data_path/conf/ssl-dhparams.pem" ]; then
    echo "### Downloading recommended TLS parameters ..."
    mkdir -p "$data_path/conf"
    curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot_nginx/_internal/tls_configs/options-ssl-nginx
    curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl-dhparams.pem > "$data_path/conf/ssl-dhparams.pe
    echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
    openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days 1\
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
    -subj '/CN=localhost'" certbot
echo

echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
    rm -Rf /etc/letsencrypt/live/$domains && \
    rm -Rf /etc/letsencrypt/archive/$domains && \
    rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo

echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]"; do
    domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
    "") email_arg="--register-unsafely-without-email" ;;
    *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
    certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload

```

./data/nginx/default.conf 파일 설정

```

server {
    listen 80;
    server_name the-record.co.kr;
    server_tokens off;

    # return 301 https://$server_name$request_uri;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

```

```
server {
    listen 443 ssl;
    server_name the-record.co.kr;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/the-record.co.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/the-record.co.kr/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://the-record.co.kr;
        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;
    }
}
```

```
# 위 파일 작성 후
# 실행권한 부여
sudo chmod +x init-letsencrypt.sh

# 쉘 스크립트 실행
sudo bash ./init-letsencrypt.sh
```

위 명령어 실행 시, `./data/certbot`에 인증서 생성됨

`./data/certbot/conf/live` 경로에서

```
# spring boot용 인증서 생성
sudo openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem \
    -out the-record.co.kr.p12 --name ssafy -CAfile chain.pem -caname root

# docker network 생성
sudo docker network create recordnetwork
```

`/home/deploy/data/record`에 `.env` 파일 생성

```
# 파일에 다음 내용 붙여넣기
REACT_APP_SERVER_SECRET = 'Openvidu Secret Key'
REACT_APP_REMOVEBG_API_TOKEN = 'Remove.bg API Key'
REACT_APP_GOOGLE_API_TOKEN = 'Google Speech API Key'
REACT_APP_API_HOST = 'Server URL'
```

Openvidu 실행

Openvidu에서 사용할 포트 열기

```
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 5000/tcp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/udp
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
```

openvidu 설치

```
cd /opt
sudo curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | sudo bash
```

openvidu .env 파일 수정

```
cd /opt/openvidu  
  
sudo vim .env
```

.env 파일 내용 수정

```
# OpenVidu configuration  
# -----  
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvidu-config/  
  
# NOTE: This file doesn't need to quote assignment values, like most shells do.  
# All values are stored as-is, even if they contain spaces, so don't quote them.  
  
# Domain name. If you do not have one, the public IP of the machine.  
# For example: 198.51.100.1, or openvidu.example.com  
DOMAIN_OR_PUBLIC_IP=<사용할 도메인 명>  
  
# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard  
#OPENVIDU_SECRET=record1014  
OPENVIDU_SECRET={사용할 비밀번호}  
  
# Certificate type:  
# - selfsigned: Self signed certificate. Not recommended for production use.  
#               Users will see an ERROR when connected to web page.  
# - owncert: Valid certificate purchased in a Internet services company.  
#            Please put the certificates files inside folder ./owncert  
#            with names certificate.key and certificate.cert  
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the  
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL  
#                variable.  
CERTIFICATE_TYPE=letsencrypt  
  
# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications  
LETSENCRYPT_EMAIL={사용할 이메일 주소}  
  
# Proxy configuration  
# If you want to change the ports on which openvidu listens, uncomment the following lines  
  
# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically  
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.  
# WARNING: the default port 80 cannot be changed during the first boot  
# if you have chosen to deploy with the option CERTIFICATE_TYPE=letsencrypt  
HTTP_PORT=8445  
  
# Changes the port of all services exposed by OpenVidu.  
# SDKs, REST clients and browsers will have to connect to this port  
HTTPS_PORT=4443  
  
# Old paths are considered now deprecated, but still supported by default.  
# OpenVidu Server will log a WARN message every time a deprecated path is called, indicating  
# the new path that should be used instead. You can set property SUPPORT_DEPRECATED_API=false  
# to stop allowing the use of old paths.  
# Default value is true  
# SUPPORT_DEPRECATED_API=true  
  
# If true request to with www will be redirected to non-www requests  
# Default value is false  
# REDIRECT_WWW=false  
  
# How many workers to configure in nginx proxy.  
# The more workers, the more requests will be handled  
# Default value is 10240  
# WORKER_CONNECTIONS=10240  
  
# Access restrictions  
# In this section you will be able to restrict the IPs from which you can access to  
# Openvidu API and the Administration Panel  
# WARNING! If you touch this configuration you can lose access to the platform from some IPs.  
# Use it carefully.  
  
# This section limits access to the /dashboard (OpenVidu CE) and /inspector (OpenVidu Pro) pages.  
# The form for a single IP or an IP range is:  
# ALLOWED_ACCESS_TO_DASHBOARD=198.51.100.1 and ALLOWED_ACCESS_TO_DASHBOARD=198.51.100.0/24  
# To limit multiple IPs or IP ranges, separate by commas like this:  
# ALLOWED_ACCESS_TO_DASHBOARD=198.51.100.1, 198.51.100.0/24  
# ALLOWED_ACCESS_TO_DASHBOARD=  
  
# This section limits access to the Openvidu REST API.
```

```

# The form for a single IP or an IP range is:
# ALLOWED_ACCESS_TO_RESTAPI=198.51.100.1 and ALLOWED_ACCESS_TO_RESTAPI=198.51.100.0/24
# To limit multiple IPs or or IP ranges, separate by commas like this:
# ALLOWED_ACCESS_TO_RESTAPI=198.51.100.1, 198.51.100.0/24
# ALLOWED_ACCESS_TO_RESTAPI=

# Whether to enable recording module or not
OPENVIDU_RECORDING=false

# Use recording module with debug mode.
OPENVIDU_RECORDING_DEBUG=false

# Openvidu Folder Record used for save the openvidu recording videos. Change it
# with the folder you want to use from your host.
OPENVIDU_RECORDING_PATH=/opt/openvidu/recordings

# System path where OpenVidu Server should look for custom recording layouts
OPENVIDU_RECORDING_CUSTOM_LAYOUT=/opt/openvidu/custom-layout

# if true any client can connect to
# https://OPENVIDU_SERVER_IP:OPENVIDU_PORT/recordings/any_session_file.mp4
# and access any recorded video file. If false this path will be secured with
# OPENVIDU_SECRET param just as OpenVidu Server dashboard at
# https://OPENVIDU_SERVER_IP:OPENVIDU_PORT
# Values: true | false
OPENVIDU_RECORDING_PUBLIC_ACCESS=false

# Which users should receive the recording events in the client side
# (recordingStarted, recordingStopped). Can be all (every user connected to
# the session), publisher_moderator (users with role 'PUBLISHER' or
# 'MODERATOR'), moderator (only users with role 'MODERATOR') or none
# (no user will receive these events)
OPENVIDU_RECORDING_NOTIFICATION=publisher_moderator

# Timeout in seconds for recordings to automatically stop (and the session involved to be closed)
# when conditions are met: a session recording is started but no user is publishing to it or a session
# is being recorded and last user disconnects. If a user publishes within the timeout in either case,
# the automatic stop of the recording is cancelled
# 0 means no timeout
OPENVIDU_RECORDING_AUTOSTOP_TIMEOUT=120

# Maximum video bandwidth sent from clients to OpenVidu Server, in kbps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MAX_RECV_BANDWIDTH=1000

# Minimum video bandwidth sent from clients to OpenVidu Server, in kbps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MIN_RECV_BANDWIDTH=300

# Maximum video bandwidth sent from OpenVidu Server to clients, in kbps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MAX_SEND_BANDWIDTH=1000

# Minimum video bandwidth sent from OpenVidu Server to clients, in kbps.
# 0 means unconstrained
OPENVIDU_STREAMS_VIDEO_MIN_SEND_BANDWIDTH=300

# All sessions of OpenVidu will try to force this codec. If OPENVIDU_STREAMS_ALLOW_TRANSCODING=true
# when a codec can not be forced, transcoding will be allowed
# Values: MEDIA_SERVER_PREFERRED, NONE, VP8, VP9, H264
# Default value is MEDIA_SERVER_PREFERRED
# OPENVIDU_STREAMS_FORCED_VIDEO_CODEC=MEDIA_SERVER_PREFERRED

# Allow transcoding if codec specified in OPENVIDU_STREAMS_FORCED_VIDEO_CODEC can not be applied
# Values: true | false
# Default value is false
# OPENVIDU_STREAMS_ALLOW_TRANSCODING=false

# true to enable OpenVidu Webhook service. false' otherwise
# Values: true | false
OPENVIDU_WEBHOOK=false

# HTTP endpoint where OpenVidu Server will send Webhook HTTP POST messages
# Must be a valid URL: http(s)://ENDPOINT
#OPENVIDU_WEBHOOK_ENDPOINT=

# List of headers that OpenVidu Webhook service will attach to HTTP POST messages
#OPENVIDU_WEBHOOK_HEADERS=

# List of events that will be sent by OpenVidu Webhook service
# Default value is all available events
OPENVIDU_WEBHOOK_EVENTS=[sessionCreated,sessionDestroyed,participantJoined,participantLeft,webrtcConnectionCreated,webrtcConnectionDes

# How often the garbage collector of non active sessions runs.
# This helps cleaning up sessions that have been initialized through
# REST API (and maybe tokens have been created for them) but have had no users connected.
# Default to 900s (15 mins). 0 to disable non active sessions garbage collector

```



```

OPENVIDU_SESSIONS_GARBAGE_INTERVAL=900

# Minimum time in seconds that a non active session must have been in existence
# for the garbage collector of non active sessions to remove it. Default to 3600s (1 hour).
# If non active sessions garbage collector is disabled
# (property 'OPENVIDU_SESSIONS_GARBAGE_INTERVAL' to 0) this property is ignored
OPENVIDU_SESSIONS_GARBAGE_THRESHOLD=3600

# Call Detail Record enabled
# Whether to enable Call Detail Record or not
# Values: true | false
OPENVIDU_CDR=false

# Path where the cdr log files are hosted
OPENVIDU_CDR_PATH=/opt/openvidu/cdr

# Kurento Media Server image
# -----
# Docker hub kurento media server: https://hub.docker.com/r/kurento/kurento-media-server
# Uncomment the next line and define this variable with KMS image that you want use
# KMS_IMAGE=kurento/kurento-media-server:6.16.0

# Kurento Media Server Level logs
# -----
# Uncomment the next line and define this variable to change
# the verbosity level of the logs of KMS
# Documentation: https://doc-kurento.readthedocs.io/en/stable/features/logging.html
# KMS_DOCKER_ENV_GST_DEBUG=

# Openvidu Server Level logs
# -----
# Uncomment the next line and define this variable to change
# the verbosity level of the logs of Openvidu Service
# RECOMENDED VALUES: INFO for normal logs DEBUG for more verbose logs
# OV_CE_DEBUG_LEVEL=INFO

# Java Options
# -----
# Uncomment the next line and define this to add
# options to java command
# Documentation: https://docs.oracle.com/cd/E37116_01/install.111210/e23737/configuring_jvm.htm#OUDIG00058
# JAVA_OPTIONS=-Xms2048m -Xmx4096m -Duser.timezone=UTC

```

openvidu 서버 실행

```

# openvidu 실행
./openvidu start
# openvidu 재실행
./openvidu restart

```

[S3] bucket, IAM 설정

출처 : https://velog.io/@_koiii/Springboot-AWS-S3로-파일-저장소-연동하기

Bucket 설정

bucket 생성

1. bucket 생성

서비스

서비스, 기능, 블로그, 설명서 등을 검색합니다.

[Option+S]

더 빠르고 쉽게 사용할 수 있도록 S3 콘솔을 계속 개선하고 있습니다. 업데이트된 환경에 대한 피드백이 있는 경우 [피드백 제공]을 선택합니다.

일반 구성

버킷 이름

bucket-name

버킷 이름은 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. [버킷 이름 지정 규칙 참조](#)

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지정 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 지정에만 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(엑세스 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(엑세스 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 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
 S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지정 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지정 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 지정에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

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

정책은 버킷에 속한 객체 간의 액세스를 허용합니다. 이는 공개된 퍼블릭 액세스를 허용하는 경우와 다른 모든 퍼블릭 액세스

- bucket 이름은 바로 아래 버킷 이름 지정 규칙 참고
- 퍼블릭 액세스 차단을 모두 풀어야 spring이 S3로 접근가능하며, bucket 정책 수정이 가능

버킷 정책 편집

1. 버킷 정책 편집으로 이동

- 생성된 버킷 선택 > 권한 탭 > 버킷 정책 > 편집으로 이동
- 버킷 정책 편집 선택

버킷 정책

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

편집

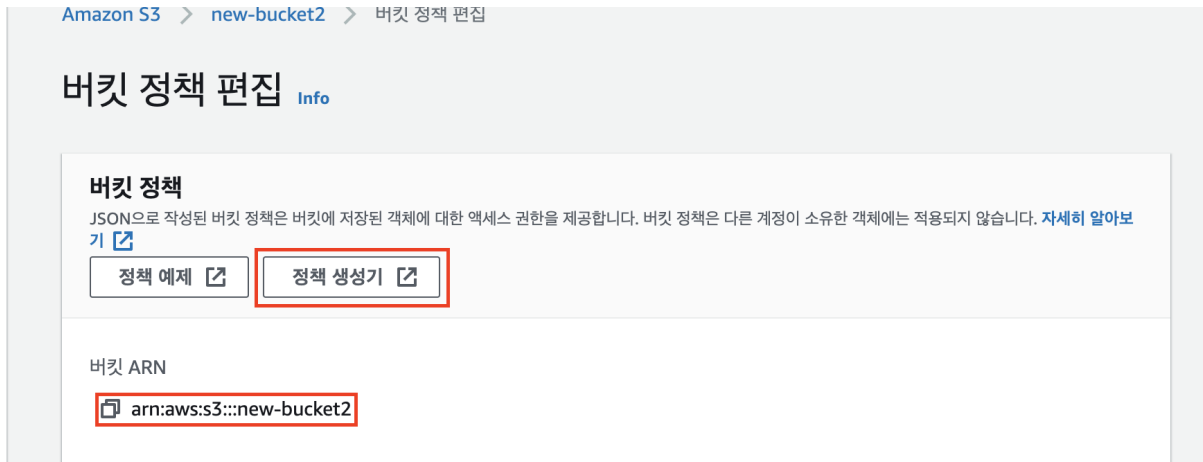
표시할 정책이 없습니다.

2. 정책 생성기로 정책 생성

- 버킷 정책 편집에서 정책 생성기 선택
- 버킷 ARN 복사 필요

포팅 매뉴얼

10



- Type of Policy : S3 Bucket Policy 선택
- principal : *
- Actions : * 모두 허용 또는 GetObject, PutObject, DeleteObject 등 필요 기능 선택
- ARN : 복사한 bucket ARN 입력
- Add Statement 클릭

creating 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 Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS 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 -- Select Actions -- ☒ All Actions (*)

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

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

Add Statement

- Generate Policy로 정책 생성해서 복사하기

Principal(s)	Effect	Action	Resource	Conditions
* *	Allow	s3:*	arn:aws:s3:::new-bucket2	None

Step 3: Generate Policy

A *policy* is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy [Start Over](#)

- 버킷 정책에 생성한 버킷 정책 붙여넣기

정책

```
1 {  
2   "Id": "Policy1637493633297",  
3   "Version": "2012-10-17",  
4   "Statement": [  
5     {  
6       "Sid": "Stmt1637493415559",  
7       "Action": "s3:*",  
8       "Effect": "Allow",  
9       "Resource": "arn:aws:s3:::new-bucket2",  
10      "Principal": "*"   
11    }  
12  ]  
13 }
```

IAM 추가

1. 사용자 추가

- 액세스 키 - 프로그래밍 방식 액세스로 AWS 자격 증명 유형 선택

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* s3user

[+ 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.


- 기존 정책 직접 연결 선택
- AmazonS3FullAccess 정책 선택


사용자 추가

1 2 3 4 5

▼ 권한 설정

 그룹에 사용자 추가

 기존 사용자에서 권한 복사

 기존 정책 직접 연결

정책 생성

↺

정책 필터		Q S3F		1 결과 표시	
정책 이름		유형		사용 용도	
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS 관리형		Permissions policy (1)	

▶ 권한 경계 설정

- 3, 4 페이지 넘어가기
- 5 페이지에서 .csv 다운로드해서 Access-key, Secret-key 보관하기
- 추후 Spring 설정에 필요

사용자 추가

1 2 3 4 5

성공

아래에 표시된 사용자를 생성했습니다. 사용자 보안 자격 증명을 보고 다운로드할 수 있습니다. AWS Management Console 로그인을 위한 사용자 지침을 이메일로 보낼 수도 있습니다. 지금 이 자격 증명을 다운로드할 수 있는 마지막 기회입니다. 하지만 언제든지 새 자격 증명을 생성할 수 있습니다.

AWS Management Console 액세스 권한이 있는 사용자가 <https://702608302891.signin.aws.amazon.com/console>에 로그인할 수 있습니다.

.csv 다운로드

	사용자	액세스 키 ID	비밀 액세스 키
▶	✓ s3user		***** 표시

[S3] Spring Boot, S3 설정

참고자료

- <https://antdev.tistory.com/93>
 - AwsConfig 파일 없어서 이상.. 이사람은 왜 됐는지 모르겠음
 - 하지만 package 구조가 제일 깔끔해서 선택
- <https://earth-95.tistory.com/117>
 - AwsConfig 파일 설정

1. build gradle 설정

```
// S3
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
```

2. application 설정

```
cloud:
  aws:
    credentials:
      access-key: 'IAM ACCESS Key' # IAM ACCESS Key
      secret-key: 'IAM SECRET Key' # IAM SECRET Key
    region:
      static: ap-northeast-2 # bucket에서 설정한 지역(아시아 태평양(서울))
      auto: false #
    s3:
      bucket: the-record.bucket
    stack:
      auto: false

logging:
  level:
    com:
      amazonaws:
        util:
          EC2MetadataUtils: ERROR
```

- cloud.aws.stack.auto = false

- Spring Cloud AWS는 기본적으로 서비스가 Cloud Formation 스택내에서 실행된다고 가정하기 때문에 그렇지 않은 경우 임의로 `false` 값으로 설정을 해줘야됩니다.
- `logging.level.com.amazonaws.util.EC2MetadataUtils = ERROR`
 - 불필요한 Warning 로그 제거

```
2022-03-07 10:16:16.883 WARN 2568 --- [ restartedMain] com.amazonaws.util.EC2MetadataUtils : Unable to retrieve the request
```

3. SpringBootApplication 설정

```
@SpringBootApplication
public class TheRecordApplication {

    static {
        System.setProperty("com.amazonaws.sdk.disableEc2Metadata", "true");
    }

    public static void main(String[] args) {
        SpringApplication.run(TheRecordApplication.class, args);
    }

}
```

- `com.amazonaws.sdk.disableEc2Metadata` 속성을 true로 설정해주는것입니다.

만약 해당 설정을 하지 않을 경우 서비스가 실행되는 시점에 약간의 지연이 발생하고 다음과 같은 예외 메시지가 발생합니다.

```
2022-03-07 10:16:16.883 WARN 2568 --- [ restartedMain] com.amazonaws.util.EC2MetadataUtils : Unable to retrieve the request

Caused by: java.net.ConnectException: Host is down (connect failed)
    at java.base/java.net.PlainSocketImpl.socketConnect(Native Method) ~[na:na]
    at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:399) ~[na:na]
    at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:242) ~[na:na]
```

4. Aws S3 Config

```
@Configuration
public class AwsS3Config {

    @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() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(accessKey, secretKey);
        return (AmazonS3Client) AmazonS3ClientBuilder.standard()
            .withRegion(region)
            .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
            .build();
    }

}
```

- AwsS3Client Bean 등록

5. FileUploadService

```
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.record.the_record.s3.dto.FileDetailDto;
import com.record.the_record.s3.util.MultipartUtil;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
```

```

import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;

@Service
@RequiredArgsConstructor
public class FileUploadService {

    @Value("${cloud.aws.s3.bucket}")
    private String bucket;
    private final AmazonS3Client amazonS3Client; // Config에서 등록한 AwsS3Client 빈이 주입됨

    public FileDetailDto save(MultipartFile multipartFile, Long userPk) {
        FileDetailDto fileDetailDto = FileDetailDto.convertFile(userPk, multipartFile); // multipartfile을 FileDetailDto로 변환
        File file = new File(MultipartUtil.getLocalHomeDirectory(), fileDetailDto.getOriginName());
        try {
            multipartFile.transferTo(file); // 파일 변환(putObject에 넣기 위해)
            amazonS3Client.putObject(new PutObjectRequest(bucket, fileDetailDto.getUploadName(), file) // S3에 Upload
                .withCannedAcl(CannedAccessControlList.PublicRead)); // CannedAccessControlList.PublicRead로 설정해야 누구나 파
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (file.exists()) {
                file.delete();
            }
        }
        return fileDetailDto;
    }
}

```

6. 업로드할 파일 DTO

- 사진의 확장자, 원본 이름, 업로드 이름 등의 DTO

```

import com.record.the_record.s3.util.MultipartUtil;
import lombok.*;
import org.springframework.web.multipart.MultipartFile;

import java.time.LocalDateTime;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor(access = AccessLevel.PROTECTED)
@Builder
public class FileDetailDto {
    private String format; // 이미지 확장자
    private String originName; // 원래 이미지 이름
    private String uploadName; // 유저 PK / 날짜 + UUID로 변환된 업로드 이름

    @Builder.Default
    private LocalDateTime created = LocalDateTime.now();

    public static FileDetailDto convertFile(Long userPk, MultipartFile multipartFile) {
        final String fileId = MultipartUtil.createFileId();
        final String format = MultipartUtil.getFormat(multipartFile.getContentType());
        return FileDetailDto.builder()
            .format(format)
            .originName(multipartFile.getOriginalFilename())
            .uploadName(MultipartUtil.createPath(userPk, fileId, format))
            .build();
    }
}

```

7. MultipartFile Util

- multipartfile에서 이미지 변환시 필요한 메소드 정의
- 홈 디렉터리 반환, UUID 생성, 파일확장자 반환, S3 저장 경로 생성 메소드 정의

```

import org.springframework.util.StringUtils;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;

public final class MultipartUtil {

```

```

/**
 *
 * @return 홈 디렉터리 반환
 */
public static String getLocalHomeDirectory() {
    return System.getProperty("user.home");
}

/**
 *
 * @return UUID값 반환
 */
public static String createFileId() {
    return UUID.randomUUID().toString();
}

/**
 *
 * @param contentType 확장자
 * @return 파일 확장자 반환
 */
public static String getFormat(String contentType) {
    if(StringUtils.hasText(contentType)) {
        return contentType.substring(contentType.lastIndexOf('/') + 1);
    }
    return null;
}

/**
 *
 * @param userPk 유저 PK
 * @param fileId 저장할 파일 이름(UUID)
 * @param format 파일 확장자
 * @return 유저PK/날짜_UUID.확장자 형태로 반환
 */
public static String createPath(Long userPk, String fileId, String format) {
    return String.format("%s/%s.%s", userPk, new SimpleDateFormat("yyMMdd").format(new Date()) + "_" + fileId, format);
}
}

```