



PORTING MANUAL

목차

I. 빌드 및 배포

1. 실행 환경 – 2p
2. 설정 파일 목록 – 2p
3. 설정 파일 및 환경 변수 정보 – 3p
4. Openvidu 배포 및 ProxyServer를 위한 포트 개방 – 22p
5. SSL 인증서 발급 – 23p
6. Jenkins에 Gitlab Webhook 연결 – 23p

II. 외부 서비스

1. Amazon S3 – 27p

I. 빌드 및 배포

1. 실행 환경

- a. Server : AWS EC2 Ubuntu 20.04.6 LTS
- b. Visual Studio Code : 1.81
- c. IntelliJ IDEA : 2023.1.3 (Ultimate Edition) 17.0.7+10-b829.16 amd64
- d. JVM : Zulu OpenJDK 11
- e. Docker : 24.0.5
- f. Node.js : 14.17.0
- g. MySQL : 8.0.34
- h. Redis : 7.0.12
- i. Jenkins : 2.416
- j. Openvidu : 2.28.0
- k. TypeScript : 4.9.5
- l. Cloud Storage : AWS S3

2. 설정 파일 목록 (경로와 파일은 배포 과정 중 생성 될 수 있음)

- a. React
 - i. .env : /var/jenkins_home/workspace/Dasoni/frontend
 - ii. Dockerfile : /var/jenkins_home/workspace/Dasoni/frontend/docker

iii. entrypoint.sh: /var/jenkins_home/workspace/Dasoni/frontend/docker

iv. create_image.sh:
/var/jenkins_home/workspace/Dasoni/frontend/docker

b. Spring Boot

i. application.properties:
/var/jenkins_home/workspace/Dasoni/backend/src/main/resources

ii. application.yml:
/var/jenkins_home/workspace/Dasoni/backend/src/main/resources

iii. deploy.sh: /home/Ubuntu

c. Docker

i. docker-compose.yml: /opt/openvidu

d. Nginx

i. project.conf: /etc/nginx/sites-available

e. Openvidu

i. .env : /opt/openvidu

3. 설정 파일 및 환경 변수 정보

a. React

i. .env

DANGEROUSLY_DISABLE_HOST_CHECK=true

ii. Dockerfile

FROM node:18.16.1

COPY . ./openvidu-react

WORKDIR /openvidu-react

Install openvidu-react dependencies and build it

RUN npm install && ₪

npm run build && ₪

cp -r ./build/ ./openvidu-basic-node/public

Copy openvidu-basic-node

RUN cp -r ./openvidu-basic-node /opt/openvidu-basic-node && ₪

rm -rf ./openvidu-react

Install openvidu-basic-node dependencies

RUN npm --prefix /opt/openvidu-basic-node install

WORKDIR /opt/openvidu-basic-node

COPY docker/entrypoint.sh .

ENTRYPOINT ["./entrypoint.sh"]

iii. entrypoint.sh

```
#!/bin/sh
```

```
exec node index.js "$@"
```

iv. create_image.sh

```
#!/bin/bash
```

```
if [ $# -eq 0 ]; then
```

```
    echo "No version argument provided. Usage: $0 <IMAGE_NAME> <VERSION>"
```

```
    exit 1
```

```
fi
```

```
pushd ../
```

```
cp -r ../openvidu-basic-node .
```

```
trap 'rm -rf ./openvidu-basic-node' ERR
```

```
docker build --pull --no-cache --rm=true -f docker/Dockerfile -t "$1" .
```

```
rm -rf ./openvidu-basic-node
```

b. Spring

i. application.properties

security:

jwt:

secret: {JWT Secret}

logging:

level:

org.hibernate.sql: debug

org.hibernate.type: trace

cloud:

aws:

credentials:

access-key: {S3 Access Key}

secret-key: {S3 Secret Key}

region:

static: ap-northeast-2

stack:

auto: false

ii. application.yml

server.port=8081

MySQL ₩uC124₩uC815

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

DB Source URL

spring.datasource.url=jdbc:mysql://{DB컨테이너}:3306/{Schema}?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul

DB username

spring.datasource.username=admin

DB password

spring.datasource.password={admin Password}

true ₩uC124₩uC815₩uC2DC JPA ₩uCFFC₩uB9AC₩uBB38 ₩uD655₩uC778 ₩uAC00₩uB2A5

spring.jpa.show-sql=true

DDL(create, alter, drop) ₩uC815₩uC758₩uC2DC DB₩uC758 ₩uACE0₩uC720 ₩uAE30₩uB2A5₩uC744 ₩uC0AC₩uC6A9₩uD560 ₩uC218 ₩uC788₩uB2E4.

spring.jpa.hibernate.ddl-auto=update

JPA₩uC758 ₩uAD6C₩uD604₩uCCB4₩uC778 Hibernate₩uAC00 ₩uB3D9₩uC791₩uD558₩uBA74₩uC11C ₩uBC1C₩uC0DD₩uD55C SQL₩uC758

WuAC00WuB3C5WuC131WuC744 WuB192WuC5ECWuC900WuB2E4.

spring.jpa.properties.hibernate.format_sql=true

spring.mvc.async.request-timeout=900000

spring.mvc.pathmatch.matching-strategy=ant_path_matcher

spring.devtools.livereload.enabled=true

spring.devtools.restart.enabled=false

spring.freemarker.cache=false

spring.jackson.serialization.fail-on-empty-beans=false

redis

spring.redis.host={Redis 컨테이너}

spring.redis.port=6379

iii. deploy.sh

#!/bin/bash

tar -xvf build.tar

rm -rf build.tar

sudo service nginx restart

```
pid=$(pgrep -f heartsigniel)
```

```
echo ">pid 확인 $pid"
```

```
if [ -n "${pid}" ]
```

```
then
```

```
    kill -15 ${pid}
```

```
    echo kill process ${pid}
```

```
else
```

```
    echo no process
```

```
fi
```

```
chmod +x ./Dasoni/heartsigniel-0.0.1-SNAPSHOT.jar
```

```
nohup java -jar ./Dasoni/heartsigniel-0.0.1-SNAPSHOT.jar >>  
application.log 2> /dev/null &
```

c. Docker

i. docker-compose.yml

```
version: '3.1'
```

services:

openvidu-server:

image: openvidu/openvidu-server:2.28.0

restart: on-failure

network_mode: host

entrypoint: ['/usr/local/bin/entrypoint.sh']

volumes:

- ./coturn:/run/secrets/coturn
- /var/run/docker.sock:/var/run/docker.sock
- \${OPENVIDU_RECORDING_PATH}:\${OPENVIDU_RECORDING_PATH}
-

\${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:\${OPENVIDU_RECORDING_CUSTOM_LAYOUT}

- \${OPENVIDU_CDR_PATH}:\${OPENVIDU_CDR_PATH}
- /opt/openvidu/.env:/env

env_file:

- /opt/openvidu/.env

environment:

- SERVER_SSL_ENABLED=false
- SERVER_PORT=5443
- KMS_URI=["ws://localhost:8888/kurento"]
- COTURN_IP=\${COTURN_IP:-auto-ipv4}
- COTURN_PORT=\${COTURN_PORT:-3478}

logging:

options:

max-size: "\${DOCKER_LOGS_MAX_SIZE:-100M}"

kms:

image: \${KMS_IMAGE:-kurento/kurento-media-server:7.0.1}

restart: always

network_mode: host

ulimits:

core: -1

volumes:

- /opt/openvidu/kms-crashes:/opt/openvidu/kms-crashes
- \${OPENVIDU_RECORDING_PATH}:\${OPENVIDU_RECORDING_PATH}
- /opt/openvidu/kurento-logs:/opt/openvidu/kurento-logs

environment:

- KMS_MIN_PORT=40000
- KMS_MAX_PORT=57000
- GST_DEBUG=\${KMS_DOCKER_ENV_GST_DEBUG:-}
- KURENTO_LOG_FILE_SIZE=\${KMS_DOCKER_ENV_KURENTO_LOG_FILE_SIZE:-100}
- KURENTO_LOGS_PATH=/opt/openvidu/kurento-logs

logging:

options:

max-size: "\${DOCKER_LOGS_MAX_SIZE:-100M}"

coturn:

image: openvidu/openvidu-coturn:2.28.0

restart: on-failure

ports:

- "\${COTURN_PORT:-3478}:\${COTURN_PORT:-3478}/tcp"
- "\${COTURN_PORT:-3478}:\${COTURN_PORT:-3478}/udp"

env_file:

- .env

volumes:

- ./coturn:/run/secrets/coturn

command:

- --log-file=stdout
- --listening-port=\${COTURN_PORT:-3478}
- --fingerprint
- --min-port=\${COTURN_MIN_PORT:-57001}
- --max-port=\${COTURN_MAX_PORT:-65535}
- --realm=openvidu
- --verbose
- --use-auth-secret
- --static-auth-secret=\${COTURN_SHARED_SECRET_KEY}

logging:

options:

max-size: "\${DOCKER_LOGS_MAX_SIZE:-100M}"

nginx:

image: openvidu/openvidu-proxy:2.28.0

restart: always

network_mode: host

volumes:

- ./certificates:/etc/letsencrypt
- ./owncert:/owncert
- ./custom-nginx-vhosts:/etc/nginx/vhost.d/
- ./custom-nginx-locations:/custom-nginx-locations
- \${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:/opt/openvidu/custom-layout

environment:

- DOMAIN_OR_PUBLIC_IP=\${DOMAIN_OR_PUBLIC_IP}
- CERTIFICATE_TYPE=\${CERTIFICATE_TYPE}
- LETSENCRYPT_EMAIL=\${LETSENCRYPT_EMAIL}
- PROXY_HTTP_PORT=\${HTTP_PORT:-}
- PROXY_HTTPS_PORT=\${HTTPS_PORT:-}
- PROXY_HTTPS_PROTOCOLS=\${HTTPS_PROTOCOLS:-}
- PROXY_HTTPS_CIPHERS=\${HTTPS_CIPHERS:-}
- PROXY_HTTPS_HSTS=\${HTTPS_HSTS:-}
-

ALLOWED_ACCESS_TO_DASHBOARD=\${ALLOWED_ACCESS_TO_DASHBOARD:-}

- ALLOWED_ACCESS_TO_RESTAPI=\${ALLOWED_ACCESS_TO_RESTAPI:-}
- PROXY_MODE=CE

- WITH_APP=true
- SUPPORT_DEPRECATED_API=\${SUPPORT_DEPRECATED_API:-false}
- REDIRECT_WWW=\${REDIRECT_WWW:-false}
- WORKER_CONNECTIONS=\${WORKER_CONNECTIONS:-10240}
- PUBLIC_IP=\${PROXY_PUBLIC_IP:-auto-ipv4}

logging:

options:

max-size: "\${DOCKER_LOGS_MAX_SIZE:-100M}"

redis_container:

image: redis:latest

container_name: redis_prj

ports:

- 6379:6379

set storage mount volume

volumes:

- ./redis/data:/data
- ./redis/conf/redis.conf:/usr/local/conf/redis.conf

add meta data using docker label

labels:

- "name=redis"
- "mode=standalone"

set restart when container off

restart: always

command: redis-server /usr/local/conf/redis.conf

d. Nginx

i. project.conf

```
# user nginx;
```

```
server {
```

```
    #root /home/ubuntu/build;
```

```
    #index index.html index.htm;
```

```
    autoindex_localtime on;
```

```
    proxy_set_header Host $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-Forwarded-Proto https;
```

```
    proxy_headers_hash_bucket_size 512;
```

```
    proxy_redirect off;
```

```
    # Websockets
```

```
    proxy_http_version 1.1;
```



```
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

```
location / {
    root /home/ubuntu/build;
    index index.html index.htm;
    try_files $uri $uri/ /index.html;
#    proxy_pass http://localhost:3000/;
    proxy_connect_timeout 300s;
    proxy_read_timeout 600s;
    proxy_send_timeout 600s;
    proxy_buffer_size      128k;
    proxy_buffers           4 256k;
    proxy_busy_buffers_size 256k;
}
```

```
location /api {
    proxy_pass http://localhost:8081;
}
```

```
location /openvidu/api {
    proxy_pass http://서비스도메인:8443;
}
```

```

location ~ /openvidu$ {
    proxy_pass http://서비스도메인:8443;
}

location /ws {
    proxy_pass http://localhost:8081;
}

listen 443 ssl;
ssl_certificate /etc/letsencrypt/live/서비스도메인/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/서비스도메인/privkey.pem;
}

server {
    # 서비스도메인

    listen 80;

    server_name 서비스도메인;

    return 301 https://$host$request_uri;

}

```

e. Openvidu

i. .env

DOMAIN_OR_PUBLIC_IP=서비스도메인

OPENVIDU_SECRET= 시크릿

CERTIFICATE_TYPE=letsencrypt

LETSencrypt_EMAIL=인증서 발급 이메일

HTTP_PORT=81

HTTPS_PORT=8443

f. Jenkins Pipeline Script

```
pipeline {  
    agent any  
    stages {  
        stage('git_clone') {  
            steps {  
                git branch: '{branch}', credentialsId: 'signiel', url: '{gitlab}'  
            }  
        }  
    }  
}
```

```

    }
}
stage('BE-Build'){
    steps{
        dir("./backend"){
            sh'''
                chmod +x ./gradlew
                ./gradlew clean bootJar
            '''
        }
    }
}
stage('FE-build') {
    steps {
        dir("./frontend") {
            nodejs('18.16.0') {
                sh 'npm install'
                sh 'CI=false npm run build'
            }
        }
    }
}
stage('Compression') {
    steps {

```

```

        dir("./frontend") {
            sh"""
                rm -rf node_modules

                tar -cvf build.tar build
            """
        }
    }

    stage('Deploy'){
        steps{
            sshagent(credentials:['ec2-server']){
                sh"""
                    ssh -o StrictHostKeyChecking=no ubuntu@서비스도메인
uptime
                    scp      /var/jenkins_home/workspace/Dasoni/frontend/build.tar
ubuntu@서비스도메인:/home/ubuntu

                    scp
/var/jenkins_home/workspace/Dasoni/backend/build/libs/heartsigniel-0.0.1-SNAPSHOT.jar
ubuntu@서비스도메인:/home/ubuntu/Dasoni

                    ssh -t ubuntu@서비스도메인 ./deploy.sh
                """
            }
        }
    }
}

```

}

4. Openvidu 배포 및 ProxyServer를 위한 포트 개방

```
sudo apt-get install ufw
```

```
sudo ufw allow ssh
```

```
sudo ufw allow http
```

```
sudo ufw allow https
```

```
sudo ufw allow 8080
```

```
sudo ufw allow 3000
```

```
sudo ufw allow 8081
```

```
sudo ufw allow 80
```

```
sudo ufw allow 81
```

```
sudo ufw allow 80/tcp
```

```
sudo ufw allow 22/tcp
```

```
sudo ufw allow 443/tcp
```

```
sudo ufw allow 3478/tcp
```

```
sudo ufw allow 3478/udp
```

```
sudo ufw allow 8082/tcp
```

```
sudo ufw allow 8443/tcp
```

```
sudo ufw allow 8080/tcp
```

```
sudo ufw allow 6379
```

- a. Openvidu 컨테이너 실행

```
# /opt/openvidu 에서
```

```
sudo ./openvidu start
```

5. SSL 인증서 발급

```
sudo apt-get install letsencrypt
```

```
sudo apt-get install certbot python3-certbot-nginx
```

```
sudo certbot --nginx
```

```
# 이메일 입력
```

```
# 약관 동의 - Y
```

```
# 이메일 발송 동의 - Y or N
```

```
# 도메인 입력
```

6. Jenkins에 Gitlab Webhook 연결

- a. Jenkins 대시보드>Jenkins 관리>Plugins
- b. Gitlab 플러그인 설치
- c. Gitlab API Token 발급

User Settings

- Profile
- Account
- Applications
- Chat
- Access Tokens**
- Emails
- Password
- Notifications
- SSH Keys
- GPG Keys
- Preferences
- Comment Templates
- Active Sessions
- Authentication Log

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Token name

gitlab-jenkins-token

Expiration date

2023-09-17

Select scopes

Scopes set the permission levels granted to the token. [Learn more.](#)

- ☐ **api**
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read_api**
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read_user**
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **read_repository**
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ **write_repository**
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

[Create personal access token](#)

d. 발급 받은 Token을 Jenkins에 등록한다

Domain

Global credentials (unrestricted)

Kind

GitLab API token

Scope

Global (Jenkins, nodes, items, all child items, etc)

API token

ID

Description

[Add](#) [Cancel](#)

e. Jenkins관리>시스템 설정으로 이동 후, Gitlab 경로와 위에서 등록한 API Token Credential을 사용하여 Gitlab과 Jenkins를 연동한다.

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

gitlab

GitLab host URL ?

The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com/

Credentials ?

API Token for accessing GitLab

GitLab API token

Add ▾

고급 ▾

저장

Apply

f. Pipeline 프로젝트 생성 및 Webhook 경로 확인

A. CI/CD를 수행할 Pipeline을 생성한다

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://lab.ssafy.com/project/Dasoni ?

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 ?

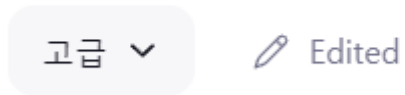
Never

☒ Approved Merge Requests (EE-only) ?

☒ Comments ?

g. Pipeline Secret Token 발급

A. Build Triggers 메뉴 아래 '고급' 탭 클릭



B. Generate를 통해 Secret Token 발급 (꼭 따로 저장해야함)

Secret token ?

Generate

h. Repository Webhook Event 설정

A. Gitlab의 Repository에서 Settings>Webhook

B. Webhook Event는 Repository 별로 저장해주어야함

C. 앞서 기억해둔 URL과 Secret을 입력해주고 원하는 Trigger 체크

URL

http://example.com/trigger-ci.json

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

☒ Show full URL
☐ Mask portions of URL
Do not show sensitive data such as tokens in the UI.

Secret token

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

i. 등록 확인

A. 등록 후, Test를 통해 연동 확인 테스트를 할 수 있다

☐ Wiki page events
A wiki page is created or updated.

☐ Deployment events
A deployment starts, finishes, fails, or is canceled.

☐ Feature flag events
A feature flag is turned on or off.

☐ Releases events
A release is created or updated.

SSL verification
☒ Enable SSL verification

Project Hooks (1)

Push events

SSL Verification: enabled

Push events

Tag push events

Issues events

Confidential issues events

Comments

Confidential comments

Merge request events

Job events

Pipeline events

Wiki page events

Deployment events

Feature flag events

Releases events

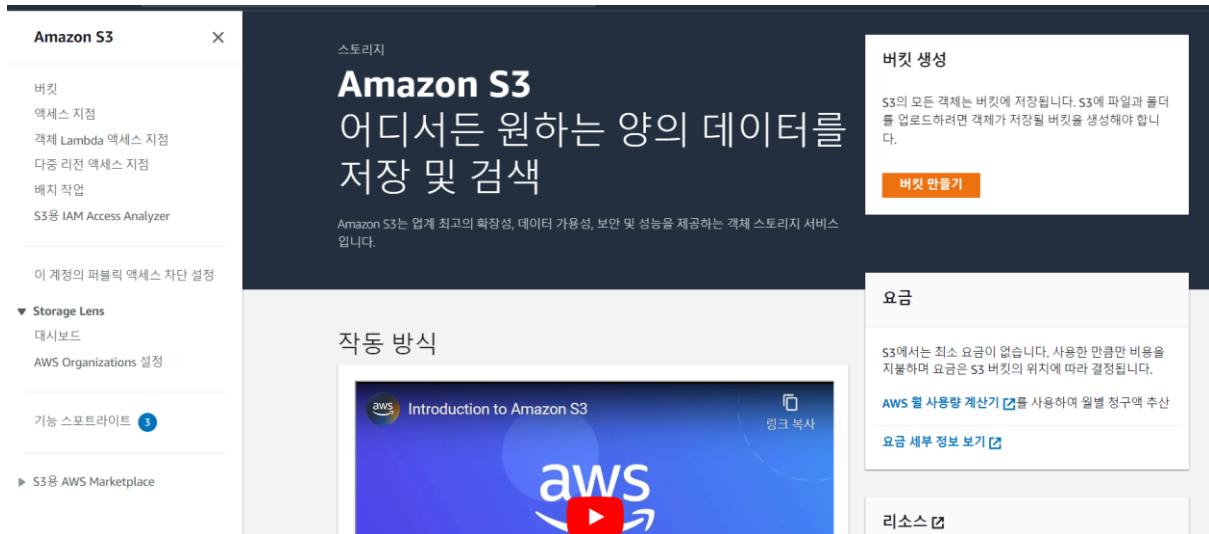
B. Test를 하고 나서 이와 같이 표시되면 연동 성공



II. 외부 서비스

1. Amazon S3 Cloud Storage

- <https://aws.amazon.com/ko/> 로 접속
- 로그인 후, 서비스 창에서 S3 선택
- 버킷 만들기 클릭



d. 버킷 이름과 리전 설정

i. 버킷 이름은 고유한 값이어야 함

버킷 만들기 [정보](#)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

일반 구성

버킷 이름

버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름 지정 규칙 보기](#)

AWS 리전

기존 버킷에서 설정 복사 - [선택 사항](#)
다음 구성의 버킷 설정만 복사됩니다.

[버킷 선택](#)

e. 퍼블릭 액세스 설정 후 버킷 만들기 클릭

i. 외부에 S3를 공개할 경우 모든 퍼블릭 액세스 차단을 체크 해제, 공개하지 않는다면 체크

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

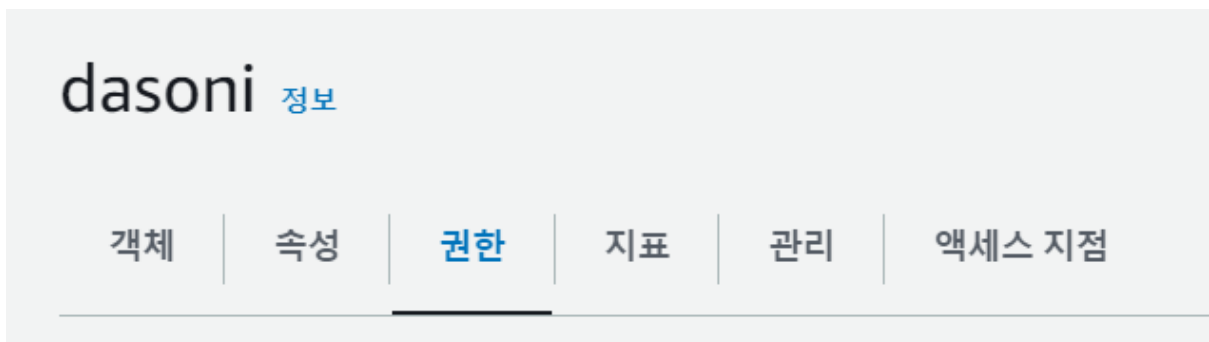
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☒ 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.

f. 버킷 정책 생성

- i. 생성된 버킷을 클릭 후, 권한 메뉴를 클릭한 다음 버킷 정책 메뉴의 편집을 클릭



- ii. 버킷 ARN을 복사한 뒤, AWS 정책 생성기로 접속 (<http://awspolicygen.s3.amazonaws.com/policygen.html>) 하여 정책 타입과 상태를 추가한 뒤, 정책 생성

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 VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy IAM 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

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:::dasoni

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

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

Add Statement

Policy JSON Document

Click below to edit. To save the policy, copy the text below to a text editor. Changes made below will **not be reflected in the policy generator tool**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1692284187941",
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::dasoni"
    }
  ]
}
```

- g. 생성된 정책을 복사 후, 버킷의 정책란에 붙여넣기를 하고 변경 사항을 저장