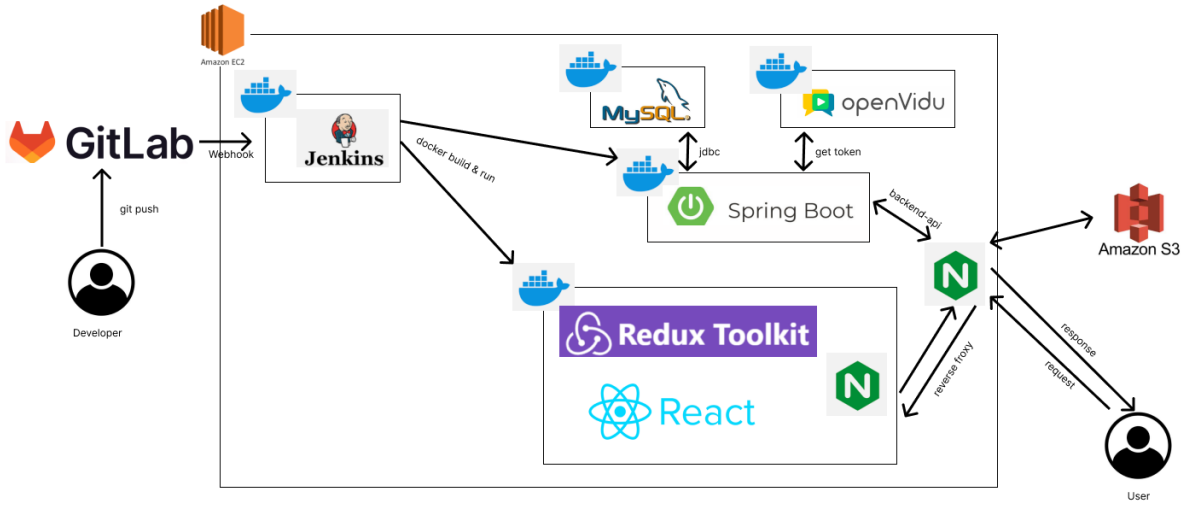
CICD 환경



전체 로직

CI

깃랩: 개발자가 깃랩에 개발내용을 푸쉬하게 되면, 깃랩의 웹훅(webhook)이 젠킨스에게 전송된다.

젠킨스 : 깃랩의 웹훅을 전송받은 젠킨스는 개발자가 미리 짜둔 파이프라인을 따라서 명령을 실행한다.

도커 : 젠킨스에서 클론받아온 프론트 및 , 백엔드 코드를 도커파일을 기반으로 빌드하여 도커이미지를 만든다.

CD

엔진엑스 , 도커 ,젠킨스

젠킨스 : 빌드했던 도커이미지를 런하여 컨테이너 생성

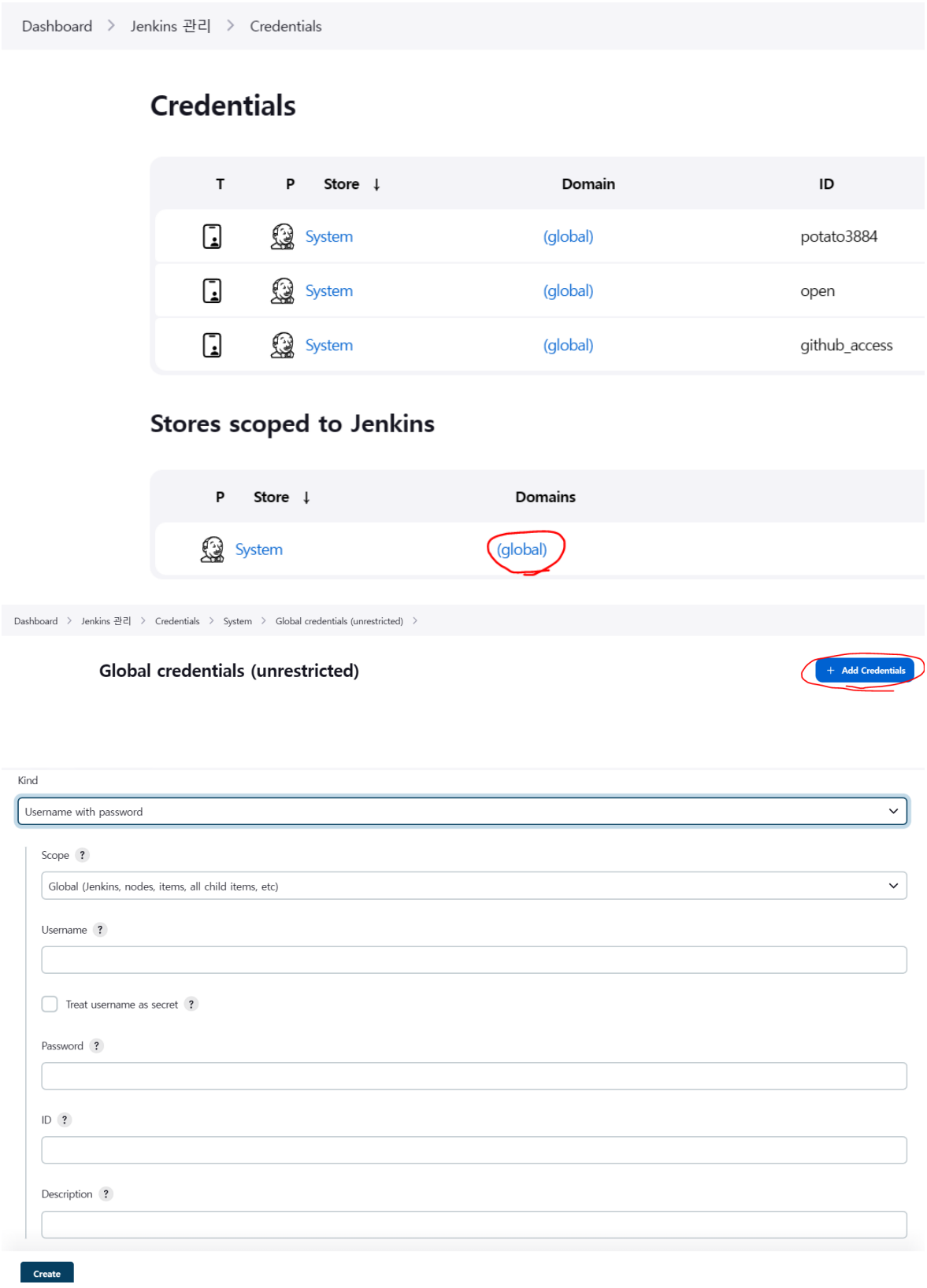
도커 : 백프로젝트의 .jar 파일 실행 및 프론트프로젝트에선 컨테이너 내부에 엔진엑스를 활용하여 실행시킨다.

엔진엑스 : 외부에서 들어오는 요청들을 ssl인증을 거친경로로 리버스프록시해준다. 또한 들어온 경로에 따라서 백 혹은 프론트 포트로 분기시킨다.

1. EC2서버에 도커 설치
2. EC2 서버에서 최신 젠킨스 이미지로 젠킨스컨테이너 생성(docker run)
3. 젠킨스에서 credential 설정 및 여러 플러그인 설정

   웹훅을 이용해 깃랩 저장소를 클론해오기 위해선 크레덴셜 설정이 필요하다.

그러나 깃랩의 엑세스 토큰은 어째선지 인식을 하지못해 Username 과 passowrd를 이용한 크레덴셜을 이용하였다.

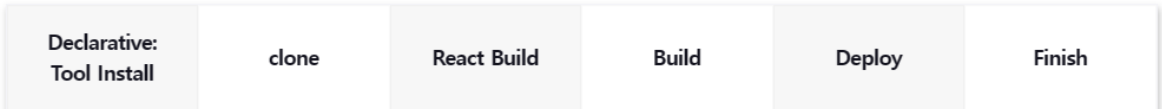


Username 에 깃랩에서 저장한 username , password에는 깃랩의 password를 입력한다.

ID는 파이프라인에서 이 크렌덴셜을 가져올때 사용할 이름을 정해준다.

4. 젠킨스에서 새로운 아이템으로 파이프라인 설정

해당 파이프라인이 한번 빌드될때 해야될 명령들의 순서는 크게



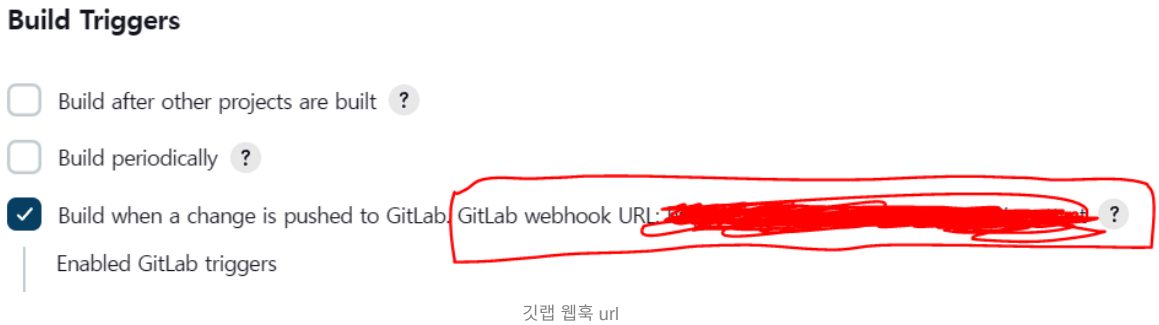
- nodejs 환경을 설치한다.
- clone : 깃랩에서 소스코드를 받아온다.
- React Build : npm 환경을 준비하고 , 코드들을 빌드한다.
- build : 빌드된 파일로 도커 이미지를 생성
- deploy : 만든 도커이미지를 서버위에 컨테이너 생성
- finish : 전에 만들었던 이미지를 제거한다.

5. 깃랩과 젠킨스의 웹훅 설정

먼저 깃랩에선 , 멤버로서 owner 권한이 필요하다.

settings → webhook 설정으로 웹훅을 추가한다.

젠킨스에선 해당 아이템으로 들어가서,



깃랩 웹훅 url

Build Triggers 밑의 고급(advanced) 를 클릭하고 더 내려가면

Secret token

을 깃랩에 추가한다.

6. back의 포트설정을 위한 application.proeprties 새로 생성

   서버의 8080포트는 이미 젠킨스 컨테이너가 사용하고 있으므로 ,

   back서버가 8081 포트를 사용할 수 있도록 application.properties 파일을 커스텀해서 젠킨스에 추가하고, 파이프라인에서 이 파일을 깃랩에서 클론받아온 파일과 교체한다.

```
server.port=8081
server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://프로젝트도메인주소:3306/yogurt?serverTimezone=UTC&characterEncoding=UTF-8&useSSL=false
spring.datasource.username=Mysql사용할유저네임
spring.datasource.password=Mysql사용할비밀번호

spring.mvc.pathmatch.matching-strategy=ant_path_matcher

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

logging.level.org.springframework=debug
logging.level.org.springframework.web=debug


cloud.aws.credentials.accessKey =S3클라우드엑세스키
cloud.aws.credentials.secretKey = S3클라우드시크릿키
cloud.aws.s3.bucket = yogurt-bucket
cloud.aws.region.static = ap-northeast-2
cloud.aws.stack.auto= false
```

7. mysql 컨테이너 실행 및 백업설정
   1. mysql-container 도커 이미지 실행
   2. root 설정 및 DB 생성 , 유저생성 및 권한 설정

```
mysql -u root -p
비밀번호 : //비밀번호 입력
create database yogurt;
CREATE USER 'ssafy'@'%' IDENTIFIED BY 'q1w2e3r4t5';
grant all privileges on yogurt.% to 'ssafy'@'%' identified by 'q1w2e3r4t5';
flush privileges;
```

   3. 백업 설정

크론탭(crontab)을 이용해 매시각 마다 서버에서 mysql 컨테이너에 덤프파일을 생성하는 명령파일(mysql_dumb.sh)을 실행한다. 그러면 컨테이너에 덤프된 파일을 서버의 backup 폴더 안에 복사한다.

```
10,20,30,40,50 * * * * docker exec DB컨테이너이름 sh 컨테이너의sh파일경로/mysql_dump.sh
8,11,21,31,41,50 * * * * docker cp DB컨테이너이름:sh 컨테이너의sh파일경로 EC2서버백업경로
```

```
cd /usr/local/mysql/
mysqldump -u root -p1234 yogurt > /usr/local/mysql/yogurt-$(date +%Y_%m_%d).sql
```

8. zero ssl 설정

   https://zerossl.com/

   위의 링크를 들어가 회원가입 후 무료 ssl인증서를 받는다.

   ca_bundle.crt

   certificate.crt

   private.key

   파일들을 알아볼수 있는 경로에다가 저장한다.

9. 엔진엑스 설정
   1. nginx 설치
   2. default 파일 설정

```
server{
        if ($host =  도메인주소){
                return 308   https://$host$request_uri;
                // 301 로 하면 제대로 인식을 못하는 에러 발생
        }
                listen 80 ;
                server_name 도메인주소;
        return 404;
}


server {
        listen 443 ssl;
        server_name 도메인 주소;


        ssl_certificate         /etc/zerossl/certificate.crt;
        ssl_certificate_key     /etc/zerossl/private.key;

        location / {
                proxy_pass http://localhost:3000;
        }
        location /be-api/ {
                proxy_pass http://localhost:8081/;
        }
        location /api/ {
                proxy_pass http://localhost:5000/;
        }
}
```

   3. nginx.conf 설정

```
user  root;#www-data
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
```

```
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

4. 오픈비두 .env 파일

```
# 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=i8b204.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=MY_SECRET

# 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=owncert

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=user@example.com

# 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=8444

# Changes the port of all services exposed by OpenVidu.
```

```
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=8445

# 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=false

# 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,webrtcConnectio

# 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:6.18.0

# 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
```

10. 젠킨스 pipeline 설정

```
pipeline {
    agent any

    stages {
```

```
        stage('clone') {
            steps {
                git branch: 'back', credentialsId: 'potato3884', url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B204.git'
            }
        }
        stage('Change application.yml') {
            steps {
                sh '''
                    rm ./backend/src/main/resources/application.properties
                    cp /home/env/application.properties /var/jenkins_home/workspace/test_CICD/backend/src/main/resources
                '''
            }
        }
        stage('build') {
            steps {
                dir('backend'){
                    sh('chmod 755 gradlew')
                    sh(' ./gradlew clean build -x test')
                }
            }
        }
        stage('stop & rm before docker container'){



            steps{

                script
                {
                    try
                    {
                        sh('docker rm -f deploy')
                        sh('docker rmi test-img:0.0')
                    } catch (e)
                    {
                        catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE')
                        {
                            sh "exit 0"
                        }
                        echo 'rm TEST Fail!! But Im Running!'
                    }
                }

            }
        }
        stage('docker build') {
            steps {
                dir('backend'){
                    sh('ls -al')
                    sh('docker build -t test-img:0.0 .')
                    sh('docker images')
                }
            }
        }
        stage('deploy') {
            steps {
                sh('docker run -d -p 8081:8081 --name deploy --network back-net test-img:0.0')
                sh('docker logs deploy')

            }
        }
    }
}
```

```
pipeline {
    agent any

    tools {
        nodejs "nodejs-18.13.0"
    }

    stages {
        stage('clone') {
            steps {
                git branch: 'front', credentialsId: 'potato3884', url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B204.git'
            }
        }


        stage('React Build') {
```

```
            steps {
                dir ('frontend')
                {
                    sh 'npm install react@^17.0.2 react-dom@17.0.2 --legacy-peer-deps'

                }
                dir('frontend'){
                    sh 'npm run build'
                }
            }
        }

        stage('Build') {
            steps {
                dir('frontend'){
                    sh 'docker build -t basepage/nginx ./'
                }
            }
        }

        stage('Deploy') {

            steps{
                sh '''
                docker stop nginx
                '''
            }
            post{
                success{
                  sh 'docker rm nginx'
                  sh 'docker run -d --name nginx -p 3000:3000 -v /etc/letsencrypt/archive:/etc/letsencrypt/archive -u root basepage/ng
                }
                failure{
                    sh 'docker run -d --name nginx -p 3000:3000 -v /etc/letsencrypt/archive:/etc/letsencrypt/archive -u root basepage/
                }
            }


        }

      stage('Finish') {
            steps{
                  dir ('frontend'){
                    sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
                }
            }
        }
    }
}
```

```
pipeline {
    agent any

    stages {
        stage('clone') {
            steps {
                git branch: 'master', credentialsId: 'github_connection', url: 'https://github.com/cwelly/openvidu-tutorials.git'
            }
        }
        stage('Change application.yml') {
            steps {
                sh '''
                    rm ./openvidu-basic-java/src/main/resources/application.properties
                    cp /home/env/openvidu_doc/application.properties /var/jenkins_home/workspace/open_vidu_java/openvidu-basic-java/sr
                    cp /home/env/openvidu_doc/Dockerfile /var/jenkins_home/workspace/open_vidu_java/openvidu-basic-java
                '''
            }
        }
        stage('build') {
            steps {
                dir('openvidu-basic-java'){
                    // sh('chmod 755 gradlew')
                    sh(' mvn clean  package')
                }
            }
        }
        stage('stop & rm before docker container'){
            steps{
                script
                {
```

```
                    try
                    {
                        sh('docker rm -f open-deploy')
                        sh('docker rmi test-openvidu-img:0.0')
                    } catch (e)
                    {
                        catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE')
                        {
                            sh "exit 0"
                        }
                        echo 'rm TEST Fail!! But Im Running!'
                    }
                }

            }
        }
        stage('docker build') {
            steps {
                dir('openvidu-basic-java'){
                    sh('ls -al')
                    sh('docker build -t  test-openvidu-img:0.0 .')
                    sh('docker images')
                }
            }
        }
        stage('deploy') {
            steps {
                sh('docker run -d -p 5000:5000 --name open-deploy  test-openvidu-img:0.0')
                sh('docker logs open-deploy')

            }
        }
    }
}
```