

서버

1. 서버 접속

1. Git bash 접속

Pem이 있는 폴더에서 git bash 를 열고 다음 명령어 수행

```
ssh -i I7A503T.pem ubuntu@i7A503.p.ssafy.io
```

접속이 되었다면 공통 아이피를 찾아내는 명령어 수행

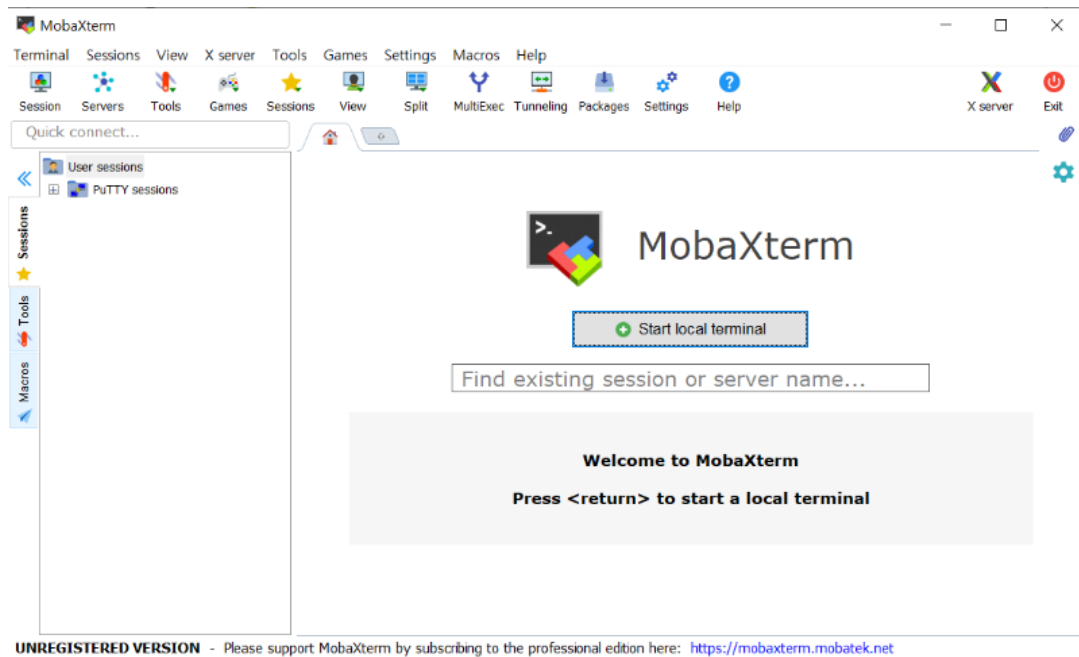
```
Sudo curl ifconfig.me
```

2. MobaXterm 접속

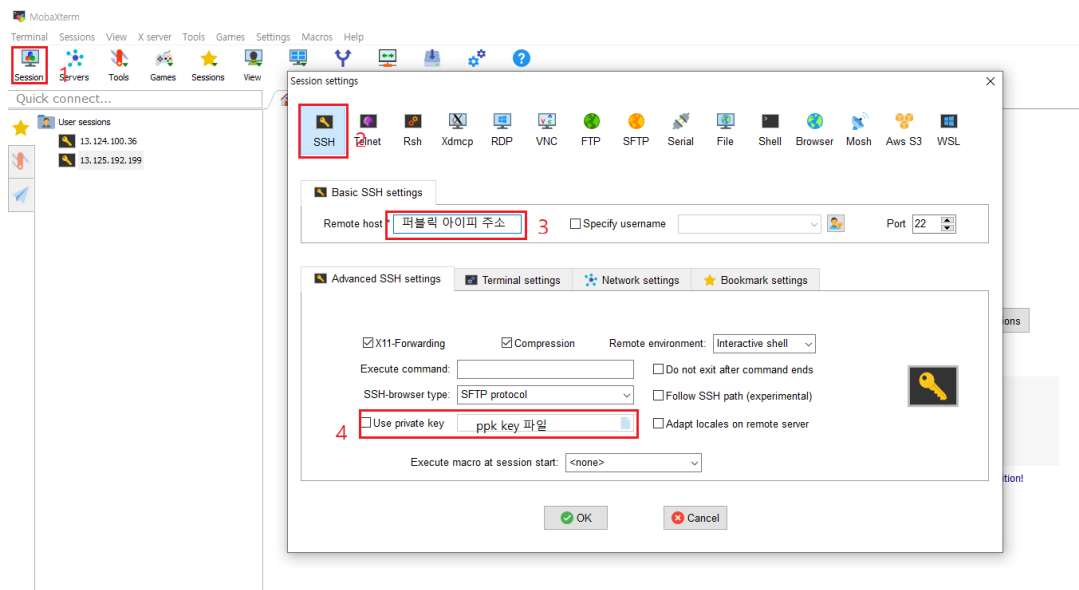
[MobaXterm 설치](#)

The screenshot shows the MobaXterm website's download page. At the top, there is a navigation bar with links: Home, Demo, Features, Download (highlighted), Plugins, Help, and Contact. There are also social media icons and buttons for 'Customer area' and 'Buy'. The main content area is titled 'MobaXterm Home Edition'. It instructs users to 'Download MobaXterm Home Edition (current version):' and provides two options: 'MobaXterm Home Edition v21.2 (Portable edition)' in a blue button and 'MobaXterm Home Edition v21.2 (Installer edition)' in a green button, which is highlighted with a red box and a red circle with the number '1'. Below this, it offers 'Download previous stable version' with links for 'MobaXterm Portable v21.1' and 'MobaXterm Installer v21.1'. It also mentions 'You can also get early access to the latest features and improvements by downloading MobaXterm Preview version:' with a corresponding orange button. A disclaimer states that by downloading the software, users accept the 'MobaXterm terms and conditions'. At the bottom, it provides a link to download sources and a note about the 'MobaXterm Professional Edition' subscription, which includes professional support and a 'Customizer' software.

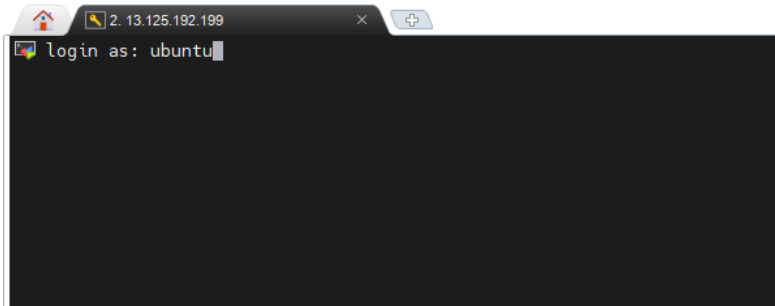
설치는 next만 누르면 됩니다. 설치 후 실행 화면은 아래와 같습니다.



Aws 서버 연결 방법은 다음의 순서대로 하면 됩니다.



서버 로그인은 해당 aws의 운영체제를 입력하면 된다.



2. 기본 셋팅

1. 타임존 바꾸기

```
sudo rm /etc/localtime  
  
sudo ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime  
  
date
```

2. vi설치([명령어모음](#))

```
sudo apt-get install vim
```

3. 자바 설치

1. 설치 전 최신 버전 업데이트

```
sudo apt-get update  
  
sudo apt-get upgrade
```

2. Java 설치 (원하는 버전 설치) 및 확인 (\$ 는 명령어 줄을 뜻한다. 복사x)

```
$ sudo apt-get install openjdk-8-jdk  
$ java -version  
openjdk version "11.0.11" 2021-04-20  
$ javac -version  
javac 11.0.11
```

3. 환경 변수 설정

```
vim /etc/profile
```

위의 명령어로 파일을 열고 맨 아래에 다음 구문을 추가 한다

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64    // 본인의 자바 설치 경로  
export PATH=$JAVA_HOME/bin:$PATH  
export CLASSPATH=$CLASSPATH:$JAVA_HOME/jre/lib/ext:$JAVA_HOME/lib/tools.jar
```

이후 설정한 환경 변수를 적용하기 위해 다음 명령어 실행

```
source /etc/profile
```

환경 변수 확인

```
$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64
```

4. Docker 설치

1. 패키지 최신 버전 업데이트

```
$ sudo apt-get update && upgrade
```

2. Apt가 https를 통해 repository를 이용하는 것을 허용할 수 있도록 패키지 설치

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

3. Docker 공식 GPG key 추가

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

4. Docker Repository 등록

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

5. Docker 설치 후 버전 확인

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

6. Ubuntu 에서 sudo 없이 도커 명령어 사용하기 설정 및 재로그인

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

5. Jenkins 설치 및 서버 연결(docker-compose)

1. Docker-compose 설치 및 확인

```
$ sudo curl -L https://github.com/docker/compose/releases/download/1.26.2/docker-  
r-compose-\$\(uname -s\)-\$\(uname -m\) -o /usr/local/bin/docker-  
compose  
$ sudo chmod +x /usr/local/bin/docker-compose  
$ docker-compose --version
```

2. 젠킨스 설치

```
$ mkdir compose && cd compose  
  
$ mkdir jenkins-dockerfile && cd jenkins-dockerfile  
  
$ vim Dockerfile  
  
...  
  
FROM jenkins/jenkins:its  
  
USER root  
  
RUN apt-get update &&W  
  
    apt-get upgrade -y &&W  
  
    apt-get install -y openssh-client  
  
...
```

```
$ cd ..

$ vim docker-compose.yml

...

version: "3"

services:

  jenkins:

    container_name: jenkins-compose

    build:

      context: jenkins-dockerfile

      dockerfile: Dockerfile

    user: root

    ports:

      - 8080:8080

      - 50000:50000

    volumes:

      - /home/ubuntu/compose/jenkins:/var/jenkins_home

      - /home/ubuntu/compose/.ssh:/root/.ssh

...

$ mkdir jenkins

$ mkdir .ssh
```

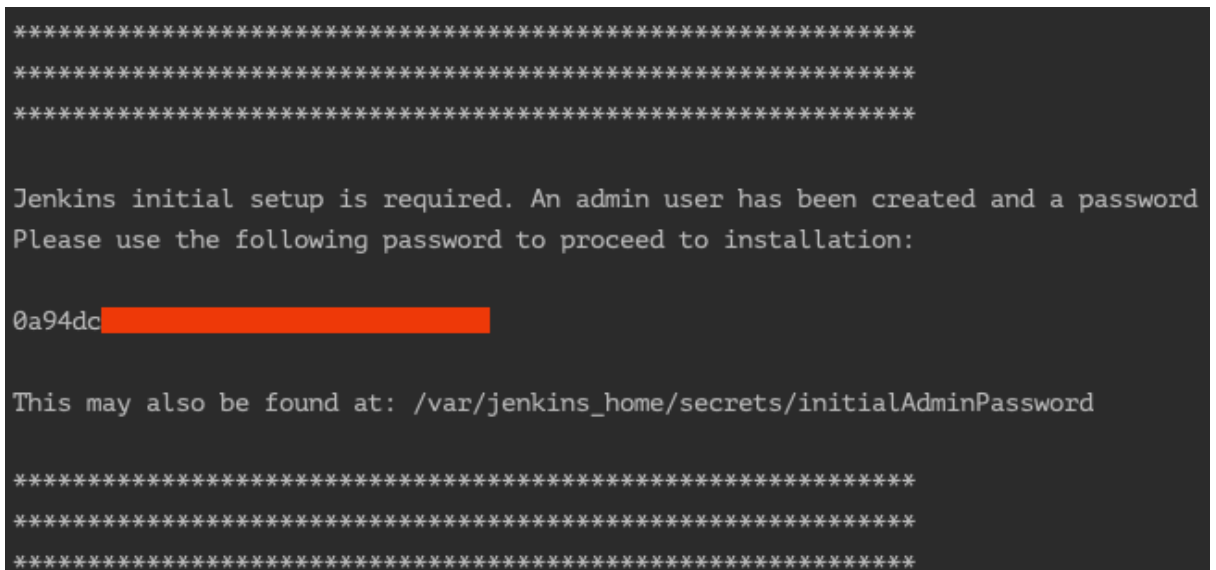
3. 젠킨스 접속

http://[ec2-주소-입력]:8000

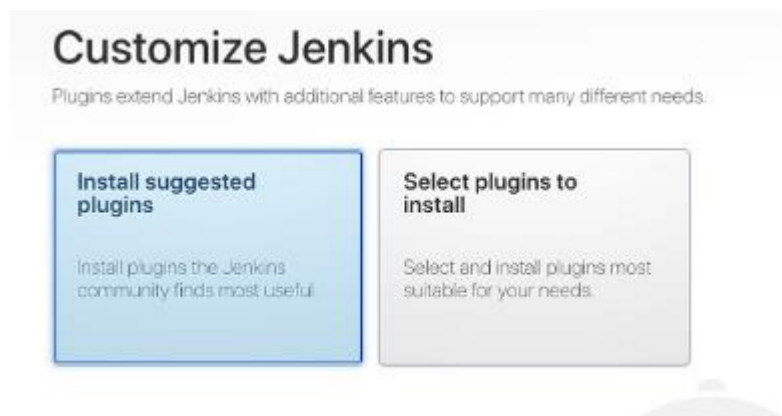


비밀번호는 다음 명령어

```
$ docker logs jenkins-compose
```



플러그인 설치



계정 생성

Getting Started

Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 2.235.2

[Skip and continue as admin](#) [Save and Continue](#)

GIT과 DOCKER 관련된 플러그인 모두 설치 후 젠킨스 컨테이너 재시작

Jenkins 관리

System Configuration

사istem 설정
환경변수 및 경로 정보등을 설정합니다.

Global Tool Configuration
Configure tools, their locations and automatic installers.

플러그인 관리
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 비사용으로 설정할 수 있습니다.

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

Lockable Resources

New View

노드 관리
Add, remove, control and monitor the various nodes that

4. 아이템 등록

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

Lockable Resources

New View

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.



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, 플러그인 특성 빌드, 기타 등등 처럼 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.



GitHub Organization

Scans a GitHub organization (or user account) for all repositories matching some defined markers.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Git 주소를 repository url에 입력 후 add를 통해 git 사용자 아이디 비밀번호 입력

The image displays two screenshots from the Jenkins web interface. The top screenshot shows the 'General' tab of a job configuration, where 'Git' is selected under '소스 코드 관리' (Source Code Management). The 'Repository URL' is set to 'https://github.com/ddaaac/hello-jenkins.git', and the 'Credentials' dropdown is set to 'Add'. The 'Branches to build' section shows a 'Branch Specifier' of '*/master'. The bottom screenshot shows the 'Add Credentials' dialog, where 'Global credentials (unrestricted)' is selected for the domain, 'Username with password' for the kind, and 'Global (Jenkins, nodes, items, all child items, etc)' for the scope. The 'Username' field contains 'ddaaac', and the 'Password' field is masked with dots. Both the 'Add' button and the 'Username' field are highlighted with red boxes.

General 소스 코드 관리 빌드 유발 빌드 환경 Build 빌드 후 조치

☐ 이 빌드는 매개변수가 있습니다
☐ 빌드 안함
☐ 필요한 경우 concurrent 빌드 실행

고급...

소스 코드 관리

☐ None
☒ Git

Repositories

Repository URL

Credentials

고급...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

Repository browser (자동)

Additional Behaviours

☐ Subversion

빌드 유발

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Kind

Scope

Username

Password

ID

Description

Git webhook 설정

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://10.216.253.160:7009/project/webhook_test`

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

Comment (regex) for triggering a build

Jenkins please retry a build

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

고급...

Generate 버튼을 누르고 표시되는 토큰 값도 별도로 복사하고 저장한다

General | 소스 코드 관리 | **빌드 유발** | 빌드 환경 | Build | 빌드 후 조치

Comment (regex) for triggering a build

Jenkins please retry a build

- ☒ Enable [cd-skip]
- ☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

- ☒ Set build description to build cause (eg. Merge request or Git Push)
- ☐ Build on successful pipeline events

Pending build name for pipeline

- ☐ Cancel pending merge request builds on update

Allowed branches

- ☐ Allow all branches to trigger this job
- ☒ Filter branches by name

Include

master

Following patterns don't match any branch in source repository: master

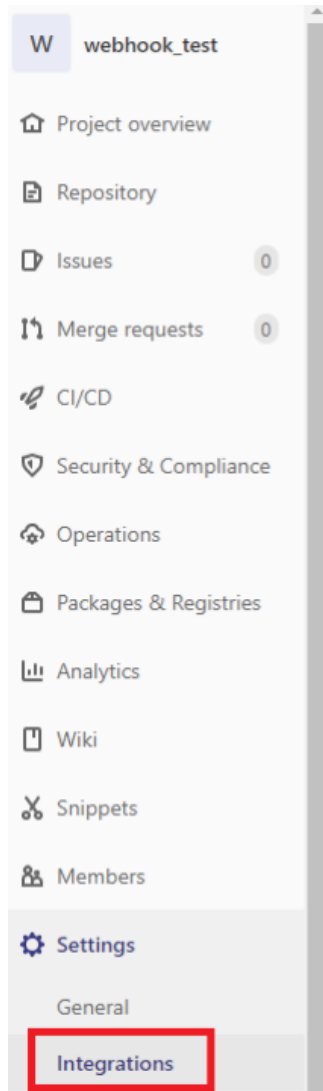
Exclude

- ☐ Filter branches by regex
- ☐ Filter merge request by label

Secret token

Generate

Gitlab 프로젝트에 들어간다



Junho Lee > webhook_test > Integration

Search settings

Webhooks have moved. The

Go to Webhooks

Integrations

Integrations enable you to make this webhooks.

Active integrations

Integration

Add an integration

Integration

Asana

Assembla

Atlassian Bamboo

Bugzilla

복사한 토큰을 넣어주고 어떤 브랜치에 푸시 됐을 때 웹훅을 보낼지 설정 한다.

Search settings

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://[redacted]/project/webhook_test

URL must be percent-encoded if necessary.

Secret token

[redacted]da63f9b0ad

Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☒ Push events

master

URL is triggered by a push to the repository

빌드 시 수행할 명령어 작성 후 아이템 최종 저장

Build

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
cd BackEnd
chmod +x ./gradlew
./gradlew clean build
cd ..
cd BackEnd-Service
chmod +x ./gradlew
./gradlew clean build
cd ..
cd FrontEnd
npm install
npm run build
ssh -t -t ubuntu@ip-172-26-15-203 <<EOF
  cd /home/ubuntu/compose
  docker-compose up --build -d
EOF
```

고급...

Add build step ▾

cd BackEnd

chmod +x ./gradlew

./gradlew clean build

cd ..

cd BackEnd-Service

chmod +x ./gradlew

./gradlew clean build

cd ..

cd FrontEnd

npm install

npm run build

스프링 서버와 nginx 설치 이후에 아래 명령어 추가할 내용이므로 좀 있다 다시 온다.

```
ssh -t -t ubuntu@ip-172-26-15-203 <<EOF

    cd /home/ubuntu/compose

    docker-compose up --build -d

    docker-compose build --no-cache back-service

    docker-compose build --no-cache back-auth

    docker restart hello-nginx

    exit

EOF
```

5. 스프링 부트 컨테이너(서버) 생성

서버의 도커 파일 생성

```
cd /home/ubuntu/compose

$ mkdir back-auth-dockerfile && cd back-auth-dockerfile

$ vim Dockerfile

...

FROM openjdk:8-jdk

ENTRYPOINT java -jar /deploy/BackEnd-0.0.1-SNAPSHOT.jar

EXPOSE 8080

...


cd /home/ubuntu/compose

$ mkdir back-service-dockerfile && cd back-service-dockerfile

$ vim Dockerfile

...

FROM openjdk:8-jdk

ENTRYPOINT java -jar /deploy/BackEnd-Service-0.0.1-SNAPSHOT.jar

EXPOSE 8080

...
```


Docker-compose.yml 설정

```
version: "3"

services:

  jenkins:

    container_name: jenkins-compose

    build:

      context: jenkins-dockerfile

      dockerfile: Dockerfile

    user: root

    ports:

      - 8080:8080

      - 50000:50000

    volumes:

      - /home/ubuntu/compose/jenkins:/var/jenkins_home

      - /home/ubuntu/compose/.ssh:/root/.ssh

  back-service:

    container_name: back-service-compose

    build:

      context: back-service-dockerfile

      dockerfile: Dockerfile

    ports:

      - 8091:8080
```

```
volumes:
  - /home/ubuntu/compose/jenkins/workspace/a606-ci-cd/BackEnd-Service/build/libs:/deploy
  - /tmp/a606object:/tmp/a606objectcard

back-auth:

  container_name: back-auth-compose

  build:

    context: back-auth-dockerfile

    dockerfile: Dockerfile

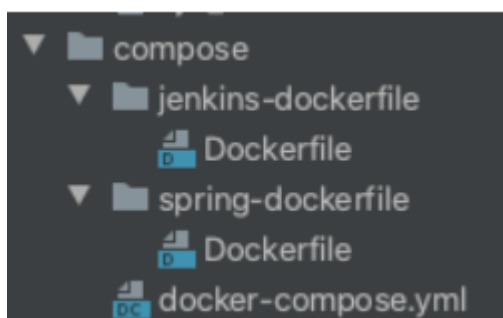
  ports:

    - 8090:8080

  volumes:

    - /home/ubuntu/compose/jenkins/workspace/a606-ci-cd/BackEnd/build/libs:/deploy
    - /tmp/a606/profile:/tmp/a606/profile
```

현재까지 디렉터리 구조



도커 compose 명령어 실행

```
Docker-compose up --build -d
```

6. Jenkins 에서 ec2에 명령을 보내기 위한 ssh 설정


다음 명령어로 젠킨스 도커에 진입한다.(성공시 root@어쩌구로 시작하는 이름 나온다)

```
docker exec -it jenkins-compose bash
```

다음 명령어를 통해 키를 생성한다(중간 질문에는 모두 입력하지 않고 엔터)

제일 아래 키는 복사한다.

```
$ ssh-keygen -t rsa  
  
$ cat /root/.ssh/id_rsa.pub  
  
$ exit
```



```
root@06dfc2eea269:/# ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /root/.ssh/id_rsa.  
Your public key has been saved in /root/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:ZuBwQ790J6ZLxgvfC9fhPHJ0wESTFa902ustHuVyeQ root@06dfc2eea269  
The key's randomart image is:  
----[RSA 2048]-----  
  . o . . . |  
  . = . . . |  
  . + + o . |  
  + o o o . . |  
  . S + o + o |  
  . O X . . + E |  
  . * * B . + . |  
  = * * . o . |  
  + o . . o . |  
-----[SHA256]-----  
root@06dfc2eea269:/# cat /root/.ssh/id_rsa.pub  
ssh-rsa
```

복사한 키를 ec2로 나와 다음 파일의 맨 밑에 등록한다.

```
$ vim ~/.ssh/authorized_keys
```

접속이 잘되나 젠킨스 도커에 들어가서 ec2에 접속해 본다.

여기서 \$(/sbin/ip route | awk '/default/ { print \$3 }')는 ec2의 프라이빗아이피이다.

```
$ docker exec -it jenkins-compose bash  
  
$ ssh ubuntu@$(/sbin/ip route | awk '/default/ { print $3 }')
```

여기까지 했다면 젠킨스 아이템 등록중 빌드에서 설정한 명령어

```
$ ssh -t -t ubuntu@$(/sbin/ip route | awk '/default/ { print $3 }') <<EOF
> cd /home/ubuntu/compose
> docker-compose up --build -d
> exit
> EOF
```

정상적으로 실행 된다.

7. Nginx 설치, 프론트 서버(docker에 nginx 웹서버로 사용하는 서버) 생성

먼저 docker에 nginx를 깔고 vue를 연결하도록 하겠습니다.

Nginx 도커 이미지 다운

```
docker pull nginx
```

Docker로 nginx 실행하기

이때 -v 에는 빌드된 vue 경로를 설정한다. 실행 후 <http://ec2아이피:포트> 접속

```
docker run --name hello-nginx -v /home/ubuntu/compose/jenkins/workspace/a606-ci-cd/FrontEnd/dist/index.html:/usr/share/nginx/html -d -p 8000:80 nginx
```

이후 리버스 프록시용 nginx 설치 후 실행

```
sudo apt update
sudo apt upgrade
sudo apt install nginx
sudo service start nginx
sudo service status nginx
```

8. https 설정(certbot 설치, standalone 방식) 및 openvidu 도커 서버 생성 후 연동

certbot 설치

```
sudo apt update  
sudo apt-get install letsencrypt -y
```

Nginx 중단 후 인증서 발급 받고 재실행

```
cd /root/  
sudo service nginx stop  
sudo certbot certonly --standalone -d 사이트명  
sudo service nginx restart
```

Nginx의 설정 파일 수정(일단 nginx의 설정 파일이 default인지 아니면 nginx.conf 인지 확인하고 설정 파일을 수정해야한다.)

```
sudo vim /etc/nginx/sites-available/default  
  
...  
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
    index index.html index.htm index.nginx-debian.html;  
    server_name _;  
    location / {  
        proxy_pass http://14.125.192.199:8000;  
    }  
}
```

```

server {

    index index.html index.htm index.nginx-debian.html;

    server_name i7a606.q.ssafy.io; # managed by Certbot

    root /home/ubuntu/compose/jenkins/workspace/a606-ci-cd/FrontEnd/dist;

    location / {

        root                /home/ubuntu/compose/jenkins/workspace/a606-ci-
cd/FrontEnd/dist;

        try_files $uri $uri/ @router;

    }

    location /service-api {

        proxy_pass http://13.125.192.199:8091;

    }

    location /auth-api {

        proxy_pass http://13.125.192.199:8090;

    }

    location @router{

        rewrite ^(.+)$ /index.html last;

    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot

    listen 443 ssl; # managed by Certbot

    ssl_certificate /etc/letsencrypt/live/i7a606.q.ssafy.io/fullchain.pem; # managed by
Certbot

    ssl_certificate_key /etc/letsencrypt/live/i7a606.q.ssafy.io/privkey.pem; # managed
by Certbot

}

```

```
server {  
  
    if ($host = i7a606.q.ssafy.io) {  
  
        return 301 https://$host$request_uri;  
  
    } # managed by Certbot  
  
  
    listen 80 ;  
  
    listen [::]:80 ;  
  
    server_name i7a606.q.ssafy.io;  
  
    return 404; # managed by Certbot  
  
}
```

Openvidu 설치

```
# 관리자 권한  
  
$ sudo su  
  
$ cd /opt  
  
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh  
| bash  
  
$ exit  
  
$ cd openvidu
```

Openvidu 도커의 볼륨 수정(임시방편)

```
sudo vim docker-compose.yml
```

```

nginx:
  image: openvidu/openvidu-proxy:2.22.0
  restart: always
  network_mode: host
  volumes:
    - ./certificates:/etc/letsencrypt
    - ./owncert:/owncert
    - ./custom-nginx-vhosts:/etc/nginx/vhost.d/
    - ./custom-nginx-locations:/custom-nginx-locations
    - ./certificates/live/i7a606.q.ssafy.io:/etc/letsencrypt/live/i7a606.q.ssafy.io
    - ./certificates/live/i7a606.q.ssafy.io/privkey.pem:/etc/letsencrypt/live/i7a606.q.ssafy.io/privkey.pem
    - ./certificates/live/i7a606.q.ssafy.io/fullchain.pem:/etc/letsencrypt/live/i7a606.q.ssafy.io/fullchain.pem
    - ./certificates/live/i7a606.q.ssafy.io/chain.pem:/etc/letsencrypt/live/i7a606.q.ssafy.io/chain.pem
    - ./certificates/live/i7a606.q.ssafy.io/cert.pem:/etc/letsencrypt/live/i7a606.q.ssafy.io/cert.pem
    - ${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:/opt/openvidu/custom-layout

```

Openvidu 설정 수정

```
sudo vim .env
```

```

DOMAIN_OR_PUBLIC_IP=i7a606.q.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access it
OPENVIDU_SECRET=A606

# 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=selfsigned

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notification
LETSENCRYPT_EMAIL=a606@ssafy.io

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following

# 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=8442

# Changes the port of all services exposed by OpenVidu.
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=8443

```

기존 인증서 삭제 후 Openvidu 실행

```

$ sudo rm -rf certificates

$ sudo ./openvidu start

```


9. 젠킨스 아이템 빌드 (명령어 추가 후)



6. Redis 서버 설치

레디스 설치 후 버전 확인

```
sudo apt-get update  
  
sudo apt-get install redis redis-tools  
  
redis-server --version
```

외부 입력 허용, 암호 설정 한 뒤 재시작 및 확인

```
sudo vi /etc/redis/redis.conf  
  
'/bind 127.0.0.1 ' 을 검색하여 외부입력 허용 모드인 'bind 0.0.0.0' 으로 수정  
  
'/requirepass ' 명령어를 사용하여 패스워드 입력란 검색  
  
sudo systemctl restart redis-server.service  
  
redis-cli -h <redis server ip> -p <redis port> -a <password>
```

7. Mysql 설치

Mysql 설치 및 세팅

```
$sudo apt-get install mysql-server  
  
$sudo ufw allow mysql  
  
$sudo systemctl start mysql  
  
$sudo systemctl enable mysql
```

접속

```
$sudo /usr/bin/mysql -u root -p
```

비밀번호 변경

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('바꿀비번');
```

사용자 등록 및 권한 설정

```
CREATE USER sss@'%' identified by 'a606fe54';  
  
GRANT ALL PRIVILEGES ON *.* to sss@'%';  
  
FLUSH PRIVILEGES;
```

Workbench 연결

Connection Name:

sss

Type a name for the connection.

Connection Method:

Standard (TCP/IP)

Method to use to connect.

Parameters

SSL

Advanced

Hostname:

i7a606.p.ssafy.io

Port:

3306

Name or IP address of the server host and TCP/IP port.

Username:

sss

Name of the user to connect with.

Password:

Store in Vault ...

Clear

The user's password. Will be requested if not set.

Default Schema:

The schema to use as default schema. Blank to select it later.