

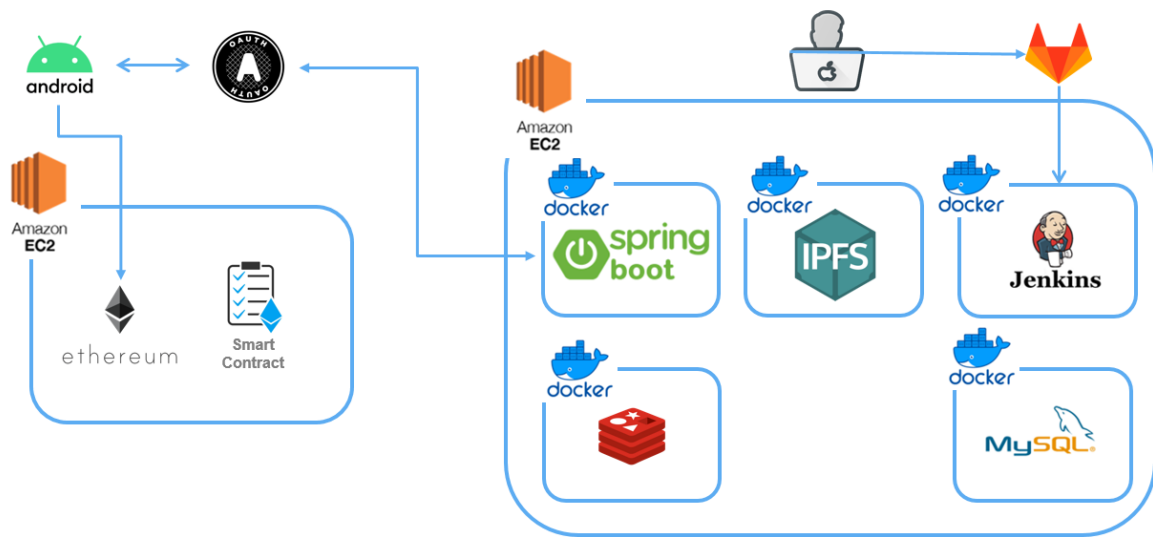


포팅 매뉴얼

1. 프로젝트 기술 스택

Blockchain	
AOS	• Retrofit 2.9.0 (통신 라이브러리) • OkHttp • Ted Permission 3.3.0 (안드로이드 권한 라이브러리) • Dagger-Hilt (의존성 주입 라이브러리) • JetPack Paging3 (페이징 라이브러리) • Coroutines Flow (비동기 데이터 처리 라이브러리) • Glide 4.12.0 (이미지 로드 라이브러리) • ViewModel-ktx 2.3.1 • Fragment-ktx 1.3.6 • Navigation 2.3.5 (화면 전환, 스택 관리 라이브러리) • OAuth (로그인 보안 라이브러리) • ExoPlayer (미디어 플레이어 라이브러리) • Room (내부 데이터 베이스 라이브러리) • Biometric (생체 인식 라이브러리) • Zxing Qr Scan (QR 스캔 라이브러리)
BE	Infra • AWS EC2 • Docker 20.10.17 • Docker-compose 1.25.0 • Jenkins 2.346.2 Development • Java 1.8.0_192(Zulu 8.33.0.1-win64) • Spring boot 2.7.3 • spring-data-jpa 2.7.3 • spring-data-redis 2.7.3 • hibernate-core-5.6.9.Final • spring-security:5.7.3 • projectlombok:1.18.24 Test • junit-jupiter:5.8.2 • mockito-core:4.5.1 • Apache JMeter 5.5 DB • mysql 8.0.28

2. 서버 아키텍처



본 프로젝트의 아키텍처는 위와 같습니다. 각 서버 리소스는 특정 포트로 식별 가능하며 접근할 수 있습니다.

각 서버의 포트 번호는 다음과 같습니다. (도커 배포 기준)

서버	HTTP 포트	HTTPS 포트
tomcat	8080	8443
jenkins	8088	-
redis	6379	-
mysql	3306	-

3. 프로젝트 빌드 방법 (로컬 서버)

3.1. Gitlab에서 프로젝트 클론하기

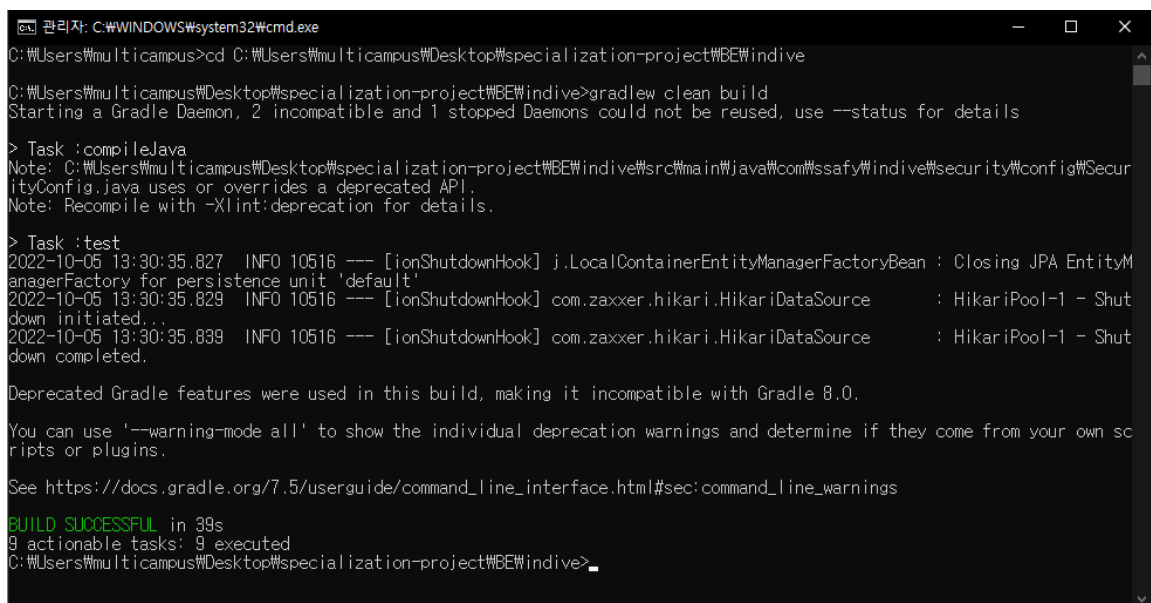
1. 작업할 공간에 폴더를 하나 생성합니다.
2. 생성한 폴더를 열고 해당 위치에서 `Git Bash` 를 열어줍니다. (`CMD` 와 같은 다른 터미널도 상관없습니다!)
3. `git clone https://lab.ssafy.com/s07-blockchain-contract-sub2/S07P22D102.git` 를 터미널에 입력해줍니다.

4. gitlab에서 내려받은 파일이 생깁니다. 앞으로 해당 파일이 위치한 폴더를 `root directory` 라고 하겠습니다. 이후 작업 공간으로 가서 빌드 과정을 수행해주시면 됩니다.

3.2. 스프링부트 WAS 빌드

3.2.1. gradle로 직접 빌드하는 방법 (CMD 버전)

1. `Win + R` 을 누르고 `cmd` 를 입력하고 확인 버튼을 누릅니다. 그러면 명령 프롬프트 창을 띄울 수 있습니다.
2. 백엔드 작업 공간으로 이동해줍니다. 저의 경우에는 백엔드 작업 공간이 `C:\Users\multicampus\Desktop\specialization-project\BE\indive` 입니다. 앞에 `cd` 명령어를 붙이시면 해당 디렉토리로 이동할 수 있습니다.
3. 그 후 `gradle` 을 이용하여 빌드해줍니다. `cmd` 에 `gradlew clean build` 명령어를 입력합니다. 그러면 서버 내부에서 진행하는 테스트 코드를 수행한 후 빌드 파일이 생깁니다.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\multicampus>cd C:\Users\multicampus\Desktop\specialization-project\BE\indive
C:\Users\multicampus\Desktop\specialization-project\BE\indive>gradlew clean build
Starting a Gradle Daemon, 2 incompatible and 1 stopped Daemons could not be reused, use --status for details
> Task :compileJava
Note: C:\Users\multicampus\Desktop\specialization-project\BE\indive\src\main\java\com\ssafy\indive\security\config\SecurityConfig.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
> Task :test
2022-10-05 13:30:35.827 INFO 10516 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityM
anagerFactory for persistence unit 'default'
2022-10-05 13:30:35.829 INFO 10516 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shut
down initiated...
2022-10-05 13:30:35.839 INFO 10516 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shut
down completed.
Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own sc
ripts or plugins.
See https://docs.gradle.org/7.5/userguide/command_line_interface.html#sec:command_line_warnings
BUILD SUCCESSFUL in 39s
9 actionable tasks: 9 executed
C:\Users\multicampus\Desktop\specialization-project\BE\indive>
```

그림 2) gradle build

4. `cd build/libs` 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 `java -jar indive-0.0.1-SNAPSHOT.jar` 명령어를 입력해줍니다.
5. 위의 과정을 마치면 로컬 환경에서 서버 빌드 및 배포가 되었습니다.
<http://localhost:8080/swagger-ui/> 로 접속하시면 API를 확인하실 수 있습니다.

3.2.2. gradle로 직접 빌드하는 방법 (IntelliJ)

1. 인텔리제이를 통해 해당 프로젝트를 열어줍니다.

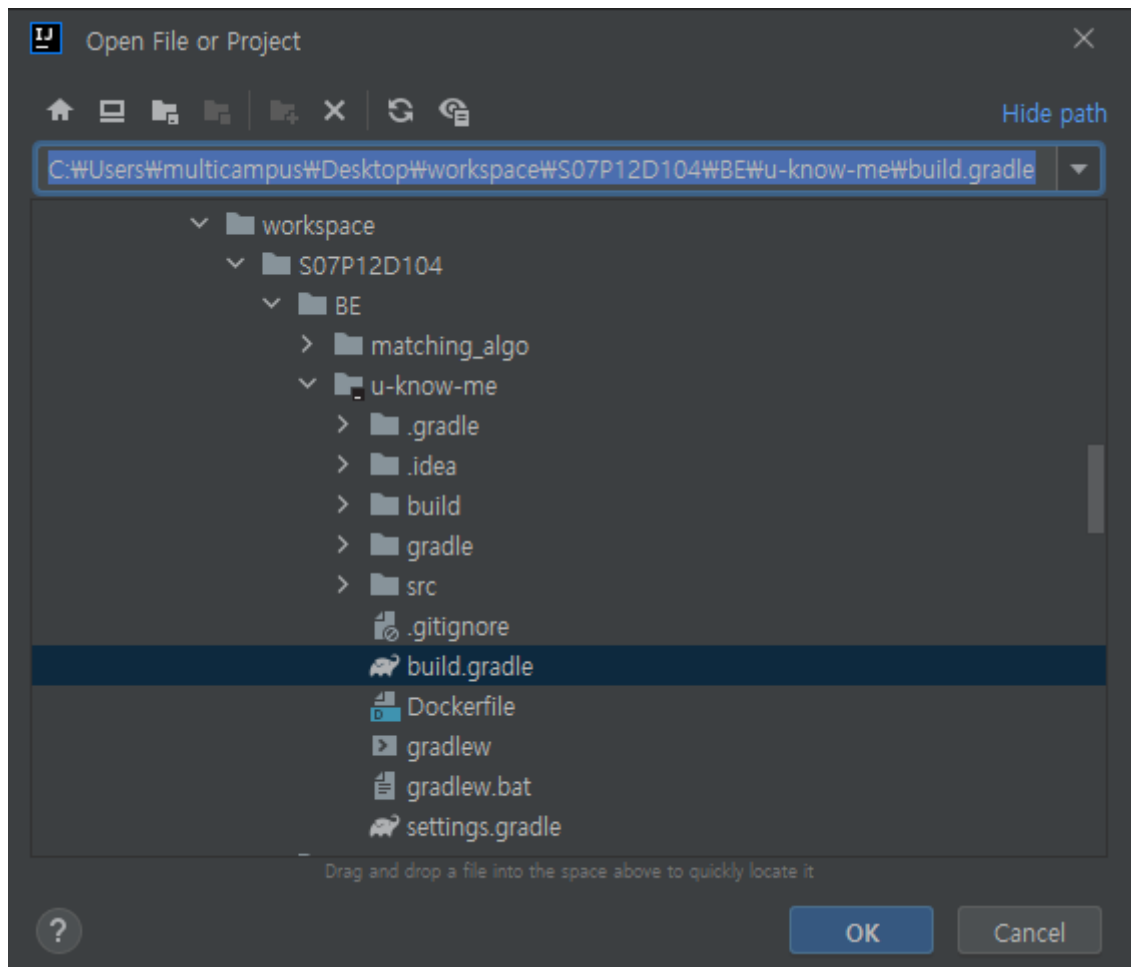


그림 3) 오픈 프로젝트를 통해 백엔드 프로젝트 열기

2. **Alt + F12** 를 눌러 터미널을 열어줍니다.

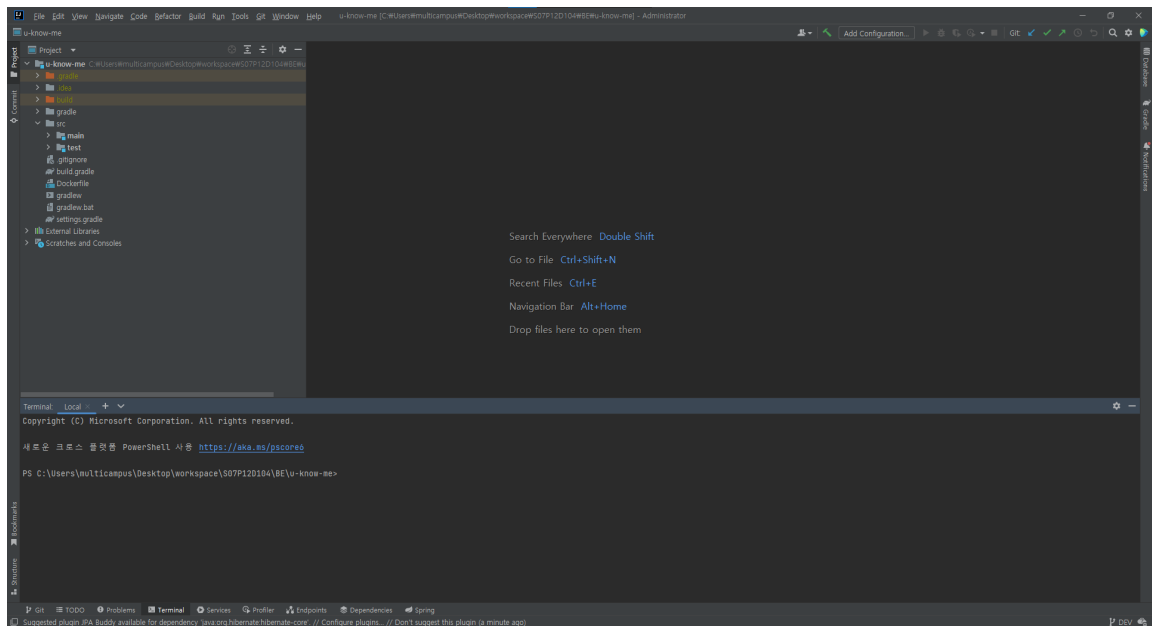


그림 4) 터미널을 연 상태의 IntelliJ

3. 터미널에 `./gradlew clean build` 명령어를 입력합니다.
4. `BUILD SUCCESSFUL` 이 뜨면 `cd build/libs` 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 `java -jar indive-0.0.1-SNAPSHOT.jar` 명령어를 입력해줍니다.
5. 스프링부트 서버가 정상적으로 올라가면 <http://localhost:8080/swagger-ui/> 로 접속해서서 API를 확인하실 수 있습니다.

4. 프로젝트 빌드 방법 (운영 서버)

4.1. VSCode를 이용한 ssh 접속

1. **Remote - SSH** 설치

vscode에서 `ctrl + shift + x` 를 누르면 extensions 탭으로 넘어갈 수 있습니다. 해당 화면에서 검색창에 `ssh` 를 검색하면 `Remote - SSH` 라는 extension이 나오는데 `install` 버튼을 눌러 다운로드 받아줍니다.

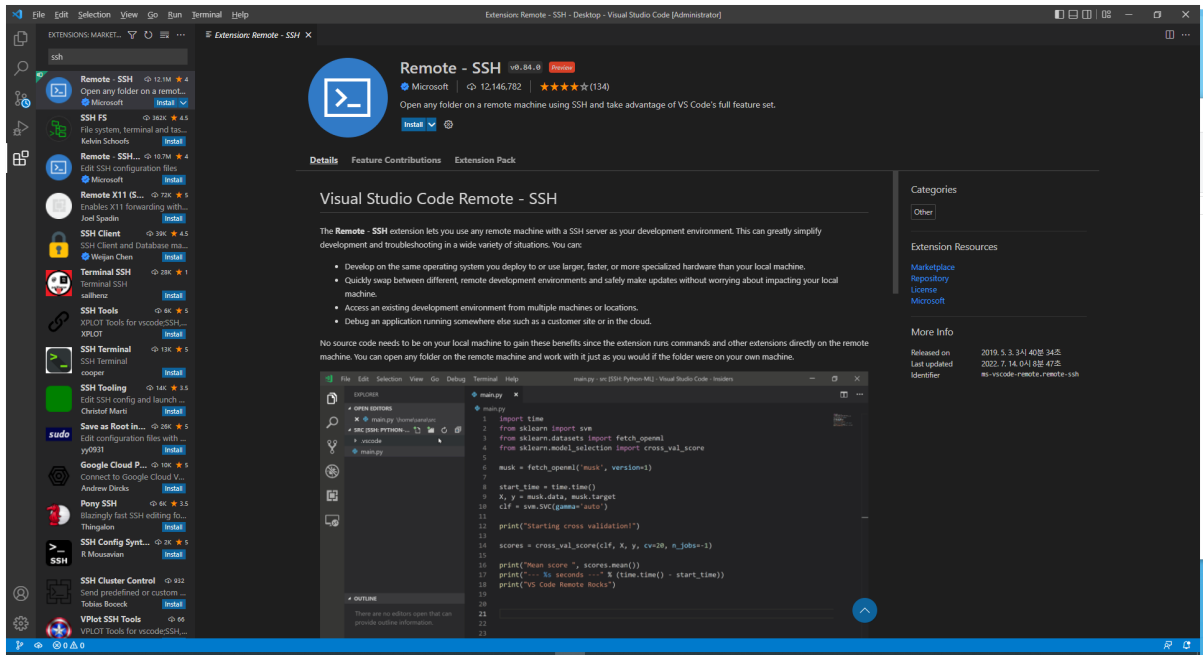


그림 5) extensions 탭

2. SSH 설정 파일 등록

원래 터미널에서 `ssh -i 계정명@IP주소` 로 연결할 수 있지만 계속 터미널에 명령어를 입력하기는 번거로우니 설정 파일을 등록해줍니다.

우선 **f1** 버튼을 눌러 **ssh** 를 검색합니다. 그리고 **Remote-SSH:Open SSH Configuration File...** 이라는 탭을 선택합니다.

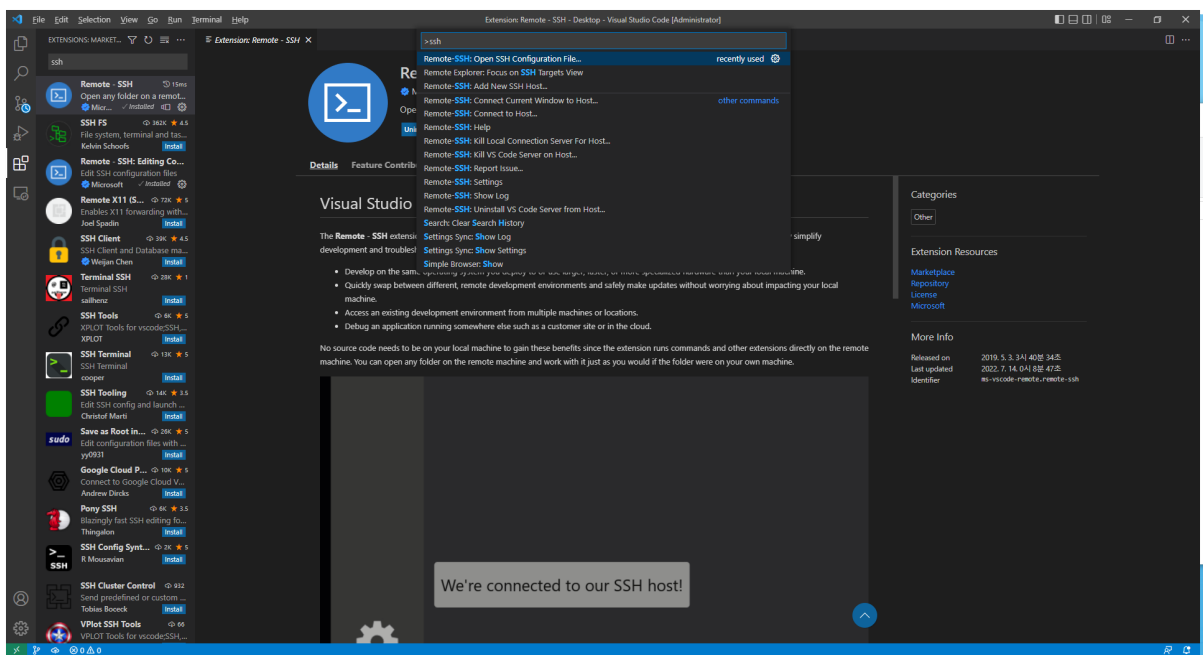


그림 6) ssh 검색

위의 버튼을 눌렀다면 같은 자리에 SSH 구성 파일 리스트가 나열됩니다. 여기서 `C:\Users\<계정명>\.ssh\config` 를 선택합니다.

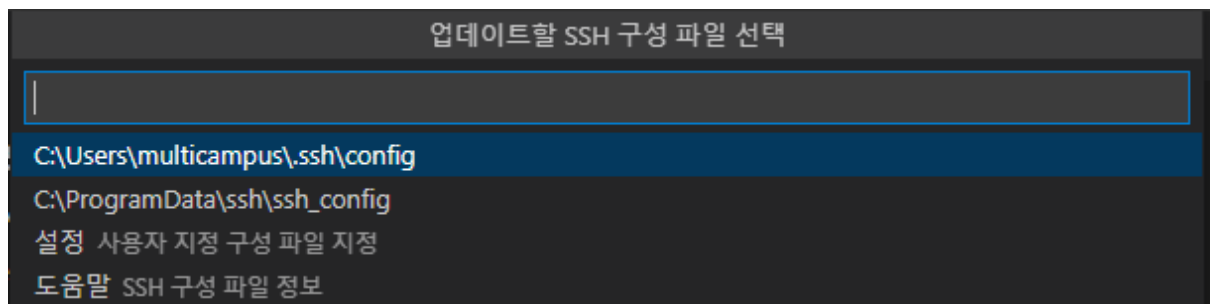


그림 7) SSH 구성 파일 리스트

그러면 SSH 구성 파일이 열립니다.

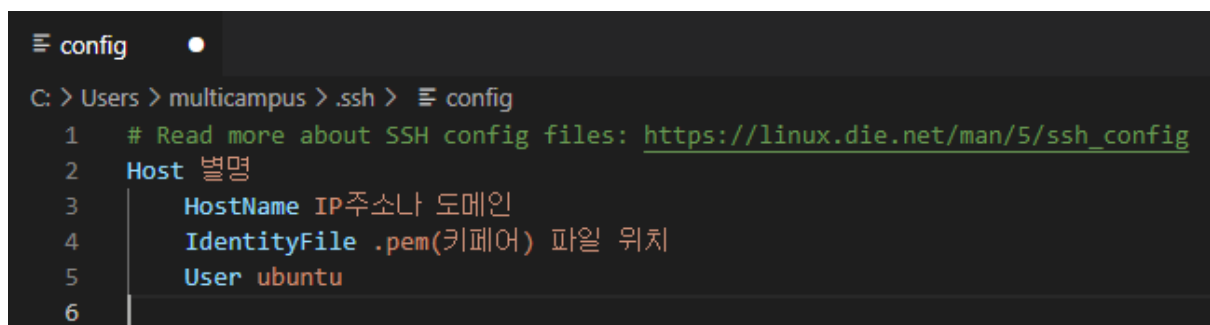


그림 8) SSH 구성 파일

각 요소에 대해 자세히 알아보겠습니다.

- **Host** : Remote SSH 의 이름을 설정해주면 됩니다. (해당 인스턴스가 무엇인지 알기 쉽게 이름을 정합니다.)
- **HostName** : AWS EC2 인스턴스의 public IP 나 도메인을 적으면 됩니다.
- **IdentityFile** : 현재 .pem 파일이 저장되어있는 위치를 작성하면 됩니다.
- **User** : 계정 이름을 설정한다. 우리는 ubuntu 를 사용합니다.
- **Port** : 기본값인 22번 포트가 아니라 다른 포트로 ssh 접근을 한다면 입력해줍니다.

3. SSH 세션 접속

입력을 다하고 저장한 후 좌측 탭에서 Remote Explorer 탭으로 이동합니다.

SSH TARGET 에 config에서 설정한 Host명으로 아이콘이 하나 생깁니다. Host명 우측의 폴더 아이콘을 클릭합니다.

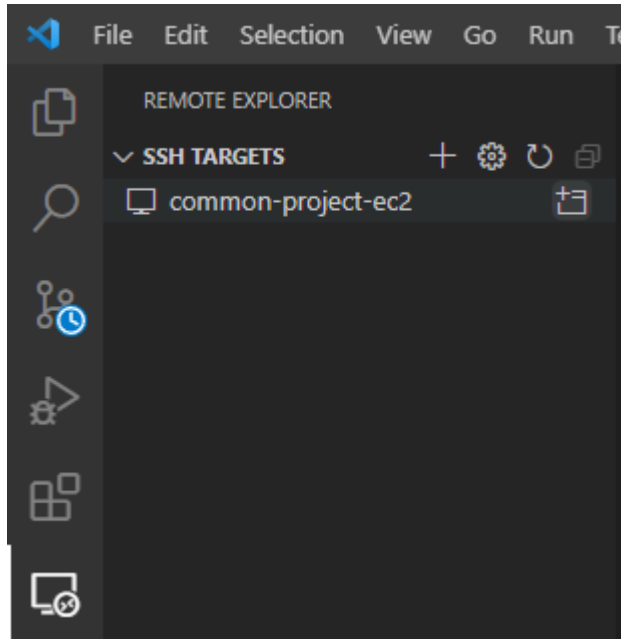


그림 9) Remote Explorer

그럼 새 vscode 창이 뜨면서 Linux, Windows, macOS를 선택하는 창이 나옵니다. 우리는 우분투를 사용하기 때문에 리눅스를 선택합니다.

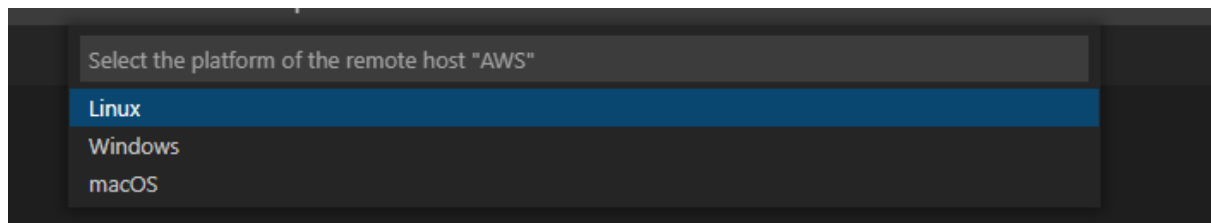


그림 10) AWS 플랫폼 선택창

그러면 SSH를 이용하여 AWS EC2 인스턴스를 vscode에서 편집할 수 있게 됩니다.

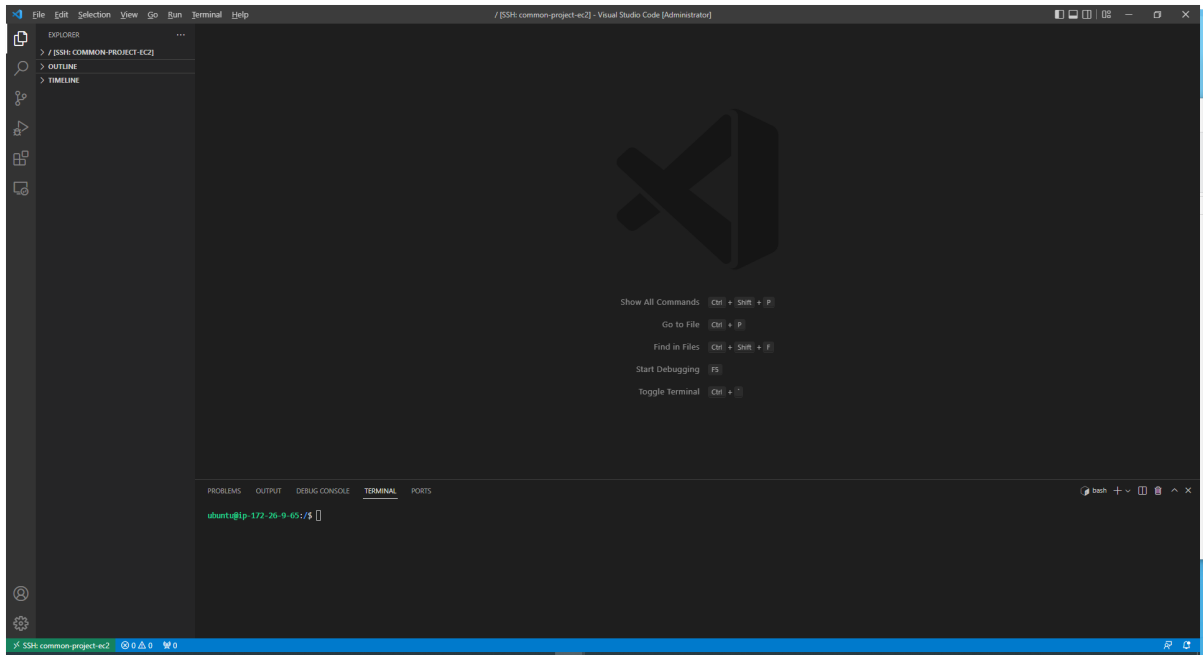


그림 11) Remote SSH에 연결된 모습

4.2. Let's encrypt 인증서 발급

우리 프로젝트에서는 openvidu를 사용하기 위해 ssl 인증서를 발급받아야 했습니다. 저희는 let's encrypt 인증서 발급 방식 중 **standalone** 방식을 이용하여 인증서를 발급받았습니다.

이 방식은 80번 포트로 가상 standalone 웹 서버를 띄워 인증서를 발급받는 방식으로 동시에 여러 도메인에 대해 인증서를 발급받을 수 있다는 장점이 있지만 인증서 발급 전에 nginx 서버를 중단해야 한다는 단점이 있습니다.

```
sudo apt update

// letsencrypt 패키지 설치
sudo apt-get install letsencrypt -y

// 실행중인 nginx 종료
service nginx stop

// SSL 인증
certbot certonly --standalone -d uknowme.moou.com
```

certbot 명령을 실행시켰을 때 만약 인증서가 없다면 인증서를 발급받는 과정을 거칩니다.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
```

Enter email address (used for urgent renewal and security notices) (Enter 'c' to cancel):

다음은 서비스 약관에 동의하는지 묻습니다. 동의 해줍니다.

Please read the Terms of Service at <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>. You must agree in order to register with the ACME server at <https://acme-v02.api.letsencrypt.org/directory>

(A)gree/(C)ancel:

다음은 이메일 주소를 공유할 것인지를 묻습니다. 공유한다면 Y, 아니면 N을 입력하면 됩니다.

Would you be willing to share your email address with the Electronic Frontier Foundation, a founding partner of the Let's Encrypt project and the non-profit organization that develops Certbot? We'd like to send you email about our work encrypting the web, EFF news, campaigns, and ways to support digital freedom.

(Y)es/(N)o:

인증 완료 후 `certbot certificates` 명령어를 통해 제대로 발급이 되었는지 확인해줍니다.

```
ubuntu@ip-172-26-10-252:~$ sudo certbot certificates
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-----
Found the following certs:
Certificate Name: j7d102.p.ssafy.io
Domains: j7d102.p.ssafy.io
Expiry Date: 2022-12-24 04:09:47+00:00 (VALID: 79 days)
Certificate Path: /etc/letsencrypt/live/j7d102.p.ssafy.io/fullchain.pem
Private Key Path: /etc/letsencrypt/live/j7d102.p.ssafy.io/privkey.pem
-----
```

그림 12) letsencrypt 인증서가 발급된 모습

4.3. Jenkins 설정

4.3.1. 컨테이너 실행

우선 Jenkins 컨테이너를 서버에 올려줍니다. 명령어는 다음과 같습니다.

```
docker run --name jenkins-server -itd -p 8088:8080 -v /jenkins:/var/jenkins_home -u root jenkins/jenkins:lts
```

컨테이너를 올린 후 `{hostname}:8088` 에 접속한 후 조금 기다리면 패스워드를 입력해달라는 화면이 등장합니다.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password



Continue

이 때 터미널에서 `docker logs jenkins-server` 를 입력하면 비밀번호를 확인할 수 있습니다.

```
*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

9598d100ec6f40e0a9d6474c1b008aa2

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

해당 비밀번호를 입력 후 `Install suggested plugins` 를 클릭한 후 플러그인을 다운로드 받아줍니다.

Getting Started

Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 2.361.1

Skip and continue as admin

Save and Continue

플러그인을 모두 다운로드 받으면, 유저를 등록하는 화면이 등장합니다. 모두 입력해주시고 **Save and Continue** 버튼을 클릭해 넘어갑니다.

그 후 마지막으로 바로 **Save and Finish** 버튼을 누르고 Jenkins 메인 화면으로 넘어갑니다.

4.3.2. 플러그인 설치

왼쪽 탭 **Jenkins 관리** 를 클릭 후 **플러그인 관리** 로 들어가줍니다. **설치 가능** 탭으로 이동해서 **GitLab** 과 **Publish Over SSH** 를 체크하고 다운로드 받아줍니다.

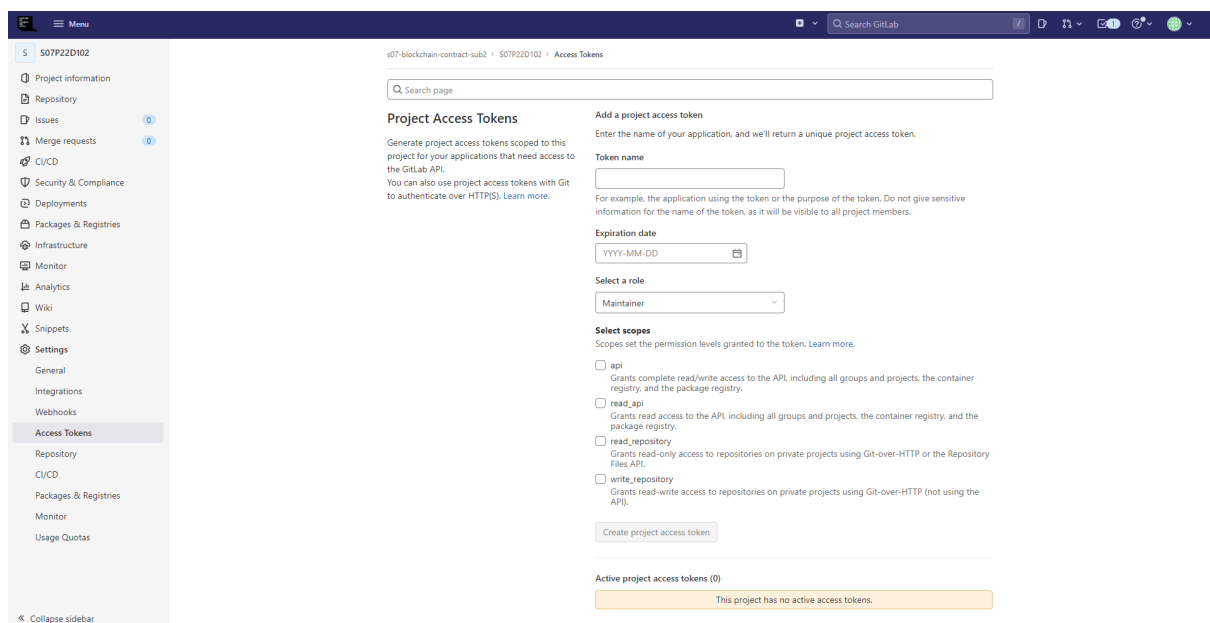
4.3.3. 시스템 설정

- **Gitlab**

왼쪽 탭 **Jenkins 관리** 를 클릭 후 **시스템 설정** 으로 들어가줍니다.

쪽 내려가서 **Gitlab** 에서 **Gitlab Connection** 을 설정해줍니다. **Connection name** 은 원하시는 이름으로 설정하시면 되고 **Gitlab host URL** 은 `https://lab.ssafy.com`, Credential은 Gitlab 에서 발급한 Access Token으로 등록해줍니다.

Access Token을 획득하는 방법은 Gitlab Repository 왼쪽 탭에서 **Settings** → **Access Token** 에서 발급 받으실 수 있습니다.



- **Publish over SSH**

Gitlab에서 프로젝트를 가져오고, 빌드한 후 만들어진 자바 압축 파일을 배포하기 위해서는 해당 파일을 도커 컨테이너 내부에서 ec2 서버로 전송해야 합니다. 방법은 다음과 같습니다. 우선 **Publish over SSH** 으로 이동해줍니다.

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

Path to key ?

Key ?

☐ Disable exec ?

Key 를 입력해줍니다. key는 ssh에 접속할 때 사용하는 키 값을 입력하면 되는데, 저희 같은 경우에는 제공받은 pem 파일의 값을 입력해줍니다.

SSH Server 에서 추가 버튼을 누릅니다.

SSH Servers

SSH Server

Name ?

ec2-server

Hostname ?

j7d102.p.ssafy.io

Username ?

ubuntu

Remote Directory ?

/opt/specialization-project

고급...

Test Configuration

- **Name** : 원하는 값을 입력해주시면 됩니다.
- **Hostname** : ec2 서버의 Public IP 주소나 도메인 네임을 입력해주시면 됩니다.
- **Username** : ec2 서버의 username을 입력해주시면 됩니다. 보통 **ubuntu** 로 입력합니다.

- **Remote Directory** : 파일을 전송할 디렉토리 위치를 입력해주시면 됩니다.

4.3.4. Pipeline 생성

- **Item 생성**

Jenkins 메인 화면 왼쪽 탭에서 **새로운 Item** 을 클릭합니다.

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 동등 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.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

OK

이름을 입력하고 Pipeline을 선택한 후 **OK** 를 누릅니다.

- **GitLab Webhook**

Item이 생성되면 해당 Item을 클릭 후 **구성** 으로 들어갑니다.

우선 **Build Triggers** 를 설정합니다. **Build when a change is pushed to GitLab. GitLab webhook URL: ~~~** 을 체크해줍니다.

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://3.34.252.202:8089/project/specialization-project-pipeline> ?

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

고급...

고급 버튼을 눌러 **Allowed Branches** 에서 **Filter branches by name** 을 체크하고 **Include** 에서 develop을 입력해줍니다.

Allowed branches

☐ Allow all branches to trigger this job ?

☒ Filter branches by name ?

Include

develop

Exclude

그 후 **Secret Token** 에서 **Generate** 버튼을 눌러 토큰을 발급받아 줍니다.

Secret token ?

6fd66fbfc42b03f0a5b38e26ee4ee0c7d

Generate

그리고 GitLab으로 들어가줍니다. **Settings** → **Webhooks** 에서 Push Event가 일어날 경우 웹 후크를 생성해줍니다.

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

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

Secret token

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

- **URL** : Build when a change is pushed to GitLab. GitLab webhook URL: ~~~ 에서 ~~~ 부분을 입력해줍니다.
- **Secret token** : 방금 발급 받은 Secret Token을 입력해줍니다.

그 후 **Add Webhook** 을 클릭하고 등록해줍니다.

• Pipeline

파이프라인 스크립트를 입력해줍니다.

```
pipeline {
  agent any

  stages {
    stage('git clone') {
      steps {
        git branch: 'develop', credentialsId: 'gitlab-authentication', url: 'https://lab.ssafy.com/s07-blockchain-contract-sub2/S07P22D102.git'
      }
    }

    stage('build') {
      steps {
        dir('BE/indive') {
          sh '''
            chmod +x gradlew
            ./gradlew clean build -Pprofile=prod
          '''
        }
      }
    }

    stage('deploy') {
      steps {
        sshPublisher(publishers: [sshPublisherDesc(configName: 'ec2-server', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '''docker stop be-server

        docker rm be-server

        cd /opt/specialization-project

        docker build -t mungmnb777/be-server .

        docker run -d -p 8443:443 -p 8080:8080 -v /etc/letsencrypt:/etc/letsencrypt -v /opt/specialization-project/files:/opt/specialization-project/files --name be-server mungmnb777/be-server''', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false, removePrefix: 'BE/indive/build/libs', sourceFiles: 'BE/indive/build/libs/indive-0.0.1-SNAPSHOT.jar']], usePromotionTimestamp: false, useWorkspaceInPromotion: false, verbose: false)])
      }
    }
  }
}
```

```
}  
}  
}
```

4.4. MySQL 컨테이너 생성

다음 명령어로 MySQL 컨테이너를 생성합니다.

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mysql:latest
```

workbench를 이용해 `{hostname}:3306` 으로 해당 DB 서버에 접속한 후 `create database indive` 를 통해서 데이터베이스를 생성해줍니다.

추가로 저희는 테스트용 MySQL 서버를 만들어두었습니다.

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 33066:3306 mysql:latest
```

4.5. Redis 컨테이너 생성

다음 명령어로 Redis 컨테이너를 생성합니다.

```
docker run --name redis-server -d -p 6379:6379 redis
```

4.6. IPFS 컨테이너 생성

1. Golang 설치

```
sudo apt install golang -y
```

2. IPFS 도커 이미지 가져오기

```
docker pull ipfs/go-ipfs
```

3. 개인 체인 키 생성

```
go get -u github.com/Kubuxu/go-ipfs-swarm-key-gen/ipfs-swarm-key-gen
cd ~/go/src/github.com/Kubuxu/go-ipfs-swarm-key-gen/ipfs-swarm-key-gen
go build
./ipfs-swarm-key-gen > ~/.ipfs/swarm.key
```

4. 디렉토리 생성

```
mkdir -p ~/ipfs/ipfs1/data ~/ipfs/ipfs1/export ~/ipfs/ipfs2/data ~/ipfs/ipfs2/export
```

5. 체인 키 복사

```
cp ~/.ipfs/swarm.key ~/ipfs/ipfs1/data
cp ~/.ipfs/swarm.key ~/ipfs/ipfs2/data
```

6. 컨테이너 실행

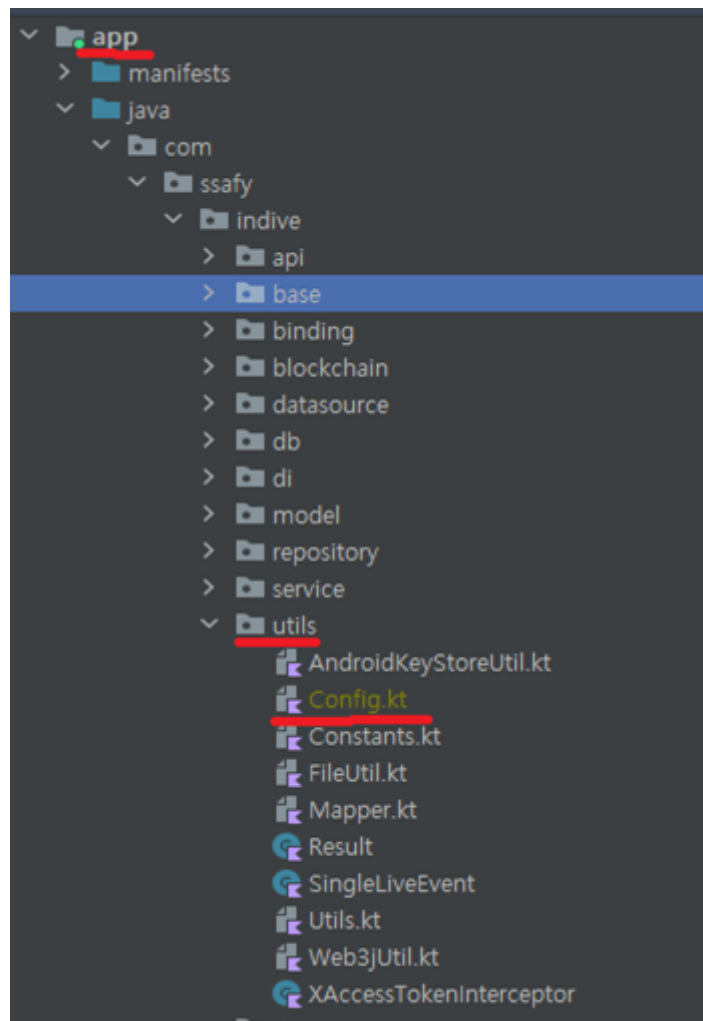
```
docker run -d --name ipfs_host1 --privileged=true --entrypoint="sh" -v ~/ipfs/ipfs1/export:/export -v ~/ipfs/ipfs1/data:/data/ipfs -p 4001:4001 -p 8085:8080 -p 5001:5001 ipfs/go-ipfs:latest -c "while true; do sleep 1;done"

docker run -d --name ipfs_host2 --privileged=true --entrypoint="sh" -v ~/ipfs/ipfs2/export:/export -v ~/ipfs/ipfs2/data:/data/ipfs -p 4011:5001 -p 9085:8080 -p 5011:5001 ipfs/go-ipfs:latest -c "while true; do sleep 1;done"
```

안드로이드

https://drive.google.com/file/d/12_U3f_FkO5hVujfE8xeDg_1LeY6bwkBW/view?usp=sharing

1. Config.kt



App -> java -> com.ssafy.indive -> utils 에 Config.kt 추가