포팅 매뉴얼

1. Version

AOS

- Android Studio Dolphin 2021.3.1 patch 1
- Kotlin 1.7.20
- RoomDB 2.5.0
- SdkVersion
    - compileSdkVersion : 33
    - targetSdkVersion : 33
    - minSdkVersion : 21

Back-end

- Intellij Ultimate 2022.3.1
    - 낮은 버전 사용 시 gradle 버전 낮춰서 사용해야 함
- Java 11
- Spring boot 2.7.7
- gradle 7.6
- MySQL 8.0.32
- QueryDSL 5.0.0

2. 외부 API

AOS

- Kakao 소셜 로그인
- Naver Map
- Kakao 주소 API
- Firebase hosting service
- Bootpay
- firebase

Back-end

- AWS S3
- Kakao 소셜 로그인
- Firebase cloud messaging
- Google Cloud language
- nurigo

▼ AOS 설정 파일

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'org.jetbrains.kotlin.kapt'
    id 'kotlin-parcelize'
    id 'androidx.navigation.safeargs.kotlin'
    id 'com.google.gms.google-services'
}

android {
    namespace 'com.ssafy.smile'
    compileSdk 33

    defaultConfig {
        applicationId "com.ssafy.smile"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

```
    kotlinOptions {
        jvmTarget = '1.8'
    }
    viewBinding{
        enabled = true
    }
    dataBinding{
        enabled = true
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

    // Jetpack Navigation Kotlin 추가
    def nav_version = "2.4.2"
    implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
    implementation "androidx.navigation:navigation-ui-ktx:$nav_version"

    // Room 의존성 추가
    def room_version = "2.5.0"
    implementation "androidx.room:room-runtime:$room_version"
    annotationProcessor("androidx.room:room-compiler-processing:$room_version")
    kapt "androidx.room:room-compiler:$room_version"
    implementation "androidx.room:room-ktx:$room_version"

    // Retrofit 추가
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    // https://github.com/square/retrofit
    // https://github.com/square/retrofit/tree/master/retrofit-converters/gson
    // okhttp3 추가
    implementation 'com.squareup.okhttp3:okhttp:4.9.2'
    implementation 'com.squareup.okhttp3:logging-interceptor:4.9.2'

    // material design 추가
    implementation 'com.google.android.material:material:1.6.1'
    // viewmodel dependency 추가
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.1'
    // liveData dependency 추가
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.4.1'
    // framework ktx dependency 추가
    implementation "androidx.fragment:fragment-ktx:1.4.1"

    // FireStore SDK 추가
    implementation platform('com.google.firebase:firebase-bom:31.2.0')
    // FCM 사용 위한 plugins
    implementation 'com.google.firebase:firebase-messaging-ktx'

    // permission Library 추가
    implementation 'io.github.ParkSangGwon:tedpermission-normal:3.3.0'
```

```
    // 네이버 map api sdk 추가
    // implementation 'com.google.android.gms:play-services-maps:18.0.2'
    implementation 'com.google.android.gms:play-services-location:16.0.0'
    implementation 'com.naver.maps:map-sdk:3.16.0'

    // 카카오 로그인 api sdk 추가
    implementation 'com.kakao.sdk:v2-user:2.10.0'

    // 부트페이 결제 sdk 추가
    implementation 'io.github.bootpay:android:4.2.3'

    // Glide Library 추가
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'

    // Crop Library 추가
    implementation 'com.github.CanHub:Android-Image-Cropper:4.3.0'


    // 코루틴 Library 추가
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.2'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.2'
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.3.1"

    // Spin-kit 추가 - 로딩화면 https://github.com/ybq/Android-SpinKit
    implementation 'com.github.ybq:Android-SpinKit:1.4.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'

    // lottie  Library 추가
    implementation 'com.airbnb.android:lottie:5.2.0'

    // toasty Library 추가
    implementation 'com.github.GrenderG:Toasty:1.3.0'

    // multiple ImagePicker Library 추가
    implementation 'io.github.ParkSangGwon:tedimagepicker:1.4.0'

    // page indicator view Library 추가
    implementation 'com.romandanylyk:pageindicatorview:1.0.3'

    // custom calendar view Library 추가
    implementation 'com.github.prolificinteractive:material-calendarview:2.0.1'

    // custom ratingBar view Library 추가
    implementation 'tw.com.oneup.www:reviewbar:1.0.2'

    // swipe refresh layout dependency 추가
    implementation("androidx.swiperefreshlayout:swiperefreshlayout:1.1.0")

}
```

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
plugins {
```

```
    id 'com.android.application' version '7.3.1' apply false
    id 'com.android.library' version '7.3.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.7.20' apply false
    // Jetpack Nav safeArgs 추가
    id 'androidx.navigation.safeargs' version '2.4.2' apply false
    // FireStore SDK 추가
    id 'com.google.gms.google-services' version '4.3.15' apply false
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
# http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decou
pled_projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are bundled with th
e
# Android operating system, and which are packaged with your app's APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
# Enables namespacing of each library's R class so that its R class includes only
 the
# resources declared in the library itself and none from the library's dependencie
s,
# thereby reducing the size of the R class for that library
android.nonTransitiveRClass=true
# naver map api error
android.enableJetifier=true
```

```
pluginManagement {
    repositories {
        gradlePluginPortal()
        google()
        mavenCentral()
        maven { url "https://jitpack.io" }
        jcenter()
    }
}
dependencyResolutionManagement {
```

```
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven { url "https://jitpack.io" }
        maven {url 'https://naver.jfrog.io/artifactory/maven/'}
        maven { url 'https://devrepo.kakao.com/nexus/content/groups/public/'}
        jcenter()
    }
}
rootProject.name = "Smile_Android"
include ':app'
```

▼ AOS api key

```
📂 app
└ 📂 src/main/res
  └ 📂 values
    └ strings.xml
    └ appkey.xml
└ google-services.json
```

```
<string name="address_search_url">https://{프로젝트이름(소문자)}.web.app</string>
<string name="address_search_execute_url">javascript:execDaumPostcode();</string>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- AndroidManifest.xml에서 쓰일 앱키(kakao+네이티브 앱 키) -->
    <string name="kakao_native_key_manifest"></string>
    <!-- kakao developers에 명시된 네이티브 앱 키 -->
    <string name="kakao_native_key"></string>
    <string name="naver_map_key"></string>
    <string name="bootpay_key"></string>
</resources>
```

```
1. https://console.firebase.google.com 에서 프로젝트 생성
2. 앱 추가에서 안드로이드 앱 선택 후 패키지명 입력(com.ssafy.smile)
3. google-services.json 파일 다운로드 후 app 폴더 하위에 파일 넣기
4. 설정 파일 세팅
5. google-services.json 파일 다시 다운로드 받고 싶다면 프로젝트 설정>내 앱에서 다운
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    ...
```

```
    <application
        ...>
        <activity
            android:name="com.kakao.sdk.auth.AuthCodeHandlerActivity"
            android:exported="true">
            <intent-filter>
                ...
                <data
                    android:host="oauth"
                    android:scheme="@string/kakao_native_key_manifest" />
            </intent-filter>
        </activity>
        ...
        <meta-data
            android:name="com.naver.maps.map.CLIENT_ID"
            android:value="@string/naver_map_key" />
    </application>
</manifest>
```

```
class Application : Application()  {
    ...
    override fun onCreate() {
        super.onCreate()
        ...
        bootPayInit()
    }

    ...

    private fun kakaoInit() {
        KakaoSdk.init(this, getString(R.string.kakao_native_key))
    }

    private fun bootPayInit() {
        BootpayAnalytics.init(this, getString(R.string.bootpay_key))
    }
}
```

```
class ReservationFragment : BaseFragment<FragmentReservationBinding>(FragmentReser
vationBinding::bind, R.layout.fragment_reservation) {
    ...
    private fun goBootPayRequest(itemInfo: BootItem) {
        ...
        val payload = Payload().apply {
            applicationId = getString(R.string.bootpay_key)
            ...
        }
    ...
}
```

```
1. https://console.firebase.google.com 에서 생성했던 프로젝트 클릭
2. 앱 추가에서 웹 선택 후 앱 닉네임 등록
3. Firebase CLI 설치
  1) npm install -g firebase-tools
4. firebase 호스팅 배포
  1) firebase login 입력해 로그인
  2) firebase init 입력해 호스팅 선택
  3) Use an existing project 선택
  4) 앞서 만든 앱 선택
  5) 기본 설정 선택
    - What do you want to use as your public directory? public
    - Configure as a single-page app? No
    - Set up automatic builds and deploys with GitHub? No
  6) firebase deploy 입력해 배포
    - index.html 수정해야 한다면 수정 후 firebase deploy 입력해 다시 배포
```

▼ BACK build.gradle

```
//querydsl 추가
buildscript {
    ext {
        queryDslVersion = "5.0.0"
        springBootVersion = '2.7.7'
    }
    repositories {
        maven {url 'https://repo.spring.io/release'}
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBoot
Version}")
    }
}

plugins {
    id 'java'
    id 'org.springframework.boot' version '2.7.7'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'

    //querydsl 추가
    id 'com.ewerk.gradle.plugins.querydsl' version '1.0.10'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
```

```
    mavenCentral()
}

allprojects {}

subprojects {
    apply plugin: 'java'
    apply plugin: 'org.springframework.boot'
    apply plugin: 'io.spring.dependency-management'

    group = 'com.ssafy'
    version = '1.0'
    sourceCompatibility = '11'
    compileJava.options.encoding = 'UTF-8'

    repositories {
        mavenCentral()

        flatDir {
            dirs 'libs'
        }
    }

    dependencies {
        implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
        implementation 'org.springframework.boot:spring-boot-starter-web'
        implementation 'org.springframework.boot:spring-boot-starter-security'
        implementation('org.projectlombok:lombok')
        implementation 'io.jsonwebtoken:jjwt:0.9.1'
        implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.R
ELEASE'
        implementation 'com.vladmihalcea:hibernate-types-52:2.16.2'
        developmentOnly 'org.springframework.boot:spring-boot-devtools'
        runtimeOnly 'com.h2database:h2'
        runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'
        annotationProcessor 'org.projectlombok:lombok'
        testImplementation 'org.springframework.boot:spring-boot-starter-test'
        implementation 'org.springframework.boot:spring-boot-starter-validation'
        implementation 'io.github.bootpay:backend:+'
        implementation("com.mysql:mysql-connector-j:8.0.32")

        // Add querydsl
        implementation 'com.querydsl:querydsl-jpa'
        implementation 'com.querydsl:querydsl-apt'
        annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.impo
rtedProperties['querydsl.version']}:jpa"
        annotationProcessor "jakarta.persistence:jakarta.persistence-api"
        annotationProcessor "jakarta.annotation:jakarta.annotation-api"

        // Add nurigo
        implementation 'net.nurigo:sdk:4.1.3'

        //  retrofit2
        implementation 'com.squareup.retrofit2:adapter-rxjava2:2.7.2'

        // smile-clustering
        implementation 'com.github.haifengl:smile-core:3.0.0'
```

```
        // firebase
        implementation("com.google.firebase:firebase-admin:9.1.1")
        implementation("com.squareup.okhttp3:okhttp:4.10.0")

        // google cloud language
        implementation platform('com.google.cloud:libraries-bom:26.1.4')
        implementation 'com.google.cloud:google-cloud-language'


    }
}

project(":core-module") {
    apply plugin: 'org.springframework.boot'
    apply plugin: 'io.spring.dependency-management'

    bootJar.enabled = false
    jar.enabled = true

    dependencies {
        implementation fileTree(dir: 'libs', include: ['*.jar'])
    }
}

project(":api-module") {
    apply plugin: 'org.springframework.boot'
    apply plugin: 'io.spring.dependency-management'
    apply plugin: 'java-library'

    bootJar {
        archivesBaseName = 'ssafy'
        archiveFileName = "ssafy-api-module-0.0.1.jar"
        archiveVersion = "0.0.1"
    }
    dependencies {
        implementation project(':core-module')
        implementation fileTree(dir: 'libs', include: ['*.jar'])
    }
}

project(':batch-module') {
    apply plugin: 'org.springframework.boot'
    apply plugin: 'io.spring.dependency-management'
    apply plugin: 'java-library'

    bootJar {
        archivesBaseName = 'ssafy'
        archiveFileName = "ssafy-batch-module-0.0.1.jar"
        archiveVersion = "0.0.1"
    }
    dependencies {
        implementation project(':core-module')
        implementation fileTree(dir: 'libs', include: ['*.jar'])
    }
}

/*
 * queryDSL 설정 추가
```

```
 */
// querydsl에서 사용할 경로 설정
def querydslDir = "$buildDir/generated/querydsl"
// JPA 사용 여부와 사용할 경로를 설정
querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}
// build 시 사용할 sourceSet 추가
sourceSets {
    main.java.srcDir querydslDir
}
// querydsl 이 compileClassPath 를 상속하도록 설정
configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
    querydsl.extendsFrom compileClasspath
}
// querydsl 컴파일시 사용할 옵션 설정
compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl
}
```

▼ BACK yml files

```
📂 resources
└ 📂 firebase
  └ firebase_service_key.json
└ 📂 gcloud
  └ gcloud_service_key.json
└ application-pay.yml
└ application-aws.yml
└ application-login.yml
└ application-coolsms.yml
└ application-real.yml
└ application.yml
```

```
cloud:
  aws:
    path: {저장할 위치}
    s3:
      bucket: {S3 버킷 이름}
    region:
      static: {지역}
    # 작성하지 않으면 오류가 발생할 수 있음
    stack:
      auto: false
    credentials:
      accessKey:
      secretKey:
```

```
coolsms:
  smile:
    apiKey:
    apiSecret:
    fromNumber: {메세지를 전송하는 번호}
```

```
kakao:
  oauth2:
    secretPassword:

spring:
  jwt:
    # Random Key
    secret: {암호화할 비밀키}
```

```
pay:
  rest-api:
  private-key:
```

```
# Real Server Configuration File
# Be careful on level

logging:
  level:
    root: warn
    com.ssafy: info
    com.ssafy.api: info
    org.hibernate.type.descriptor.sql: trace
    com.amazonaws.util.EC2MetadataUtils: error
spring:
  profiles:
    active: real
    include:
      - aws
      - coolsms
      - login
      - pay
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    generate-ddl: true
    hibernate:
      # create(new table), update(add new column), none(nothing)
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        use_sql_comments: true
    open-in-view: false
```

```
    defer-datasource-initialization: true
  datasource:
    url: jdbc:mysql://localhost:3306/smile?serverTimeZone=Asia/Seoul&useLegacyDate
timeCode=false
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: {이름}
    password: {비밀번호}
    hikari:
      connection-timeout: 58000
      max-lifetime: 580000
  sql:
    init:
      mode: always
      continue-on-error: true
server:
  port: 80
  domain: http://127.0.0.1
  servlet:
    session:
      timeout: 1440m
  max-http-header-size: 3145728
```

---

3. Deploy

EC2 환경 세팅

> MobaXterm 프로그램 사용

Jenkins 설치

```
$sudo wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key
 add -
$sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
    /etc/apt/sources.list.d/jenkins.list'
```

```
$sudo apt-get update
$sudo apt-get install jenkins
$sudo service jenkins start
```

> *http://도메인:8080* 에 방문하면 Jenkins 시작하면을 볼 수 있음

```
$sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

▼ 에러 발생 시 java 설치

- `java --version`을 통해 설치된 JAVA 버전 확인 후 11 혹은 17 버전이 아닌 경우 두 버전 중 원하는 버전 설치

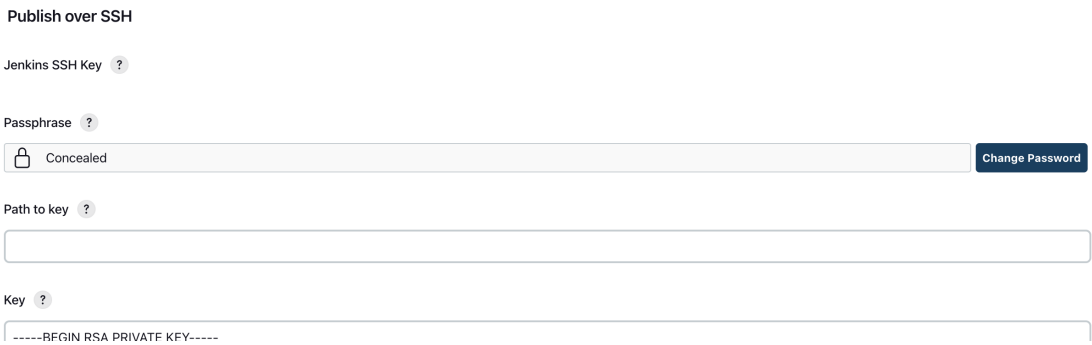
Docker 설치

```
# 패키지 업데이트
sudo apt-get update -y

# 기존에 있던 도커 삭제
sudo apt-get remove docker docker-engine docker.io -y

# 도커 설치
sudo apt-get install docker.io -y

# docker 서비스 실행
sudo service docker start

# /var/run/docker.sock 파일의 권한을 666으로 변경하여 그룹 내 다른 사용자도 접근 가능하게 변경
sudo chmod 666 /var/run/docker.sock

# ubuntu 유저를 docker 그룹에 추가
sudo usermod -a -G docker ubuntu
```

Jenkins 빌드

▼ 기본 환경설정

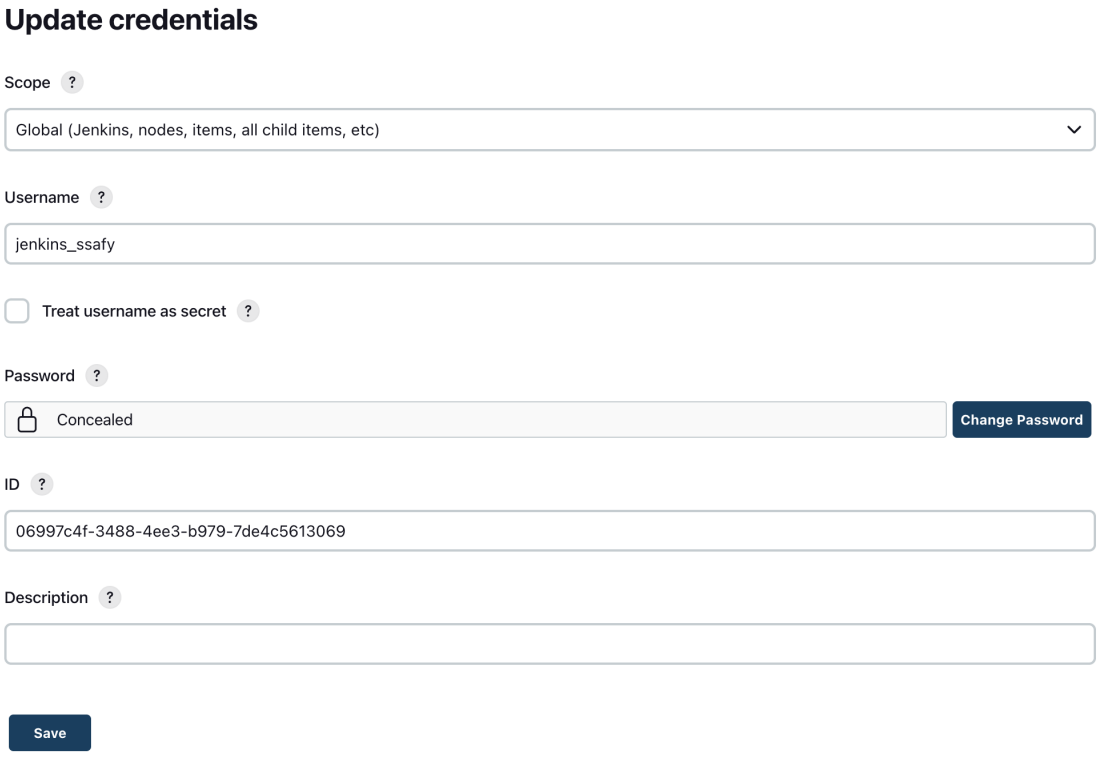
- Installed Plugin
  - GitLab
  - GitLab Plugin
  - SSH Agent Plugin
  - Publish Over Ssh
- 시스템 설정



- Key에 EC2 .pem 파일에 작성된 키 값 복사+붙여넣기

- `Hostname`: EC2 서버 IP 주소 입력
- `Remote Directory`: EC2 서버에 저장할 폴더

- Credentials



Username with password

  - `password` : Repository에서 AccessToken 발급

- `New Item` 클릭 후 `pipeline` 선택
- GitLab 특정 브랜치에 Push Event 발생할 때 자동 빌드 후 배포


```
pipeline {
    agent any

    stages{
        stage('git clone') {
            steps {
                git branch: 'develop',
                    credentialsId: '06997c4f-3488-4ee3-b979-7de4c5613069',
                    url: 'https://lab.ssafy.com/s08-webmobile4-sub2/S08P12D102.git'
            }
        }
```

```
        stage('build') {
            steps{
                dir('Smile_Server'){
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean"
                    sh "./gradlew api-module:bootJar"
                    sh "./gradlew batch-module:bootJar"
                }
            }
        }
        stage('SSH-EC2'){

            steps{
                sh 'pwd'
                script {
                    if(fileExists('*.jar')){
                        sh 'rm -rf *.jar'
                    }
                }
                sshPublisher(publishers: [
                    sshPublisherDesc(
                        configName: 'jenkins_ssafy',
                        transfers: [
                            sshTransfer(
                                cleanRemote: false,
                                excludes: '',
                                execCommand: 'sh /home/ubuntu/smile/script/api_server.
sh',
                                execTimeout: 120000,
                                flatten: false,
                                makeEmptyDirs: false,
                                noDefaultExcludes: false,
                                patternSeparator: '[, ]+',
                                remoteDirectory: '',
                                remoteDirectorySDF: false,
                                removePrefix: 'Smile_Server/api-module/build/libs',
                                sourceFiles: 'Smile_Server/api-module/build/libs/*.ja
r')],
                                usePromotionTimestamp: false,
                                useWorkspaceInPromotion: false,
                                verbose: true)])
                                sshPublisher(publishers: [
                    sshPublisherDesc(
                        configName: 'jenkins_ssafy',
                        transfers: [
                            sshTransfer(
                                cleanRemote: false,
                                excludes: '',
                                execCommand: 'sh /home/ubuntu/smile/script/batch_serve
r.sh',
                                execTimeout: 120000,
                                flatten: false,
                                makeEmptyDirs: false,
                                noDefaultExcludes: false,
                                patternSeparator: '[, ]+',
                                remoteDirectory: '',
                                remoteDirectorySDF: false,
                                removePrefix: 'Smile_Server/batch-module/build/libs',
```

```
                                sourceFiles: 'Smile_Server/batch-module/build/libs/*.j
ar')],
                                usePromotionTimestamp: false,
                                useWorkspaceInPromotion: false,
                                verbose: true)])
            }
        }
    }

}
```

```
echo 'server open!!'

# 실행되고 있는 파일 중지시킴
echo 'CHECK PID'
CURRENT_PID=$(ps -ef |
        grep java | grep smile |
        awk '{print $2}')
        echo "$CURRENT_PID"
if [ -z ${CURRENT_PID} ]; then
        echo "NO EXIT CAUSE NO PROCESSING"
else
        echo "> sudo kill -9 $CURRENT_PID"
        sudo kill -9 $CURRENT_PID
        sleep 10
fi

# 백그라운드로 실행
nohup sudo java -jar -DServer.port=8888 -Dspring.profiles.active=real -Duser.timezone
="Asia/Seoul" /home/ubuntu/smile/ssafy-api-module-0.0.1.jar >> /home/ubuntu/smile/log
s/smile.log &
```

```
echo "batch start!!!"

nohup sudo java -jar -DServer.port=8899 -Dspring.profiles.active=real -Duser.timezone
="Asia/Seoul" /home/ubuntu/smile/ssafy-batch-module-0.0.1.jar >> /home/ubuntu/smile/lo
gs/smile_batch.log &
```

Flask 실행

- SFTP를 사용해서 윈도우 환경의 파일을 Ubuntu 폴더로 옮김

```
$nohup python3 smile/python/runserver.py >> /home/ubuntu/smile/logs/python.log &
```

DB 설치

MySQL

```
# download (default version: latest)
$ docker pull mysql
# docker 이미지 확인
$ docker images
# Mysql 컨테이너 생성 및 실행
$ docker run --name mysql-container -e MYSQL_ROOT_PASSWORD={password} -d -p 3306:3306
 mysql
# 실행중인 컨테이너 확인
$ docker ps -a
# mysql 실행
$ docker start mysql-container
# mysql 컨테이너 접속
$ docker exec -it mysql-container bash
  bash# mysql -u root -p
  Enter password:
  # mysql 접속
  mysql>
```

Google Cloud SDK 환경 설정

```
$ export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
$ echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" | sudo tee -a /
etc/apt/sources.list.d/google-cloud-sdk.list
$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
$ sudo apt-get update && sudo apt-get install google-cloud-sdk
$ gcloud init
# 1. url로 방문해서 로그인
# 2. 적용된 프로젝트 선택
# 3. Region and zone 선택 시 asia-northeast3가 서울

# 적용되면 access token 확인가능
$ gcloud auth print-access-token
```

- 프로젝트에서 발급받은 서비스 키를 EC2 서버에 옮긴 후 `/root/.config/gcloud/application_default_credentials.json`으로 변경

4. Nginx

설치

```
# 설치
sudo apt-get install nginx
```

```
# 설치 확인 및 버전 확인
nginx -v
```

```
server{
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name i8d102.p.ssafy.io www.i8d102.p.ssafy.io;

        # 포트없이 도메인으로 들어와도 8888번 포트로 리다이렉트
        location / {
                proxy_pass http://localhost:8888;
        }
}
```

Certbot

```
# 설치
sudo apt-get install letsencrypt
# nginx 중단
sudo systemctl stop nginx
# 인증서 발급
sudo letsencrypt certonly --standalone -d [도메인]
```

```
# sites-enabled로 복사
sudo ln -s /etc/nginx/sites-available/[도메인] /etc/nginx/sites-enabled/[도메인]
# 다음 명령어에서 successful이 뜨면 nginx를 실행할 수 있다.
sudo nginx -t
# nginx 재시작
sudo systemctl restart nginx
```