포팅 매뉴얼

서버 버전 설정

백엔드

- JAVA - 17
- Spring Boot - 3.2.3
- Spring Data JPA - 3.2.3
- Spring Security - 6.2.2
- jjwt - 0.12.3
- MySQL - 8.3.0
- Redis - 7.2.4

프론트엔드

- react: 18.2.0
- react-dom: 18.2.0
- react-native: 0.73.6
- expo: ~50.0.14
- axios: ^1.6.7
- nativewind: ^2.0.11
- react-navigation/bottom-tabs: ^6.5.18
- react-navigation/native: ^6.1.15
- react-navigation/native-stack: ^6.9.24
- react-native-async-storage/async-storage: 1.21.0

인프라

- Ubuntu - 22.04.3 LTS
- Jenkins - 2.448
- Docker - 25.0.5
- Nginx - 1.18.0

포트 설정

- Jenkins : 8080 → 8080

- Backend : 8081 → 8080
- MYDATA : 8082 → 8080
- BANK : 8083 → 8080
- CARD : 8084 → 8080
- CA : 8085 → 8080
- MySQL : 3307 → 3306
- Redis : 6379 → 6379

Ubuntu

EC2 접속

```
ssh -i J10A501T.pem ubuntu@j10a501.p.ssafy.io
```

서버 기본 세팅

```
sudo timedatectl set-timezone Asia/Seoul

sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g' /e
tc/apt/sources.list

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

Docker

패키지 설치

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docke

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
# 최신 버전을 설치 후 작동
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plug
```

Jenkins

Jenkins 이미지

```
docker pull jenkins/jenkins:jdk17
```

Jenkins 컨테이너 실행

```
docker run -d --restart always --env JENKINS_OPTS=--httpPort=8080 -v /etc/l
ocaltime:/etc/localtime:ro -e TZ=Asia/Seoul -p 8080:8080 -v /jenkins:/var/j
enkins_home -v /var/run/docker.sock:/var/run/docker.sock -v /usr/local/bin/
docker-compose:/usr/local/bin/docker-compose --name jenkins -u root jenkin
s/jenkins:jdk17

sudo docker restart jenkins
```

Jenkins 환경설정 (플러그인 미러서버 변경)

```
# Jenkins 컨테이너 종료
sudo docker stop jenkins

# Jenkins 데이터가 있는 디렉토리에 update-center-rootCAs 하위 디렉토리 생성
sudo mkdir /jenkins/update-center-rootCAs

# CA 파일 다운로드
sudo wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/u
pdate-center.crt -O /jenkins/update-center-rootCAs/update-center.crt

# Jenkins 플러그인 다운로드 시 미러사이트로 대체될 수 있도록 설정
sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.gi
thubusercontent.com/lework/jenkins-update-center/master/updates/tencent/upd
ate-center.json#' /jenkins/hudson.model.UpdateCenter.xml

# Jenkins 컨테이너 재시작
sudo docker restart jenkins
```

Jenkins 내부에 Docker 패키지 설치

```
# Jenkins 컨테이너 접속
docker exec -it jenkins /bin/bash

# Docker Repository 등록 및 docker-ce 패키지 설치
```

```
apt-get update && apt-get -y install apt-transport-https ca-certificates cu
rl gnupg2 software-properties-common && curl -fsSL https://download.docker.
com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tm
p/dkey && add-apt-repository "deb [arch=amd64] https://download.docker.com/
linux/$(. /etc/os-release; echo "$ID") $(lsb_release -cs) stable" && apt-ge
t update && apt-get -y install docker-ce
```

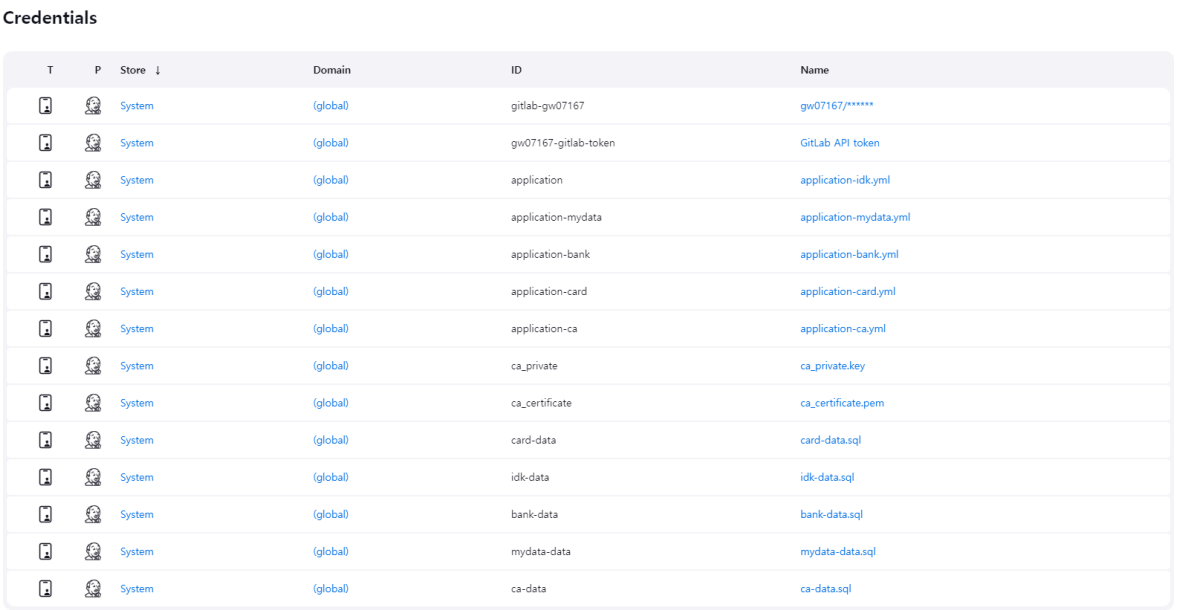
Docker Jenkins에서 Host Docker 접근권한 부여

```
groupadd -f docker

usermod -aG docker jenkins

chown root:docker /var/run/docker.sock
```

Jenkins 파이프라인 플러그인 설치

- Gitlab
- Docker
- GitLab Authentication
- Generic WebHook Trigger
- ssh

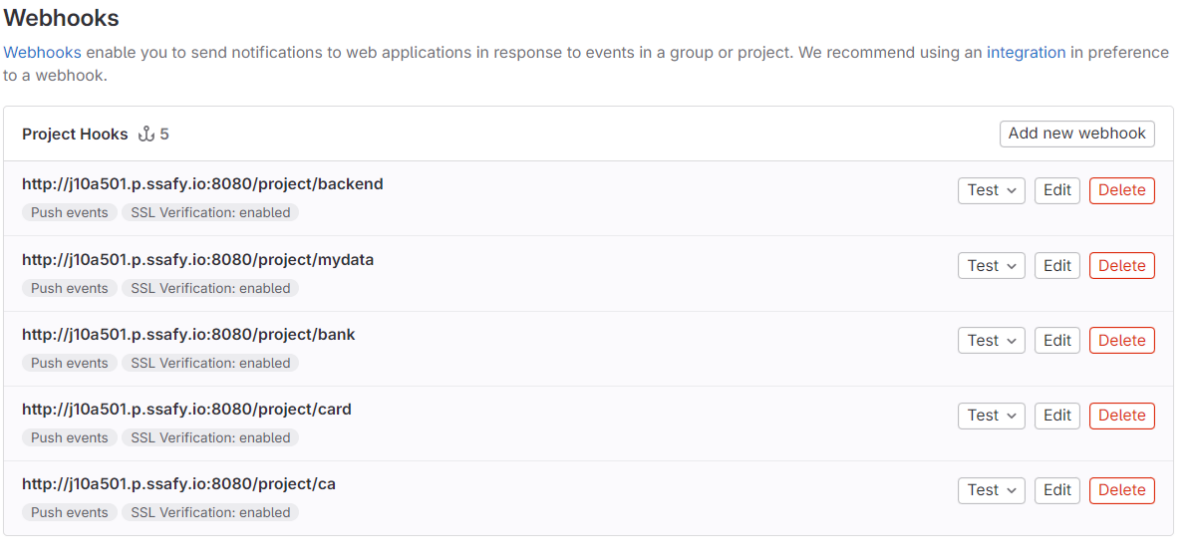
환경 변수 및 Credential 설정



Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab-gw07167	gw07167/******
		System	(global)	gw07167-gitlab-token	GitLab API token
		System	(global)	application	application-idk.yml
		System	(global)	application-mydata	application-mydata.yml
		System	(global)	application-bank	application-bank.yml
		System	(global)	application-card	application-card.yml
		System	(global)	application-ca	application-ca.yml
		System	(global)	ca_private	ca_private.key
		System	(global)	ca_certificate	ca_certificate.pem
		System	(global)	card-data	card-data.sql
		System	(global)	idk-data	idk-data.sql
		System	(global)	bank-data	bank-data.sql
		System	(global)	mydata-data	mydata-data.sql
		System	(global)	ca-data	ca-data.sql

Gitlab Connection, Webhook 설정



Tools 설정

- JDK - 17
- Gradle - 8.5
- Docker - latest

nginx 설정

nginx 설치 (HostOS)

```
sudo apt-get -y install nginx
```

Certbot, SSL 인증서 발급

```
# certbot 다운로드
sudo snap install --classic certbot

# repository에 certbot 설치
sudo apt-add-repository -r ppa:certbot/certbot

# python-certbot-nginx 설치
sudo apt-get -y install python3-certbot-nginx

# SSL 인증서 발급
sudo certbot --nginx -d j10a501.p.ssafy.io
```

nginx 리버스 프록시 설정

- default.conf

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;
            server_name j10a501.p.ssafy.io; # managed by Certbot


        location / {
                try_files $uri $uri/ =404;
        }

        location /api {
                proxy_pass http://localhost:8081;
                proxy_set_header Connection '';
                                proxy_http_version 1.1;
        }

        location /api/mydata {
                proxy_pass http://localhost:8082;
        }

        location /api/bank {
                proxy_pass http://localhost:8083;
        }

        location /api/card {
                proxy_pass http://localhost:8084;
        }

        location /api/ca {
                proxy_pass http://localhost:8085;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j10a501.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j10a501.p.ssafy.io/privkey.pe
m;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = j10a501.p.ssafy.io) {
```

```
        return 301 https://$host$request_uri;
    }

        listen 80 ;
        listen [::]:80 ;
    server_name j10a501.p.ssafy.io;
    return 404; # managed by Certbot


}
```

MySQL

MySQL 이미지 받기, 컨테이너 실행

```
docker pull mysql:latest

docker run -d --name mysql-container -p 3307:3306 -v mysql-volume:/var/lib/
mysql -e MYSQL_ROOT_PASSWORD=idka501 mysql:latest
```

MySQL 접속

```
docker exec -it mysql-container bash
```

Redis

Redis 이미지 받기, 컨테이너 실행

```
sudo docker pull redis

docker run -d --restart always -p 6379:6379 --name redis redis
```

Redis 접속

```
docker exec -it redis redis-cli
```

Redis 비밀번호 설정

```
config set requirepass idka501
```

Jenkins Pipeline Item - Backend

Pipeline Script

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }

    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'BE', url: 'https://lab.ssafy.com/s10-fintech-finance-sub2/S10P22A501.git',
                credentialsId: 'gitlab-gw07167'
            }
        }
        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application', variable: 'secretFile')]) {
                    script {
                        sh 'mkdir -p backend/idk/src/main/resources/'
                        sh 'cp $secretFile backend/idk/src/main/resources/application.yml'
                    }
                }
            }
        }
        stage('Apply idk-data files') {
            steps {
                withCredentials([file(credentialsId: 'idk-data', variable: 'secretFile')]) {
                    script {
                        sh 'mkdir -p backend/idk/src/main/resources/db'
                        sh 'cp $secretFile backend/idk/src/main/resources/db/data.sql'
                    }
                }
            }
```

```
        }
        stage('BE-Build') {
            steps {
                dir("./backend/idk") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect idk_server >/dev/null 2>&1;
then
                        echo "container exists locally"
                        docker stop idk_server
                        docker rm idk_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect server >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./backend/idk') {
                    sh'''
                        echo [BE] Build Docker Image!
                        docker build -t server .
                        echo [BE] Run Docker Container!
                        docker run -dp 8081:8081 -e TZ=Asia/Seoul --name id
k_server server
                    '''
                }
            }
        }
    }
}
```

application.yml

```
server:
  port: 8081


# MySQL8
spring:
  datasource:
    url: jdbc:mysql://j10a501.p.ssafy.io:3307/idk?useSSL=false&characterEnc
oding=UTF-8&serverTimezone=UTC
    username: idkbank
    password: idka501
    driver-class-name: com.mysql.cj.jdbc.Driver

  # redis
  data:
    redis:
      host: j10a501.p.ssafy.io
      port: 6379
      password: idka501

  # SQL
  jpa:
    #show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    defer-datasource-initialization: true

  # JWT
  jwt:
    header: Authorization
    secret: vmfhaltmskddsrhsdrhsdrhsdjsrtjkjtsrkjrtsksrksryktsrtykxfcghdsxc
fgrduty
    access-token-expiration-millis: 1800000
    refresh-token-expiration-millis: 18000000
  sms:
    apiKey: NCSIML602UKFSVIT
    apiSecret: IQOT5R6Z1MBYH2AKFZKV0YDFRNZKBN7U
    from: 01020804512
  mydata:
    unique-code: dsgsgsgh43262
    token: skjfkefhsjkfhskfe12519510518958
    agree-terms: 동의합니다.
    ca-url: https://j10a501.p.ssafy.io
    mydata-url: https://j10a501.p.ssafy.io
    bank-url: https://j10a501.p.ssafy.io
```

```
    card-url: https://j10a501.p.ssafy.io
  idk:
    org-code: 501
    client-id: idkBankClientId
    client-secret: idkBankSecret
  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql

# logging:
  # level:
    # org.hibernate.SQL: DEBUG
    # org.hibernate.type.descriptor.sql.BasicBinder: TRACE
```

data.sql

```
INSERT IGNORE INTO organization (org_name, org_code, org_type)
VALUES ('IDK은행', '501', 'BANK'),
       ('IBK기업은행', '003', 'BANK'),
       ('KB국민은행', '004', 'BANK'),
       ('NH농협은행', '011', 'BANK'),
       ('우리은행', '020', 'BANK'),
       ('하나은행', '081', 'BANK'),
       ('신한은행', '088', 'BANK'),
       ('케이뱅크', '089', 'BANK'),
       ('카카오뱅크', '090', 'BANK'),
       ('우리카드', '041', 'CARD'),
       ('하나카드', '044', 'CARD'),
       ('삼성카드', '365', 'CARD'),
       ('신한카드', '366', 'CARD'),
       ('현대카드', '367', 'CARD'),
       ('NH농협카드', '371', 'CARD'),
       ('KB국민카드', '381', 'CARD');

INSERT IGNORE INTO item (item_id, category, name, price, shop)
VALUES  (1,1,'프리미엄 퓨어스펙 블랙라벨 고당도 오렌지 2kg',19800,'온브릭스'),
        (2,1,'순천참한우 1등급',119000,'바담'),
        (3,1,'오쏘몰 이뮨 멀티비타민',38000,'동아제약'),
        (4,1,'페레로로쉐 하트 초콜릿 (8개입)',9900,'페레로로쉐'),
        (5,1,'구운란 중란 30알x2판',19900,'자연애'),
        (6,2,'갤럭시24',1269900,'삼성전자'),
        (7,2,'인스탁스 미니11 패키지',119000,'인스탁스'),
        (8,2,'\"카페 in 홈\" 가정용 커피머신',79000,'빈크루즈'),
        (9,2,'LED 무드등 블루투스 스피커',14900,'로이체'),
        (10,2,'코닥 미니샷3 C300R',180000,'코닥'),
        (11,3,'레저렉션 아로마틱 핸드 밤',39000,'이솝'),
```

```
        (12,3,'오드뚜왈렛 50ml',176000,'딥티크'),
        (13,3,'NEW 벨벳틴트 세트',49000,'입생로랑'),
        (14,3,'트리트먼트 50ml',37000,'모로칸오일'),
        (15,3,'루쥬 코코 플래쉬',59000,'샤넬'),
        (16,4,'C 로고 후디',79000,'커버낫'),
        (17,4,'루키 언스트럭쳐 볼캡 NY',36000,'MLB'),
        (18,4,'후드 니트 바라클라바',39900,'LAP'),
        (19,4,'플레이 하트 스니커즈',215000,'꼼데가르송'),
        (20,4,'시그니처 슬림핏 스트레치 CK 반팔티',44200,'캘빈클라인'),
        (21,5,'마그네틱 클리어 케이스',77500,'케이스티파이'),
        (22,5,'SS 블랙 탱크 텀블러 503ml',35000,'스타벅스'),
        (23,5,'춘식이 꿀잠 팔쿠션',9900,'카카오프렌즈'),
        (24,5,'홀리몰리 데일리 머그잔',13800,'엘리건트테이블'),
        (25,5,'돈의 속성',17800,'교보문고');
```

Jenkins Pipeline Item - MYDATA

Pipeline Script

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }
    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'MYDATA', url: 'https://lab.ssafy.com/s10-finte
ch-finance-sub2/S10P22A501.git',
                credentialsId: 'gitlab-gw07167'
            }
        }
        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-mydata',
variable: 'secretFile')]) {
                    script {
```

```
                        sh 'mkdir -p mydata/src/main/resources/'
                        sh 'cp $secretFile mydata/src/main/resources/applic
ation.yml'
                    }
                }
            }
        }
        stage('Apply mydata-data files') {
            steps {
                withCredentials([file(credentialsId: 'mydata-data', variabl
e: 'secretFile')]) {
                    script {
                        sh 'mkdir -p mydata/main/resources/db'
                        sh 'cp $secretFile mydata/main/resources/db/data.sq
l'
                    }
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./mydata") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect mydata_server >/dev/null 2>
&1; then
                        echo "container exists locally"
                        docker stop mydata_server
                        docker rm mydata_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect mydata_server >/dev/null 2>&1;
then
                        echo "Image exists locally"
                        docker rmi mydata_server
                    else
                        echo "Image does not exist locally"
                    fi
```

```
                '''
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./mydata') {
                    sh'''
                        echo [MYDATA] Build Docker Image!
                        docker build -t mydata_server .
                        echo [MYDATA] Run Docker Container!
                        docker run -dp 8082:8082 -e TZ=Asia/Seoul --name my
data_server mydata_server
                    '''
                }
            }
        }
    }
}
```

application.yml

```
server:
  port: 8082


# MySQL8
spring:
  datasource:
    url: jdbc:mysql://j10a501.p.ssafy.io:3307/mydata?useSSL=false&character
Encoding=UTF-8&serverTimezone=UTC
    username: idkbank
    password: idka501
    driver-class-name: com.mysql.cj.jdbc.Driver

  # SQL
  jpa:
    show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    defer-datasource-initialization: true

  # JWT
  jwt:
    header: Authorization
    secret: aiovnzoivnlemxzsaieghsiaehqwsdhgjkladhuvbnxcmzxzviijezsrksrykts
rtykxfcghdsxcfgrduty
    access-token-expiration-millis: 15552000000
```

```yaml
  mydata:
    unique-code: dsgsgsgh43262
    token: skjfkefhsjkfhskfe12519510518958
    agree-terms: 동의합니다.
    ca-url: https://j10a501.p.ssafy.io
    mydata-url: https://j10a501.p.ssafy.io
    bank-url: https://j10a501.p.ssafy.io
    card-url: https://j10a501.p.ssafy.io
  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql

logging:
  level:
    org.hibernate.SQL: DEBUG
    org.hibernate.type.descriptor.sql.BasicBinder: TRACE
```

data.sql

```sql
INSERT IGNORE INTO organization (org_name, org_code, org_type, client_id, c
lient_secret)
VALUES ('IDK은행', '501', 'BANK', 'idkBankClientId', 'idkBankSecret'),
            ('IBK기업은행', '003', 'BANK', 'ibkClientId', 'ibkSecret'),
            ('KB국민은행', '004', 'BANK', 'kbClientId', 'kbSecret'),
            ('NH농협은행', '011', 'BANK', 'nhClientId', 'nhSecret'),
            ('우리은행', '020', 'BANK', 'wooriClientId', 'wooriSecret'),
            ('하나은행', '081', 'BANK', 'hanaClientId', 'hanaSecret'),
            ('신한은행', '088', 'BANK', 'shinhanClientId', 'shinhanSecret'),
            ('케이뱅크', '089', 'BANK', 'kBankClientId', 'kBankSecret'),
            ('카카오뱅크', '090', 'BANK', 'kakaoBankClientId', 'kakaoBankSecr
et'),
            ('우리카드', '041', 'CARD', 'wooriCardClientId', 'wooriCardSecre
t'),
            ('하나카드', '044', 'CARD', 'hanaCardClientId', 'hanaCardSecre
t'),
            ('삼성카드', '365', 'CARD', 'samsungCardClientId', 'samsungCardS
ecret'),
            ('신한카드', '366', 'CARD', 'shinhanCardClientId', 'shinhanCardS
ecret'),
            ('현대카드', '367', 'CARD', 'hyundaiCardClientId', 'hyundaiCardS
ecret'),
            ('NH농협카드', '371', 'CARD', 'nhCardClientId', 'nhCardSecret'),
            ('KB국민카드', '381', 'CARD', 'kbCardClientId', 'kbCardSecret');

INSERT IGNORE INTO organization_permissions (org_id, permission)
VALUES (1, 'MY_DATA_RECEIVER'),
```

```
            (2, 'DATA_PROVIDER'),
            (3, 'DATA_PROVIDER'),
            (4, 'DATA_PROVIDER'),
            (5, 'DATA_PROVIDER'),
            (6, 'DATA_PROVIDER'),
            (7, 'DATA_PROVIDER'),
            (8, 'DATA_PROVIDER'),
            (9, 'DATA_PROVIDER'),
            (10, 'DATA_PROVIDER'),
            (11, 'DATA_PROVIDER'),
            (12, 'DATA_PROVIDER'),
            (13, 'DATA_PROVIDER'),
            (14, 'DATA_PROVIDER'),
            (15, 'DATA_PROVIDER'),
            (16, 'DATA_PROVIDER');
```

Jenkins Pipeline Item - BANK

Pipeline Script

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }
    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'BANK', url: 'https://lab.ssafy.com/s10-fintech
-finance-sub2/S10P22A501.git',
                credentialsId: 'gitlab-gw07167'
            }
        }
        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-bank', va
riable: 'secretFile')]) {
```

```
                    script {
                        sh 'mkdir -p bank/src/main/resources/'
                        sh 'cp $secretFile bank/src/main/resources/applicat
ion.yml'
                    }
                }
            }
        }
        stage('Apply bank-data files') {
            steps {
                withCredentials([file(credentialsId: 'bank-data', variable:
'secretFile')]) {
                    script {
                        sh 'mkdir -p bank/src/main/resources/db'
                        sh 'cp $secretFile bank/src/main/resources/db/data.
sql'
                    }
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./bank") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect bank_server >/dev/null 2>&
1; then
                        echo "container exists locally"
                        docker stop bank_server
                        docker rm bank_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect bank_server >/dev/null 2>&1; th
en
                        echo "Image exists locally"
                        docker rmi bank_server
                    else
                        echo "Image does not exist locally"
```

```
                    fi
                '''
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./bank') {
                    sh'''
                        echo [BANK] Build Docker Image!
                        docker build -t bank_server .
                        echo [BANK] Run Docker Container!
                        docker run -dp 8083:8083 -e TZ=Asia/Seoul --name ba
nk_server bank_server
                    '''
                }
            }
        }
    }
}
```

application.yml

```
server:
  port: 8083


# MySQL8
spring:
  datasource:
    url: jdbc:mysql://j10a501.p.ssafy.io:3307/bank?useSSL=false&characterEn
coding=UTF-8&serverTimezone=UTC
    username: idkbank
    password: idka501
    driver-class-name: com.mysql.cj.jdbc.Driver

  # SQL
  jpa:
    show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    defer-datasource-initialization: true

  # JWT
  jwt:
    header: Authorization
    secret: aiovnzoivnlemxzsaieghsiaehqwsdhgjkladhuvbnxcmzxzviijezsrksrykts
rtykxfcghdsxcfgrduty
```

```
    access-token-expiration-millis: 15552000000
  mydata:
    unique-code: dsgsgsgh43262
    token: skjfkefhsjkfhskfe12519510518958
    agree-terms: 동의합니다.
    ca-url: https://j10a501.p.ssafy.io
    mydata-url: https://j10a501.p.ssafy.io
    bank-url: https://j10a501.p.ssafy.io
    card-url: https://j10a501.p.ssafy.io

  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql

logging:
  level:
    org.hibernate.SQL: DEBUG
    org.hibernate.type.descriptor.sql.BasicBinder: TRACE
```

data.sql

```
INSERT IGNORE INTO organization (org_name, org_code, org_type)
VALUES ('IDK은행', '501', 'BANK'),
       ('IBK기업은행', '003', 'BANK'),
       ('KB국민은행', '004', 'BANK'),
       ('NH농협은행', '011', 'BANK'),
       ('우리은행', '020', 'BANK'),
       ('하나은행', '081', 'BANK'),
       ('신한은행', '088', 'BANK'),
       ('케이뱅크', '089', 'BANK'),
       ('카카오뱅크', '090', 'BANK'),
       ('우리카드', '041', 'CARD'),
       ('하나카드', '044', 'CARD'),
       ('삼성카드', '365', 'CARD'),
       ('신한카드', '366', 'CARD'),
       ('현대카드', '367', 'CARD'),
       ('NH농협카드', '371', 'CARD'),
       ('KB국민카드', '381', 'CARD');

-- Bank 서버 bank 테이블 더미 데이터 생성
INSERT IGNORE INTO bank (name, client_id, client_secret, org_code)
VALUES ('IDK은행', 'idkBankClientId', 'idkBankSecret', '501'),
           ('IBK기업은행', 'ibkClientId', 'ibkSecret', '003'),
           ('KB국민은행', 'kbClientId', 'kbSecret', '004'),
           ('NH농협은행', 'nhClientId', 'nhSecret', '011'),
           ('우리은행', 'wooriClientId', 'wooriSecret', '020'),
```

```
            ('하나은행', 'hanaClientId', 'hanaSecret', '081'),
            ('신한은행', 'shinhanClientId', 'shinhanSecret', '088'),
            ('케이뱅크', 'kBankClientId', 'kBankSecret', '089'),
            ('카카오뱅크', 'kakaoBankClientId', 'kakaoBankSecret', '090');

-- 회원 데이터 삽입
INSERT IGNORE INTO member (name, birth_date, phone_number, connection_infor
mation)
VALUES ('김성민', '9412051552674', '01012345678', 'abcdefghij0123456789'),
            ('이지원', '9403222345678', '01023456789', 'bcdefghija123456789
0'),
            ('박영호', '9305153456789', '01034567890', 'cdefghijab234567890
1'),
            ('최민지', '9207094567890', '01045678901', 'defghijabc345678901
2'),
            ('정민호', '9110015678901', '01056789012', 'efghijabcd456789012
3'),
            ('장하늘', '9011276789012', '01067890123', 'fghijabcde567890123
4'),
            ('신주연', '8903087890123', '01078901234', 'ghijabcdef678901234
5'),
            ('오진우', '8805238901234', '01089012345', 'hijabcdefg789012345
6'),
            ('강태희', '8707279012345', '01090123456', 'ijabcdefgh890123456
7'),
            ('나현지', '8609030123456', '01091234567', 'jabcdefghi901234567
8'),
            ('배민준', '8512181234567', '01092345678', '0123456789abcdefghi
j'),
            ('임주원', '8404012345678', '01093456789', '123456789abcdfghij
k'),
            ('최태연', '8305153456789', '01094567890', '23456789abcdefghi
j'),
            ('한서연', '8206214567890', '01095678901', '3456789abcdefghij
k'),
            ('고윤지', '8107195678901', '01096789012', '456789abcdefghijk
a'),
            ('김주원', '8008036789012', '01097890123', '56789abcdefghijka
b'),
            ('이민서', '7909247890123', '01098901234', '6789abcdefghijklmno
p'),
            ('장태언', '7810018901234', '01090012345', '789abcdefghijklmnop
q'),
            ('신서준', '7702179012345', '01091123456', '89abcdefghijklmnopq
r'),
            ('박서현', '7603030123456', '01092234567', '9abcdefghijklmnopqr
s'),
            ('최하윤', '7504211234567', '01093345678', 'abcdefghijklmnopqrs
```

```
t'),
            ('이승우', '7405302345678', '01094456789', 'bcdefghijklmnopqrst
u'),
            ('정가은', '7306163456789', '01095567890', 'cdefghijklmnopqrstu
v'),
            ('김세은', '7207034567890', '01096678901', 'defghijklmnopqrstuv
w'),
            ('최성민', '7108195678901', '01097789012', 'efghijklmnopqrstuvw
x'),
            ('한예준', '7009026789012', '01098890123', 'fghijklmnopqrstuvwx
y'),
            ('고주원', '6910147890123', '01099901234', 'ghijklmnopqrstuvwxy
z'),
            ('이서준', '6811018901234', '01010012345', 'hijklmnopqrstuvwxyza
b'),
            ('김주원', '6702159012345', '01011123456', 'ijklmnopqrstuvwxyzab
c'),
            ('최서준', '6603020123456', '01012234567', 'jklmnopqrstuvwxyzabc
d'),
            ('신예준', '6504211234567', '01013345678', 'klmnopqrstuvwxyzabcd
e'),
            ('박승우', '6405072345678', '01014456789', 'lmnopqrstuvwxyzabcde
f'),
            ('이다은', '6306083456789', '01015567890', 'mnopqrstuvwxyzabcdef
g'),
            ('김주원', '6207044567890', '01016678901', 'nopqrstuvwxyzabcdefg
h'),
            ('오예준', '6108185678901', '01017789012', 'opqrstuvwxyzabcdefgh
i'),
            ('최지안', '6009016789012', '01018890123', 'pqrstuvwxyzabcdefghi
j'),
            ('장승민', '5910197890123', '01019901234', 'qrstuvwxyzabcdefghij
k'),
            ('박준우', '5811068901234', '01020012345', 'rstuvwxyzabcdefghijk
l'),
            ('김주원', '5702219012345', '01021123456', 'stuvwxyzabcdefghijkl
m'),
            ('홍길동', '5603270123456', '01022234567', 'tuvwxyzabcdefghijklm
n');

INSERT INTO account (balance, account_number, member_id, bank_id)
VALUES (10000000, '5011010011', 1, 1),
            (10000000, '0033020011', 2, 2),
            (10000000, '0044031011', 3, 3),
            (10000000, '0111040011', 4, 4),
            (10000000, '0202050011', 5, 5),
            (10000000, '0811060011', 6, 6),
            (10000000, '0881070011', 7, 7),
```

```
            (10000000, '0891080011', 8, 8),
            (10000000, '0901090011', 9, 9),
            (10000000, '5011100012', 10, 1),
            (10000000, '0033200012', 11, 2),
            (10000000, '0044300012', 12, 3),
            (10000000, '0111400012', 13, 4),
            (10000000, '0202500012', 14, 5),
            (10000000, '0811600012', 15, 6),
            (10000000, '0881700012', 16, 7),
            (10000000, '0891800012', 17, 8),
            (10000000, '0901900012', 18, 9),
            (10000000, '5011001013', 19, 1),
            (10000000, '0033002013', 20, 2),
            (10000000, '0044003013', 21, 3),
            (10000000, '0111004013', 22, 4),
            (10000000, '0202005013', 23, 5),
            (10000000, '0811006013', 24, 6),
            (10000000, '0881007013', 25, 7),
            (10000000, '0891008013', 26, 8),
            (10000000, '0901009013', 27, 9),
            (10000000, '5011011014', 28, 1),
            (10000000, '0033012014', 29, 2),
            (10000000, '0044013014', 30, 3),
            (10000000, '0111014014', 31, 4),
            (10000000, '0201015011', 32, 5),
            (10000000, '0811016011', 33, 6),
            (10000000, '0881017011', 34, 7),
            (10000000, '0891018011', 35, 8),
            (10000000, '0901019011', 36, 9),
            (10000000, '5011021011', 37, 1),
            (10000000, '0033022011', 38, 2),
            (10000000, '0044023011', 39, 3),
            (10000000, '0111024011', 40, 4);
```

Jenkins Pipeline Item - CARD

Pipeline Script

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }
    stages {
```

```
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'CARD', url: 'https://lab.ssafy.com/s10-fintech
-finance-sub2/S10P22A501.git',
                credentialsId: 'gitlab-gw07167'
            }
        }
        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-card', va
riable: 'secretFile')]) {
                    script {
                        sh 'mkdir -p card/src/main/resources/'
                        sh 'cp $secretFile card/src/main/resources/applicat
ion.yml'
                    }
                }
            }
        }
        stage('Apply card-data files') {
            steps {
                withCredentials([file(credentialsId: 'card-data', variable:
'secretFile')]) {
                    script {
                        sh 'mkdir -p card/src/main/resources/db'
                        sh 'cp $secretFile card/src/main/resources/db/data.
sql'
                    }
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./card") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
```

```
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect card_server >/dev/null 2>&
1; then
                        echo "container exists locally"
                        docker stop card_server
                        docker rm card_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect card_server >/dev/null 2>&1; th
en
                        echo "Image exists locally"
                        docker rmi card_server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./card') {
                    sh'''
                        echo [CARD] Build Docker Image!
                        docker build -t card_server .
                        echo [CARD] Run Docker Container!
                        docker run -dp 8084:8084 -e TZ=Asia/Seoul --name ca
rd_server card_server
                    '''
                }
            }
        }
    }
}
```

application.yml

```
server:
  port: 8084


# MySQL8
spring:
  datasource:
    url: jdbc:mysql://j10a501.p.ssafy.io:3307/card?useSSL=false&characterEn
```

```
coding=UTF-8&serverTimezone=UTC
    username: idkbank
    password: idka501
    driver-class-name: com.mysql.cj.jdbc.Driver

    # SQL
  jpa:
    show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    defer-datasource-initialization: true

  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql

logging:
  level:
    org.hibernate.SQL: DEBUG
    org.hibernate.type.descriptor.sql.BasicBinder: TRACE

# JWT
jwt:
  header: Authorization
  secret: asehywyeszuyuyzsaieghsiaescoiaewhratnaesjstudrhsdfdsgsuvbnxcmzxzv
iijezsrksryktsrtykxfcghdsxcfgrduty
  access-token-expiration-millis: 15552000000
mydata:
  unique-code: dsgsgsgh43262
  token: skjfkefhsjkfhskfe12519510518958
  agree-terms: 동의합니다.
  ca-url: https://j10a501.p.ssafy.io
  mydata-url: https://j10a501.p.ssafy.io
  bank-url: https://j10a501.p.ssafy.io
  card-url: https://j10a501.p.ssafy.io
```

data.sql

```
INSERT IGNORE INTO member (name, birth_date, phone_number, connection_infor
mation)
VALUES ('김덕배', '0101012345678', '01012345678', '12345678910');

INSERT IGNORE INTO organization (org_name, org_code, org_type, access_toke
n)
VALUES ('IBK기업은행', '003', '은행', NULL),
```

```
        ('KB국민은행', '004', '은행', NULL),
        ('NH농협은행', '011', '은행', NULL),
        ('우리은행', '020', '은행', NULL),
        ('하나은행', '081', '은행', NULL),
        ('신한은행', '088', '은행', NULL),
        ('케이뱅크', '089', '은행', NULL),
        ('카카오뱅크', '090', '은행', NULL),
        ('우리카드', '041', '카드', NULL),
        ('하나카드', '044', '카드', NULL),
        ('삼성카드', '365', '카드', NULL),
        ('신한카드', '366', '카드', NULL),
        ('현대카드', '367', '카드', NULL),
        ('NH농협카드', '371', '카드', NULL),
        ('KB국민카드', '381', '카드', NULL);

INSERT IGNORE INTO company (name, org_code, client_id, client_secret)
VALUES ('우리카드', '041', 'wooriCardClientId', 'wooriCardSecret'),
            ('하나카드', '044', 'hanaCardClientId', 'hanaCardSecret'),
            ('삼성카드', '365', 'samsungCardClientId', 'samsungCardSecret'),
            ('신한카드', '366', 'shinhanCardClientId', 'shinhanCardSecret'),
            ('현대카드', '367', 'hyundaiCardClientId', 'hyundaiCardSecret'),
            ('NH농협카드', '371', 'nhCardClientId', 'nhCardSecret'),
            ('KB국민카드', '381', 'kbCardClientId', 'kbCardSecret');
```

Jenkins Pipeline Item - CA

Pipeline Script

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
        jdk 'jdk-17'
        dockerTool 'Docker'
    }
    stages {
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }
        stage('Git Clone') {
            steps {
                git branch: 'CA', url: 'https://lab.ssafy.com/s10-fintech-f
```

```
inance-sub2/S10P22A501.git',
                credentialsId: 'gitlab-gw07167'
            }
        }
        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-ca', vari
able: 'secretFile')]) {
                    script {
                        sh 'mkdir -p ca/src/main/resources/'
                        sh 'cp $secretFile ca/src/main/resources/applicatio
n.yml'
                    }
                }
            }
        }
        stage('Apply ca_certificate.pem files') {
            steps {
                withCredentials([file(credentialsId: 'ca_certificate', vari
able: 'secretFile')]) {
                    script {
                        sh 'mkdir -p ca/src/main/resources/certs/'
                        sh 'cp $secretFile ca/src/main/resources/certs/ca_c
ertificate.pem'
                    }
                }
            }
        }
        stage('Apply ca_private.key files') {
            steps {
                withCredentials([file(credentialsId: 'ca_private', variabl
e: 'secretFile')]) {
                    script {
                        sh 'mkdir -p ca/src/main/resources/certs/'
                        sh 'cp $secretFile ca/src/main/resources/certs/ca_p
rivate.key'
                    }
                }
            }
        }
        stage('Apply ca-data files') {
            steps {
                withCredentials([file(credentialsId: 'ca-data', variable:
'secretFile')]) {
                    script {
                        sh 'mkdir -p ca/src/main/resources/db'
                        sh 'cp $secretFile ca/src/main/resources/db/data.sq
l'
```

```
                    }
                }
            }
        }
        stage('BE-Build') {
            steps {
                dir("./ca") {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }
        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect ca_server >/dev/null 2>&1;
then
                        echo "container exists locally"
                        docker stop ca_server
                        docker rm ca_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect ca_server >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi ca_server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }
        stage('Build and Deploy Docker') {
            steps {
                dir('./ca') {
                    sh'''
                        echo [CA] Build Docker Image!
                        docker build -t ca_server .
                        echo [CA] Run Docker Container!
                        docker run -dp 8085:8085 -e TZ=Asia/Seoul --name ca
_server ca_server
                    '''
                }
            }
        }
```

```
    }
}
```

application.yml

```
server:
  port: 8085

# MySQL8
spring:
  datasource:
    url: jdbc:mysql://j10a501.p.ssafy.io:3307/ca?useSSL=false&characterEnco
ding=UTF-8&serverTimezone=UTC
    username: idkbank
    password: idka501
    driver-class-name: com.mysql.cj.jdbc.Driver

  # SQL
  jpa:
    show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    defer-datasource-initialization: true

  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql

logging:
  level:
    org.hibernate.SQL: DEBUG
    org.hibernate.type.descriptor.sql.BasicBinder: TRACE
```

data.sql

```
INSERT IGNORE INTO organization (org_name, org_code, org_type, access_toke
n)
VALUES ('IDK은행', '501', 'BANK', 'idk_token'),
       ('IBK기업은행', '003', 'BANK', 'ibk_token'),
       ('KB국민은행', '004', 'BANK', 'kb_token'),
       ('NH농협은행', '011', 'BANK', 'nk_token'),
       ('우리은행', '020', 'BANK', 'woori_token'),
       ('하나은행', '081', 'BANK', 'hana_token'),
       ('신한은행', '088', 'BANK', 'sinhan_token'),
```

```
        ('케이뱅크', '089', 'BANK', 'kbank_token'),
        ('카카오뱅크', '090', 'BANK', 'kakao_token'),
        ('우리카드', '041', 'CARD', 'wooricard_token'),
        ('하나카드', '044', 'CARD', 'hanacard_token'),
        ('삼성카드', '365', 'CARD', 'samsung_token'),
        ('신한카드', '366', 'CARD', 'sinhancard_token'),
        ('현대카드', '367', 'CARD', 'hyundae_token'),
        ('NH농협카드', '371', 'CARD', 'nhcard_token'),
        ('KB국민카드', '381', 'CARD', 'kbcard_token');
```

ca_certificate.pem

```
-----BEGIN CERTIFICATE-----
MIID4zCCAsugAwIBAgIUCp9/v7rDjFxZESnII1Yl8/0ueXAwDQYJKoZIhvcNAQEL
BQAwgYAxCzAJBgNVBAYTAktSMQ4wDAYDVQQIDAVTZW91bDETMBEGA1UEBwwKR2Fu
Z25hbS1ndTEOMAwGA1UECgwFbG9jYWwxDjAMBgNVBAsMBWxvY2FsMQ4wDAYDVQQD
DAVsb2NhbDEcMBoGCSqGSIb3DQEJARYNdGVzdEB0ZXN0LmNvbTAeFw0yNDAzMjQx
NjA4MjZaFw0yNTAzMjQxNjA4MjZaMIGAMQswCQYDVQQGEwJLUjEOMAwGA1UECAwF
U2VvdWwxEzARBgNVBAcMCkdhbmduYW0tZ3UxDjAMBgNVBAoMBWxvY2FsMQ4wDAYD
VQQLDAVsb2NhbDEOMAwGA1UEAwwFbG9jYWwxHDAaBgkqhkiG9w0BCQEWDXRlc3RA
dGVzdC5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC4udtmjXIL
7nVNsPkyhIAH31btnyLfF/8erh+wByU18fuPq+OcoAYKWOsM1VGTm1kWtNoYwwjc
n/NzMjxc+5NA9SQYU9zQM/ZU0Bh0gBXLmgxqm2JthGdWkYtaHGqnlHHwqEo8WnZs
ar6DQ/BPGLB/W4H/3491mFFIpKRoGUbFTn9xF9Ke0ojpa/Z1ONxdTFaMln+467Pj
F7tGAH+AM18nDHGdhlGyILB7TUyffUGETvGFO8WT0rLZbFF6JCEbnmsc7c3Oxhsb
J2xzglxldUSl+jNvRXXGrtoYjrvu1bnKdwSMoViycBUYpUKk1Pm0/3ctvaHNeDg0
tfhVtWOWUIcnAgMBAAGjUzBRMB0GA1UdDgQWBBSxyMvsf63CyDhVojY5pu0hayrG
+jAfBgNVHSMEGDAWgBSxyMvsf63CyDhVojY5pu0hayrG+jAPBgNVHRMBAf8EBTAD
AQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCh3Td15ktlL3SaoCifD7JcjX7KZUTFGhrG
eXwNxS5TAnJ/yGdxkf5GESJgUL2t7QEme1R+O65Jmp67/n2o9p6760L2nkv0JXqy
d2LeWILUVDTnxed5vz0XiZQvU3KZmDLbE0U19NwjRZk/bj1UGrMhrDcRxSj3VYoR
UN3LaExFwHMvuzjMetI5VF8K5Jfcxty5seNRBi+JImLbTrxXyujwO1EImF935CRp
fjwgfgcMmVD+Uw1yUtOrNTNRW0cIpTJkL1cxeyiusDcaSSWlffOAbexzAZla/sur
mglT8EQD3DWNd47d25P8+wRwiPtre18pjfksQdiJNCopF1jVwY4s
-----END CERTIFICATE-----
```

ca_private.key

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQC4udtmjXIL7nVN
sPkyhIAH31btnyLfF/8erh+wByU18fuPq+OcoAYKWOsM1VGTm1kWtNoYwwjcn/Nz
Mjxc+5NA9SQYU9zQM/ZU0Bh0gBXLmgxqm2JthGdWkYtaHGqnlHHwqEo8WnZsar6D
Q/BPGLB/W4H/3491mFFIpKRoGUbFTn9xF9Ke0ojpa/Z1ONxdTFaMln+467PjF7tG
AH+AM18nDHGdhlGyILB7TUyffUGETvGFO8WT0rLZbFF6JCEbnmsc7c3OxhsbJ2xz
glxldUSl+jNvRXXGrtoYjrvu1bnKdwSMoViycBUYpUKk1Pm0/3ctvaHNeDg0tfhV
tWOWUIcnAgMBAAECggEACE3hSpVuPJITjOgOnnztYoKMSQdf4nE/jfSxm6YJQAkT
pYGf1wEsnwIgovA4ssQyN5PuwAaN20Pc27S07lB9mKx3MG7IskVmZqlU3XE1SZ9b
```

```
EonPXXFX0eueEkbNFqEqPsbZ+3zs7kfILmTQpJr4CD79XkWBu1LFyK26GayVOAu7
434OjAM/SS1mu0fFN/z/y7s6RQgTeKONVf3hyiS6K4UE/0B3lkyUb8ArnZsR7aSX
qzTbXLHnedmo/yavVA2YabvK2Qwr9WrbYbWCBsZHfay4z5wCyyfRvNQc6b7j/IQN
FPMSBQHxtXEtMSWb5rk2h0oHA4K4G3bapyGADThImQKBgQDslVbqgaRPXoFjkU8U
Uo/QMG/AFrHsRNnXIJLCU2YmKHTPa76wgI9jOA9BmyGIOxKYl39VILqg5ueV0JHh
Qa6YqNOsTLk3RIlhP6pPKCiGs7MNEE2Upx/z8TZ3eeY+YlwLF9YQjpgxt2+1+Dll
JriOwEbjDaRpLyFeYrMu2hH5PwKBgQDH4vwe3WA4UXbMH0olNjUMkxDAC67pI9ZJ
2aOwdR+vdy7IIOHe6FxOSogz9n9Npq/YDjPlZwlIlKXwd015ybk2iieqIauZRtrq
PEvudHbnvufquCocmewOof3+dm4Bpjr093tULq3XpQY5j/cH0Z+PVWPGtg+Jfeas
+uNX4PTQGQKBgBVyUEe/6to7EfWptYZ9GYZg7bB3Hjfx4eGLlWJCVy88WaJ3/PhZ
HyZ6bX+gBYEpAb2NdkCMT/7ID9RX7PGDl8VFALNM3TlyYBXNxK2aUVosrh2QDchd
AhuGEU7xYPrkxb1HsuMbWhafr2PKlcFiqjZenc51bSMP37GDtwSXRivXAoGAecnw
afHiXY4tPuPVrmSf4AB/VTQJJ54hQ6/qz2odmzCX/XIjjUiwbTd2U4reN/bW72J/
9kdYXbPMg359YzYotnVpA30ua97DJbMFZM3rYPptnhMF/ZU/Qg8rje6kyjZ5d6aO
q1UhsOGosEvqbQYpHREfEQT+5Hwkn8Lh4Dp9j4kCgYBSDjl6hE/5CEmjFXol+C9i
JYmpsKuYztGMMFu8/blzgPDz4NSu3GyqMHbXh6LvApxdvDw04Np3r6Jyr5w0mJL0
6Phf9l3lsVD/80nBOgJ4v4WSCy3jgVr6e/rpa9YN55TEJVvF8L5lJz0p7tEhzAqS
cJo2GJbmBEO4Uqn89ueVWA==
-----END PRIVATE KEY-----
```

FE

1. 안드로이드 에뮬레이터 설치 또는 핸드폰(안드로이드)에 Expo Go 깔기

   에뮬레이터 설치 방법

   - **안드로이드 스튜디오 설치**
   - **Virtual Device Manager를 찾아주기**
     - 안드로이드 스튜디오를 실행하고 우측 상단 점3개 혹은 More Actions를 클릭
     - Virtual Device Manager 실행
   - `+ 버튼` **클릭 또는 Create New Device**
     - Pixel 5 선택 Next
     - UpsideDownCake 선택 Next
     - Graphics → Hardware - Gles 2.0 → Finish
2. 라이브러리 설치

   ```
   npm install
   ```
3. 실행

   에뮬레이터 실행 시 터미널 명령

   ```
   npm run android
   ```

   핸드폰에 직접 설치 → expo go 앱으로 실행 시 터미널 명령

- 주의 사항 : 컴퓨터와 같은 Wi-fi 접속

```
npm start
```

핸드폰 apk 파일 실행 시

- apk 설치
- 내 파일에서 설치
- 이 앱을 설치하시겠습니까 -> 설치
- 출처를 알 수 없는 앱 설치 -> 무시하고 설치