포팅 메뉴얼

서버 버전 설정

```
Ubuntu: 20.04
Java: 11.0.20
Node: 18.18
Jenknins: 2.422
MariaDB: 10.3.23
Redis: 7.2.1
Python: 3.9.1
```

포트 설정

```
FrontEnt: 3000->3000
BackEnd: 62111->8080
MyData: 62119->62119
AIModel: 8099->8099
Redis: 6379->6379
Config: 62110->62110
Jenkins: 8080->8080
```

Docker

패키지 설치

```
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

레포 등록

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

패키지 업데이트

```
sudo apt-get -y update
```

Jenkins설정

Jenkins 이미지, 컨테이너 실행

```
docker pull jenkins/jenkins:lts

docker run -d --env JENKINS_OPTS=--httpPort=8080 -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -p 8080:8080 -v /jenkins:/var/je
```

Jenkins 파이프라인 플러그인 설치

```
# ssh 커맨드 입력에 사용
SSH Agent

# docker 이미지 생성에 사용
Docker
Docker Commons
Docker Pipeline
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드에 사용
```

```
Generic Webhook Trigger

# 타사 레포지토리 이용시 사용 (GitLab, Github 등)
GitLab
GitLab API
GitLab Authentication
GitHub Authentication

# Node.js 빌드시 사용
NodeJS
```

환경 변수 및 Credential 설정

Gitlab Connection, Webhook설정

DockerHub repo 생성 및 Credential 등록

Ubuntu Credential 추가

Jenkins Item 추가

- 백엔드, 프론트, 컨피그, 마이데이터(카드, 뱅킹 서버)

Backend 파이프라인 스크립트

```
pipeline {
    agent any

    options {
        timeout(time: 10, unit: 'MINUTES')
    }

    environment {
        imageName = "gyuram/ts-backend"
        registryCredential = 'ts-frontend'
        dockerImage = 'tsbe'
        dockerContainer = "tsbe"

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'j9c211.p.ssafy.io'
        releasePort = '62111'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'backend', credentialsId: 'gyulife7301', url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22C211'\
            }
        }
        stage('Gradle Build') {
            steps {
                dir ('Backend') {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean bootJar'
                }
            }
            post {
                failure {
                    error 'Fail Build'
                }
            }
        }
        stage('Image Build'){
            steps{
                sh 'docker build -t ${imageName} --build-arg ssh_encrypt_key=c211goforprize ./Backend'
            }
        }
        stage('Docker Image Push') {
            steps {
                withDockerRegistry([ credentialsId: "ts-frontend", url: "" ]) {
                    sh 'docker push ${imageName}'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    def containersToStop = sh(script: "docker ps -q --filter name=${dockerContainer}", returnStdout: true).trim()
```

```
                    if (containersToStop) {
                        containersToStop.split('\n').each { containerId ->
                            sh(script: "docker stop $containerId", returnStatus: true)
                        }
                    }
                    sh 'docker image prune -f'
                    sh 'docker run --rm -d -p ${releasePort}:8080 --name ${dockerContainer} --net mybridge ${imageName} sleep infinity
                }
            }
        }
    }
}
```

Frontend 파이프라인 스크립트

```
pipeline {
    agent any

    options {
        timeout(time: 10, unit: 'MINUTES')
    }

    environment {
        imageName = "gyuram/ts-frontend"
        registryCredential = 'ts-frontend'
        dockerImage = 'ts-frontend'

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'j9c211.p.ssafy.io'
        releasePort = '3000'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'frontend', credentialsId: 'gyulife7301', url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22C211'
            }
        }
        stage('React Build') {
            steps {
                dir ('Frontend') {
                    nodejs('NodeJS 18.16.1') {
                        sh "npm install"
                        sh "npm run build"
                    }
                }
            }
        }
        stage('Build Image'){
            steps{
                sh 'docker build -t gyuram/ts-frontend ./Frontend/'
            }
        }
        stage('docker image push'){
            steps{
                withDockerRegistry([ credentialsId: "ts-frontend", url: "" ]) {
                    sh 'docker push gyuram/ts-frontend'
                }
            }
        }
        stage('Deploy') {
            steps{
                // sh 'docker ps -q --filter name=tsfe | grep -q . && docker stop tsfe'
                // sh 'docker ps | grep tsfe && docker stop tsfe'
                sh 'if (docker ps -q --filter name=tsfe | grep -q .); then docker stop tsfe; fi'
                sh 'docker run --rm -d -p 3000:3000 --name tsfe gyuram/ts-frontend'
            }
        }
    }
}
```

MyData 파이프라인 스크립트

```
pipeline {
    agent any

    options {
        timeout(time: 10, unit: 'MINUTES')
```

```
    }

    environment {
        imageName = "gyuram/ts-mydata"
        registryCredential = 'ts-frontend'
        dockerImage = 'tsmd'
        dockerContainer = "tsmd"

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'j9c211.p.ssafy.io'
        releasePort = '62119'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'mydata', credentialsId: 'gyulife7301', url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22C211'\
            }
        }
        stage('Gradle Build') {
            steps {
                dir ('Backend/mydata') {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean bootJar'
                }
            }
            post {
                failure {
                    error 'Fail Build'
                }
            }
        }
        stage('Image Build') {
            steps{
                sh 'docker build -t ${imageName} --build-arg ssh_encrypt_key=c211goforprize ./Backend/mydata'
            }
        }
        stage('Docker Image Push') {
            steps {
                withDockerRegistry([ credentialsId: "ts-frontend", url: "" ]) {
                    sh 'docker push ${imageName}'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    def containersToStop = sh(script: "docker ps -q --filter name=${dockerContainer}", returnStdout: true).trim()
                    if (containersToStop) {
                        containersToStop.split('\n').each { containerId ->
                            sh(script: "docker stop $containerId", returnStatus: true)
                        }
                    }
                    sh 'docker image prune -f'
                    sh 'docker run --rm -d -p ${releasePort}:62119 --name ${dockerContainer} --net mybridge ${imageName} sleep infinity
                }
            }
        }
    }
}
```

Config 파이프라인 스크립트

```
pipeline {
    agent any

    options {
        timeout(time: 5, unit: 'MINUTES')
    }

    environment {
        imageName = "gyuram/ts-config"
        registryCredential = 'ts-config'
        dockerImage = 'ts-config'

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'j9c211.p.ssafy.io'
        releasePort = '62110'
    }

    stages {
```

```
        stage('Git Clone') {
            steps {
                git branch: 'config', credentialsId: 'gyulife7301', url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22C211'\
            }
        }
        stage('Gradle Build') {
            steps {
                dir ('Backend/config') {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean bootJar'
                }
            }
            post {
                failure {
                    error 'Fail Build'
                }
            }
        }
        stage('Image Build'){
            steps{
                sh 'docker build -t gyuram/ts-config --build-arg ssh_encrypt_key=c211goforprize ./Backend/config'
            }
        }
        stage('Docker Image Push') {
            steps {
                withDockerRegistry([ credentialsId: "ts-frontend", url: "" ]) {
                    sh 'docker push gyuram/ts-config'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    def containersToStop = sh(script: "docker ps -q --filter name=tscf", returnStdout: true).trim()
                    if (containersToStop) {
                        containersToStop.split('\n').each { containerId ->
                            sh(script: "docker stop $containerId", returnStatus: true)
                        }
                    }
                    sh 'docker image prune -f'
                    sh 'docker run --rm -d -p 62110:62110 --name tscf gyuram/ts-config sleep infinity'
                }
            }
        }
    }
}
```

nginx 설정

nginx 설치

Cerbot, SSL 인증서 발급

/etc/nginx/sites-available/default

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
```

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

  # SSL configuration
  #
  # listen 443 ssl default_server;
  # listen [::]:443 ssl default_server;
  #
  # Note: You should disable gzip for SSL traffic.
  # See: https://bugs.debian.org/773332
  #
  # Read up on ssl_ciphers to ensure a secure configuration.
  # See: https://bugs.debian.org/765782
  #
  # Self signed certs generated by the ssl-cert package
  # Don't use them in a production server!
  #
  # include snippets/snakeoil.conf;

  root /var/www/html;

  # Add index.php to the list if you are using PHP
  index index.html index.htm index.nginx-debian.html;

  server_name _;

  include /etc/nginx/conf.d/service-url.inc;

  location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;

  }

  # pass PHP scripts to FastCGI server
  #
  #location ~ \.php$ {
  # include snippets/fastcgi-php.conf;
  #
  # # With php-fpm (or other unix sockets):
  # fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
  # # With php-cgi (or other tcp sockets):
  # fastcgi_pass 127.0.0.1:9000;
  #}

  # deny access to .htaccess files, if Apache's document root
  # concurs with nginx's one
  #
  #location ~ /\.ht {
  # deny all;
  #}
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
# listen 80;
# listen [::]:80;
#
# server_name example.com;
#
# root /var/www/example.com;
# index index.html;
#
# location / {
#   try_files $uri $uri/ =404;
# }
#}

server {

  # SSL configuration
  #
  # listen 443 ssl default_server;
  # listen [::]:443 ssl default_server;
  #
  # Note: You should disable gzip for SSL traffic.
  # See: https://bugs.debian.org/773332
```

```
#
# Read up on ssl_ciphers to ensure a secure configuration.
# See: https://bugs.debian.org/765782
#
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
  server_name j9c211.p.ssafy.io; # managed by Certbot

underscores_in_headers on;

location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|v3|csrf) {
             proxy_pass http://localhost:62111;
             proxy_set_header Host $host;
             proxy_set_header X-Real-IP $remote_addr;
             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
             proxy_set_header X-Forwarded-Proto $scheme;
}

location /mydata/ {
              proxy_pass http://localhost:62119;
              proxy_set_header X-Real-IP $remote_addr;
              proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
              proxy_set_header Host $http_host;
              proxy_pass_request_headers on;
      }


location / {
  # First attempt to serve request as file, then
  # as directory, then fall back to displaying a 404.
  # try_files $uri $uri/ =404;
  proxy_pass http://localhost:3000;
}
      location /ws {
          proxy_pass http://localhost:3000;
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
          proxy_set_header Connection "upgrade";
          proxy_set_header Host $host;
          proxy_set_header Origin "";
      }

location /api/ {

  proxy_pass http://localhost:62111;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header Host $http_host;
  proxy_pass_request_headers on;

}

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
# include snippets/fastcgi-php.conf;
#
# # With php-fpm (or other unix sockets):
# fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
# # With php-cgi (or other tcp sockets):
# fastcgi_pass 127.0.0.1:9000;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
# deny all;
#}


  listen [::]:443 ssl ipv6only=on; # managed by Certbot
  listen 443 ssl; # managed by Certbot
  ssl_certificate /etc/letsencrypt/live/j9c211.p.ssafy.io/fullchain.pem; # managed by Certbot
  ssl_certificate_key /etc/letsencrypt/live/j9c211.p.ssafy.io/privkey.pem; # managed by Certbot
  include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```

```
}
server {
    if ($host = j9c211.p.ssafy.io) {
        return 308 https://$host$request_uri;
    } # managed by Certbot


  listen 80 ;
  listen [::]:80 ;
    server_name j9c211.p.ssafy.io;
    return 404; # managed by Certbot


}
```

Redis

- Redis 이미지 받기, 컨테이너 실행

```
sudo docker pull redis
docker run -d -p 6379:6379 --name redis redis
```

Config Server

core: application-dev.yml

```
spring:
  datasource:
    url: jdbc:mysql://127.0.0.1:3306/config_test?useUnicode=true&characterEncoding=utf8&autoReconnect=true&serverTimezone=Asia/Seoul&a
    username: "{cipher}492647d4617a15fb201830766ce4f8edfd5e6a28e4f65ab7800c51b51b5f8590"
    password: "{cipher}9cd0e5674c12458217d1ca145b190a9f010e0a86efa3ce9943f52600a25a8826"

  jpa:
    hibernate:
      ddl-auto: none
```

core: application-local.yml

```
spring:
  datasource:
    url: jdbc:mysql://127.0.0.1:3306/config_test?useUnicode=true&characterEncoding=utf8&autoReconnect=true&serverTimezone=Asia/Seoul&a
    username: "{cipher}492647d4617a15fb201830766ce4f8edfd5e6a28e4f65ab7800c51b51b5f8590"
    password: "{cipher}9cd0e5674c12458217d1ca145b190a9f010e0a86efa3ce9943f52600a25a8826"

  jpa:
    hibernate:
      ddl-auto: create
```

core: application.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
```

member: application-dev.yml

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: "{cipher}2d07188d350f9ce487fcfd85c6cbc70bf263c87769280c1f7957ecfab8561c29618df78e665cb5850bfb52c8b9f4329a9fc80ed7de4869b8247c5
    username: "{cipher}bb5c329ffa0113bd18d85a96183fff54f881f97715db1546679d21588aa14594"
    password: "{cipher}5445eb1cf0f91eee34bfb0ef04fe27cedbddfa770bbb9ca2c75184a0b2f1aaa7"


  redis:
```

```
    host: redis
    port: 6379

  jpa:
    database: mysql
    hibernate:
      ddl-auto: create

file:
  dir: "{cipher}18385f0553958441792c19af6ef402567095ee4a0ca5e7ae05ef227b5453dfb85b8fbb9918d4ab3b3c565012c24e64f6"
```

member: application-local.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "{cipher}291cae5aaac55312962e14a788c86f379a4d1ec2b3d66a30c636ccc08e38e7dfd1bf0c1460a53585ea27511c5824698176b838e47493d2b4c8db
    username: "{cipher}744cd30239f57dc92ced14868d44de37f42132afd73e20b995136d1d85c7448b"
    password: "{cipher}f6942826d76ae30d0747b12302f962869dfa516f605c508c33a2fe46bfd82eb8"


  redis:
    host: localhost
    port: 6379

  jpa:
    database: mysql
    hibernate:
      ddl-auto: create

file:
  dir: "{cipher}9d19a62fd20d33cedf55ff97a47e3e8f108634492d8dc2a8106d4b3acfd988d7"
```

member: application-mine.yml

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: "{cipher}892fbc988791c52cb4e1e6284b81ff03327e4764ce7553b839f6e79d916525880fff1f3ba05a5f2b786e88f33e8e11a99bc83cf5a05b174ec8ab
    username: "{cipher}744cd30239f57dc92ced14868d44de37f42132afd73e20b995136d1d85c7448b"
    password: "{cipher}f6942826d76ae30d0747b12302f962869dfa516f605c508c33a2fe46bfd82eb8"


  redis:
    host: localhost
    port: 6379

  jpa:
    hibernate:
      ddl-auto: create

file:
  dir: "{cipher}9d19a62fd20d33cedf55ff97a47e3e8f108634492d8dc2a8106d4b3acfd988d7"
```

member: application.yml

```
naver-cloud-sms:
  accessKey: "{cipher}14db8b8e31b1aaf61d5af0ada5326d3a89be14aec18748db5c644242b880dc37e18e96768fe5033856e731d0738e4774"
  secretKey: "{cipher}0f33f386cf4e4e1365503b3a30cff057da11c54c63cb0a69e4d1bf42eaf5d4f02322fe6599ed2499e430dd76ad50dae27db7093aaccbf172
  serviceId: "{cipher}44514af53b7078c1721ea216e0ed25ded724942517f4fb0fc92f8d1e6e8c01ac96c4d363b47e2e88985e36ef113076da3e71df43572ddbfe
  senderPhone: "{cipher}6be8a17b9c39cc3302057996c39b4112f71e229ea46d17493bf12839c76d95ae"

spring:
  datasource:
    hikari:
      maximumPoolSize: 1
  servlet:
    multipart:
      maxFileSize: 10MB # 파일 하나의 최대 크기
      maxRequestSize: 30MB # 한 번에 최대 업로드 가능 용량

secret:
  access: "{cipher}b494139a074f54c51ff3fa51abf895e1b3b4353d215605fcdd4a826947bf1001f689c848373a64c79e1013e3691ba4caeb942621e2ff337db634
  refresh: "{cipher}e3b767bb4e58298be33f0381bf756ad54ce65da3ff39207f706b78651e08f43e1f6ac0ef98db873c660e4eb61e9750a61c0fd668e133c16f6f9
```

```
naver-key: "{cipher}331b9411f592a00a01effcbac466b89687a16d97605a7804cca799794df084402c38d7f138f41144440952b6071f2af97d3000638200465abf
naver-api: "{cipher}d0846f6ac0ae4e83b7a01488f0a6a2689f1e5c77257b96fc728def86d146bc104f572f8ff294cceeba43bc62cbcc5f90e163a4385176f2588c
kakao-key: "{cipher}94de1160a694d6734552c024993f1f049a31ae4417f407f83830eba8d9d31d92e4f111500d3c4a9c7f59ab942cef8b4ed3066daa214570372c

api:
  base-uri:
    mydata: https://j9c211.p.ssafy.io/mydata

org-code: mydata-ssafy
```

my-data: application-dev.yml

```
spring:
  datasource:
    url: "{cipher}bd09aa96fdc7dafd2031c6218ad86906163ac7047a7f2e107a92154062a2d88574c04ea510f1845b1544eb96805076978e128f56c9dc9753a6ec
    username: "{cipher}e64fb347e6c1cc4e29b42f620f45673ca4a13a8a8aaa4bf8463942e757fc9ec0"
    password: "{cipher}0855fb5b2f9c970f3ba8f3c6e009c73adb0bdba84a2009b0627b3606dfb96cc0"
  jpa:
    hibernate:
      ddl-auto: create

jwt:
  access:
    key: "{cipher}af0efa1b4e4f5d2ae9e7c9be17845250ec18a1e1a2a416cf4fd6a2b654b1e6e4da1c555b4327ea190b8f080391b9ececa459cb754b6641ccf92d
    valid-time: "7776000000"    # 90일
  refresh:
    key: "{cipher}e5aa90a9faca2a3f515ea45b5682f6827d966aaa74985e05058958b378b24ce4e02eadabf6d8f9d55677ba3f7c0126ba9026c0c23248d78b9382
    valid-time: "31536000000"    # 365일
```

my-data: application-local.yml

```
spring:
  datasource:
    url: "{cipher}892fbc988791c52cb4e1e6284b81ff03327e4764ce7553b839f6e79d916525880fff1f3ba05a5f2b786e88f33e8e11a99bc83cf5a05b174ec8ab
    username: "{cipher}744cd30239f57dc92ced14868d44de37f42132afd73e20b995136d1d85c7448b"
    password: "{cipher}f6942826d76ae30d0747b12302f962869dfa516f605c508c33a2fe46bfd82eb8"
  jpa:
    hibernate:
      ddl-auto: create
    properties:
      hibernate:
        show_sql: true
        format_sql: true

logging:
  level:
    org:
      hibernate:
        type:
          descriptor:
            sql: trace

jwt:
  access:
    key: "{cipher}af0efa1b4e4f5d2ae9e7c9be17845250ec18a1e1a2a416cf4fd6a2b654b1e6e4da1c555b4327ea190b8f080391b9ececa459cb754b6641ccf92d
    valid-time: "7776000000"    # 90일
  refresh:
    key: "{cipher}e5aa90a9faca2a3f515ea45b5682f6827d966aaa74985e05058958b378b24ce4e02eadabf6d8f9d55677ba3f7c0126ba9026c0c23248d78b9382
    valid-time: "31536000000"    # 365일
```

my-data: application.yml

```
server:
  port: 62119

spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    hikari:
      maximumPoolSize: 1
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

api:
```

```
  base-uri:
    taesan: https://j9c211.p.ssafy.io/api
```

Backend

Dockerfile

```
# gredle 이미지 불러오기
FROM adoptopenjdk/openjdk11

# jar 파일을 Docker Container의 WORKDIR 위치로 복사 (이미지 생성할 때 동작)
#WORKDIR /config
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} /app.jar

# 환경변수 설정
ARG ssh_encrypt_key
ENV ssh-encrypt-key $ssh_encrypt_key

ENTRYPOINT ["java", "-jar","-Dspring.cloud.config.profile=dev", "/app.jar"]
```

환경 변수 설정

- local 서버 구동 시 프로젝트에 환경 변수 설정 필요

> **ssh-encrypt-key=c211goforprize**

application.yml

```
spring:
  config:
    import: "optional:configserver:http://j9c211.p.ssafy.io:62110" #config server uri
  cloud:
    config:
      name: member
      profile: local

  jpa:
    hibernate:
      ddl-auto: none
    open-in-view: false


  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

  servlet:
    multipart:
      maxFileSize: 10MB # 파일 하나의 최대 크기
      maxRequestSize: 30MB  # 한 번에 최대 업로드 가능 용량
#  sql:
#    init:
#      mode: always

encrypt:
  key: ${ssh-encrypt-key}

file:
  dir: C:/dev/

logging:
  level:
    com.ts.taesan.domain.transaction.req: DEBUG
    com.ts.taesan.domain.analyst.req: DEBUG
```

Mydata

application.yml

```
spring:
  config:
    import: "optional:configserver:http://j9c211.p.ssafy.io:62110" #config server uri
  cloud:
    config:
      name: mydata
      profile: local

encrypt:
  key: ${ssh-encrypt-key}
```

AI 모델

📮 같은 폴더내에 requirements.txt와 model의 bin파일, main.py, fasttext가 같이 존재해야함

shall

```
git clone https://github.com/facebookresearch/fastText.git
docker build -t ai-model .
```

requirements.txt

```
fastapi>=0.103.2
pydantic>=2.4.2
uvicorn >= 0.23.2
numpy >= 1.21.5
scipy >= 1.9.1
```

Dockerfile

```
FROM python:3.9.13

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./main.py /code/main.py

COPY ./fastText /code/fastText

COPY ./final_model_02.bin /code/final_model_02.bin

RUN cd /code/fastText && pip install .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8099"]
```

docker-compose.yml

```
services:
    ai-model:
        image: ai-model
        ports:
            - 8099:8099
networks:
    default:
        external: true
        name: mybridge
```

Front

Dockerfile

```
# 가져올 이미지를 정의
FROM node:18
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.
COPY . .

# 각각의 명령어들은 한줄 한줄씩 캐싱되어 실행된다.
# package.json의 내용은 자주 바뀌진 않을 거지만
# 소스 코드는 자주 바뀌는데
# npm install과 COPY . . 를 동시에 수행하면
# 소스 코드가 조금 달라질때도 항상 npm install을 수행해서 리소스가 낭비된다.

# 3000번 포트 노출
EXPOSE 3000

# npm start 스크립트 실행
CMD ["npm", "start"]

# 그리고 Dockerfile로 docker 이미지를 빌드해야한다.
# $ docker build .
```

주의 사항

- 모든 환경이 구축되고 실행이 되었을 경우 Backend Container가 재시작 시 마이데이터 Container를 재시작해야함
- 서비스 더미 데이터를 넣는 경우 회원 가입 후 자산 연동, 티끌 적금통을 모두 생성한 뒤 4가지 기능에 대한 더미 데이터를 삽입해야함.