⚠️

포팅메뉴얼

CI/CD

📢 **EC2 서버 도메인 :** `j11b202.p.ssafy.io`

🐳 컨테이너 정보

	컨테이너	포트	URL
1	jenkins	`8080:8080`	
2	nginx	`80:80` , `443:443`	
3	mysql	`3306:3306`	
4	springboot	`5000:5000`	
5	react(nginx)	`3000:3000`	
6	redis	`6379:6379`	
7	elasticsearch	`9200:9200` , `9300:9300`	
8	kibana	`5601:5601`	
9	logstash	`5044:5044` , `9600:9600` , `4000:4000`	
10	certbot	`80` , `443`	

서버 접속

📢 ssh -i j11b202.p.ssafy.io.pem ubuntu@j11b202.p.ssafy.io

서버 설정

1. 서버 시간 설정

```
sudo timedatectl set-timezone Asia/Seoul
```

도커 & 도커 컴포즈 설치

1. Docker 설치 문서

Ubuntu
Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install Docker Engine on Ubuntu.
https://docs.docker.com/engine/install/ubuntu/

2. Docker-compose 설치

Install the Compose plugin
Download and install Docker Compose on Linux with this step-by-step handbook. This plugin can be installed manually or by using a repository.
https://docs.docker.com/compose/install/linux/

EC2 폴더 구조

```
📁home/ubuntu
├── 📁 certbot
|    ├── 📁conf
|    └── 📁www
├── 📄 docker-compose.yml
├── 📁 elasticsearch
├── 📁 elk
|    ├── 📁 elasticsearch
|    ├── 📁 kibana
|    ├── 📁 logstash
|    └── 📁 setup
├── 📄 init-letsencrypt.sh
├── 📁 jenkins-data
├── 📁 nginx
|    └── default.conf
└── 📁 redis-data
     ├── data
     └── redis.conf
:folder
```

이미지 빌드 Shell Script

Backend 폴더 구조

```
.
├── 📄 Dockerfile
├── 📁 build
├── 📄 build.gradle
├── 📄 build_image.sh
├── 📁 gradle
├── 📄 gradlew
├── 📄 gradlew.bat
├── 📄settings.gradle
└── 📁 src
     ├──📁 main
     |    ├── 📁 java
     |    |    └── 📁 com
     |    |        └── 📁 ssafy
     |    |            └── 📁 triplet
     │   │               ├── 📄 TripletApplication.java
     │   │               ├──📁 account
     │   │               │   ├── 📁 controller
     │   │               │   ├──📁 dto
     │   │               │   ├──📁 entity
     │   │               │   ├──📁 repository
     │   │               │   └──📁 service
     │   │               ├── 📁 auth
     │   │               │   ├──📁 controller
     │   │               │   ├──📁 dto
     │   │               │   ├──📁 jwt
     │   │               │   └──📁 service
     │   │               ├── 📁 config
     │   │               │   ├──📄 ElasticsearchConfig.java
     │   │               │   ├──📄 MultipartJackson2HttpMess
ageConverter.java
     │   │               │   ├──📄 RedisConfig.java
     │   │               │   ├──📄 ScriptConfig.java
     │   │               │   ├──📄 SecurityConfig.java
     │   │               │   └──📄 WebConfig.java
     │   │               ├── 📁 exception
     │   │               │   ├──📄 CustomErrorCode.java
     │   │               │   ├──📄 CustomException.java
     │   │               │   └──📁 controller
     │   │               ├──📁 exchange
     │   │               │   ├──📁 controller
     │   │               │   ├──📁 dto
     │   │               │   ├──📁 entity
     │   │               │   ├──📁 repository
     │   │               │   ├──📁 service
     │   │               │   └──📁 util
     │   │               ├──📁 member
     │   │               │   ├──📁 controller
     │   │               │   ├──📁 dto
     │   │               │   ├──📁 entity
     │   │               │   ├──📁 repository
     │   │               │   └──📁 service
     │   │               ├──📁 notification
     │   │               │   ├── 📄 FcmTokenController.java
     │   │               │   ├── 📁 config
     │   │               │   ├── 📁 dto
     │   │               │   └── 📁 service
     │   │               ├── 📁 payment
     │   │               │   ├── 📁 controller
     │   │               │   ├── 📁 dto
     │   │               │   └── 📁 service
     │   │               ├── 📁 response
     │   │               │   └── 📄 ApiResponse.java
     │   │               ├── 📁 scheduler
     │   │               │   └── 📄 ScheduledTasks.java
     │   │               ├── 📁 sms
     │   │               │   ├── 📁 controller
     │   │               │   ├── 📁 dto
     │   │               │   ├── 📁 repository
     │   │               │   ├── 📁 service
     │   │               │   └── 📁 util
     │   │               ├── 📁 travel
     │   │               │   ├── 📁 controller
     │   │               │   ├── 📁 dto
     │   │               │   ├── 📁 entity
     │   │               │   ├── 📁 repository
     │   │               │   ├── 📁 service
     │   │               │   ├── 📁 specification
     │   │               │   └── 📁 util
     │   │               └── 📁 validation
     │   │                   └── 📄 CustomValidator.java
     │   └── 📁 resources
     │       ├── 📄 application-secret.properties
     │       ├── 📄 application.properties
     │       ├── 📄 certification.json
     │       ├── 📄 defaultImages.json
     │       ├── 📄 elastic-member-mapping.json
     │       ├── 📄 elastic-travel-mapping.json
     │       └── 📁 scripts
     │           └── 📄 refreshExchangeRates.lua
     └── 📁 test
```

```
        └──📁 java
            └──📁 com
                └──📁 ssafy
                    └──📁 triplet
                        └──📁 TripletApplicationTests.java
```

1. Backend build_image.sh

```
#!/bin/bash

# 변수 설정
IMAGE_NAME="spring"
DOCKERFILE_PATH="."

# Docker 이미지가 있는지 확인
if [[ "$(docker images -q $IMAGE_NAME 2> /dev/null)" != ""
]]; then
  echo "이미지가 존재합니다. 삭제 중..."
  docker rmi -f $IMAGE_NAME
fi

# Docker 이미지 빌드
echo "이미지 빌드 중..."
docker build -t $IMAGE_NAME $DOCKERFILE_PATH

echo "완료되었습니다."
```

2. Backend DockerImage

```
FROM openjdk:17-jdk

# 컨테이너 내부에서의 작업 디렉토리 설정
WORKDIR /home/app
# build/libs의 jar파일을 컨테이너 내부의 app.jar로 복사
COPY build/libs/*.jar app.jar
# app.jar  실행
```

```
ENTRYPOINT ["java","-jar","./app.jar","--spring.profiles.active=secret"]
```

3. Frontend build_image.sh

```
.
├──📄 Dockerfile
├──📄 README.md
├──📄 build_image.sh
├──📁 conf
│   └── conf.d
├──📄 package-lock.json
├──📄 package.json
├──📁 public
│   ├──📁 assets
│   ├──📄 favicon.ico
│   ├──📄 firebase-messaging-sw.js
│   ├──📁 fonts
│   ├──📄 index.html
│   ├──📄 logo192.png
│   ├──📄 logo512.png
│   ├──📄 manifest.json
│   ├──📄 robots.txt
│   └──📄 service-worker.js
├──📁 src
│   ├──📄 App.css
│   ├──📄 App.test.tsx
│   ├──📄 App.tsx
│   ├──📁 assets
│   ├──📁 components
│   ├──📁 features
│   ├──📁 firebaseNotification
│   ├──📁 hooks
│   ├──📄 index.css
│   ├──📄 index.tsx
│   ├──📄 logo.svg
│   ├──📁 pages
│   ├──📄 react-app-env.d.ts
```

```
|   ├──📄 reportWebVitals.ts
|   ├──📁 routes
|   ├──📄 serviceWorkerRegistration.ts
|   ├──📁 services
|   ├──📄 setupTests.ts
|   ├──📄 store.ts
|   └──📁 types
└──📄 tsconfig.json
```

4. Frontend build_image.sh

```
#!/bin/bash

# 변수 설정
IMAGE_NAME="reactapp"
DOCKERFILE_PATH="."

# Docker 이미지가 있는지 확인
if [[ "$(docker images -q $IMAGE_NAME 2> /dev/null)" != ""
]]; then
  echo "이미지가 존재합니다. 삭제 중..."
  docker rmi -f $IMAGE_NAME
fi

# Docker 이미지 빌드
echo "이미지 빌드 중..."
docker build --no-cache -t $IMAGE_NAME $DOCKERFILE_PATH

echo "완료되었습니다."
```

5. Front Dockerfile

```
# step 1 빌드를 하기 위한 과정
FROM node:20.16.0-alpine AS build
COPY    ./package* /usr/src/app/
WORKDIR /usr/src/app
RUN     npm install
COPY . /usr/src/app
RUN apk add tzdata && ln -snf /usr/share/zoneinfo/Asia/Seou
l /etc/localtime
RUN npm run build

# step 2 실행 스테이지를 위한 과정
FROM nginx:stable-alpine
RUN rm -rf /etc/nginx/conf.d
COPY conf /etc/nginx
COPY --from=build /usr/src/app/build /usr/share/nginx/html
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

6. Front conf/conf.d/default.conf

```
server {
    listen 3000;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }
}
```

컨테이너 실행하기

1. docker-compose.yml 작성

```
version: "3"
services:
  setup:
    build:
      context: /home/ubuntu/elk/setup/

      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    init: true
    volumes:
      - /home/ubuntu/elk/setup/entrypoint.sh:/entrypoint.s
h:ro,Z
      - /home/ubuntu/elk/setup/lib.sh:/lib.sh:ro,Z
      - /home/ubuntu/elk/setup/roles:/roles:ro,Z
    environment:
      ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
      LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSW
ORD:-}
      KIBANA_SYSTEM_PASSWORD: ${KIBANA_SYSTEM_PASSWORD:-}
      METRICBEAT_INTERNAL_PASSWORD: ${METRICBEAT_INTERNAL_P
ASSWORD:-}
      FILEBEAT_INTERNAL_PASSWORD: ${FILEBEAT_INTERNAL_PASSW
ORD:-}
      HEARTBEAT_INTERNAL_PASSWORD: ${HEARTBEAT_INTERNAL_PAS
SWORD:-}
      MONITORING_INTERNAL_PASSWORD: ${MONITORING_INTERNAL_P
ASSWORD:-}
      BEATS_SYSTEM_PASSWORD: ${BEATS_SYSTEM_PASSWORD:-}
    depends_on:
      - elasticsearch
      - kibana
      - logstash
  jenkins:
    image: jenkins-main
    container_name: jenkins-main
    user: root
    ports:
      - "8080:8080"
      - "50000:50000"
    environment:
      - JAVA_OPTS=-Xmx1g
      - JENKINS_OPTS=--prefix=/jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/ubuntu/jenkins-data:/var/jenkins_home
    restart: always
  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /home/ubuntu/nginx:/etc/nginx/conf.d
      - /home/ubuntu/certbot/conf:/etc/letsencrypt
      - /home/ubuntu/certbot/www:/var/www/certbot
    command: '/bin/sh -c ''while :; do sleep 6h & wait
$${!}; nginx -s reload; done & nginx -g "daemon off;"'''
    depends_on:
      - spring
      - jenkins
      - reactapp

  certbot:
    image: certbot/certbot
    volumes:
      - /home/ubuntu/certbot/conf:/etc/letsencrypt
      - /home/ubuntu/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do ce
rtbot renew; sleep 12h & wait $${!}; done;'"
  mysql:
    container_name: mysql
    image: mysql:8.0.39
    environment:
      MYSQL_ROOT_PASSWORD: b202j@bK8gEX9B
      MYSQL_DATABASE: triplet
      MYSQL_USER: ssafyb202j
      MYSQL_PASSWORD: b202j@EiK8g1X9
      TZ: 'Asia/Seoul'
    ports:
      - "3306:3306"
    volumes:
      - /home/ubuntu/mysql-data:/var/lib/mysql
  spring:
    container_name: spring
    image: spring
    environment:
      - TZ=Asia/Seoul
    ports:
      - "5000:5000"
    depends_on:
      - mysql
  reactapp:
    container_name: reactapp
    image : reactapp
    ports:
      - "3000:3000"
    environment:
      - TZ=Asia/Seoul
  redis:
    container_name: redis
    image : redis:latest
    ports:
      - "6379:6379"
    command: redis-server /usr/local/etc/redis/redis.conf
    volumes:
      - /home/ubuntu/redis-data/data:/data
      - /home/ubuntu/redis-data/redis.conf:/usr/local/etc/r
edis/redis.conf
  elasticsearch:
    container_name: elasticsearch
    build:
      context: /home/ubuntu/elk/elasticsearch/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - /home/ubuntu/elk/elasticsearch/config/elasticsearc
h.yml:/usr/share/elasticsearch/config/elasticsearch.yml:ro,
Z
      - elasticsearch:/usr/share/elasticsearch/data:Z
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      node.name: elasticsearch
      ES_JAVA_OPTS: -Xms512m -Xmx512m
      ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
      discovery.type: single-node
    restart: unless-stopped

  logstash:
    container_name: logstash
    build:
      context: /home/ubuntu/elk/logstash/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - /home/ubuntu/elk/logstash/config/logstash.yml:/usr/
share/logstash/config/logstash.yml:ro,Z
      - /home/ubuntu/elk/logstash/config/pipelines.yml:/us
r/share/logstash/config/pipelines.yml:ro,Z
      - /home/ubuntu/elk/logstash/pipeline:/usr/share/logst
ash/pipeline:ro,Z
      - /home/ubuntu/elk/logstash/lib:/usr/share/logstash/c
onfig/mysql-connector-java-8.0.26
    ports:
      - 5044:5044
      - 9600:9600
      - 4000:4000/tcp
      - 4000:4000/udp
    environment:
      LS_JAVA_OPTS: -Xms256m -Xmx256m
      LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSW
ORD:-}
```

```
    depends_on:
      - elasticsearch
    restart: unless-stopped

  kibana:
    container_name: kibana
    build:
      context: /home/ubuntu/elk/kibana/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - /home/ubuntu/elk/kibana/config/kibana.yml:/usr/shar
e/kibana/config/kibana.yml:ro,Z
    ports:
      - 5601:5601
    environment:
      KIBANA_SYSTEM_PASSWORD: ${KIBANA_SYSTEM_PASSWORD:-}
    depends_on:
      - elasticsearch
    restart: unless-stopped

volumes:
  elasticsearch:
```

2. docker-compose 실행

```
sudo docker-compose up setup
sudo docker-compose up -d
```

Nginx SSL 설정

1. nginx/default.conf

```
server {
    listen 80;
    listen [::]:80;
    server_name j11b202.p.ssafy.io;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        rewrite ^ https://$server_name$request_uri? permane
nt;
    }
}
```

2. certbot 실행 및 SSL/TSL 인증서 설치

```
root init-letsencrypt.sh  생성 후 아래내용 작성 후 실행

#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=(j11b202.p.ssafy.io)
rsa_key_size=4096
data_path="/home/ubuntu/certbot"
email="wlstjq447@naver.com" # Adding a valid address is str
ongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid
hitting request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and r
```

```
eplace existing certificate? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi
```

3. nginx/default.conf 에 추가

```
upstream app {
 server reactapp:3000;
}



server {
    listen 80;
    listen [::]:80;
    server_name j11b202.p.ssafy.io;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        rewrite ^(.*) https://$server_name:443$request_uri?
permanent;
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name j11b202.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/j11b202.p.ssafy.i
o/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j11b202.p.ssa
fy.io/privkey.pem;
```

```
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://app/;
        proxy_redirect default;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }


    location /jenkins {
        proxy_pass  http://jenkins-main:8080;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP           $remote_add
r;
        proxy_set_header    X-Forwarded-For     $proxy_add_
x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
    location /api {
        proxy_pass http://spring:5000/api;
        proxy_redirect default;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
    location /elasticsearch/ {
        proxy_pass http://elasticsearch:9200/;

        proxy_redirect default;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
    location /kibana {
        proxy_pass http://kibana:5601;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

}
```

4. 인증서 자동 발급을 위해서 docker-compose nginx, certbot에 아래내용 추가

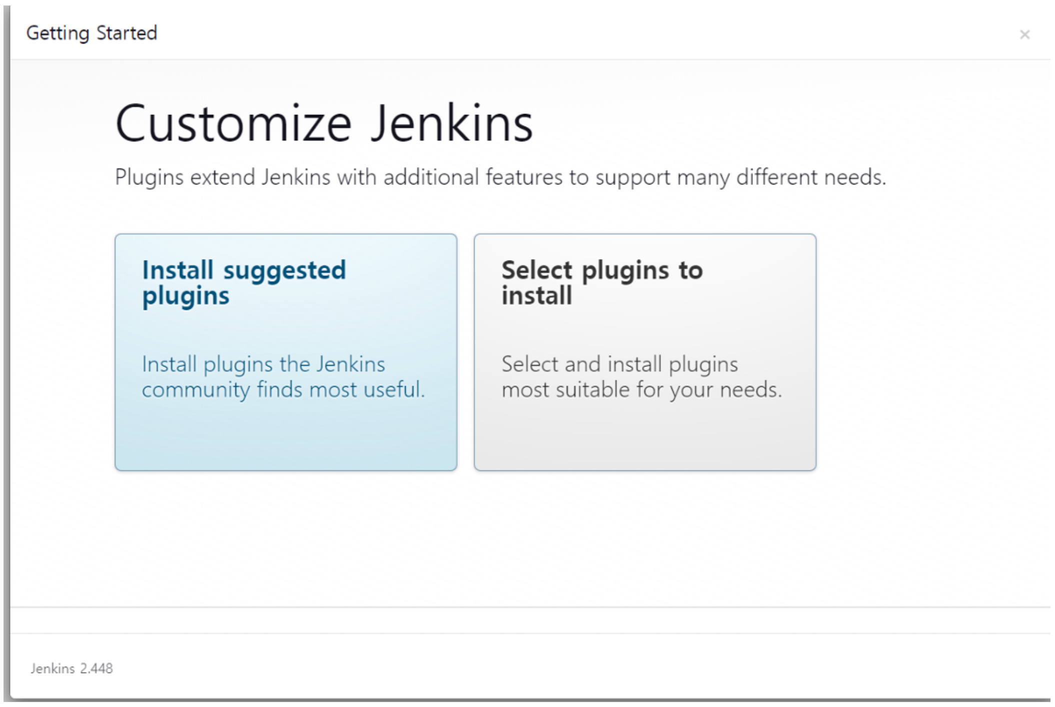
```
nginx:
image: nginx:latest
container_name: nginx
ports:
- "80:80"
- "443:443"
volumes:
- /home/ubuntu/nginx:/etc/nginx/conf.d
- /home/ubuntu/certbot/conf:/etc/letsencrypt
- /home/ubuntu/certbot/www:/var/www/certbot
command: '/bin/sh -c ''while :; do sleep 6h & wait $${!}; n
ginx -s reload; done & nginx -g "daemon off;"'''

certbot:
image: certbot/certbot
volumes:
- /home/ubuntu/certbot/conf:/etc/letsencrypt
- /home/ubuntu/certbot/www:/var/www/certbot
entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbo
t renew; sleep 12h & wait $${!}; done;'
```

Jenkins

jenkins 접속

```
sudo docker logs {docker container name}
```

Jenkins plugin 설치 및 계정 생성


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.448

2. plugin 추가 설치

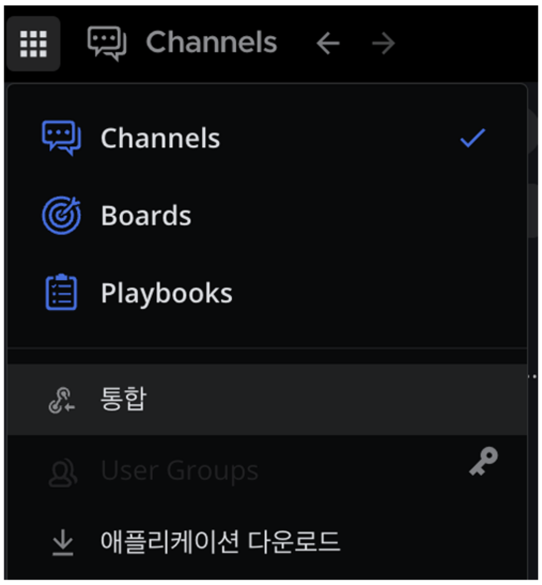
`Dashboard` → `Jenkins관리` → `Plugins`

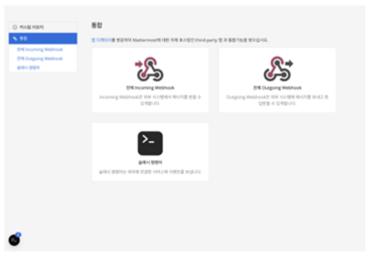
- **Mattermost Notification**
- **GitLab**
- **nodeJs**
- **ssh agent**

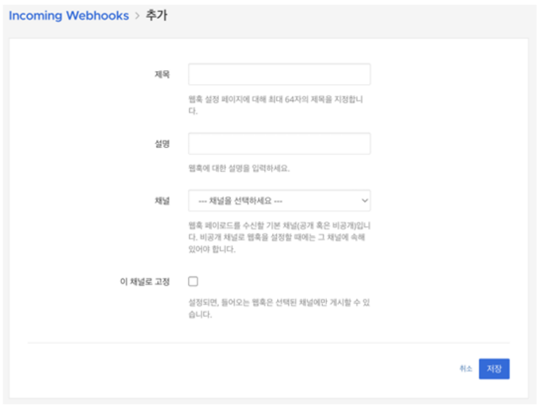
GitLab Token 생성 → Jenkins credential 에 등록 → Dashboard → Jenkins 관리 → System 의 Gitlab에 저장

Mattermost 알림

- 하단 Global Mattermost Notifier Settins 이동
- mattermost 웹훅 생성
  통합 → 전체 Incomming Webhooks → incomming webhook 추가 →생성
- 저장 후 생기는 웹훅 저장






Incoming Webhooks > 추가

제목

웹훅 설정 페이지에 대해 최대 64자의 제목을 지정합니다.

설명

웹훅에 대한 설명을 입력하세요.

채널 --- 채널을 선택하세요 ---

웹훅 페이로드를 수신할 기본 채널(공개 혹은 비공개)입니다. 비공개 채널로 웹훅을 설정할 때에는 그 채널에 속해 있어야 합니다.

이 채널로 고정 ☐

설정되면, 들어오는 웹훅은 선택된 채널에만 게시할 수 있습니다.

취소 저장

Jenkins items 생성→ pipeline 선택 후 등록한 gitlab 선택 후 생성

1. pipeline 작성

- Backend pipeline

```
pipeline {
    agent any
    environment {
        IP = credentials('server_IP')
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-be', credentialsId: 'ssafy
gitlab', url: 'https://lab.ssafy.com/s11-fintech-finance-su
b1/S11P21B202.git'
            }
            post {
                failure {
                    echo 'Repository clone failure!'
                }
                success {
                    echo 'Repository clone success!'
                }
            }
        }

        stage('Copy Security Properties') {
            steps {
                withCredentials([file(credentialsId: 'secur
ityfile', variable: 'SECRET_FILE')]) {
                    sh 'cp $SECRET_FILE ./triplet/src/main/
resources/application-secret.properties'
                }
            }
        }

        stage('Copy Properties') {
            steps {
                withCredentials([file(credentialsId: 'prope
rties', variable: 'PROPERTIES_FILE')]) {
                    sh 'cp $PROPERTIES_FILE ./triplet/src/m
ain/resources/application.properties'
```

```
                }
            }
        }
        stage('Copy springbuild.sh') {
            steps {
                withCredentials([file(credentialsId: 'sprin
gbuild', variable: 'BUILD_FILE')]) {
                    sh 'cp $BUILD_FILE ./triplet/build_imag
e.sh'
                }
            }
        }

        stage('Copy fcmCertification') {
            steps {
                withCredentials([file(credentialsId: 'fcmCe
rtification', variable: 'FCM_FILE')]) {
                    sh 'cp $FCM_FILE ./triplet/src/main/res
ources/certification.json'
                }
            }
        }


        stage('Build Backend') {
            steps {
                echo 'Starting Backend Build Process'
                dir('./triplet') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean build'
                    sh '''
                        chmod +x build_image.sh
                        ./build_image.sh
                    '''
                }
                echo 'Backend Build Completed Successfully'
            }
            post {
                success {
                    script {
                        echo 'Build Backend Success'
                    }
                }
                failure {
                    script {
                        echo 'Build Backend Failed'
                    }
                }
            }
        }

        stage('spring container stop&remove') {
            steps {
                echo 'Docker-compose stoping & removing'
                sh 'echo %IP%'
                sshagent(credentials: ['ec2']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubu
ntu@$IP "sudo docker-compose stop spring"
                        ssh -o StrictHostKeyChecking=no ubu
ntu@$IP "sudo docker-compose rm -f spring"
                        ssh -o StrictHostKeyChecking=no ubu
ntu@$IP "sudo docker ps -a"
                    '''
                }

                echo 'docker-compose stoped & removed'
            }
            post {
                success {
                    script {
                        echo 'stop & remove Success'
                    }
                }
                failure {
                    script {
                        echo 'stop & remove Failed'
                    }
                }
            }
        }

        stage('Deploy Backend') {
            steps {
                echo 'Docker-compose up spring'
                sh 'echo %IP%'
                sshagent(credentials: ['ec2']) {
                     sh 'ssh -o StrictHostKeyChecking=no ub
untu@$IP "sudo docker-compose up -d --remove-orphans sprin
g"'
                }
                echo 'Docker-compose up spring finish'
            }
            post {
                success {
                    script {
                        echo 'Deploy Backend Success'
                    }
                }
                failure {
                    script {
                        echo 'Deploy Backend Failed'
                    }
                }
            }
        }
    }

    post {
        always {
            echo 'Pipeline Execution Complete.'
        }
        success {
            echo 'Pipeline Execution Success.'
            script {
                echo '빌드/배포 Success'
                    def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                    def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                    mattermostSend(color: 'good',
                        message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                        endpoint: '여기에 MatterMost hook url',
                        channel: 'V-Team'
                    )
            }
        }
        failure {
            echo 'Pipeline Execution Failed.'
            script {
                echo '빌드/배포 Failed'
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend(color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: ' 여기에 MatterMost hook url',
                    channel: 'V-Team'
                )
            }
        }
    }
}
```

- frontend pipeline

```
pipeline {
    agent any
    environment {
        IP = credentials('server_IP')
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-fe', credentialsId: 'ssafy
gitlab', url: 'https://lab.ssafy.com/s11-fintech-finance-su
b1/S11P21B202.git'
            }
            post {
                failure {
                    echo 'Repository clone failure!'
                }
                success {
                    echo 'Repository clone success!'
                }
            }
        }


        stage('Copy envFile') {
            steps {
                withCredentials([file(credentialsId: 'envFi
le', variable: 'ENV_FILE')]) {
                    sh 'cp $ENV_FILE ./triplet/.env'
                }
            }
        }

        stage('Copy reactbuild.sh') {
            steps {
                withCredentials([file(credentialsId: 'react
build', variable: 'BUILD_FILE')]) {
                    sh 'cp $BUILD_FILE ./triplet/build_imag
e.sh'
```

```
                }
            }
        }


        stage('Build Frontend') {
            steps {
                dir('./triplet') {
                    echo 'Starting Frontend Build Process'
                    sh '''
                        chmod +x build_image.sh
                        ./build_image.sh
                    '''
                }
                echo 'Frontend Build Completed Successfull
y'
            }
            post {
                success {
                    script {
                        echo 'Build Frontend Success'
                    }
                }
                failure {
                    script {
                        echo 'Build Frontend Failed'
                    }
                }
            }
        }

        stage('reactapp container stop&remove') {
            steps {
                echo 'Docker-compose stopping & removing'
                sh 'echo %IP%'
                sshagent(credentials: ['ec2']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubu
```

```
ntu@$IP "sudo docker-compose stop reactapp"
                        ssh -o StrictHostKeyChecking=no ubu
ntu@$IP "sudo docker-compose rm -f reactapp"
                    '''
                }
                echo 'docker-compose stopped & removed'
            }
            post {
                success {
                    script {
                        echo 'stop & remove Success'
                    }
                }
                failure {
                    script {
                        echo 'stop & remove Failed'
                    }
                }
            }
        }

        stage('Deploy Frontend') {
            steps {
                echo 'Docker-compose up reactapp'
                sh 'echo %IP%'
                sshagent(credentials: ['ec2']) {
                    sh 'ssh -o StrictHostKeyChecking=no ubu
ntu@$IP "sudo docker-compose up -d --remove-orphans reactap
p"'
                }
                echo 'Docker-compose up reactapp finish'
            }
            post {
                success {
                    script {
                        echo 'Deploy Frontend Success'
                    }
                }
                failure {
                    script {
                        echo 'Deploy Frontend Failed'
                    }
                }
            }
        }
    }

    post {
        always {
            echo 'Pipeline Execution Complete.'
        }
        success {
            echo 'Pipeline Execution Success.'
            script {
                echo '빌드/배포 Success'
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(color: 'good',
                    message: "빌드 성공: ${env.JOB_NAME} #${e
nv.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.B
UILD_URL}|Details>)",
                    endpoint: 'Mattemost 웹훅을 입력하세요여기
에',
                    channel: 'V-Team'
                )
            }
        }
        failure {
            echo 'Pipeline Execution Failed.'
            script {
                echo '빌드/배포 Failed'
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend(color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${e
nv.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.B
UILD_URL}|Details>)",
                    endpoint: 'Mattemost 웹훅을 입력하세요여기
에',
                    channel: 'V-Team'
                )
            }
        }
    }
}
```

ELK 실행하기

1.https://github.com/deviantony/docker-elk clone

2. elk/.env 열고 password 설정

3. docker-compose.yml 기존 docker-compose.yml에 이어서 작성

4. kibana 접속 및 devTools 에서 아래 명령어 실행

```
PUT /_index_template/travel_template
{
  "index_patterns": ["travel*"],
  "template": {
    "mappings": {
      "properties": {
        "country_id": { "type": "integer" },
        "days": { "type": "integer" },
        "member_count": { "type": "integer" },
        "total_budget": { "type": "double" },
        "total_budget_won": { "type": "double" },
        "image": { "type": "text" },
        "creator_id": { "type": "long" },
        "id": { "type": "long" },
        "is_shared": { "type": "boolean" },
        "share_status": { "type": "boolean" },
        "status": { "type": "boolean" },
        "title": { "type": "text" }
      }
    }
  }
}

PUT /_index_template/member_template
{
  "index_patterns": ["member*"],
  "template": {
    "mappings": {
      "properties": {
        "age": {
          "type": "integer"
        },
        "gender": {
          "type": "integer"
        },
        "id": {
          "type": "long"
        },
        "travels": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
  }
}
```

logstash 설정

1. mysql-connector-java-{version}.jar 파일 다운

2. logstash/lib 에 저장

3. logstash/config/ pipelines.yml 생성 후 아래 내용 작성

```
- pipeline.id: travel
  path.config: "/usr/share/logstash/pipeline/travel.conf"
- pipeline.id: member
  path.config: "/usr/share/logstash/pipeline/member.conf"
```

4. logstash/pipeline/member.conf 생성 후 작성

```
input {
  jdbc {
    jdbc_driver_library => "/usr/share/logstash/config/mysq
l-connector-java-8.0.26/mysql-connector-java-8.0.26.jar"
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://mysql:3306/trip
let?useUnicode=true&serverTimezone=Asia/Seoul"
    jdbc_user => "ssafyb202j"
    jdbc_password => "b202j@EiK8g1X9"
    statement => "SELECT m.*, GROUP_CONCAT(tm.travel_id) as
travels
FROM member m
LEFT JOIN travel_member tm ON m.id = tm.member_id
GROUP BY m.id
"
  }
}

filter {
   ruby {
  code => "
    require 'date'
    birth = event.get('birth')
    if birth
      birth_date = Date.strptime(birth, '%y%m%d')
      if birth_date > Date.today
        birth_date = birth_date.prev_year(100)
      end
      age = Date.today.year - birth_date.year
      event.set('age', age)
    end
    gender_boolean = event.get('gender') == true ? 0 : 1
    event.set('gender', gender_boolean)

travels = event.get('travels').split(',')
      event.set('travels', travels)
  "
}
    mutate {
      add_field => { "[@metadata][_id]" => "%{id}" }
      remove_field => [ "birth", "role", "name", "passwor
d", "simple_password", "krw_account", "member_id", "phone_n
umber", "krw_account_id", "@version", "@timestamp", "tags"
]
    }

}

output {

    elasticsearch {
      hosts => ["http://elasticsearch:9200"]
      user => "elastic"
      password => "nu9qk5Fkid*iT_wLp2g2"
      index => "member"
      document_id => "%{[@metadata][_id]}"
    }

}
```

5. logstash/pipeline/travel.conf 작성

```
input {
  jdbc {
    jdbc_driver_library => "/usr/share/logstash/config/mysq
l-connector-java-8.0.26/mysql-connector-java-8.0.26.jar"
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://mysql:3306/trip
let?useUnicode=true&serverTimezone=Asia/Seoul"
    jdbc_user => "ssafyb202j"
    jdbc_password => "b202j@EiK8g1X9"
    statement => "SELECT t.*, c.name as country_name
FROM travel t
         JOIN country c ON t.country_id = c.id"
  }
}

filter {
    ruby {
      code => "
        require 'date'
        start_date = Date.parse(event.get('start_date').to_
s)
        end_date = Date.parse(event.get('end_date').to_s)
        days = (end_date - start_date).to_i + 1
        event.set('days', days)
      "
    }
    mutate {
convert => {
        "country_id" => "integer"
        "days" => "integer"
        "member_count" => "integer"
      }
      add_field => { "[@metadata][_id]" => "%{id}" }
 remove_field => [ "start_date", "airport_cost", "invite_co
de", "end_date", "@version", "@timestamp", "tags" ]
```

```
    }

}

output {
    elasticsearch {
      hosts => ["http://elasticsearch:9200"]
      user => "elastic"
      password => "nu9qk5Fkid*iT_wLp2g2"
      index => "travel"
      document_id => "%{[@metadata][_id]}"
    }

}
```

SpringBoot 설정

1. application.yml

```
spring.application.name=triplet
spring.profiles.include=secret

spring.jpa.database-platform=org.hibernate.dialect.MySQLDia
lect
spring.jpa.hibernate.ddl-auto=none
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
server.port=5000




cloud.aws.credentials.accessKey=${AWS_ACCESS_KEY}
cloud.aws.credentials.secretKey=${AWS_SECRET_KEY}
cloud.aws.region.static=ap-northeast-2
cloud.aws.s3.bucket=${AWS_BUCKET}
```

2. application-secret.yml 작성 (git push x, jenkins credentials secret file 로 관리)

```
# Database
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Drive
r
spring.datasource.url=jdbc:mysql://mysql:3306/triplet?allow
PublicKeyRetrieval=true&useSSL=false&characterEncoding=UTF-
8&serverTimezone=Asia/Seoul
spring.datasource.username=""
spring.datasource.password=

# s3
AWS_ACCESS_KEY=
AWS_SECRET_KEY=
AWS_BUCKET=

spring.jwt.secret=

# sms
coolsms.api.key=
coolsms.api.secret=
coolsms.fromnumber=

# kakao login
spring.security.oauth2.client.registration.kakao.client-nam
e=Kakao
spring.security.oauth2.client.registration.kakao.client-id=
spring.security.oauth2.client.registration.kakao.client-sec
ret=spring.security.oauth2.client.registration.kakao.scope=
spring.security.oauth2.client.registration.kakao.client-aut
hentication-method=
spring.security.oauth2.client.registration.kakao.redirect-u
ri=
spring.security.oauth2.client.registration.kakao.authorizat
```

```
ion-grant-type=

# kakao Provider
spring.security.oauth2.client.provider.kakao.authorization-
uri=
spring.security.oauth2.client.provider.kakao.token-uri=
spring.security.oauth2.client.provider.kakao.user-info-uri=
spring.security.oauth2.client.provider.kakao.user-name-attr
ibute=
# redis
spring.data.redis.host=redis
spring.data.redis.port=6379
spring.data.redis.password=
spring.data.redis.repositories.enabled=false

# ssafy api
ssafy.api.key=
ssafy.api.url=

# Elasticsearch
elasticsearch.username=elastic
elasticsearch.password=

#Fcm
fcm.certification=certification.json
```