SSAFY D108

FinCatch 포팅 매뉴얼

목차

1 EC2

1.1 ufw 방화벽 설정

- 필요한 포트 열기

```
ufw allow 22            #ssh
ufw allow 80            #http
ufw allow 443           #https
ufw allow 8083          #Jenkins
ufw allow 15432         #Postgres
ufw allow 7000          #redis
```

1.2 사용도구

- Backend

IDE : IntelliJ IDEA 23.3

JVM : OpenJDK 17

Server Engine : Apache Tomcat 10.1.36

- Frontend

IDE : Visual Studio Code 1.99.0

Node.js : node 20.19.0 / npm 11.2.0

- Database

PostgreSQL : 17..4

Redis : 7.4.2

2 Docker

2.1 Docker 설치

```
#패키지 업데이트
sudo apt-get update

#https 관련 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-
properties-common

#docker repository 접근을 위한 gpg 키 설정
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -

#docker repository 등록
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"

#다시 패키지 업데이트
sudo apt-get update

#도커 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io

#설치 확인
sudo docker -version

#도커로 hello-world 이미지로 테스트
sudo docker run hello-world

#도커 네트워크 생성
sudo docker network create fincatch
```

아래와 같은 메시지가 나오면 성공!

```
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest:
sha256:926fac19d22aa2d60f1a276b66a20eb765fbeea2db5dbdaafeb456ad8ce81598
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker
Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs
the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which
sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/#다시 패키지 업데이트
```

2.2 Docker Compose 설치

```
#docker compose 설치
sudo curl -SL
https://github.com/docker/compose/releases/download/v2.20.0/docker-
compose-linux-x86_64 -o /usr/local/bin/docker-compose

#설치한 파일에 실행권한 추가
sudo chmod +x /usr/local/bin/docker-compose

#docker-compose 명령어 심볼릭 추가
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

#docker-compose 버전 확인
sudo docker-compose -v
```

3 Database

3.1 docker-compose.yml 작성

```
version: "3.8"

services:
  postgres:
    image: postgres:17.4
    container_name: postgres
    restart: unless-stopped
    networks:
      - fincatch
    ports:
      - "15432:5432"
    environment:
      POSTGRES_USER: {postgres_user_id}
      POSTGRES_PASSWORD: {postgres_password}
      POSTGRES_DB: finbattles
    volumes:
      - /home/ubuntu/postgres_data:/var/lib/postgresql/data
      - ./postgres-config:/docker-entrypoint-initdb.d

  postgres-exporter:
    image: prometheuscommunity/postgres-exporter
    container_name: postgres-exporter
    environment:
      DATA_SOURCE_NAME:
"postgresql://root:admin456rootmanager12@postgres:5432/finbattles?sslmod
e=disable"
    ports:
      - "9187:9187"
    networks:
      - fincatch
    depends_on:
      - postgres

  redis:
    image: redis:latest
    container_name: redis
    restart: unless-stopped
    networks:
      - fincatch
    ports:
      - "7000:6379"
    command: ["redis-server", "--bind", "0.0.0.0", "--requirepass",
"{redis-password}"]
    volumes:
      - /home/ubuntu/redis:/data

networks:
  fincatch:
    external: true
```

4 Jenkins

4.1 Jenkins 데이터 사전 설정

```
#jenkins volume 폴더 생성
mkdir /home/ubuntu/jenkins-data

#pull jenkins image & run container
sudo docker run -d \
        --name jenkins --user root \
        --network fincatch \
        -p 8083:8080 -p 60003:60000 \
        -v /home/ubuntu/jenkins-data:/var/jenkins_home \
        -v /var/run/docker.sock:/var/run/docker.sock \
        -e JENKINS_OPTS="--prefix=/jenkins --httpPort=8083" \
        -e JENKINS_URL="http://j12d108.p.ssafy.io:8083/jenkins" \
jenkins/jenkins:lts

# stop Jenkins container
sudo docker stop jenkins

# Jenkins 업데이터 센터 변경
cd /home/ubuntu/jenkins-data
mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-
center.crt -O ./update-center-rootCAs/update-center.crt
sudo sed -i 's#https://updates.jenkins.io/update-
center.json#https://raw.githubusercontent.com/lework/jenkins-update-
center/master/updates/tencent/update-
center.json#' ./hudson.model.UpdateCenter.xml
```

4.2 docker-compose.yml 작성

```
version: "3.8"

services:
  jenkins:
    image: jenkins/jenkins:latest
    container_name: jenkins
    restart: unless-stopped
    user: root
    volumes:
      - /home/ubuntu/jenkins-data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - fincatch
    environment:
      JENKINS_OPTS: "--prefix=/jenkins"  # Jenkins 접속 경로 설정
      JENKINS_URL: "http://j12d108.p.ssafy.io:8083/jenkins/"  # Jenkins 기본
        #JENKINS_URL: "https://j12d108.p.ssafy.io/jenkins/" #nginx 설정 이후
    entrypoint: ["/bin/sh", "-c", "apt update && apt install -y git docker.io
curl && exec /usr/bin/tini -- /usr/local/bin/jenkins.sh --prefix=/jenkins"]

networks:
  fincatch:
    external: true
```

4.3 Jenkins config.xml 주의 사항

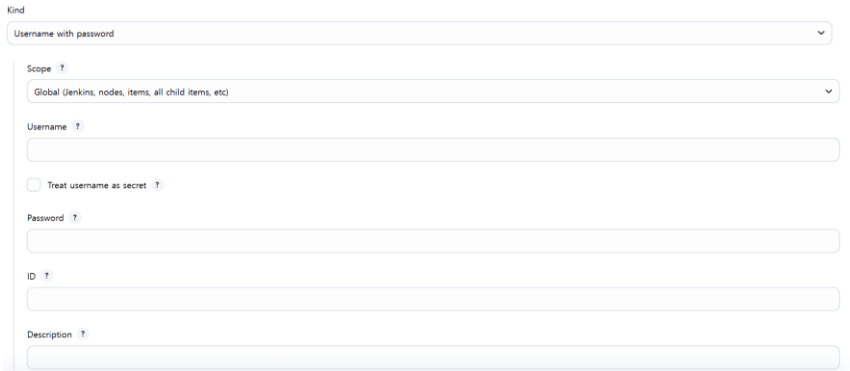
- 아래 명령어 작동 시 true 값이 나와야 함.

```
cat jenkins-data/config.xml | grep useSecurity
<useSecurity>true</useSecurity>
```

4.4 Jenkins 내부 설정

4.4.1 Credentials

- GITLAB (username & password)



Username : Gitlab ID

Password : Gitlab Personal Access Token

- DOCKERHUB_CREDS (username & password)

Username: Docker Hub ID

Password: Docker Hub Password

- BACKEND_CONTAINER_NAME (secret text) : backend container 이름

- BACKEND_IMAGE_NAME (secret text) : backend image 이름

- FRONTEND_CONTAINER_NAME (secret text) : frontend container 이름

- FRONTEND_IMAGE_NAME (secret text) : frontend image 이름

- MM_WEBHOOK_URL (secret text) : Build & Deploy 후 알림 전송

- MERGE_MM_WEBHOOK_URL (secret text) : Merge Requset event 발생 시 알림 전송

- BACKEND_APPLICATION_YAML (secret file) : backend application-secret.yml 업로드

5 Nginx

5.1 docker-compose.yml 작성

```yaml
version: "3.8"

services:
  nginx:
    image: nginx:latest
    container_name: nginx
    restart: always
    networks:
      - fincatch
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /home/ubuntu/nginx-data/default.conf:/etc/nginx/conf.d/default.conf
      - /home/ubuntu/nginx-data/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/:/etc/letsencrypt/
      - /etc/nginx/sites-available/:/etc/nginx/sites-available/
      - /etc/nginx/sites-enabled/:/etc/nginx/sites-enabled/

networks:
  fincatch:
    external: true
```

5.2 nginx.conf 작성

```
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    proxy_headers_hash_max_size 1024;
    proxy_headers_hash_bucket_size 128;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    keepalive_timeout 65;

    include /etc/nginx/conf.d/*.conf;
}
```

5.3 default.conf 작성

```
resolver 127.0.0.11 valid=10s ipv6=off;
resolver_timeout 5s;

upstream frontend {
    zone frontend_zone 64k; # shared memory 설정 추가
    server frontend:3209 resolve;
}

upstream backend {
    zone backend_zone 64k;
    server backend:9091 resolve;
}

upstream jenkins {
    zone jenkins_zone 64k;
    server jenkins:8080 resolve;
}

upstream sonarqube {
        server sonarqube:9000;
}
```

```
server {
        listen 80;
        listen [::]:80;
        server_name j12d108.p.ssafy.io;

        location / {
                return 301 https://j12d108.p.ssafy.io$request_uri;
        }
}

server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name j12d108.p.ssafy.io;

        ssl_certificate
/etc/letsencrypt/live/j12d108.p.ssafy.io/fullchain.pem;
        ssl_certificate_key
/etc/letsencrypt/live/j12d108.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {
                proxy_pass http://frontend/;
                proxy_http_version 1.1;
                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 https;
                proxy_set_header X-Forwarded-Host $host;
                error_page 502 503 504 = @fallback;
        }

         location @fallback {
                return 200 "Frontend Service Not Available";
        }

        # ✅ Jenkins 설정 (/jenkins/)
        location /jenkins {
                proxy_pass http://jenkins;
                proxy_http_version 1.1;

                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 https;
                proxy_set_header X-Forwarded-Host $host;
                proxy_set_header X-Forwarded-Prefix /jenkins;

                proxy_set_header X-Jenkins-Context-Path /jenkins;

                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
```

```
        proxy_redirect off;
        error_page 502 503 504 = @jenkins_fallback;
    }

    location @jenkins_fallback {
        return 200 "Jenkins Service Not Available";
    }

    # Spring Boot 데이터 수집 프록시
    location /metrics {
        proxy_pass http://backend/actuator/prometheus;
        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 https;
        proxy_redirect off;
        error_page 502 503 504 = @backend_fallback;
    }

    # ✅ Prometheus 모니터링
    location /prometheus/ {
        proxy_pass http://prometheus:9090/;
        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 https;
        proxy_redirect / /prometheus/;
    }

    # ✅ Grafana 모니터링
    location /grafana/ {
        proxy_pass http://grafana:3000;
        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;
        proxy_redirect off;
    }

    # ✅ Spring Boot API 요청 처리
    location /api/ {
        proxy_pass http://backend/api/;
        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 https;
        proxy_set_header Authorization $http_authorization;
        error_page 502 503 504 = @backend_fallback;
    }

    location @backend_fallback {
        return 200 "Backend Service Not Available";
    }
```

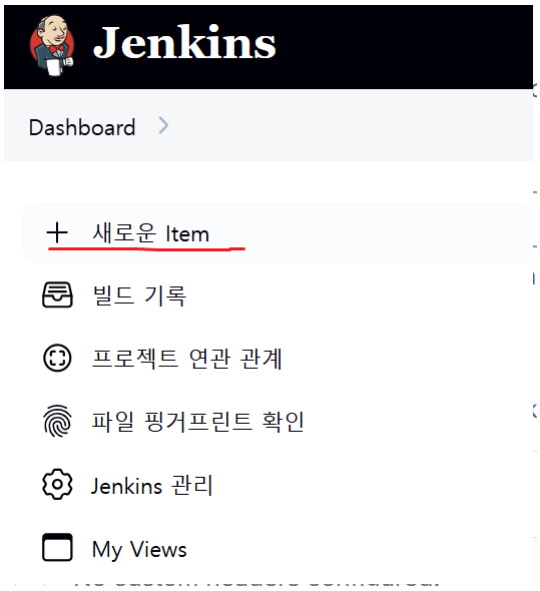
```
        location @sonarqube_fallback {
                return 200 "Sonarqube Service Not Available";
        }

        location /ws/firechat {
                proxy_pass http://backend;
                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 https;
                proxy_set_header Authorization $http_authorization;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                error_page 502 503 504 = @backend_fallback;
        }

        location /oauth2/authorization/ {
                proxy_pass http://backend;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                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;
                error_page 502 503 504 = @backend_fallback;
        }

        location /login/oauth2/code/ {
                proxy_pass http://backend;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                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;
                error_page 502 503 504 = @backend_fallback;
        }

        location /swagger-ui/ {
                proxy_pass http://backend/swagger-ui/;
                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 https;  # $scheme 대신
https로 변경
                error_page 502 503 504 = @backend_fallback;
        }

        location /v3/api-docs {
                proxy_pass http://backend/v3/api-docs;
                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 https;  # $scheme 대신
https로 변경
                error_page 502 503 504 = @backend_fallback;
        }

        location /v3/api-docs/ {
            proxy_pass http://backend/v3/api-docs/;
            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 https;  # $scheme 대신 https로
변경
            error_page 502 503 504 = @backend_fallback;
        }

        location /webjars/ {
                proxy_pass http://backend/webjars/;
                proxy_http_version 1.1;
                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 https;
                error_page 502 503 504 = @backend_fallback;
        }

        location /sonarqube/ {
                proxy_pass http://sonarqube/sonarqube/;
                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 https;
                error_page 502 503 504 = @sonarqube_fallback;
        }
}
```

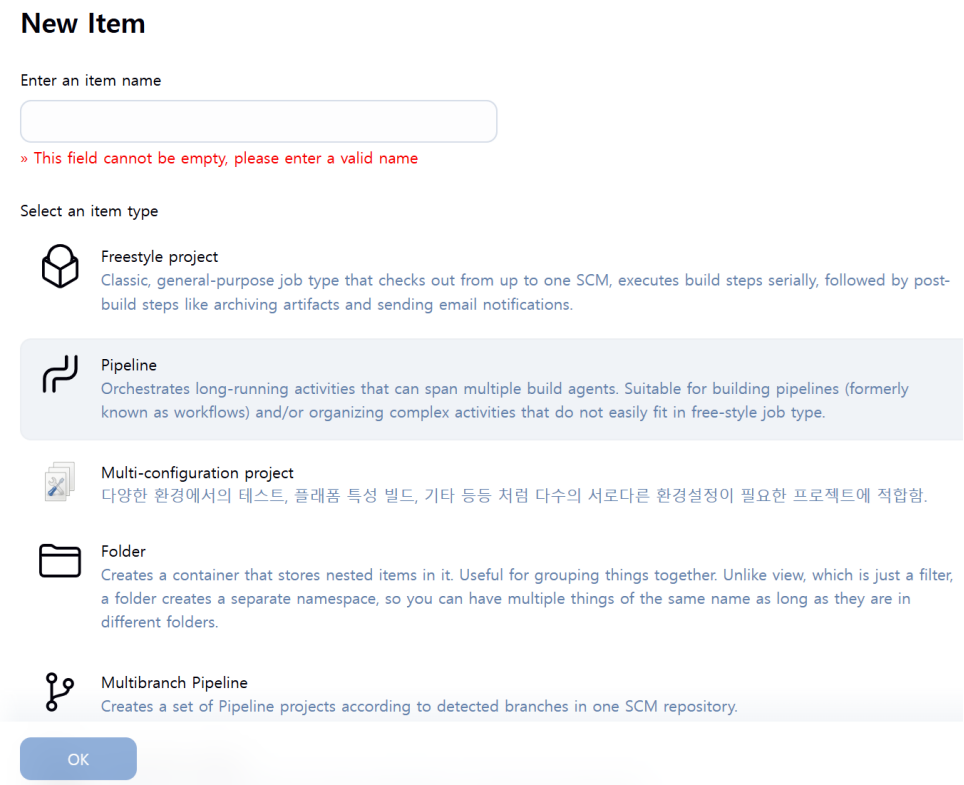
6 Jenkins & Gitlab Webhooks 설정

6.1 Jenkins Pipeline 설정

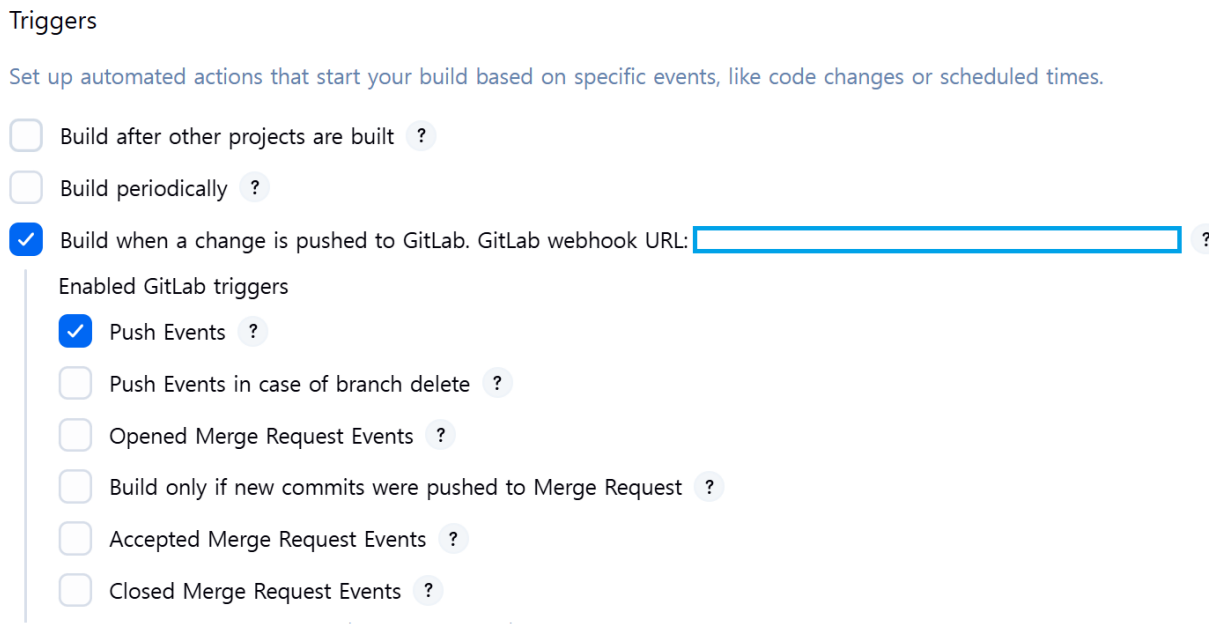
1. 새로운 item 클릭



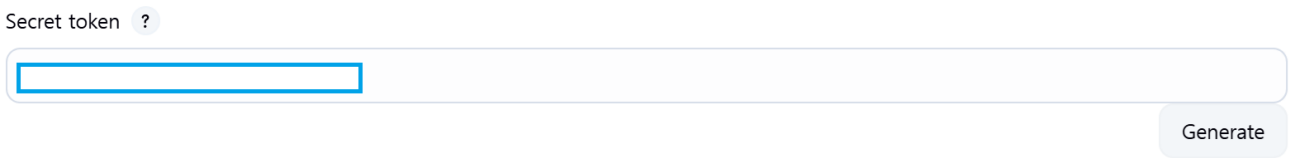
2. Item 이름 작성 > Pipeline 선택 > OK 클릭



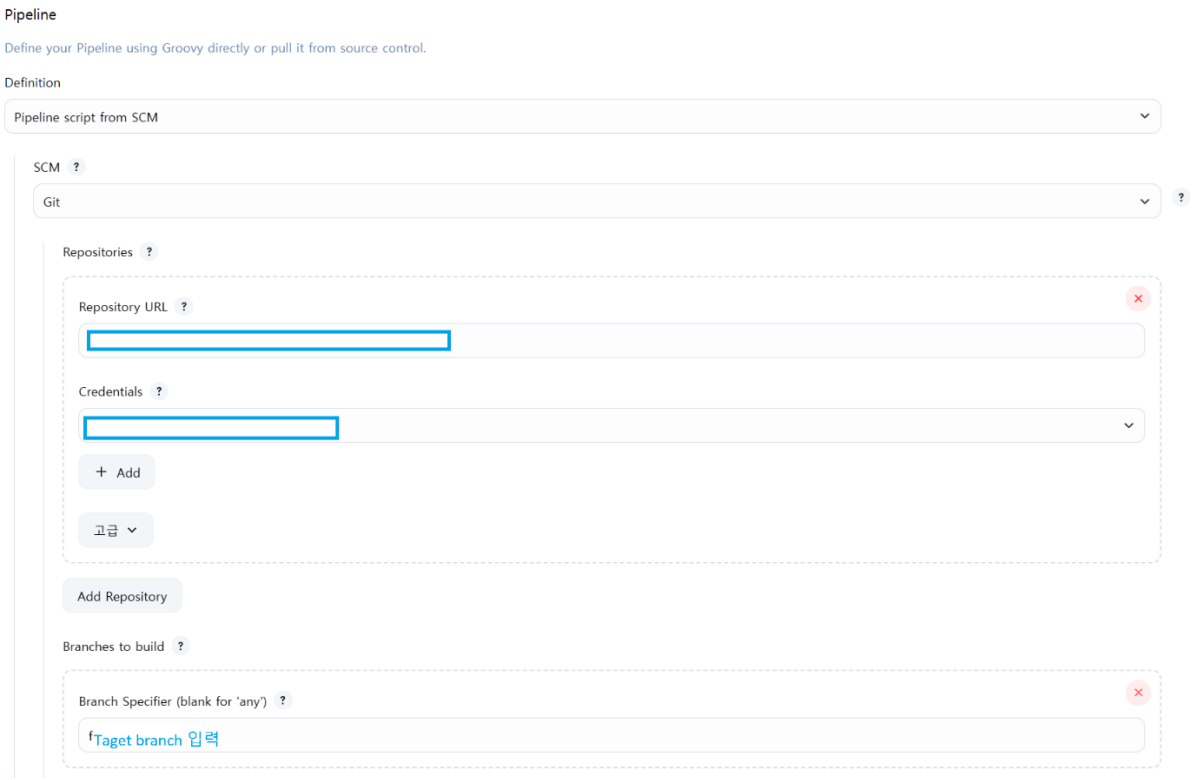
3. Triggers > Build when a change is pushed to Gitlab 클릭 > URL 복사 > Push Event 선택



4. 고급 클릭 > Secret token Generate 후 복사

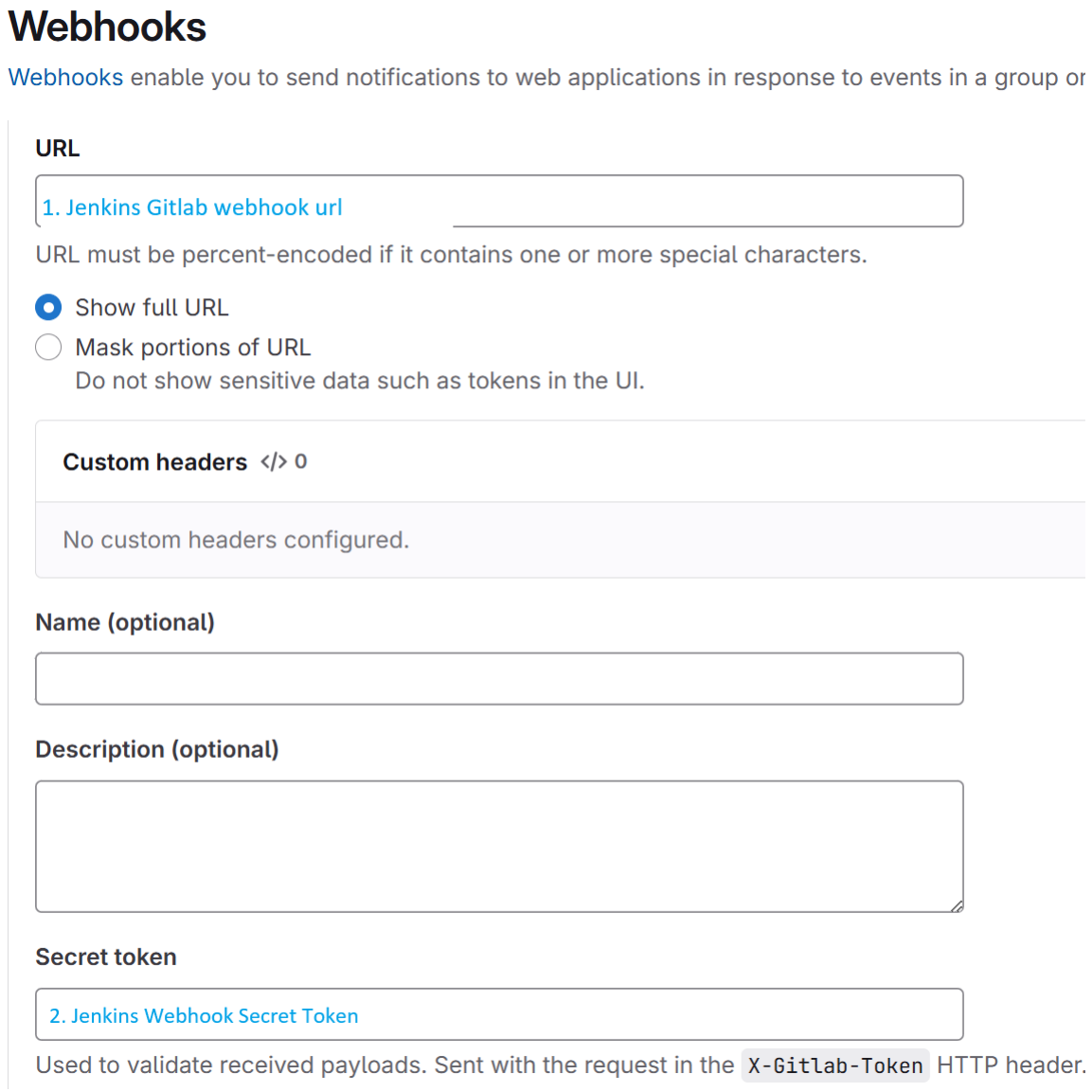


5. Pipeline Definition 설정 > Pipeline script from SCM

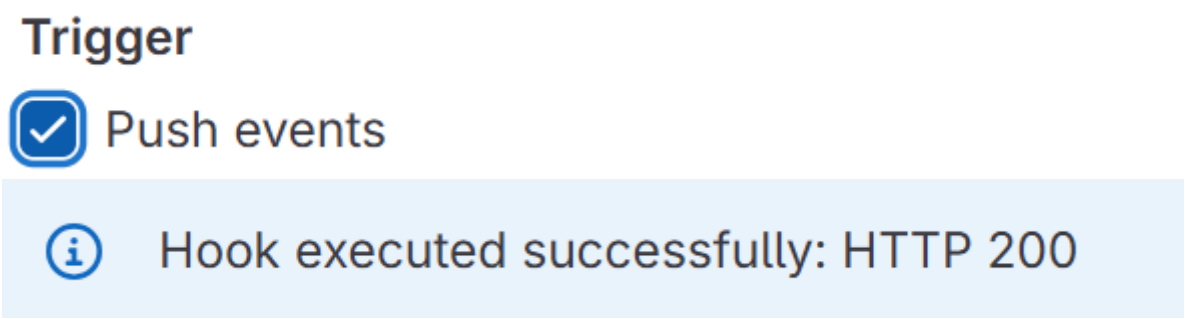


6.2 Gitlab Webhook 설정

1. Project > Settgins > Webhooks > Add new webhook



2. Push event 선택 > Add webhook > Send Test push evnet



7 Backend Build & Deploy

7.1 build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.3'
    id 'io.spring.dependency-management' version '1.1.7'
    id 'org.sonarqube' version '6.0.1.5171'
}

group = 'com'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    //spring-boot develop
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.8.5'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    //security + Oauth2
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-
client'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity6'
    implementation 'org.springframework.security:spring-security-messaging'

    //DB
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.hibernate.validator:hibernate-validator:8.0.1.Final'
    implementation 'jakarta.validation:jakarta.validation-api:3.0.2'
    implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.9.0'
```

```
    runtimeOnly 'org.postgresql:postgresql'

    //Monitoring
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    runtimeOnly 'io.micrometer:micrometer-registry-prometheus'
    implementation 'com.github.loki4j:loki-logback-appender:1.6.0'

    // JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-impl:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.12.3'

    //Test
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    testCompileOnly 'org.projectlombok:lombok'
    testAnnotationProcessor 'org.projectlombok:lombok:1.18.28'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

}

tasks.named('test') {
    useJUnitPlatform()
}
```

7.2 Jenkinfile

```
pipeline {
    agent any

    tools {
        jdk 'JDK17'
    }

    stages {
        stage('Checkout SCM') {
            steps {
                script {
                    checkout scm
                }
            }
        }

        stage('Load Credentials') {
            steps {
                withCredentials([file(credentialsId: 'BACKEND-APPLICATION',
variable: 'SECRET_FILE')]) {
                sh 'cp "$SECRET_FILE" ./backend/finbattle/src/main/resources/
                }
            }
        }

        stage('Build with Gradle') {
            steps {
```

```
            dir('backend/finbattle') {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build -x test'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    dir('backend/finbattle') {
                      withCredentials([string(credentialsId:
'BACKEND_IMAGE_NAME', variable: 'BACKEND_IMAGE_NAME')]) {
                        // 이 때 DOCKER_IMAGE_NAME 은 "myaccount/myrepo:tag"
형태라고 가정
                        sh 'docker build -t ${BACKEND_IMAGE_NAME} .'
                      }
                    }
                }
            }
        }

        stage('Push to Docker Hub') {
            steps {
                script {
                    // 1) Docker Hub 에 로그인
                    withCredentials([
                      usernamePassword(
                          credentialsId: 'DOCKERHUB_CREDS',
                          usernameVariable: 'DOCKER_HUB_USERNAME',
                          passwordVariable: 'DOCKER_HUB_PASSWORD'
                      ),
                      string(credentialsId: 'BACKEND_IMAGE_NAME', variable:
'BACKEND_IMAGE_NAME')
                    ]) {
                        sh """
                            docker login -u ${DOCKER_HUB_USERNAME} -p
${DOCKER_HUB_PASSWORD}
                            docker push ${BACKEND_IMAGE_NAME}
                            docker logout
                        """
                    }
                }
            }
        }

        stage('Deploy Backend') {
            steps {
                script {

                  withCredentials([
                      string(credentialsId: 'BACKEND_IMAGE_NAME', variable:
'BACKEND_IMAGE_NAME'),
                      string(credentialsId: 'BACKEND_CONTAINER_NAME', variable:
'BACKEND_CONTAINER_NAME'),
                  ]){
```

```
                    sh """
                          # 우선 컨테이너 중지/삭제
                          docker stop ${BACKEND_CONTAINER_NAME} || true
                          docker rm ${BACKEND_CONTAINER_NAME} || true

                          # 최신 이미지를 Docker Hub 에서 Pull
                          docker pull ${BACKEND_IMAGE_NAME}

                          # 컨테이너 실행
                          docker run -d --name ${BACKEND_CONTAINER_NAME} --
network fincatch \
                              -p 9097:9091 \
                              -v /home/ubuntu/logs:/logs \
                              -e TZ=Asia/Seoul \
                              -e JAVA_TOOL_OPTIONS="-Duser.timezone=Asia/Seoul"
\
                              -e SPRING_PROFILES_ACTIVE=prod \
                              ${BACKEND_IMAGE_NAME}
                       """

                  }
                }
            }
        }
    }

    post {
        always {
           sh 'rm -f ./backend/finbattle/src/main/resources/application-
secret.yml'
        }
        success {
            script {
                withCredentials([string(credentialsId:
'BACKEND_MM_WEBHOOK_URL', variable: 'BACKEND_MM_WEBHOOK_URL')]) {
                    def jsonMessage = """{
                        "attachments": [{
                            "text": "**✅ Backend Build 성공**\\\\n- 상태:
SUCCESS\\\\n- [🔗 상세 정보](${env.BUILD_URL})",
                            "color": "#00FF00"
                        }]
                    }"""

                    sh """
                    echo '${jsonMessage}' > mattermost_payload.json
                    cat mattermost_payload.json
                    curl -X POST -H "Content-Type: application/json" --data
@mattermost_payload.json '${BACKEND_MM_WEBHOOK_URL}'
                    rm -f mattermost_payload.json
                    """
                }
            }
        }
```

```
        failure {
            script {
                withCredentials([string(credentialsId:
'BACKEND_MM_WEBHOOK_URL', variable: 'BACKEND_MM_WEBHOOK_URL')]) {
                    def jsonMessage = """{
                        "attachments": [{
                            "text": "**❌ Backend Build 실패**\\\\n- 상태:
FAILURE\\\\n- [🔗 상세 정보](${env.BUILD_URL}/console) ",
                            "color": "#FF0000"
                        }]
                    }"""

                    sh """
                    echo '${jsonMessage}' > mattermost_payload.json
                    cat mattermost_payload.json
                    curl -X POST -H "Content-Type: application/json" --data
@mattermost_payload.json '${BACKEND_MM_WEBHOOK_URL}'
                    rm -f mattermost_payload.json
                    """
                }
            }
        }
    }
}
```

7.3 Dockerfile

```
FROM openjdk:17
ARG JAR_FILE=build/libs/finbattle-0.0.1-SNAPSHOT.jar
ADD ${JAR_FILE} app.jar
EXPOSE 9091
ENTRYPOINT ["java", "-Duser.timezone=Asia/Seoul", "-jar", "/app.jar"]
```

7.4 application-secret.yml

```
spring:
  datasource:
    url: jdbc:{postgres url}
    username: {postgres username}
    password: {postgres password}
    driver-class-name: org.postgresql.Driver
  data:
    redis:
      password: {redis password}


  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {kakao oauth client id}
            client-secret: {kakao oauth secret key}
            redirect-uri: {service url}/login/oauth2/code/kakao
```

```
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_post
            scope: profile_nickname,account_email

          google:
            client-name: google
            client-id: {google cloud oauth client id}
            client-secret: {google cloud oatuh secret key}
            redirect-uri: {service url}/login/oauth2/code/google
            authorization-grant-type: authorization_code
            scope: profile,email

        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

  jwt:
    secret-access: {jwt secret access key}
    secret-refresh: {jwt secret refresh key}
    access-token-validity: 3600000  # 60 분 (밀리초)
    refresh-token-validity: 86400000 # 1 일 (밀리초)

  ai:
    openai:
      api-key: {openai api key}
app:
  financeKey: {finance api key}
  financeApi: {finance api url}
```

8 Frontend Build & Deploy

8.1 vite.confin.ts : server port, cors 설정

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";


export default defineConfig({
  plugins: [react()],
  server: {
    port: 3210,
    cors: true,
  },
  define: {
    global: "globalThis",
  },
});
```

8.2 package.json

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@reduxjs/toolkit": "^2.6.0",
    "@stomp/stompjs": "^7.0.1",
    "@types/sockjs-client": "^1.5.4",
    "@types/styled-components": "^5.1.34",
    "axios": "^1.8.1",
    "chart.js": "^4.4.8",
    "lucide-react": "^0.477.0",
    "pixi.js": "^6.5.9",
    "react": "^19.0.0",
    "react-chartjs-2": "^5.3.0",
    "react-dom": "^19.0.0",
    "react-redux": "^9.2.0",
    "react-router-dom": "^7.2.0",
    "redux-persist": "^6.0.0",
    "socket.io": "^4.8.1",
    "socket.io-client": "^4.8.1",
    "sockjs-client": "^1.6.1",
    "styled-components": "^6.1.15"
  },
  "devDependencies": {
    "@eslint/js": "^9.21.0",
    "@types/react": "^19.0.10",
    "@types/react-dom": "^19.0.4",
    "@types/redux-persist": "^4.3.1",
    "@vitejs/plugin-react": "^4.3.4",
    "autoprefixer": "^10.4.20",
    "eslint": "^9.21.0",
    "eslint-plugin-react-hooks": "^5.1.0",
    "eslint-plugin-react-refresh": "^0.4.19",
    "globals": "^15.15.0",
    "postcss": "^8.5.1",
    "tailwind-scrollbar": "^3.0.1",
    "tailwindcss": "^3.4.1",
    "typescript": "~5.7.2",
    "typescript-eslint": "^8.24.1",
    "vite": "^6.2.0"
  }
}
```

8.3 Jenkinfile

```
pipeline {
    agent any

    environment {
        FRONTEND_BRANCH = 'feature-frontend'
    }

    stages {
        stage('Checkout SCM') {
            steps {
                script {
                    checkout scm
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script{
                    withCredentials([string(credentialsId:
'FRONTEND_IMAGE_NAME', variable: 'FRONTEND_IMAGE_NAME')]) {
                        dir('frontend') {
                            sh """docker build -t ${FRONTEND_IMAGE_NAME} ."""
                        }
                    }
                }
            }
        }

        stage('Push to Docker Hub') {
            steps {
                script {
                    withCredentials([
                      usernamePassword(
                          credentialsId: 'DOCKER_CERD_FRONT',
                          usernameVariable: 'DOCKER_HUB_USERNAME',
                          passwordVariable: 'DOCKER_HUB_PASSWORD'
                      ),
                      string(credentialsId: 'FRONTEND_IMAGE_NAME', variable:
'FRONTEND_IMAGE_NAME')
                    ]) {
                        sh """
                            docker login -u ${DOCKER_HUB_USERNAME} -p
${DOCKER_HUB_PASSWORD}
                            docker push ${FRONTEND_IMAGE_NAME}
                            docker logout
                        """
                    }
                }
            }
        }

        stage('Deploy Frontend') {
            steps {
```

```
script {
                    withCredentials([
                        string(credentialsId: 'FRONTEND_IMAGE_NAME', variable:
'FRONTEND_IMAGE_NAME'),
                        string(credentialsId: 'FRONTEND_CONTAINER_NAME',
variable: 'FRONTEND_CONTAINER_NAME')
                    ]) {
                        sh """
                            docker stop ${FRONTEND_CONTAINER_NAME} | true
                            docker rm ${FRONTEND_CONTAINER_NAME} | true
                            docker run -d --name ${FRONTEND_CONTAINER_NAME} \
                                --network fincatch -p 3210:3209 \
                            ${FRONTEND_IMAGE_NAME}
                        """
                    }
                }
            }
        }
    }

    post {
        success {
            script {
                withCredentials([string(credentialsId:
'FRONTEND_MM_WEBHOOK_URL', variable: 'FRONTEND_MM_WEBHOOK_URL')]) {
                    def jsonMessage = """{
                        "attachments": [{
                            "text": "**✅ Frontend Build 성공**\\\\n- 상태:
SUCCESS\\\\n- [🔗 상세 정보](${env.BUILD_URL})",
                            "color": "#00FF00"
                        }]
                    }"""

                    sh """
                    echo '${jsonMessage}' > mattermost_payload.json
                    cat mattermost_payload.json
                    curl -X POST -H "Content-Type: application/json" --data
@mattermost_payload.json '${FRONTEND_MM_WEBHOOK_URL}'
                    rm -f mattermost_payload.json
                    """
                }
            }
        }
        failure {
            script {
                withCredentials([string(credentialsId:
'FRONTEND_MM_WEBHOOK_URL', variable: 'FRONTEND_MM_WEBHOOK_URL')]) {
                    def jsonMessage = """{
                        "attachments": [{
                            "text": "**❌ Frontend Build 실패**\\\\n- 상태:
FAILURE\\\\n- [🔗 상세 정보](${env.BUILD_URL}/console) ",
                            "color": "#FF0000"
                        }]
                    }"""

                    sh """
```

```
                    echo '${jsonMessage}' > mattermost_payload.json
                    cat mattermost_payload.json
                    curl -X POST -H "Content-Type: application/json" --data
@mattermost_payload.json '${FRONTEND_MM_WEBHOOK_URL}'
                    rm -f mattermost_payload.json
                    """
                }
            }
        }
    }
}
```

8.4 Dockerfile

```
FROM node:20 AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY tsconfig.json ./
COPY . .
RUN npm run build
FROM nginx:latest
COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/dist /usr/share/nginx/html
EXPOSE 3209
CMD ["nginx", "-g", "daemon off;"]
```

8.5 nginx.conf

```
server {
    listen 3209;
    server_name j12d108.p.ssafy.io;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri /index.html;
    }
    error_page 404 /index.html;
}
```

9 Prometheus & Grafana & Loki

9-1 Docker-compose.yml 작성

```yaml
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus:/etc/prometheus  # 설정 파일 마운트
      - /home/ubuntu/prometheus-data:/prometheus  # 데이터 저장소 마운트
    ports:
      - "9090:9090"
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
      - "--storage.tsdb.path=/prometheus"  # 데이터 저장 경로 지정
    restart: always
    networks:
      - fincatch

  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - "3001:3000"
    volumes:
      - /home/ubuntu/grafana-data:/var/lib/grafana
    environment:
      - GF_SERVER_ROOT_URL=https://j12d108.p.ssafy.io/grafana/
      - GF_SERVER_SERVE_FROM_SUB_PATH=true
      - GF_SECURITY_ADMIN_PASSWORD=find108** # 기본 관리자 비밀번호 설정
    restart: always
    networks:
      - fincatch

  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    ports:
      - "9100:9100"
    restart: always
    networks:
      - fincatch

  loki:
    image: grafana/loki:2.9.3
    container_name: loki
    ports:
      - "3100:3100"
    volumes:
      - /home/ubuntu/loki-data:/loki
      - ./loki:/etc/loki
    command: -config.file=/etc/loki/loki-config.yml
    restart: always
    networks:
      - fincatch
```

```yaml
  promtail:
    image: grafana/promtail:2.9.3
    container_name: promtail
    volumes:
      - /home/ubuntu/logs:/logs       # 스프링 로그가 저장되는 곳
      - ./promtail:/etc/promtail      # promtail 설정 파일 위치
    command: -config.file=/etc/promtail/promtail-config.yml
    restart: always
    networks:
      - fincatch

networks:
  fincatch:
    external: true
```

9.2 prometheus..yml 작성

```yaml
global:
  scrape_interval: 15s  # 5 초마다 메트릭 수집

scrape_configs:
  - job_name: 'spring-boot' # Springboot 데이터 가져오기
    static_configs:
      - targets: ['nginx:80'] # nginx:80 의 metrics 를 통해 수집
  - job_name: 'node-exporter' # node-exporter 데이터 가져오기
    static_configs:
      - targets: ['node-exporter:9100'] # node-exporter:9100 을 통해 수집
  - job_name: 'postgres'
    static_configs:
      - targets: ['postgres-exporter:9187']
```

9.3 loki-config.yml 작성

```yaml
auth_enabled: false

server:
  http_listen_port: 3100

common:
  path_prefix: /loki  # ✅ 모든 디렉토리 기준 경로


ingester:
  lifecycler:
    address: 127.0.0.1
    ring:
      kvstore:
        store: inmemory
      replication_factor: 1
    final_sleep: 0s
  chunk_idle_period: 5m
  chunk_retain_period: 30s
  wal:
    enabled: true
    dir: /loki/wal  # ✅ 권한 문제 없는 위치에 명시적 WAL 경로


schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
      schema: v13  # ✅ 최신 스키마 버전
      index:
        prefix: index_
        period: 24h

storage_config:
  tsdb_shipper:
    active_index_directory: /loki/tsdb-shipper-active
    cache_location: /loki/tsdb-shipper-cache
    cache_ttl: 24h
  filesystem:
    directory: /loki/chunks

compactor:
  working_directory: /loki/compactor
  shared_store: filesystem

limits_config:
  reject_old_samples: true
  reject_old_samples_max_age: 168h

chunk_store_config:
  chunk_cache_config:
    embedded_cache:
      enabled: true
      max_size_mb: 100  # ✅ 조회 속도 향상을 위한 캐시


table_manager:
  retention_deletes_enabled: true
  retention_period: 168h  # ✅ 7일 후 로그 자동 삭제
```

9.4 promail-config.yml 작성

```yaml
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
  - job_name: springboot-backend
    static_configs:
      - targets:
          - localhost
        labels:
          job: springboot-backend
          __path__: /logs/*.log
```

9.5 application.yml(backend) 작성

```yaml
management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always
    prometheus:
      enabled: true
  metrics:
    export:
      prometheus:
        enabled: true

logging:
  file:
    name: logs/app.log
```