newzy 포팅 메뉴얼

1. 개발환경
    - **1.1. Frontend**
    - **1.2. Backend**
    - **1.3. Server**
    - **1.4. Database**
    - **1.5. UI/UX**
    - **1.6. IDE**
    - **1.7. 형상/이슈관리**
    - **1.8. 기타 툴**
2. 환경변수
    - **2.1. crawl(.env)**
    - **2.2. recommend(.env)**
    - **2.3. Backend(API)**
    - **2.4. Backend(scheduling)**
    - **2.3. 민감 환경변수 관리**
3. EC2 세팅
    - **3.1. Docker 설치**
    - **3.2. Docker-compose 설정**
    - **3.3. Nginx 설정**
    - **3.4. EC2 Port**
    - **3.5. 방화벽(UFW) 설정**
4. CI/CD 구축
    - **4.1. Jenkins 도커 이미지 + 컨테이너**
    - **4.2. Jenkins 설정**
        - 4.2.1 GitLab Credentials 설정
        - 4.2.2 Jenkins Item 생성
        - 4.2.3. GitLab Webhook 설정
        - 4.2.4. 빌드 및 배포
5. Redis 설정
    - **5.1. Redis 설정**
    - **5.2. Docker redis-cli**
    - **5.3. redis.conf**
    - **5.4. sentinel.conf**
6. ELK 설정
    - **6.1. filebeat 설정**
    - **6.2. logstash 설정**

7. DB 덤프 파일 최신본
8. 시연 시나리오
    8.1 홈화면
    8.2 마이 페이지
    8.3 뉴스 목록
    8.4 뉴스 디테일
    8.5 뉴지 목록
    8.6 뉴지 디테일
    8.7 마이 페이지

1. 개발환경

1.1. Frontend

- Node JS 20.13.1
- React 18.3.1
- zustand 4.5.5
- Axios 1.7.7
- Tailwind CSS 3.4.12

1.2. Backend

- Java
    - Java OpenJDK 17
    - Spring Boot 3.3.2
        - Spring Data JPA 3.3.2
        - Spring Security 3.3.2
        - JUnit 5.8.2
        - Lombok 1.18.26
    - Gradle 7.6
- Jakarta API:
    - Jakarta Persistence API: 3.1.0
    - Jakarta Servlet API: 6.0.0

- Swagger: Springdoc OpenAPI 2.0.4
- Database:
  - MySQL Driver: 8.0.33
  - Redis: Spring Boot Starter Data Redis
  - MongoDB: Spring Boot Starter Data MongoDB
- QueryDSL: 5.0.0
- Security:
  - OAuth 2.0 (Client & Resource Server)
  - JWT (Java JWT, jjwt-api 0.11.5)
- AWS S3: AWS Java SDK S3 1.11.1000
- ElasticSearch:
  - Spring Boot Starter Data Elasticsearch
  - Elasticsearch High-Level Client: 7.17.0
- Other Libraries:
  - Guava: 29.0-jre
  - JSON: org.json 20210307
  - Jsoup: 1.7.2
- Scheduler:
  - Spring Retry & Spring Aspects

1.3. Server

- Ubuntu 20.04 LTS
- Docker-compose 2.6.1
- Nginx 1.27.0
- Docker 27.2.1
- Jenkins 2.452.3

1.4. Database

- MySQL 9.0.1
- Redis 7.4.0
- Mongo 8.0.0

1.5. UI/UX

- Figma

1.6. IDE

- Visual Studio Code 1.91.1
- IntelliJ IDEA 2024.01
- PyCharm 2024.01

1.7. 형상/이슈관리

- GitLab
- Jira

1.8. 기타 툴

- Postman 11.6.2
- Termius 9.2.0

2. 환경변수

2.1. crawl(.env)

```
DB_HOST=j11b305.p.ssafy.io
DB_PORT=3306
DB_USER=newzy
DB_PASSWORD=newzy
DB_NAME=newzy
```

2.2. recommend(.env)

```
DB_HOST=mysql
DB_PORT=3306
DB_USER=newzy
DB_PASSWORD=newzy
DB_NAME=newzy

REDIS_HOST=redis-master
REDIS_PORT=6379
REDIS_DB=0

MONGO_HOST=mongo
MONGO_PORT=27017
MONGO_DB_NAME=newzy
MONGO_USER=newzy
MONGO_PASSWORD=newzy

OPENAI_API_KEY=
```

2.3. Backend(API)

application.yml

```
spring:
  application:
    name: backend

  datasource:
    url: jdbc:mysql://mysql:3306/newzy?createDatabaseIfNotExi
    username: newzy
    password: newzy
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      maximum-pool-size: 20        # 최대 연결 수
      minimum-idle: 10             # 최소 대기 연결 수
      connection-timeout: 30000    # 연결 대기 시간 (30초)
      idle-timeout: 600000         # 연결 유지 시간 (10분)
      max-lifetime: 1800000        # 최대 연결 유지 시간 (30분)
```

```
jpa:
  hibernate:
    ddl-auto: update
  show-sql: true
  properties:
    hibernate:
      naming:
        physical-strategy: org.hibernate.boot.model.naming.
security:
  oauth2:
    client:
      registration:
        kakao:
          client-id:
          client-secret:
          client-name: kakao
          authorization-grant-type: authorization_code
          redirect-uri: https://j11b305.p.ssafy.io/api/oaut
          client-authentication-method: POST
          scope:
            - profile_nickname
            - account_email
        google:
          client-id:
          client-secret:
          client-authentication-method: POST
          authorization-grant-type: authorization_code
          redirect-uri: https://j11b305.p.ssafy.io/api/oaut
          scope:
            - email
            - profile
      provider:
        kakao:
          authorization-uri: https://kauth.kakao.com/oauth/
          token-uri: https://kauth.kakao.com/oauth/token
          user-info-uri: https://kapi.kakao.com/v2/user/me
          user-name-attribute: id
```

```
          google:
            authorization-uri: https://accounts.google.com/o/
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://www.googleapis.com/oauth2/
            user-name-attribute: sub
  data:
    redis:
      host: redis-master
      port: 6379
    mongodb:
      uri: mongodb://newzy:newzy@mongo:27017/newzy?authSource=
      username: newzy
      password: newzy
      database: newzy
      authentication-database: admin
  elasticsearch:
    uris:
      elasticsearch:9200
    username:
      newzy
    password:
      newzy

  servlet:
    multipart:
      enabled: true

server:
  port: ${SERVER_PORT:8081} # 기본값은 8081, 환경 변수 SERVER_POR
  address: 0.0.0.0
  servlet:
    context-path: /api
    encoding:
      charset: UTF-8
      enabled: true
      force: true

management:
```

```
  endpoints:
    web:
      exposure:
        include: health
  health:
    redis:
      enabled: false # Redis health check 비활성화

jwt:
  secretkey:

# S3 bucket connection
cloud:
  aws:
    credentials:
      accessKey:
      secretKey:
    s3:
      bucketName: plogbucket
    region:
      static: ap-northeast-2
    stack:
      auto: false
```

logback-spring.xml

```
<configuration>
    <!-- Console에 출력될 로깅 패턴 -->
    <appender name="CONSOLE" class="ch.qos.logback.core.Con
soleAppender">
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread]
%-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <!-- 파일에 로깅 - 환경변수 'CONTAINER_NAME'에 따라 파일명 구
분 -->
```

```
    <appender name="FILE" class="ch.qos.logback.core.rollin
g.RollingFileAppender">
        <file>/var/log/${LOG_FILE_PATH}/${LOG_FILE_NAME}.lo
g</file> <!-- 환경변수를 이용한 로그 파일명 구분 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.T
imeBasedRollingPolicy">
            <fileNamePattern>/var/log/${LOG_FILE_PATH}/${LO
G_FILE_NAME}.%d{yyyy-MM-dd}.gz</fileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread]
%-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="INFO">
        <appender-ref ref="CONSOLE" />
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

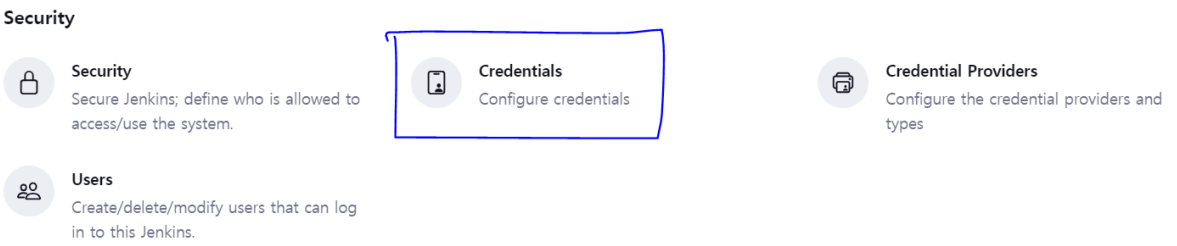
2.4. Backend(scheduling)

```
spring:
  application:
    name: scheduling

  datasource:
    url: jdbc:mysql://mysql:3306/newzy?createDatabaseIfNotExi
    username: newzy
    password: newzy
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
```

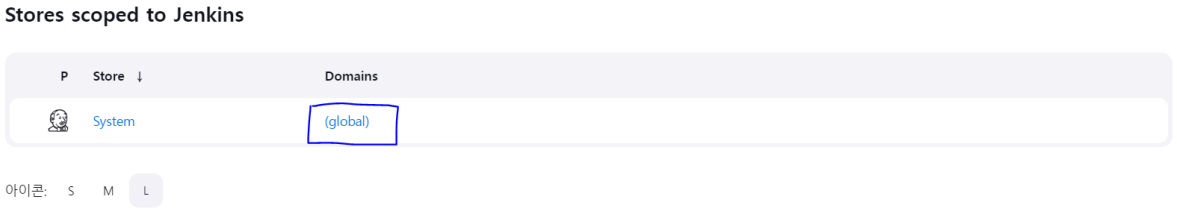
```
    show-sql: true
    properties:
      hibernate:
        naming:
          physical-strategy: org.hibernate.boot.model.naming.

  data:
    redis:
      host: redis-master
      port: 6379
    mongodb:
      uri: mongodb://newzy:newzy@mongo:27017/newzy?authSource=
      username: newzy
      password: newzy
      database: newzy
      authentication-database: admin
  elasticsearch:
    uris:
      elasticsearch:9200
    username:
      newzy
    password:
      newzy

  servlet:
    multipart:
      enabled: true

server:
  port: 8083
  address: 0.0.0.0
  servlet:
    context-path: /scheduling
    encoding:
      charset: UTF-8
      enabled: true
      force: true
```

2.3. 민감 환경변수 관리

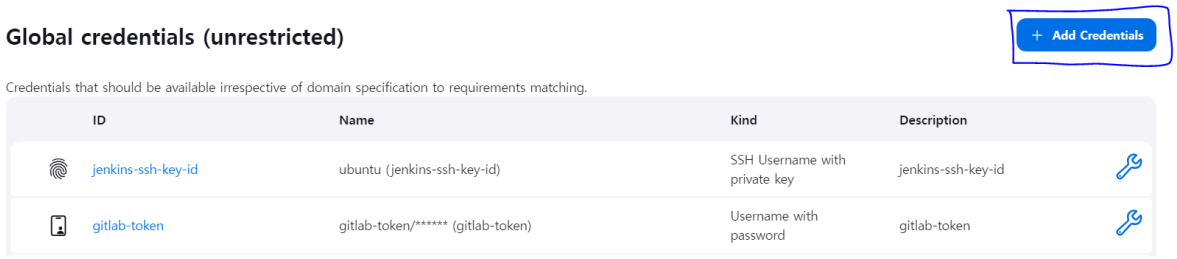
Jenkins credentials로 민감 환경변수 설정파일 수동 저장 및 관리(.gitignore에 추가하여 GitLab에 푸시되는 일이 없도록 함)



Jenkins 설정의 Credentials로 간다.



(global) 도메인으로 만든다.



Add Credentials로 새로운 Crdential을 만든다.

Secret 파일에 민감한 환경변수 파일을 넣고, 파이프라인 구성시 필요한 환경변수 파일을 복사해서 이미지를 빌드하는 형식으로 진행했다.

3. EC2 세팅

3.1. Docker 설치

```
# 1. 리눅스 업데이트
sudo apt update -y && sudo apt upgrade -y

# 1.1 필수 패키지 설치
sudo apt-get install -y ca-certificates curl gnupg lsb-releas

# 2. Docker의 공식 GPG 키를 추가할 디렉토리 생성
sudo mkdir -p /etc/apt/keyrings

# 3. Docker의 GPG 키 다운로드 및 바이너리 형식으로 변환하여 저장
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud

# 4. Docker 저장소를 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list
```

```
# 5. 패키지 목록 업데이트
sudo apt-get update

# 6. Docker 패키지 설치
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# 7. Docker 데몬을 시작하고 부팅 시 자동으로 시작하도록 설정
sudo systemctl start docker
sudo systemctl enable docker
```

3.2. Docker-compose 설정

```
version: '3.8'

services:
        nginx:
                image: nginx:latest
                restart: always
                environment:
                        TZ: Asia/Seoul
                volumes:
                        - ./data/nginx/nginx.conf:/etc/nginx/
                        - ./data/certbot/conf:/etc/letsencryp
                        - ./data/certbot/www:/var/www/certbot
                        - ./data/nginx/log:/var/log/nginx  #
                ports:
                        - "80:80"
                        - "443:443"
                command: "/bin/sh -c 'while :; do sleep 320h

        certbot:
                image: certbot/certbot
                restart: unless-stopped
                environment:
                        TZ: Asia/Seoul
                volumes:
                        - ./data/certbot/conf:/etc/letsencryp
```

```
                - ./data/certbot/www:/var/www/certbot
        entrypoint: "/bin/sh -c 'trap exit TERM; whil

jenkins:
        image: jenkins/jenkins:lts
        restart: unless-stopped
        volumes:
                - jenkins_home:/var/jenkins_home
                - ./data/jenkins:/var/jenkins_shared
                - /var/run/docker.sock:/var/run/docke
                - /usr/local/bin/docker-compose:/usr/
                - /usr/bin/docker:/usr/bin/docker  # 
                - /home/ubuntu/newzy:/home/ubuntu/new
                - ./data/jenkins/log:/var/jenkins_hom

        environment:
                JENKINS_OPTS: --prefix=/jenkins
                JAVA_OPTS: -Djava.util.logging.config
                TZ: Asia/Seoul

backend-v1:
        image: backend:latest
        restart: unless-stopped
        volumes:
                - ./data/backend/log:/var/log/backend
        deploy:
                replicas: 3 # 3개의 인스턴스를 실행
        expose:
                - "8081" # 내부포트 8081
        environment:
                SPRING_PROFILES_ACTIVE: prod
                SERVER_PORT: 8081  # 환경 변수로 포트 설정
                TZ: Asia/Seoul
                LOG_FILE_NAME: backend-v1
                LOG_FILE_PATH: backend


backend-v2:
```

```
                image: backend:latest
                restart: unless-stopped
                volumes:
                        - ./data/backend/log:/var/log/backend
                deploy:
                        replicas: 3 # 3개의 인스턴스를 실행
                expose:
                        - "8082" # 내부포트 8082
                environment:
                        SPRING_PROFILES_ACTIVE: prod
                        SERVER_PORT: 8082  # 환경 변수로 포트 설정
                        TZ: Asia/Seoul
                        LOG_FILE_NAME: backend-v2
                        LOG_FILE_PATH: backend

        frontend-v1:
                image: frontend:latest
                restart: unless-stopped
                volumes:
                        - ./data/frontend/log:/var/log/fronte
                deploy:
                        replicas: 2 # 2개의 인스턴스를 실행
                environment:
                        NODE_ENV: production
                        TZ: Asia/Seoul
                        LISTEN_PORT: 3001  # Nginx에서 사용할 포
                expose:
                        - "3001" # 내부포트 3001

        frontend-v2:
                image: frontend:latest
                restart: unless-stopped
                volumes:
                        - ./data/frontend/log:/var/log/fronte
                deploy:
                        replicas: 2 # 2개의 인스턴스를 실행
                environment:
                        NODE_ENV: production
```

```
                        TZ: Asia/Seoul
                        LISTEN_PORT: 3002  # Nginx에서 사용할 포
                expose:
                        - "3002" # 내부포트 3002

        # Redis Master 설정
        redis-master:
                image: redis:latest
                restart: unless-stopped
                command: redis-server /usr/local/etc/redis/re
                volumes:
                        - ./data/redis/master:/data
                        - ./data/redis/master/log:/var/log/re
                        - ./data/redis/master/redis.conf:/usr
                expose:
                        - "6379"  # 외부에 노출하지 않음, 내부 네트
                environment:
                        TZ: Asia/Seoul
                        REDIS_REPLICATION_MODE: master

        # Redis Slave 1 설정
        redis-slave-1:
                image: redis:latest
                restart: unless-stopped
                command: sh -c "redis-server /usr/local/etc/r
                volumes:
                        - ./data/redis/slave1:/data
                        - ./data/redis/slave1/log:/var/log/re
                        - ./data/redis/slave1/redis.conf:/usr
                expose:
                        - "6379"  # 외부에 노출하지 않음, 내부 네트
                depends_on:
                        - redis-master
                environment:
                        TZ: Asia/Seoul
                        REDIS_REPLICATION_MODE: slave

        # Redis Slave 2 설정
```

```
    redis-slave-2:
        image: redis:latest
        restart: unless-stopped
        command: sh -c "redis-server /usr/local/etc/r
        volumes:
            - ./data/redis/slave2:/data
            - ./data/redis/slave2/log:/var/log/re
            - ./data/redis/slave2/redis.conf:/usr
        expose:
            - "6379"  # 외부에 노출하지 않음, 내부 네트
        depends_on:
            - redis-master
        environment:
            TZ: Asia/Seoul
            REDIS_REPLICATION_MODE: slave

    # Redis Sentinel 1 설정
    redis-sentinel-1:
        image: redis:latest
        restart: unless-stopped
        command: redis-sentinel /etc/sentinel/sentine
        volumes:
            - ./data/redis/sentinel1/sentinel.con
            - ./data/redis/sentinel1/log:/var/log
        expose:
            - "26379"  # 외부에 노출하지 않음
        depends_on:
            - redis-master
            - redis-slave-1
            - redis-slave-2
        environment:
            TZ: Asia/Seoul
            REDIS_REPLICATION_MODE: sentinel

    # Redis Sentinel 2 설정
    redis-sentinel-2:
        image: redis:latest
        restart: unless-stopped
```

```
                command: redis-sentinel /etc/sentinel/sentine
                volumes:
                        - ./data/redis/sentinel2/sentinel.con
                        - ./data/redis/sentinel2/log:/var/log
                expose:
                        - "26380"  # 외부에 노출하지 않음
                depends_on:
                        - redis-master
                        - redis-slave-1
                        - redis-slave-2
                environment:
                        TZ: Asia/Seoul
                        REDIS_REPLICATION_MODE: sentinel


        # Redis Sentinel 3 설정
        redis-sentinel-3:
                image: redis:latest
                restart: unless-stopped
                command: redis-sentinel /etc/sentinel/sentine
                volumes:
                        - ./data/redis/sentinel3/sentinel.con
                        - ./data/redis/sentinel3/log:/var/log
                expose:
                        - "26381"  # 외부에 노출하지 않음
                depends_on:
                        - redis-master
                        - redis-slave-1
                        - redis-slave-2
                environment:
                        TZ: Asia/Seoul
                        REDIS_REPLICATION_MODE: sentinel

        # Redis Stack 설정 (외부 접근 가능)
        redis-stack:
                image: redis/redis-stack:latest
                restart: unless-stopped
                expose:
```

```
                - "8001"  # 내부에서 사용할 포트
        environment:
                TZ: Asia/Seoul
                REDIS_PASSWORD: newzy
        volumes:
                - ./data/redis/stack:/data
        ports:
                - "8001:8001"

redisinsight:
        image: redis/redisinsight:latest
        restart: unless-stopped
        volumes:
                - ./data/redis/insight:/db
                - ./data/redis/insight/log:/db/logs
        depends_on:
                - redis-master
        environment:
                TZ: Asia/Seoul

mysql:
        image: mysql:latest
        restart: unless-stopped
        environment:
                MYSQL_ROOT_PASSWORD: newzy
                MYSQL_DATABASE: newzy
                MYSQL_USER: newzy
                MYSQL_PASSWORD: newzy
                TZ: Asia/Seoul
        volumes:
                - ./data/mysql:/var/lib/mysql
                - ./data/mysql/log:/var/log/mysql  #
                - ./data/mysql/my.cnf:/etc/mysql/my.c
        expose:
                - "3306"

mongo:
        image: mongo:latest
```

```
                restart: unless-stopped
                environment:
                        MONGO_INITDB_ROOT_USERNAME: newzy
                        MONGO_INITDB_ROOT_PASSWORD: newzy
                        TZ: Asia/Seoul
                expose:
                        - "27017"
                ports:
                        - "27017:27017"
                volumes:
                        - ./data/mongo:/data/db  # 호스트의 로컬
                        - ./data/mongo/log:/var/log/mongo  # 
                        - ./data/mongo/mongod.conf:/etc/mongo
                command: mongod --config /etc/mongo/mongod.co


        crawl:
                image: crawl:latest
                restart: unless-stopped
                expose:
                        - "8000"
                environment:
                        DJANGO_SETTINGS_MODULE: crawl.setting
                        TZ: Asia/Seoul
                volumes:
                        - ./data/crawl/log:/var/log/crawl  # 
                command: ["python", "manage.py", "runserver",


        elasticsearch:
                image: elasticsearch:7.17.0
                restart: unless-stopped
                environment:
                        - discovery.type=single-node
                        - ES_JAVA_OPTS=-Xms2g -Xmx2g  # 메모리
                        - TZ=Asia/Seoul
                volumes:
                        - ./data/elasticsearch:/usr/share/ela
```

```
                expose:
                        - "9200"

        logstash:
                image: logstash:7.17.0
                restart: unless-stopped
                environment:
                        - XPACK_MONITORING_ENABLED=true
                depends_on:
                        - elasticsearch
                volumes:
                        - ./config/logstash/logstash.conf:/us
                expose:
                        - "5044"

        kibana:
                image: kibana:7.17.0
                restart: unless-stopped
                depends_on:
                        - elasticsearch
                environment:
                        - ELASTICSEARCH_HOSTS=http://elastics
                        - TZ=Asia/Seoul
                        - SERVER_BASEPATH=/kibana
                        - SERVER_REWRITEBASEPATH=true
                        - SERVER_PUBLICBASEURL=https://j11b30
                expose:
                        - "5601"

        filebeat:
                image: docker.elastic.co/beats/filebeat:7.17.
                restart: unless-stopped
                environment:
                        - TZ=Asia/Seoul
                user: root # 로그파일에 접근하기 위함
                depends_on:
                        - logstash
                volumes:
```

```
                        - ./config/filebeat/filebeat.yml:/usr
                        - /var/lib/docker/containers:/var/lib
                        - /var/run/docker.sock:/var/run/docke
                        # 각 컨테이너의 로그 디렉토리 마운트
                        - ./data/nginx/log:/var/log/nginx:ro
                        - ./data/backend/log:/var/log/backend
                        - ./data/frontend/log:/var/log/fronte
                        - ./data/mysql/log:/var/log/mysql:ro
                        - ./data/mongo/log:/var/log/mongo:ro
                        - ./data/redis/master/log:/var/log/re
                        - ./data/redis/slave1/log:/var/log/re
                        - ./data/redis/slave2/log:/var/log/re
                        - ./data/redis/sentinel1/log:/var/log
                        - ./data/redis/sentinel2/log:/var/log
                        - ./data/redis/sentinel3/log:/var/log
                        - ./data/jenkins/log:/var/log/jenkins
                        - ./data/redis/insight/log:/var/log/r
                        - ./data/crawl/log:/var/log/crawl:ro
                        - ./data/recommend/log:/var/log/recom
                        - ./data/scheduling/log:/var/log/sche

        recommend:
                image: recommend:latest
                restart: unless-stopped
                expose:
                        - "8001"
                environment:
                        DJANGO_SETTINGS_MODULE: recommend.set
                        TZ: Asia/Seoul
                volumes:
                        - ./data/recommend/log:/var/log/recom
                command: ["python", "manage.py", "runserver",

        scheduling:
                image: scheduling:latest
                restart: unless-stopped
                volumes:
                        - ./data/scheduling/log:/var/log/sche
```

```
                expose:
                        - "8083" # 내부포트 8083
                environment:
                        SPRING_PROFILES_ACTIVE: prod
                        SERVER_PORT: 8083  # 환경 변수로 포트 설정
                        TZ: Asia/Seoul
                        LOG_FILE_NAME: scheduling
                        LOG_FILE_PATH: scheduling



volumes:
        jenkins_home:
        redis-master:
        redis-slave-1:
        redis-slave-2:
        redis-sentinel:
        redis-stack:
        mysql_data:
        redisinsight:
        mongo_data:
```

3.3. Nginx 설정

```
user  nginx;
worker_processes  1;

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;

    log_format main escape=json
        '{'
            '"remote_addr": "$remote_addr", '
            '"remote_user": "$remote_user", '
            '"time_local": "$time_local", '
            '"request": "$request", '
            '"status": "$status", '
            '"body_bytes_sent": "$body_bytes_sent", '
            '"http_referer": "$http_referer", '
            '"http_user_agent": "$http_user_agent", '
            '"http_x_forwarded_for": "$http_x_forwarded_for",
            '"host": "$host", '
            '"server_name": "$server_name", '
            '"request_uri": "$request_uri", '
            '"uri": "$uri", '
            '"request_body": "$request_body", '
            '"args": "$args", '
            '"upstream_addr": "$upstream_addr", '
            '"upstream_status": "$upstream_status", '
            '"request_time": "$request_time"'
        '}';

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

    sendfile        on;
    keepalive_timeout  65;
    # 요청 제한 설정
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=

    # backend_v1과 backend_v2로 각각 설정 (외부와 내부 포트를 통일)
    upstream backend_v1 {
        server newzy-backend-v1-1:8081;
        server newzy-backend-v1-2:8081;
        server newzy-backend-v1-3:8081;
    }
```

```
upstream backend_v2 {
    server newzy-backend-v2-1:8082;
    server newzy-backend-v2-2:8082;
    server newzy-backend-v2-3:8082;
}

upstream frontend_v1 {
    server newzy-frontend-v1-1:3001;
    server newzy-frontend-v1-2:3001;
}

upstream frontend_v2 {
    server newzy-frontend-v2-1:3002;
    server newzy-frontend-v2-2:3002;
}

server {
    listen 80;
    server_name j11b305.p.ssafy.io;
    server_tokens off;

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

    location / {
        return 301 https://$host$request_uri;
    }

}

server {
    listen 443 ssl;
    server_name j11b305.p.ssafy.io;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/j11b305.p.ssafy
```

```
        ssl_certificate_key /etc/letsencrypt/live/j11b305.p.s
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # client_max_body-size 설정 추가
        client_max_body_size 50M;

        # 요청 제한 설정 적용
        location / {
            limit_req zone=mylimit burst=1000 nodelay;

            # 요청이 너무 많을 경우 429 에러 반환
            limit_req_status 429;

                                                # 배포 시 배포 :
            proxy_pass http://frontend_v2;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;

            # 캐시 방지 헤더 설정
            add_header Cache-Control "no-store, no-cache, mus
            add_header Pragma "no-cache";
            add_header Expires "0";

            # CORS 헤더 추가
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, P
            add_header 'Access-Control-Allow-Headers' 'Origin
            add_header 'Access-Control-Allow-Credentials' 'tr
        }

        # Jenkins 리버스 프록시 설정 (8080 포트)
        location /jenkins {
            proxy_pass http://jenkins:8080/jenkins;
            proxy_set_header Host $host:443;
            proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_for
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port 443;

        # 캐시 방지 헤더 설정
        add_header Cache-Control "no-store, no-cache, mus
        add_header Pragma "no-cache";
        add_header Expires "0";

        proxy_http_version 1.1;
    }

    # backend 리버스 프록시 설정
    location /api/ {
        proxy_pass http://backend_v1/api/;  # 기본적으로 ba
        # 배포 시 배포 스크립트에서 backend_v2로 전환할 수 있음

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_for
        proxy_set_header X-Forwarded-Proto $scheme;

        # 캐시 방지 헤더 설정
        add_header Cache-Control "no-store, no-cache, mus
        add_header Pragma "no-cache";
        add_header Expires "0";
    }

    # Kibana 리버스 프록시 설정
    location /kibana/ {
        proxy_pass http://kibana:5601/kibana/;
        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_for
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
            # 캐시 방지 헤더 설정
            add_header Cache-Control "no-store, no-cache, mus
            add_header Pragma "no-cache";
            add_header Expires "0";
        }
    }
}
```

3.4. EC2 Port

Port 번호	내용
22	SSH
80	HTTP (HTTPS로 redirect)
443	HTTPS
3001, 3002	Nginx, React(Docker)
3306	MySQL
5044	logstash
5601	kibana
5540	redisInsight(GUI)
6379	redis-master, redis-slave-1, redis-slave-2
8000	crawl(django)
8001	recommend(django)
8080	Jenkins
8081, 8082	spring boot(api)
9200	elasticsearch
26379	redis-sentinel-1
26380	redis-sentinel-2
26381	redis-sentinel-3
27017	mongoDB

3.5. 방화벽(UFW) 설정

```
ufw 상태 확인
sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
587                        ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
587 (v6)                   ALLOW       Anywhere (v6)

사용할 포트 허용하기
sudo ufw allow 포트번호

ufw 활성화하기
sudo ufw enable

등록한 포트 삭제하기
sudo ufw status numbered
sudo ufw delete 4
```

4. CI/CD 구축

4.1. Jenkins 도커 이미지 + 컨테이너

docker-compose.yml 내부

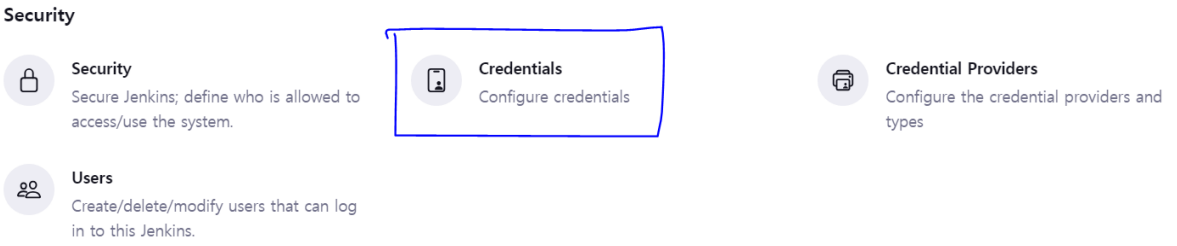
```
        jenkins:
                image: jenkins/jenkins:lts
                restart: unless-stopped
                volumes:
```

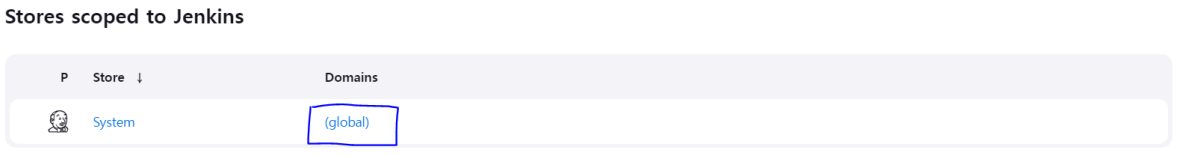
```
      - jenkins_home:/var/jenkins_home
      - ./data/jenkins:/var/jenkins_shared
      - /var/run/docker.sock:/var/run/docke
      - /usr/local/bin/docker-compose:/usr/
      - /usr/bin/docker:/usr/bin/docker  #
      - /home/ubuntu/newzy:/home/ubuntu/new
      - ./data/jenkins/log:/var/jenkins_hom

    environment:
      JENKINS_OPTS: --prefix=/jenkins
      JAVA_OPTS: -Djava.util.logging.config
      TZ: Asia/Seoul
```

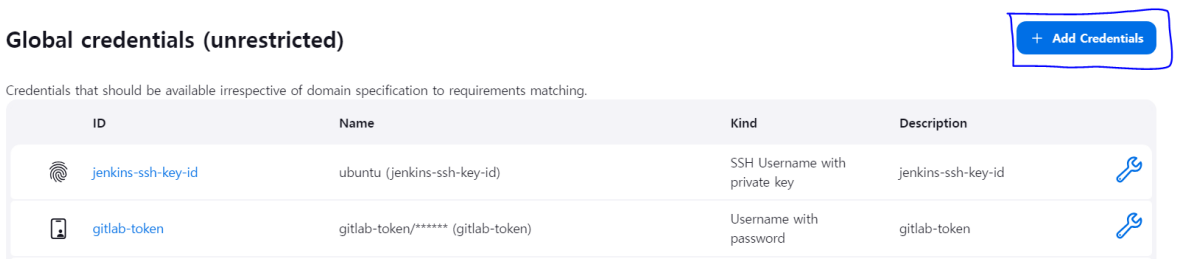
4.2. Jenkins 설정

4.2.1 GitLab Credentials 설정

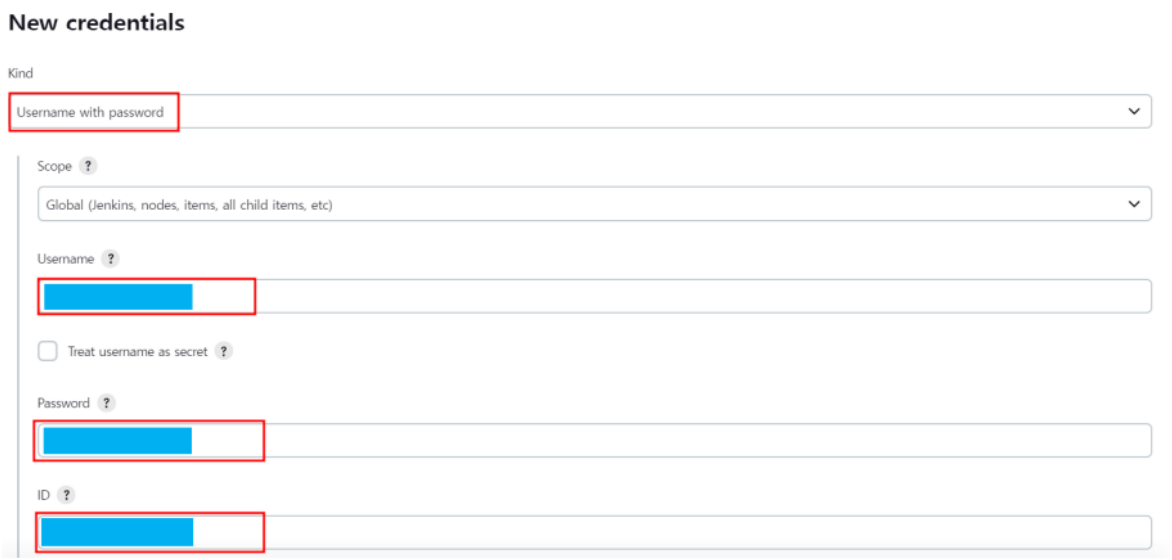
1. 아이디 → “Credentials” 클릭
2. “Store : System” → “(global)” → “+ Add Credentials” 클릭







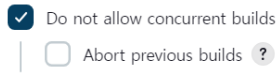
3. “Kind”에 “Username with password” 입력 → “Username”에 GitLab ID 혹은 원하는 ID 입력(gitlab-token) → “Password”에 Gitlab Personal Access Tokens 입력 → “ID”에 임의 아이디 입력(gitlab-token) → 생성
   *** Personal Access Token은 Gitlab > User Settings > Access Tokens에서 생성



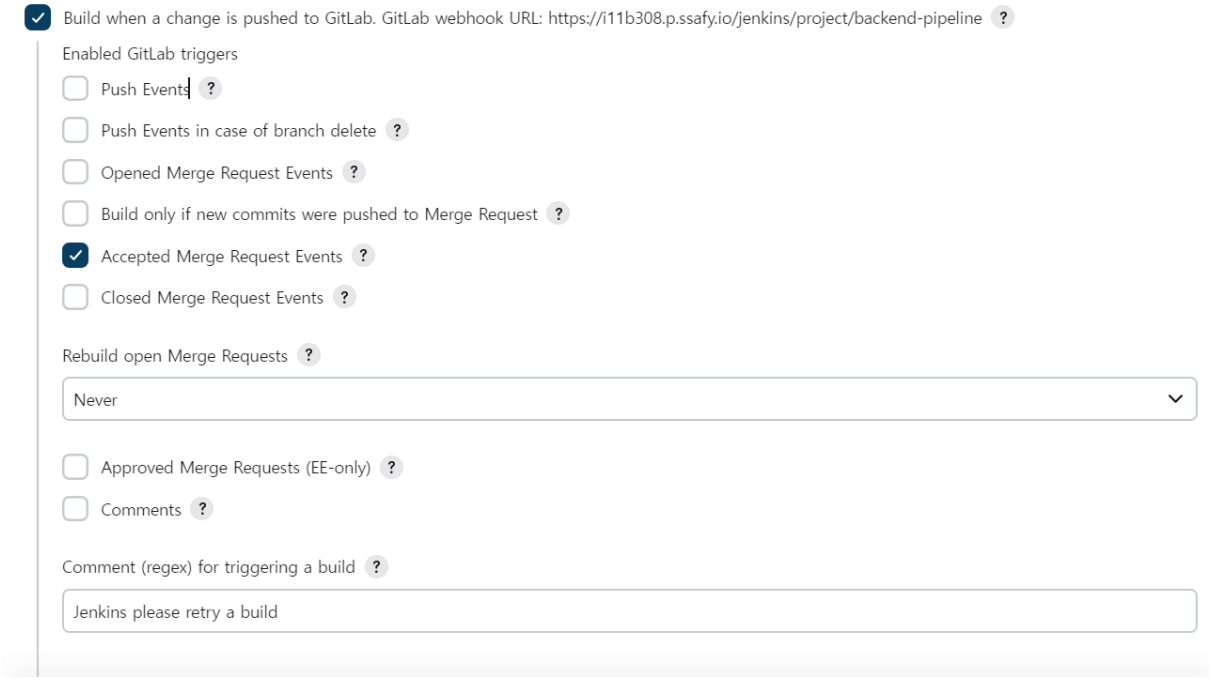
4.2.2 Jenkins Item 생성

1. “새로운 Item” 클릭
2. “Enter an item name”에 임의 Item 이름 입력 → “Pipeline” 클릭

3. “General” → “Do not allow concurrent builds” 클릭
   (한 빌드를 진행중이면 동시에 빌드를 진행하지 않게 한다)



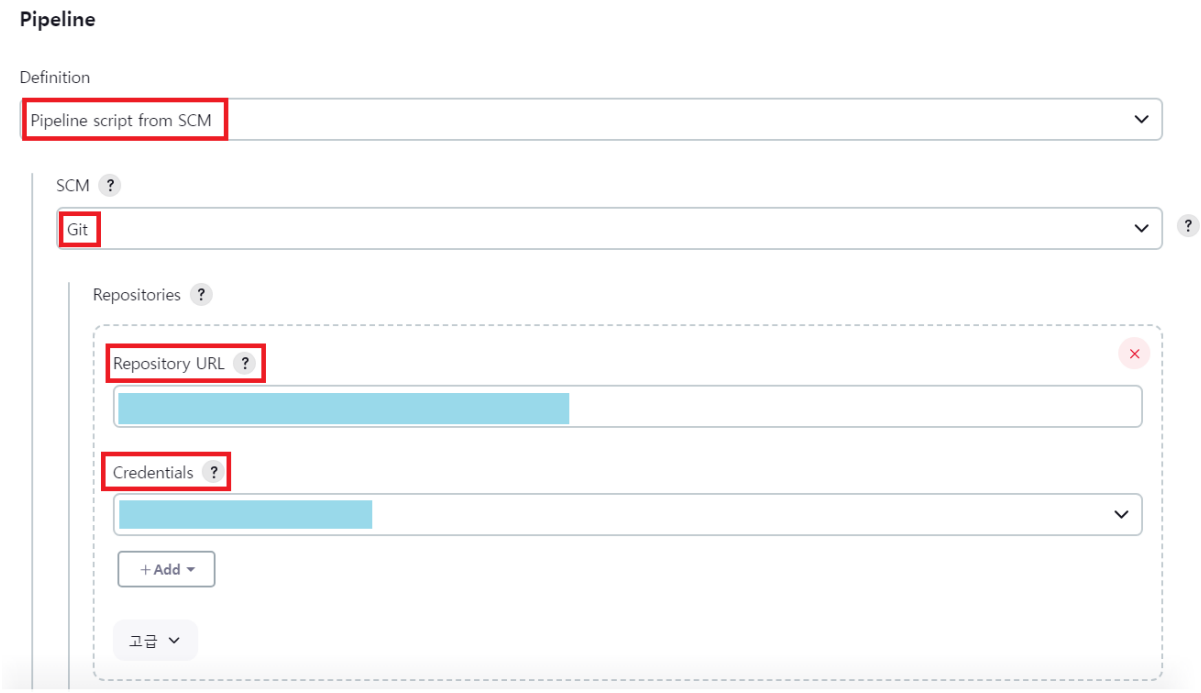
4. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭
   (WebHook 설정 : GitLab 특정 브랜치 merge 시 자동 빌드 + 배포 설정)
   (해당 URL 복사 → WebHook 설정 시 사용 예정)



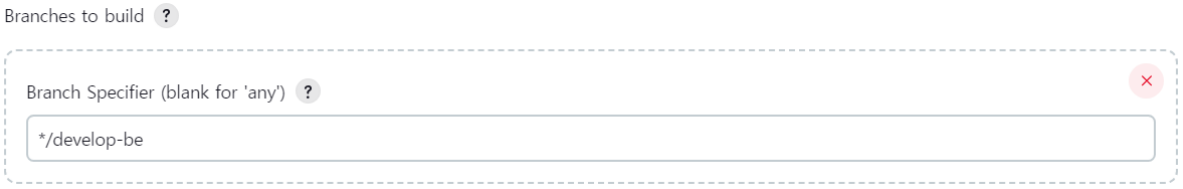
5. “Build when a change is pushed to GitLab” 하위의 “고급...” 클릭

6. 특정 브랜치에서 타겟 브랜치로 머지를 할 경우 빌드 + 배포가 진행되도록 설정
   Secret token의 “Generate” 클릭 후 생성된 토큰값 복사

7. “Pipeline” → “Definition”에 Pipeline script from SCM 설정 → “SCM”에 “Git” 설정 → “Repository URL”에 프로젝트 GitLab URL
   입력 → “Credentials”에 사전에 추가한 Credentials 입력



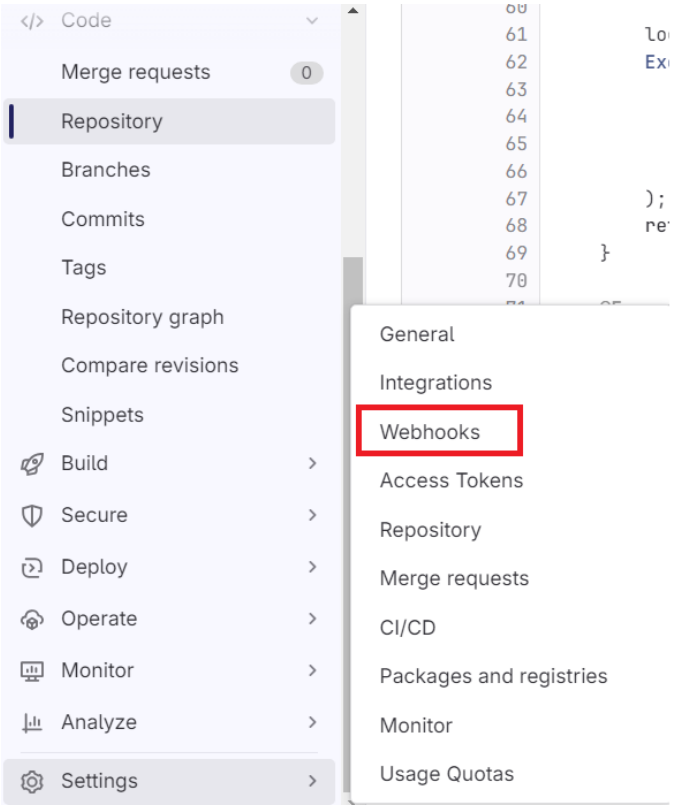
8. “Branch Specifier”에 빌드 할 브랜치명 입력 (master일 시 “*/master)



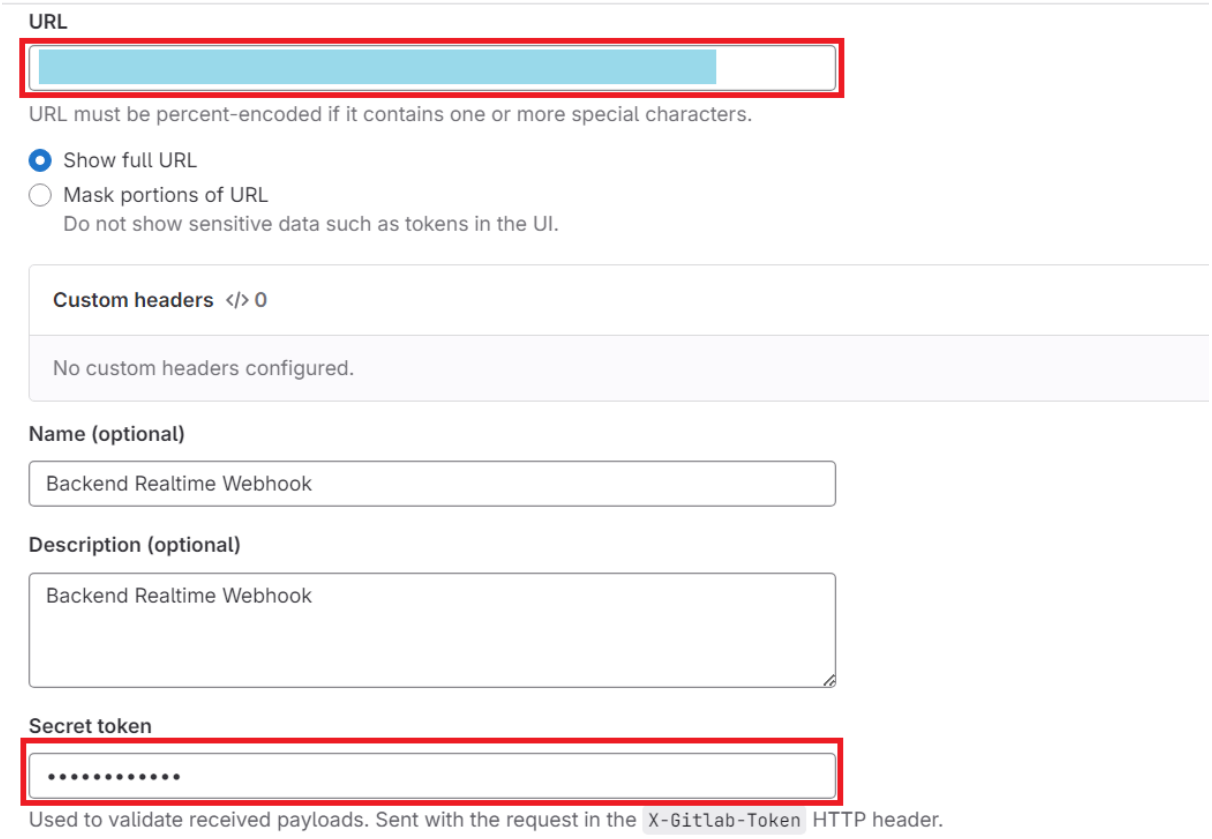
9. “Script Path”에 Jenkinsfile 경로 입력

4.2.3. GitLab Webhook 설정

1. 프로젝트 GitLab → “Settings” → “Webhooks” 클릭



2. “URL”에 사전에 복사해놓은 Jenkins URL 입력 → “Secret token”에 사전에 복사해 놓은 Secret token 입력 → “Merge request events” 클릭
   후 WebHook 적용 브랜치 입력 (Jenkins Branch Specifier과 일치하여야 함)



4.2.4. 빌드 및 배포

> Option 1. 상기 WebHook 설정한 브랜치로 merge

> Option 2. Jenkins 홈 화면 → Jenkins Item 클릭 → “지금 빌드” 클릭

5. Redis 설정

5.1. Redis 설정

docker-compose.yml 내부

```
        # Redis Master 설정
        redis-master:
                image: redis:latest
                restart: unless-stopped
                command: redis-server /usr/local/etc/redis/re
                volumes:
```

```
                - ./data/redis/master:/data
                - ./data/redis/master/log:/var/log/re
                - ./data/redis/master/redis.conf:/usr
        expose:
                - "6379"  # 외부에 노출하지 않음, 내부 네트
        environment:
                TZ: Asia/Seoul
                REDIS_REPLICATION_MODE: master

    # Redis Slave 1 설정
    redis-slave-1:
        image: redis:latest
        restart: unless-stopped
        command: sh -c "redis-server /usr/local/etc/r
        volumes:
                - ./data/redis/slave1:/data
                - ./data/redis/slave1/log:/var/log/re
                - ./data/redis/slave1/redis.conf:/usr
        expose:
                - "6379"  # 외부에 노출하지 않음, 내부 네트
        depends_on:
                - redis-master
        environment:
                TZ: Asia/Seoul
                REDIS_REPLICATION_MODE: slave

    # Redis Slave 2 설정
    redis-slave-2:
        image: redis:latest
        restart: unless-stopped
        command: sh -c "redis-server /usr/local/etc/r
        volumes:
                - ./data/redis/slave2:/data
                - ./data/redis/slave2/log:/var/log/re
                - ./data/redis/slave2/redis.conf:/usr
        expose:
                - "6379"  # 외부에 노출하지 않음, 내부 네트
        depends_on:
```

```
                - redis-master
        environment:
                TZ: Asia/Seoul
                REDIS_REPLICATION_MODE: slave

    # Redis Sentinel 1 설정
    redis-sentinel-1:
        image: redis:latest
        restart: unless-stopped
        command: redis-sentinel /etc/sentinel/sentine
        volumes:
                - ./data/redis/sentinel1/sentinel.con
                - ./data/redis/sentinel1/log:/var/log
        expose:
                - "26379"  # 외부에 노출하지 않음
        depends_on:
                - redis-master
                - redis-slave-1
                - redis-slave-2
        environment:
                TZ: Asia/Seoul
                REDIS_REPLICATION_MODE: sentinel

    # Redis Sentinel 2 설정
    redis-sentinel-2:
        image: redis:latest
        restart: unless-stopped
        command: redis-sentinel /etc/sentinel/sentine
        volumes:
                - ./data/redis/sentinel2/sentinel.con
                - ./data/redis/sentinel2/log:/var/log
        expose:
                - "26380"  # 외부에 노출하지 않음
        depends_on:
                - redis-master
                - redis-slave-1
                - redis-slave-2
        environment:
```

```
                    TZ: Asia/Seoul
                    REDIS_REPLICATION_MODE: sentinel


    # Redis Sentinel 3 설정
    redis-sentinel-3:
            image: redis:latest
            restart: unless-stopped
            command: redis-sentinel /etc/sentinel/sentine
            volumes:
                    - ./data/redis/sentinel3/sentinel.con
                    - ./data/redis/sentinel3/log:/var/log
            expose:
                    - "26381"  # 외부에 노출하지 않음
            depends_on:
                    - redis-master
                    - redis-slave-1
                    - redis-slave-2
            environment:
                    TZ: Asia/Seoul
                    REDIS_REPLICATION_MODE: sentinel

    # Redis Stack 설정 (외부 접근 가능)
    redis-stack:
            image: redis/redis-stack:latest
            restart: unless-stopped
            expose:
                    - "8001"  # 내부에서 사용할 포트
            environment:
                    TZ: Asia/Seoul
                    REDIS_PASSWORD: newzy
            volumes:
                    - ./data/redis/stack:/data
            ports:
                    - "8001:8001"

    redisinsight:
            image: redis/redisinsight:latest
```

```
    restart: unless-stopped
    volumes:
      - ./data/redis/insight:/db
      - ./data/redis/insight/log:/db/logs
    depends_on:
      - redis-master
    environment:
      TZ: Asia/Seoul
```

5.2. Docker redis-cli

```
docker exec -it newzy-redis-master-1 /bin/bash
redis-cli
AUTH newzy
```

5.3. redis.conf

```
logfile "/var/log/redis/master/redis.log"
notify-keyspace-events Ex

# RDB 스냅샷 설정
save 900 1
save 300 10
save 60 10000
dbfilename dump.rdb
dir /data

# AOF 설정
appendonly yes
appendfilename "appendonly.aof"

# 커맨드 변경
rename-command FLUSHALL "reallyreally"
rename-command FLUSHDB "realreal"
```

5.4. sentinel.conf

```
# Sentinel 인스턴스의 포트 설정 (기본 포트 26379)
port 26379

# Sentinel이 모니터링할 Redis 마스터의 설정
# 형식: sentinel monitor <master-name> <master-ip> <master-por
# quorum: 마스터가 다운되었다고 선언하려면 동의해야 하는 Sentinel의 개수
sentinel resolve-hostnames yes
sentinel monitor mymaster redis-master 6379 2

# Sentinel에서 마스터에 연결할 때 사용하는 비밀번호 (마스터에 설정된 req
sentinel auth-pass mymaster newzy

# Redis 마스터가 다운되었다고 판단하는 기준 시간 (밀리초 단위)
# 이 시간을 초과하면 마스터가 다운된 것으로 판단
sentinel down-after-milliseconds mymaster 5000

# 새로운 마스터로 승격된 슬레이브가 기존 마스터로부터 복구한 후
# 클라이언트 요청을 처리하기 시작하는 데 필요한 슬레이브의 동기화 시간 (밀리
sentinel failover-timeout mymaster 60000

# 최소 슬레이브 개수 설정 (최소 1개의 슬레이브가 마스터와 동기화되어야 함)
sentinel parallel-syncs mymaster 1

# Redis Sentinel 로그 파일 위치 (옵션)
logfile "/var/log/redis/sentinel.log"
```

6. ELK 설정

6.1. filebeat 설정

```
filebeat.inputs:
  # Django 로그 - JSON 형식
  - type: log
    enabled: true
    paths:
      - /var/log/crawl/*.log
```

```
    exclude_files: ['.gz$']
    json.keys_under_root: true
    json.add_error_key: true
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  - type: log
    enabled: true
    paths:
      - /var/log/recommend/*.log
    exclude_files: ['.gz$']
    json.keys_under_root: true
    json.add_error_key: true
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  # Spring 로그 - 멀티라인 설정 적용
  - type: log
    enabled: true
    paths:
      - /var/log/backend/*.log
    exclude_files: ['.gz$']
    multiline.pattern: '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\
    multiline.negate: true
    multiline.match: after
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  # Spring 로그 - 멀티라인 설정 적용
```

```
  - type: log
    enabled: true
    paths:
      - /var/log/scheduling/*.log
    exclude_files: ['.gz$']
    multiline.pattern: '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\
    multiline.negate: true
    multiline.match: after
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  # Redis Sentinel 로그 수집 - 커스텀 입력
  - type: log
    enabled: true
    paths:
      - /var/log/redis/sentinel1/sentinel1.log
      - /var/log/redis/sentinel2/sentinel2.log
      - /var/log/redis/sentinel3/sentinel3.log
    fields_under_root: true
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  # Jenkins 로그 수집 - 커스텀 입력
  - type: log
    enabled: true
    paths:
      - /var/log/jenkins/jenkins.log
    fields:
      service: jenkins
    fields_under_root: true
    processors:
      - add_docker_metadata: ~
```

```
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  - type: log
    enabled: true
    paths:
      - "/var/log/redis/master/*.log"  # 마스터 로그 파일
    fields:
      service_type: redis-master
    fields_under_root: true
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  # nginx 로그
  - type: log
    enabled: true
    paths:
      - /var/log/nginx/access.log   # NGINX access log 경로
    # 로그 포맷이 JSON 형태일 경우 JSON 파싱
    json.keys_under_root: true      # JSON 필드를 루트에 두고 개별
    json.add_error_key: true        # JSON 파싱 오류 시 추가
    json.message_key: message       # JSON 데이터가 포함된 필드 설
    fields:
      service_type: nginx-access
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

  - type: log
    enabled: true
    paths:
      - /var/log/nginx/error.log    # NGINX error log 경로
```

```
    # json.keys_under_root: true
    # json.add_error_key: true
    # json.message_key: message
    fields:
      service_type: nginx-error
    processors:
      - add_docker_metadata: ~
      - add_host_metadata:
          when.not.contains.tags: forwarded
      - add_cloud_metadata: ~

filebeat.modules:
  - module: nginx
    access:
      enabled: false
      var.paths: ["/var/log/nginx/access.log"]
    error:
      enabled: false
      var.paths: ["/var/log/nginx/error.log"]

  - module: mysql
    error:
      enabled: true
      var.paths: ["/var/log/mysql/error.log"]
    slowlog:
      enabled: true
      var.paths: ["/var/log/mysql/mysql-slow.log"]

  - module: mongodb
    log:
      enabled: true
      var.paths: ["/var/log/mongo/*.log"]
    audit:
      enabled: false
    slowlog:
      enabled: false
```

```
processors:
  - add_docker_metadata: ~
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~

output.logstash:
  hosts: ["logstash:5044"]
```

6.2. logstash 설정

```
input {
  beats {
    port => 5044
  }
}

filter {

  date {
    match => ["[timestamp]", "ISO8601"]  # 타임스탬프 필드가 'tim
    timezone => "Asia/Seoul"  # KST로 시간대 변환
  }

  # 서비스 이름 추출 (예: /var/log/nginx/error.log 에서 'nginx' 추
  grok {
    match => { "[log][file][path]" => "/var/log/(?<service_pa
  }

  # 새로운 필드 'service_type'을 초기화
  mutate {
    add_field => { "service_type" => "%{service_path}" }
  }

  # Redis 관련 서비스는 역할에 따라 'redis-<role>' 형식으로 변경
  if [service_path] == "redis" {
    grok {
```

```
      match => { "[log][file][path]" => "/var/log/redis/(?<ro
    }

    mutate {
      replace => { "service_type" => "redis-%{role}" }
    }
    mutate {
      remove_field => ["role"]
    }
  }

  # MongoDB 서비스의 경우 'service_type'을 'mongodb'로 설정
  if [service][type] == "mongodb" {
    mutate {
      replace => { "service_type" => "mongodb" }
    }
  }

  if [service_type] == "nginx" {
    # 불필요한 필드 제거
    mutate {
      remove_field => [
        "agent.ephemeral_id",
        "agent.id",
        "agent.name",
        "cloud.account.id",
        "cloud.image.id",
        "cloud.machine.type",
        "cloud.provider",
        "ecs.version",
        "fileset.name",
        "host.mac",
        "host.architecture",
        "host.containerized",
        "host.os.codename",
        "host.os.family",
        "host.os.kernel"
      ]
```

```
    }

    if [fields][service_type] == "nginx-access" {
      json {
        source => "message"
      }
      # 필요시 message 필드 제거
      mutate {
        remove_field => ["message"]
      }
    } else {
      # error log의 grok 패턴을 적용해 필드를 분리
      grok {
        match => {
          "message" => '%{TIMESTAMP_ISO8601:timestamp} \[%{WO
        }
      }
      # 필요시 message 필드 제거
      mutate {
        remove_field => ["message"]
      }
    }
  }

    # backend 로그 처리
  if [service_type] == "backend" {
    grok {
      match => {
        # 로그 형식: 2024-10-07 17:19:25.853 [http-nio-0.0.0.0-
        "message" => "%{TIMESTAMP_ISO8601:log_timestamp} \[%{
      }
    }

    # 불필요한 필드 제거 (선택 사항)
    mutate {
      remove_field => [
        "message", "agent.ephemeral_id", "agent.id", "agent.n
        "cloud.image.id", "cloud.machine.type", "cloud.provid
```

```
        "fileset.name", "host.mac", "host.architecture", "hos
        "host.os.codename", "host.os.family", "host.os.kernel
      ]
    }
  }
  # 기타 서비스에 대한 필터링이 필요할 경우 여기에 추가
}

output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "%{service_type}-%{+YYYY.MM.dd}"  # 'service_typ
  }
  # stdout { codec => rubydebug }
}
```

7. DB 덤프 파일 최신본

newzydump.zip

8. 시연 시나리오

8.1 홈화면

- 비로그인 상태
  - 데일리 퀴즈 접속 안 됨
  - 소셜 로그인 → 어휘력 테스트
- 로그인 상태
  - 데일리 기사 / 퀴즈 접근 가능
  - 워드 클라우드 조회
  - 랭킹(지난 주의 카드왕 / 뉴포터, 지난 날 뉴스와 뉴지 조회수) 조회

8.2 마이 페이지

- 경험치, 획득한 뉴스 카드 목록 조회
- 팔로우 / 팔로잉 목록 조회

  → 다른 유저 프로필 조회
- 북마크한 뉴스 및 뉴지 조회
- '나만의 단어장' 조회

8.3 뉴스 목록

- 최신순 / 인기순 정렬
- 제목 키워드 검색
- 경제 / 사회 / 세계 카테고리 필터링

8.4 뉴스 디테일

- 어휘 검색 → '나만의 단어장' 저장
- 카드 획득
- 좋아요 / 북마크

8.5 뉴지 목록

- 최신순 정렬 / 구독한 사용자가 작성한 뉴지 모아보기
- 제목 키워드 검색
- 시사 / 문화 / 자유 카테고리 필터링

8.6 뉴지 디테일

- 뉴지 게시글 작성
- 좋아요 / 북마크 / 댓글 / 어휘검색
- 뉴지 작성자 구독 버튼

8.7 마이 페이지

- 경험치 늘어난 거 확인
- '나만의 단어장' 확인 → 단어 테스트 진행
- 획득한 뉴스 카드 리스트 확인