



포팅 메뉴얼

개발 환경

[프로젝트 기술 스택](#)

환경변수 설정

[Frontend : .env](#)

[Backend : application.yml 등](#)

[env.properties](#)

설정 파일

[main server](#)

[docker-compose-prod.yml](#)

[spring Dockerfile](#)

[react](#)

[elastic, spark master 서버](#)

[docker-compose-elastic.yml](#)

[elasticsearch Dockerfile](#)

[kibana](#)

[logstash](#)

계정 정보 및 덤프 파일

➔ [로컬 MySQL 접속 계정 정보](#)

➔ [EC2 MySQL 접속 계정 정보](#)

➔ [Jenkins 접속 계정 정보](#)

배포

[docker를 ec2에 설치](#)

[repository 설정](#)

[docker engine 설치](#)

[jenkins 이미지 설치](#)

[jenkins 접속](#)

[gitlab 플러그인 설치](#)

[credential 설정\(인증 설정\)](#)

[아이템 생성\(프로젝트 생성이라고 생각하면 될듯\)](#)

프로젝트 local 실행 가이드

1. Backend 실행 가이드

(1) IntelliJ [2023.1.3](#) 설치하기

(2) Java 17 설치

(3) MySQL [8.0.33](#) 설치

(4) Project Git Clone

(5) Springboot Application 실행시키기

2. Frontend 실행 가이드

(1) Visual Studio Code(VSC) 설치

(2) Project Git Clone

(3) Node.js [18.16.1](#) 설치

(4) 패키지 설치

(5) 프로젝트 실행

빅데이터

Hadoop 실행 가이드

0. 기본 환경 설정

1. Java 11 설치(master, worker1 ...)

2. hadoop 3.3.6 설치(master)

3. 설정파일

4. 실행

Spark

1. spark 3.4.1 설치(master)

2. 설정 파일

3. 실행

4. Spark 프로그램 실행

Elasticsearch

외부 서비스

Kakao karlo(그림 ai)

naver clova(감정 분석)

- (1) <https://www.ncloud.com/> 접속
- (2) 회원가입 후 콘솔 눌러 접속하기
- (3) 어플리케이션 추가하기
- (4) 사용하기

시연 시나리오

시연 화면

1. Intro 페이지
2. LoginPage
3. MainPage
4. CreateDiaryPage
5. DreamShopPage
6. GalleryPage
7. CloudPage
8. StatisticsPage
9. MyPage

개발 환경

프로젝트 기술 스택

- Frontend
 - Typescript : 4.9.5
 - React : 18.2.0
 - ReactQuery : 3.39.3
 - ReactRouter : 6.15.0
 - React-progressive-graceful-image : 0.7.0
 - Redux : 8.1.2
 - Three.js : 0.156.0
 - aixos : 1.5.0
 - lodash : 4.17.21
 - openai : 4.10.0
- backend
 - IntelliJ(IDE) : 2023.1.3
 - Java : 11, 17
 - SpringBoot : 3.1.3
 - Gradle
 - ORM : JPA(Hibernate)
 - Spring Security
 - java-jwt : 0.11.5
 - Swagger : 2.0.2
 - spark: 3.4.1
 - hadoop(hdfs): 3.3.6
 - kafka: 3.5.1
 - zookeeper: 3.9.0
 - elasticsearch: 1.13.2
 - logstash: 7.10.2

- kibana: 1.13.2
- CI/CD
 - AWS EC2
 - Nginx : docker hub nginx tag - mainline-alpine3.18-slim
 - Ubuntu : Ubuntu 20.04 LTS
 - Docker : 24.0.6
 - Jenkins : docker hub jenkins/jenkins tag - jdk17
- Cooperation
 - Notion
 - MatterMost
 - Discord, Webex
 - Figma
 - Git-Flow
 - Github

환경변수 설정

Frontend : .env

- 환경변수 설정 위치

```
frontend
└─ .env
```

- .env (Frontend 프로젝트 설정)

```
WDS_SOCKET_PORT=0
GENERATE_SOURCEMAP=false
REACT_APP_KAKAO_KEY="" // 비밀
REACT_APP_GPT_KEY="" // 비밀
```

Backend : application.yml 등

- 환경변수 설정 위치

```
backend
└─ back
    └─ dream
        └─ src
            └─ main
                └─ resources
                    └─ properties
                        ├── application.yml
                        ├── application-local.yml
                        └─ application-prod.yml
```

- application.yml (Backend 프로젝트 설정)

```
server:
  port: 8080

spring:
  ##### method #####
  profiles:
    active: prod
  #####
  message:
```

```

topic:
  sparkDiaryName: spark_diary
  sparkListenerName: spark_diary_result
  sparkDiaryStatisticName: statistics
  pointName: point_log
  diaryName: mysql_diary
  recommendName: log_recommend
  statisticDailyName: mysql_daily_statistic
  statisticMonthName: mysql_month_statistic
  transactionName: log_transaction
  strictName: log_strict

```

- **application-local.yml (local 설정)**

```

server:
  port: 8082

spring:
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect

  datasource:
    url: jdbc:mysql://127.0.0.1:3306/dream?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    username: root
    password: 7818
    driver-class-name: com.mysql.cj.jdbc.Driver

  security:
    user:
      password: 1234

  data:
    redis:
      lettuce:
        pool:
          max-active: 10
          max-idle: 10
          min-idle: 2

      port: 6379
      host: 127.0.0.1

  servlet:
    multipart:
      enabled: true
      location: ./media/img/

  mvc:
    static-path-pattern: /static/**

  kafka:
    bootstrap-servers: localhost:9094

  consumer:
    # consumer bootstrap servers가 따로 존재하면 설정
    # bootstrap-servers: 192.168.0.4:9092

    # 식별 가능한 Consumer Group Id
    group-id: loggroup
    # Kafka 서버에 초기 offset이 없거나, 서버에 현재 offset이 더 이상 존재하지 않을 경우 수행할 작업을 설정
    # latest: 가장 최근에 생산된 메시지로 offset reset
    # earliest: 가장 오래된 메시지로 offset reset
    # none: offset 정보가 없으면 Exception 발생
    auto-offset-reset: earliest
    # 데이터를 받을 때, key/value를 역직렬화
    # JSON 데이터를 받을 것이라면 JsonDeserializer
    #   key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    #   value-deserializer: org.apache.kafka.common.serialization.StringDeserializer

    # producer:
    # producer bootstrap servers가 따로 존재하면 설정
    # bootstrap-servers: 3.34.97.97:9092

```

```

# 데이터를 보낼 때, key/value를 직렬화
# JSON 데이터를 보낼 것이라면 JsonDeserializer
#   key-serializer: org.apache.kafka.common.serialization.StringSerializer
#   value-serializer: org.apache.kafka.common.serialization.StringSerializer

springdoc:
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  paths-to-match: /**
  swagger-ui:
    path: /
    display-request-duration: true
    groups-order: desc
    operations-sorter: alpha
    disable-swagger-default-url: true
  api-docs:
    groups:
      enabled: true

logging:
  level:
    org.hibernate.sql: debug
    org.hibernate.type: trace

jwt:
  header: Authorization
  #HS512 알고리즘을 사용할 것이기 때문에 512bit, 즉 64byte 이상의 secret key를 사용해야 한다.
  secret: ${secret}

token:
  token-validity-in-seconds: 86400 # ttl (초)
  access-token-expiration-sec: 1800
  refresh-token-expiration-sec: 1209600

app:
  fileupload:
    uploadPath: C:\SSAFY\dream
    uploadDir: image
    readPath: C:\SSAFY\dream

elk:
  host: "localhost:9200"

```

- application-prod.yml (배포 설정)

```

server:
  port: 8082

spring:
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect

  datasource:
    url: jdbc:mysql://13.124.105.1:3306/dream?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    username: root
    password: 7818
    driver-class-name: com.mysql.cj.jdbc.Driver

  security:
    user:
      password: ssafy124

  data:
    redis:
      lettuce:
        pool:
          max-active: 10
          max-idle: 10
          min-idle: 2

    port: 6379
    host: 13.124.105.1

```

```

servlet:
  multipart:
    enabled: true
    location: ./media/img/

mvc:
  static-path-pattern: /static/**

kafka:
  bootstrap-servers: 13.124.105.1:9094

  consumer:
    # consumer bootstrap servers가 따로 존재하면 설정
    # bootstrap-servers: 192.168.0.4:9092

    # 식별 가능한 Consumer Group Id
    group-id: loggroup
# Kafka 서버에 초기 offset이 없거나, 서버에 현재 offset이 더 이상 존재하지 않을 경우 수행할 작업을
# 설정
# latest: 가장 최근에 생산된 메시지로 offset reset
# earliest: 가장 오래된 메시지로 offset reset
# none: offset 정보가 없으면 Exception 발생
auto-offset-reset: earliest
# 데이터를 받아들 때, key/value를 역직렬화
# JSON 데이터를 받아들 것이라면 JsonDeserializer
#   key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
#   value-deserializer: org.apache.kafka.common.serialization.StringDeserializer

#   producer:
# producer bootstrap servers가 따로 존재하면 설정
# bootstrap-servers: 3.34.97.97:9092

# 데이터를 보낼 때, key/value를 직렬화
# JSON 데이터를 보낼 것이라면 JsonSerializer
#   key-serializer: org.apache.kafka.common.serialization.StringSerializer
#   value-serializer: org.apache.kafka.common.serialization.StringSerializer

springdoc:
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  paths-to-match: /**
  swagger-ui:
    path: /
    display-request-duration: true
    groups-order: desc
    operations-sorter: alpha
    disable-swagger-default-url: true
  api-docs:
    groups:
      enabled: true

logging:
  level:
    org.hibernate.sql: debug
    org.hibernate.type: trace

jwt:
  header: Authorization
  #HS512 알고리즘을 사용할 것이기 때문에 512bit, 즉 64byte 이상의 secret key를 사용해야 한다.
  secret: ${secret}

  token:
    token-validity-in-seconds: 86400 # ttl (초)
    access-token-expiration-sec: 1800
    refresh-token-expiration-sec: 1209600

app:
  fileupload:
    uploadPath: /home/ubuntu
    uploadDir: image
    readPath: //home/ubuntu

elk:
  host: "43.201.35.196:9200"

```

env.properties

- 위치

```
graph TD
    backend --> back
    back --> dream
    dream --> src
    src --> main
    main --> resources
    resources --> properties
    properties --> env_properties[env.properties]
```

[illegible]

설정 파일

main server

docker-compose-prod.yml

- 위치

```
backend/back/dream
```

- docker-compose-prod.yml

```
version: '3' # docker-compose 버전 지정

services: # docker-compose의 경우 docker 컨테이너로 수행될 서비스들은 services 하위에 기술

  mysql:
    image: mysql
    container_name: mysql
    volumes:
      - ./app/server/mysql/
    environment:
      MYSQL_DATABASE: dream
      MYSQL_ROOT_PASSWORD: ssafy124
    ports:
      - "3306:3306"
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --default-authentication-plugin=mysql_native_password # 추가한 부분
    restart: always

  redis:
    image: redis
    container_name: redis
    ports:
      - 6379:6379
    restart: always

  spring:
    build:
      context: ./
      dockerfile: Dockerfile
    container_name: spring
    volumes:
      - ./app/server/dream/dream-spring/
      - /home/ubuntu/image/:/home/ubuntu/image/
    ports:
      - 8082:8082
    expose:
      - 8082
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/dream
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: ssafy124

  depends_on:
    - mysql
```

```

- redis
links:
- mysql
- redis
restart: always

zookeeper: # 서비스 이름. service 하위에 작성하면 해당 이름으로 동작
image: bitnami/zookeeper:3.9.0
container_name: zookeeper
environment:
- ALLOW_ANONYMOUS_LOGIN=yes
ports: # 외부포트: 컨테이너내부포트
- "2181:2181"

kafka:
image: bitnami/kafka:3.5.1
container_name: kafka
ports: # 외부포트: 컨테이너내부포트
- "9094:9094"
- "9095:9095"
environment: # kafka 브로터를 위한 환경 변수 지정
KAFKA_ADVERTISED_LISTENERS: INSIDE://:9092,OUTSIDE://13.124.105.1:9094,SPARK://172.26.1.149:9095
KAFKA_LISTENERS: INSIDE://:9092,OUTSIDE://0.0.0.0:9094,SPARK://0.0.0.0:9095
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT,SPARK:PLAINTEXT
KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
KAFKA_CREATE_TOPICS: "member_log:1:1"
KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
volumes:
- /var/run/docker.sock:/var/run/docker.sock
depends_on:
- zookeeper
- spring

kafka-ui:
image: provectuslabs/kafka-ui
container_name: kafka-ui
ports:
- "8089:8089"
restart: always
environment:
- KAFKA_CLUSTERS_0_NAME=local
- KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9092
- KAFKA_CLUSTERS_0_ZOOKEEPER=zookeeper:2181
depends_on:
- kafka

```

spring Dockerfile

- 위치

backend/back/dream

- Dockerfile

```

FROM openjdk:17-alpine

WORKDIR /usr/src/app

ARG JAR_PATH=./build/libs

COPY ./build/libs/dream-0.0.1-SNAPSHOT.jar /build/libs/dream-0.0.1-SNAPSHOT.jar

CMD ["java", "-jar", "/build/libs/dream-0.0.1-SNAPSHOT.jar"]

```

react

- 위치

frontend/front

- Dockerfile


```

FROM node:16 as builder

# 작업 폴더를 만들고 npm 설치
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install
#RUN npm install -g npm@10.2.0
RUN npm install react-scripts@3.4.1 -g --silent

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build

FROM nginx:latest
# nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm -rf /etc/nginx/conf.d
COPY conf /etc/nginx

# 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/build /usr/share/nginx/html

# 80포트 열고 nginx 실행
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]

```

elastic, spark master 서버

docker-compose-elastic.yml

- 위치

```
back/elastic
```

- docker-compose-elastic.yml

```

version: '3' # docker-compose 버전 지정

services: # docker-compose의 경우 docker 컨테이너로 수행될 서비스들은 services 하위에 기술

  elasticsearch:
    restart: unless-stopped
    build: ./elasticsearch
    container_name: elasticsearch
    volumes:
      - ./elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml
      - ./elasticsearch/config/dict.txt:/usr/share/elasticsearch/config/dict.txt
      - ./elasticsearch/data:/usr/share/elasticsearch/data
    environment:
      - TZ=Asia/Seoul
      - opendistro_security.disabled=true
      #   - ELASTIC_PASSWORD=elasticpassword
      - node.name=elasticsearch
      - bootstrap.memory_lock=true
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - opendistro_security.ssl.http.enabled=false
      - opendistro_security.ssl.transport.enabled=false
    ulimits:
      memlock:
        soft: -1
        hard: -1
    ports:
      - "9200:9200"

  kibana:
    restart: unless-stopped
    build: ./kibana
    container_name: kibana
    environment:
      - TZ=Asia/Seoul
    volumes:
      - ./kibana/config/kibana.yml:/usr/share/kibana/config/kibana.yml:ro,z
    ports:
      - "5601:5601" ## kibana의 기본포트는 5601 이다.
    depends_on:

```

```

- elasticsearch

logstash:
  restart: unless-stopped
  build: ./logstash
  container_name: logstash
  volumes:
    - ./logstash/config/logstash.yml:/usr/share/logstash/config/logstash.yml
    - ./logstash/pipeline/logstash.conf:/usr/share/logstash/pipeline/logstash.conf
    - ./logstash/drivers:/opt/logstash/vendor/jar/jdbc/
  environment:
    - TZ=Asia/Seoul
    - "LS_JAVA_OPTS=-Xmx256m -Xms256m"
  ports:
    - "5044:5044"
  depends_on:
    - elasticsearch

```

elasticsearch Dockerfile

- 위치

```
back/elastic/elasticsearch
```

- Dockerfile

```

#FROM elasticsearch:7.16.3
FROM amazon/opendistro-for-elasticsearch:1.13.2
RUN bin/elasticsearch-plugin install --batch analysis-nori ## 한글 형태소 분석기

# Elasticsearch configuration file from host to image
COPY ./config/elasticsearch.yml /usr/share/elasticsearch/config/elasticsearch.yml

RUN ./bin/elasticsearch-plugin install --batch repository-azure

```

kibana

- 위치

```
back/elastic/kibana
```

- Dockerfile

```

#FROM kibana:7.16.3
FROM amazon/opendistro-for-elasticsearch-kibana:1.13.2
# open distro
RUN /usr/share/kibana/bin/kibana-plugin remove opendistroSecurityKibana
COPY ./config/kibana.yml /usr/share/elasticsearch/config/kibana.yml

```

logstash

- 위치

```
back/elastic/logstatsh
```

- Dockerfile

```

#FROM logstash:7.16.3
FROM docker.elastic.co/logstash/logstash-oss:7.10.2

COPY ./config/logstash.yml /usr/share/logstash/config/logstash.yml
COPY ./pipeline/logstash.conf /usr/share/logstash/pipeline/logstash.conf

RUN bin/logstash-plugin install logstash-output-opsgenie
RUN bin/logstash-plugin install logstash-output-opensearch
RUN bin/logstash-plugin install logstash-integration-jdbc

```

- logstash conf 파일 위치

back/elastic/logstatsh

- logstash conf 파일 설정

```
input {
##### 카프카

##### 사용자 이벤트 로그 기반 추천용 #####
kafka {
  bootstrap_servers => "172.26.1.149:9095"
  group_id => "loggroup"
  topics => ["log_recommend"]
  consumer_threads => 1
  decorate_events => true
  type => "log_recommend"
}

##### 일별 키워드 통계 #####
kafka {
  bootstrap_servers => "172.26.1.149:9095"
  group_id => "loggroup"
  topics => ["mysql_daily_statistic"]
  consumer_threads => 1
  decorate_events => true
  type => "mysql_daily_statistic"
}

##### 월별 키워드 통계 #####
kafka {
  bootstrap_servers => "172.26.1.149:9095"
  group_id => "loggroup"
  topics => ["mysql_month_statistic"]
  consumer_threads => 1
  decorate_events => true
  type => "mysql_month_statistic"
}

##### 거래 내역 이상탐지 #####
kafka {
  bootstrap_servers => "172.26.1.149:9095"
  group_id => "loggroup"
  topics => ["log_transaction"]
  consumer_threads => 1
  decorate_events => true
  type => "log_transaction"
}

##### 잔디 깎기 #####
kafka {
  bootstrap_servers => "172.26.1.149:9095"
  group_id => "loggroup"
  topics => ["log_strict"]
  consumer_threads => 1
  decorate_events => true
  type => "log_strict"
}

##### mysql jdbc -> mysql 과 elasticsearch 연동
##### emotion 통계 및 검색 엔진
jdbc {
  jdbc_driver_library => "/opt/logstash/vendor/jar/jdbc/mysql-connector-java-8.0.28.jar"
  jdbc_driver_class => "com.mysql.jdbc.Driver"
  jdbc_connection_string => "jdbc:mysql://172.26.1.149:3306/dream"
  jdbc_user => "root"
  jdbc_password => "ssafy124"
  jdbc_paging_enabled => true
  tracking_column => "unix_ts_in_secs_1"
  use_column_value => true
  tracking_column_type => "numeric"
  last_run_metadata_path => "/usr/share/logstash/.logstash_jdbc_last_run_1"
  schedule => "* * * * *"
  jdbc_validate_connection => true
  statement => "SELECT D.*, UNIX_TIMESTAMP(D.created_at) AS unix_ts_in_secs_1 FROM diaries AS D WHERE D.created_at > FROM_UNIXTIME(::"
#   statement => "SELECT * FROM diaries"
  type => "mysql_diary"
}
}
```

```

filter {
  json {
    source => "message"
  }
}

output {
  if [type] == "log_recommend" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "log_recommend"
    }
  }

  if [type] == "mysql_daily_statistic" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "mysql_daily_statistic"
    }
  }

  if [type] == "mysql_month_statistic" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "mysql_month_statistic"
    }
  }

  if [type] == "log_transaction" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "log_transaction"
    }
  }

  if [type] == "mysql_diary" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "mysql_diary"
    }
  }

  if [type] == "log_strict" {
    elasticsearch{
      hosts => "elasticsearch:9200"
      index => "log_strict"
    }
  }
}
}

```

계정 정보 및 덤프 파일

→ 로컬 MySQL 접속 계정 정보

- ID : root
- PW : root

→ EC2 MySQL 접속 계정 정보

- ID : root
- PW : ssafy124
- RDS 덤프 파일

→ Jenkins 접속 계정 정보

- ID : admin
- PW : 78957895
- Jenkins URL : <http://9e206.p.ssafy.io:8080/>

배포

docker를 ec2에 설치

- docker 공식 문서: <https://docs.docker.com/engine/install/ubuntu/>

도커를 새로운 장치에 설치하려면 docker repository를 설정해야한다. 그리고 그 repository에 도커를 설치하거나 업데이트 할 수 있다.

repository 설정

1. apt 패키지를 업데이트하고 레포지토리에서 https를 사용할 수 있게 해주는 패키지를 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
```

2. Add Docker's official GPG key:

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
echo \
"deb [arch=$(dpkg --print-architecture)] signed-by=/etc/apt/keyrings/docker.gpg https://download.docker.com/linux/ubuntu \
${. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

docker engine 설치

1. 최신 버전 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

2. 설치 잘 됐는지 확인

```
sudo docker run hello-world
```

jenkins 이미지 설치



```
docker pull jenkins/jenkins
```

jenkins 접속

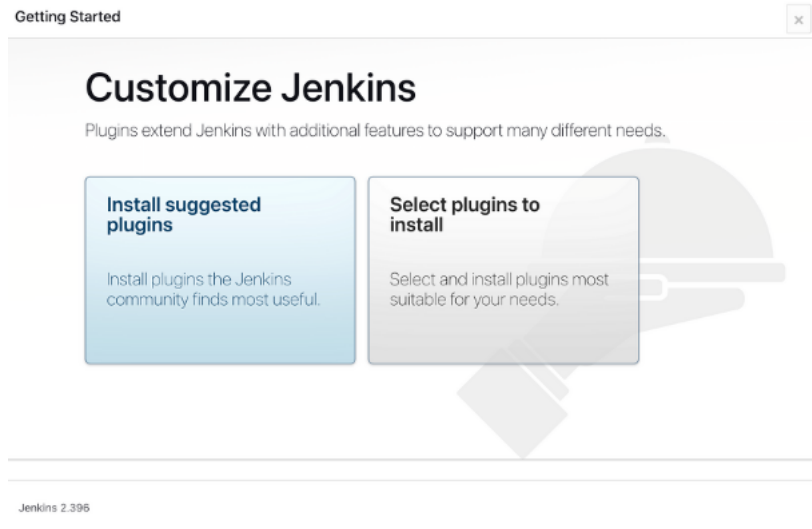
```
docker run -d --name jenkins --restart=on-failure \
-p 8080:8080 \
-v /var/jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-e TZ=Asia/Seoul \
-u root \
jenkins/jenkins
```

- **-restart** -> on-failure 옵션은 비정상 종료시 컨테이너를 재실행합니다.
- **p** -> 외부 접속을 위해 호스트의 8080 포트를 바인딩 해주었습니다.
- **v** -> 호스트의 var/jenkins 디렉토리를 호스트 볼륨으로 설정하여 jenkins 컨테이너의 home 디렉토리에 마운트시켰습니다.

docker.sock 파일은 도커 데몬과 통신할 수 있는 소켓 파일입니다. docker.sock 파일을 컨테이너에 마운트시켜서 도커 명령을 실행할 수 있게 해줍니다. 이러한 방식을 dood(docker out of docker)라고 합니다.

-  -> 젠킨스의 timezone을 KST 기준으로 설정해줍니다.
-  -> 추후 권한 문제가 발생할 수 있기 때문에 user 옵션을 root 사용자로 주었습니다.

```
cat /var/jenkins_home/secrets/initialAdminPassword // 비밀번호
```



- 일단 왼쪽 경로 설치. 필요한 건 나중에 설치 가능

계정명

whwhdnfl2

암호

.....

암호 확인

.....

이름

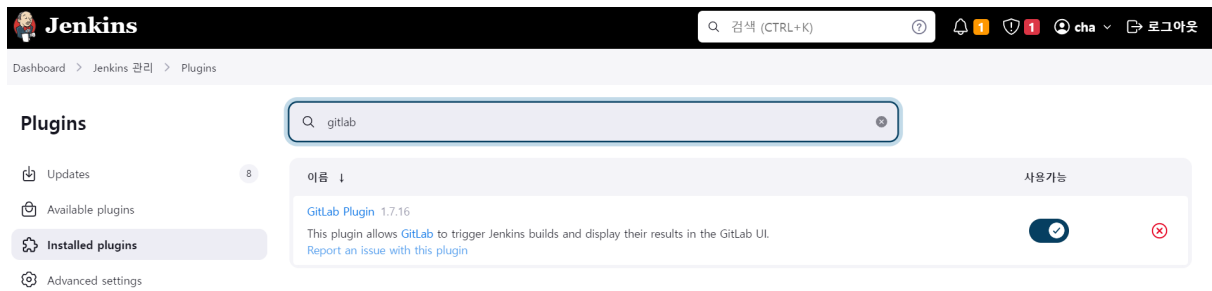
차민준

이메일 주소

whwhdnfl2@naver.com

- 위 빈칸을 채워서 계정을 만든다.

gitlab 플러그인 설치



- jenkins 관리 → plugins로 이동 후 gitlab plugin 설치

credential 설정(인증 설정)

1. 좌측 메뉴에서 jenkins 관리 메뉴 클릭

Security

Security
 Secure Jenkins; define who is allowed to access/use the system.

Credentials
 Configure credentials

Credential Providers
 Configure the credential providers and types

Users
 Create/delete/modify users that can log in to this Jenkins.

2. Credentials 클릭

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

3. global 클릭
4. add credential 클릭

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

whwhdnfl2

☐ Treat username as secret ?

Password ?

.....

ID ?

test


Description ?

5. 다 채운다.

아이템 생성(프로젝트 생성이라고 생각하면 될듯)

Enter an item name

» This field cannot be empty, please enter a valid name

 **Freestyle project**

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

1. new item클릭 후 freestyle project 선택

Configure

General

소스 코드 관리

빌드 유발

빌드 환경

Build Steps

빌드 후 조치

소스 코드 관리

None

Git ?

Repositories ?

Repository URL ?

Please enter Git repository.

Credentials ?

- none -

Add

고급

Add Repository

2. git으로 설정하고 밑에 채워넣는다

Configure

- General
- 소스 코드 관리**
- 빌드 유발
- 빌드 환경
- Build Steps
- 빌드 후 조치

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL:
http://54.180.104.15:8080/project/%E3%85%81%E3%84%B4%E3%85%87%E3%85%81%E3%84%B4%E3%85%87

Enabled GitLab triggers

- ☒ Push Events ?
- ☐ Push Events in case of branch delete ?
- ☒ Opened Merge Request Events ?
- ☐ Build only if new commits were pushed to Merge Request ?
- ☐ Accepted Merge Request Events ?
- ☐ Closed Merge Request Events ?

3. 빌드 유발에서 다음과 같이 한다. 첫번째 체크 박스의 url을 기억하자. 근데 저 주소 그대로 복사하면 안될 수도 있고 jenkins의 주소와 같아야함.

Allowed branches

- ☒ Allow all branches to trigger this job ?
- ☐ Filter branches by name ?
- ☐ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token ?

Generate
Clear

4. 밑에 고급 누르고 secret token 생성 그리고 기억해두기

S S09P12E208

- Project Information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security and Compliance
- Deployments
- Packages and registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings**
 - General
 - Integrations
 - Webhooks**

s09-webmobile2-sub2 > S09P12E208 > Webhook Settings

Webhook was deleted

Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

- ☒ Show full URL
- ☐ Mask portions of URL
Do not show sensitive data such as tokens in the UI.

Secret token

Used to validate received payloads. Sent with the request in the `X-GitLab-Token` HTTP header.

Trigger

- ☐ Push events
- ☐ Tag push events
A new tag is pushed to the repository.
- ☐ Comments
A comment is added to an issue or merge request.

5. 연결하고자 하는 repository에서 webhook을 누르고 아까 기억해둔 url과 secret token을 입력한다.

Hook executed successfully: HTTP 200

A feature flag is turned on or off.

☐ Releases events
A release is created or updated.

SSL verification

☒ Enable SSL verification

[Add webhook](#)

Project Hooks (3)

http://13.125.224.169:8080/project/test	Test	Edit	Delete
SSL Verification: enabled			

6. 그리고 밑에 테스트에서 push event를 하고 위에 hook이 성공했다고 뜨면 연결 성공

7. jenkins 관리 > tools에 들어가서 gradle을 다음과 같이 설정해준다.

Dashboard > Jenkins 관리 > Tools

Gradle installations

Gradle installations ^ Edited

[Add Gradle](#)

Gradle name ?

spring

☒ Install automatically ?

Install from Gradle.org

Version

Gradle 8.2.1

[Add Installer](#)

[Add Gradle](#)

8. 다음과 같이 build steps 설정. wrapper location은 gradlew가 위치한 곳으로 해줘야 한다. workspace 환경변수는 default로 /var/jenkins_home/workspace/backend다. (backend는 프리스타일 이름 설정한 것이다)

Build Steps

Invoke Gradle script ?

☐ Invoke Gradle ?

☒ Use Gradle Wrapper ?

☒ Make gradlew executable

Wrapper location ?

`\${workspace}/back/dream

Tasks ?

clean build

[고급](#) Edited

Execute shell ?

Command

See [the list of available environment variables](#)

```

GRADLE_BUILD_DIR=/var/jenkins_home/workspace/backend/back/dream
cd $GRADLE_BUILD_DIR
sudo docker-compose -f ./docker-compose-prod.yml build --no-cache spring
sudo docker-compose -f ./docker-compose-prod.yml up -d
sudo echo "y" | docker container prune
sudo echo "y" | docker image prune

```

고급 ▾

```

GRADLE_BUILD_DIR=/var/jenkins_home/workspace/backend/back/dream
cd $GRADLE_BUILD_DIR
sudo docker-compose -f ./docker-compose-prod.yml build --no-cache spring
sudo docker-compose -f ./docker-compose-prod.yml up -d
sudo echo "y" | docker container prune
sudo echo "y" | docker image prune

```

9. frontend도 backend와 똑같은 방식으로 설정하고 아래 같이 build steps를 설정

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```

BUILD_PATH=/var/jenkins_home/workspace/frontend/front
cd $BUILD_PATH
sudo docker build -f Dockerfile -t react .
sudo docker rm -f react
docker run -d --name react -p 80:80 react
sudo echo "y" | docker container prune
sudo echo "y" | docker image prune

```

고급 ▾

Add build step ▾

```

BUILD_PATH=/var/jenkins_home/workspace/frontend/front
cd $BUILD_PATH
sudo docker build -f Dockerfile -t react .
sudo docker rm -f react
docker run -d --name react -p 80:80 react
sudo echo "y" | docker container prune
sudo echo "y" | docker image prune

```

프로젝트 local 실행 가이드


1. Backend 실행 가이드

(1) IntelliJ **2023.1.3** 설치하기

- 이전 버전 설치


IntelliJ IDEA

IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains

 <https://www.jetbrains.com/idea/download/other.html>

IntelliJ IDEA

The Leading Java and Kotlin IDE



- 인코딩 설정 UTF-8로 통일시키기

(2) Java 17 설치


- jdk 버전17


 https://www.java.com/download/ie_manual.jsp

- java 환경변수 설정하기

JDK 8 다운로드 및 설치하기, 환경변수 설정 [Java개발환경 구축하기 1]

JDK8 설치하기 (윈도우10) jdk는 지금 14버전까지 나와있다. 근데 왜 우리는 왜 8버전을 쓸까?? 그러게여.. 알려주세요.. 아마 9버전 이상부터는 상업적 이용을 위해선 돈을 지불하고 사용해야 하기 때문일 것이다. 그래도 양심은 있는지 무료로 사용 가능한 8버전은 현재 Java SE 8u241 버전인데 업데이트를 241번이나 해 주었다. // 이미

 <https://the-duchi.tistory.com/4>




(3) MySQL 8.0.33 설치


- 설치 가이드

MySQL 다운로드 및 설치하기(MySQL Community 8.0)

SQL을 본격적으로 사용하려면 DBMS를 설치해야 합니다. 여러 가지 DBMS 중에서 MySQL 설치 하는 방법을 알아보고, 정상적으로 설치가 되었는지 확인하는 방법을 알아보겠습니다. 2021년 10월 기준 MySQL Community 8.0.21...

 <https://hongong.hanbit.co.kr/mysql-다운로드-및-설치하기mysql-community-8-0/>

MySQL 설치 방법과 정상 설치 확인하기

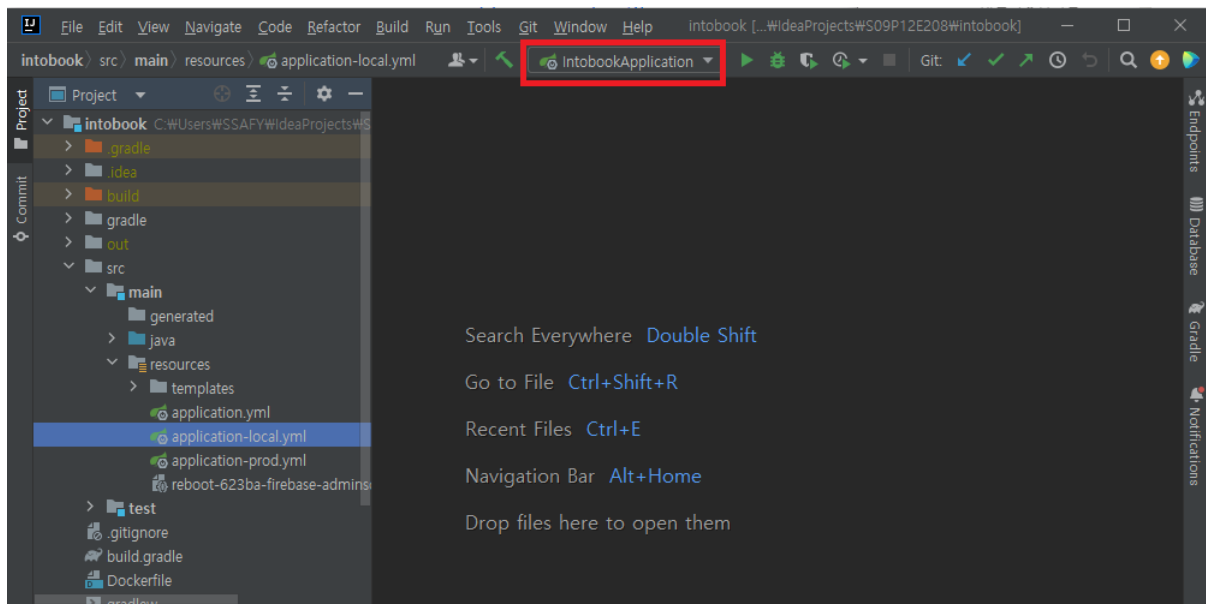


- 로컬 사용자 계정 (ID: root, PW: 1234) 로 설정하기 → `application-local.yml` 에 명시

(4) Project Git Clone

```
> git clone https://lab.ssfy.com/s09-bigdata-dist-sub2/S09P22E206.git # git repo 클론하기
> cd S09P22E206# 프로젝트가 클론된 폴더로 이동
```

(5) Springboot Application 실행시키기



2. Frontend 실행 가이드

(1) Visual Studio Code(VSC) 설치

<https://code.visualstudio.com/download>

(2) Project Git Clone

```
> git clone https://lab.ssafty.com/s09-webmobile2-sub2/S09P12E208.git # git repo 클론하기
> cd S09P12E208 # 프로젝트가 클론된 폴더로 이동
```

(3) Node.js 18.16.1 설치

Node.js
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

<https://nodejs.org/ko>



(4) 패키지 설치

```
> npm i # package.json에 연결된 라이브러리를 모두 한 번에 다운로드하기
```

(5) 프로젝트 실행

```
> npm start # 프로젝트 시작
```

빅데이터

Hadoop 실행 가이드

0. 기본 환경 설정

- spark 계정 생성

```
sudo adduser spark
```

- ssh key 생성(master, worker1 ...)

```
cd ~/
ssh-keygen -t rsa
```

입력하라고 나오면 공백으로 두고 전부 enter를 쳐서 넘어간다.

```
# id_rsa.pub의 key 내용을 현재 컴퓨터와 다른 컴퓨터의
# authorized_keys에 복사한다.
vi authorized_keys
```

- ip 이름 설정

```
sudo vi /etc/hosts

{master로 사용할 컴퓨터 ip} master
{worker로 사용할 컴퓨터 ip} worker
...
```

1. Java 11 설치(master, worker1 ...)

```
apt-get update
apt-get upgrade
apt install openjdk-11-jdk
```

2. hadoop 3.3.6 설치(master)

```
cd /mydir
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6-src.tar.gz
tar -zxvf hadoop-3.3.6.tar.gz
mv hadoop-3.3.6 hadoop
```

3. 설정파일

- hadoop-env.sh 수정

```
cd /mydir/hadoop/etc/hadoop
vi hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

- core-site.xml 수정

```
vi core-site.xml
```

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
  <property>
    <name>hadoop.http.staticuser.user</name>
    <value>hadoop</value>
  </property>
</configuration>
```

- hdfs-site.xml 수정

```
vi hdfs-site.xml
```

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///opt/hadoop/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///opt/hadoop/dfs/data</value>
  </property>
  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>file:///opt/hadoop/dfs/namesecondary</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
</configuration>
```

- yarn-site.xml 수정

```
vi yarn-site.xml
```

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>
  <!-- port 변경이 필요할 경우... (resourcemanager....) --> <!-- Stop attacks: get-shell(YARN) etc. -->
  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <!--
    <value>spark-master-01:8088</value>
    -->
    <value>master:8188</value>
  </property>
</configuration>
```

- workers 설정

```
vi workers
```

```
# 설정할 worker의 수만큼
worker1
worker2
...
```

- 다른 worker로 hadoop 파일 복사

```
scp -r /mydir/hadoop spark@worker1:/mydir/
...
```

4. 실행

```
# NameNode, SecondaryNameNode, DataNode 실행
/mydir/hadoop/sbin/start-dfs.sh
jps
# jps를 수행
# master에 NameNode, SecondaryNameNode 수행 확인
# worker에 DataNode 수행 확인
```

- 안되면 포트열기

Spark

1. spark 3.4.1 설치(master)

```
cd /mydir
wget https://d1cdn.apache.org/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz
tar -zxvf spark-3.4.1-bin-hadoop3.tgz
mv spark-3.4.1-bin-hadoop3 spark
```

2. 설정 파일

- spark-env.sh 수정

```
cd /mydir/spark/conf
cp spark-env.sh.template spark-env.sh
```

```
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
SPARK_MASTER_PORT=7177
SPARK_MASTER_WEBUI_PORT=8180
SPARK_WORKER_PORT=7178
SPARK_WORKER_WEBUI_PORT=8181
SPARK_WORKER_CORES=8
SPARK_WORKER_MEMORY=8G
SPARK_PUBLIC_DNS=master
SPARK_MASTER_HOST=master
```

- spark-defaults.conf 설정

```
cp spark-defaults.conf.template spark-defaults.conf
vi spark-defaults.conf
```

```
spark.master                spark://master:7177
spark.driver.port           38841
spark.blockManager.port     38842
```

- workers 설정

```
cp workers.template workers
vi workers
```

```
worker1
worker2
...
```

- 다른 worker로 spark 폴더 복사

```
scp -r /mydir/spark spark@worker1:/mydir/
scp -r /mydir/spark spark@worker2:/mydir/
...
```

- worker별로 spark-env.sh 수정

```
# worker에 맞게 설정
SPARK_PUBLIC_DNS=worker1
```

3. 실행


```
/mydir/spark/sbin/start-all.sh
jps
# jps로 Master와 Worker가 켜졌는지 확인
```

- 안될 경우 포트 열기

4. Spark 프로그램 실행

- spark structured streaming 실행

```
nohup /mydir/spark/bin/spark-submit --total-executor-cores 4 --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.1 kafka_spark_hc
# 이후 ctrl+c로 빠져나오면 백그라운드에서 실행
```

- spark 분석 실행

```
crontab -e
*/10 * * * * /mydir/spark/bin/spark-submit batch_analysis.py
```

Elasticsearch

- 키바나 GUI 접속 { Management → Dev Tools }

http://43.201.35.196:5601/app/dev_tools#/console

(43.201.35.196 → ip)

- mysql에서 es 로 데이터 모두 가져와 저장하기

```
# 로그스테시의 sql_last_value 값 확인
docker exec -u=0 logstash cat /usr/share/logstash/.logstash_jdbc_last_run_1
```

- 로그 스테시 재시작

```
docker-compose -f docker-compose-elastic.yml restart logstash
```

mysql에서 es 로 데이터를 복사 저장을 하기 위해서는 명령어 부터 빠르게 mysql_diary 세팅 및 매핑을 등록해야 합니다.

- mysql_diary 인덱스 세팅 및 매핑

```
# 인덱스 생성 및 설정
PUT /mysql_diary
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    },
    "analysis": {
      "tokenizer": {
        "nori_tokenizer": {
          "type": "nori_tokenizer",
          "decompound_mode": "mixed",
          "user_dictionary": "dict.txt"
        }
      },
      "analyzer": {
        "nori_analyzer": {
          "type": "custom",
          "tokenizer": "nori_tokenizer",
          "filter": "nori_filter"
        }
      }
    }
  }
}
```

```

        "nfd_analyzer": {
            "filter": [
                "lowercase"
            ],
            "char_filter": [
                "nfd_normalizer"
            ],
            "tokenizer": "standard"
        }
    },
    "char_filter": {
        "nfd_normalizer": {
            "mode": "decompose",
            "name": "nfc",
            "type": "icu_normalizer"
        }
    },
    "filter": {
        "nori_filter": {
            "type": "nori_part_of_speech",
            "stoptags": [
                "E", "IC", "J", "MAG", "MAJ", "MM", "SP", "SSC", "SSO", "SC", "SE", "XPN", "XSA", "XSN", "XSV", "UNA", "NA", "VSV"
            ]
        },
        "nfd_filter": {
            "mode": "decompose",
            "name": "nfc",
            "type": "icu_normalizer"
        }
    }
}
}
}

# 인덱스 매핑
PUT /mysql_diary/_mappings
{
  "properties": {
    "id": {
      "type": "long"
    },
    "title": {
      "type": "keyword",
      "copy_to": ["title_nori", "combined_title_content"]
    },
    "title_nori": {
      "type": "text",
      "analyzer": "nori_analyzer",
      "fielddata": true
    },
    "content": {
      "type": "keyword",
      "copy_to": ["content_nori", "combined_title_content"]
    },
    "content_nori": {
      "type": "text",
      "analyzer": "nori_analyzer",
      "fielddata": true
    },
    "emotion": {
      "type": "keyword",
      "copy_to": ["emotion_nori"]
    },
    "emotion_nori": {
      "type": "text",
      "analyzer": "nori_analyzer"
    },
    "member_id": {
      "type": "long"
    },
    "like_count": {
      "type": "integer"
    },
    "open": {
      "type": "boolean"
    },
    "combined_title_content": {
      "type": "text",
      "analyzer": "nori_analyzer",
      "fielddata": true,
      "fields": {
        "row": {
          "type": "keyword"
        }
      }
    },
  },

```

```

        "spell": {
          "type": "text",
          "analyzer": "nfd_analyzer"
        }
      }
    }
  }
}

```

- log_recommend 인덱스 세팅 및 매핑

```

# 사용자 추천 인덱스 생성 및 설정
PUT /log_recommend
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    },
    "analysis": {
      "tokenizer": {
        "nori_tokenizer": {
          "type": "nori_tokenizer",
          "decompound_mode": "mixed",
          "user_dictionary": "dict.txt"
        }
      },
      "analyzer": {
        "nori_analyzer": {
          "type": "custom",
          "tokenizer": "nori_tokenizer",
          "filter": "nori_filter"
        }
      },
      "filter": {
        "nori_filter": {
          "type": "nori_part_of_speech",
          "stoptags": [
            "E", "IC", "J", "MAG", "MM", "NA", "NR", "SC",
            "SE", "SF", "SH", "SL", "SN", "SP", "SSC", "SSO",
            "SY", "UNA", "UNKNOWN", "VA", "VCN", "VCP", "VSV",
            "VV", "VX", "XPN", "XR", "XSA", "XSN", "XSV"
          ]
        }
      }
    }
  }
}

# 사용자 추천
PUT /log_recommend/_mappings
{
  "properties": {
    "diaryId" : {
      "type": "long"
    },
    "memberId": {
      "type": "long"
    },
    "title": {
      "type": "keyword",
      "copy_to": ["title_nori"]
    },
    "title_nori": {
      "type": "text",
      "analyzer": "nori_analyzer",
      "fielddata": true
    },
    "content": {
      "type": "keyword",
      "copy_to": ["content_nori"]
    },
    "content_nori": {
      "type": "text",
      "analyzer": "nori_analyzer",
      "fielddata": true
    },
    "emotion": {
      "type": "keyword"
    }
  }
}

```

```

    }
  }
}

```

- log_transaction 인덱스 세팅 및 매핑

```

PUT /log_transaction
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    }
  }
}

PUT /log_transaction/_mappings
{
  "properties": {
    "transaction_id": {
      "type": "long"
    },
    "diary_id": {
      "type": "long"
    },
    "member_id": {
      "type": "long"
    },
    "point": {
      "type": "integer"
    }
  }
}

```

- mysql_daily_statistic 인덱스 세팅 및 매핑

```

PUT /mysql_daily_statistic
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    }
  }
}

PUT /mysql_daily_statistic/_mappings
{
  "properties": {
    "keyword": {
      "type": "keyword"
    },
    "memberId": {
      "type": "long"
    }
  }
}

```

- mysql_month_statistic 인덱스 세팅 및 매핑

```

PUT /mysql_month_statistic
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    }
  }
}

```

```
PUT /mysql_month_statistic/_mappings
{
  "properties": {
    "emotion": {
      "type": "keyword"
    }
  }
}
```

- log_strict 인덱스 세팅 및 매핑

```
PUT /log_strict
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    }
  }
}

PUT /log_strict/_mappings
{
  "properties": {
    "memberId": {
      "type": "long"
    },
    "registDate": {
      "type": "keyword"
    }
  }
}
```

외부 서비스

Kakao karlo(그림 ai)

(1) <https://developers.kakao.com/> 접속

(2) 로그인 후, 내 애플리케이션 > 애플리케이션 추가하기



애플리케이션 추가하기

앱 아이콘

이미지 업로드

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

파일 선택

앱 이름

내 애플리케이션 이름

사업자명

사업자 정보와 동일한 이름

• 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
 • 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영정책을 위반하지 않는 앱입니다.

취소

저장

- 앱 이름 : 임의 설정
- 사업자명 : 임의 설정 (팀명 입력하였음)
- ☒ 서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영 정책을 위반하지않는 앱입니다. 체크

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

앱 키

네이티브 앱 키	8708dd1c6
REST API 키	4868cf449
JavaScript 키	061d10011
Admin 키	db0ec7ffd

- REST API 키를 사용한다.

<input checked="" type="checkbox"/> Authorization	KakaoAK
<input checked="" type="checkbox"/> Content-Type	application/json
Key	Value

- POST 요청의 body에 Authorization에 key를 넣는다.
- 요청주소: <https://api.kakaobrain.com/v2/inference/karlo/t2i>

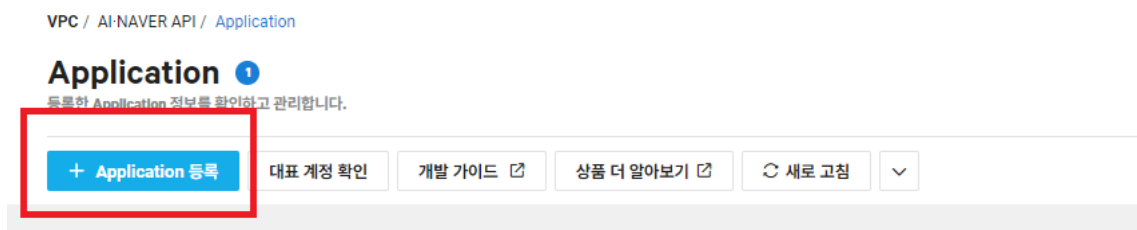
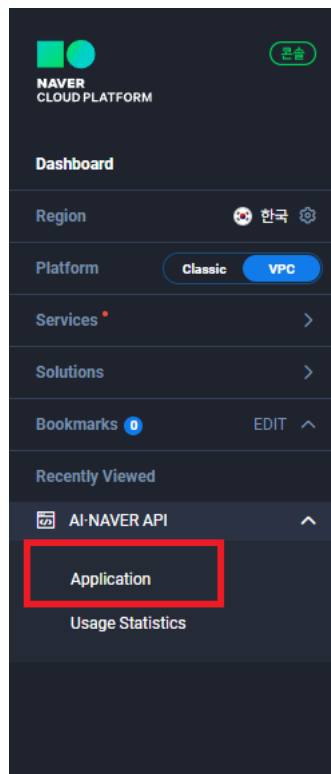
naver clova(감정 분석)

(1) <https://www.ncloud.com/> 접속

(2) 회원가입 후 콘솔 눌러 접속하기



(3) 어플리케이션 추가하기




- 어플리케이션 이름 등록, CLOVA Sentiment 서비스 선택

Application 이름 설정
AI - Naver Service 를 이용할 Application 을 등록하세요

Application 이름

Service 선택
Application에서 이용할 Service 를 선택하세요



☐ CLOVA Speech Recognition (CSR)
☐ CLOVA Face Recognition (CFR)
☒ CLOVA Voice - Premium
☐ CLOVA Summary
☐ Papago Image Translation(Text)

서비스 설명/요금안내

개발 가이드

서비스 설명/요금안내

개발 가이드

서비스 설명/요금안내

개발 가이드

서비스 설명/요금안내

개발 가이드

서비스 설명/요금안내

개발 가이드

- 서비스 환경 등록 Web 서비스 URL에 서비스를 사용하는 서버의 URL을 작성하고 추가 버튼을 누른다.

서비스 환경 등록
Application에서 이용할 서비스 환경을 등록하세요

Web 서비스 URL(최대 10개)

Android 앱 패키지 이름(최대 10개)

iOS Bundle ID(최대 10개)

(4) 사용하기

- Client ID와 Client Secret가 필요하다.
- 요청

VPC / AI-NAVER API / Application

Application 대표 계정 확인 개발 가이드

등록한 Application 정보를 확인하고 관리합니다.

App 이름	서비스구분	담당 사용량	담당 사용량
dream <input type="button" value="인증 정보"/> <input type="button" value="수정"/>	CLOVA Sentiment	0%	0/1,000 회 2%

인증 정보

Application key

Application 이름

Client ID (X-NCP-APIGW-API-KEY-ID)

Client Secret (X-NCP-APIGW-API-KEY)

서비스환경

Web 서비스 URL

Android 앱 패키지 이름

iOS Bundle ID

정보 변경을 원하시면, Application 선택 후 변경 버튼을 통해 진행해주세요.

POST <https://naveropenapi.apigw.ntruss.com/sentiment-analysis/v1/analyze>

- 요청 헤더

X-NCP-APIGW-API-KEY-ID	앱 등록 시 발급받은 Client ID X-NCP-APIGW-API-KEY-ID:{Client ID}
X-NCP-APIGW-API-KEY	앱 등록 시 발급 받은 Client Secret X-NCP-APIGW-API-KEY:{Client Secret}
Content-Type	바이너리 전송 형식 Content-Type: application/json

- 요청 Body 예시

```
{
  "content": "싸늘하다. 가슴에 비수가 날아와 꽃인다."
}
```

시연 시나리오

1. 페이지 최초 접속 (인트로)
 - 화면 스크롤 후 아래 버튼으로 로그인 페이지 이동
2. 로그인 페이지
 - 로그인 실시
3. 메인페이지
 - 메인페이지 인트로 감상
 - 스크롤 내리며 페이지 위치별 요소들 설명
4. 좌측 네브바 클릭 => 다이어리 클릭 => 꿈 일기 작성페이지 이동
 - 내용 작성 후 그림생성 버튼 클릭
 - 다시생성 / 저장하기 설명
5. 디테일 페이지
 - 대략적인 설명
6. 네브바 => 서치 => 검색결과 페이지
 - 검색할 내용(추석 -> cntjr) 으로 검색
7. 네브바 => shop 페이지
 - 스크롤 내리며 페이지 요소 설명 (무한스크롤, 인터랙티브 요소)
 - 구매완료 일기 중 1개 클릭
 - 디테일 페이지로 이동하여 구매 과정 설명
8. 네브바 => 코인교환
 - 코인교환 모달창에서 코인교환 실시
 - 아래쪽 추천일기 설명
9. 네브바 => 메인 => 우상단 구름버튼 클릭 => 클라우드 페이지
 - 클라우드 페이지 설명(인터랙티브 요소)
10. 네브바 => 메인 => 갤러리 페이지
 - 입장효과 및 구현효과 설명
 - 대략적인 설명
11. 네브바 => 마이페이지
 - 마이페이지 설명
12. 네브바 => 통계페이지
 - 통계에서 보여주는 것들 설명

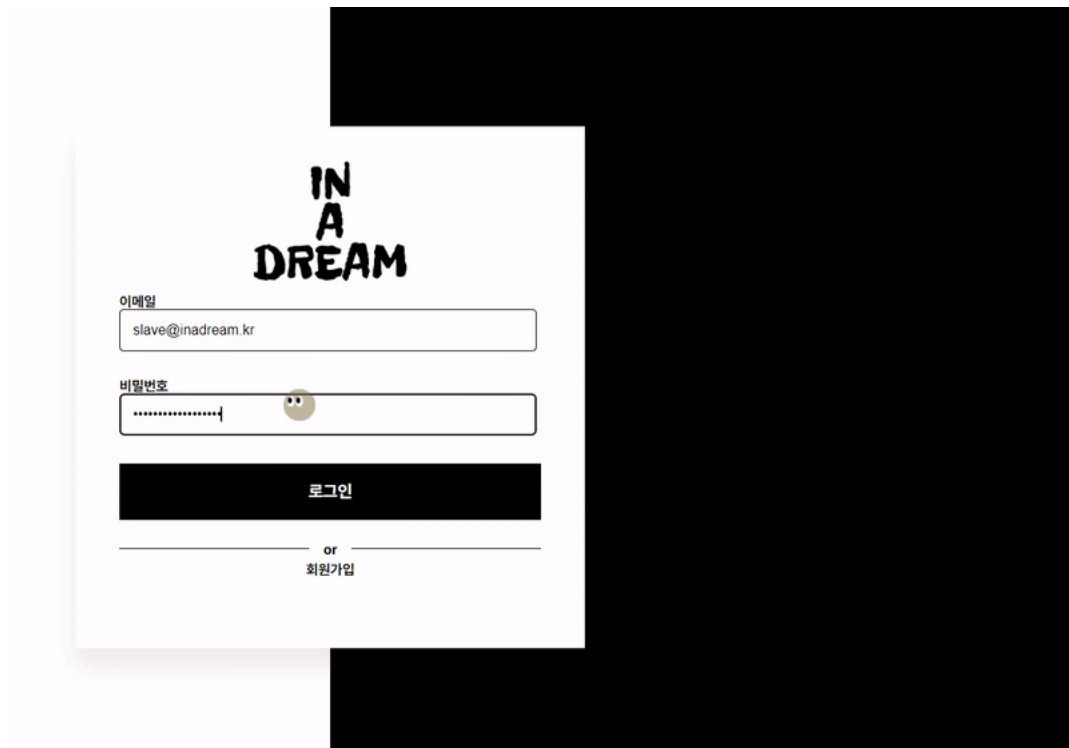
- 날짜 변경하여 변경된 데이터 설명
 - 스크롤 내려서 아래쪽 막대그래프 설명
- 끝

시연 화면

1. Intro 페이지



2. LoginPage



3. MainPage



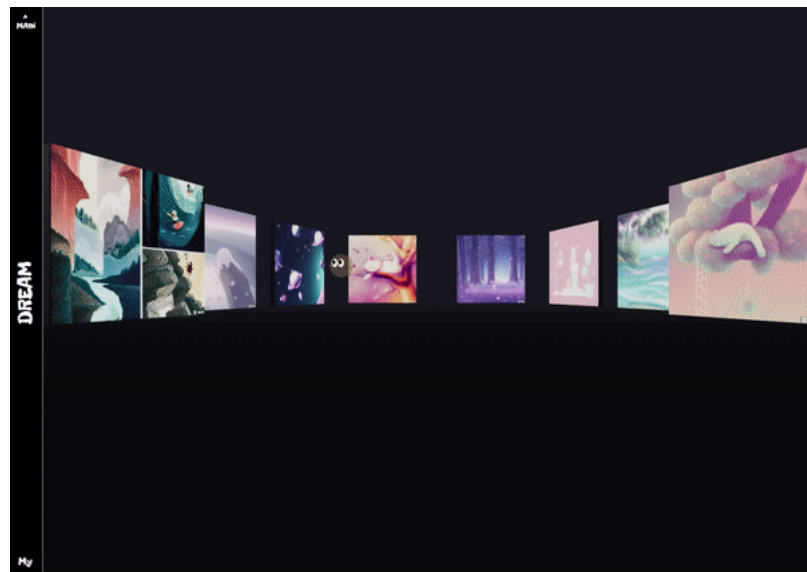
4. CreateDiaryPage



5. DreamShopPage



6. GalleryPage



7. CloudPage



8. StatisticsPage



9. MyPage

