📜 포팅메뉴얼

📌 개발 환경

FrontEnd

-

BackEnd

- apigateway-service
  - Java OpenJDK `17.0.12`
  - Spring Boot `3.3.5`
    - Spring Cloud gateway `4.1.5`
    - Spring Cloud netflix eureka client `4.1.3`
    - Spring Cloud bootstrap `4.1.4`
    - Spring Cloud config `4.1.3`
    - Spring Cloud actuator `3.3.5`
    - Spring openapi webflux `2.6.0`
    - Spring Data redis `3.3.5`
  - JWT `0.12.3`
- user-service
  - Java OpenJDK `17.0.12`
  - Spring Boot `3.3.5`
    - Spring Cloud `4.1.4`
    - Spring Cloud netflix eureka client `4.1.3`
    - Spring Cloud bootstrap `4.1.4`
    - Spring Cloud config `4.1.3`
    - Spring Cloud actuator `3.3.5`
    - Spring Cloud aws `2.2.6`
    - Spring openapi webmvc `2.2.0`
    - Spring Data redis `3.3.5`
    - Spring Data jpa `3.3.5`

    - Gradle `8.10`
    - AWS S3 Bucket Cloud `2.2.6`
    - Lombok `1.18.34`

- config-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud config server `4.1.3`
        - Spring Cloud actuator `3.3.5`
    - Gradle `8.10`

- discovery-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud netflix eureka client server `4.1.3`
        - Spring Cloud actuator `3.3.5`
    - Gradle `8.10`

- drawing-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud `4.1.4`
        - Spring Cloud netflix eureka client `4.1.3`
        - Spring Data mail `3.3.5`
        - Spring web `3.3.5`
        - Spring kafka `3.2.4`
        - Spring security crypto `3.2.4`
    - JWT `0.12.3`
    - Gradle `8.10`
    - AWS S3 Bucket Cloud `2.2.6`
    - Lombok `1.18.20`
    - Swagger `2.2.0`

- lecture-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud `4.1.4`
        - Spring Cloud netflix eureka client `4.1.3`
        - Spring Cloud bootstrap `4.1.4`
        - Spring Cloud config `4.1.3`
        - Spring Cloud actuator `3.3.5`
        - Spring Data redis `3.3.5`
        - Spring Data jpa `3.3.5`
        - Spring Data jdbc `3.3.5`
        - Spring Data mongodb `3.3.5`
        - Spring web `3.3.5`
        - Spring kafka `3.2.4`

        - Spring Cloud bootstrap `4.1.4`
        - Spring Cloud config `4.1.3`
        - Spring Cloud actuator `3.3.5`
        - Spring Data mongodb `3.3.5`
        - Spring web `3.3.5`
        - Spring websocket `3.3.5`
    - Gradle `8.10`
    - Lombok `1.18.20`

- notification-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud `4.1.4`
        - Spring Cloud netflix eureka client `4.1.3`
        - Spring Cloud bootstrap `4.1.4`
        - Spring Cloud config `4.1.3`
        - Spring web `3.3.5`
        - Spring Data jpa `3.3.5`
        - Spring Data redis `3.3.5`
        - Spring kafka `3.2.4`
    - Gradle `8.10`
    - Lombok `1.18.20`
    - Firebase admin `9.4.1`
    - Gradle `8.10`
    - Lombok `1.18.20`

- questionbox-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud netflix eureka client `4.1.3`
        - Spring Cloud bootstrap `4.1.4`
        - Spring Cloud config `4.1.3`
        - Spring Cloud actuator `3.3.5`
        - Spring Data jpa `3.3.5`
        - Spring web `3.3.5`
        - Spring kafka `3.2.4`
    - Gradle `8.10`
    - Lombok `1.18.20`

- user-service
    - Java OpenJDK `17.0.12`
    - Spring Boot `3.3.5`
        - Spring Cloud `4.1.4`
        - Spring Cloud netflix eureka client `4.1.3`
        - Spring Cloud bootstrap `4.1.4`
        - Spring Cloud config `4.1.3`

        - Spring Cloud actuator `3.3.5`
        - Spring openapi webmvc `2.2.0`
        - Spring Data jpa `3.3.5`
        - Spring web `3.3.5`
    - Gradle `8.10`
    - Lombok `1.18.20`

UI/UX

- Figma

IDE

- IntelliJ `2024-01`
- Visual Studuio Code `1.94.1`

Server 배포 환경

- AWS EC2 `ubuntu 20.04.6 LTS`
- Docker `27.2.0`
- Docker Compose `2.29.2`
- Nginx `1.18.0`
- SSL
- Docker Hub

CI/CD

- jenkins `2.475`

DB

- MySQL `8.0.38`
- redis `7.4.0`
- mongoDB `7.0.14`
- AWS S3

Collaboration

형상관리

- GitLab

커뮤니케이션

- Mattermost
- Notion

이슈관리

- Jira

📌 환경 변수 설정

[FrontEnd]

📃 .env

```
```

[BackEnd]

📃 apigateway-service.yml

```
server:
  port: 8000

springdoc:
  swagger-ui:
    use-root-path: true
    urls[0]:
      name: user 서비스
      url: /api/user/v3/api-docs

jwt:
  secret: {{ JWT SecretKey }}
spring :
  data:
    redis:
      host: k11d101.p.ssafy.io
      password: {{ Password }}
      port: 6379
      repository: false

  application:
    name: apigateway-service

  cloud:
    gateway:
      default-filters:
        - DedupeResponseHeader=Access-Control-Allow-Origin Ac
      globalcors:
        cors-configurations:
          '[/**]':
            allowedOrigins:
              - 'http://k11d101.p.ssafy.io:8082'
              - 'http://k11d101.p.ssafy.io:8087'
              - 'http://k11d101.p.ssafy.io'
              - 'https://k11d101.p.ssafy.io'
              - 'http://localhost:5554'
              - 'http://10.0.2.2:5554'
            allow-credentials: true # JWT 나 쿠키를 사용해 메시지
            allowedHeaders: '*'
            allowedMethods: # 메서드를 명시하지 않으면 안되는 경우도 
              - PUT
              - GET
              - POST
              - PATCH
              - DELETE
              - OPTIONS
      routes:
        - id: logout
          uri: lb://USER-SERVICE
          predicates:
            - Path=/api/user/logout
          filters:
            - StripPrefix=1
            - name: AuthorizationHeaderFilter # 특정 라우트에만 
            - name: LogoutAuthorizationHeaderFilter

        - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/api/user/info/**
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

        - id: user-service-actuator
          uri: lb://USER-SERVICE
          predicates:
            - Path=/api/user/actuator/**
            - Method=GET,POST
          filters:
            - RewritePath=/api/user/(?<segment>.*), /$\{segme
            - RemoveRequestHeader=Cookie,Set-Cookie

        - id: auth-user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/api/user/**, /api/mail/**
          filters:
            - StripPrefix=1 # "/api/user-service" 부분을 제거하여

        - id: folder-service-actuator
          uri: lb://FOLDER-SERVICE
          predicates:
            - Path=/api/folder/actuator/**
            - Method=GET,POST
          filters:
            - RewritePath=/api/folder/(?<segment>.*), /$\{seg
            - RemoveRequestHeader=Cookie,Set-Cookie

        - id: folder-service
          uri: lb://FOLDER-SERVICE
          predicates:
            - Path=/api/folder/**, /api/file/**
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

        - id: lecture-service-actuator
          uri: lb://LECTURE-SERVICE
          predicates:
            - Path=/api/lecture/actuator/**
            - Method=GET,POST
          filters:
            - RewritePath=/api/lecture/(?<segment>.*), /$\{se
            - RemoveRequestHeader=Cookie,Set-Cookie

        - id: lecture-service
          uri: lb://LECTURE-SERVICE
          predicates:
            - Path=/api/lecture/** , /api/exam/** , /api/homew
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

        - id: ocr-service
          uri: lb://OCR-SERVICE
          predicates:
            - Path=/api/ocr/**
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

        - id: user-service-docs
          uri: lb://USER-SERVICE  # 'user-service'의 Load Bala
          predicates:
            - Path=/api/user/v3/api-docs  # Gateway에서 접근할 
          filters:
            - StripPrefix=2  # '/api/user'를 제거하고 'v3/api-d

        - id: websocket_route
          uri: lb:ws://drawing-service
          predicates:
            - Path=/ws-gateway/**
          filters:
            - StripPrefix=1

        - id: todo-service
          uri: lb://TODO-SERVICE
          predicates:
            - Path=/api/todo/**
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

        - id: notification-service
          uri: lb://NOTIFICATION-SERVICE
          predicates:
            - Path=/api/notification/**
          filters:
            - StripPrefix=1 # "/api/u
            - name: AuthorizationHeaderFilter

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://discovery-service:8761/eureka

management:
  endpoints:
    web:
      exposure:
        include: "*"
```

📃 config-service.yml

```
server :
  port : 8888

management:
  endpoints:
    web:
      exposure:
        include: "*"

spring:
  application:
    name: config-service
  cloud:
    config:
      server:
        git:
          uri: https://lab.ssafy.com/final-module/submodule.g
          username: {{ Gitlab ID }}
          password: {{ Access Token }}
          default-label: master
```

📃 discovery-service.yml

```
server:
  port: 8761

spring:
  application:
    name: discovery-service

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false

management:
  endpoints:
    web:
      exposure:
        include: "*"
```

📃 drawing-service.yml

```
server:
  port: 0

spring:
  application:
    name: drawing-service

  data:
    mongodb:
      host: 54.180.158.95
      port: 27017
      database: drawing
      username: {{ Username }}
      password: {{ Password }}
      authentication-database: admin

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    metrics:
      enabled: true
    prometheus:
      enabled: true
```

📃 folder-service.yml

```
server:
  port: 0

spring:
  application:
    name: folder-service

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{{ RDS Host }}:3306/folder-service
    username: {{ Username }}
    password: {{ Password }}
  jpa:
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect

  kafka:
    bootstrap-servers: 54.180.158.95:9092
    consumer:
      group-id: folder-group

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    metrics:
      enabled: true
    prometheus:
      enabled: true
  metrics:
    enable:
      all: true
```

> 📃 **lecture-service.yml**

```
server :
  port : 8083

# 스프링 설정
spring:
  application:
    name: lecture-service

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{{ RDS Host }}:3306/lecture-service?ser
    username: {{ Username }}
    password: {{ Password }}

  # JPA 설정
  jpa:
    defer-datasource-initialization: false
    generate-ddl: true

    hibernate:
      ddl-auto: update
    open-in-view: false
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        show_sql: true
  data:
    mongodb:
      host: 54.180.158.95
      port: 27017
      database: lecture
      username: {{ Username}}
      password: {{ Password }}
      authentication-database: admin

  kafka:
    bootstrap-servers: 54.180.158.95:9092
    consumer:
      key-deserializer: org.apache.kafka.common.serialization
      value-deserializer: org.springframework.kafka.support.s
      properties:
        spring.deserializer.value.delegate.class: org.springf
        spring.json.trusted.packages: "com.eum.lecture_servic
        spring.json.type.mapping: >
          com.eum.user_service.domain.event.dto.ClassEvent:co
          com.eum.user_service.domain.event.dto.StudentInfoEv
          com.eum.user_service.domain.event.dto.TeacherInfoEv
```

```
management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    metrics:
      enabled: true
    prometheus:
      enabled: true
```

📃 notification-service.yml

```
server :
  port : 8089

# 스프링 설정
spring:
  application:
    name: notification-service

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{{ RDS Host }}:3306/notification-servic
    username: {{ Username }}
    password: {{ Password }}

  # JPA 설정
  jpa:
    defer-datasource-initialization: false
    generate-ddl: true

    hibernate:
      ddl-auto: update
    open-in-view: false
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        show_sql: true

  kafka:
    bootstrap-servers: 54.180.158.95:9092
    consumer:
      key-deserializer: org.apache.kafka.common.serialization
      value-deserializer: org.springframework.kafka.support.s
      properties:
        spring.deserializer.value.delegate.class: org.springf
        spring.json.trusted.packages: "com.eum.notification_s
        spring.json.type.mapping: >
          com.eum.lecture_service.event.event.notification.Le
          com.eum.lecture_service.event.event.notification.Le
          com.eum.lecture_service.event.event.notification.Ho
          com.eum.lecture_service.event.event.notification.Ex



  data:
    redis:
      host: 54.180.158.95
      port: 6379
      password: {{ Password }}
```

📃 todo-service.yml

```
server :
  port : 8087

# 스프링 설정
spring:
  application:
    name: todo-service

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{{ RDS Host }}:3306/todo-service?useSSL
    username: {{ Username }}
    password: {{ Password }}
      # 유휴 커넥션을 유지할 시간
  data:
    mongodb:
      host: 54.180.158.95
      port: 27017
      database: homeworks
      username: {{ Username }}
      password: {{ Password }}
      authentication-database: admin

  # JPA 설정
  jpa:
    defer-datasource-initialization: false
    generate-ddl: true

    hibernate:
      ddl-auto: update
      dialect: org.hibernate.dialect.MySQLDialect
    open-in-view: false
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        show_sql: true

  swagger:
    server:
      url: https://k11d101.p.ssafy.io/api

  kafka:
    bootstrap-servers: 54.180.158.95:9092
    consumer:
      key-deserializer: org.apache.kafka.common.serialization
      value-deserializer: org.springframework.kafka.support.s
      properties:
        spring.deserializer.value.delegate.class: org.springf
        spring.json.trusted.packages: "com.eum.lecture_servic
        spring.json.type.mapping: >
          com.eum.lecture_service.event.event.homework.Homewo
          com.eum.lecture_service.event.event.homework.Homewo

springdoc:
  swagger-ui:
    path: /swagger-ui
  api-docs:
    version: openapi_3_1
    path: /v3/api-docs
```

📃 user-service.yml

```
server :
  port : 8082

springdoc:
  swagger-ui:
    path: /swagger-ui
  api-docs:
    version: openapi_3_1
    path: /v3/api-docs

# 스프링 설정
spring:
  application:
    name: user-service

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{{ RDS Host }}:3306/user-service?useSSL
    username: {{ Username }}
    password: {{ Password }}                # 유휴 커넥션을 유지
  data:
    redis:
      host: k11d101.p.ssafy.io
      password: {{ Password }}
      port: 6379

  mail:
    host: smtp.gmail.com
    port: 587
    username: {{ Username }}
    password: {{ Password }}
    properties:
      mail:
        smtp:
          auth: true
          starttls.enable: true
          connectiontimeout: 18000
          timeout: 18000
          writetimeout: 18000

  # JPA 설정
  jpa:
    defer-datasource-initialization: false
    generate-ddl: true

    hibernate:
      ddl-auto: update
      dialect: org.hibernate.dialect.MySQLDialect
    open-in-view: false

  kafka:
    bootstrap-servers:
      - k11d101.p.ssafy.io:9092

  # s3
  cloud:
    aws:
      credentials:
        access-key: {{ Access Key }}
        secret-key: {{ Secret Key }}
      s3:
        bucket: d101-eum-bucket
      region:
        static: ap-northeast-2
      stack:
        auto: false
  swagger:
    server:
      url: https://k11d101.p.ssafy.io/api
jwt:
  secret: {{ JWT SecretKey }}
  access-token:
    expire-time: 86400000  # 60sec * 60min * 24hour
  refresh-token:
    expire-time: 8640000000  # 60sec * 60min * 24hour * 10day

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    metrics:
      enabled: true
    prometheus:
      enabled: true
```

📌 배포 환경 설정

0. 초기 세팅

1. EC2 접속

```
# sudo ssh -i [pem키 위치] [접속 계정]@[접속할 도메인]
$ sudo ssh -i K11D101T.pem ubuntu@k11d101.p.ssafy.io
```

2. Docker & Docker Engine 설치

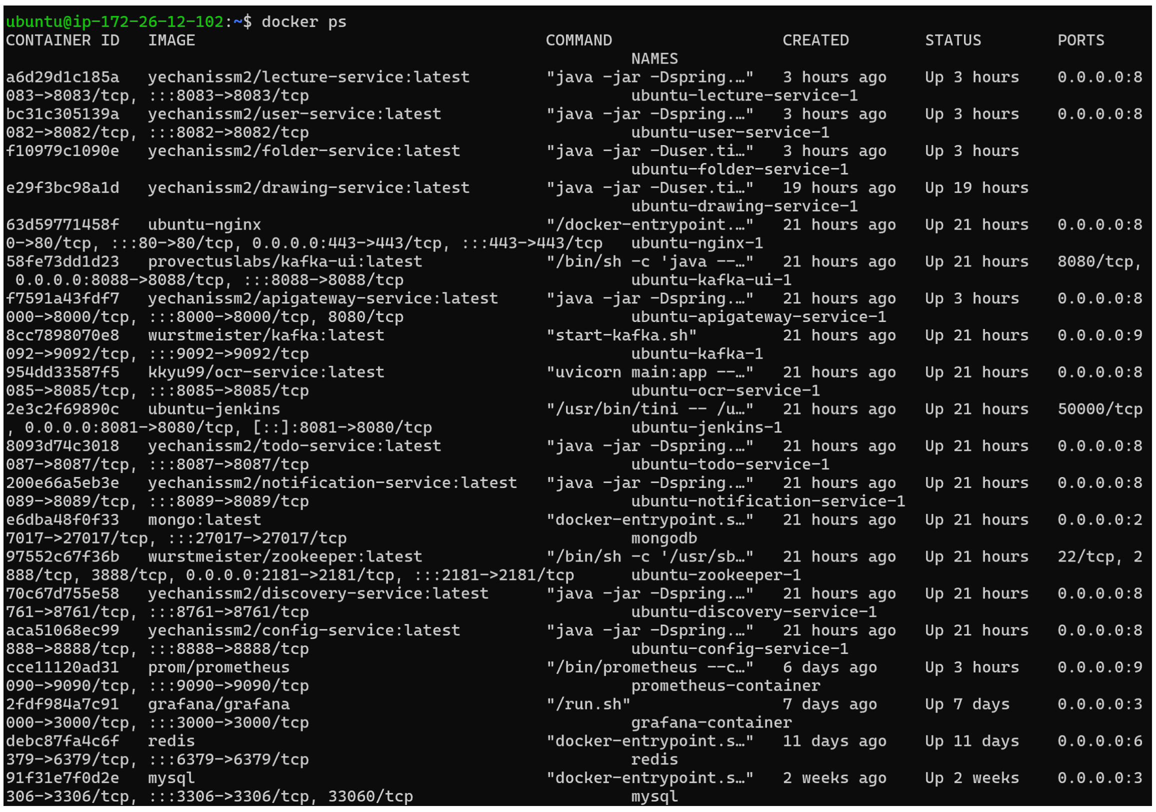
3. Docker Compose 설치

1. Docker 컨테이너 생성

: 백엔드 Spring서버(discovery, config, apigateway, user, lecture, drawing, folder, ocr, todo, nofification)

mysql, mongodb, redis, kafka, kafka-ui

- `docker ps` 결과



```
ubuntu@ip-172-26-12-102:~$ docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED         STATUS          PORTS
                                                                  NAMES
a6d29d1c185a   yechanissm2/lecture-service:latest        "java -jar -Dspring.…"   3 hours ago     Up 3 hours      0.0.0.0:8
083->8083/tcp, :::8083->8083/tcp                                  ubuntu-lecture-service-1
bc31c305139a   yechanissm2/user-service:latest           "java -jar -Dspring.…"   3 hours ago     Up 3 hours      0.0.0.0:8
082->8082/tcp, :::8082->8082/tcp                                  ubuntu-user-service-1
f10979c1090e   yechanissm2/folder-service:latest         "java -jar -Duser.ti…"   3 hours ago     Up 3 hours
                                                                  ubuntu-folder-service-1
e29f3bc98a1d   yechanissm2/drawing-service:latest        "java -jar -Duser.ti…"   19 hours ago    Up 19 hours
                                                                  ubuntu-drawing-service-1
63d59771458f   ubuntu-nginx                              "/docker-entrypoint.…"   21 hours ago    Up 21 hours     0.0.0.0:8
0->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   ubuntu-nginx-1
58fe73dd1d23   provectuslabs/kafka-ui:latest             "/bin/sh -c 'java --…"   21 hours ago    Up 21 hours     8080/tcp,
 0.0.0.0:8088->8088/tcp, :::8088->8088/tcp                        ubuntu-kafka-ui-1
f7591a43fdf7   yechanissm2/apigateway-service:latest     "java -jar -Dspring.…"   21 hours ago    Up 3 hours      0.0.0.0:8
000->8000/tcp, :::8000->8000/tcp, 8080/tcp                        ubuntu-apigateway-service-1
8cc7898070e8   wurstmeister/kafka:latest                 "start-kafka.sh"         21 hours ago    Up 21 hours     0.0.0.0:9
092->9092/tcp, :::9092->9092/tcp                                  ubuntu-kafka-1
954dd33587f5   kkyu99/ocr-service:latest                 "uvicorn main:app --…"   21 hours ago    Up 21 hours     0.0.0.0:8
085->8085/tcp, :::8085->8085/tcp                                  ubuntu-ocr-service-1
2e3c2f69890c   ubuntu-jenkins                            "/usr/bin/tini -- /u…"   21 hours ago    Up 21 hours     50000/tcp
, 0.0.0.0:8081->8080/tcp, [::]:8081->8080/tcp                     ubuntu-jenkins-1
8093d74c3018   yechanissm2/todo-service:latest           "java -jar -Dspring.…"   21 hours ago    Up 21 hours     0.0.0.0:8
087->8087/tcp, :::8087->8087/tcp                                  ubuntu-todo-service-1
200e66a5eb3e   yechanissm2/notification-service:latest   "java -jar -Dspring.…"   21 hours ago    Up 21 hours     0.0.0.0:8
089->8089/tcp, :::8089->8089/tcp                                  ubuntu-notification-service-1
e6dba48f0f33   mongo:latest                              "docker-entrypoint.s…"   21 hours ago    Up 21 hours     0.0.0.0:2
7017->27017/tcp, :::27017->27017/tcp                              mongodb
97552c67f36b   wurstmeister/zookeeper:latest             "/bin/sh -c '/usr/sb…"   21 hours ago    Up 21 hours     22/tcp, 2
888/tcp, 3888/tcp, 0.0.0.0:2181->2181/tcp, :::2181->2181/tcp      ubuntu-zookeeper-1
70c67d755e58   yechanissm2/discovery-service:latest      "java -jar -Dspring.…"   21 hours ago    Up 21 hours     0.0.0.0:8
761->8761/tcp, :::8761->8761/tcp                                  ubuntu-discovery-service-1
aca51068ec99   yechanissm2/config-service:latest         "java -jar -Dspring.…"   21 hours ago    Up 21 hours     0.0.0.0:8
888->8888/tcp, :::8888->8888/tcp                                  ubuntu-config-service-1
cce11120ad31   prom/prometheus                           "/bin/prometheus --c…"   6 days ago      Up 3 hours      0.0.0.0:9
090->9090/tcp, :::9090->9090/tcp                                  prometheus-container
2fdf984a7c91   grafana/grafana                           "/run.sh"                7 days ago      Up 7 days       0.0.0.0:3
000->3000/tcp, :::3000->3000/tcp                                  grafana-container
debc87fa4c6f   redis                                     "docker-entrypoint.s…"   11 days ago     Up 11 days      0.0.0.0:6
379->6379/tcp, :::6379->6379/tcp                                  redis
91f31e7f0d2e   mysql                                     "docker-entrypoint.s…"   2 weeks ago     Up 2 weeks      0.0.0.0:3
306->3306/tcp, :::3306->3306/tcp, 33060/tcp                       mysql
```

`/home/ubuntu/` **`Dockerfiles`** 경로에 docker-compose파일 모아둠

```
$tree ./
├── docker-compose.yml
├── jenkins
│   └── Dockerfile
├── monitoring
│   └── prometheus.yml
└── nginx
    ├── Dockerfile
    ├── conf.d
    │   └── default.conf
    └── ssl
        ├── k11d101.p.ssafy.io.crt
        └── k11d101.p.ssafy.io.key
```

- 실행

  $ `docker compose up -d`

- 컨테이너 접속

  $ `sudo docker exec -it [컨테이너 이름] bash`

2. Nginx 설치 + SSL 인증키 발급

1. Nginx 설치

   $ `sudo apt update && sudo apt upgrade`

   $ `sudo apt install nginx`

   $ `sudo service nginx start`

2. Encrypt, Certbot 설치

   $ `sudo apt-get install letsencrypt`

   $ `sudo apt-get install certbot python3-certbot-nginx`

3. SSL 인증서 발급

   ```
   # Certbot 동작 (nginx 중지하고 해야함)
   $ sudo systemctl stop nginx

   # Nginx 상태확인 & 80번 포트 확인
   $ sudo service nginx status
   $ netstat -na | grep '80.*LISTEN'

   # SSL 인증서 발급 (인증서 적용 및 .pem 키 발급)
   $ sudo certbot --nginx
   $ sudo letsencrypt certonly --standalone -d k11d101.p.ss
   afy.io
   ```

```
# 설치한 인증서 확인 및 위치 확인
$ sudo certbot certificates

# nginx 설정 적용
# nginx 재시작
$ sudo service nginx restart
$ sudo systemctl reload nginx
```

📄Nginx conf 설정

- nginx 설정파일

  $ `sudo vim /etc/nginx/sites-available/default`

  + https (ssl 키 적용) , `service-url.inc` 를 통한 무중단 배포 진행

```
server {
    listen 80;
    server_name k11d101.p.ssafy.io;

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

server {
    listen 443 ssl;
    server_name k11d101.p.ssafy.io;

    ssl_certificate /etc/nginx/ssl/k11d101.p.ssafy.io.crt;
    ssl_certificate_key /etc/nginx/ssl/k11d101.p.ssafy.io.
    ssl_prefer_server_ciphers on;

    client_max_body_size 50M;

    location /api/ {
        proxy_pass http://apigateway-service:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
    location /actuator/ {
        proxy_pass http://apigateway-service:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }


    location /ws-gateway/ {
        proxy_pass http://apigateway-service:8000;
        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_forw
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
        proxy_connect_timeout 3600s;
    }

    location /jenkins/ {
        proxy_pass http://jenkins:8080/jenkins/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }

    location / {
        root /usr/share/nginx/html;
        index index.html index.html;
    }
}
```

- nginx Docker file

```
FROM nginx:latest

# Nginx 설정 파일 복사
COPY conf.d/default.conf /etc/nginx/conf.d/default.conf

# SSL 인증서 복사
COPY ssl/k11d101.p.ssafy.io.crt /etc/nginx/ssl/k11d101.
p.ssafy.io.crt
COPY ssl/k11d101.p.ssafy.io.key /etc/nginx/ssl/k11d101.
p.ssafy.io.key
```

3. Jenkins 설정

📃 젠킨스 파이프라인 스크립트

: 특정 브랜치(backend-develop)를 추적하여 자동 배포가 진행하도록 한다.

post{} 는 mattermost 알림을 위한 설정

▼ **백엔드**

```
pipeline {
    agent any

    environment {
        DOCKERHUB_CREDENTIALS = 'dockerhub-credentials'
        DEPLOY_SERVER_CREDENTIALS = 'ec2-ssh-key'
        DEPLOY_SERVER = 'ubuntu@k11d101.p.ssafy.io'
        CHECK_SERVER = 'k11d101.p.ssafy.io'
        PROJECT_ROOT = '/home/ubuntu'
        SERVICE_NAME = {{ service_name }}
        SERVICE_DIR = "backend/${SERVICE_NAME}"
        IMAGE_NAME = "yechanissm2/${SERVICE_NAME}"
    }

    stages {
        stage('Checkout') {
            steps {
                deleteDir()
                checkout scm
            }
        }

        stage('Check for Changes') {
            steps {
                script {
                    def buildRequired = false
                    for (changeSet in currentBuild.changeS
                        for (entry in changeSet.items) {
                            for (file in entry.affectedFil
                                if (file.path.startsWith("
                                    buildRequired = true
                                    echo "Changes detected
                                    break
                                }
                            }
                            if (buildRequired) {
                                break
                            }
                        }
                        if (buildRequired) {
                            break
                        }
                    }
                    env.BUILD_REQUIRED = buildRequired.toS
                    echo "BUILD_REQUIRED: ${env.BUILD_REQU
                }
            }
        }

        stage('Build Docker Image') {
            when {
                expression { return env.BUILD_REQUIRED ==
            }
            steps {
                script {
                    docker.build("${env.IMAGE_NAME}:latest
                }
            }
        }

        stage('Push to DockerHub') {
            when {
                expression { return env.BUILD_REQUIRED ==
            }
            steps {
                script {
                    docker.withRegistry('https://registry.
                        docker.image("${env.IMAGE_NAME}:la
                    }
                }
            }
        }

        stage('Deploy to Server') {
            when {
                expression { return env.BUILD_REQUIRED ==
            }
            steps {
                sshagent([env.DEPLOY_SERVER_CREDENTIALS])
                    sh """
                        ssh -o StrictHostKeyChecking=no ${
                            cd ${env.PROJECT_ROOT} &&
                            docker-compose stop ${SERVICE_
                            docker-compose rm -f ${SERVICE_
                            docker-compose pull ${SERVICE_
                            docker-compose up -d --force-r
                            docker image prune -f --filter
                        '
                    """
                }
            }
        }
    }
    post {
        success {
            script {
                if (env.BUILD_REQUIRED == 'true') {
                    mattermostSend(color: 'good', message:
                } else {
                    echo "{{ service_name }} 변경 사항 없음.
                }
            }
        }
        failure {
            script {
                if (env.BUILD_REQUIRED == 'true') {
                    mattermostSend(color: 'danger', messag
                }
            }
        }
    }
}
```

Jenkins Dockerfile 생성

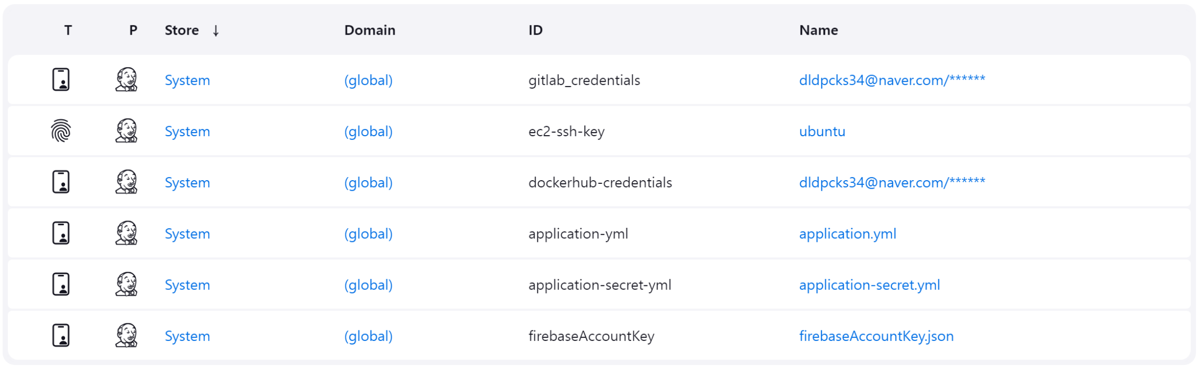
Dockerfile

```
FROM jenkins/jenkins:lts

# root 사용자로 변경
USER root

# Docker, Git 및 Buildx 설치
RUN apt-get update && \
    apt-get install -y docker.io git curl && \
    mkdir -p /usr/lib/docker/cli-plugins && \
    curl -SL https://github.com/docker/buildx/releases/dowr
    chmod +x /usr/lib/docker/cli-plugins/docker-buildx && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
```

```
# Jenkins 사용자를 docker 그룹에 추가
RUN usermod -aG docker jenkins
```

🔑 Credential 관리

빌드에 필요한 env 파일들을 저장해두고 배포 시 파일을 옮겨 서버에 올린다.



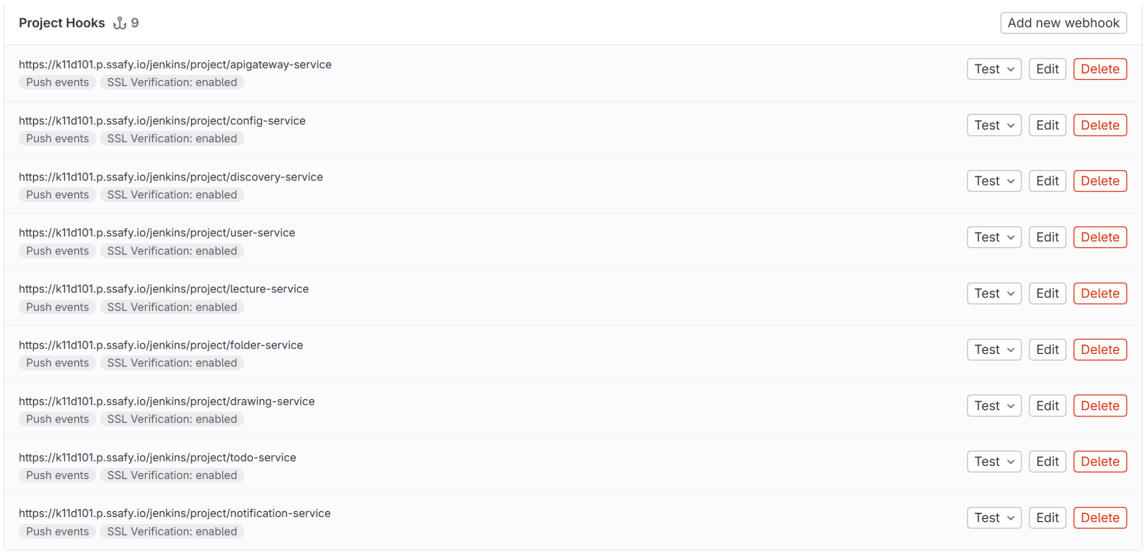
T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab_credentials	dldpcks34@naver.com/******
		System	(global)	ec2-ssh-key	ubuntu
		System	(global)	dockerhub-credentials	dldpcks34@naver.com/******
		System	(global)	application-yml	application.yml
		System	(global)	application-secret-yml	application-secret.yml
		System	(global)	firebaseAccountKey	firebaseAccountKey.json

- **gitlab_credentials** : gitlab의 프로젝트를 clone 해오기위한 credential
  - `gitlab_token` : gitlab API 토큰
  - `gitlab` : gitlab ID/PW
- `ec2-ssh-key` : jenkins에서 우리의 aws ec2의 ssh에 접속하기위한 credential
- **dockehub-credentials** : dockerhub에 있는 이미지를 끌어오기 위함
  - `DOCKER_USER` , `DOCKER_FE_USER` : 도커허브 아이디 / 비밀번호
  - `DOCKER_REPO` , `DOCKER_FE_REPO` : 도커허브 nameSpace / 도커허브 RepositoryName
- **백엔드 설정파일들**

  프로젝트 최종 배포시 중요한 정보들이 들어있는 Spring, React 설정 파일들을 gitlab에 올리지않기 때문에 Jenkins에 미리 저장 해두고 파이프라인 속 build 전 단계에 가져오기 위함
  - `application.yml` : 백엔드 SpringBoot yml파일
  - `application-secret.yml` : 백엔드 SpringBoot yml파일
- **firebaseAccountKey 관련**
  - 알림을 위한 firebase 알림키

Gitlab 웹훅 설정

- 백엔드 ; backend-develop 브랜치



Project Hooks 9 | Add new webhook

https://k11d101.p.ssafy.io/jenkins/project/apigateway-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/config-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/discovery-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/user-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/lecture-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/folder-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/drawing-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/todo-service
Push events SSL Verification: enabled

https://k11d101.p.ssafy.io/jenkins/project/notification-service
Push events SSL Verification: enabled

젠킨스 플러그인 추가 설치

- GitLab
- SSH Agent
- Pipeline Graph View
- Mattermost Notification

4. 배포 위한 파일 생성

[Backend - Spring]

1. **SpringBoot Dockerfile 생성**

   Dockerfile

```
FROM gradle:7.6-jdk17 AS builder
WORKDIR /app
COPY . .
RUN chmod +x gradlew && \
    ./gradlew clean build -x test --no-daemon
```

```
# 실행 단계
FROM openjdk:17-alpine
# 타임존 설정
RUN apk add --no-cache tzdata && \
    cp /usr/share/zoneinfo/Asia/Seoul /etc/localtime && \
    echo "Asia/Seoul" > /etc/timezone

# 빌드된 JAR 파일 복사
COPY --from=builder /app/build/libs/*.jar /app/service.jar

# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod
```

2. **DockerHub에 올린 이미지를 가져와 docker compose로 서버 띄우기**

   $ `vi /home/ubuntu/docker-compose.yml`

   📃 **docker-compose.yml**

```
version: '3'
services:
 config-service:
   image: yechanissm2/config-service:latest
   restart: always
   ports:
     - "8888:8888"
   networks:
     - msa-network

 discovery-service:
   image: yechanissm2/discovery-service:latest
   restart: always
   ports:
     - "8761:8761"
   depends_on:
     - config-service
   networks:
     - msa-network
```

```
mongodb:
  image: mongo:latest
  container_name: mongodb
  restart: always
  ports:
    - "27017:27017"
  volumes:
    - mongo_data:/data/db
  environment:
    MONGO_INITDB_ROOT_USERNAME: silvertown
    MONGO_INITDB_ROOT_PASSWORD: silvertown
  depends_on:
    - discovery-service
  networks:
    - msa-network

zookeeper:
  image: wurstmeister/zookeeper:latest
  platform: linux/amd64
  ports:
    - "2181:2181"
  environment:
    ZOOKEEPER_CLIENT_PORT: 2181
    ZOOKEEPER_TICK_TIME: 2000
  depends_on:
    - discovery-service
  networks:
    - msa-network

kafka:
  image: wurstmeister/kafka:latest
  platform: linux/amd64
  ports:
    - "9092:9092"
  depends_on:
    - zookeeper
  environment:
    KAFKA_ADVERTISED_LISTENERS: INSIDE://kafka:29092
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLA
    KAFKA_LISTENERS: INSIDE://0.0.0.0:29092,OUTSIDE:
    KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
    KAFKA_ZOOKEEPER_CONNECT: ubuntu-zookeeper-1:2181
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  networks:
    - msa-network

jenkins:
  build: ./jenkins
  ports:
    - "8081:8080"
  environment:
    - JENKINS_OPTS=--prefix=/jenkins
  volumes:
    - jenkins_home:/var/jenkins_home
    - /var/run/docker.sock:/var/run/docker.sock
    - /usr/bin/docker:/usr/bin/docker
  depends_on:
    - discovery-service
  networks:
    - msa-network

user-service:
  image: yechanissm2/user-service:latest
  restart: always
  ports:
    - "8082:8082"
  depends_on:
    - discovery-service
    - mongodb
  networks:
    - msa-network

lecture-service:
  image: yechanissm2/lecture-service:latest
  restart: always
  ports:
    - "8083:8083"
  depends_on:
    - discovery-service
    - mongodb
  networks:
    - msa-network

folder-service:
  image: yechanissm2/folder-service:latest
  restart: always
  depends_on:
    - discovery-service
    - mongodb
  networks:
    - msa-network

drawing-service:
  image: yechanissm2/drawing-service:latest
  restart: always
  depends_on:
    - discovery-service
    - mongodb
  networks:
    - msa-network

ocr-service:
  image: kkyu99/ocr-service:latest
  restart: always
  ports:
    - "8085:8085"
  depends_on:
    - discovery-service
  networks:
    - msa-network

todo-service:
  image: yechanissm2/todo-service:latest
  restart: always
  depends_on:
    - discovery-service
  ports:
    - "8087:8087"
  networks:
    - msa-network

notification-service:
  image: yechanissm2/notification-service:latest
  restart: always
  depends_on:
    - discovery-service
  ports:
    - "8089:8089"
  networks:
    - msa-network

kafka-ui:
  image: provectuslabs/kafka-ui:latest
  platform: linux/amd64
  ports:
    - "8088:8088"
  environment:
    SERVER_PORT: 8088
    KAFKA_CLUSTERS_0_NAME: k11d101.p.ssafy.io
    KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka:29092
    KAFKA_CLUSTERS_0_ZOOKEEPER: ubuntu-zookeeper-1:2
    KAFKA_CLUSTERS_0_READONLY: "false"
  depends_on:
    - kafka
  networks:
    - msa-network

apigateway-service:
  image: yechanissm2/apigateway-service:latest
  restart: always
  ports:
    - "8000:8000"
  depends_on:
    - config-service
    - discovery-service
    - user-service
    - lecture-service
    - folder-service
    - ocr-service
    - drawing-service
    - todo-service
    - notification-service
    - jenkins
    - mongodb
    - kafka
  networks:
    - msa-network

nginx:
  build: ./nginx
  ports:
    - "80:80"
    - "443:443"
  depends_on:
    - apigateway-service
  volumes:
    - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d
  networks:
    - msa-network

networks:
msa-network:
  external: true

volumes:
jenkins_home:
mongo_data:
```