

D102 포팅매뉴얼

환경 설정

1. 개발 환경

- A. Frontend
- B. Backend
- C. CI/CD
- D. 협업 툴

2. 설정 파일

빌드 및 배포

1. 준비

- A. Ubuntu에 Docker 설치
- B. Docker Compose 설치
- C. Jenkins 설치
- D. Jenkins와 Gitlab 연동
- E. AWS 설정

2. 빌드, 배포

- A. NGINX
- B. Docker
- C. Jenkins Pipeline

시연 시나리오

환경 설정

1. 개발 환경

A. Frontend

- **React** : 18.2.0
- **Node.js** : 20.12.7
- **Tailwindcss** : 3.4.3
- **Vite** : 5.2.0
- **Typescript** : 5.2.2
- **React-router-dom** : 6.22.3
- **Axios** : 1.6.8
- **Styled-components** : 6.1.8
- **상태 관리**
 - **tanstack / react-query** : ^5.32.0
 - **zustand** : ^4.5.2

B. Backend

- **JDK** : OpenJDK 17
- **Spring Boot** : 3.2.5
- **Spring Boot JPA** : 3.2.5
- **QueryDSL** : 5.0.0
- **MySQL** : 8.3.0
- **Redis** : 7.2.4
- **Spring Security 6, JWT** 0.12.3
- **Lombok, Swagger, Spring Websocket, STOMP**
- **AWS S3, AWS CloudFront**

C. CI/CD

- **AWS EC2**

- **Docker & Docker Compose** : 2.26.1
- **Nginx** : 1.18.0
- **Jenkins** : 2.455

D. 협업 툴

- **Jira** : 이슈 추적
- **GitLab** : 형상 관리
- **Mattermost, Notion** : 소통, 산출물 관리
- **Figma** : 와이어프레임, 디자인

2. 설정 파일

- application.yml

```
server:
  port: 8081
  servlet:
    context-path: /api

spring:
  profiles:
    active: server
  group:
    local:
      - db-local
    server:
      - db-server
  include:
    - key
    - db

  servlet:
    multipart:
      max-file-size: 30MB
      max-request-size: 30MB

  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update

  properties:
    hibernate:
      format_sql: true
    jdbc:
      time_zone: Asia/Seoul
  show-sql: true
  open-in-view: false

springdoc: #swagger
  packages-to-scan: com.mnot.quizdot
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  swagger-ui:
    path: /quizdot.html
    tags-sorter: alpha
    operations-sorter: alpha
  api-docs:
    path: /api-docs/json
```

```

    groups:
      enabled: true
    cache:
      disabled: true

cloud:
  aws:
    s3:
      bucket: ${s3.bucket}
    credentials:
      access-key: ${s3.access-key}
      secret-key: ${s3.secret-key}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
    cloudfront:
      domain: d3eb2qgkbb4eu.cloudfront.net

logging:
  level:
  org:
    hibernate:
      orm:
        jdbc:
          bind: trace

```

- application-db.yml

```

---
spring:
  config:
    activate:
      on-profile: db-local
  datasource:
    driver-class-name: ${local.db.driver}
    url: ${local.db.url}
    username: ${local.db.username}
    password: ${local.db.password}
  data:
    redis:
      host: ${local.redis.host}
      port: ${local.redis.port}
      username: ${local.redis.username}
      password: ${local.redis.password}
---
spring:
  config:
    activate:
      on-profile: db-server
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${server.db.url}
    username: ${server.db.username}
    password: ${server.db.password}
  data:
    redis:
      host: ${server.redis.host}
      port: ${server.redis.port}
      username: ${server.redis.username}
      password: ${server.redis.password}

```

- application-key.yml

```
spring:
  jwt:
    secret: wlqdpkrhtlvekwlqdpkrhtlvektlsdudgkschl rh

local:
  db:
    driver: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/quizdot?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: root
    password: root

  redis:
    host: localhost
    port: 6379
    username:
    password:

server:
  db:
    driver: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://mysql:3306/quizdot?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8 # mysql 컨테이너 서비스명:내부포트
    username: quizdot
    password: # password

  redis:
    host: redis # redis 컨테이너 서비스 명
    port: 6379
    username: default
    password: # password

s3:
  bucket: quizdot
  access-key: # S3 access-key
  secret-key: # S3 secret-key
```


빌드 및 배포


1. 준비

A. Ubuntu에 Docker 설치

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install Docker Engine on Ubuntu.

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



1. 이전 버전 삭제

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do su do apt-get remove $pkg; done
```

2. apt 업데이트 및 패키지 설치

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

```
sudo install -m 0755 -d /etc/apt/keyrings
```

3. 도커 공식 GPG Key 추가

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. stable repository로 등록

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

5. Docker 패키지 설치

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

6. 설치 확인

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

B. Docker Compose 설치

Install Compose standalone
How to install Docker Compose - Other Scenarios

 <https://docs.docker.com/compose/install/standalone/>



1. docker compose 다운로드 및 설치

```
sudo curl -SL https://github.com/docker/compose/releases/download/v2.24.6/docker-compose-linux-x86_64  
-o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

2. 설치 확인

```
docker-compose -v
```

C. Jenkins 설치

1. docker-compose.yml 작성 (전체 내용 2.B.Docker 참고)

```
version: '3'  
  
services:  
  jenkins:  
    image: jenkins/jenkins:lts  
    container_name: jenkins  
    environment:  
      - TZ=Asia/Seoul  
    volumes:  
      - /usr/bin/docker:/usr/bin/docker  
      - /var/run/docker.sock:/var/run/docker.sock  
      - /var/jenkins_home:/var/jenkins_home  
    restart : always  
    ports:
```

```
- "9090:8080"
user: root
networks:
- app-network
```

2. 실행

```
sudo docker-compose up -d jenkins
```

3. 실행 확인

```
sudo docker ps
```

4. 설정

a. `http://도메인:docker-compose.yml에서 설정한 포트` 에서 진행

b. 초기 비밀번호 입력

```
# 로그 및 설정 파일에서 확인 가능
sudo docker logs [container ID]
sudo cat /var/jenkins_home/secrets/initialAdminPassword
```

c. 플러그인 설치

- Install suggested plugins 선택

d. 계정 설정

- 계정명, 암호, 이름, 이메일 주소 설정

e. 추가 플러그인 설치

- i. gitlab
- ii. ssh agent
- iii. nodejs plugin

D. Jenkins와 Gitlab 연동

1. Jenkins Credential 생성

- gitlab에 사용한 이메일로 username 작성, Password와 ID 입력

New credentials

Kind

Username with password

Scope ?

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

Username ?

☐ Treat username as secret ?

Password ?

ID ?

Description ?

Create

2. Gitlab connection 등록

- Jenkins 관리 - System에서 등록. 아래에 보이는 부분을 설정한다.

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?
A name for the connection

! GitLab connection name required.

GitLab host URL ?
The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

! GitLab host URL required.

Credentials ?
API Token for accessing GitLab

+ Add ▾

고급 ▾

- GitLab API Token은 내 gitlab 계정의 Access Token을 추가한다.
토큰은 생성 직후에만 볼 수 있음을 주의한다.
- Credentials에 미리 추가하지 않았다면, +Add를 눌러서 바로 추가할 수 있다.
Kind를 GitLab API token으로 선택하여 Credential을 생성한다.

Personal Access Tokens

Active personal access tokens ⓘ 1

Add a personal access token

Token name

For example, the application using the token or the purpose of the token.

Expiration date
2024-06-17 ⓘ ⓘ

Select scopes
Scopes set the permission levels granted to the token. [Learn more.](#)

☒ api
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.
☐ read_api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
☒ read_user
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
☐ create_runner
Grants create access to the runners.
☐ k8s_proxy
Grants permission to perform Kubernetes API calls using the agent for Kubernetes.
☒ read_repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
☐ write_repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
☐ all_features
Grants access to GitLab Duo related API endpoints.

Create personal access token Cancel

3. Jenkins Project 생성

- a. Pipeline 프로젝트 생성
- b. Build Triggers : gitlab에서 push 될 때 빌드 되도록 설정
- c. 고급 - Secret token 생성

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: [redacted] ?

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 ?

Rebuild open Merge Requests ?

Never

☒ Approved Merge Requests (EE-only) ?

☒ Comments ?

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급

Secret token ?

[redacted]

Generate

4. Webhook 연결

- a. gitlab에서 프로젝트 settings - webhooks 선택
- b. URL : Jenkins 프로젝트에서 저장한 GitLab webhook URL 입력
- c. Secret Token : Jenkins 프로젝트에서 생성한 토큰 입력
- d. Trigger : 특정 브랜치에 push가 일어나는 등, 빌드가 수행될 조건을 선택해준다.

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.

Project Hooks 3

URL

http://example.com/trigger-ci.json

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

All branches

☒ Wildcard pattern

Wildcard patterns such as */stable or production/* are supported.

Regular expression

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

- e. Test - Push events(Trigger에서 설정한 조건) 에서 테스트

Recent events

GitLab events trigger webhooks. Use the request details of a webhook to help troubleshoot problems. [How do I troubleshoot?](#)

Status	Trigger	Elapsed time	Request time	
200	Push Hook	0.05 sec	1 day ago	View details
200	Push Hook	0.04 sec	1 day ago	View details

5. SSH credentials 추가

D102 포팅매뉴얼

8

New credentials

Kind: SSH Username with private key

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

ID:

Description:

Username:

☐ Treat username as secret

Private Key: ☒ Enter directly

Key: No Stored Value

Add

- ID : 사용할 ID
- username : EC2 ubuntu의 계정명 입력
- Private Key : pem 키를 메모장으로 열어서 내용 복사-붙여넣기

6. 설정파일 credentials 추가

- .yml 파일 혹은 .env 파일을 업로드하고 ID를 설정한다.

New credentials

Kind: Secret file

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

File: 선택된 파일 없음

ID:

Description:

Create

7. Pipeline 작성 (내용은 2.C. Jenkins Pipeline 참고)

Pipeline

Definition: Pipeline script

```

1= pipeline {
2=   agent any
3=
4=   stages {
5=     stage('Clone Repository') {
6=       steps {
7=         git branch: 'BE/ik',
8=           credentialsId: 'quidiot',
9=           url: 'https://lab.ssafty.com/s10-final/S10P310102'
10=        }
11=      }
12=
13=     stage('Build Spring-boot') {
14=       steps {
15=         dir('quidiot-server') {
16=           script {
17=
18=           }
19=         }
20=       }
21=     }
22=   }
23= }
  
```

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

E. AWS 설정

1. S3 버킷 생성

- AWS 리전은 ap-northeast-2(아시아 태평양 (서울))로 설정하고 버킷 이름을 입력한다.
- 모든 퍼블릭 액세스는 차단 한다.
- 버킷 버전 관리 : 비활성화

2. S3와 Cloudfront 연결

- Cloudfront : AWS Global Edge Server를 통해 CDN(Content Delivery Network) 역할을 해주는 AWS 서비스

- S3 버킷과 Cloudfront를 연결 후 OAI 설정으로 연결한 Cloudfront를 통해서만 버킷에 접근할 수 있도록 설정하여, S3를 퍼블릭으로 공개하지 않고도 Cloudfront를 통해 S3에 퍼블릭하게 접근 할 수 있다.
- Cloudfront를 우회하여 S3에 직접 접근할 수 없도록 한다.

배포 생성

원본

Origin domain
Choose an AWS origin, or enter your origin's domain name.

quizdot.s3.ap-northeast-2.amazonaws.com

Origin path - optional
Enter a URL path to append to the origin domain name for origin requests.

Enter the origin path

이름
이 원본의 이름을 입력합니다.

quizdot.s3.ap-northeast-2.amazonaws.com

원본 액세스 | 정보

☐ 공개
버킷은 공개 액세스를 허용해야 합니다.

☐ 원본 액세스 제어 설정(권장)
버킷은 CloudFront에 대한 액세스만 제한할 수 있습니다.

☒ Legacy access identities
CloudFront 원본 액세스 ID(OAI)를 사용하여 S3 버킷에 액세스합니다.

원본 액세스 ID
기존 원본 액세스 ID를 선택하거나(권장) 새 ID를 생성합니다.

quizdot.s3.ap-northeast-2.amazonaws.com

새 OAI 생성

버킷 정책
OAI에 대한 읽기 액세스를 허용하도록 S3 버킷 정책을 업데이트합니다.

☐ 아니요. 제가 버킷 정책을 업데이트하겠습니다

☒ 예, 버킷 정책 업데이트

- 기본 캐시 동작에서 Redirect HTTP to HTTPS를 설정하여 http:// 로 접근하더라도 https:// 로 redirect 될 수 있도록 한다.

기본 캐시 동작

경로 패턴

정보

기본값(*)

자동으로 객체 압축

정보

☐ No
☒ Yes

뷰어

뷰어 프로토콜 정책

☐ HTTP and HTTPS
☒ Redirect HTTP to HTTPS
☐ HTTPS only

허용된 HTTP 방법

☒ GET, HEAD
☐ GET, HEAD, OPTIONS
☐ GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

뷰어 액세스 제한

뷰어 액세스를 제한하는 경우 뷰어는 CloudFront 서명된 URL 또는 서명된 쿠키를 사용하여 사용자의 콘텐츠에 액세스해야 합니다.

☒ No
☐ Yes

캐시 키 및 원본 요청

캐시 정책 및 원본 요청 정책을 사용하여 캐시 키 및 원본 요청을 제어할 것을 권장합니다.

Cloudfront의 OAI가 추가된 S3 버킷의 정책



이 버킷에 대해 퍼블릭 액세스 차단 설정이 활성화되어 있기 때문에 퍼블릭 액세스가 차단됩니다.
 활성화된 설정을 확인하려면 이 버킷의 퍼블릭 액세스 차단 설정을 확인하세요. [Amazon S3 퍼블릭 액세스](#)

```
{
  "Version": "2012-10-17",
  "Id": "Policy1714577834911",
  "Statement": [
    {
      "Sid": "2",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity E52VK0ADSDSH4"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::quizdot/*"
    }
  ]
}
```

2. 빌드, 배포

A. NGINX

- nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
```

```

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

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

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
    ssl_prefer_server_ciphers on;

    server {
        listen 80;
        server_name k10d102.p.ssafy.io;
        client_max_body_size 10M;

        location /api {
            #백엔드
            proxy_pass http://k10d102.p.ssafy.io:8081;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }

        location / {
            #프론트엔드
            proxy_pass http://k10d102.p.ssafy.io:8082;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_read_timeout 1m;
            proxy_connect_timeout 1m;

            proxy_http_version 1.1;

            proxy_set_header Origin "";

            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }

        location /sub {
            proxy_pass http://k10d102.p.ssafy.io:8082;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
            proxy_set_header Origin "";
        }

        location ~* ^/(api/ws|pub) {
            proxy_pass http://k10d102.p.ssafy.io:8081;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;

```

```

        proxy_set_header Connection "upgrade";

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

    location /ws {
        proxy_pass http://k10d102.p.ssafy.io:8082;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header Origin "";
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name k10d102.p.ssafy.io;
    client_max_body_size 100M;

    ssl_certificate /etc/letsencrypt/live/p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/p.ssafy.io/privkey.pem;

    location /api {
        #백엔드
        proxy_pass http://k10d102.p.ssafy.io:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
        #프론트엔드
        proxy_pass http://k10d102.p.ssafy.io:8082;
        proxy_read_timeout 1m;
        proxy_connect_timeout 1m;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Origin "";

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location /sub {
        proxy_pass http://k10d102.p.ssafy.io:8082;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header Origin "";
    }

    location ~* ^/(api/ws|pub) {
        proxy_pass http://k10d102.p.ssafy.io:8081;
        proxy_http_version 1.1;
    }
}

```

```

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

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

    location /wss {
        proxy_pass http://k10d102.p.ssafy.io:8082;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header Origin "";
    }
}

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}

```

B. Docker

- docker-compose

```

version: '3'

services:
  quizdotbe:
    build:
      context: ./quizdot-server
      dockerfile: Dockerfile
    image: feelgod/quizdotbe:latest
    container_name: quizdotbe
    ports:
      - "8081:8081"
    depends_on:
      - redis
    environment:
      - SPRING_PROFILES_ACTIVE=server
    networks:
      - app-network

  quizdotfe:
    build:
      context: ./quizdot-client
      dockerfile: Dockerfile
    image: feelgod/quizdotfe:latest
    container_name: quizdotfe
    ports:
      - "8082:5173"
    networks:
      - app-network

  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /usr/bin/docker:/usr/bin/docker

```

```

- /var/run/docker.sock:/var/run/docker.sock
- /var/jenkins_home:/var/jenkins_home
restart : always
ports:
- "9090:8080"
user: root
networks:
- app-network

```

```

mysql:
  image: mysql:latest
  container_name: mysql
  ports:
  - "3305:3306"
  environment:
  - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
  command:
  - --character-set-server=utf8mb4
  - --collation-server=utf8mb4_unicode_ci
  volumes:
  - ./mysql_data:/var/lib/mysql
  restart : always
  networks:
  - app-network

redis:
  image: redis:alpine
  command: redis-server --requirepass ${REDIS_PASSWORD}
  container_name: redis
  restart: always
  ports:
  - "6379:6379"
  volumes:
  - ./redis_data:/data
  networks:
  - app-network

```

```

networks:
  app-network:
    external: true

```

- .env

```

MYSQL_ROOT_PASSWORD=비밀번호내용
REDIS_PASSWORD=비밀번호내용

```

- Dockerfile
 - Frontend

```

FROM node:20.12.2 as builder
WORKDIR /app
COPY package.json .
COPY package-lock.json .
COPY . .
RUN npm install
RUN npm run build
EXPOSE 8082
CMD ["npm", "run", "dev"]

```

- Backend

```
FROM eclipse-temurin:17-jdk-alpine
VOLUME /tmp

ARG JAR_FILE=build/libs/*.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java","-Duser.timezone=Asia/Seoul","-jar","/app.jar"]
EXPOSE 8081
```

C. Jenkins Pipeline

- Frontend

```
pipeline {
    agent any

    stages {
        stage('Clone Repository') {
            steps {
                git branch: 'FE/ik',
                    credentialsId: 'quizdot',
                    url: 'https://lab.ssafy.com/s10-final/S10P31D102'
            }
        }

        stage('Build React') {
            steps {
                dir('quizdot-client') {
                    sh '''
                        docker run --rm -v "$PWD":/app -w /app node:20.12.2 /bin/sh -c "
                            npm install &&
                            npm run build
                        "
                    '''
                }
                sh 'docker-compose build quizdotfe'
            }
        }

        stage('Deployment') {
            steps {
                script {
                    sshagent(credentials: ['quizdot_ssh']) {
                        sh '''
                            docker-compose stop quizdotfe
                            docker rm -f quizdotfe
                            docker-compose up -d quizdotfe
                        '''
                    }
                }
            }
        }
    }
    // 공통 : 빌드 여부 성공 알림
}
```

- Backend

```
pipeline {
    agent any
```



```

stages {
    stage('Clone Repository') {
        steps {
            git branch: 'BE/ik',
                credentialsId: 'quizdot',
                url: 'https://lab.ssafy.com/s10-final/S10P31D102'
        }
    }

    stage('Build Spring-boot') {
        steps {
            dir('quizdot-server') {
                script {
                    withCredentials([file(credentialsId: 'application', variable: 'KEY_FILE')]) {
                        sh "cp $KEY_FILE src/main/resources/application-key.yml"
                        sh "chmod +x gradlew"
                        sh "./gradlew bootJar"
                    }
                }
            }
            withCredentials([file(credentialsId: 'dockerEnv', variable: 'ENV_FILE')]) {
                sh '''
                set -a && . $ENV_FILE && set +a
                docker-compose build quizdotbe
                '''
            }
        }
    }

    stage('Deployment') {
        steps {
            script {
                sshagent(credentials: ['quizdot_ssh']) {
                    withCredentials([file(credentialsId: 'dockerEnv', variable: 'ENV_FILE')]) {
                        sh '''
                        set -a && . $ENV_FILE && set +a
                        docker-compose stop quizdotbe redis
                        docker rm -f quizdotbe redis
                        docker-compose up -d quizdotbe redis
                        '''
                    }
                }
                sleep(15)
                // 로그 검사 : 스프링 컨테이너 실행 후 에러 발생 확인
                def logsBe = sh(script: "docker logs quizdotbe", returnStdout: true).trim()
                if (logsBe.contains("Application run failed")) {
                    error("Deployment failed, application run failed found in logs")
                }
            }
        }
    }
}
// 공통 : 빌드 여부 성공 알림
}

```

- 공통 : Mattermost 연동 빌드 성공 여부 알림

```

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()

```

- 최종 Pipeline

D102 포팅매뉴얼

```

        sh 'docker-compose build quizdotfe'
    }
}

stage('Deployment') {
    steps {
        script {
            sshagent(credentials: ['quizdot_ssh']) {
                withCredentials([file(credentialsId: 'dockerEnv', variable: 'ENV_FILE')]) {
                    sh '''
                        set -a && . $ENV_FILE && set +a
                        docker-compose stop quizdotbe quizdotfe redis
                        docker rm -f quizdotbe quizdotfe redis
                        docker-compose up -d quizdotbe quizdotfe redis
                    '''
                }
            }
            sleep(15)
            // 로그 검사 : 스프링 컨테이너 실행 후 에러 발생 확인
            def logsBe = sh(script: "docker logs quizdotbe", returnStdout: true).trim()
            if (logsBe.contains("Application run failed")) {
                error("Deployment failed, application run failed found in logs")
            }
        }
    }
}

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})"
            \n(<${env.BUILD_URL}|Details>"),
            )
        }
    }
    failure {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            mattermostSend (color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})"
            \n(<${env.BUILD_URL}|Details>"),
            )
        }
    }
}
}

```

시연 시나리오

채널 선택

- 동시 접속 유저 확인
- 현재 방 목록 확인

노말 모드

- 방 생성 - 노말 모드, 랜덤 카테고리
- 노말 모드 게임 진행
- 노말 모드 게임 종료

서바이벌 모드

- 방 생성 - 서바이벌 모드, 랜덤 카테고리
- 서바이벌 매칭 시작
 - 처음에는 최소 인원 부족으로 대기 중이라는 메시지가 뜬다
 - 이야기하면서 한 10초정도 시간을 끔
 - 매칭에 성공해서 게임 시작
- 서바이벌 게임 진행
 - 첫 번째 문제에서 두 명 살아 남고, 나머지는 다 죽기
 - 두 번째 문제에서 두 명이 죽고, 탈락자 중에 한 명이 맞춰서 부활
 - 세 번째 문제에서 한 명 빼고 다 죽고 게임 종료
- 서바이벌 게임 종료
 - 게임 결과 확인 - 도전과제(칭호), 레벨 업 등을 확인
 - 원래 있던 방으로 다시 이동

일대일 모드

- 방 생성 - 일대일 모드, 랜덤 카테고리
- 두 명의 플레이어로 시작한다
- 일대일 모드 시작
 - 첫 번째 문제 선택 - 친구가 틀려서 피가 깎이는 걸 확인한다
 - 두 번째 문제 선택 - 문제를 푸는데 친구가 탈주해서 게임 종료 페이지로 이동
- 일대일 모드 종료
 - 게임 결과 확인 - 도전과제(칭호), 레벨 업 등을 확인

마이페이지 확인

- 캐릭터 뽑기
- 닉네임 색상 뽑기
- 내가 획득한 도전과제나 칭호 확인하기