하렴(haryeom) - 포팅 매뉴얼

개발 환경

기본 정보

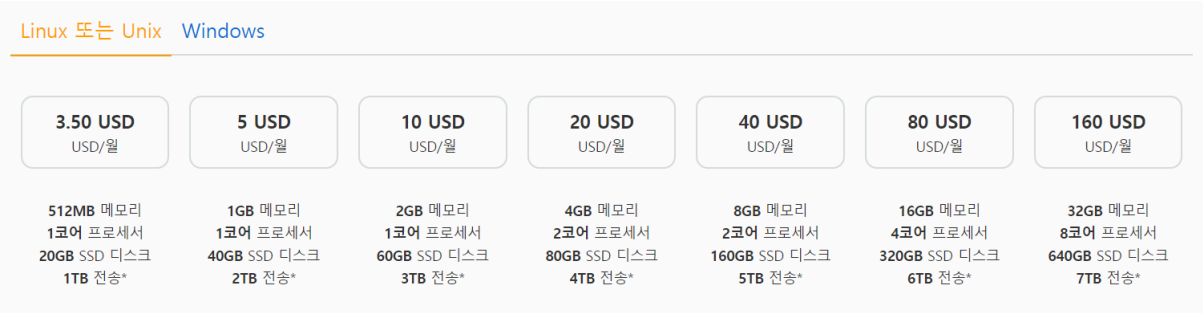
- Java : 11
- Spring Boot : 2.7.18
- gradle : 8.5
- TypeScript : 5.3.3
- Next.js : 14.0.4
- React : 18.2.0
- Node.js : 20.10.5
- yarn : 1.22.21
- MariaDB : 11.2.2
- MongoDB : 7.0.5
- Redis : 7.2.4

사용 IDE 정보

- Intellij : 2023.3.3 (Ultimate Edition)
- VSCode : 1.86.1

배포 환경

서버 구성(AWS)



- AWS Lightsail 기준 80 USD 옵션인 16GB 메모리 / 4코어 프로세서 / 320GS SSD 디스크 / 6TB 전송 인스턴스 또는 이와 대응하는 EC2 인스턴스
    - Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1051-aws x86_64)

서버 환경 설정

서버 방화벽 설정

```
# ufw 설치명령
sudo apt-get install ufw

# ufw 상태확인 명령
sudo ufw status verbose
sudo ufw status

sudo ufw allow 22 # ssh
sudo ufw enable

# ...
sudo ufw status
```

서버 허용 포트 목록

PORT	INFO
22	ssh
80	Nginx
443	Nginx
8080	Back-end Server(server, Spring Boot)
3000	Front-end Server(client, Next.js)
3306	MariaDB
~~27017~~ 8027	MongoDB
8081	Jenkins
6379	Redis

서버 기본 세팅

- Ubuntu의 서버 시간을 UTC에서 KST로 변경
  - AWS의 Ubuntu는 서버 시간 기본값 UTC

```
sudo timedatectl set-timezone Asia/Seoul
```

- 미러 서버를 카카오 서버로 변경
  - 기본 서버가 *.ubuntu.com 이라는 해외망을 이용하는 서버.
  - 국내망을 이용할 수 있는 카카오 미러 서버를 사용.
  - AWS EC2 혹은 AWS Lightsail에서 아래 코드 사용 가능.
    - 타 Ubuntu 서버를 사용할 경우 ap-northeast-2.ec2.archive.ubuntu.com 부분을 “sudo vi /etc/apt/sources.list”으로 확인해서 다른 서버로 변경 후 사용해야 함.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g' /etc/apt/sources.list
```

- 패키지 목록 업데이트 & 패키지 업데이트
  - 미러 서버가 변경됨. 따라서 update & upgrade 진행.

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

- Swap 영역 할당
  - 용량 확인

    ```
    free -h
    ```

  - 할당 (ex. 4GB)

    ```
    sudo fallocate -l 4G /swapfile
    ```

  - swapfile 권한 수정

    ```
    sudo chmod 600 /swapfile
    ```

  - swapfile 생성

    ```
    sudo mkswap /swapfile
    ```

  - swapfile 활성화

    ```
    sudo swapon /swapfile
    ```

  - 시스템 재부팅 시에도 swap 유지 설정

    ```
    sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
    ```

  - 할당 확인

    ```
    free -h
    ```

Docker 환경 설정

Docker 설치

- 버전 정보
  - 25.0.0
- 필요 Package 설치

```
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent s
oftware-properties-common
```

    - apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근 가능하게 함.
    - ca-certificates : certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결이 되어있는지 확인할 수 있다
    - curl : 특정 웹사이트에서 데이터를 다운로드 받을 경우 사용하는 패키지
    - gnupg-agent : OpenPGP 표준 규격의 데이터 통신을 암호화하고 서명할 수 있는 패키지
    - software-properties-common : PPA를 추가하거나 제거할 때 사용
        - PPA : Personal Package Archive(개인 패키지 저장소)를 의미. 캐노니컬社의 우분투에서 기본적으로 제공하는 패키지 외의 사적으로 만든 패키지를 의미
- GPG Key 인증 진행

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Repository 등록
    - AMD64

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linu
x/ubuntu $(lsb_release -cs) stable"
```

    - ARM64

```
sudo add-apt-repository "deb [arch=arm64] https://download.docker.com/linu
x/ubuntu $(lsb_release -cs) stable"
```

- Package List 갱신

```
sudo apt-get -y update
```

- Docker Package 설치

```
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
```

    - apt-get을 이용하여 설치
        - docker-ce : Docker Community Edition
        - docker-ce-cli : Docker Community Edition의 cli 환경에서 추가로 설치해야 하는 패키지
        - containerd.io : Docker 컨테이너 런타임
- 일반 유저에게 권한 부여
    - Docker는 항상 root로 실행되기 때문에 sudo를 사용하여 명령어를 입력해야 함.
        - 사용자를 docker 그룹에 추가하여 sudo를 사용하지 않아도 docker 명령어 사용 가능.

```
sudo usermod -aG docker ubuntu
```

    - Docker 서비스 재시작. 추가적으로 사용자 세션 재로그인 필요

```
sudo service docker restart
```

Docker Compose

- 버전 정보
  - 2.21.0
- 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/dock
er-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- 권한 변경

```
sudo chmod +x /usr/local/bin/docker-compose
```

Jenkins

- 버전 정보
  - 2.426.3
- 도커 이미지 받기(docker image pull)

```
sudo docker pull jenkins:lts
```

```
sudo docker pull jenkins:2.426.3-lts
```

- 도커 컨테이너 실행(docker run)

```
docker run -d --env JENKINS_OPTS=--httpPort=8080 \
-v /etc/localtime:/etc/localtime:ro \
-e TZ=Asia/Seoul -p 8081:8080 \
-v /jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /usr/local/bin/docker-compose:/usr/local/bin/docker-compose \
--name jenkins -u root jenkins/jenkins:lts
```

```
docker run -d --env JENKINS_OPTS=--httpPort=8080 \
-v /etc/localtime:/etc/localtime:ro \
-e TZ=Asia/Seoul -p 8081:8080 \
-v /jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /usr/local/bin/docker-compose:/usr/local/bin/docker-compose \
--name jenkins -u root jenkins/jenkins:2.426.3-lts
```

- 접속

- http://도메인:8081 로 접속
- 암호 입력

```
# 방법 1 : log를 통해 암호 확인.
docker logs jenkins
# 방법 2 : 컨테이너 내부에 접속. 암호 파일을 확인.
docker exec -it jenkins /bin/bash
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- 플러그인 설치
- 계정 생성
- Credential 설정
  - Jenkins 관리 - Credentials
- node 설정
  - Jenkins 관리 - System
- gradle 설정
  - Jenkins 관리 - System
- 필요한 플러그인은 추가적으로 설치
  - Jenkins 관리 - Plugin

Nginx

- 버전 정보
  - 1.25.3
- 도커 이미지 받기(docker image pull)

```
sudo docker pull nginx:latest
```

```
sudo docker pull nginx:1.25.3
```

- 도커 컨테이너 실행(docker run)

```
docker run -d -p 80:80 -p 443:443 \
-e TZ=Asia/Seoul \
--name nginx -u root nginx:latest
```

```
docker run -d -p 80:80 -p 443:443 \
-e TZ=Asia/Seoul \
--name nginx -u root nginx:1.25.3
```

- OS 재시작 후 자동 실행 설정
  - docker-nginx.service 등록

```
sudo vim /etc/systemd/system/docker-nginx.service
```

```
[Unit]
Description=docker-nginx
Wants=docker.service
After=docker.service

[Service]
RemainAfterExit=yes
ExecStart=/usr/bin/docker start nginx
ExecStop=/usr/bin/docker stop nginx

[Install]
WantedBy=multi-user.target
```

- docker 서비스 활성화

```
sudo systemctl enable docker
```

- docker 서비스 시작

```
sudo systemctl start docker
```

- docker-nginx 서비스 활성화

```
sudo systemctl enable docker-nginx.service
```

- docker-nginx 서비스 시작

```
sudo systemctl start docker-nginx.service
```

- Certbot으로 SSL 인증서 발급 및 https 적용
  - nginx 컨테이너 내부 접속

```
docker exec -it nginx /bin/bash
```

  - 컨테이너에 certbot 패키지 설치

```
apt-get update
```

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

  - nginx 설정 파일 수정

```
vim /etc/nginx/conf.d/default.conf
```

```
server {
    listen       80;

    # 수정
    server_name  자신의 도메인;

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

    # 이 부분 추가
    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

- `bash: vim: command not found` 의 경우, 패키지 매니저를 통해 설치

```
apt-get install vim
```

  - Certbot 인증서 발급

```
certbot --nginx -d 도메인
```

도메인을 넣어 certbot을 실행시키면 다음과 같은 기본 선택 사항이 나옴.

1. 관리자 이메일 주소(필수)
2. 이메일 서버 등록(필수, 최초 1회)
3. 캠페인 소식 수신 여부 (Yes, No)

이후에는 Nginx 설정파일에 https 리다이렉트 구분을 자동으로 추가할 것인지 물어봄.

인증이 성공적으로 마무리되면 암호키는 기본적으로 `/etc/letsencrypt/live/도메인/` 아래에 생성됨

- nginx 서버 업로드 크기(사이즈) 설정
  - nginx 컨테이너 내부 접속

```
docker exec -it nginx /bin/bash
```

- nginx 설정 파일 수정

```
vim /etc/nginx/nginx.conf
```

```
http {
  # Set client upload size - 100Mbyte
  client_max_body_size 100M;
  ...
  ..
  .
}
```

  - `bash: vim: command not found` 의 경우, 패키지 매니저를 통해 설치

```
apt-get install vim
```

- nginx 재시작 필요

- Reverse Proxy(리버스 프록시) 설정
  - nginx 컨테이너 내부 접속

```
docker exec -it nginx /bin/bash
```

  - nginx 설정 파일 수정

```
vim /etc/nginx/conf.d/default.conf
```

```
server {
    server_name 도메인;

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

    location /api {
        proxy_pass http://도메인:8080;
        proxy_redirect off;
        charset utf-8;

        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 X-NginX-Proxy true;
    }

    location /signal {
        proxy_pass http://도메인:8080;
        proxy_http_version 1.1;
```

```
        proxy_set_header Upgrade $http_upgrade; proxy_set_header Connection
"Upgrade" ;
        proxy_ set_header Host $host;
    }

    location /chatroom {
        proxy_pass http://도메인:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade; proxy_set_header Connection
"Upgrade";
        proxy_set_header Host $host;
    }

    location / {
        # docker inspect haryeom-fe <check gateway>
        # proxy_pass http://172.17.0.1:3000;
        proxy_pass http://도메인:3000;
        # root
        /usr/share/nginx/html;
        # index index.html index.htm;
        proxy_redirect off;
        charset utf-8;

        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 X-NginX-Proxy true;
    }

    #error_page 404      /404. html;
    # redirect server error pages to the static page /50x. html
    #
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \. php$ {
    #   proxy_pass http://127.0.0.1;
    #}

    # pass the PHP scripts to FastGI server listening on 127.0.0.1:9000
    #
    #location ~ \. php$ {
    #   root html;
```

```
    #   fastcgi_pass 127.0.0.1:9000;
    #   fastcgi_index index.php;
    #   fastegi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;
    #   include fastegi_params;
    #}

    # deny access to htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ A.ht {
    #   deny all;
    #}

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/도메인/fullchain.pem; # managed b
y Certbot
    ssl_certificate_key /etc/letsencrypt/live/도메인/privkey.pem; # managed
by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = 도메인) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;
    server_name 도메인;
    return 404; # managed by Certbot
}
```

- `bash: vim: command not found` 의 경우, 패키지 매니저를 통해 설치

```
apt-get install vim
```

  - nginx 재시작 필요

MariaDB

- 버전 정보
  - 11.2.2
- 도커 이미지 받기(docker image pull)

```
sudo docker pull mariadb:latest
```

```
sudo docker pull mariadb:11.2.2
```

- 도커 컨테이너 실행(docker run)

```
docker run -d -p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=비밀번호 \
-e TZ=Asia/Seoul \
-v /var/lib/mysql:/var/lib/mysql \
--name mariadb mariadb:latest
```

```
docker run -d -p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=비밀번호 \
-e TZ=Asia/Seoul \
-v /var/lib/mysql:/var/lib/mysql \
--name mariadb mariadb:11.2.2
```

- OS 재시작 후 자동 실행 설정
  - docker-mariadb.service 등록

```
sudo vim /etc/systemd/system/docker-mariadb.service
```

```
[Unit]
Description=docker-mariadb
Wants=docker.service
After=docker.service

[Service]
RemainAfterExit=yes
ExecStart=/usr/bin/docker start mariadb
ExecStop=/usr/bin/docker stop mariadb

[Install]
WantedBy=multi-user.target
```

  - docker 서비스 활성화

```
sudo systemctl enable docker
```

  - docker 서비스 시작

```
sudo systemctl start docker
```

  - docker-mariadb 서비스 활성화

```
sudo systemctl enable docker-mariadb.service
```

  - docker-mariadb 서비스 시작

```
sudo systemctl start docker-mariadb.service
```

- character-set 설정
  - mariadb 컨테이너 내부 접속

```
docker exec -it mariadb /bin/bash
```

  - mariadb 설정 파일 수정

```
vim /etc/mysql/my.cnf
```

```
[mysql]
default-character-set=utf8mb4

[mysqld]
collation-server = utf8mb4_general_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
```

    - `bash: vim: command not found` 의 경우, 패키지 매니저를 통해 설치

```
apt-get update && apt-get install vim
```

- 사용자 계정 관련
  - DBMS 사용 / 도커 컨테이너 내부에 접속(`docker exec -it mariadb /bin/bash`)하여 `mariadb -u root -p` 사용
  - 사용자 계정 생성

```
CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';
```

  - 사용자 계정 삭제

```
DROP USER 계정이름@아이피주소;
```

  - 사용자에 데이터베이스 사용 권한 부여

```
GRANT ALL PRIVILEGES ON 데이터베이스이름.* TO '계정이름'@'%';
```

  - 변경 내용 반영

```
FLUSH PRIVILEGES;
```

- DDL

```
CREATE DATABASE `haryeom`; /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8
mb4_unicode_ci */;

-- haryeom.chat_message definition

CREATE TABLE `chat_message` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `chat_room_id` bigint(20) NOT NULL,
  `member_id` bigint(20) NOT NULL,
  `message_content` varchar(255) DEFAULT NULL,
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_chat_room_id_in_chat_message` (`chat_room_id`),
  KEY `FK_member_id_in_chat_message` (`member_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.`member` definition

CREATE TABLE `member` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `role` varchar(20) DEFAULT NULL COMMENT 'GUEST, STUDENT, TEACHER',
  `status` varchar(50) DEFAULT NULL COMMENT 'ACTIVATED, DELETED',
  `oauth_id` varchar(255) DEFAULT NULL COMMENT 'OAuth 계정 ID(Not Null)',
  `profile_url` varchar(2048) DEFAULT NULL COMMENT '프로필 사진 URL',
  `name` varchar(30) DEFAULT NULL COMMENT '이름',
  `phone` varchar(20) DEFAULT NULL COMMENT '전화번호(010XXXXXXXX)',
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.subject definition

CREATE TABLE `subject` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL COMMENT '과목명',
```

```
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.teacher definition

CREATE TABLE `teacher` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `member_id` bigint(20) NOT NULL COMMENT '멤버 테이블 FK',
  `profile_status` tinyint(1) DEFAULT 0 COMMENT '공개 1, 비공개 0',
  `college` varchar(50) DEFAULT NULL COMMENT '대학교명',
  `major` varchar(50) DEFAULT NULL COMMENT '대학 전공명',
  `college_email` varchar(100) DEFAULT NULL COMMENT '대학 인증 이메일',
  `gender` varchar(10) DEFAULT NULL COMMENT 'MALE, FEMALE, NONE',
  `salary` int(11) DEFAULT NULL COMMENT 'NULL OK, (단위 : 만원)',
  `career` int(11) DEFAULT NULL COMMENT 'NULL OK, (단위 : 년)',
  `introduce` varchar(1500) DEFAULT NULL COMMENT '자기 소개',
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_member_id_in_teacher` (`member_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.tutoring definition

CREATE TABLE `tutoring` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `chat_room_id` bigint(20) NOT NULL,
  `subject_id` bigint(20) DEFAULT NULL,
  `hourly_rate` int(11) DEFAULT NULL,
  `status` varchar(20) DEFAULT 'IN_PROGRESS' COMMENT 'IN_PROGRESS, CLOSED',
  `student_member_id` bigint(20) NOT NULL,
  `teacher_member_id` bigint(20) NOT NULL,
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_chat_room_id_in_tutoring` (`chat_room_id`),
  KEY `FK_student_member_id_in_tutoring` (`student_member_id`),
  KEY `FK_subject_id_in_tutoring` (`subject_id`),
  KEY `FK_teacher_member_id_in_tutoring` (`teacher_member_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;
```

```
-- haryeom.chat_room definition

CREATE TABLE `chat_room` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `student_member_id` bigint(20) NOT NULL,
  `teacher_member_id` bigint(20) NOT NULL,
  `is_deleted` tinyint(1) DEFAULT 0,
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_student_member_id_in_chat_room` (`student_member_id`),
  KEY `FK_teacher_member_id_in_chat_room` (`teacher_member_id`),
  CONSTRAINT `FK_student_member_id_in_chat_room` FOREIGN KEY (`student_member_
id`) REFERENCES `member` (`id`),
  CONSTRAINT `FK_teacher_member_id_in_chat_room` FOREIGN KEY (`teacher_member_
id`) REFERENCES `member` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.chat_room_state definition

CREATE TABLE `chat_room_state` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `chat_room_id` bigint(20) DEFAULT NULL,
  `member_id` bigint(20) DEFAULT NULL,
  `last_read_message_id` varchar(24) DEFAULT '000000000000000000000000',
  `is_deleted` tinyint(1) DEFAULT 0,
  PRIMARY KEY (`id`),
  KEY `chat_room_id` (`chat_room_id`),
  KEY `member_id` (`member_id`),
  CONSTRAINT `chat_room_state_ibfk_1` FOREIGN KEY (`chat_room_id`) REFERENCES
`chat_room` (`id`),
  CONSTRAINT `chat_room_state_ibfk_2` FOREIGN KEY (`member_id`) REFERENCES `me
mber` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unico
de_ci;


-- haryeom.student definition

CREATE TABLE `student` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `member_id` bigint(20) NOT NULL COMMENT '멤버 테이블 FK',
  `grade` varchar(30) DEFAULT NULL COMMENT '학생 학년',
  `school` varchar(50) DEFAULT NULL COMMENT '학생 학교',
```

```
  PRIMARY KEY (`id`),
  KEY `FK_member_id_in_homework` (`member_id`),
  CONSTRAINT `FK_member_id_in_homework` FOREIGN KEY (`member_id`) REFERENCES `
member` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.teacher_subject definition

CREATE TABLE `teacher_subject` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `subject_id` bigint(20) NOT NULL,
  `teacher_id` bigint(20) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_subject_id_in_teacher_subject` (`subject_id`),
  KEY `FK_teacher_id_in_teacher_subject` (`teacher_id`),
  CONSTRAINT `FK_subject_id_in_teacher_subject` FOREIGN KEY (`subject_id`) REF
ERENCES `subject` (`id`),
  CONSTRAINT `FK_teacher_id_in_teacher_subject` FOREIGN KEY (`teacher_id`) REF
ERENCES `teacher` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.textbook definition

CREATE TABLE `textbook` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `teacher_member_id` bigint(20) NOT NULL,
  `subject_id` bigint(20) NOT NULL,
  `textbook_name` varchar(20) DEFAULT NULL,
  `textbook_url` varchar(2048) DEFAULT NULL,
  `first_page_cover` tinyint(1) DEFAULT NULL,
  `total_page` int(11) DEFAULT NULL,
  `cover_img` varchar(2048) DEFAULT NULL,
  `is_deleted` tinyint(1) DEFAULT 0,
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_teacher_member_id_in_textbook` (`teacher_member_id`),
  CONSTRAINT `FK_teacher_member_id_in_textbook` FOREIGN KEY (`teacher_member_i
d`) REFERENCES `member` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;
```

```
-- haryeom.tutoring_schedule definition

CREATE TABLE `tutoring_schedule` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '과외일정 테이블 PK',
  `tutoring_id` bigint(20) NOT NULL COMMENT '과외 테이블 FK',
  `schedule_date` date DEFAULT NULL COMMENT '과외 날짜',
  `start_time` time DEFAULT NULL COMMENT '과외 시작 시간',
  `duration` int(11) DEFAULT NULL COMMENT '과외 진행 시간(단위: 분)',
  `title` varchar(64) DEFAULT NULL COMMENT '커리큘럼명',
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  `created_at` datetime DEFAULT current_timestamp(),
  PRIMARY KEY (`id`),
  KEY `FK_tutoring_id_in_tutoring_schedule` (`tutoring_id`),
  CONSTRAINT `FK_tutoring_id_in_tutoring_schedule` FOREIGN KEY (`tutoring_id`)
REFERENCES `tutoring` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.video definition

CREATE TABLE `video` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `tutoring_schedule_id` bigint(20) NOT NULL,
  `video_url` varchar(2048) DEFAULT NULL,
  `start_time` time DEFAULT NULL,
  `end_time` time DEFAULT NULL,
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  PRIMARY KEY (`id`),
  KEY `FK_tutoring_schedule_id_in_video` (`tutoring_schedule_id`),
  CONSTRAINT `FK_tutoring_schedule_id_in_video` FOREIGN KEY (`tutoring_schedul
e_id`) REFERENCES `tutoring_schedule` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.video_room definition

CREATE TABLE `video_room` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `tutoring_schedule_id` bigint(20) NOT NULL COMMENT '과외일정 테이블 FK',
  `room_code` varchar(100) DEFAULT NULL COMMENT '화상과외방 입장 코드',
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
```

```
  PRIMARY KEY (`id`),
  KEY `FK_tutoring_schedule_id_in_video_room` (`tutoring_schedule_id`),
  CONSTRAINT `FK_tutoring_schedule_id_in_video_room` FOREIGN KEY (`tutoring_sc
hedule_id`) REFERENCES `tutoring_schedule` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.video_timestamp definition

CREATE TABLE `video_timestamp` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `video_id` bigint(20) NOT NULL,
  `stamp_time` time DEFAULT NULL,
  `content` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_video_id_in_timestamp` (`video_id`),
  CONSTRAINT `FK_video_id_in_timestamp` FOREIGN KEY (`video_id`) REFERENCES `v
ideo` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.`assignment` definition

CREATE TABLE `assignment` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `tutoring_id` bigint(20) NOT NULL,
  `textbook_id` bigint(20) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_textbook_id_in_assignment` (`textbook_id`),
  KEY `FK_tutoring_id_in_assignment` (`tutoring_id`),
  CONSTRAINT `FK_textbook_id_in_assignment` FOREIGN KEY (`textbook_id`) REFERE
NCES `textbook` (`id`),
  CONSTRAINT `FK_tutoring_id_in_assignment` FOREIGN KEY (`tutoring_id`) REFERE
NCES `tutoring` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.homework definition

CREATE TABLE `homework` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `textbook_id` bigint(20) NOT NULL,
  `tutoring_id` bigint(20) NOT NULL,
  `deadline` date DEFAULT NULL,
  `start_page` int(11) DEFAULT NULL,
```

```
  `end_page` int(11) DEFAULT NULL,
  `status` varchar(20) DEFAULT 'UNCONFIRMED' COMMENT 'UNCONFIRMED, IN_PROGRES
S, COMPLETED',
  `created_at` datetime DEFAULT current_timestamp(),
  `updated_at` datetime DEFAULT current_timestamp() ON UPDATE current_timestam
p(),
  `is_deleted` tinyint(1) DEFAULT 0,
  `version` bigint(20) DEFAULT 0,
  PRIMARY KEY (`id`),
  KEY `FK_textbook_id_in_homework` (`textbook_id`),
  KEY `FK_tutoring_id_in_homework` (`tutoring_id`),
  CONSTRAINT `FK_textbook_id_in_homework` FOREIGN KEY (`textbook_id`) REFERENC
ES `textbook` (`id`),
  CONSTRAINT `FK_tutoring_id_in_homework` FOREIGN KEY (`tutoring_id`) REFERENC
ES `tutoring` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;


-- haryeom.drawing definition

CREATE TABLE `drawing` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `homework_id` bigint(20) NOT NULL,
  `page` int(11) DEFAULT NULL,
  `homework_drawing_url` varchar(2048) DEFAULT NULL,
  `review_drawing_url` varchar(2048) DEFAULT NULL,
  `teacher_drawing_url` varchar(2048) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_homework_id_in_drawing` (`homework_id`),
  CONSTRAINT `FK_homework_id_in_drawing` FOREIGN KEY (`homework_id`) REFERENCE
S `homework` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci;
```

MongoDB

- 버전 정보
  - 7.0.5
- 도커 이미지 받기(docker image pull)

```
sudo docker pull mongo:latest
```

```
sudo docker pull mongo:7.0.5
```

- 도커 컨테이너 실행(docker run)

```
docker run -d \
-p 8027:27017 \
-v /data/db:/data/db \
-e MONGO_INITDB_ROOT_USERNAME=adminUser \
-e MONGO_INITDB_ROOT_PASSWORD=adminPass \
--name mongodb \
mongo:latest
```

```
docker run -d \
-p 8027:27017 \
-v /data/db:/data/db \
-e MONGO_INITDB_ROOT_USERNAME=adminUser \
-e MONGO_INITDB_ROOT_PASSWORD=adminPass \
--name mongodb \
mongo:7.0.5
```

- 사용자 계정 생성 및 데이터베이스 생성
  - 컨테이너 내부 접속

    ```
    docker exec -it mongodb /bin/bash
    ```

  - mongosh 실행

    ```
    mongosh
    ```

  - 관리자 계정 생성
    - admin 데이터베이스 사용

      ```
      use admin
      ```

    - 계정 생성

      ```
      db.createUser({
          user: "",
          pwd: "",
          roles: [{"role": "root", "db": "admin"}]
      })
      ```

  - 데이터베이스 생성

    ```
    use haryeom
    ```

  - 사용자 계정 생성

```
db.createUser({
  user: "",
  pwd: "",
  roles: [{"role": "userAdmin", "db": "haryeom"}]
})
```

Redis

- 버전 정보
  - 7.2.4
- 도커 이미지 받기(docker image pull)

```
sudo docker pull redis:latest
```

```
sudo docker pull redis:7.2.4
```

- 도커 컨테이너 실행(docker run)

```
docker run -d -p 6379:6379 \
-v /home/ubuntu/redis/data:/data \
-v /home/ubuntu/redis/redis.conf:/usr/local/etc/redis/redis.conf \
--name redis redis:latest
```

```
docker run -d -p 6379:6379 \
-v /home/ubuntu/redis/data:/data \
-v /home/ubuntu/redis/redis.conf:/usr/local/etc/redis/redis.conf \
--name redis redis:7.2.4
```

- password 설정
  - redis 컨테이너 내부 접속

```
docker exec -it redis /bin/bash
```

  - redis-cli 실행

```
redis-cli
```

  - redis password 설정
    - password 설정 정보 확인

```
config get requirepass
```

    - password 설정

```
config set requirepass strongPassword
```

- password 인증

```
AUTH 'strongPassword'
```

Front-end Server (Next.js)

- 도커 이미지 받기(docker image pull)

```
sudo docker pull demise1426/haryeom-fe
```

- 도커 컨테이너 실행(docker run)

```
docker run -d -p 3000:3000 \
--name haryeom-fe demise1426/haryeom-fe
```

Back-end Server (Spring Boot)

- 도커 이미지 받기(docker image pull)

```
sudo docker pull demise1426/haryeom-be
```

- 도커 컨테이너 실행(docker run)

```
docker run -d -p 8080:8080 \
--name haryeom-be demise1426/haryeom-be
```

환경 변수

Back-end (server, Spring Boot)

application.yml (server/src/main/resources 에 위치)

```
oauth:
  kakao:
    client-id: # 카카오 소셜 로그인을 위한 클라이언트 ID (REST API KEY)
    client-secret: # 카카오 소셜 로그인을 위한 클라이언트 Secret Key
    redirect-uri: # 카카오 소셜 로그인 시 리다이렉트될 주소
    scope: profile_nickname,profile_image,account_email # 카카오 소셜 로그인 시 연
동될 정보의 범위

spring:
  datasource:
```

```
    url: # 데이터베이스 연결을 위한 URL
    username: # 데이터베이스 사용자명
    password: # 데이터베이스 암호
    driver-class-name: org.mariadb.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect # Hibernate가 사용할 데이터베
이스 방언

  data:
    redis:
      host: # Redis 호스트 주소
      port: 6379 # Redis 포트 번호
      password: # Redis 암호
    mongodb:
      uri: # MongoDB 연결 URI

  servlet:
    multipart:
      max-file-size: 100MB # 최대 파일 크기 설정
      max-request-size: 100MB # 최대 요청 크기 설정

  jwt:
    token:
      secret-key: # JWT 토큰의 서명에 사용할 비밀 키
      refresh-secret-key: # JWT 리프레시 토큰의 서명에 사용할 비밀 키

  univcert:
    api-key: # Univcert API 키

  jackson:
    serialization:
      write-dates-as-timestamps: false  # 날짜를 타임스탬프로 직렬화할지 여부를 결정
하는 설정

cloud:
  aws:
    s3:
      bucket: # AWS S3 버킷 이름
    credentials:
      access-key: # AWS 액세스 키
      secret-key: # AWS 시크릿 키
    region:
      static:  # AWS S3 정적 리소스를 위한 리전 설정
```

```
      auto: false # AWS 리전 자동 설정 여부
    stack:
      auto: false # AWS 스택 자동 설정 여부
```

Front-end (client, Next.js)

.env (client/ 에 위치)

```
# Backend API 접근 주소
NEXT_PUBLIC_API_SERVER=
# 웹소켓 연결 (채팅 서비스)
NEXT_PUBLIC_CHAT_SERVER=
# 웹소켓 연결 (화상 과외 서비스)
NEXT_PUBLIC_SIGNALING_SERVER=
# 카카오 소셜 인증을 위한 REST API KEY
NEXT_PUBLIC_REST_API_KEY=
# 카카오 소셜 인증 후 리다이렉트될 주소
NEXT_PUBLIC_REDIRECT_URI=
```

외부 API

카카오

API 키 발급 및 플랫폼 등록

- https://developers.kakao.com/ 에 접속한다.
- 로그인 후 상단의 '내 애플리케이션'을 클릭한다.
- 애플리케이션 추가하기를 통해 새로운 프로젝트를 생성한다.
- 생성한 프로젝트를 클릭하여 상세보기 페이지로 이동한다.
- REST API 키를 확인할 수 있다.
    - 재발급은 '앱 설정 - 앱 키'를 통해 할 수 있다.
- '앱 설정 - 플랫폼'을 클릭한 후 하단의 Web에서 사이트 도메인을 등록한다.

카카오 로그인 설정

- 프로젝트 상세보기 페이지의 좌측 메뉴 중 '제품 설정 - 카카오 로그인'을 클릭한다.
- 활성화 상태를 ON으로 설정한다.
- Redirect URI를 등록한다.
- 좌측 '카카오 로그인'의 하위 메뉴 '동의항목'을 클릭한다.
- 동의항목을 설정한다.

    - 필수 동의: 닉네임, 프로필 사진, 카카오계정(이메일)
- 좌측 '카카오 로그인'의 하위 메뉴 '보안'을 클릭한다.
- 코드 생성을 클릭하여 Client Secret을 발급 받는다.

Amazon S3 및 Amazon CloudFront

공통

- https://aws.amazon.com/ 에 접속하여 회원가입 및 콘솔에 로그인 한다.

Amazon S3

- S3 서비스 페이지로 이동한다. (검색 또는 https://s3.console.aws.amazon.com/s3)
- '버킷 만들기' 버튼을 클릭하여 버킷을 생성한다.
    - 일반 구성
        - AWS 리전과 버킷 이름을 설정.
    - 객체 소유권
        - 'ACL 비활성화됨' 선택.
    - 이 버킷의 퍼블릭 액세스 차단 설정
        - '새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단'만 설정하지 않음.
    - 버킷 버전 관리
        - '비활성화' 선택.
    - 기본 암호화
        - 암호화 유형
            - 'Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화' 선택.
        - 버킷 키
            - '활성화' 선택
    - 고급 설정
        - 객체 잠금
            - '비활성화' 선택
- S3 버킷 목록 확인 페이지로 이동한다. (https://s3.console.aws.amazon.com/s3/buckets)
- 생성한 버킷을 클릭하여 상세(관리) 페이지로 이동한다.
- 버킷 이름 하단의 탭 중 '권한' 탭을 클릭한다.
- 버킷 정책을 편집한다.(CloudFront 설정 후 작업)
    - 해당 정책은 CloudtFront를 통해서만 해당 버킷에 접근할 수 있다는 것을 의미한다.
    - 이 설정은 CloudFront의 원본 액세스 제어 설정을 해두어야 적용 가능하다.
    - 하단 CloudFront 설정을 통해 S3 정책을 확인할 수 있다.
- S3 Access Key & Secret Key 발급

- IAM 서비스 페이지로 이동한다. (검색 또는 https://us-east-1.console.aws.amazon.com/iam)
- 좌측 메뉴 '액세스 관리 - 사용자'에서 사용자 생성을 진행한다.
  - 사용자 이름 입력 후 다음
  - 권한 옵션은 '직접 정책 연결' 선택하여 권한 정책에서 'AmazonS3FullAccess' 체크 후 다음
  - '사용자 생성' 버튼 클릭하여 생성
  - 사용자 목록에서 생성한 사용자 클릭하여 상세(관리) 페이지로 이동
  - '보안 자격 증명' 탭에서 '액세스 키 만들기' 클릭
    - 'Command Line Interface(CLI)' 선택하여 다음
    - 필요 시 태그를 설정하고 액세스 키 만들기
    - Access Key와 Secret Key를 확인 가능.
      - csv 파일을 다운받아 관리.

Amazon CloudFront

- CloudFront 서비스 페이지로 이동한다. (검색 또는 https://us-east-1.console.aws.amazon.com/cloudfront/v4/home)
- 'CloudFront 배포 생성' 버튼을 클릭하여 배포를 생성한다.
  - Origin domain
    - 생성해둔 S3 Bucket을 선택.
  - 원본 액세스
    - '원본 액세스 제어 설정'을 선택.
    - 'Create new OAC' 버튼을 클릭하여 OAC를 생성한다.
  - 웹 애플리케이션 방화벽(WAF)
    - '보안 보호 비활성화' 선택
  - 나머지 사항
    - 기본값 적용
- 배포를 생성하여 이동되는 페이지에서 S3 버킷 정책을 업데이트해야 한다는 알림을 확인한다.
  - '정책 복사'를 클릭하여 정책을 복사한다.
  - S3 버킷 권한으로 이동하여 해당 정책으로 업데이트한다.

UNIVCERT

API 키 발급

- https://univcert.com/ 에 접속한다.
- 우측 상단 API 키 발급 버튼을 누른다.
- 회원 가입 및 로그인을 하면 API 키를 확인할 수 있다.

CI/CD

Dockerfile

Back-end (server, Spring Boot)

```
# 기본 이미지로 OpenJDK 11 사용
FROM adoptopenjdk:11-jdk

# 애플리케이션 JAR 파일 복사
COPY ./build/libs/haryeom-0.0.1-SNAPSHOT.jar /app.jar

# 실행할 명령어 설정
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Front-end (client, Next.js)

```
FROM node:20-alpine AS base

# Install dependencies only when needed
FROM base AS deps
# Check https://github.com/nodejs/docker-node/tree/b4117f9333da4138b03a546ec926ef50a31506c3#nodealpine to understand why libc6-compat might be needed.
RUN apk add --no-cache libc6-compat
WORKDIR /app

# Install dependencies based on the preferred package manager
# COPY package.json yarn.lock* package-lock.json* pnpm-lock.yaml* ./
# RUN \
#   if [ -f yarn.lock ]; then yarn --frozen-lockfile; \
#   elif [ -f package-lock.json ]; then npm ci; \
#   elif [ -f pnpm-lock.yaml ]; then yarn global add pnpm && pnpm i --frozen-lockfile; \
#   else echo "Lockfile not found." && exit 1; \
#   fi
COPY package.json package-lock.json* ./
RUN npm config set registry http://registry.npmjs.org/
RUN npm ci


# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY .env .env
COPY . .
```

```
# Next.js collects completely anonymous telemetry data about general usage.
# Learn more here: https://nextjs.org/telemetry
# Uncomment the following line in case you want to disable telemetry during the b
uild.
# ENV NEXT_TELEMETRY_DISABLED 1

# RUN yarn build

# If using npm comment out above and use below instead
RUN npm run build

# Production image, copy all the files and run next
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production
# Uncomment the following line in case you want to disable telemetry during runti
me.
# ENV NEXT_TELEMETRY_DISABLED 1

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public

# Set the correct permission for prerender cache
RUN mkdir .next
RUN chown nextjs:nodejs .next

# Automatically leverage output traces to reduce image size
# https://nextjs.org/docs/advanced-features/output-file-tracing
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000
# set hostname to localhost
# ENV HOSTNAME "0.0.0.0"

# server.js is created by next build from the standalone output
# https://nextjs.org/docs/pages/api-reference/next-config-js/output
CMD ["node", "server.js"]
```

Docker Compose

Back-end (server, Spring Boot)

```
version: '3.8'

services:
  server:
    build:
      context: ../server
      dockerfile: Dockerfile
    image: demise1426/haryeom-be
    ports:
      - "8080:8080"
    container_name: haryeom-be
```

Front-end (client, Next.js)

```
version: '3.8'

services:
  client:
    build:
      context: ../client
      dockerfile: Dockerfile
    image: demise1426/haryeom-fe
    ports:
      - "3000:3000"
    container_name: haryeom-fe
    environment:
      - NODE_ENV=production
    restart: always
```

Jenkins Pipeline

Back-end (server, Spring Boot)

```
pipeline {
    agent any
    tools {
        gradle "gradle"
    }

    environment {
        serverPath = 'server/'
        gitBranch = 'develop'
        gitCredential =
        gitUrl =
        dockerCredential =
```

```
        latestImage =
        MATTERMOST_ENDPOINT = credentials('mattermost_endpoint')
        MATTERMOST_CHANNEL = credentials('mattermost_channel')
    }

    stages {
//        stage('Checkout') {
//            steps {
//                // Checkout 소스 코드
//                checkout scm
//            }
//        }
        stage('Check Changes') {
            steps {
                script {
                    // GitLab webhook payload contains information about the chan
ges
//                      def changes = currentBuild.rawBuild.changeSets.collect { i
t.items }.flatten()
                    def changes = currentBuild.rawBuild.changeSets.collect { chan
geLogSet ->
                        changeLogSet.collect { changeSet ->
                            changeSet.getAffectedFiles()
                        }
                    }.flatten()

                    // Check if changes include server directory
                     def serverChanged = changes.any { it.path.startsWith(serverP
ath) }

//                      def serverChanged = changes.any {
//                          if (it.path.startsWith('server/')) {
//                              echo "Change detected in: ${it.path}"
//                              true
//                          } else {
//                              echo "else Change detected in: ${it.path}"
//                              false
//                          }
//                      }

                    if (serverChanged) {
                        echo 'Changes detected in server directory. Running the p
ipeline.'
                    } else {
                        echo 'No changes in server directory. Skipping the pipeli
ne.'
                        currentBuild.result = 'ABORTED'
                        error 'No changes in server directory. Skipping the pipel
```

```
ine.'
                    }
                }
            }
        }
        stage('Git Clone') {
            steps {
                git branch: gitBranch,
                        credentialsId: gitCredential,
                        url: gitUrl
            }
        }
        stage('Jar Build') {
            steps {
                dir('server') {
                                    withCredentials([file(credentialsId: 'Spring
BootEnv', variable: 'SpringBootEnv')]) {
                        sh "cp ${SpringBootEnv} ./src/main/resources/application.
yml"
                    }
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                }
            }
        }
        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('deploy') {
                    script {
                        docker.withRegistry('', dockerCredential) {
                            // Use the credentials for Docker Hub login
                            // Build and push Docker image using docker-compose
                            sh "docker-compose -f docker-compose-server.yml buil
d"
                            sh "docker-compose -f docker-compose-server.yml push"
                        }
                    }
                }
            }
        }
        stage('Service Stop & Service Remove') {
            steps {
                dir('deploy') {
                    sh 'docker stop haryeom-be'
                    sh 'docker rm haryeom-be'
                    sh "docker rmi $latestImage"
                }
            }
```

```
        }
        stage('DockerHub Pull & Service Start') {
            steps {
                dir('deploy') {
                    sh 'docker-compose -f docker-compose-server.yml up -d'
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdo
ut: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnSt
dout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "Server Build Success: ${env.JOB_NAME} #${env.BUILD_
NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdo
ut: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnSt
dout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "Server Build Failure: ${env.JOB_NAME} #${env.BUILD_
NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
}
```

Front-end (client, Next.js)

```
pipeline {
    agent any
```

```
    tools {
        nodejs "nodejs20.11.0"
    }

    environment {
        clientPath = 'client/'
        gitBranch = 'develop'
        gitCredential =
        gitUrl =
        dockerCredential =
        latestImage =
        NEXT_PUBLIC_API_SERVER = credentials('NEXT_PUBLIC_API_SERVER')
        NEXT_PUBLIC_REST_API_KEY = credentials('NEXT_PUBLIC_REST_API_KEY')
        NEXT_PUBLIC_REDIRECT_URI = credentials('NEXT_PUBLIC_REDIRECT_URI')
        NEXT_PUBLIC_CHAT_SERVER = credentials('NEXT_PUBLIC_CHAT_SERVER')
        NEXT_PUBLIC_SIGNALING_SERVER = credentials('NEXT_PUBLIC_SIGNALING_SERVE
R')
        MATTERMOST_ENDPOINT = credentials('mattermost_endpoint')
        MATTERMOST_CHANNEL = credentials('mattermost_channel')
    }

    stages {
        stage('Check Changes') {
            steps {
                script {
                    // GitLab webhook payload contains information about the chan
ges
                    def changes = currentBuild.rawBuild.changeSets.collect { chan
geLogSet ->
                        changeLogSet.collect { changeSet ->
                            changeSet.getAffectedFiles()
                        }
                    }.flatten()

                    // Check if changes include client directory
                    def clientChanged = changes.any { it.path.startsWith(clientPa
th) }

                    if (clientChanged) {
                        echo 'Changes detected in client directory. Running the p
ipeline.'
                    } else {
                        echo 'No changes in client directory. Skipping the pipeli
ne.'
                        currentBuild.result = 'ABORTED'
                        error 'No changes in client directory. Skipping the pipel
ine.'
                    }
```

```
                }
            }
        }
        stage('Git Clone') {
            steps {
                git branch: gitBranch,
                        credentialsId: gitCredential,
                        url: gitUrl
            }
        }
        stage('Node Build') {
            steps {
                dir('client') {
                    sh "echo 'NEXT_PUBLIC_API_SERVER=${NEXT_PUBLIC_API_SERVER}' >
.env"
                    sh "echo 'NEXT_PUBLIC_REST_API_KEY=${NEXT_PUBLIC_REST_API_KE
Y}' >> .env"
                    sh "echo 'NEXT_PUBLIC_REDIRECT_URI=${NEXT_PUBLIC_REDIRECT_UR
I}' >> .env"
                    sh "echo 'NEXT_PUBLIC_CHAT_SERVER=${NEXT_PUBLIC_CHAT_SERVER}'
>> .env"
                    sh "echo 'NEXT_PUBLIC_SIGNALING_SERVER=${NEXT_PUBLIC_SIGNALIN
G_SERVER}' >> .env"
                    sh 'npm install'
                    sh 'npm run build'
                }
            }
        }
        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('deploy') {
                    script {
                        docker.withRegistry('', dockerCredential) {
                            // Build and push Docker image using docker-compose
                            sh "docker-compose -f docker-compose-client.yml buil
d"
                            sh "docker-compose -f docker-compose-client.yml push"
                        }
                    }
                }
            }
        }
        stage('Service Stop & Service Remove') {
            steps {
                dir('deploy') {
                    sh 'docker stop haryeom-fe'
                    sh 'docker rm haryeom-fe'
                    sh "docker rmi $latestImage"
```

```
                }
            }
        }
        stage('DockerHub Pull & Service Start') {
            steps {
                dir('deploy') {
                    sh 'docker-compose -f docker-compose-client.yml up -d'
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdo
ut: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnSt
dout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "Client Build Success: ${env.JOB_NAME} #${env.BUILD_
NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdo
ut: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnSt
dout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "Client Build Failure: ${env.JOB_NAME} #${env.BUILD_
NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
}
```