

**A308 : HelloWorld**

삼성 청년 SW아카데미 서울캠퍼스 8기

자율 프로젝트

23.04.10 ~ 23.05.19

**포팅 매뉴얼**

담당 컨설턴트 : 서성수

조현동(팀장), 배정현, 안영훈, 유송희, 변유정, 서주광

**목차**

1. 프로젝트 기술 스택…………………………………………………………….3
2. 서버 세팅……………..……………………………………………………………..5
3. 빌드 상세내용 …………….……………………………………………………15
4. 외부 서비스……………………………………………………………………….26
5. **프로젝트 기술 스택**
6. 이슈관리: Jira
7. 형상관리: Git
8. 커뮤니케이션: Mattermost, Webex, notion
9. 개발환경
   * OS: Window 10
   * IDE
     + IntelliJ
     + VSCode
     + Figma
   * Database
     + DBMS: Mariadb 10.11.2
     + NoSQL: Mongodb 4.4
     + Cache: Redis 7.0.11
     + SearchEngine: ElasticSearch 7.10.1
   * Server: AWS EC2
     + OS: Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86\_64) 2
   * File Server: AWS S3
   * CI/CD: Jenkins, Docker, Nginx
10. 상세 기술
    * Frontend
      + Vue.js 3
      + npm 9.6.4
      + Node 18.12.1
      + JavaScript
      + CSS
    * Backend
      + JDK: 11
      + Spring Boot: 2.7.10
      + Gradle
      + Spring Security
      + Spring Data JPA
      + Spring Data Elasticsearch
      + Spring Cloud AWS: 2.2.6.RELEASE
      + Spring Boot Starter Oauth2 Client
      + Spring Boot Starter Validation
      + Springfox Swagger UI: 3.0.0
      + Json Web Token(jwt): 0.11.2
      + Spring Boot Starter Data Redis
      + Spring Kafka
      + Spring Boot Starter Mail
      + Nurigo SDK: 4.2.7 (SMS 인증)
      + Lombok
      + Spring Boot Devtools
      + MariaDB Java Client
      + Spring Boot Starter Test
      + Spring Security Test
      + Spring Data Redis
      + Jedis (Redis client library)
      + Spring Boot Starter AOP
    * Server
      + AWS EC2
      + AWS S3
      + Ubuntu 20.04 LTS
      + Docker
      + Jenkins
      + CertBot
    * IDE
      + IntelliJ IDEA
      + VSCode

**2. 서버 세팅**

**2-1. 환경세팅**

아래의 명령어를 EC2 서버에 접속해서 순차적으로 실행한다.

|  |
| --- |
| $ sudo apt-get update  $ sudo apt-get install -y ca-certificates \ curl \ software-properties-common \ apt-transport-https \ gnupg \ lsb-release |

|  |
| --- |
| $ sudo mkdir -p /etc/apt/keyrings  $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg |

GPG 키 및 저장소 추가

Docker engine install

|  |
| --- |
| $ sudo apt update  $ sudo apt install docker-ce docker-ce-cli containerd.io |

Docker-compose install

|  |
| --- |
| $sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -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  $ docker-compose --version |

**2-2. HTTPS 설정 (Certbot)**

(1) Ubuntu 18.04 이상에서는 Python 3 버전

|  |
| --- |
| $ apt-get update  $ sudo apt-get install certbot  $ apt-get install python3-certbot-nginx |

(2) nginx helloworld.conf

|  |
| --- |
| server {  listen 80 default\_server;  listen [::]:80 default\_server;  root /var/www/html;  server\_name k8a308.p.ssafy.io;  } |

(3) NGINX 플러그인으로 인증서를 생성

|  |
| --- |
| $ sudo certbot --nginx -d k8a308.p.ssafy.io |

(4) 인증서 완료

|  |
| --- |
| Congratulations! You have successfully enabled <https://example.com> and <https://www.example.com>  -------------------------------------------------------------------------------------  IMPORTANT NOTES:  Congratulations! Your certificate and chain have been saved at:  /etc/letsencrypt/live/example.com/fullchain.pem  Your key file has been saved at:  /etc/letsencrypt/live/example.com//privkey.pem  Your cert will expire on 2017-12-12. |

(5) nginx 시작

|  |
| --- |
| sudo nginx |

(6) 추후에 nginx 설정이 변경되면 아래와 같은 명령어들을 이용한다.

|  |
| --- |
| sudo nginx –t  sudo service nginx restart |

**2-3. 젠킨스 설정**

1. /home/ubuntu/jenkins/Dockerfile

|  |
| --- |
| FROM jenkins/jenkins:lts  USER root  # Install Docker  RUN apt-get update && \  apt-get -y install apt-transport-https \  ca-certificates \  curl \  gnupg2 \  zip \  unzip \  software-properties-common && \  . /etc/os-release && \  curl -fsSL https://download.docker.com/linux/${ID}/gpg > /tmp/dkey; apt-key add /tmp/dkey && \  add-apt-repository \  "deb [arch=amd64] https://download.docker.com/linux/${ID} \  $(lsb\_release -cs) \  stable" && \  apt-get update && \  apt-get -y install docker-ce  # Install docker-compose  RUN curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose && \  chmod +x /usr/local/bin/docker-compose && \  ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose  # Add jenkins user to docker group  RUN usermod -aG docker jenkins  USER jenkins |

1. /home/ubuntu/jenkins/docker-compose.yml

|  |
| --- |
| # docker-compose.yml  version: '3.7'  services:  jenkins:  build:  context: .  container\_name: jenkins  user: root  ports:  - 8080:8080  volumes:  - ./jenkins\_home:/var/jenkins\_home  - //var/run/docker.sock:/var/run/docker.sock |

이후, sudo docker logs jenkins명령어를 통해 jenkins의 암호를 확인한다.

Nginx로 라우팅처리한 url을 통해 jenkins에 접속한다.

Dashboard > Jenkins 관리 > Plugin Manager로 접속한 뒤 Available plugins에서 필요한 플러그인을 설치한다.

Gitlab 플러그인

* + Generic Webhook Trigger Plugin
  + GitLab
  + Gitlab API Plugin
  + GitLab Authentication plugin

Docker 플러그인

* + Docker API Plugin
  + Docker Commons Plugin
  + Docker Compose Build Step Plugin
  + Docker Pipeline
  + Docker plugin
  + docker-build-step

이 후 EC2에서 아래의 명령어를 통해 Jenkins 컨테이너의 bash로 접근한다.

|  |
| --- |
| sudo docker exec –it jenkins bash |

jenkins 내부에서 docker에 접근할 수 있도록 (도커 인 도커) 아래의 명령어를 실행한다.

|  |
| --- |
| apt-get update  apt-get install \  ca-certificates \  curl \  gnupg \  sb-release  mkdir -p /etc/apt/keyrings  curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg  echo \  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \  $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null  apt-get update  apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin |

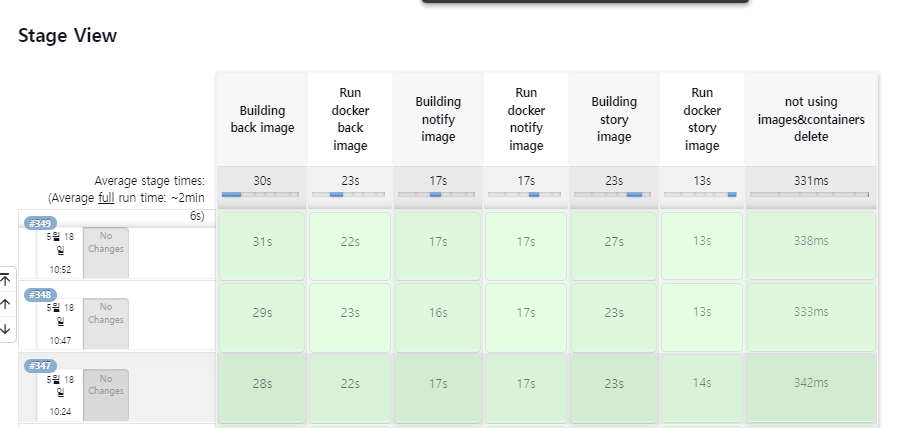
docker-compose를 사용하기 위해 아래 명령어를 추가로 입력한다.

|  |
| --- |
| sudo apt install jq  VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r)  DESTINATION=/usr/bin/docker-compose  sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION  sudo chmod 755 $DESTINATION |

도커 인 도커의 준비가 끝났다면, Jenkins 웹훅을 설정한다.

1. 젠킨스 빌드 및 배포 과정





[back\_build]

|  |
| --- |
| cd /var/jenkins\_home/workspace/back\_build  cp /var/jenkins\_home/workspace/yml/application-db.yml back/src/main/resources/  cd /var/jenkins\_home/workspace/back\_build/back  chmod +x gradlew  ./gradlew clean build  cp /var/jenkins\_home/workspace/back\_build/back/build/libs/helloworld-0.0.1-SNAPSHOT.jar /var/jenkins\_home/workspace/deploy\_test/back  cd /var/jenkins\_home/workspace/back\_build  cp /var/jenkins\_home/workspace/nyml/application-db.yml notify/src/main/resources/  cd /var/jenkins\_home/workspace/back\_build/notify  chmod +x gradlew  ./gradlew clean build  cp /var/jenkins\_home/workspace/back\_build/notify/build/libs/notify-0.0.1-SNAPSHOT.jar /var/jenkins\_home/workspace/deploy\_test/notify  cd /var/jenkins\_home/workspace/back\_build  cp /var/jenkins\_home/workspace/yml\_story/application-db.yml StoryModule/src/main/resources/  cd /var/jenkins\_home/workspace/back\_build/StoryModule  chmod +x gradlew  ./gradlew clean build  cp /var/jenkins\_home/workspace/back\_build/StoryModule/build/libs/StoryModule-0.0.1-SNAPSHOT.jar /var/jenkins\_home/workspace/deploy\_test/StoryModule |

[back\_build\_pipeline]

|  |
| --- |
| pipeline{  environment {  PATH = "/usr/local/bin:$PATH"  dockerImage = ''  }  agent any  stages{    stage('Building back image'){  steps{  script {  dockerImage = docker.build("bae3007/back-zero-downtime", "/var/jenkins\_home/workspace/deploy\_test/back")  withCredentials([usernamePassword(credentialsId: 'bae3007', usernameVariable: 'DOCKER\_HUB\_USERNAME', passwordVariable: 'DOCKER\_HUB\_PASSWORD')]) {  sh 'docker login -u $DOCKER\_HUB\_USERNAME -p $DOCKER\_HUB\_PASSWORD'  sh 'docker push bae3007/back-zero-downtime'  }  }  }  }  stage('Run docker back image') {  steps {  dir('/var/jenkins\_home/workspace/deploy\_test/back'){  sh 'chmod +x deploy.sh'  sh './deploy.sh'  }  }  }    stage('Building notify image'){  steps{  script {  dockerImage = docker.build("bae3007/notify-zero-downtime", "/var/jenkins\_home/workspace/deploy\_test/notify")  withCredentials([usernamePassword(credentialsId: 'bae3007', usernameVariable: 'DOCKER\_HUB\_USERNAME', passwordVariable: 'DOCKER\_HUB\_PASSWORD')]) {  sh 'docker login -u $DOCKER\_HUB\_USERNAME -p $DOCKER\_HUB\_PASSWORD'  sh 'docker push bae3007/notify-zero-downtime'  }  }  }  }  stage('Run docker notify image') {  steps {  dir('/var/jenkins\_home/workspace/deploy\_test/notify'){  sh 'chmod +x deploy.sh'  sh './deploy.sh'  }  }  }    stage('Building story image'){  steps{  script {  dockerImage = docker.build("bae3007/story-zero-downtime", "/var/jenkins\_home/workspace/deploy\_test/StoryModule")  withCredentials([usernamePassword(credentialsId: 'bae3007', usernameVariable: 'DOCKER\_HUB\_USERNAME', passwordVariable: 'DOCKER\_HUB\_PASSWORD')]) {  sh 'docker login -u $DOCKER\_HUB\_USERNAME -p $DOCKER\_HUB\_PASSWORD'  sh 'docker push bae3007/story-zero-downtime'  }  }  }  }  stage('Run docker story image') {  steps {  dir('/var/jenkins\_home/workspace/deploy\_test/StoryModule'){  sh 'chmod +x deploy.sh'  sh './deploy.sh'  }  }  }    stage('not using images&containers delete '){  steps{  sh "docker image prune -a -f"  }  }  }  } |

develop

|  |
| --- |
| cd /var/jenkins\_home/workspace/develop/FE  npm install  npm run build  cp -r /var/jenkins\_home/workspace/develop/FE/dist /var/jenkins\_home/workspace/deploy\_test/front |

develop\_pipeline

|  |
| --- |
| pipeline{  environment {  dockerImage = ''  }  agent any  stages{  stage('Building front image') {  steps {  script {  dockerImage = docker.build("bae3007/front-zero-downtime", "/var/jenkins\_home/workspace/deploy\_test/front")  withCredentials([usernamePassword(credentialsId: 'bae3007', usernameVariable: 'DOCKER\_HUB\_USERNAME', passwordVariable: 'DOCKER\_HUB\_PASSWORD')]) {  sh 'docker login -u $DOCKER\_HUB\_USERNAME -p $DOCKER\_HUB\_PASSWORD'  sh 'docker push bae3007/front-zero-downtime'  }  }  }  }      stage('Run docker front image') {  steps {  dir('/var/jenkins\_home/workspace/deploy\_test/front'){  sh 'chmod +x deploy.sh'  sh './deploy.sh'  }  }  }    stage('not using images&containers delete '){  steps{  sh "docker image prune -a -f"  }  }  }  } |

**2-4. DB 설정**

사용한 DB는 MongoDB, MariaDB, Redis

1. /home/ubuntu/mariadb/docker-compose.yml

|  |
| --- |
| version: '3.7'  services:  db:  container\_name: mariadb  image: mariadb  restart: always  environment:  MARIADB\_USER: root  MARIADB\_ROOT\_PASSWORD: \*\*\*\*\*\*\*  ports:  - 3306:3306  volumes:  - mariadb:/var/lib/mysql  volumes:  mariadb: |

1. /home/ubuntu/mongodb/docker-compose.yml

|  |
| --- |
| version: '3.8'  services:  mongodb:  image: mongo:4.4  container\_name: mongodb  environment:  MONGO\_INITDB\_ROOT\_USERNAME: root  MONGO\_INITDB\_ROOT\_PASSWORD: \*\*\*\*\*\*  ports:  - "27017:27017"  volumes:  - mongodb-data:/data/db  volumes:  mongodb-data: |

1. /home/ubuntu/redis/docker-compose.yml

|  |
| --- |
| [docker-compose.yml]  version: "3.8"  services:  redis:  image: redis:latest  container\_name: redis  ports:  - "6379:6379"  volumes:  - ./redis.conf:/usr/local/etc/redis/redis.conf  command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  restart: always  volumes:  redis\_data:  ---  [redis.conf]  requirepass \*\*\*\*\*\* |

**3. 빌드 상세 내용**

**3-1. 백엔드 (빌드)**

백엔드의 경우 각종 데이터베이스의 컨테이너나 현재 사용중인 환경에 민감한 정보들이 많이 존재한다. 따라서 이러한 부분들을 환경에 맞게 적절하게 변경해주어야 한다. 빌드 시점에 이를 수정하기 위해서는 서버에 application-db.yml을 따로 저장하여 프로젝트 resource/application-db.yml로 copy 한다.

\*참고로 application-db.yml에서 개인정보들과 관련된 내용들은 지웠고, 간단한 정보들만 기입하였다. 실제 서버에 올라가는 application-db.yml은 더 많은 정보들을 담고 있다.

1. /home/ubunut/Jenkins/Jenkins\_home/workspace/yml/application-db.yml

|  |
| --- |
| spring:  jpa:  hibernate:  ddl-auto: none  properties:  hibernate:  dialect: org.hibernate.dialect.MySQL8Dialect  show-sql: true  datasource:  driverClassName: org.mariadb.jdbc.Driver  url: jdbc:mariadb://k8a308.p.ssafy.io:3306/helloworld?serverTimezone=KST&characterEncoding=UTF-8&useSSL=false&useUnicode=true  username: root  password: \*\*\*\*\*  redis:  host: k8a308.p.ssafy.io  port: 6379  password: \*\*\*\*\*  elasticsearch:  host: 43.202.5.187  port: 9200  username: elastic  password: \*\*\*\*\*  logging:  level:  root: INFO  file:  name: /var/log/app/app.log |

1. /home/ubunut/Jenkins/Jenkins\_home/workspace/nyml/application-db.yml

|  |
| --- |
| spring:  data:  mongodb:  uri: mongodb://root:\*\*\*\*\*@k8a308.p.ssafy.io:27017/notify?serverTimezone=KST&characterEncoding=UTF-8&useSSL=false&useUnicode=true  server:  port: 9090  app:  auth:  token:  secret-key: "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  redis:  host: k8a308.p.ssafy.io  port: 6379  logging:  level:  root: INFO  file:  name: /var/log/app/app.log |

1. /home/ubunut/Jenkins/Jenkins\_home/workspace/yml\_story/application-db.yml

|  |
| --- |
| spring:  jpa:  hibernate:  ddl-auto: update  properties:  hibernate:  dialect: org.hibernate.dialect.MySQL8Dialect  show-sql: true  datasource:  driverClassName: org.mariadb.jdbc.Driver  url: jdbc:mariadb://k8a308.p.ssafy.io:3307/helloworld\_story?serverTimezone=UTC&characterEncoding=UTF-8&useSSL=false&useUnicode=true  username: root  password: \*\*\*\*\*\*  app:  auth:  token:  secret-key: "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  redis:  host: k8a308.p.ssafy.io  port: 6379  logging:  level:  root: INFO  file:  name: /var/log/app/app.log |

**3-2. 백엔드 & 프론트 (배포)**

**[back]**

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/back/Dockerfile

|  |
| --- |
| FROM adoptopenjdk/openjdk11  CMD ["./mvnw", "clean", "package"]  CMD ["./gradlew build"]  ARG JAR\_FILE\_PATH=./\*.jar  COPY ${JAR\_FILE\_PATH} app.jar  ENTRYPOINT ["java", "-jar", "app.jar"] |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/back/docker-compose.blue.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-blue  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8081:8080'  volumes:  - back-logs:/var/log/app  volumes:  back-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/back/docker-compose.green.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-green  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8082:8080'  volumes:  - back-logs:/var/log/app  volumes:  back-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/back/deploy.sh

|  |
| --- |
| #!/bin/bash  export DOCKER\_REGISTRY=bae3007 DOCKER\_APP\_NAME=back-zero-downtime IMAGE\_TAG=latest  EXIST\_BLUE=$(docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml ps | grep Up)  if [ -z "$EXIST\_BLUE" ]; then  echo "blueis is not exist. so make blue container"  echo "blue up"  docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml up -d  BEFORE\_COMPOSE\_COLOR="green"  AFTER\_COMPOSE\_COLOR="blue"  echo "end"  else  echo "blue is exist. so make green container"  echo "green up"  docker compose -p ${DOCKER\_APP\_NAME}-green -f docker-compose.green.yaml up -d  BEFORE\_COMPOSE\_COLOR="blue"  AFTER\_COMPOSE\_COLOR="green"  fi  sleep 20  EXIST\_AFTER=$(docker compose -p ${DOCKER\_APP\_NAME}-${AFTER\_COMPOSE\_COLOR} -f docker-compose.${AFTER\_COMPOSE\_COLOR}.yaml ps | grep Up)  if [ -n "$EXIST\_AFTER" ]; then  docker compose -p ${DOCKER\_APP\_NAME}-${BEFORE\_COMPOSE\_COLOR} -f docker-compose.${BEFORE\_COMPOSE\_COLOR}.yaml down  echo "$BEFORE\_COMPOSE\_COLOR down"  fi |

**[notify]**

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/notify/Dockerfile

|  |
| --- |
| FROM adoptopenjdk/openjdk11  CMD ["./mvnw", "clean", "package"]  CMD ["./gradlew build"]  ARG JAR\_FILE\_PATH=./\*.jar  COPY ${JAR\_FILE\_PATH} app.jar  ENTRYPOINT ["java", "-jar", "app.jar"] |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/notify/docker-compose.blue.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-blue  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8085:9090'  volumes:  - notify-logs:/var/log/app  volumes:  notify-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/notify/docker-compose.green.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-green  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8086:9090'  volumes:  - notify-logs:/var/log/app  volumes:  notify-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/notify/deploy.sh

|  |
| --- |
| #!/bin/bash  export DOCKER\_REGISTRY=bae3007 DOCKER\_APP\_NAME=notify-zero-downtime IMAGE\_TAG=latest  EXIST\_BLUE=$(docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml ps | grep Up)  if [ -z "$EXIST\_BLUE" ]; then  echo "blueis is not exist. so make blue container"  echo "blue up"  docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml up -d  BEFORE\_COMPOSE\_COLOR="green"  AFTER\_COMPOSE\_COLOR="blue"  echo "end"  else  echo "blue is exist. so make green container"  echo "green up"  docker compose -p ${DOCKER\_APP\_NAME}-green -f docker-compose.green.yaml up -d  BEFORE\_COMPOSE\_COLOR="blue"  AFTER\_COMPOSE\_COLOR="green"  fi  sleep 10  EXIST\_AFTER=$(docker compose -p ${DOCKER\_APP\_NAME}-${AFTER\_COMPOSE\_COLOR} -f docker-compose.${AFTER\_COMPOSE\_COLOR}.yaml ps | grep Up)  if [ -n "$EXIST\_AFTER" ]; then  docker compose -p ${DOCKER\_APP\_NAME}-${BEFORE\_COMPOSE\_COLOR} -f docker-compose.${BEFORE\_COMPOSE\_COLOR}.yaml down  echo "$BEFORE\_COMPOSE\_COLOR down"  fi |

**[StoryModule]**

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/StoryModule/Dockerfile

|  |
| --- |
| FROM adoptopenjdk/openjdk11  CMD ["./mvnw", "clean", "package"]  CMD ["./gradlew build"]  ARG JAR\_FILE\_PATH=./\*.jar  COPY ${JAR\_FILE\_PATH} app.jar  ENTRYPOINT ["java", "-jar", "app.jar"] |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/StoryModule/docker-compose.blue.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-blue  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8087:8080'  volumes:  - story-logs:/var/log/app  volumes:  story-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/StoryModule/docker-compose.green.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-green  environment:  - LANG=ko\_KR.UTF-8  ports:  - '8088:8080'  volumes:  - story-logs:/var/log/app  volumes:  story-logs: |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/notify/deploy.sh

|  |
| --- |
| #!/bin/bash  export DOCKER\_REGISTRY=bae3007 DOCKER\_APP\_NAME=story-zero-downtime IMAGE\_TAG=latest  EXIST\_BLUE=$(docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml ps | grep Up)  if [ -z "$EXIST\_BLUE" ]; then  echo "blueis is not exist. so make blue container"  echo "blue up"  docker compose -p ${DOCKER\_APP\_NAME}-blue -f docker-compose.blue.yaml up -d  BEFORE\_COMPOSE\_COLOR="green"  AFTER\_COMPOSE\_COLOR="blue"  echo "end"  else  echo "blue is exist. so make green container"  echo "green up"  docker compose -p ${DOCKER\_APP\_NAME}-green -f docker-compose.green.yaml up -d  BEFORE\_COMPOSE\_COLOR="blue"  AFTER\_COMPOSE\_COLOR="green"  fi  sleep 10  EXIST\_AFTER=$(docker compose -p ${DOCKER\_APP\_NAME}-${AFTER\_COMPOSE\_COLOR} -f docker-compose.${AFTER\_COMPOSE\_COLOR}.yaml ps | grep Up)  if [ -n "$EXIST\_AFTER" ]; then  docker compose -p ${DOCKER\_APP\_NAME}-${BEFORE\_COMPOSE\_COLOR} -f docker-compose.${BEFORE\_COMPOSE\_COLOR}.yaml down  echo "$BEFORE\_COMPOSE\_COLOR down"  fi |

**[front]**

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/front/Dockerfile

|  |
| --- |
| FROM nginx:latest  COPY dist/ /usr/share/nginx/html/  COPY hw.conf /etc/nginx/conf.d/hw.conf  RUN rm /etc/nginx/conf.d/default.conf  EXPOSE 80  ENTRYPOINT ["nginx", "-g", "daemon off;"] |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/front/docker-compose.blue.yaml

|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-blue  environment:  - LANG=ko\_KR.UTF-8  ports:  - '3000:80' |

/home/ubuntu/jenkins/jenkins\_home/workspace/deploy\_test/front/docker-compose.green.yaml

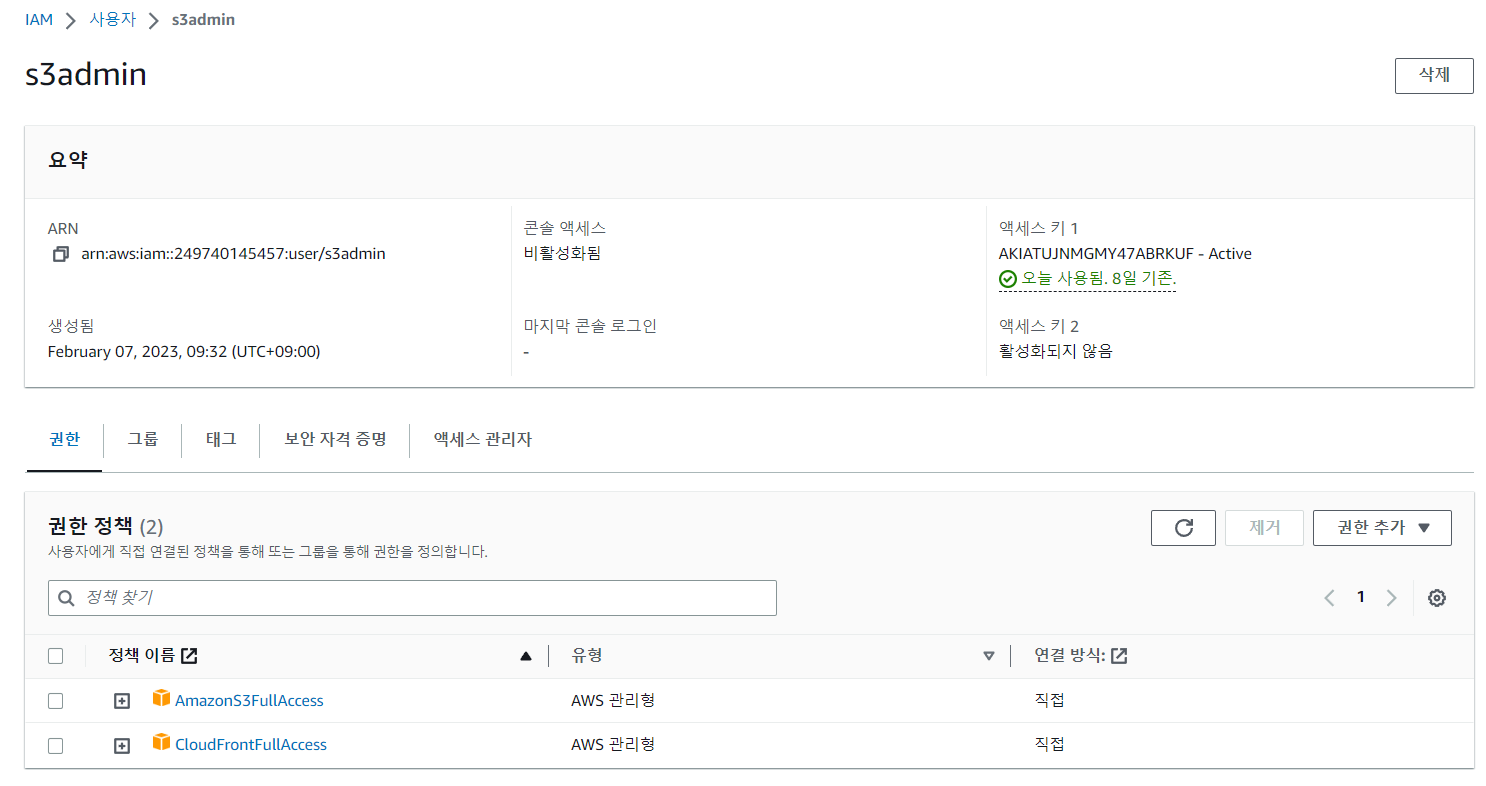
|  |
| --- |
| version: '3.7'  services:  api:  image: ${DOCKER\_REGISTRY}/${DOCKER\_APP\_NAME}:${IMAGE\_TAG}  container\_name: ${DOCKER\_APP\_NAME}-green  environment:  - LANG=ko\_KR.UTF-8  ports:  - '3001:80' |

**4. 외부 서비스**

**4-1. AWS S3**

1. 계정 설정 및 백엔드 프로젝트 연동

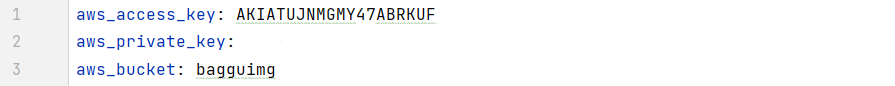
키에 대한 외부 접근의 위험을 막기 위해 S3 의 접근 권한만을 가진 IAM 사용자를 생성하여 등록한다.



S3 서버에 연결하기 위한 라이브러리 의존성을 주입한다.

|  |
| --- |
| implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.0.1.RELEASE' |

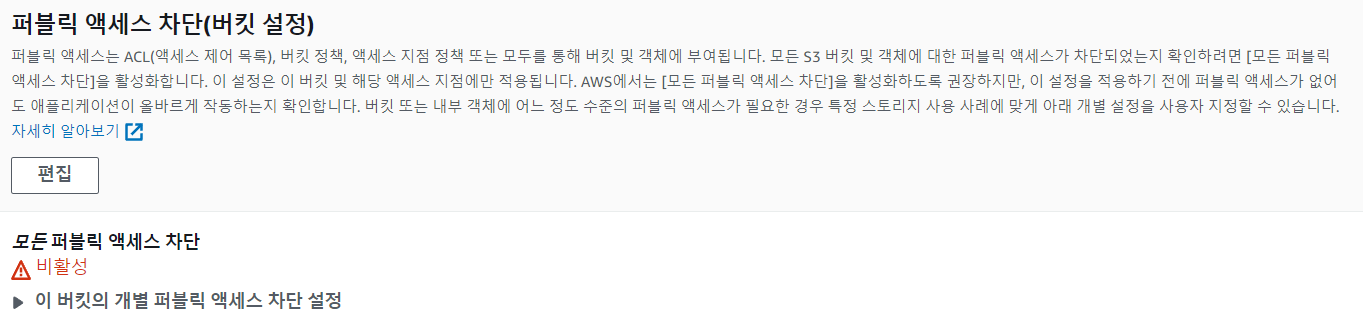
AWS Configuration을 생성하기 위한 access key와 private key는 application-SECRET.yml으로 별도 관리하며, profiles 속성을 통해 주입시킨다. (아래 이미지는 참고용 사진으로 실제 서비스와 값이 다름.)





1. S3 버킷 생성 및 설정

서버에서 버킷에 접근하여 이미지를 직접해야하므로, 퍼블릭 액세스를 허용한다.



다음은 생성한 버킷 정책이다. 앞서 생성한 S3 IAM에 대해선 모든 액션을 허용한다.

시연 시나리오는 링크로 대체합니다.

https://www.figma.com/file/CERrSSpb0XWzN7IA1uCFyi/A308-HELLO-WORLD-UI-DESIGN?type=design&node-id=0-1