# 포팅메뉴

## EC2 세팅

### 도커 및 젠킨스 설치

1. 패키지 업데이트 진행

```
sudo apt-get update
```

2. 필요 패키지 설치

```
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

3. Docker의 Official GPG key 를 등록

```
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

4. stable repository 등록

```
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
```

5. 도커 엔진 설치

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

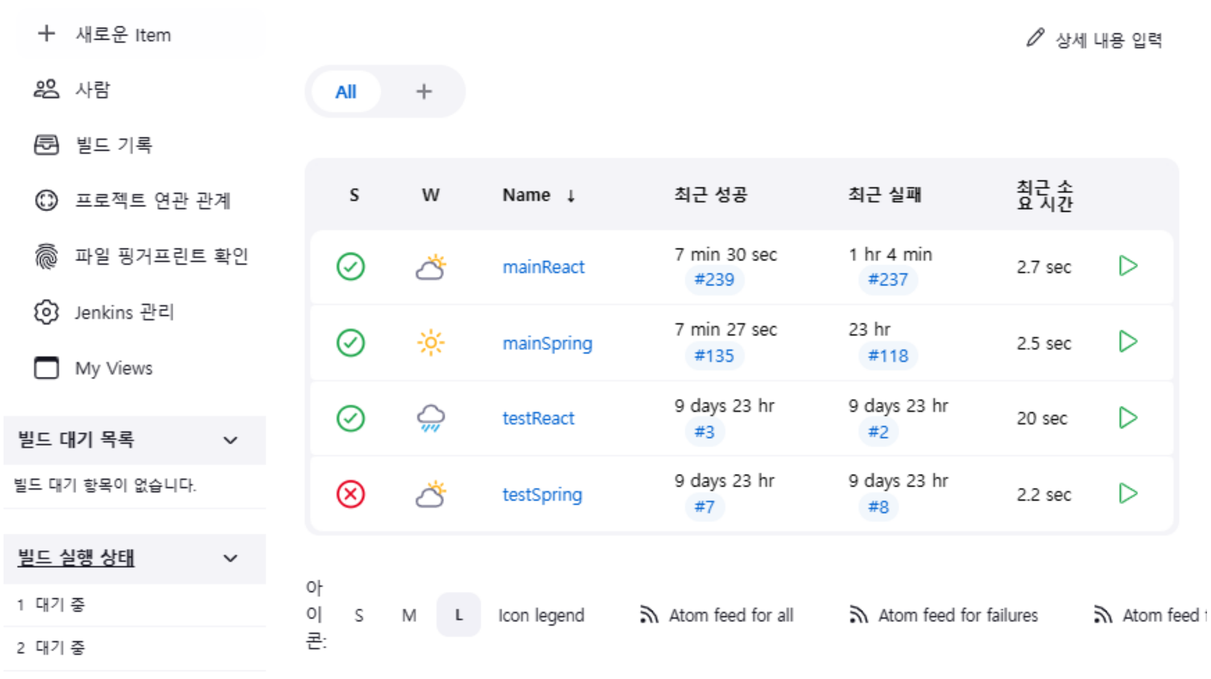
6. 젠킨스 내부 도커 설치 및 도커엔진 연결

```
# 아래 옵션을 추가함
-v /var/run/docker.sock:/var/run/docker.sock
# 추가하면 아래와 같음
sudo mkdir /home/jenkins
sudo docker run \
--name jenkins \
-d \
-p 5000:8080 \
-p 50000:50000 \
--restart=always \
-v /home/jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
jenkins/jenkins:lts

# jenkins container 접속
docker exec -it jenkins /bin/bash

# linux 버전 확인
cat /etc/issue
# -------------- OS -------------------------------
# root@DESKTOP-R4P59B3:/home/opendocs# cat /etc/issue
# Ubuntu 20.04.4 LTS \n \l
# -------------- jenkins Container OS -------------------------------
# root@DESKTOP-R4P59B3:/home/opendocs# docker exec -it jenkins /bin/bash
# root@8fc963af71bb:/# cat /etc/issue
# Debian GNU/Linux 11 \n \l
# Docker 설치
## - Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc
## - Setup Repo
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-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
## - Install Docker Engine
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

## 젠킨스 설정



빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

| S | W | Name ↓ | 최근 성공 | 최근 실패 | 최근 소요 시간 | |
|---|---|---|---|---|---|---|
| ✓ | ⛅ | mainReact | 7 min 30 sec #239 | 1 hr 4 min #237 | 2.7 sec | ▷ |
| ✓ | ☀ | mainSpring | 7 min 27 sec #135 | 23 hr #118 | 2.5 sec | ▷ |
| ✓ | 🌧 | testReact | 9 days 23 hr #3 | 9 days 23 hr #2 | 20 sec | ▷ |
| ✗ | ⛅ | testSpring | 9 days 23 hr #7 | 9 days 23 hr #8 | 2.2 sec | ▷ |

### mainReact

```
docker rm -f mainreact | true
docker rmi mainfrontend | true
docker build --tag mainfrontend ./Front/.
docker run -itd --name mainreact -p 3000:80 --restart=always mainfrontend
echo 'server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

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

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

    #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;
    }


}' > default.conf
docker cp default.conf mainreact:/etc/nginx/conf.d/default.conf
docker restart mainreact
docker rmi $(docker images -f "dangling=true" -q) | true
```

### mainSpring

```
docker rm -f mainspring | true
docker rmi mainbackend |true
docker build -t mainbackend Back/specialized/
docker run -itd --name mainspring --restart=always -p 8085:8080 mainbackend
docker rmi $(docker images -f "dangling=true" -q) | true
```

## Spring yml 설정

### application.yml

```
server:
#  host: localhost
#  port: 8085
  servlet:
    context-path: /spring

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://j8d110.p.ssafy.io:4000/specialized
    username: root
    password: root
  kafka:
    bootstrap-servers: j8d110.p.ssafy.io:9092
    producer:
      acks: all
      retries: 0
      batch-size: 16384
      linger-ms: 1
      buffer-memory: 33554432
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      value-serializer: org.apache.kafka.common.serialization.StringSerializer

  #    url: jdbc:mysql://127.0.0.1:3306/teukhwa
  #    username: root
  #    password: ssafy


  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    open-in-view: false
    hibernate:
      ddl-auto: update
  #    properties:
  #      hibernate:
  #        show_sql: true
  #        format_sql: true

  profiles:
    include: key

  redis:
    host: j8d110.p.ssafy.io
    port: 7963
  #    host: localhost
  #    port: 6379
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

cloud:
  aws:
    credentials:
      access-key: ESCb1U9YUC1iPdriv1Qc
      secret-key: 1M49n1x3q4COn0KtlZ2rKt63AQ4ermzvsCg9yk3l
    stack:
      auto: false
    region:
      static: ap-northeast-2
    s3:
      endpoint: https://kr.object.ncloudstorage.com
      bucket: d110
```

## application-key.yml

```
jwt:
  secret: VlwEyVBsYt9V7zq57TejMnVUyzblYcfPQye08f7MGVA9XkHa
```