포팅메뉴얼

프로젝트 기술 스택

가. 이슈관리 : Jira

나. 형상관리 : Gitlab

다. 커뮤니케이션 : Mattetmost, Notion, Webex

라. 개발 환경

1. OS : Windows 10
2. IDE
   가) IntelliJ IDEA Ultimate ver 212.5457.46
   나) Visual Studio Code ver 1.62.2
3. Database

   가) mariadb image **10.7.1**

   나) (view) MySQL Workbench 8.0.22
4. Server : AWS EC2 (MobaXterm)
   가) Ubuntu 20.04.3 LTS

마. 상세 사용

1. Backend
   가) Java (Open-JDK Corretto ver 11.0.13)
   나) Spring Boot ver 2.5.2
   다) `swagger2`, `actuator`, `firebase`, `spring-cloud-starter-aws:2.2.5.RELEASE`
2. Frontend
   가) React Native 0.64.2
   나) Node.js 16.13.0
   다) `expo 43.0.0`, `redux 4.1.2`, `reduxjs/toolkit 1.6.2`
3. AWS

   가) S3 / bucket (Image upload)

   나) nginx/1.18.0 (Ubuntu)

   다) jenkins2.303.2

   라) Docker 20.10.10

   마) kubernetes 1.22.3

빌드 상세내용

1. Frontend (React Native)
   a. 폴더 확인

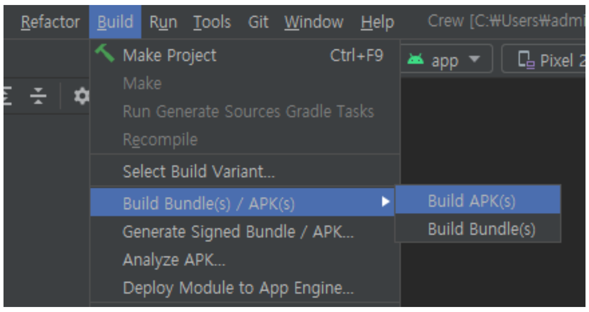
      android/app/src/main/assets 폴더가 있는지 확인, 없다면 생성

   b. terminal에서 다음 명령어로 bundle 파일 생성

```
react-native bundle --platform android --dev false --entry-file index.js --bundle-output
android/app/src/main/assets/index.android.bundle --assets-dest android/app/src/main/res/
```

c. Android Studio에서 프로젝트 로드

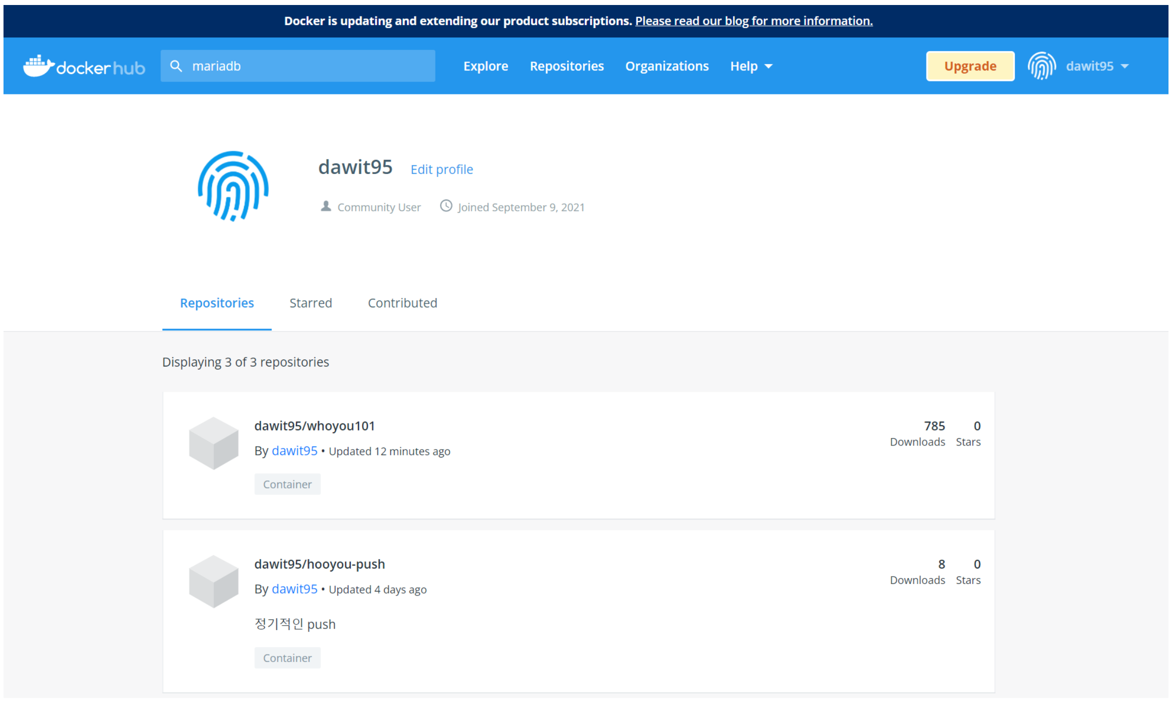
d. 상단 메뉴바의 `Build` > `Build Bundle / APK` > `Build APK`



e. `-assets-dest` 옵션에서 명시한 경로에 app-debug.apk 파일이 생성된 것을 확인

2. Backend (server)

gitlab의 프로젝트 repository(branch : develop)에 webhook을 트리거로 jenkins에서 server를 build및 docker image로 만듬. 만들어진 server image는 docker hub(아래 사진)에 저장되어 있음.



Displaying 3 of 3 repositories

dawit95/whoyou101
By dawit95 • Updated 12 minutes ago
Container
785 Downloads 0 Stars

dawit95/hooyou-push
By dawit95 • Updated 4 days ago
정기적인 push
Container
8 Downloads 0 Stars

Kubernetes Cluster 만들기

a. Control-Plane & Worker 컴퓨터 공통 설정

```
## Swap OFF
sudo swapoff -a
vi /etc/fstab  # SWAP이 정의된 줄을 '#'으로 주석처리해준다.

## NTP(Network Time Protocol) 설정 [EC2에서 설정함으로 AWS에서 제공하는 NTP를 사용
sudo apt install chrony
# /etc/chrony/chrony.conf에 아래 문장 추가
# server 169.254.169.123 prefer iburst minpoll 4 maxpoll 4
sudo /etc/init.d/chrony restart

## apt가 HTTPS로 리포지터리를 사용하는 것을 허용하기 위한 패키지 설치
sudo apt-get update && sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common gnupg2

## 도커 공식 GPG 키 추가:
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key --keyring /etc/apt/trusted.gpg.d/docker.gpg add -
```

```
## 도커 apt 리포지터리 추가:
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

## 도커 CE 설치
sudo apt-get update && sudo apt-get install -y containerd.io=1.2.13-2 docker-ce=5:19.03.11~3-0~ubuntu-$(lsb_release -cs) dock

## /etc/docker 생성
sudo mkdir /etc/docker

## 도커 데몬 설정
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

## /etc/systemd/system/docker.service.d 생성
sudo mkdir -p /etc/systemd/system/docker.service.d

## 도커 재시작 & 부팅시 실행 설정 (systemd)
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker

## Kubernetes util 설치
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

b. Control-Plane 설정

```
sudo kubeadm init --apiserver-advertise-address (마스터 노드 접속 가능한 IP)
```

c. Worker 설정

```
sudo kubeadm join (마스터 노드 접속 가능한 IP):6443 --token (TOKEN) --discovery-token-ca-cert-hash (DISCOVERY_HASH)
```

kubernetes를 통해 Main server 실행

1. Main server deployment와 service를 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: whoyou
  name: whoyou-deployment
spec:
  selector:
    matchLabels:
      type: app
  replicas: 2
  strategy:
    type: RollingUpdate
  revisionHistoryLimit: 5
  minReadySeconds: 10
  template:
    metadata:
      labels:
        type: app
        app: whoyou
    spec:
      containers:
      - name: container
        image: dawit95/whoyou101:v1.0.1
        ports:
        - name: svc-whoyou
```

```
          containerPort: 8080
          protocol: TCP
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: demo-svc-8080
        readinessProbe:
          httpGet:
            path: /actuator/health
            port: demo-svc-8080
        startupProbe:
          httpGet:
            path: /actuator/health
            port: demo-svc-8080
      terminationGracePeriodSeconds: 0
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: whoyou
  name: svc-whoyou
spec:
  selector:
    app: whoyou
  ports:
  - name: svc-whoyou
    port: 9000
    nodePort: 31111
    targetPort: 8080
  type: LoadBalancer
```

2. Push server deployment와 service를 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: push
    type: push-server
  name: push-deployment
spec:
  selector:
    matchLabels:
      type: push
  replicas: 1
  strategy:
    type: RollingUpdate
  revisionHistoryLimit: 1
  template:
    metadata:
      labels:
        type: push
        app: push
    spec:
      containers:
      - name: container
        image: dawit95/hooyou-push:v1.0
        ports:
        - name: svc-push
          containerPort: 8080
          protocol: TCP
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: svc-push
        readinessProbe:
          httpGet:
            path: /actuator/health
            port: svc-push
        startupProbe:
          httpGet:
            path: /actuator/health
            port: svc-push
      terminationGracePeriodSeconds: 0
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: push
  name: svc-push
spec:
  selector:
    app: push
  ports:
  - name: svc-push
    port: 9900
    protocol: TCP
    nodePort: 31199
    targetPort: 8080
  type: LoadBalancer
```

3. Backend (DB)
    a. mariadb image pull

       `docker pull mariadb`

    b. docker 환경에 mariadb container 실행

       `docker container run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD='비밀번호' --name mariadb mariadb`

배포 특이사항

서버가 총 3개로 main server 2개, push server 1개 입니다.

nginx를 이용한 Reverse Proxy를 구현했습니다.

```
## nginx 설정
## /etc/nginx/nginx.conf
worker_processes  1;

events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    proxy_buffering off;
    underscores_in_headers on;
    client_max_body_size 100M;

    server {
        listen       80;
        server_name k5a101.p.ssafy.io;
        client_max_body_size 30M;
        keepalive_timeout 5;
        return 301 https://$server_name$request_uri;

    }
    # HTTPS server
    server {
        listen       443 default_server ssl;
        server_name  k5a101.p.ssafy.io;

        ssl_certificate      /etc/letsencrypt/live/k5a101.p.ssafy.io/fullchain.pem;
        ssl_certificate_key  /etc/letsencrypt/live/k5a101.p.ssafy.io/privkey.pem;
        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        location /api {
             proxy_set_header X-Forwarded-For             $proxy_add_x_forwarded_for;
             proxy_set_header X-Forwarded-Proto https;
             proxy_set_header X-Real-IP $remote_addr;
             proxy_set_header HOST $http_host;
             proxy_set_header X-NginX-Proxy true;

             proxy_pass http://10.104.189.84:9000/api;
             proxy_redirect off;
        }

        location /oauth2 {
             proxy_set_header X-Forwarded-For             $proxy_add_x_forwarded_for;
             proxy_set_header X-Forwarded-Proto https;
             proxy_set_header X-Real-IP $remote_addr;
             proxy_set_header HOST $http_host;
             proxy_set_header X-NginX-Proxy true;

             proxy_pass http://10.104.189.84:9000/oauth2;
             proxy_redirect off;
        }

        location /swagger-ui {
             proxy_set_header X-Forwarded-For             $proxy_add_x_forwarded_for;
             proxy_set_header X-Forwarded-Proto https;
             proxy_set_header X-Real-IP $remote_addr;
             proxy_set_header HOST $http_host;
             proxy_set_header X-NginX-Proxy true;

             proxy_pass http://10.104.189.84:9000/swagger-ui.html;
             proxy_redirect off;
        }
    }
}
```

프로퍼티 정의

1. DB 계정

   ```
   ### root
   id       : root
   password : a101dawit!

   ### whoyou DB 권한만 가진
   id       : userBE
   password : whoyou1admin!
   ```

외부 서비스

1. Google OAUTH 2.0
   - `react-Native-google-signin/google-signin`
   - FireBase App ID : 1:5095342969:android:30cb48fd1f3a9dae33e5a8
   - OAuth 2.0 WEB 클라이언트 ID : 5095342969-dcob776t7ckfeu2gddkb2j4ke2cprfst.apps.googleusercontent.com
2. FCM
   - `com.google.firebase.messaging.FirebaseMessaging`
   - `com.google.gms:google-services:4.3.10`
3. Google Map
   a. `com.google.android.geo`
   b. Geo API_KEY : AIzaSyDUkchpqVWjMDn30qd7lI-q0h6_DpsdVrg
   c. `react-native-geolocation-service`
4. AWS S3
   a. Bucket for Image Upload
   b. Bucket Name: whoyou-bucket