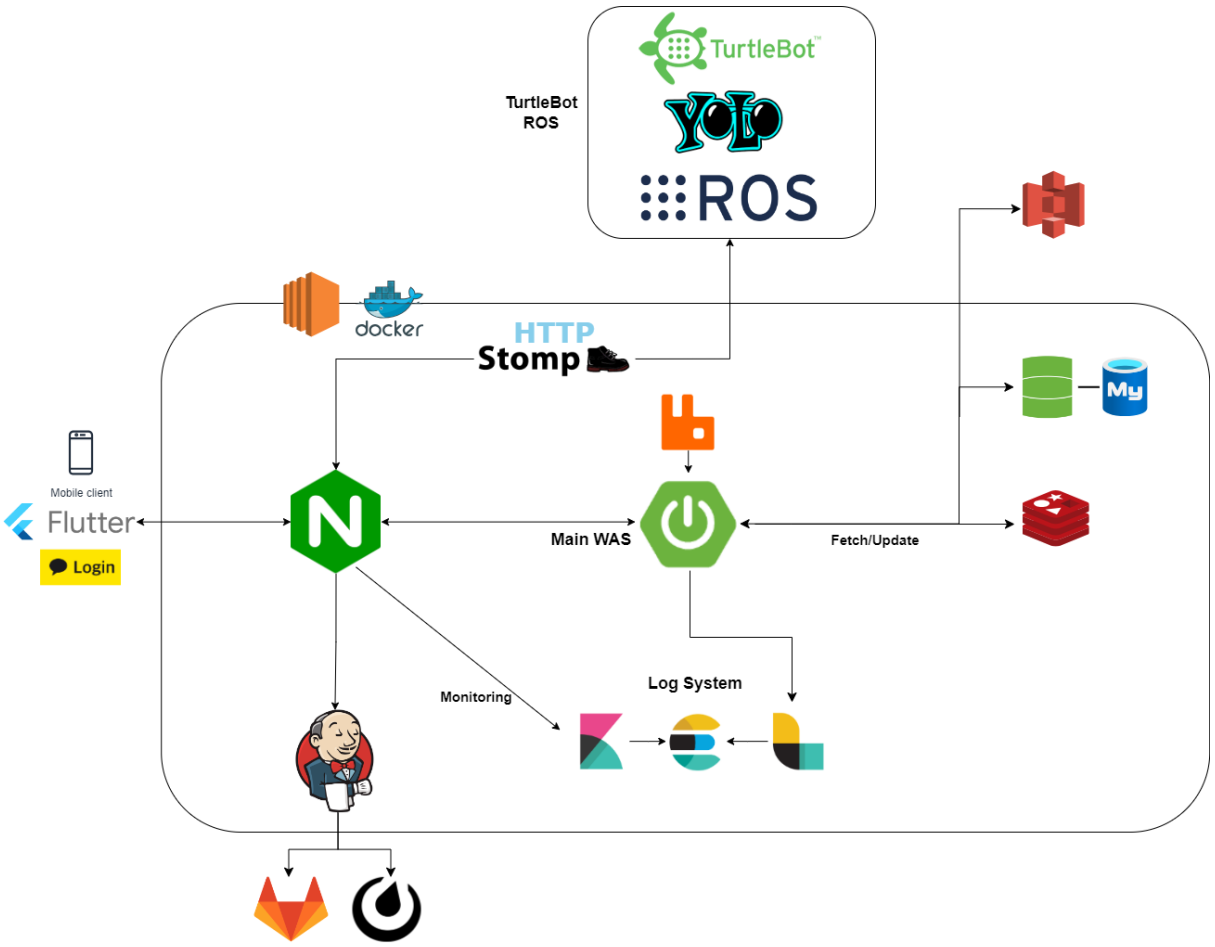
특화_서울2반_A209_포팅메뉴얼

1. 서비스 아키텍처



2. 배포 환경 / 기술 스택

2.1 COMMON

- **Server : Ubuntu 20.04.6 LTS**
- **Docker : 25.0.4**
- **docker-compose : 2.21.0**
- **mySQL : 8.0.36**
- **jenkins : 2.448 (jdk17)**
- **ELK : 8.13.0**
- **nginx : 1.32.0**
- **rabbitMQ : 3.13.0**
- **redis : 7.2**

2.2 FRONT

- **Flutter : 3.19.1**
- **Dart: 3.3.0**
- **Flutter DevTools: 2.31.1**

2.3 BACK

- **JDK : Open JDK 17.0.9**
- **Spring Boot : 3.2.3**
- **Gradle : 8.5**

2.4 IOT

- **Python : 3.7.5**
- **ROS2 : eloquent(20200124 release)**
- **opencv : 3.4.6**
- **YOLOv8**
- **Websocket**
- **Stomp**
- **CUDA**
- **cuDNN**
- **torch**

2.5 DEV TOOL & etc

- **Visual Studio Code**
- **IntelliJ IDE**
- **Jupyter Notebook**
- **AWS EC2**
- **AWS S3**

3. 환경 변수 설정 파일 목록

3.1 elk

- /usr/share/elasticsearch/config/elasticsearch.yml
- /usr/share/logstash/config/logstash.yml
- /usr/share/logstash/pipeline/logstash.conf
- /usr/share/kibana/config/kibana.yml

3.2 rabbitMQ

- **/etc/rabbitmq/rabbitmq.conf**

3.3 nginx

- **/etc/nginx/conf.d/default.conf**

3.4 Back

- **application.properties**
- **JAVA_OPTS=-Djasypt.encryptor.password**

3.5 Front

- \frontend\pubspec.yaml
- \frontend\.env
- \frontend\android\app\src\main\AndroidManifest.xml
- \frontend\android\app\build.gradle

4. 서버 기본 설정

4.1 서버 시간 설정

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

4.2 미러 서버 카카오 서버로 변경

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

4.3 패키지 목록 업데이트 및 패키지 업데이트

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

4.4 SWAP 영역 할당

```
//용량 확인
free -h
//스왑영역 할당
sudo fallocate -l 4G /swapfile
//swapfile 관한 수정
sudo chmode 600 /swapfile
//swapfile 생성
sudo mkswap /swapfile
//swapfile 활성화
sudo swapon /swapfile
//시스템 재부팅 되어도 swap 유지할수 있도록 설정.
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/f
stab
//swap 영역 할당 되었는지 확인
free -h
```

5. 방화벽 설정

포트 번호

```
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
```

```
80                         ALLOW       Anywhere
3306                       ALLOW       Anywhere
8080                       ALLOW       Anywhere
5044                       ALLOW       Anywhere
4885                       ALLOW       Anywhere
4886                       ALLOW       Anywhere
4887                       ALLOW       Anywhere
```

6. 필요 리소스 설치

6.1 Java 17 설치

```
sudo apt-get install openjdk-17-jdk
```

6.2 Docker

6.2.1 도커 설치 세팅

```
sudo apt-get -y install apt-transport-https ca-certificates c
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
    OK
```

6.2.2 도커 저장소 등록

```
sudo add-apt-repository "deb [arch=amd64] https://download.do
```

6.2.3 패키지 리스트 갱신

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

6.2.4 Docker 패키지 설치

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

6.2.5 Docker 일반 유저 권한

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

6.2.6 Docker compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/down
  % Total    % Received % Xferd  Average Speed   Time    Time
                                 Dload  Upload   Total   Spen
  0     0    0     0    0     0      0      0 --:--:-- --:--:
100 56.7M  100 56.7M    0     0  35.4M      0  0:00:01  0:00:
```

6.2.7 Docker compose 실행 권한

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

6.3 MySQL

6.3.1 mySQL 이미지 가져오기

```
sudo docker pull mysql:8.0.36
```

6.3.2 mySQL 컨테이너 생성 및 실행

```
sudo docker run --name mysql-container -e MYSQL_ROOT_PASSWORD
```

6.4 Nginx

6.4.1 Nginx 이미지 다운로드

```
sudo docker pull nginx
```

6.4.2 ginx 서버 가동

```
sudo docker container run --name nginx-container -d -p 80:80
```

6.4.3 Nginx 설정 파일 수정하기

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

```
// 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;
    }
}
```

```
$ certbot —-nginx -d 도메인
email
```

```
y
y
```

6.4.4 도커 이미지 생성

```
ubuntu@ip-172-26-8-88:~$ sudo docker commit nginx nginx_img
```

6.5 Redis& S3

6.5.1 Redis 네트워크 생성

```
sudo docker network create redis-n
```

6.5.2 Redis 이미지 가져오기

```
sudo docker pull redis
```

6.5.3 Redis 실행하기

```
sudo docker run --name redis-container -p 6379:6379 --network
```

6.5.4 S3

6.6 ElasticSearch

6.6.1. Docker-compose 설치

6.6.2. ELK 환경 설치

```
git clone https://github.com/deviantony/docker-elk.git
```

6.6.3. 환경설정 수정

docker-compose

> 💡 **JVM heap space** 사이즈 설정

service:

environment:

ES_JAVA_OPTS : “-Xmx256m -Xms256m”

kibana.yml

> 💡 kibana_system을 elastic으로 바꾸면 ELK 8버전부터 kibana elastic 연동 시 접근이 거절됨

username이 kibana_system인데

kibana_system은 Built-in Role로**, elasticsearch와 Kibana의 연동에만 사용된다.**

kibana 자체의 로그인은 elasticsearch 계정으로 들어가서

kibana용을 새로 만들거나 그냥 elastic 그대로 쓰거나

logstash.yml

```
xpack.monitoring.elasticsearch.username: elastic
xpack.monitoring.elasticsearch.password: ${ELASTIC_PASSWORD}
```

> 💡 X-Pack is an Elastic Stack extension that provides **security, alerting, monitoring**, reporting, machine learning, and many other capabilities.

logstash.conf

```
input {
  rabbitmq {
    host => "<HOST>"
    port => "<PORT>"
    queue => "<QNAME>"
    user => "<USERNMAE>"
    password => "<PASSWORD>"
    add_field => { "source" => "rabbitmq" }
  }
  file {
    path => "/var/log/spring-boot-app.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
    add_field => { "source" => "logfile" }
  }
}

filter {
  if [source] == "rabbitmq" {
    json {
      source => "message"
      target => "parsed_message"
    }
  }
  else if [source] == "logfile" {
    if "executed in" in [message] {
      grok {
        match => { "message" => "%{TIMESTAMP_ISO8601:timestam
      }



    } else {
      drop { }
    }
  }
}
```

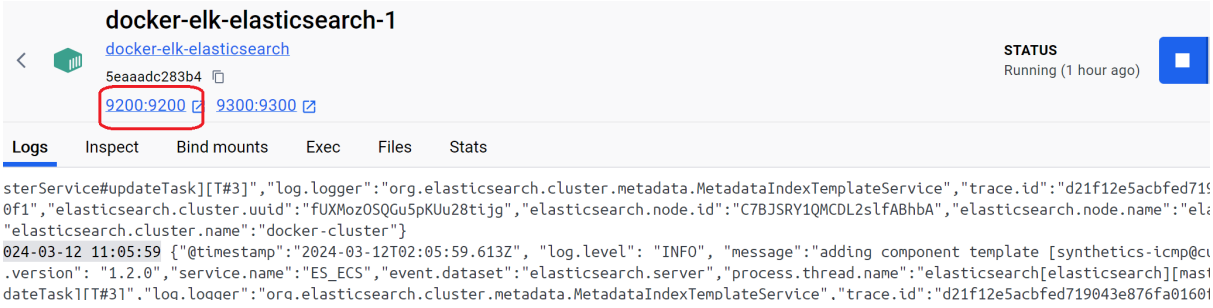
```
output {
  if [source] == "rabbitmq" {
    elasticsearch {
      hosts => "elasticsearch:<ELASTICPORT>"
      user => "<ELASTICUSERNAME>"
      password => "${LOGSTASH_INTERNAL_PASSWORD}"
      index => "ros2-rabbit-%{+YYYY.MM.DD}"
    }
  }
  if [source] == "logfile" {
    elasticsearch {
      hosts => "elasticsearch:<ELASTICPORT>"
      user => "<ELASTICUSERNAME>"
      password => "${LOGSTASH_INTERNAL_PASSWORD}"
      index => "spring-boot-app-logs-%{+YYYY.MM.DD}"
    }
  }
}
```

6.3.4. Docker-compose

Docker-compose build & up

1. docker-elk 루트 디렉토리에서
2. docker-compose build
3. docker-compose up -d
4. docker-compose up setup

docker container 실행



localhost:9200으로 이동

elastic

${password}

입력 후 아래와 같이 뜨면 성공

```
{
  "name" : "elasticsearch",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "fUXMoz0SQGu5pKUu28tijg",
  "version" : {
    "number" : "8.12.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "48a287ab9497e852de30327444b0809e55d46466",
    "build_date" : "2024-02-19T10:04:32.774273190Z",
    "build_snapshot" : false,
    "lucene_version" : "9.9.2",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


kibana도 동일하게 확인

Filebeat

```
## Filebeat configuration
## https://github.com/elastic/beats/blob/main/deploy/docker/f
#

name: filebeat

filebeat.inputs:
  - type: log
    enabled: true
```

```
    paths:
      - ./logs/*    ## /usr/share/filebeat/가 기준.

filebeat.config:
  modules:
    path: ${path.config}/modules.d/*.yml
    reload.enabled: false

filebeat.autodiscover:
  providers:
    # The Docker autodiscover provider automatically retrieve
    # containers as they start and stop.
    - type: docker
      hints.enabled: true
      hints.default_config:
        type: container
        paths:
          - /var/lib/docker/containers/${data.container.id}/*
      templates:
        - condition:
            contains:
              docker.container.image: elasticsearch
          config:
            - module: elasticsearch
              server:
                input:
                  type: container
                  paths:
                    - /var/lib/docker/containers/${data.conta

processors:
  - add_cloud_metadata: ~

# monitoring:
#   enabled: true
#   elasticsearch:
#     username: beats_system
#     password: ${BEATS_SYSTEM_PASSWORD}
```

```
# output.elasticsearch:
#   hosts: [ http://elasticsearch:9200 ]
#   username: filebeat_internal
#   password: ${FILEBEAT_INTERNAL_PASSWORD}


output.logstash:
  hosts: [ "logstash:5044" ] ## localhost 아닌 logstash로 "" 넣
## HTTP endpoint for health checking
## https://www.elastic.co/guide/en/beats/filebeat/current/htt
#

http:
  enabled: true
  host: 0.0.0.0
```

EC2에 올릴 때는 linux에서 curl ifconfig.me 로 public ip 확인 후 아래와 같이 수정

```
output.logstash:
  hosts: [ "EC2_PUBLIC_IP:5044" ]
```

로그 파일이 위치할 디렉토리를 설정

```
filebeat.inputs:
  - type: log
    enabled: true
    paths:
      - ./logs/*    ## /usr/share/filebeat/가 기준.
```

6.3.5. 스프링 부트 Controller AOP 로깅 시스템

각 Controller에 대한 @Aspect클래스에 @Around 적용

```
@Around("execution(* com.ssafy.petpal..controller.*Controll
er.*(..))"
            +
            " && !execution(* com.ssafy.petpal.control.cont
roller.ControlController.*(..))"
            +
            " && !execution(* com.ssafy.petpal.schedule.con
troller.ScheduleController.*(..))")
```

각 Service 레이어 메서드의 수행시간에 대해 로그 생성

```
LOGGER.info("{}#{} executed in {} ms", className, methodName, executionTimeInMs);
```

6.7 Flutter

6.7.1 Libraries

```
// pubspec.yaml
dependencies:
  flutter:
    sdk: flutter
  intl: ^0.19.0
  web_socket_channel: ^2.1.0
  http: ^1.2.1
  collection: ^1.15.0
  timezone: ^0.9.2
  kakao_flutter_sdk_user: ^1.9.0
  cupertino_icons: ^1.0.6
  flutter_secure_storage: ^9.0.0
  flutter_dotenv: ^5.0.2
  logger: ^2.2.0
  stomp_dart_client: ^1.0.3
  firebase_messaging: ^14.7.21
  firebase_core: ^2.27.2
  flutter_local_notifications: ^17.0.0
  get:
```

6.7.2 KAKAO LOGIN

```
// main.dart의 main 함수 내에서 kakao 초기 설정 수행
final kakaoApiKey = dotenv.env['KAKAO_API_KEY']; // load kaka
KakaoSdk.init(nativeAppKey: '$kakaoApiKey'); // KAKAO SDK ini
```

6.7.3 Firebase Cloud Messaging

```
// main.dart
// FCM 관련 인스턴스 선언
final GlobalKey<NavigatorState> navigatorKey = GlobalKey<Navi
const AndroidNotificationChannel channel = AndroidNotificatio
  'high_importance_channel',
  'High Importance Notifications',
  description: 'This channel is used for important notificati
  importance: Importance.max,
);

FlutterLocalNotificationsPlugin flutterLocalNotificationsPlug
    FlutterLocalNotificationsPlugin();
```

```
// main.dart
// main 함수 내에서 Firebase 초기화 및 관련 함수 실행
await Firebase.initializeApp(options: DefaultFirebaseOptions.
setFCM();
```

```
// main.dart
// setFCM 메서드, 앱이 Foreground일 때, 동작중일 때 설정 및 안드로이드
Future<void> setFCM() async {
  FirebaseMessaging messaging = FirebaseMessaging.instance;

  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBac

  NotificationSettings settings = await messaging.requestPerm
    alert: true,
    announcement: false,
    badge: true,
    carPlay: false,
```

```
    criticalAlert: false,
    provisional: false,
    sound: true,
  );

  await FirebaseMessaging.instance.setForegroundNotificationP
    alert: true, // Required to display a heads up notificati
    badge: true,
    sound: true,
  );

  androidNotiSet();

  logger.d('User granted permission: ${settings.authorization

  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    if (navigatorKey.currentContext != null) {
      await GlobalAlertDialog.show(
        navigatorKey.currentContext!,
        message: '알림이 도착했습니다.',
      );
    }
    const AndroidNotificationDetails androidNotificationDetai
        AndroidNotificationDetails(
            'high_importance_channel', 'High Importance Notif
            channelDescription:
                'This channel is used for important notificat
            importance: Importance.max,
            priority: Priority.high,
            ticker: 'ticker');

    const NotificationDetails notificationDetails =
        NotificationDetails(android: androidNotificationDetai
    await flutterLocalNotificationsPlugin.show(
        0,
        '${message.notification!.title}',
        '${message.notification!.body}',
        notificationDetails,
```

```
        payload: 'item x');
  });

  String token = await FirebaseMessaging.instance.getToken() 
  debugPrint("fcmToken : $token");
}

Future<void> androidNotiSet() async {
  if (Platform.isAndroid) {
    FlutterLocalNotificationsPlugin flutterLocalNotifications
        FlutterLocalNotificationsPlugin();
    flutterLocalNotificationsPlugin
        .resolvePlatformSpecificImplementation<
            AndroidFlutterLocalNotificationsPlugin>()!
        .requestNotificationsPermission();

    await flutterLocalNotificationsPlugin
        .resolvePlatformSpecificImplementation<
            AndroidFlutterLocalNotificationsPlugin>()
        ?.createNotificationChannel(channel);
  }

  const AndroidInitializationSettings initializationSettingsA
      AndroidInitializationSettings('app_icon');

  const DarwinInitializationSettings initializationSettingsIO
      DarwinInitializationSettings(
    requestAlertPermission: true,
    requestBadgePermission: true,
    requestSoundPermission: true,
  );

  const InitializationSettings initializationSettings = Initi
    android: initializationSettingsAndroid,
    iOS: initializationSettingsIOS,
  );

  await flutterLocalNotificationsPlugin.initialize(initializa
```

```
}

Future<void> _firebaseMessagingBackgroundHandler(RemoteMessag
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  logger.d("Handling a background message: ${message.messageI
}
```

```
// frontend\android\app\src\main\AndroidManifest.xml
// android 관련 xml 파일 설정 추가

// activity 내에 추가
<meta-data
      android:name="com.google.firebase.messaging.default_not
      android:value="high_importance_channel" />
<intent-filter>
      <action android:name="FLUTTER_NOTIFICATION_CLICK" />
      <category android:name="android.intent.category.DEFAULT
</intent-filter>

// manifest에 추가
<uses-permission android:name="android.permission.WAKE_LOCK"/
<uses-permission android:name="android.permission.POST_NOTIFI
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NOTI
```

```
// \frontend\android\app\build.gradle
// android-defaultConfig에 다음 설정 추가

minSdkVersion 21
targetSdkVersion flutter.targetSdkVersion
versionCode flutterVersionCode.toInteger()
versionName flutterVersionName
```

7. 프로젝트 실행

- **세팅 → 프로젝트 clone**

```
git clone ${our_gitlab_link}
```

7.1 Flutter 다운로드

j10a209.p.ssafy.io 에 접속하여 APP .apk 다운 및 설치

7.1.1 Flutter apk파일

- `\frontend\build\app\outputs\apk\release` 폴더의 `app-release.apk` 파일

7.2 Spring Boot

7.2.1 application.properties

```
server.port=port
server.ssl.enabled=false

#RabbitMQ
spring.rabbitmq.username=username
spring.rabbitmq.password=password
spring.rabbitmq.host=host
spring.rabbitmq.virtual-host=/
spring.rabbitmq.port=port

#RabbitMQ client
rabbitmq.client.id=id
rabbitmq.client.pw=pw

spring.messaging.stomp.buffer-size-limit=2097152 # 2MB

#MySQL
spring.datasource.url=url
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.b

#Redis
spring.data.redis.host=host
spring.data.redis.port=6379



#AWS S3

cloud.aws.s3.bucket = ssafy-petpal-bucket

cloud.aws.credentials.access-key= ak
cloud.aws.credentials.secret-key= sk
cloud.aws.region.static= ap-northeast-2
cloud.aws.region.auto= false
cloud.aws.stack.auto=false

#JWT
jwt.access.token.expiration.seconds=3600
jwt.refresh.token.expiration.seconds=604800
jwt.token.secret-key=sk
#swagger
# swagger-ui custom path
springdoc.swagger-ui.path=/swagger-ui.html

#Firebase
firebase.private-key=pk
firebase.project-id=petpal-d6248
firebase.private-key-id=id
firebase.client-id=id
firebase.auth-uri=https://accounts.google.com/o/oauth2/auth
firebase.token-uri=https://oauth2.googleapis.com/token
firebase.auth-provider-x509-cert-url=https://www.googleapis.c
firebase.client-x509-cert-url=url
```

```
# Spring Boot Logging
#logging.file.name=/var/log/myapp/myapp.log
```

7.2.2 프로젝트 빌드 & 생성 & 실행

```
chmod +x ./gradlew
./gradlew clean bootJar
docker buildx create --use --name mybuilder
docker buildx build --platform linux/amd64,linux/arm64 -t $im
docker buildx build --platform linux/amd64,linux/arm64 -t $im
sudo docker pull $imageName:latest
sudo docker run -i -e TZ=Asia/Seoul -e SPRING_PROFILES_ACTIVE
```

7.3 IOT

7.3.1 세팅

```
git clone ${our_gitlab_link}
```

- **ROS 설치 : 첨부된 'ROS2 개발환경 구축 메뉴얼' 참고**
- **시뮬레이션 설치 : 첨부된 'SSAFY 시뮬레이션 메뉴얼' 참고**

7.3.2 ROS 실행

```
cd /d C:\Users\{Users}\Desktop\S10P22A209\catkin_ws

call C:\dev\ros2-eloquent\setup.bat
call C:\Users\{Users}\Desktop\S10P22A209\catkin_ws\install
\local_setup.bat

colcon build
```

```
ros - simulator (UDP)
cd /d C:\Users\{Users}\Desktop\S10P22A209\catkin_ws
```

```
call C:\dev\ros2-eloquent\setup.bat
call C:\Users\{Users}\Desktop\S10P22A209\catkin_ws\install
\local_setup.bat

ros2 launch C:\Users\SSAFY\Desktop\S10P22A209\catkin_ws\src
\ssafy_bridge\launch\ssafybridge_launch.py
```

```
ros 제어
cd /d C:\Users\{Users}\Desktop\S10P22A209\catkin_ws

call C:\dev\ros2-eloquent\setup.bat
call C:\Users\{Users}\Desktop\S10P22A209\catkin_ws\install
\local_setup.bat

ros2 launch C:\Users\SSAFY\Desktop\S10P22A209\catkin_ws\src
\test_209\launch\default_launch.py
```

```
ros - yolo/server (web socket)
cd /d C:\Users\{Users}\Desktop\S10P22A209\catkin_ws

call C:\dev\ros2-eloquent\setup.bat
call C:\Users\{Users}\Desktop\S10P22A209\catkin_ws\install
\local_setup.bat

ros2 launch C:\Users\SSAFY\Desktop\S10P22A209\catkin_ws\src
\server_communicator\launch\commuication.py
```

```
FSM 중앙 제어 노드
cd /d C:\Users\{Users}\Desktop\S10P22A209\catkin_ws

call C:\dev\ros2-eloquent\setup.bat
call C:\Users\{Users}\Desktop\S10P22A209\catkin_ws\install
\local_setup.bat

ros2 run ssafy_bridge state_check
```

8. 시연 시나리오

1. **APP 실행 및 로그인**
2. **Home Scan**
    a. [APP] - 하단 바에서 ‘메뉴’ - ‘홈 스캔’ 실행
3. **IoT 기기 등록**
    a. 모빌리티 로봇을 등록할 IoT 기기 근처에 배치
    b. [APP] - 하단 바에서 ‘메뉴’ - ‘가전 관리’ - 우측 하단 ‘+’ 버튼 클릭
    c. 모빌리티 기기와 IoT 기기 통신 대기
    d. IoT 기기 등록 완료
4. **IoT 기기 스케줄링**
    a. [APP] - 하단 바에서 ‘예약’ - 우측 하단 ‘+’ 버튼 클릭
    b. 방, 날짜, 시간 선택
    c. ‘저장’
5. **IoT 기기 실시간 제어**
    a. [APP] - 하단 바에서 ‘제어’
    b. 좌측 상단 ‘방’ 선택
    c. 등록된 가전 아이콘 클릭
6. **모드 전환**
    a. [APP] - 하단 바에서 ‘홈’ - 우측 상단 ‘순찰/트래킹 모드’ 버튼 클릭
    b. 모드 전환 성공 메세지 확인
7. **스트리밍 및 맵 확인**
    a. [APP] - 하단 바에서 ‘홈’ - 좌측 상단 ‘ON’ 버튼 클릭
    b. 연결된 모빌리티 기기의 카메라 영상과 스캔된 지도 확인
8. **알림 확인**
    a. [APP] - 하단 바에서 ‘알림’
    b. 받은 알림 목록 확인
    c. 알림 메세지 클릭 - 알림 메세지 관련 이미지 확인