포팅메뉴얼

개발 환경

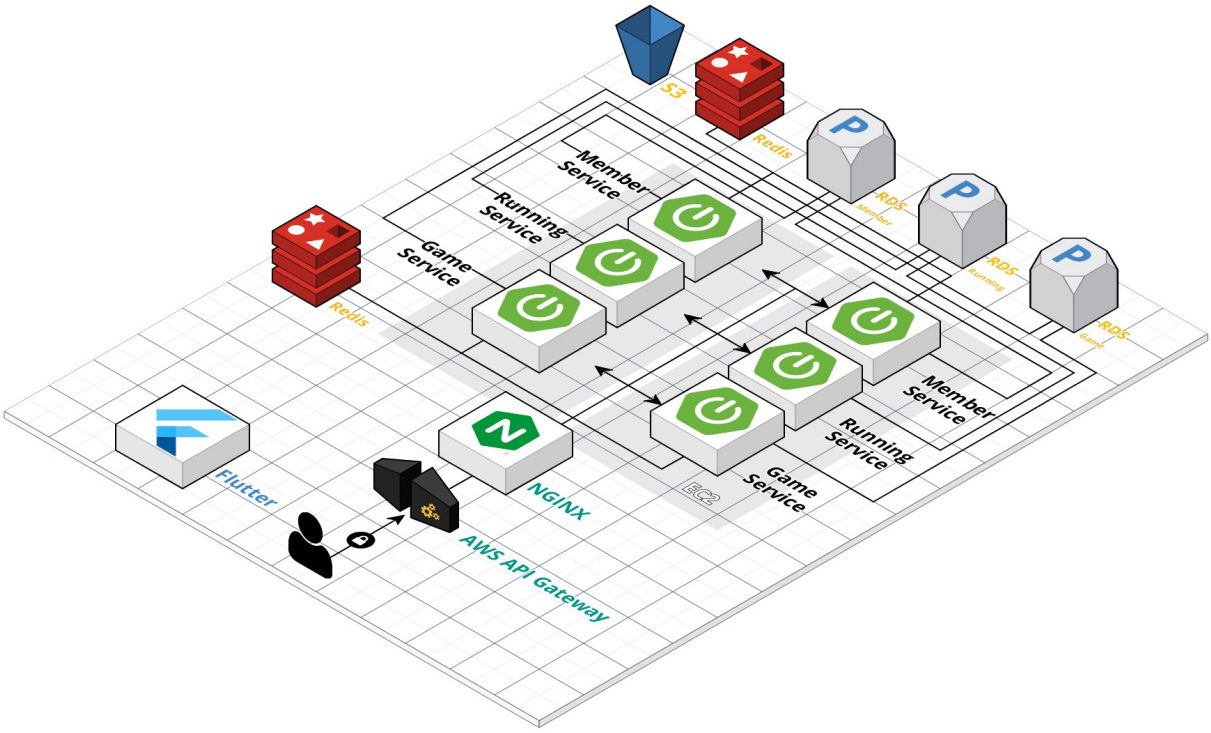
Frontend

Flutter	`3.11.0`
Dart	`3.1.0`

Backend

Spring Boot	`2.7.10`
Java	`11`
Gradle	`gradle-7.6.1`
Amazon RDS	`8.0.32`
Redis	`7.0.11`
NGINX	`1.18.0`

System Architecture



Docker

1. Member Service Docker File

```
# MEMBER Server

FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

2. Running Service Docker File

```
# RUNNING Server

FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

3. Game Service Docker File

```
# GAME Server

FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR
```

```
FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8082
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

NGINX

nginx 설치후 etc/nginx 폴더내 nginx.conf 파일을 확인후 아래와 같은 설정이 적용되어있는지 확인합니다.

```
http{
        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

include가 되어있던 site-enabled 내에 있던 기존의 default 파일을 지운후 YgmgConfig파일을 아래와 같이 생성하였습니다.

```
upstream member {
    server 3.36.117.79:8080;
    server 3.39.250.198:8080;
  }

  upstream game {
    server 3.36.117.79:8082;
    server 3.39.250.198:8082;
  }

  upstream running {
    server 3.36.117.79:8081;
    server 3.39.250.198:8081;
  }


server {
    listen 80;

    location /api/member {
      proxy_pass http://member;
    }

    location /api/game {
      proxy_pass http://game;
    }

    location /api/running {
      proxy_pass http://running;
    }
  }
```

해당 설정후 nginx를 재실행하게 되면 /api/ {service명}으로 들어오게되면 upstream으로 선언된 주소로 로드밸런싱 되게 됩니다.

TeamCity

TeamCity는 JetBrains가 개발한 지속적인 통합(CI) 및 지속적인 배포(CD) 도구입니다. TeamCity는 소프트웨어 개발 프로세스를 자동화하고 효율적으로 관리하기 위해 사용됩니다.

TeamCity는 두 가지 주요 구성 요소로 구성됩니다: TeamCity 서버(Server)와 TeamCity 에이전트(Agent)입니다.

- TeamCity 서버(Server)
  TeamCity 서버는 사용자 인터페이스를 제공하고 전반적인 시스템 관리를 담당합니다. 서버는 빌드 구성, 소스 코드 관리 설정, 빌드 실행 및 결과 보고서 생성과 같은 작업을 수행합니다. 사용자는 웹 브라우저를 통해 TeamCity 서버에 액세스하여 빌드 및 배포 작업을 관리하고 모니터링할 수 있습니다.
- TeamCity 에이전트(Agent)
  TeamCity 에이전트는 실제 빌드 및 배포 작업을 수행하는 컴퓨터 또는 가상 머신입니다. 에이전트는 서버와 통신하여 빌드 및 테스트를 실행하고 결과를 서버로 보고합니다. 서버는 사용 가능한 에이전트 목록을 유지 관리하고 빌드 작업을 가능한 에이전트에 할당하여 분산 처리 및 병렬 실행을 지원합니다. 이를 통해 빌드 프로세스의 속도와 효율성을 향상시킬 수 있습니다.

DockerCompose로 Teamcity Server 설치하기

```
services:
        teamcity:
                image: jetbrains/teamcity-server:2022.04.2
                container_name: teamcity
                restart: always
                ports:
                  - "8111:8111"
                volumes:
                  - ./data:/data/teamcity_server/datadir
                  - ./logs:/opt/teamcity/logs
```

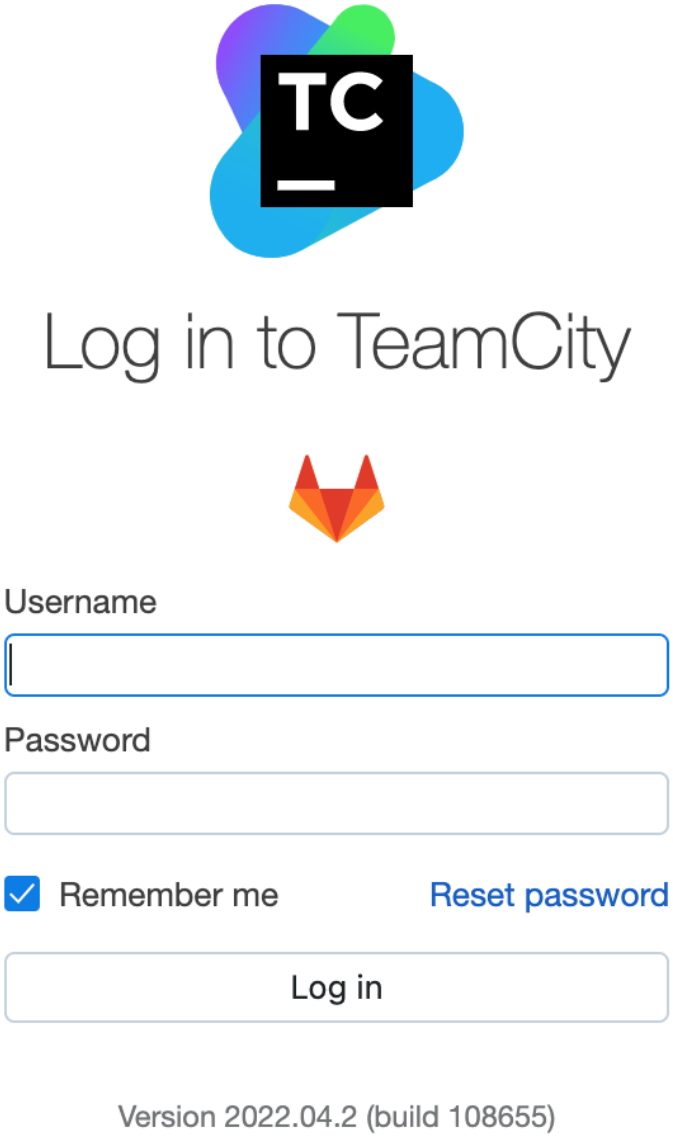
DockerCompose로 Teamcity Agent 설치하기(아래는 Agent를 3개를 설치하는 예시입니다.)

```
services:
        teamcity-agent-01:
                image: jetbrains/teamcity-agent:2022.04.2-linux-sudo
                container_name: agent01
                restart: always
                user: root
                privileged: true
                ports:
                  - 9090:9090
                environment:
                  - SERVER_URL=http://k8c107.p.ssafy.io:8111
                  - AGENT_NAME=agent01
```

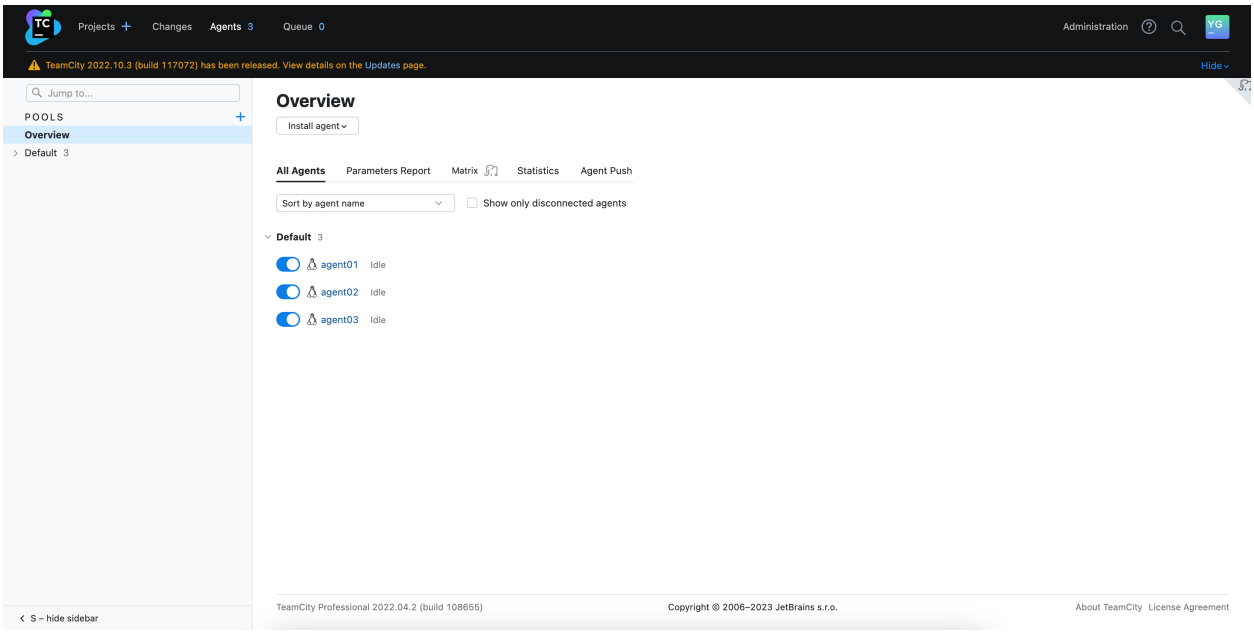
```
          - DOCKER_IN_DOCKER=start
    teamcity-agent-02:
        image: jetbrains/teamcity-agent:2022.04.2-linux-sudo
        container_name: agent02
        restart: always
        user: root
        privileged: true
        ports:
          - 9091:9090
        environment:
          - SERVER_URL=http://k8c107.p.ssafy.io:8111
          - AGENT_NAME=agent02
          - DOCKER_IN_DOCKER=start

    teamcity-agent-03:
        image: jetbrains/teamcity-agent:2022.04.2-linux-sudo
        container_name: agent03
        restart: always
        user: root
        privileged: true
        ports:
          - 9092:9090
        environment:
          - SERVER_URL=http://k8c107.p.ssafy.io:8111
          - AGENT_NAME=agent03
          - DOCKER_IN_DOCKER=start
```

설치가 완료 됬다면 http://{인스턴스 주소}:8111 로 접속합니다.



접속후 Server에서 Agent의 등록을 Authorize 해주게 되면 아래와 같이 Agent를 확인할수 있습니다.



아래는 빌드 파이프라인을 정의하는 구성 코드입니다.

```
package _Self.buildTypes

import jetbrains.buildServer.configs.kotlin.*
import jetbrains.buildServer.configs.kotlin.buildFeatures.dockerSupport
import jetbrains.buildServer.configs.kotlin.buildSteps.dockerCommand
import jetbrains.buildServer.configs.kotlin.buildSteps.gradle
import jetbrains.buildServer.configs.kotlin.triggers.vcs

object Build : BuildType({
    name = "Build"

    vcs {
        root(HttpsLabSsafyComS08finalS08p31c107gitRefsHeadsDevelop)
    }

    steps {
        gradle {
            name = "Build Running"
            tasks = "clean build"
            buildFile = "Backend/Running/build.gradle"
            gradleWrapperPath = "Backend/Running"
        }
        gradle {
            name = "Build Member"
            tasks = "clean build"
            buildFile = "Backend/Member/build.gradle"
            gradleWrapperPath = "Backend/Member"
        }
        gradle {
            name = "Build Game"
            tasks = "clean build"
```

```
            buildFile = "Backend/Game/build.gradle"
            gradleWrapperPath = "Backend/Game"
        }
        dockerCommand {
            name = "Build Running Docker image"
            commandType = build {
                source = file {
                    path = "Backend/Running/Dockerfile"
                }
                namesAndTags = "peter5539/backend-running:lastest"
                commandArgs = "--pull"
            }
        }
        dockerCommand {
            name = "Builid Member Docker Image"
            commandType = build {
                source = file {
                    path = "Backend/Member/Dockerfile"
                }
                namesAndTags = "peter5539/backend-member:lastest"
                commandArgs = "--pull"
            }
        }
        dockerCommand {
            name = "Build Game Docker image"
            commandType = build {
                source = file {
                    path = "Backend/Game/Dockerfile"
                }
                namesAndTags = "peter5539/backend-game:lastest"
                commandArgs = "--pull"
            }
        }
        dockerCommand {
            name = "Running Docker Push"
            commandType = push {
                namesAndTags = "peter5539/backend-running:lastest"
            }
        }
        dockerCommand {
            name = "Member Docker Push"
            commandType = push {
                namesAndTags = "peter5539/backend-member:lastest"
            }
        }
        dockerCommand {
            name = "Game Docker Push"
            commandType = push {
                namesAndTags = "peter5539/backend-game:lastest"
            }
        }
    }

    triggers {
        vcs {
        }
    }

    features {
        dockerSupport {
            loginToRegistry = on {
                dockerRegistryId = "PROJECT_EXT_2"
            }
        }
    }
})
```

- VCS (Version Control System) 설정:
  - HTTPS를 통해 Git 저장소(`HttpsLabSsafyComS08finalS08p31c107gitRefsHeadsDevelop`)를 사용하여 소스 코드를 가져옵니다.
- 빌드 단계(Steps):
  - Gradle을 실행하여 `Backend/Running/build.gradle` 파일에 정의된 빌드 작업을 수행합니다.
  - Gradle을 실행하여 `Backend/Member/build.gradle` 파일에 정의된 빌드 작업을 수행합니다.
  - Gradle을 실행하여 `Backend/Game/build.gradle` 파일에 정의된 빌드 작업을 수행합니다.
  - Docker 커맨드를 실행하여 `Backend/Running/Dockerfile`을 기반으로 Docker 이미지를 빌드합니다.
  - Docker 커맨드를 실행하여 `Backend/Member/Dockerfile`을 기반으로 Docker 이미지를 빌드합니다.
  - Docker 커맨드를 실행하여 `Backend/Game/Dockerfile`을 기반으로 Docker 이미지를 빌드합니다.
  - Docker 이미지를 Docker 레지스트리에 푸시합니다.
- 트리거(Triggers):
  - VCS 변경을 감지하여 자동으로 빌드를 트리거합니다.
- 기능(Features):
  - Docker 지원 기능을 활성화하고, 프로젝트 확장 속성 `PROJECT_EXT_2`를 사용하여 Docker 레지스트리에 로그인합니다.

아래는 배포 파이프라인을 정의하는 구성 코드입니다.

```
package _Self.buildTypes

import jetbrains.buildServer.configs.kotlin.*
import jetbrains.buildServer.configs.kotlin.buildSteps.sshExec
import jetbrains.buildServer.configs.kotlin.triggers.finishBuildTrigger

object Deploy : BuildType({
    name = "Deploy"

    enablePersonalBuilds = false
    type = BuildTypeSettings.Type.DEPLOYMENT
    maxRunningBuilds = 1

    steps {
        sshExec {
            name = "Member_Deploy"
            commands = """
                sudo docker stop backend-member
                sudo docker rm backend-member
                sudo docker image rm peter5539/backend-member
                sudo docker pull peter5539/backend-member:lastest
                sudo docker run --name=backend-member -p 8080:8080 --restart=always -d peter5539/backend-member:lastest
            """.trimIndent()
            targetUrl = "k8c107.p.ssafy.io"
            authMethod = uploadedKey {
```

```
                username = "ubuntu"
                key = "K8C107T.pem"
            }
        }
        sshExec {
            name = "Running_Deploy"
            commands = """
                sudo docker stop backend-running
                sudo docker rm backend-running
                sudo docker image rm peter5539/backend-running
                sudo docker pull peter5539/backend-running:lastest
                sudo docker run --name=backend-running -p 8081:8081 --restart=always -d peter5539/backend-running:lastest
            """.trimIndent()
            targetUrl = "k8c107.p.ssafy.io"
            authMethod = uploadedKey {
                username = "ubuntu"
                key = "K8C107T.pem"
            }
        }
        sshExec {
            name = "Game_Deploy"
            commands = """
                sudo docker stop backend-game
                sudo docker rm backend-game
                sudo docker image rm peter5539/backend-game
                sudo docker pull peter5539/backend-game:lastest
                sudo docker run --name=backend-game -p 8082:8082 --restart=always -d peter5539/backend-game:lastest
            """.trimIndent()
            targetUrl = "k8c107.p.ssafy.io"
            authMethod = uploadedKey {
                username = "ubuntu"
                key = "K8C107T.pem"
            }
        }
    }

    triggers {
        finishBuildTrigger {
            buildType = "${Build.id}"
            successfulOnly = true
        }
    }
})
```

- 배포(BuildType) 설정
  - `name` 속성을 통해 배포 파이프라인의 이름을 지정합니다.
  - `enablePersonalBuilds`를 `false`로 설정하여 개인 빌드를 비활성화합니다.
  - `type` 속성을 `BuildTypeSettings.Type.DEPLOYMENT`로 설정하여 배포 유형을 지정합니다.
  - `maxRunningBuilds`를 1로 설정하여 동시 실행되는 빌드의 최대 수를 제한합니다.
- 단계(Steps) 설정
  - `sshExec`를 사용하여 SSH를 통해 원격 호스트에 명령을 실행합니다.
  - 각 단계(`Member_Deploy`, `Running_Deploy`, `Game_Deploy`)에서는 Docker 컨테이너를 정지하고, 삭제하고, 이미지를 제거한 후, 새로운 이미지를 풀(Pull)하고 컨테이너를 실행합니다.
  - `targetUrl`을 사용하여 배포할 대상 호스트를 지정합니다.
  - `authMethod`를 사용하여 SSH 인증 방법을 설정하고, `username`과 `key`를 지정합니다.

- 트리거(Triggers) 설정
  - `finishBuildTrigger`를 사용하여 이전 빌드의 완료를 트리거로 설정합니다.
  - `buildType`을 `${Build.id}`로 설정하여 이전 빌드(Build)의 ID를 참조합니다.
  - `successfulOnly`를 `true`로 설정하여 성공한 빌드에 대해서만 트리거를 발생시킵니다.

Amazon API Gateway

```
swagger: "2.0"
info:
  description: "니땅내땅 swagger api 입니다."
  version: "1.0.0"
  title: "YGMG Spring Boot REST API"
host: "xofp5xphrk.execute-api.ap-northeast-2.amazonaws.com"
basePath: "/ygmg"
tags:
- name: "running-controller"
  description: "Running Controller"
schemes:
- "https"
paths:
  /api/game:
    get:
      responses:
        "200":
          description: "200 response"
  /api/game/area:
    put:
      responses:
        "200":
          description: "200 response"
  /api/game/area/day/{memberId}/{date}:
    get:
      parameters:
      - name: "memberId"
        in: "path"
        required: true
        type: "string"
      - name: "date"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/game/area/game/{gameId}:
    get:
      parameters:
      - name: "gameId"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/game/area/member/{memberId}:
    get:
      parameters:
      - name: "memberId"
        in: "path"
```

```
      required: true
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/game/area/member/{memberId}/{gameId}:
  get:
    parameters:
    - name: "memberId"
      in: "path"
      required: true
      type: "string"
    - name: "gameId"
      in: "path"
      required: true
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/game/area/{areaId}:
  get:
    parameters:
    - name: "areaId"
      in: "path"
      required: true
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/game/coordinate:
  post:
    responses:
      "200":
        description: "200 response"
/api/game/coordinate/area/{areaId}:
  get:
    parameters:
    - name: "areaId"
      in: "path"
      required: true
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/game/gameId:
  get:
    responses:
      "200":
        description: "200 response"
/api/game/ranking:
  post:
    responses:
      "200":
        description: "200 response"
/api/game/ranking/count:
  get:
    responses:
      "200":
        description: "200 response"
/api/game/ranking/sub:
  post:
    responses:
      "200":
        description: "200 response"
/api/game/ranking/top:
  get:
    responses:
      "200":
        description: "200 response"
/api/game/ranking/{memberId}:
```

```
    get:
      parameters:
      - name: "memberId"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/game/result/{gameId}:
    get:
      parameters:
      - name: "gameId"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/game/result/{gameId}/{memberId}:
    get:
      parameters:
      - name: "memberId"
        in: "path"
        required: true
        type: "string"
      - name: "gameId"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/game/{gameId}:
    get:
      parameters:
      - name: "gameId"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/member/app:
    post:
      responses:
        "200":
          description: "200 response"
  /api/member/check/{nickname}:
    get:
      parameters:
      - name: "nickname"
        in: "path"
        required: true
        type: "string"
      responses:
        "200":
          description: "200 response"
  /api/member/kakao:
    post:
      responses:
        "200":
          description: "200 response"
  /api/member/me/{memberId}:
    get:
      parameters:
      - name: "memberId"
        in: "path"
        required: true
        type: "string"
```

```
    responses:
      "200":
        description: "200 response"
/api/member/mypage:
  get:
    parameters:
    - name: "Authorization"
      in: "header"
      required: false
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/member/profiles:
  get:
    consumes:
    - "application/json"
    parameters:
    - name: "memberList"
      in: "query"
      required: true
      type: "string"
    responses:
      "200":
        description: "200 response"
/api/member/reissue:
  post:
    consumes:
    - "application/json"
    responses:
      "200":
        description: "200 response"
/api/running:
  post:
    tags:
    - "running-controller"
    summary: "런닝 기록 저장"
    description: "런닝 기록 저장합니다."
    operationId: "saveRunningRecordUsingPOST"
    consumes:
    - "application/json"
    produces:
    - "application/json"
    parameters:
    - in: "body"
      name: "RunningRequest"
      description: "runningRequest"
      required: true
      schema:
        $ref: "#/definitions/RunningRequest"
    responses:
      "200":
        description: "저장 성공"
        schema:
          $ref: "#/definitions/MODEL9c4bc6"
      "201":
        description: "Created"
      "401":
        description: "Unauthorized"
      "403":
        description: "Forbidden"
      "404":
        description: "Not Found"
/api/running/coordinate/{runningDetailId}:
  get:
    tags:
    - "running-controller"
    summary: "런닝 좌표 리스트 조회"
    description: "런닝 좌표 리스트를 조회합니다."
    operationId: "findRunningCoordinateUsingGET"
```

```
      produces:
      - "application/json"
      parameters:
      - name: "runningDetailId"
        in: "path"
        description: "runningDetailId"
        required: true
        type: "string"
      responses:
        "200":
          description: "조회 성공"
          schema:
            $ref: "#/definitions/RunningCoordinateResponse"
        "401":
          description: "Unauthorized"
        "403":
          description: "Forbidden"
        "404":
          description: "Not Found"
  /api/running/detail/sum:
    get:
      responses:
        "200":
          description: "조회 성공"
  /api/running/detail/{runningId}:
    get:
      tags:
      - "running-controller"
      summary: "런닝 상세 기록 조회"
      description: "런닝 상세 기록을 조회합니다."
      operationId: "findRunningRecordUsingGET"
      produces:
      - "application/json"
      parameters:
      - name: "runningId"
        in: "path"
        description: "runningId"
        required: true
        type: "string"
      responses:
        "200":
          description: "조회 성공"
          schema:
            $ref: "#/definitions/RunningResponse"
        "401":
          description: "Unauthorized"
        "403":
          description: "Forbidden"
        "404":
          description: "Not Found"
  /api/running/{memberId}:
    get:
      tags:
      - "running-controller"
      summary: "회원 단위 런닝 기록 전체 조회"
      description: "회원이 가지고 있는 런닝 내역을 조회합니다."
      parameters:
      - name: "memberId"
        in: "path"
        description: "memberId"
        required: true
        type: "string"
      responses:
        "200":
          description: "조회 성공"
definitions:
  RunningCoordinateResponse:
    type: "object"
    properties:
      runningCoordinateList:
```

```
        type: "array"
        items:
          $ref: "#/definitions/RunningCoordinateDto"
      runningDetailId:
        type: "integer"
        format: "int64"
  title: "RunningCoordinateResponse"
Coordinate:
  type: "object"
  properties:
    coordinateTime:
      type: "string"
    lat:
      type: "number"
      format: "double"
    lng:
      type: "number"
      format: "double"
  title: "Coordinate"
MODEL9c4bc6:
  type: "object"
RunningCoordinateDto:
  type: "object"
  properties:
    coordinateTime:
      type: "string"
    runningLat:
      type: "number"
      format: "double"
    runningLng:
      type: "number"
      format: "double"
  title: "RunningCoordinateDto"
RunningResponse:
  type: "object"
  properties:
    runningDetailId:
      type: "integer"
      format: "int64"
    runningDistance:
      type: "number"
      format: "double"
    runningEnd:
      type: "string"
    runningKcal:
      type: "number"
      format: "double"
    runningMode:
      type: "string"
    runningPace:
      type: "number"
      format: "double"
    runningStart:
      type: "string"
    runningTime:
      type: "string"
  title: "RunningResponse"
RunningRequest:
  type: "object"
  properties:
    coordinateList:
      type: "array"
      items:
        $ref: "#/definitions/Coordinate"
    memberId:
      type: "integer"
      format: "int64"
    runningDistance:
      type: "number"
      format: "double"
```

```
      runningEnd:
        type: "string"
      runningKcal:
        type: "number"
        format: "double"
      runningPace:
        type: "number"
        format: "double"
      runningStart:
        type: "string"
      runningTime:
        type: "string"
    title: "RunningRequest"
```

- 기본적으로 Amazon API Gateway는 SSL/TLS를 지원하며, API Gateway의 엔드포인트에 대한 HTTPS 액세스를 자동으로 제공합니다.

1. API Gateway의 엔드포인트는 HTTPS를 통해 보호됩니다. API Gateway가 생성한 엔드포인트는 SSL/TLS 인증서를 사용하여 암호화된 HTTPS 연결을 제공합니다.
2. Amazon API Gateway는 사용자 정의 도메인 이름을 지원하며, 사용자는 API Gateway에 대해 자체 SSL/TLS 인증서를 구성할 수 있습니다. 이를 통해 사용자는 자체 도메인과 관련된 SSL/TLS 인증서를 사용하여 HTTPS 연결을 설정할 수 있습니다.

```
URL 호출: https://xofp5xphrk.execute-api.ap-northeast-2.amazonaws.com/ygmg
```

Flutter 라이브러리 파일

```
name: ygmg
description: ygmg Flutter project.
# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFound
ationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: '>=3.1.0-20.0.dev <4.0.0'

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
```

```
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
  sensors:

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  flutter_svg: ^2.0.5
  change_app_package_name: ^1.1.0
  google_maps_flutter: ^2.2.5
  slide_countdown: ^0.5.0
  fl_chart: ^0.62.0
  location: ^4.2.0
  sleek_circular_slider: ^2.0.1
  flutter_polyline_points: ^1.0.0
  google_maps_utils: ^1.5.0+0
  carousel_slider: ^4.2.1
  kakao_flutter_sdk: ^1.4.2
  dio: ^4.0.0
  flutter_secure_storage: ^8.0.0
  tap_to_expand: ^0.6.1
  image_picker: ^0.8.7+4
  shared_preferences: ^2.1.0
  flutter_riverpod: ^1.0.0
  http_parser: ^4.0.0
  intl: ^0.18.1
  rflutter_alert: ^2.0.4
  confetti: ^0.7.0
  flutter_image_compress: ^2.0.3


dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the  icons in
  # the material Icons class.
  uses-material-design: true
  assets:
    - assets/images/
    - assets/style/

  # To add assets to your application, add an assets section, like this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages
```

```
# To add custom fonts to your application, add a fonts section here,
# in this "flutter" section. Each entry in this list should have a
# "family" key with the font family name, and a "fonts" key with a
# list giving the asset and other descriptors for the font. For
# example:
fonts:
  - family: Yuiyeon
    fonts:
      - asset: fonts/yuiyeon.ttf
  - family: NotoSans
    fonts:
      - asset: fonts/NotoSansKR-Black.otf
        weight: 900
      - asset: fonts/NotoSansKR-Bold.otf
        weight: 700
      - asset: fonts/NotoSansKR-Medium.otf
        weight: 500
      - asset: fonts/NotoSansKR-Regular.otf
        weight: 400
      - asset: fonts/NotoSansKR-Light.otf
        weight: 300
      - asset: fonts/NotoSansKR-Thin.otf
        weight: 100

# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```