포팅 매뉴얼

태그	운영 & 유지보수
담당자	Taemin Im 정여진 Dongjoon Lee
마지막 수정	
비고	
진행 상황	
필수 여부	필수

1. 개요
    1-1. 프로젝트 개요
    1-2. 프로젝트 사용 도구
    1-3. 개발 환경
        1-3-1. Frontend
        1-3-2. Backend
        1-3-3. Jenkins
        1-3-4. Storage
        1-3-5. AWS
2. 빌드
    2-1. 환경 변수 및 버전 관리
        2-1-1. FrontEnd
        2-1-2. BackEnd
    2-2. 배포 하기
        2-2-1. FrontEnd
        2-2-2. BackEnd
3. DB 덤프 파일
4. 시연 시나리오

1. 개요

1-1. 프로젝트 개요

- **말랑 연구소는 게임 기반 실시간 온라인 브레인 스토밍 플랫폼입니다.**
- 브레인 스토밍, 협업, 아이스 브레이킹, 의견 공유 환경 조성을 위한 서비스를 제공합니다.

1-2. 프로젝트 사용 도구

분야	목록
이슈 관리	JIRA
형상 관리	Gitlab
커뮤니케이션	Mattermost, Notion, Discord
디자인	Figma
UCC	Movavi

1-3. 개발 환경

1-3-1. Frontend

Langauge	TypeScript 5.0.2
Framework	React 18 Next.js 13
Tool	Visual Studio Code
Node	18.14.1

1-3-2. Backend

Langauge	Java 17
Framework	Spring Framework 3.0.6
Tool	IntelliJ IDEA 2022.3.1
JDK	Azul Zulu version 17.0.6

1-3-3. Jenkins

server	ubuntu 20.04 lts [focal]
version	10.7.8-focal
id	malang
password	malanglab12!
host	jenkins.malang-lab.com

1-3-4. Storage

- **MariaDB**

server	ubuntu 20.04 lts [focal]
version	10.7.8-focal
id	malanglab
password	ssafy102malanglab!
host	k8c102.p.ssafy.io
port	`3306`

- **Redis**

server	ubuntu 20.04 lts [focal]
version	7.0.11-alpine
password	ssafy102malanglab@
host	k8c102.p.ssafy.io
port	`6379`

- **RabbitMQ**

server	ubuntu 20.04 lts [focal]
version	rabbitmq:3-management-alpine
id	malanglab
password	ssafy102malanglab#
host	k8c102.p.ssafy.io

port	`15672`

1-3-5. AWS

- S3

bucket name	static.malang-lab.com
region	아시아 태평양(서울) ap-northeast-2

  - 내부 폴더 구조

```
profile/
room/
  {roomId}/
static/
test/
```

  - 버킷 정책

```
{
    "Version": "2012-10-17",
    "Id": "Policy",
    "Statement": [
        {
            "Sid": "Stmt",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::static.malang-lab.com/*"
        },
        {
            "Sid": "Stmt1679882069060",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::990929040250:user/malang-member"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::static.malang-lab.com/*"
        }
    ]
}
```

  - CORS 구성

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "HEAD",
            "GET",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

- EC2

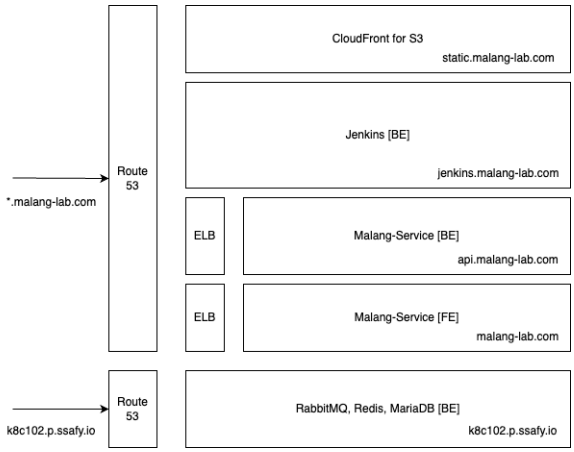
인스턴스 유형	용도
t3.medium	Frontend
t2.micro	Jenkins
m4.large	Backend

인스턴스 유형	용도
m4.xlarge	DB, Redis, RebbitMQ

- Route53

  `malang-lab.com`

- AWS Certificate Manager
  - Route53을 통해서 도메인을 구입할 시, SSL/TLS인증서를 ACM이 담당합니다.
  - `malang-lab.com`, `*.malang-lab.com` 두개로 구성되어 있으며 Apex 도메인의 경우 `*` 에 포함이 되지 않아 별도로 추가해줍니다.
  - 인증방식은 DNS 인증으로 진행합니다.
    - Route53에 CNAME으로 ACM에서 요청한 값을 등록합니다.
- ELB



  - malang-lab.com
    - 용도: FE 도메인 `SSL 적용 완료`
  - jenkins.malang-lab.com
    - 용도: Jenkins용 도메인
  - api.malang-lab.com
    - 용도: BE 도메인 `SSL 적용완료`
  - static.malang-lab.com
    - 용도: S3용 CloudFront `SSL 적용완료`
      - (단, CloudFront환경에서는 `버지니아 북부` 로 ACM이 등록되어야 함)
- IAM

  권한 정책
  - CloudFrontFullAccess
  - AmazonEC2FullAccess
  - AmazonRoute53FullAccess
  - AmazonS3FullAccess
  - AWSCertificateManagerFullAccess
  - ElasticLoadBalancingFullAccess

2. 빌드

2-1. 환경 변수 및 버전 관리

2-1-1. FrontEnd

- **package.json**

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@heroicons/react": "^2.0.17",
    "@reduxjs/toolkit": "^1.9.5",
    "@stomp/stompjs": "^7.0.0",
    "@types/dom-to-image": "^2.6.4",
    "@types/file-saver": "^2.0.5",
    "@types/howler": "^2.2.7",
    "@types/node": "18.16.1",
    "@types/react": "18.2.0",
    "@types/react-dom": "18.2.1",
    "@types/react-redux": "^7.1.25",
    "@types/sockjs-client": "^1.5.1",
    "animate.css": "^4.1.1",
    "autoprefixer": "10.4.14",
    "axios": "^1.4.0",
    "dom-to-image": "^2.6.0",
    "eslint": "8.39.0",
    "eslint-config-next": "13.3.1",
    "file-saver": "^2.0.5",
    "gsap": "^3.11.5",
    "immer": "^10.0.1",
    "konva": "7.2.5",
    "next": "13.3.1",
    "next-redux-wrapper": "^8.1.0",
    "postcss": "8.4.23",
    "qrcode.react": "^3.1.0",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-icons": "^4.8.0",
    "react-konva": "^18.2.7",
    "react-redux": "^8.0.5",
    "react-spring": "^9.4.5-beta.0",
    "react-spring-carousel": "^2.0.19",
    "react-wordcloud": "^1.2.7",
    "redux": "^4.2.1",
    "sockjs-client": "^1.6.1",
    "sweetalert2": "^11.7.5",
    "tailwind-scrollbar": "^3.0.0",
    "tailwind-scrollbar-hide": "^1.1.7",
    "tailwindcss": "3.3.2",
    "tsparticles-confetti": "^2.9.3",
    "typescript": "5.0.4",
    "use-image": "^1.1.0"
  },
  "devDependencies": {}
}
```

2-1-2. BackEnd

- **build.gradle**

```
plugins {
  id 'java'
  id 'org.springframework.boot' version '3.0.6'
  id 'io.spring.dependency-management' version '1.1.0'
}

group = 'com.c102'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
  compileOnly {
```

```
    extendsFrom annotationProcessor
  }
}

repositories {
  mavenCentral()
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-aop'
  implementation 'org.springframework.boot:spring-boot-starter-actuator'
  implementation 'org.springframework.boot:spring-boot-starter-mustache'
  implementation 'org.springframework.boot:spring-boot-starter-validation'
  implementation 'org.springframework.boot:spring-boot-starter-web'
  implementation 'io.micrometer:micrometer-tracing-bridge-brave'

  developmentOnly 'org.springframework.boot:spring-boot-devtools'

  /** lombok */
  compileOnly 'org.projectlombok:lombok'
  annotationProcessor 'org.projectlombok:lombok'

  /** WebSocket */
  implementation 'org.springframework.boot:spring-boot-starter-websocket'
  implementation 'org.springframework.boot:spring-boot-starter-reactor-netty'


  /** Message Queue */
  implementation 'org.springframework.boot:spring-boot-starter-amqp'
  testImplementation 'org.springframework.amqp:spring-rabbit-test'
  runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'

  /** Persistence */
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.springframework.boot:spring-boot-starter-data-redis'

  /** ObjectMapper with Redis [jsr: LocalDateTime 저장] */
  implementation 'com.fasterxml.jackson.core:jackson-core'
  implementation 'com.fasterxml.jackson.core:jackson-databind:2.15.0'
  implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.15.0'

  /** rest api test */
  testImplementation 'io.rest-assured:rest-assured'
  implementation 'com.google.guava:guava:31.1-jre'

  /** Test */
  testImplementation 'org.springframework.boot:spring-boot-starter-test'

  /** jasypt */
  implementation 'com.github.ulisesbocchio:jasypt-spring-boot-starter:3.0.5'

  /** AWS S3 */
  implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'

}

tasks.named('test') {
  useJUnitPlatform()
}
```

- **application.yml**

```
spring:
  rabbitmq:
    port: 5672
    stomp-port: 61613
    host: localhost
    virtual-host: /
    username: guest
    password: guest
    system:
      username: guest
      password: guest
  jpa:
    hibernate:
      ddl-auto: create
    properties:
      hibernate:
        show_sql: true
        format_sql: true
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
```

```
    url: jdbc:mariadb://localhost:3306/malang
    username: root
    password: 'root'
  data:
    redis:
      port: 6379
      host: localhost
      password:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
cloud:
  aws:
    s3:
      bucket: static.malang-lab.com
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
    credentials:
      access-key: ENC(unXu+pIUrzXUv7abvtqr803jUsWrrzJ+jyZzcU5iKQs=)
      secret-key: ENC(ASwAeKvdZ6IffS3T3vRYpjI06gS2/63FnWpXENeU6ucpjcp7/Ftki7LabKNR5+ddHLBaGYw11sA=)
```

- Jasypt를 활용한 주요 정보 암호화

- **application-prod.yml**

```
spring:
  rabbitmq:
    port: 5672
    stomp-port: 61613
    host: ENC(R63XLXvnUL0mU32Sts4KUJuqubvj0eyhoggeO7vJ+ic=)
    virtual-host: ENC(aHrz94iirnMQgilAY8VCnhyoCP/QQ4Jc)
    username: ENC(tYq3BmmTuCgAWE+pUbu60A==)
    password: ENC(c3kXGdgS5sDyjLTeQkIScKiHPLFj7XwN)
    system:
      username: ENC(bq3aIoG79aXz9Xlpu0aFId/Gy/QzmBMJ)
      password: ENC(sOjFpjOAVA0PFrShaNFEcezLLYk7WGZC4ymnxfIJHw4=)
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        show_sql: true
        format_sql: true
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: ENC(8e7kwdMKDq3c5ZRnWQAckgf3/X2G5u5C18zXbiWIYNs9utAWnwJTow3pqamuZX1kUorMypyvlrE=)
    username: ENC(NCOgpqaAu9Hb4wUjNpOY6p4tdJIkUgNT)
    password: ENC(/UoGpFo8ByaVICDxKpZVzw1gGNhdd+kv67GnYqC78Lo=)
  data:
    redis:
      port: 6379
      host: ENC(APo4roRnbfRYNUo027T3DSUR0vu5x22288tInpytFt8=)
      password: ENC(ofuv+U0GHQFknWY/ELPzXXP3FYh8H/FR24jmE4qphYA=)
cloud:
  aws:
    s3:
      bucket: static.malang-lab.com
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
    credentials:
      access-key: ENC(unXu+pIUrzXUv7abvtqr803jUsWrrzJ+jyZzcU5iKQs=)
      secret-key: ENC(ASwAeKvdZ6IffS3T3vRYpjI06gS2/63FnWpXENeU6ucpjcp7/Ftki7LabKNR5+ddHLBaGYw11sA=)
management:
  endpoints:
    web:
      exposure:
        include: health
```

- 로컬 테스트 환경과 배포 환경 각각에 대한 환경 적용을 위한 Config 분리
- Jasypt를 활용한 주요 정보 암호화

2-2. 배포 하기

2-2-1. FrontEnd

- 배포 스크립트 `jenkins pipeline`

```
pipeline {
    agent any

    tools {
        nodejs "node-18-14"
    }

    environment {
        FRONT_END='frontend'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'develop', credentialsId: 'dongjoon-gitlab', url: 'https://lab.ssafy.com/s08-final/S08P31C102'
            }
        }

        stage('Build') {
            parallel{
                stage('front-end'){
                    when {
                        changeset "$FRONT_END/**"
                    }
                    steps {
                        sh "cd $FRONT_END && yarn install && yarn build && cd .. && pwd"
                        sh "tar -czf frontend.tar.gz --exclude=$FRONT_END/node_modules $FRONT_END && rm -rf $FRONT_END/node_module
                    }
                }
            }
        }

        stage('Deploy') {
            parallel{
                stage('front-end') {
                    when {
                        changeset "$FRONT_END/**"
                    }
                    steps {
                        sshPublisher(
                            continueOnError: false, failOnError: true,
                            publishers: [
                                sshPublisherDesc(
                                    configName: "front-server",
                                    verbose: true,
                                    transfers: [
                                        sshTransfer(
                                            sourceFiles: 'frontend.tar.gz',
                                            removePrefix: "",
                                            remoteDirectory: "/",
                                            execCommand: "cd /home/ubuntu && sudo ./deploy.sh"
                                        ),
                                    ]
                                )
                            ]
                        )
                        sh "rm -rf frontend.tar.gz"
                    }
                }
            }

        }

    }
}
```

- 배포환경 스크립트 `deploy.sh`

```
#!/bin/bash

cd /home/ubuntu
```

```
tar -xzf frontend.tar.gz
cd /home/ubuntu/frontend
yarn install

pm2 describe "malang" > /dev/null
RUNNING=$?

if [ "${RUNNING}" -ne 0 ]; then
  cd /home/ubuntu/frontend && /
      pm2 start npx --name "malang" -- next start -p 80
else
  pm2 stop "malang" && /
  pm2 delete "malang" && /
  cd /home/ubuntu/frontend && /
  pm2 start npx --name "malang" -- next start -p 80
fi;
```

- PM2를 활용해, 실행하고 있습니다.

2-2-2. BackEnd

- 배포 스크립트 `jenkins pipeline`

```
pipeline {
    agent any

    tools {
        gradle "gradle-7.6"
    }

    environment {
        BUILD_COMMAND = './gradlew clean build -x test'
        BACK_END='backend/MalangLab'
        MALANG_SERVICE = 'MalangService'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'develop', credentialsId: 'dongjoon-gitlab', url: 'https://lab.ssafy.com/s08-final/S08P31C102'
            }
        }

        stage('Build') {
            parallel{
                stage('back-end'){
                    when {
                        changeset "$BACK_END/**"
                    }

                    steps {
                        dir("$BACK_END") {
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
            }
        }

        stage('Deploy') {
            parallel{
                stage('back-end') {
                    when {
                        changeset "$BACK_END/**"
                    }

                    steps {
                        dir("$BACK_END") {
                            sh "docker build -t malang-service ."
                            sh "docker save -o backend.tar malang-service"

                            sshPublisher(
                                continueOnError: false, failOnError: true,
                                publishers: [
                                    sshPublisherDesc(
                                        configName: "back-server",
                                        verbose: true,
                                        transfers: [
                                            sshTransfer(
                                                sourceFiles: 'backend.tar',
```

```
                                        removePrefix: "",
                                        remoteDirectory: "/",
                                        execCommand: "sudo docker load -i backend.tar"
                                    ),
                                    sshTransfer(
                                        execCommand: "sudo docker stop malang-service || true && docker rm malang-service
                                    ),
                                    sshTransfer(execCommand: "sudo docker-compose up -d")
                                ]
                            )
                        ]
                    )
                }
            }
        }
    }

    }
}
```

3. DB 덤프 파일

- 없음

4. 시연 시나리오

1. 호스트는 방을 생성합니다
    a. 방 제목을 작성합니다
    b. 게임 모드를 선택합니다
    c. 진행할 게임을 선택합니다
    d. 라운드 옵션을 선택합니다
        i. 라운드는 최대 3개를 선택할 수 있습니다
        ii. 라운드에 사용할 제시어를 작성합니다
        iii. 라운드에 사용할 히든 단어를 작성합니다
        iv. 라운드 진행 시간을 설정합니다
    e. 생성된 대기방으로 이동하며 호스트의 대기방 상단에는 방의 PIN 번호를 확인할 수 있습니다
2. 게스트는 생성된 방에 접속합니다
    a. 생성된 방의 PIN 번호를 입력합니다
    b. 말랑이 아바타를 생성합니다
    c. 닉네임 중복검사를 진행합니다
    d. 생성된 대기방으로 이동하며 실시간으로 접속한 게스트들의 현황을 확인할 수 있습니다
3. 호스트는 게임을 진행합니다
    a. 게임 시작을 위한 로딩이 진행됩니다

    b. 제시어와 현재 게임에 참여한 인원 수, 그리고 제한 시간을 확인할 수 있습니다
    c. 실시간으로 게스트가 입력한 중복되지 않은 단어의 수를 확인합니다
    d. 게임이 완료되면 완료창으로 이동됩니다
4. 게스트는 게임을 진행합니다
    a. 호스트가 게임을 시작하면 게임 시작을 위한 로딩이 진행됩니다
    b. 화면 상단에 제시어를 확인할 수 있습니다
    c. 입력 창을 통해 단어를 입력합니다
        i. 중복된 입력은 오류 알림이 발생합니다
        ii. 제한 길이를 초과한 입력은 오류 알림이 발생합니다
        iii. 정상 입력 처리 시 화면에 나열됩니다
        iv. 게임이 완료되면 완료창으로 이동됩니다
5. 호스트는 게임 결과를 확인합니다
    a. 워드 클라우드를 통해 입력된 단어와 단어의 수에 대한 시각화된 자료를 확인합니다
    b. 히든 단어 입력을 통해 히든 단어와 히든 단어를 맞춘 참가자를 확인합니다
6. 시상식을 진행합니다
7. 게임을 종료합니다