포팅 메뉴얼

태그

1. 사용한 JVM
2. java jdk 17
3. 웹서버:
    a. Apache Tomcat 3.3.2
    b. nginx 프록시 서버, HTtps
4. WAS 제품 등의 종류와 설정 값, 버전(IDE버전 포함) 기재
5. Gradle 8.8
6. NodeJS 20.15.0
7. java jdk 17
8. react 18.3.1
9. reudx 9.1.2
10. Spring boot 3.3.2
11. querydsl-jpa:5.0.0:jakarta
12. firebase:firebase-admin:9.2.0

1. 빌드시 사용되는 환경 변수 등의 상세 기재
    a. .env

```
# custom env
GRAFANA_ADMIN_PASSWORD=wervjlweejrlkiwqope123masd3255

# mysql
MYSQL_DATABASE=togather
MYSQL_USER=ssafy
MYSQL_PASSWORD=wervjlweejrlkiwqope123masd3255
MYSQL_ROOT_PASSWORD=wervjlweejrlkiwqope123masd3255

# port number
FRONTEND_PORT=3000
BACKEND_PORT=8080
JENKINS_PORT=8083
JENKINS_SLAVE_AGENT_PORT=5001
PROMETHEUS_PORT=9090
GRAFANA_PORT=3030
REDIS_PORT=6379
MYSQL_PORT=3306

DOMAIN_OR_PUBLIC_IP=i11b209.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to
OPENVIDU_SECRET=wervjlweejrlkiwqope123masd3255
```

```
# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#                Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#                Please put the certificates files inside folder ./owncert
#                with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#                variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notificati
LETSENCRYPT_EMAIL=tjensk26@gmail.com
```

b. backend.env

```
# backend
JWT_ACCESS_TOKEN_EXP=86400000
JWT_REFRESH_TOKEN_EXP=604800000
JWT_SECRET_KEY=63235c475b37339a1abd79cf7c211fadcf24cdfbeda7f0bb017f666369e2806853bc85

MYSQL_DATABASE=togather
MYSQL_PASSWORD=wervjlweejrlkiwqope123masd3255
MYSQL_USER=ssafy

SPRING_MAIL_HOST=smtp.naver.com
SPRING_MAIL_PASSWORD=xnrpej209
SPRING_MAIL_USERNAME=togather209

AWS_ACCESS_KEY=AKIAVRUVVSABYNCAOON5
AWS_SECRET_KEY=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
AWS_S3_BUCKET=trip-bucket-0515
AWS_S3_BASE_URL=https://trip-bucket-0515.s3.ap-northeast-2.amazonaws.com

OPENVIDU_SECRET=wervjlweejrlkiwqope123masd3255
OPENVIDU_URL=https://i11b209.p.ssafy.io/

KAKAO_LOGIN_CLIENT_ID=7fa5721d9190b0b9d8d0de769a1d1346
KAKAO_LOGIN_GRANT_TYPE=authorization_code
KAKAO_LOGIN_REDIRECT_URI=https://i11b209.p.ssafy.io/oauth/kakao/callback
KAKAO_LOGIN_SECRET_ID=9pscROSYfEiXPIALeojJcXoeIgtbCVTh

GOOGLE_APPLICATION_CREDENTIALS=fcm.json
```

c. frontend.env

```
VITE_API_URL=/api

# OCR 인식 요청 secret key
VITE_OCR_API_SECRET_KEY=TVhqY0lyalRtVkpWeHlhRmN2TGFESEpUZ3VVVm9Ea3E=
```

```
# General OCR 인식 요청 secret key
VITE_GENERAL_OCR_API_SECRET_KEY=R01laGVsVldIY2xsWXVFZUNsWW9qclF0WGpiVUVveHA=

# Openai api key
VITE_OPENAI_API_KEY=sk-proj-RFyJETacqq-AksZOsMqkOcWqLRriEVWZys-UuNf__Z8lcltz_DpmMNroI

# firebase 관련 설정
VITE_API_KEY=AIzaSyArbTNNfTpcC0PbDO0G9tieOWpYETJuLWA
VITE_AUTH_DOMAIN=togather-ea43c.firebaseapp.com
VITE_PROJECT_ID=togather-ea43c
VITE_STORAGE_BUCKET=togather-ea43c.appspot.com
VITE_MESSAGING_SENDER_ID=304435419611
VITE_APP_ID=1:304435419611:web:b69e5fc40104c1399d2ffe
VITE_MEASUREMENT_ID=G-PJK5BFHCLS
VITE_VAPID_KEY=BD0ZzwKSpjmdyvcqUwEsA37mjA7-KDMJWJ_jkMmIjSQdl7FUKHbcWOaLeWTzMW0Hnrfouc

# 카카오 로그인 key
VITE_KAKAO_LOGIN_CLIENT_ID=7fa5721d9190b0b9d8d0de769a1d1346
VITE_KAKAO_LOGIN_GRANT_TYPE=authorization_code
VITE_KAKAO_LOGIN_REDIRECT_URI=https://i11b209.p.ssafy.io/oauth/kakao/callback
```

2. 배포 시 특이사항 기재

제킨스로 자동 배포를 진행

1. 빌드 전에 ec2의 fcm.json 파일을 backend resources 파일에 copy를 한 후 빌드를 진행해야한다.
2. 빌드 후 jenkins 내부에서 docker cli를 다운 받아야 도커 이미지를 만들고 dockerhub에업로드 할 수 있다.
3. ec2에 접속해서 docker hub에서 이미지를 다운 받는다.

배포 (제킨스 환경 변수)

```
AWS_ACCESS_KEY=AKIAVRUVVSABYNCAOON5
AWS_S3_BASE_URL=https://trip-bucket-0515.s3.ap-northeast-2.amazonaws.com
AWS_S3_BUCKET=trip-bucket-0515
AWS_SECRET_KEY=U/TEe2KWGT5HVWDe8pvIY6TKEDcqknTX0LG3CsIM
EC2_HOST=i11b209.p.ssafy.io
GOOGLE_APPLICATION_CREDENTIALS=fcm.json
JWT_ACCESS_TOKEN_EXP=86400000
JWT_REFRESH_TOKEN_EXP=604800000
JWT_SECRET_KEY=63235c475b37339a1abd79cf7c211fadcf24cdfbeda7f0bb017f666369e2806853bc853
MYSQL_DATABASE=togather
MYSQL_PASSWORD=wervjlweejrlkiwqope123masd3255
MYSQL_USER=ssafy
OPENVIDU_SECRET=wervjlweejrlkiwqope123masd3255
OPENVIDU_URL=https://i11b209.p.ssafy.io/
SPRING_MAIL_HOST=smtp.naver.com
SPRING_MAIL_PASSWORD=xnrpej209
SPRING_MAIL_USERNAME=togather209
VITE_API_KEY=AIzaSyArbTNNfTpcC0PbDO0G9tieOWpYETJuLWA
VITE_APP_ID=1:304435419611:web:b69e5fc40104c1399d2ffe
```

```
VITE_AUTH_DOMAIN=togather-ea43c.firebaseapp.com
VITE_GENERAL_OCR_API_SECRET_KEYR01laGVsVldIY2xsWXVFZUNsWW9qclF0WGpiVUVveHA=
VITE_MEASUREMENT_ID=G-PJK5BFHCLS
VITE_MESSAGING_SENDER_ID=304435419611
VITE_OCR_API_SECRET_KEY=TVhqY0lyalRtVkpWeHlhRmN2TGFESEpUZ3VVVm9Ea3E=
VITE_OPENAI_API_KEY=sk-proj-RFyJETacqq-AksZOsMqkOcWqLRriEVWZys-UuNf__Z8lcltz_DpmMNroI\
VITE_PROJECT_ID=togather-ea43c
VITE_STORAGE_BUCKET=togather-ea43c.appspot.com
VITE_VAPID_KEY=BD0ZzwKSpjmdyvcqUwEsA37mjA7-KDMJWJ_jkMmIjSQdl7FUKHbcWOaLeWTzMW0Hnrfoucl
```

상세한 배포 내용 (배포 스크립트도 여기에 있음)

https://successful-mountain-ee3.notion.site/jenkins-git-lab-docker-98c2c650983144ac95cc710727570ddc?pvs=4

배포 스크립트

```
pipeline {
    agent any

    tools {
        gradle 'Gradle 8.8' // Jenkins에 설정한 Gradle 버전 이름
        nodejs 'NodeJS 20.15.0' // Jenkins에 설정한 NodeJS 버전 이름
    }

    stages {
            stage('Install Docker CLI') {
            steps {
                script {
                    // Docker CLI 설치
                    sh '''
                        apt-get install
                        apt-get update && \
apt-get -y install apt-transport-https \
     ca-certificates \
     curl \
     gnupg2 \
     software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /t
add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID"
   $(lsb_release -cs) \
   stable" && \
apt-get update && \
apt-get -y install docker-ce
                    '''
                }
            }
        }
        stage('Checkout') {
```

```
            steps {
                // GitLab에서 코드를 체크아웃합니다.
                 git branch: 'master',
                     credentialsId: 'GIT_CREDENTIALS', // Jenkins에 등록한 Credentials
                     url: 'https://lab.ssafy.com/s11-webmobile1-sub2/S11P12B209.git'
            }
        }
        stage('Copy fcm.json from EC2 to Jenkins') {
            steps {
                sshagent(['EC2_CREDENTIALS']) {
                    // EC2에서 Jenkins 작업 공간으로 파일 복사
                    sh """
                        scp -o StrictHostKeyChecking=no ubuntu@${EC2_HOST}:'/opt/openv
                    """
                }
            }
        }

        stage('Build Backend') {
            steps {
                dir('backend') {
                    sh 'chmod +x gradlew' // 실행 권한 추가
                    // Backend 빌드
                    sh './gradlew clean build'
                }
            }
        }

        stage('Build Frontend') {
            steps {
                dir('frontend') {
                    // Frontend 빌드
                    sh 'npm install'
                    sh 'npm install workbox'
                    sh 'npm install firebase'
                    sh 'npm install openvidu-browser'
                    sh 'npm run build'
                }
            }
        }

        stage('Build Docker Images') {
            parallel {
                stage('Build Backend Docker Image') {
                    steps {
                        dir('backend') {
                            script {
                                // Backend Docker 이미지 빌드
                                backendImage = docker.build("duna293/togather-backend
                            }
                        }
                    }
                }

                stage('Build Frontend Docker Image') {
```

```
                    steps {
                        dir('frontend') {
                            script {
                                // Frontend Docker 이미지 빌드
                                frontendImage = docker.build("duna293/togather-fronter
                            }
                        }
                    }
                }
            }
        }

        stage('Push Docker Images') {
            parallel {
                stage('Push Backend Docker Image') {
                    steps {
                        script {
                            docker.withRegistry('https://registry.hub.docker.com', 'DO
                                backendImage.push() // Backend Docker 이미지 푸시
                                backendImage.push('latest') // Latest 태그로도 푸시
                            }
                        }
                    }
                }

                stage('Push Frontend Docker Image') {
                    steps {
                        script {
                            docker.withRegistry('https://index.docker.io/v1/', 'DOCKER
                                frontendImage.push() // Frontend Docker 이미지 푸시
                                frontendImage.push('latest') // Latest 태그로도 푸시
                            }
                        }
                    }
                }
            }
        }

          stage('Deploy to EC2') {
            steps {
                script {
                    // SSH를 사용하여 EC2 인스턴스에 접속하고 deploy.sh 실행
                    sshagent(['EC2_CREDENTIALS']) {
                        sh """
                            ssh -o StrictHostKeyChecking=no ubuntu@${EC2_HOST} '
                                                bash /opt/openvidu/deploy.sh &&
                                                bash /opt/openvidu/switch.sh
                                            '
                        """
                    }
                }
            }
          }
    }
```

```
        post {
          always {
              cleanWs() // 작업 공간 정리
          }
        }
}
```

1. DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

이름과 닉네임이 Togather이고 role 1인 member data가 필요하다.

가상 계좌를 직접 생성해줘야한다.

application.properties

```
spring.application.name=togather
spa.default-file=/dist/index.html

spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# swagger
springdoc.swagger-ui.path=/api/swagger-ui

# MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/${MYSQL_DATABASE}
spring.datasource.username=${MYSQL_USER}
spring.datasource.password=${MYSQL_PASSWORD}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.maximum-pool-size=80
spring.datasource.hikari.connection-timeout=30000

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

# JWT
jwt.secret.key=${JWT_SECRET_KEY}
jwt.access.expiration=${JWT_ACCESS_TOKEN_EXP}
jwt.refresh.expiration=${JWT_REFRESH_TOKEN_EXP}

# Mail
spring.mail.host=${SPRING_MAIL_HOST}
spring.mail.username=${SPRING_MAIL_USERNAME}
spring.mail.password=${SPRING_MAIL_PASSWORD}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.enable=true

# S3
aws.access-key=${AWS_ACCESS_KEY}
aws.secret-key=${AWS_SECRET_KEY}
```

```
aws.s3.bucket=${AWS_S3_BUCKET}
aws.s3.base-url=${AWS_S3_BASE_URL}

#OpenVidu
openvidu.url=${OPENVIDU_URL}
openvidu.secret=${OPENVIDU_SECRET}

#KakaoLogin
kakao.login.client.id=${KAKAO_LOGIN_CLIENT_ID}
kakao.login.redirect.uri=${KAKAO_LOGIN_REDIRECT_URI}
kakao.login.grant.type=${KAKAO_LOGIN_GRANT_TYPE}
kakao.login.secret.id=${KAKAO_LOGIN_SECRET_ID}

# max-file-size
spring.servlet.multipart.max-file-size=10MB

# fcm
fcm.certification=${GOOGLE_APPLICATION_CREDENTIALS}

server.port=${BACKEND_PORT}

# max-file-size
spring.servlet.multipart.max-file-size=10MB
```

1. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

: 소셜 인증, 포톤 클라우드, 코드 컴파일 등에 활용 된 외부 서비스 가입 및 활용에 필요한 정보

- CLOVA OCR,
- OPENAI API
- kakao Map
- kakao login
- NAVER SMTP
- FCM

1. DB 덤프 파일 최신본
2. 시연 시나리오

: 시연 순서에 따른 site 화면별, 실행별(클릭 위치 등)상세 설명

로그인 화면에서 회원가입 또는 카카오 로그인으로 접속

모임 생성하기로 하면 모임이 생성됨, 모임 초대는 받은 초대 링크를 작성하면 해당 모임에 속한다.

최근에 초대받은 모임으로 들어간다.

모임에서 기존에 있는 일정에 들어간다.

해당 일정에서 공동 작업과 실시간 작업을 진행한다.

통화하기를 눌려 실시간 음성 채팅을 진행하면서

검색창에 해당 모임 장소나 위치를 검색하고 찜 리스트에 추가할 수 있다.

추가한 리스트는 다른 일정으로 선택, 순서를 바꿀 수 있다.

영수증 조회를 클릭하면 영수증 금액 확인 및 저장을 할 수 있다.

ocr과 open ai로 영수증 인식, 잘못된 값이 있으면 수정 → 저장으로 한다.

저장하면 영수증의 상세 페이지가 보인다.

여러 영수증을 등록한 후 일정 종료를 누르면 최종 정산 페이지가 뜬다.

해당 페이지에서 정산 거절을 누르면 일정 종료 상태로 돌아감, 정산 수락해도 돈이 없으면 송금 못 한

pay계좌의 충전 버튼을 누르면 pay 계좌에서 입출금 송금 기능을 할 수 있다.

pay 계좌 명세도 확인할 수 있다.

게임 기능, 사용자들에게 돈을 몰아줄 사람을 차기 위해 간단한 미니게임이 존재한다.

그리고 입출금, 초대 수락 거절, 탈퇴, 영수증 등록, 입출금 송금, 일정 끝내기, 송금 요청 알림이 온다.