



GoodIdea 포팅 메뉴얼



프로젝트 사용 도구

- 이슈 관리 : JIRA
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- CI / CD : Jenkins



개발 환경

Frontend

- VSCode
- React
- JavaScript
- node.js : 22.13.0

Backend

- IntelliJ
- openJDK : 17
- Spring Cloud
- DB : MariaDB, Redis, MongoDB

Infra

- **Server** : AWS EC2
Ubuntu 22.04 LTS
- **Docker** : 27.3.1
- **Docker Compose** :
v2.30.3
- **Nginx** : 1.18.0



외부 서비스

Gitlab OAuth : application-secrets.properties에 해당 내용 작성

Liveblocks : public-key, private-key application-secrets.properties에 해당 내용 작성

OpenAI(GPT) : .env에 해당 내용 작성



환경 변수

Frontend: .env

```
VITE_OPENAI_API_KEY=  
VITE_BASE_URL=  
VITE_AI_BASE_URL=  
VITE_LIVEBLOCKS_SECRET_KEY=  
VITE_LIVEBLOCKS_PUBLIC_KEY=
```

Backend: application.properties, application-secrets.properties

- **discovery-service (Eureka)**

- 따로 숨겨야 할 데이터가 없어서 secrets 파일이 존재하지 않음

```
### application.properties  
  
server.port=8761  
  
spring.application.name=discovery-service  
  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false
```

- **gateway-service**

```
### application.properties  
  
spring.application.name=gateway-service  
  
server.servlet.encoding.charset=UTF-8  
server.servlet.encoding.enabled=true  
server.servlet.encoding.force=true  
spring.profiles.active=secrets  
  
spring.cloud.gateway.discovery.locator.enabled=true
```

```

# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul


# Auth Service
spring.cloud.gateway.routes[0].id=auth-service
spring.cloud.gateway.routes[0].uri=lb://AUTH-SERVICE
spring.cloud.gateway.routes[0].predicates[0]=Path=/auth-service
spring.cloud.gateway.routes[0].predicates[1]=Method=GET, POST
spring.cloud.gateway.routes[0].filters[0]=RemoveRequestHeader:
spring.cloud.gateway.routes[0].filters[1]=RewritePath=/auth-s


# Auth Service-GitLab Data
spring.cloud.gateway.routes[1].id=auth-service-gitlab
spring.cloud.gateway.routes[1].uri=lb://AUTH-SERVICE
spring.cloud.gateway.routes[1].predicates[0]=Path=/auth-service
spring.cloud.gateway.routes[1].predicates[1]=Method=GET
spring.cloud.gateway.routes[1].filters[0]=RemoveRequestHeader:
spring.cloud.gateway.routes[1].filters[1]=RewritePath=/auth-s


### USER-SERVICE
# Getting User Data By Username or ID
spring.cloud.gateway.routes[2].id=user-service-getUser
spring.cloud.gateway.routes[2].uri=lb://USER-SERVICE
spring.cloud.gateway.routes[2].predicates[0]=Path=/user-service
spring.cloud.gateway.routes[2].filters[0]=RemoveRequestHeader:
spring.cloud.gateway.routes[2].filters[1]=RemoveRequestHeader:
spring.cloud.gateway.routes[2].filters[2]=RewritePath=/user-s


# Register
spring.cloud.gateway.routes[3].id=user-service-join
spring.cloud.gateway.routes[3].uri=lb://USER-SERVICE
spring.cloud.gateway.routes[3].predicates[0]=Path=/user-service
spring.cloud.gateway.routes[3].predicates[1]=Method=POST
spring.cloud.gateway.routes[3].filters[0]=RemoveRequestHeader:
spring.cloud.gateway.routes[3].filters[1]=RewritePath=/user-s

```

```

# Test
spring.cloud.gateway.routes[4].id=user-service-welcome
spring.cloud.gateway.routes[4].uri=lb://USER-SERVICE
spring.cloud.gateway.routes[4].predicates[0]=Path=/user-service-welcome
spring.cloud.gateway.routes[4].predicates[1]=Method=GET
spring.cloud.gateway.routes[4].filters[0]=RemoveRequestHeader=Cache-Control
spring.cloud.gateway.routes[4].filters[1]=RemoveRequestHeader=Pragma
spring.cloud.gateway.routes[4].filters[2]=RewritePath=/user-service-welcome,/user-service-welcome

# Auth Service-GitLab Data
spring.cloud.gateway.routes[5].id=auth-service-gitlab
spring.cloud.gateway.routes[5].uri=lb://AUTH-SERVICE
spring.cloud.gateway.routes[5].predicates[0]=Path=/auth-service-gitlab
spring.cloud.gateway.routes[5].predicates[1]=Method=GET
spring.cloud.gateway.routes[5].filters[0]=RemoveRequestHeader=Cache-Control
spring.cloud.gateway.routes[5].filters[1]=RewritePath=/auth-service-gitlab,/auth-service-gitlab

# Remaining User Service
spring.cloud.gateway.routes[6].id=user-service
spring.cloud.gateway.routes[6].uri=lb://USER-SERVICE
spring.cloud.gateway.routes[6].predicates[0]=Path=/user-service
spring.cloud.gateway.routes[6].predicates[1]=Method=POST, GET
spring.cloud.gateway.routes[6].filters[0]=RemoveRequestHeader=Cache-Control
spring.cloud.gateway.routes[6].filters[1]=RewritePath=/user-service,/user-service

#### PROJECT SERVICE
spring.cloud.gateway.routes[7].id=project-service
spring.cloud.gateway.routes[7].uri=lb://PROJECT-SERVICE
spring.cloud.gateway.routes[7].predicates[0]=Path=/project-service
spring.cloud.gateway.routes[7].predicates[1]=Method=POST, GET
spring.cloud.gateway.routes[7].filters[0]=RemoveRequestHeader=Cache-Control
spring.cloud.gateway.routes[7].filters[1]=RewritePath=/project-service,/project-service

#### application-secrets.properties

server.port={포트 번호}

```

```
# Eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
# 로컬 환경
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/e
# 배포 환경
#eureka.client.service-url.defaultZone=http://discovery-servi
eureka.instance.preferIpAddress=true
```

- **auth-service**

```
### application.properties

spring.application.name=auth-service

server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true
spring.profiles.active=secrets

# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul

### application-secrets.properties

server.port={포트 번호}

#Eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
```

```
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/e
#eureka.client.service-url.defaultZone=http://discovery-servi
eureka.instance.preferIpAddress=true

# OAuth 2.0
oauth.gitlab.url.auth=https://lab.ssafy.com/oauth/authorize
oauth.gitlab.url.token=https://lab.ssafy.com/oauth/token
oauth.gitlab.url.api=https://lab.ssafy.com/api/v4
oauth.gitlab.client-id={클라이언트 ID}
oauth.gitlab.client-secret={secret KEY}
oauth.gitlab.redirect-uri=https://goodidea.world/gateway/auth

# Redis
spring.redis.host={Redis 호스트 주소}
spring.redis.port={Redis 포트 번호}
spring.redis.password={Redis 비밀번호}

# JWT Key
jwt.secret-key={JWT KEY}
```

- **user-service**

```
### application.properties

spring.application.name=user-service

server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true
spring.profiles.active=secrets

# JPA settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
```

```

spring.jpa.properties.hibernate.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
spring.sql.init.mode=always

# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul

### application-secrets.properties

#port number
server.port={포트 번호}

#Eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/e
#eureka.client.service-url.defaultZone=http://discovery-servi
eureka.instance.preferIpAddress=true

#DB
spring.datasource.url=jdbc:mariadb://{호스트 주소}:{포트 번호}/{D
spring.datasource.username={DB username}
spring.datasource.password={DB password}

# JWT Key
jwt.secret-key={JWT KEY}

```

- **project-service**

```

### application.properties

```

```
spring.application.name=project-service

server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true
spring.profiles.active=secrets


# JPA settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
spring.sql.init.mode=always


# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul


# JPA settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
spring.sql.init.mode=always


# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul


### application-secrets.properties
```



```
# port number
server.port={포트 번호}

# Eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/e
#eureka.client.service-url.defaultZone=http://discovery-servi
eureka.instance.preferIpAddress=true

#DB
spring.datasource.url=jdbc:mariadb://{호스트 주소}:{포트 번호}/{D
spring.datasource.username={DB username}
spring.datasource.password={DB password}

# MongoDB Configuration
spring.data.mongodb.uri=mongodb://{DB username}:{DB password}

# JWT Key
jwt.secret-key={JWT KEY}

liveblocks_secret_key={라이브블록 secret key}
```

배포

Nginx 설정

```
sudo nano /etc/nginx/sites-available/default
```

```
server {

    server_name goodidea.world;
```

```

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/goodidea.world/full
ssl_certificate_key /etc/letsencrypt/live/goodidea.world/
include /etc/letsencrypt/options-ssl-nginx.conf; # manage
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

    location / {
        proxy_pass http://localhost:3000/;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
    }

    location /jenkins/ {
        proxy_pass http://localhost:8080/jenkins/;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
    }

    location /gateway/ {

        proxy_pass http://localhost:800
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
    }

    location /eureka/ {
        proxy_pass http://localhost:8761/; # 유레카 서
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x

```

```

        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
    }

    location /ws/ {
        proxy_pass http://localhost:1234;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }
}

server {
    if ($host = goodidea.world) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 default_server;
    listen [::]:80 default_server;


    server_name goodidea.world;
    return 404; # managed by Certbot


}

```

Docker 설정

- Docker 설치

<https://docs.docker.com/engine/install/ubuntu/>

```

# Add Docker's official GPG key:
sudo apt-get update

```

```

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

# To Install the latest version
sudo apt-get install docker-ce docker-ce-cli containerd.io do

```

- **Jenkins** (젠킨스 컨테이너 안에 도커 설치)

```

// 도커 방식 설치
cd /home/ubuntu && mkdir jenkins-data

sudo ufw allow 8080/tcp sudo ufw reload sudo ufw status

sudo docker run -d -p 8080:8080 -v /home/ubuntu/jenkins-data:

sudo docker logs jenkins

sudo docker stop jenkins sudo docker ps -a

// 환경 설정 변경

cd /home/ubuntu/jenkins-data

```

```

mkdir update-center-rootCAs

wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center

sudo sed -i 's#https://updates.jenkins.io/update-center.json#

sudo docker restart jenkins

// 도커 안에 도커 설치

// 도커 컨테이너 실행
sudo docker run -d \
-p 8081:8080 \
-p 50000:50000 \
--name jenkins \
-e JENKINS_OPTS="--prefix=/jenkins" \
-v /home/ubuntu/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
jenkins/jenkins:latest
// 젠킨스 컨테이너 접속
docker exec -it jenkins /bin/bash
cat /etc/issue

// 위의 도커 설치 과정 반복

```

- **Docker-Compose.yml** 사용

- 같은 도커 네트워크를 사용하여 컨테이너 주소를 쉽게 각 컨테이너 이름으로 설정할 수 있다

```
eureka.client.service-url.defaultZone=http://discovery-service:8761/eureka
```

```

networks:
  msa-network:
    external: true

```

```

    driver: bridge

services:
  discovery-service:
    container_name: discovery-service
    image: ssafy/discovery-service
    ports:
      - "8761:8761"
    networks:
      - msa-network
    environment:
      - EUREKA_INSTANCE_HOSTNAME=discovery-service
      - EUREKA_CLIENT_SERVICE_URL_DEFAULTZONE=http://discovery
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8761/eure
      interval: 30s
      timeout: 10s
      retries: 5

  gateway-service:
    container_name: gateway-service
    image: ssafy/gateway-service
    ports:
      - "8000:8000"
    networks:
      - msa-network
    depends_on:
      - discovery-service
    environment:
      - EUREKA_CLIENT_SERVICE_URL_DEFAULTZONE=http://discover

  auth-service:
    container_name: auth-service
    image: ssafy/auth-service
    ports:
      - "8001:8001"
    networks:
      - msa-network

```

```

    depends_on:
      - discovery-service
      - gateway-service
    environment:
      - EUREKA_CLIENT_SERVICE_URL_DEFAULTZONE=http://discover

user-service:
  container_name: user-service
  image: ssafy/user-service
  ports:
    - "8002:8002"
  networks:
    - msa-network
  depends_on:
    - discovery-service
    - gateway-service
  environment:
    - EUREKA_CLIENT_SERVICE_URL_DEFAULTZONE=http://discover

project-service:
  container_name: project-service
  image: ssafy/project-service
  ports:
    - "8003:8003"
  networks:
    - msa-network
  depends_on:
    - discovery-service
    - gateway-service
  environment:
    - EUREKA_CLIENT_SERVICE_URL_DEFAULTZONE=http://discover

frontend:
  container_name: frontend
  image: ssafy/frontend
  ports:
    - "3000:80"
  networks:

```

- msa-network

depends_on:

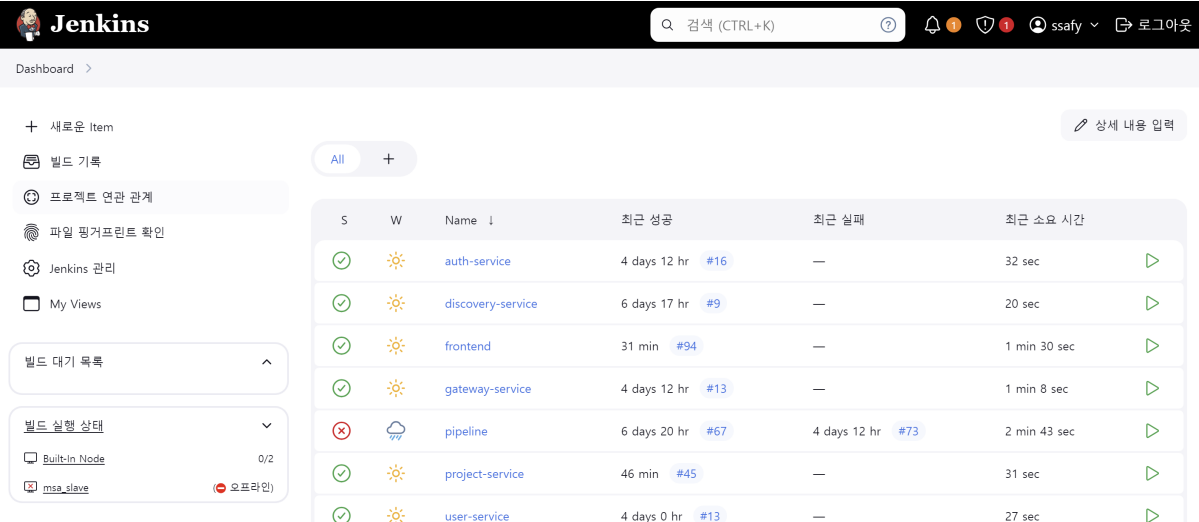
- gateway-service

```
ubuntu@ip-172-31-50-129:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
d43d6a4293a1	ssafy/frontend	"nginx -g 'daemon of...'"	frontend	7 minutes ago	Up 7 minutes
cbc4506bf35b	ssafy/project-service	"java -jar app.jar"	project-service	About an hour ago	Up About an hour
0c35effb493e	ssafy/user-service	"java -jar app.jar"	user-service	4 days ago	Up 4 days
af302ba71eec	ssafy/auth-service	"java -jar app.jar"	auth-service	4 days ago	Up 4 days
45658aed54db	ssafy/gateway-service	"java -jar app.jar"	gateway-service	4 days ago	Up 4 days
c7820ac4cbec	ipirozhenko/y-websocket	"docker-entrypoint.s..."	nervous_engelbart	5 days ago	Up 5 days
7d71bbfb21f2	f93bec74d6c6	"java -jar app.jar"	discovery-service	6 days ago	Up 6 days (unhealthy)
2ab05c58c24b	jenkins/jenkins:latest	"/usr/bin/tini -- /u..."	jenkins	9 days ago	Up 6 days

Jenkins 파이프라인 구축

각 서비스 별로 브랜치를 나눠 따로 개발할 수 있게 구축



The screenshot shows the Jenkins Dashboard with a list of services and their build status. The services listed are auth-service, discovery-service, frontend, gateway-service, pipeline, project-service, and user-service. The pipeline service is marked as unhealthy (red X icon).

S	W	Name	최근 성공	최근 실패	최근 소요 시간
✓	☀	auth-service	4 days 12 hr #16	—	32 sec
✓	☀	discovery-service	6 days 17 hr #9	—	20 sec
✓	☀	frontend	31 min #94	—	1 min 30 sec
✓	☀	gateway-service	4 days 12 hr #13	—	1 min 8 sec
✗	☁	pipeline	6 days 20 hr #67	4 days 12 hr #73	2 min 43 sec
✓	☀	project-service	46 min #45	—	31 sec
✓	☀	user-service	4 days 0 hr #13	—	27 sec

- **Back** ⇒ 다 비슷하여 대표로 project-service 파이프라인을 올린다

```
pipeline {
    agent any

    environment {
```



```

    MATTERMOST_ENDPOINT = 'https://meeting.ssafy.com/hook
    MATTERMOST_CHANNEL = '169fae57d05ab09848da924eee76e36
}

options {
    buildDiscarder(logRotator(numToKeepStr: '3'))
}

tools {
    nodejs 'nodejs'
}

stages {
    stage('Git clone') {
        steps {
            script {
                echo "Cloning repository..."
                git branch: 'dev-project_service',
                    credentialsId: 'gitlab_account',
                    url: 'https://lab.ssafy.com/s11-final
                echo "Clone complete."
            }
        }
    }

    stage('Remove existing container') {
        steps {
            script {
                def serviceName = "project-service"
                def containerId = sh(script: "docker ps -a
                def imageName = "ssafy/${serviceName}"
                if (containerId) {
                    sh "docker stop ${containerId} || true"
                    sh "docker rm -f ${containerId} || true"
                    sh "docker rmi ${imageName} || true"
                }
            }
        }
    }
}

```

```

    }
  }
  stage('Secret download') {
    steps {
      parallel (
        "project-service secret": {
          withCredentials([file(credentialsId:
            sh '''
            ls -al backend/project-service/src
            cp $PROJECT_SECRET_FILE backend/p
            cat backend/project-service/src/m
            ''')
        },
      )
    }
  }

  stage('Build services') {
    parallel {
      stage('Build Project Service') {
        steps {
          dir('backend/project-service') {
            sh '''
            ls -la
            chmod +x ./gradlew
            ./gradlew clean build -x test
            docker build -t ssafy/project-ser
            '''
          }
        }
      }
    }
  }
}

```

```

        stage('Container up') {
            steps {
                sh '''
                    docker compose up -d --no-deps project-service
                '''
            }
        }
    }

    post {
        success {
            script {
                sendNotification('good', '빌드 성공')
                cleanWs()
            }
        }
        failure {
            script {
                sendNotification('danger', '빌드 실패')
                cleanWs()
            }
        }
    }
}

def sendNotification(String color, String status) {
    def gitCommitterName = sh(script: "cd ${env.WORKSPACE} && git log --format='%n' -1")
    def gitCommitMessage = sh(script: "cd ${env.WORKSPACE} && git log --format='%s' -1")

    mattermostSend(
        color: color,
        message: """"${status}: GoodIDEA 프로젝트 서비스 :bulb: #${color} 커밋 작성자 : ${gitCommitterName} 커밋 메시지 : ${gitCommitMessage} (<${env.BUILD_URL}|Details>)"""",
        endpoint: MATTERMOST_ENDPOINT,
        channel: MATTERMOST_CHANNEL
    )
}

```

```
)  
}
```

- **Front**

```
pipeline {  
  agent any  
  
  environment {  
    MATTERMOST_ENDPOINT = 'https://meeting.ssafy.com/hook  
    MATTERMOST_CHANNEL = '169fae57d05ab09848da924eee76e36  
    FRONTEND_IMAGE = 'ssafy/frontend'  
    FRONTEND_CON = 'frontend'  
  }  
  
  options {  
    buildDiscarder(logRotator(numToKeepStr: '3'))  
  }  
  
  tools {  
    nodejs 'nodejs'  
  }  
  
  stages {  
    stage('Workspace Cleanup') {  
      steps {  
        script {  
          echo "Cleaning workspace..."  
          cleanWs() // 빌드 전 작업 디렉토리를 정리하여 이  
        }  
      }  
    }  
  
    stage('Git clone') {  
      steps {  
        script {  
          echo "Cloning repository..."
```

```

        git branch: 'dev-FE',
        credentialsId: 'gitlab_account',
        url: 'https://lab.ssafy.com/s11-final',
        echo "Clone complete."
    }
}

stage('Remove existing container') {
    steps {
        script {
            def serviceName = "frontend"
            def containerId = sh(script: "docker ps -a", returnStdout: true).trim()
            if (containerId) {
                sh "docker stop ${containerId} || true"
                sh "docker rm -f ${containerId} || true"
                sh "docker rmi ssafy/frontend || true"
            }
        }
    }
}

stage('Secret download') {
    steps {
        withCredentials([file(credentialsId: 'frontend_secret', variable: 'FrontFile')]) {
            sh '''
            ls -l

            if [ -f frontend/.env ]; then
                rm frontend/.env
            fi

            cp $FrontFile frontend/.env
            chmod 644 frontend/.env
            '''
        }
    }
}

```

```

stage('Frontend build') {
    steps {
        sh '''
        cd frontend
        npm install
        npm run build
        docker build --no-cache -t $FRONTEND_IMAGE .
        '''
    }
}

stage('Container up') {
    steps {
        sh '''
        docker compose up -d --no-deps frontend
        '''
    }
}

}

post {
    success {
        script {
            sendNotification('good', '빌드 성공')
            cleanWs()
        }
    }
    failure {
        script {
            sendNotification('danger', '빌드 실패')
            cleanWs()
        }
    }
}

}

def sendNotification(String color, String status) {

```

```

















def gitCommitterName = sh(script: "cd ${env.WORKSPACE} &&
def gitCommitMessage = sh(script: "cd ${env.WORKSPACE} &&

mattermostSend(
    color: color,
    message: ""`${status}: GoodIDEA 프론트엔드 :bulb: #${env
커밋 작성자 : ${gitCommitterName}
커밋 메시지 : ${gitCommitMessage}
(<${env.BUILD_URL}|Details>)"`,
    endpoint: MATTERMOST_ENDPOINT,
    channel: MATTERMOST_CHANNEL
)
}

```

- `.env` 와 `application-secrets.properties` 를 secret-file에 추가해준다.

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitlab	GitLab API token
		System	(global)	gitlab_account	gwpeter1000@naver.com/*****
		System	(global)	auth-secret	application-secrets.properties
		System	(global)	user-secret	application-secrets.properties
		System	(global)	project-secret	application-secrets.properties
		System	(global)	gateway-secret	application-secrets.properties
		System	(global)	slave_key	ubuntu
		System	(global)	frontend_env	.env