

CATale 포팅매뉴얼

1. 개발환경

- [1.1. Frontend](#)
- [1.2. Backend](#)
- [1.3. Server](#)
- [1.4. Database](#)
- [1.5. UI/UX](#)
- [1.6. IDE](#)
- [1.7. SCM / Cooperation](#)
- [1.8. Etc](#)

2. 환경 변수

- [2.1. Frontend](#)
- [2.2. Backend](#)
- [2.3. Recommend](#)

3. 프로그램 세팅

- [3.1. Docker](#)
- [3.2. k8s \(microk8s\)](#)
- [3.3. MySQL](#)
- [3.4. Jenkins](#)
- [3.5. argoCD](#)
 - [3.5.1. 설치](#)
 - [3.5.2. gitops repo 연동](#)

4. CI/CD 구축

- [4.1. Jenkins 설정](#)
 - [4.1.1. GitLab Credentials 설정](#)
 - [4.1.2. Jenkins Item 생성](#)
 - [4.1.3. Gitlab Webhook 설정](#)
 - [4.1.4. Pipeline Script 작성](#)
- [4.2. argoCD 설정](#)
 - [4.2.1. helm chart 저장](#)
 - [4.2.2. application 등록](#)
- [4.3 빌드 및 배포 과정](#)
 - [4.3.1. 빌드 시작](#)
 - [4.3.2. 빌드 / image 업로드](#)
 - [4.3.3. 배포](#)

1. 개발환경

1.1. Frontend

- Node.js 20.11.0 (LTS)
- React 18.2.0
 - Zustand 4.5.2
 - lottie-react 2.4.0
 - chart.js 4.4.2
 - apexcharts 3.48.0
 - swiper 11.0.7
- Axios 1.6.8

1.2. Backend

- Java
 - Azul Zulu 17.0.9+8 (LTS)
 - Spring Boot 3.2.3
 - Spring Data JPA 3.2.3
 - Spring Security 3.2.3
 - querydsl-jpa 5.0.0
 - spring-cloud-starter-aws 2.2.6
 - JUnit 5.10.1
 - Lombok 1.18.30
 - Gradle 8.4
- Python
 - Python 3.9.13
 - FastAPI 0.110.0
 - uvicorn 0.29.0
 - lightfm 1.17
 - numpy 1.26.4
 - pandas 2.2.1

1.3. Server

- Ubuntu 20.04.6 LTS
- Docker 26.0.0
- microk8s 1.29
- Jenkins 2.440.1
- argoCD 2.7.2
- Prometheus 2.51.0
- Grafana 10.4.0

1.4. Database

- MySQL 8.0.36

1.5. UI/UX

- Figma

1.6. IDE

- IntelliJ IDEA 2023.2

- Visual Studio Code 1.87

1.7. SCM / Cooperation

- Gitlab 16.7.3
- Git 2.43.0.1
- Jira
- Mattermost
- Notion

1.8. Etc

- Postman 10.24.3

2. 환경 변수

2.1. Frontend

```
SPRING_BASE_URL
KAKAO_MAP_API
```

2.2. Backend

```
SPRING_PROFILES_ACTIVE
SPRING_PROFILES_INCLUDE
SPRING_SECURITY_USER_NAME
SPRING_SECURITY_USER_PASSWORD
SPRING_DATASOURCE_URL
SPRING_DATASOURCE_USERNAME
SPRING_DATASOURCE_PASSWORD
SPRING_SERVLET_MULTIPART_MAX_FILE_SIZE
SPRING_CORS_ALLOWD_ORIGINS
CLOUD_AWS_CREDENTIALS_ACCESSKEY
CLOUD_AWS_CREDENTIALS_SECRETKEY
CLOUD_AWS_S3_BUCKET
CLOUD_AWS_REGION_STATIC
FILTER_PATHS
JWT_SECRET_KEY
JWT_ACCESS_TIME
JWT_REFRESH_TIME
MANAGEMENT_HEALTH_LIVENESSSTATE
MANAGEMENT_HEALTH_READINESSSTATE
```

2.3. Recommend

```
PROJECT_NAME
BACKEND_CORS_ORIGINS
ENCODING
MODEL_PATH
MODEL_NAME
DATASET_PATH
DATASET_NAME
ITEM_FEATURES_FILE
USER_FEATURES_FILE
RATING_FILE
TEST_DATA_FILE
BACKUP_PATH
ORIGINAL_PATH
N_USERS
ENV
LOG_LEVEL
SPRING_BASE_URL
```

3. 프로그램 세팅

3.1. Docker

```
### install-docker.sh

# 1. Uninstall all conflicting packages
for pkg in docker.io docker-doc docker-compose docker-compose-v2 \
    podman-docker containerd runc; do sudo apt-get remove $pkg; done

# 2. 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 -o \
    /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# 3. Add the repository to Apt sources:
echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
    https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 4. Install
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin \
    docker-compose-plugin
```

3.2. k8s (microk8s)

```
# microk8s 설치
snap install microk8s --classic

# microk8s 시작
microk8s.start

# microk8s 상태 확인 및 문제 여부 파악
microk8s.status
microk8s.inspect

# ip forward 허용
iptables -P FORWARD ACCEPT

# addon 활성화
microk8s.enable dns
microk8s.enable dashboard
microk8s.enable ingress
microk8s.enable registry

# alias 설정
snap alias microk8s.kubectl kubectl
snap alias microk8s.helm3 helm

# 토큰 생성
kubectl create token default

# port-forwarding (dashboard test)
kubectl port-forward -n kube-system svc/kubernetes-dashboard 8443:443 --address 0.
```

```
### dashboard-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: k8s-dashboard
  namespace: kube-system
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  tls:
    - hosts:
        - k8s.silvstone.xyz
      secretName: k8s-dashboard-ingress-tls
  rules:
    - host: k8s.silvstone.xyz
      http:
        paths:
          - pathType: Prefix
```

```
path: /
backend:
  service:
    name: kubernetes-dashboard
    port:
      number: 443
```

3.3. MySQL

```
# helm repo 추가 및 업데이트
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update

# helm chart values 다운로드
helm show values bitnami/mysql > mysql-values.yaml

# namespace 생성 (필요시)
kubectl create namespace {namespace}

# values 수정
vim mysql-values.yaml

# 변경 할 value
-----
## root 비밀번호 설정
auth:
  rootPassword: {변경 할 비밀번호}

## service 외부 노출
service:
  type: NodePort
  nodePorts:
    mysql: "30036"

## volume 권한 설정
volumePermissions:
  enabled: true
-----

# helm install
helm install -n {namespace} mysql bitnami/mysql -f mysql-values.yaml
```

3.4. Jenkins

```
# helm repo 추가 및 업데이트
helm repo add jenkins https://charts.jenkins.io
helm repo update

# helm chart values 다운로드
helm show values jenkins/jenkins > jenkins-values.yaml
```

```

# namespace 생성 (필요시)
kubectl create namespce jenkins

# values 수정
vim jenkins-values.yaml

# 변경 할 value
-----
## jdk버전 변경 필요시
controller:
  image:
    tagLabel: {jdk 버전에 맞게} ex) jdk17

## admin 정보 변경
controller:
  admin:
    username: "admin"          # admin id
    password: "{비밀번호}"     # admin pw

## plugin 설정 (관련 오류 발생 시 모두 해제, jenkins 실행 후 설치)
controller:
  installPlugins:
  -----

# helm install
helm install -n jenkins jenkins jenkins/jenkins -f jenkins-values.yaml

```

```

### jenkins-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jenkins
  namespace: jenkins
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTP"
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  tls:
  - hosts:
    - jenkins.silvstone.xyz
    secretName: jenkins-ingress-tls
  rules:
  - host: jenkins.silvstone.xyz
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: jenkins

```

```
port:
  number: 8080
```

3.5. argoCD

3.5.1. 설치

```
# Community Repository Addon 사용 가능 활성화
microk8s enable community

# argoCD addon 활성화
microk8s.enable argocd
```

```
### argocd-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd
  namespace: argocd
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  tls:
    - hosts:
        - argocd.silvstone.xyz
      secretName: argocd-ingress-tls
  rules:
    - host: argocd.silvstone.xyz
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              service:
                name: argo-cd-argocd-server
                port:
                  number: 443
```

3.5.2. gitops repo 연동

4. CI/CD 구축

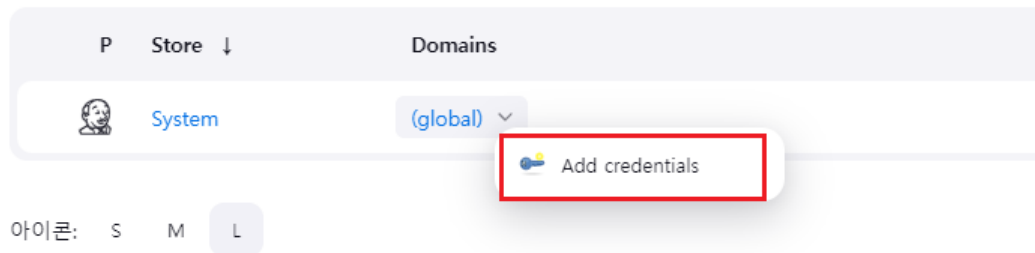
4.1. Jenkins 설정

4.1.1. GitLab Credentials 설정

1. 좌측 메뉴 "Jenkins 관리" 클릭
2. Security → Credentials 클릭

3. "Store : System" → "(global)" → "Add vredentials" 클릭

Stores scoped to Jenkins



4. "Kind"에 "GitLab Personal Access Token" 입력 → "Scope"에 "Global" 입력 → "Token"에 Gitlab Personal Access Token 입력 → "ID"에 임의의 아이디 입력 → 생성

*** Personal Access Token은 Gitlab > User Setings > Access Tokens 에서 생성

New credentials

Kind

GitLab Personal Access Token

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Token

.....

ID ?

Gitlab-Credential

Description ?

Description

Create

4.1.2. Jenkins Item 생성

1. 좌측 메뉴 "새로운 Item" 클릭
2. "Enter an item name"에 임의의 Item 이름 입력 → "Pipeline" 선택 → 생성

Enter an item name

test-pipeline

» Required field

 **Freestyle project**
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

3. "General" → "Do not allow concurrent builds" 클릭
(빌드가 진행중이면 동시에 빌드를 진행하지 않게 한다)

☒ Do not allow concurrent builds

☐ Abort previous builds ?

4. "Build Triggers" → "Build when a change is pushed to GitLab" 클릭
(Webhook 설정 : GitLab 특정 브랜치 push 시 자동 빌드 + 배포 설정)
(해당 URL 복사 → Webhook 설정 시 사용)

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: ?

Enabled GitLab triggers

☒ Push Events ?

☐ Push Events in case of branch delete ?

☒ Opened Merge Request Events ?

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events ?

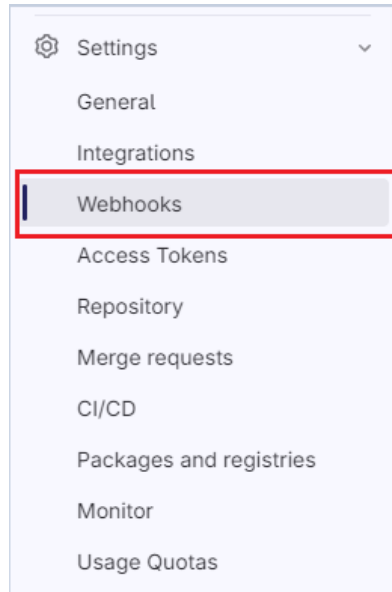
☐ Closed Merge Request Events ?

5. "Build when a change is pushed to GitLab" 하위의 "고급..." 클릭
6. "Secret token"의 "Generate" 클릭 후 생성된 토큰 값 복사 (Webhook 설정 시 사용)



4.1.3. Gitlab Webhook 설정

1. 프로젝트 GitLab → "Settings" → "Webhooks" 클릭



2. "URL"에 사전에 복사해놓은 Jenkins URL 입력 → "Secret token"에 사전에 복사해놓은 Secret token 입력 → "Push events" 클릭 후 Webhook을 적용할 브랜치 입력

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We reco

URL

URL must be percent-encoded if it contains one or more special characters.

☒ Show full URL

☐ Mask portions of URL

Do not show sensitive data such as tokens in the UI.

Secret token

Used to validate received payloads. Sent with the request in the `X-GitLab-Token` HTTP header.

Trigger

☒ Push events

☐ All branches

☒ Wildcard pattern

Wildcards such as `*-stable` or `production/*` are supported.

☐ Regular expression

4.1.4. Pipeline Script 작성

- pipeline-fe

```
pipeline {
  agent {
    kubernetes {
      yaml '''
        apiVersion: v1
        kind: Pod
        spec:
          containers:
            - name: docker
              image: docker:latest
              command:
                - cat
              tty: true
              volumeMounts:
                - mountPath: /var/run/docker.sock
                  name: docker-sock
          volumes:
            - name: docker-sock
              hostPath:
                path: /var/run/docker.sock
            ...
        }
      '''
    }
  }
  stages {
```

```

stage('Checkout') {
    steps {
        container('docker') {
            checkout scmGit(branches: [[name: '*/dev-fe']], extensions: [], userRemo
        }
    }
}

stage('Build-Docker-Image') {
    steps {
        dir('frontend'){
            container('docker') {
                sh 'docker build -t ko2sist/catale-fe:${BUILD_NUMBER} .'
            }
        }
    }
}

stage('Login-into-Docker') {
    steps {
        container('docker') {
            sh 'docker login -u {docker_ID} -p {docker_PW}'
        }
    }
}

stage('Push-Docker-Image') {
    steps {
        container('docker') {
            sh 'docker push ko2sist/catale-fe:${BUILD_NUMBER}'
        }
    }
}

stage('Modify-Helm-Chart'){
    steps {
        dir('gitops'){
            checkout scmGit(branches: [[name: '*/main']], extensions: [], user

            sh 'git checkout -b main'
            sh "sed -i 's/appVersion:.*\$/appVersion: ${BUILD_NUMBER}/g' front
            sh 'git config --global user.email "ko2sist@gmail.com"'
            sh 'git config --global user.name "ko2sist"'
            sh 'git config --global --add safe.directory /home/jenkins/agent/w
            sh "git add frontend/Chart.yaml"
            sh "git commit -m '[UPDATE] catale-fe ${BUILD_NUMBER} image versio

            withCredentials([gitUsernamePassword(credentialsId: 'github-token'
                sh "git push -u origin main"
            )
        }
    }
}

```

```

post {
  always {
    container('docker') {
      sh 'docker logout'
      sh 'docker rmi ko2sist/catale-fe:${BUILD_NUMBER}'
    }
  }
  success {
    script {
      def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
      def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
      mattermostSend (color: 'good',
      message: "FE 빌드 성공!: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
      endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmdtcjnf9ai9oci4a9r'
      channel: 'BOT'
      )
    }
  }
  failure {
    script {
      def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
      def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
      mattermostSend (color: 'danger',
      message: "FE 빌드 실패...: #${env.BUILD_NUMBER} by ${Author_ID}(${Author
      endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmdtcjnf9ai9oci4a9r'
      channel: 'BOT'
      )
    }
  }
}
}
}

```

- pipeline-be

```

pipeline {
  agent {
    kubernetes {
      yaml '''
      apiVersion: v1
      kind: Pod
      spec:
        containers:
        - name: docker
          image: docker:latest
          command:
            - cat
          tty: true
          volumeMounts:
            - mountPath: /var/run/docker.sock
              name: docker-sock
      '''
    }
  }
}

```

```

        volumes:
        - name: docker-sock
          hostPath:
            path: /var/run/docker.sock
        ...
    }
}
stages {
    stage('Checkout') {
        steps {
            container('docker') {
                checkout scmGit(branches: [[name: '*/dev-be']], extensions: [], userRemo

                dir('backend/src/main/resources'){
                    sh 'ls -a'
                    checkout scmGit(branches: [[name: '*/main']], extensions: [], userRe
                    sh 'ls -a'

                }
            }
        }
    }
}

stage('Build-Docker-Image') {
    steps {
        dir('backend'){
            sh 'chmod +x gradlew'
            sh './gradlew clean build'

            container('docker') {
                sh 'docker build -t ko2sist/catale-be:${BUILD_NUMBER} .'
            }
        }
    }
}

stage('Login-into-Docker') {
    steps {
        container('docker') {
            sh 'docker login -u {docker_ID} -p {docker_PW}'
        }
    }
}

stage('Push-Docker-Image') {
    steps {
        container('docker') {
            sh 'docker push ko2sist/catale-be:${BUILD_NUMBER}'
        }
    }
}

stage('Modify-Helm-Chart'){
    steps {

```

```

    dir('gitops'){
        checkout scmGit(branches: [[name: '*/main']], extensions: [], user

        sh 'git checkout -b main'
        sh "sed -i 's/appVersion:.*/appVersion: ${BUILD_NUMBER}/g' backe
        sh 'git config --global user.email "ko2sist@gmail.com"'
        sh 'git config --global user.name "ko2sist"'
        sh 'git config --global --add safe.directory /home/jenkins/agent/w
        sh "git add backend/Chart.yaml"
        sh "git commit -m '[UPDATE] catale-be ${BUILD_NUMBER} image versio

        withCredentials([gitUsernamePassword(credentialsId: 'github-token'
            sh "git push -u origin main"
        })
    }
}
}
}

post {
    always {
        container('docker') {
            sh 'docker logout'
            sh 'docker rmi ko2sist/catale-be:${BUILD_NUMBER}'
        }
    }
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'good',
                message: "BE 빌드 성공!: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
                endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmtdcjnf9ai9oci4a9r'
                channel: 'BOT'
            )
        }
    }
    failure {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'danger',
                message: "BE 빌드 실패..: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_
                endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmtdcjnf9ai9oci4a9r'
                channel: 'BOT'
            )
        }
    }
}
}
}

```


- pipeline-re

```

pipeline {
  agent {
    kubernetes {
      yaml '''
        apiVersion: v1
        kind: Pod
        spec:
          containers:
            - name: docker
              image: docker:latest
              command:
                - cat
              tty: true
              volumeMounts:
                - mountPath: /var/run/docker.sock
                  name: docker-sock
          volumes:
            - name: docker-sock
              hostPath:
                path: /var/run/docker.sock
      '''
    }
  }
  stages {
    stage('Checkout') {
      steps {
        container('docker') {
          checkout scmGit(branches: [[name: '*/dev-re']], extensions: [], userRemo

          dir('env'){
            checkout scmGit(branches: [[name: '*/main']], extensions: [], userRe
          }

          sh 'cp env/.env.* recommend-server'

          dir('recommend-server'){
            sh 'ls -a'
          }
        }
      }
    }
    stage('Build-Docker-Image') {
      steps {
        dir('recommend-server'){
          container('docker') {
            sh 'docker build -t ko2sist/catale-re:${BUILD_NUMBER} .'
          }
        }
      }
    }
  }
}

```

```

    }
  }
  stage('Login-into-Docker') {
    steps {
      container('docker') {
        sh 'docker login -u {docker_ID} -p {docker_PW}'
      }
    }
  }
  stage('Push-Docker-Image') {
    steps {
      container('docker') {
        sh 'docker push ko2sist/catale-re:${BUILD_NUMBER}'
      }
    }
  }
  stage('Modify-Helm-Chart'){
    steps {
      dir('gitops'){
        checkout scmGit(branches: [[name: '*/main']], extensions: [], user

        sh 'git checkout -b main'
        sh "sed -i 's/appVersion:.*\$/appVersion: ${BUILD_NUMBER}/g' recom
        sh 'git config --global user.email "ko2sist@gmail.com"'
        sh 'git config --global user.name "ko2sist"'
        sh 'git config --global --add safe.directory /home/jenkins/agent/w
        sh "git add recommend/Chart.yaml"
        sh "git commit -m '[UPDATE] catale-re ${BUILD_NUMBER} image versio

        withCredentials([gitUsernamePassword(credentialsId: 'github-token'
          sh "git push -u origin main"
        )]
      }
    }
  }
}

post {
  always {
    container('docker') {
      sh 'docker logout'
      sh 'docker rmi ko2sist/catale-re:${BUILD_NUMBER}'
    }
  }
  success {
    script {
      def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
      def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
      mattermostSend (color: 'good',
        message: "RE 빌드 성공!: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
        endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmdcjnf9ai9oci4a9r'

```

```

        channel: 'BOT'
      )
    }
  }
  failure {
    script {
      def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
      def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
      mattermostSend (color: 'danger',
      message: "RE 빌드 실패..: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_
      endpoint: 'https://meeting.ssafy.com/hooks/fst51fsfmdtcjnf9ai9oci4a9r'
      channel: 'BOT'
      )
    }
  }
}
}
}
}

```

4.2. argoCD 설정

4.2.1. helm chart 저장

- Chart.yaml

```

apiVersion: v2
name: backend
description: A Helm chart for Kubernetes

type: application

version: 0.1.0

appVersion: 85

```

- values.yaml

```

# Default values for backend.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

image:
  repository: ko2sist/catale-be
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  # tag: ""

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""

```

```

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""

podAnnotations: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #     - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 8080

ingress:
  enabled: true
  className: ""
  annotations:
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
    nginx.ingress.kubernetes.io/proxy-body-size: "5M"
    cert-manager.io/cluster-issuer: letsencrypt
  hosts:
    - host: api.silvstone.xyz
      paths:
        - path: /
          pathType: ImplementationSpecific
  tls:
    - secretName: be-ingress-tls
      hosts:
        - api.silvstone.xyz

resources: {}
  # We usually recommend not to specify default resources and to leave this as a c
  # choice for the user. This also increases chances charts run on environments wi
  # resources, such as Minikube. If you do want to specify resources, uncomment th
  # lines, adjust them as necessary, and remove the curly braces after 'resources:
  # limits:

```

```

#   cpu: 100m
#   memory: 128Mi
# requests:
#   cpu: 100m
#   memory: 128Mi

autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

nodeSelector: {}

tolerations: []

affinity: {}

```

- deployment.yaml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "backend.fullname" . }}
  labels:
    {{- include "backend.labels" . | nindent 4 }}
spec:
  {{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      {{- include "backend.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      {{- with .Values.podAnnotations }}
      annotations:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      labels:
        {{- include "backend.selectorLabels" . | nindent 8 }}
    spec:
      {{- with .Values.imagePullSecrets }}
      imagePullSecrets:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      serviceAccountName: {{ include "backend.serviceAccountName" . }}
      securityContext:
        {{- toYaml .Values.podSecurityContext | nindent 8 }}
      containers:

```

```

- name: {{ .Chart.Name }}
  securityContext:
    {{- toYaml .Values.securityContext | nindent 12 }}
  image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .C
  imagePullPolicy: {{ .Values.image.pullPolicy }}"
  ports:
    - name: http
      containerPort: 8080
      protocol: TCP
  livenessProbe:
    httpGet:
      path: /actuator/health/liveness
      port: 8080
      #initialDelaySeconds: '5'
      periodSeconds: 3
  readinessProbe:
    httpGet:
      path: /actuator/health/readiness
      port: 8080
      #initialDelaySeconds: '5'
      periodSeconds: 3
  resources:
    {{- toYaml .Values.resources | nindent 12 }}
{{- with .Values.nodeSelector }}
nodeSelector:
  {{- toYaml . | nindent 8 }}
{{- end }}
{{- with .Values.affinity }}
affinity:
  {{- toYaml . | nindent 8 }}
{{- end }}
{{- with .Values.tolerations }}
tolerations:
  {{- toYaml . | nindent 8 }}
{{- end }}

```

- service.yaml

```

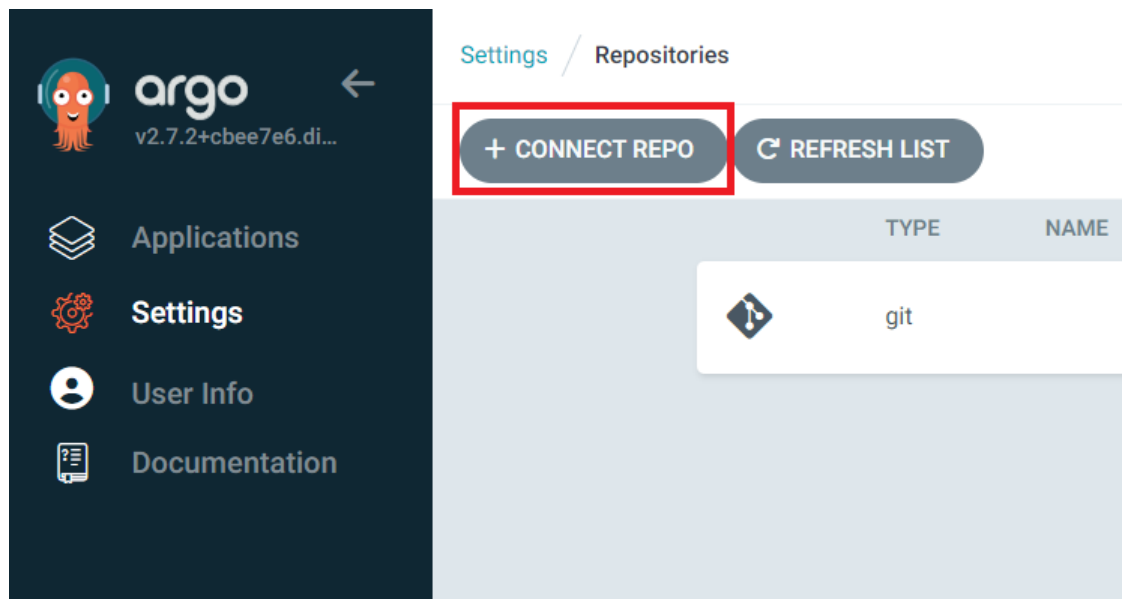
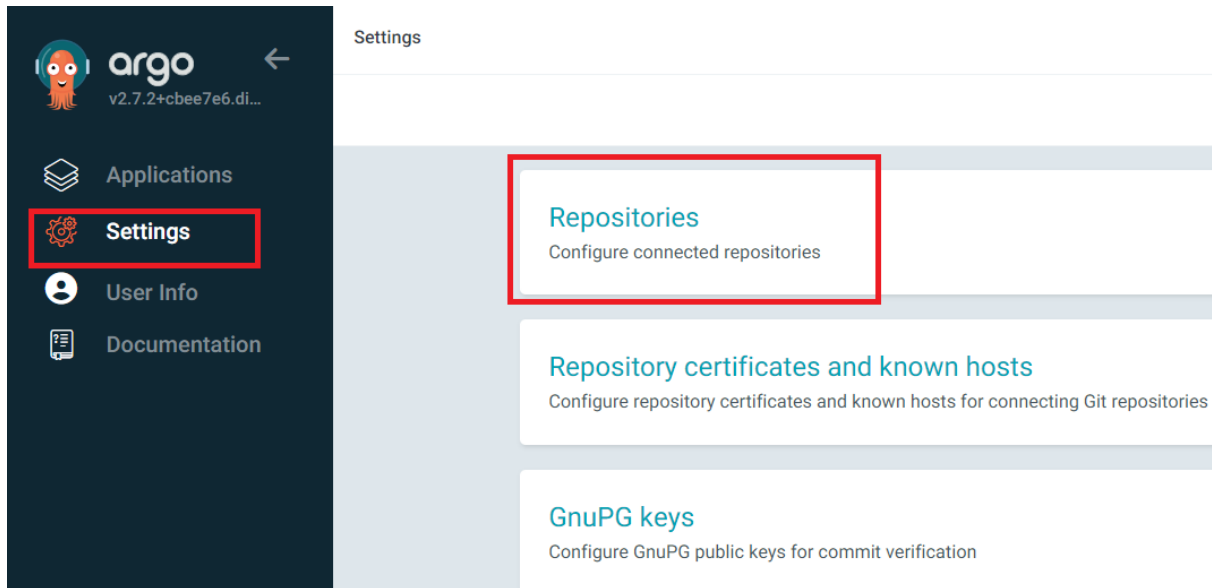
apiVersion: v1
kind: Service
metadata:
  name: {{ include "backend.fullname" . }}
  labels:
    {{- include "backend.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: 8080
      protocol: TCP
      name: http

```

```
selector:
  {{- include "backend.selectorLabels" . | nindent 4 }}
```

4.2.2. application 등록

- Git repository 등록



CONNECT
SAVE AS CREDENTIALS TEMPLATE
CANCEL

Choose your connection method:

VIA HTTPS ▾

CONNECT REPO USING HTTPS

Type

git

Project

default

Repository URL

Username (optional)

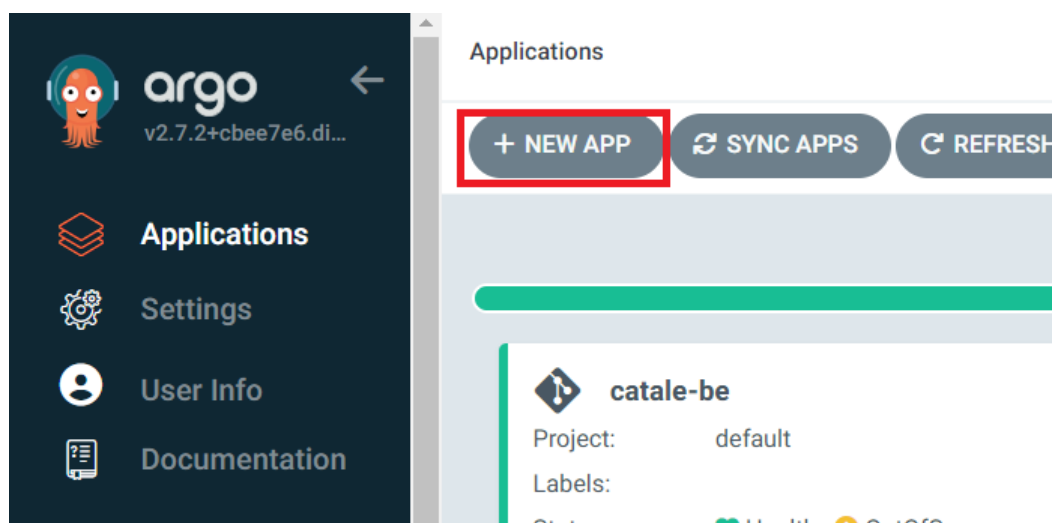
Password (optional)

Repository URL : git repo 주소 입력

Username : git id

Password : git personal access token

- app 등록



4.3 빌드 및 배포 과정

4.3.1. 빌드 시작

- | Option 1. 상기 Webhook 설정한 Branch로 Push
- | Option 2. Jenkins 홈 화면 → Jenkins Item 클릭 → “지금 빌드” 클릭

4.3.2. 빌드 / image 업로드

Jenkins를 통해 image 빌드 → docker hub에 업로드 → gitops 레포의 helm chart 수정

4.3.3. 배포

argoCD가 gitops repository의 변경 사항을 인식하여 새로운 이미지로 어플리케이션 배포