Dobie 포팅매뉴얼

1. 개발환경

1.1. Frontend

- Node.js 20.11.0 (LTS)
- React 18.2.0
    - Zustand 4.5.2
    - lottie-web 5.12.2
    - Material UI 5.15.17
    - sweetalert2 11.11.0
- Axios 1.6.8

1.2. Backend

- Java
  - Azul Zulu 17.0.9+8 (LTS)
  - Spring Boot 3.2.3
    - Spring Security 3.2.3
    - spring-cloud-starter-aws 2.2.6
    - JUnit 5.10.1
    - Lombok 1.18.30
  - Gradle 8.4

1.3. Server

- Ubuntu 20.04.6 LTS
- Docker 26.0.0
- microk8s 1.29
- Jenkins 2.440.1
- argoCD 2.7.2

1.4. UI/UX

- Figma

1.5. IDE

- IntelliJ IDEA 2023.2
- Visual Studio Code 1.87

1.6. SCM / Cooperation

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

1.7. Etc

- Postman 10.24.3

2. 환경 변수

2.1. Frontend

```
REACT_APP_SERVER
```

2.2. Backend

```
SPRING_PROFILES_INCLUDE
SPRING_SECURITY_USER_NAME
SPRING_SECURITY_USER_PASSWORD
SPRING_CORS_ALLOWD_ORIGINS
FILTER_PATHS
JWT_SECRET_KEY
JWT_ACCESS_TIME
JWT_REFRESH_TIME
MANAGEMENT_HEALTH_LIVENESSSTATE
MANAGEMENT_HEALTH_READINESSSTATE
```

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

sudo usermod -a -G microk8s ubuntu
newgrp microk8s

# 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-forwading (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. 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 namesapce 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
```

3.4. argoCD

3.4.1. 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
```

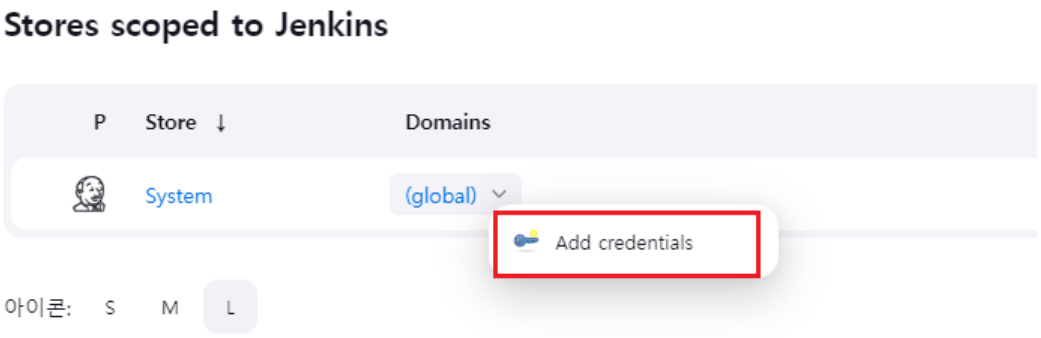
3.4.2. gitops repo 연동

4. CI/CD 구축

4.1. Jenkins 설정

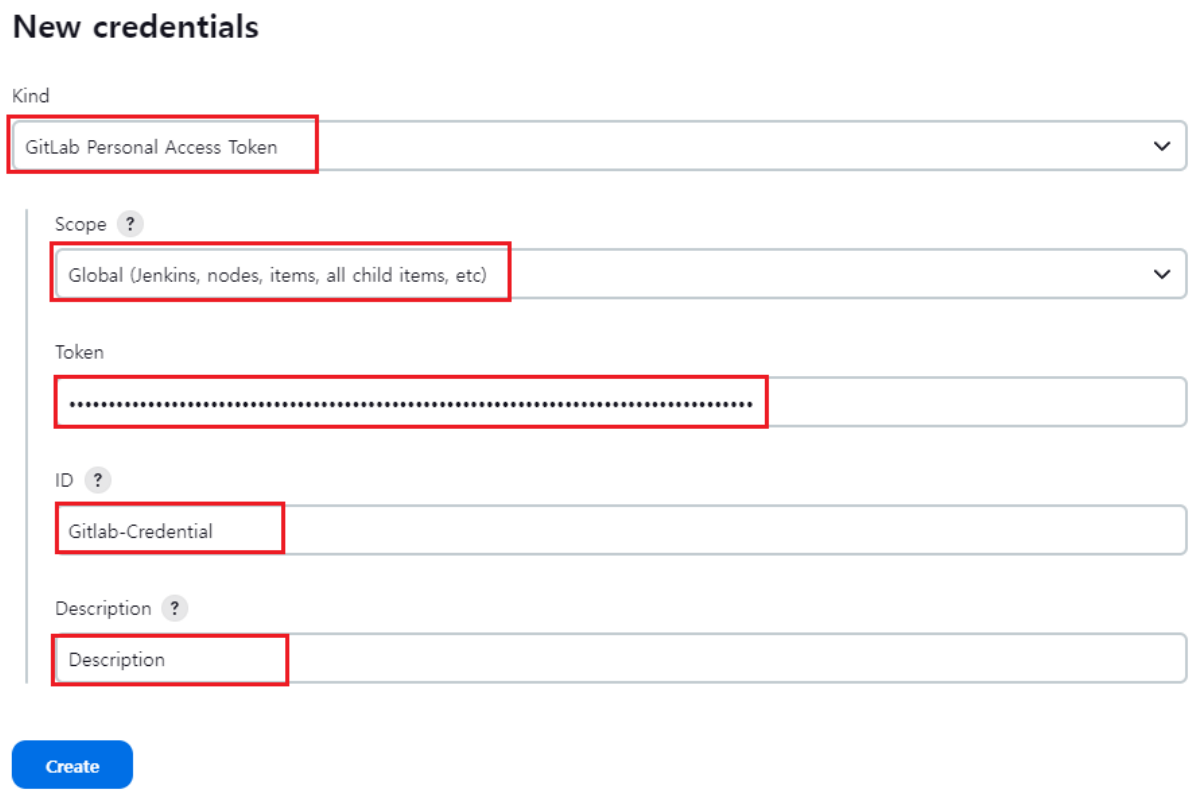
4.1.1. GitLab Credentials 설정

1. 좌측 메뉴 “Jenkins 관리” 클릭
2. Security → Credentials 클릭
3. “Store : System” → “(global)” → “Add vredentials” 클릭



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

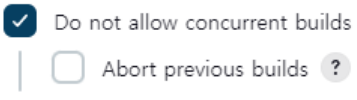
   *** Personal Access Token은 Gitlab > User Setings > Access Tokens 에서 생성



4.1.2. Jenkins Item 생성

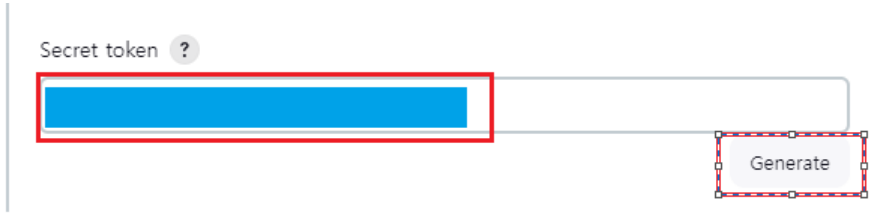
1. 좌측 메뉴 “새로운 Item” 클릭
2. “Enter an item name”에 임의의 Item 이름 입력 → “Pipeline” 선택 → 생성

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



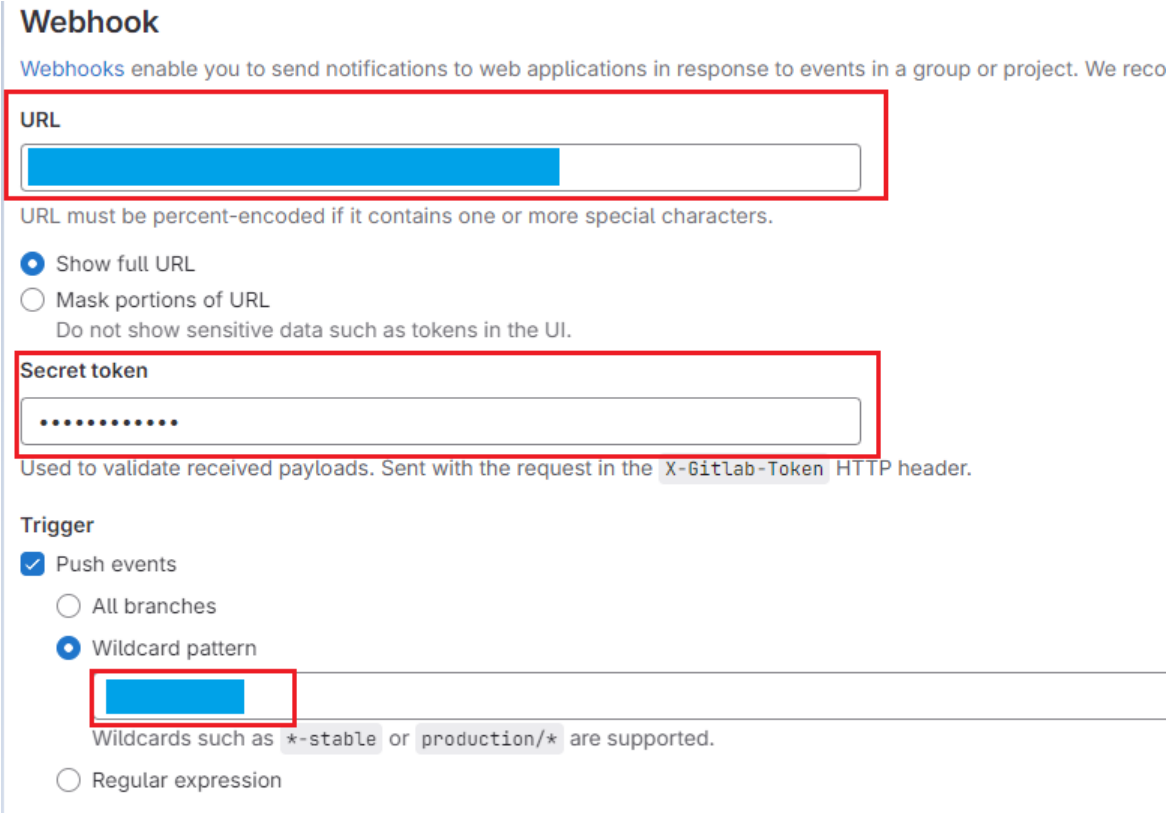
4. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭
   (Webhook 설정 : GitLab 특정 브랜치 push 시 자동 빌드 + 배포 설정)
   (해당 URL 복사 → Webhook 설정 시 사용)

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 서전에 복사해놓은 Secret token 입력 → “Push events” 클릭 후 Webhook을 적용할 브랜치 입력



4.1.4. Pipeline Script 작성

- dobie-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

        //   dir('backend/src/main/resources'){
        //       checkout scmGit(branches: [[name: '*/main']], extensions: [], use
        //   }
        }
      }
    }

    stage('Build-Docker-Image') {
      steps {
          dir('frontend'){

            container('docker') {
                sh 'docker build -t ko2sist/dobie-fe:${BUILD_NUMBER} .'
                sh 'docker build -t ko2sist/dobie-fe:latest .'
            }
          }
      }
    }
    stage('Login-into-Docker') {
      steps {
        container('docker') {
          sh 'docker login -u ${docker_ID} -p ${docker_PW}'
        }
      }
    }
    stage('Push-Docker-Image') {
      steps {
        container('docker') {
          sh 'docker push ko2sist/dobie-fe:${BUILD_NUMBER}'
          sh 'docker push ko2sist/dobie-fe:latest'
        }
      }
    }
    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 "eunnseok.ko@gmail.com"'
                sh 'git config --global user.name "eunnseok"'
                sh 'git config --global --add safe.directory /home/jenkins/agent/w
                sh "git add frontend/Chart.yaml"
                sh "git commit -m '[UPDATE] dobie-fe ${BUILD_NUMBER} image version
```

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

  post {
    always {
      container('docker') {
        sh 'docker logout'
        sh 'docker rmi ko2sist/dobie-fe:${BUILD_NUMBER}'
        sh 'docker rmi ko2sist/dobie-fe:latest'
      }
    }
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'good',
            message: "FE 빌드 성공!: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
            endpoint: 'https://meeting.ssafy.com/hooks/3ej9b8i4atd7uknnwstay5qg3h'
            channel: 'dobie-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/3ej9b8i4atd7uknnwstay5qg3h'
            channel: 'dobie-bot'
            )
        }
    }
  }
}
```

- dobie-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'){
        //       checkout scmGit(branches: [[name: '*/main']], extensions: [], use
        //   }
        }
      }
    }

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

            container('docker') {
                sh 'docker build -t ko2sist/dobie-be:${BUILD_NUMBER} .'
                sh 'docker build -t ko2sist/dobie-be:latest .'
            }
          }
      }
    }
    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/dobie-be:${BUILD_NUMBER}'
          sh 'docker push ko2sist/dobie-be:latest'
        }
      }
    }
    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 "eunnseok.ko@gmail.com"'
                sh 'git config --global user.name "eunnseok"'
                sh 'git config --global --add safe.directory /home/jenkins/agent/w
                sh "git add backend/Chart.yaml"
                sh "git commit -m '[UPDATE] dobie-be ${BUILD_NUMBER} image version

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

post {
    always {
      container('docker') {
        sh 'docker logout'
        sh 'docker rmi ko2sist/dobie-be:${BUILD_NUMBER}'
        sh 'docker rmi ko2sist/dobie-be:latest'
      }
    }
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'good',
            message: "BE 빌드 성공!: #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
            endpoint: 'https://meeting.ssafy.com/hooks/3ej9b8i4atd7uknnwstay5qg3h'
            channel: 'dobie-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/3ej9b8i4atd7uknnwstay5qg3h'
            channel: 'dobie-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: 237
```

- 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/dobie-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 }}
          volumeMounts:
          - mountPath: /data
            name: json-data

      volumes:
      - hostPath:
          path: /var/dobie/data
          type: Directory
        name: json-data
      {{- with .Values.nodeSelector }}
      nodeSelector:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      {{- with .Values.affinity }}
      affinity:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      {{- with .Values.tolerations }}
      tolerations:
        {{- toYaml . | nindent 8 }}
      {{- end }}
```

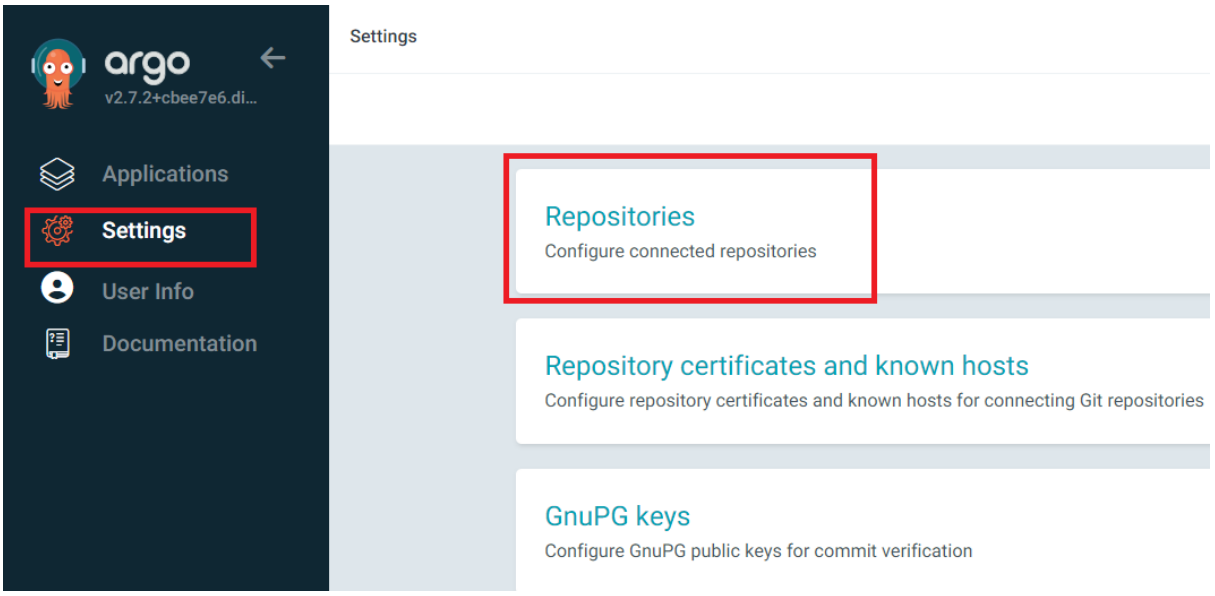
- sevice.yaml

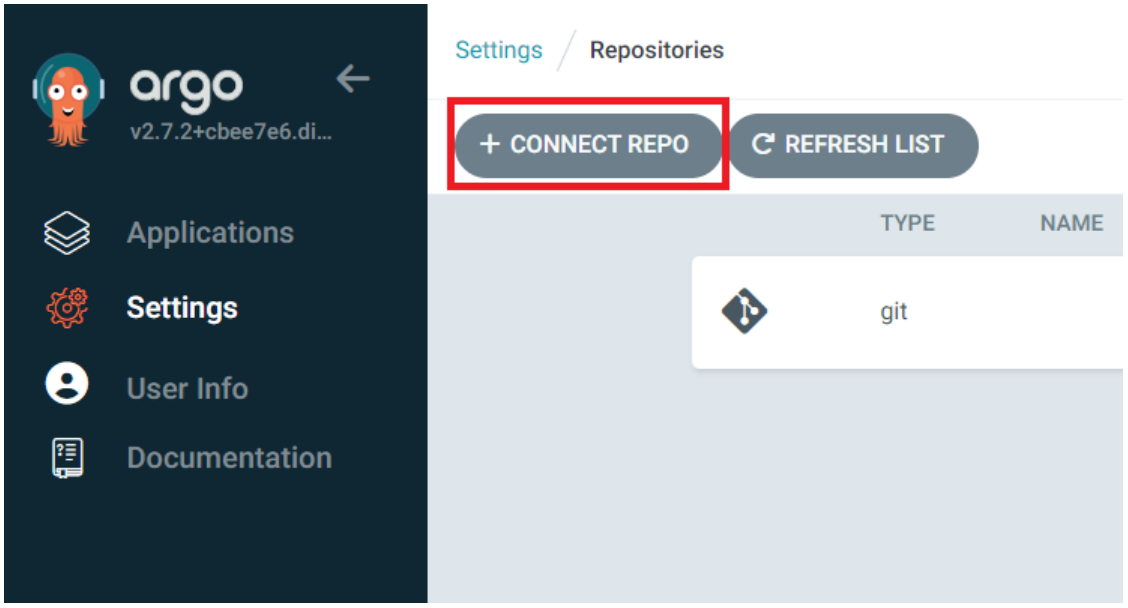
```
apiVersion: v1
kind: Service
metadata:
  name: {{ include "backend.fullname" . }}
  labels:
    {{- include "backend.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: 8080
      protocol: TCP
      name: http
```

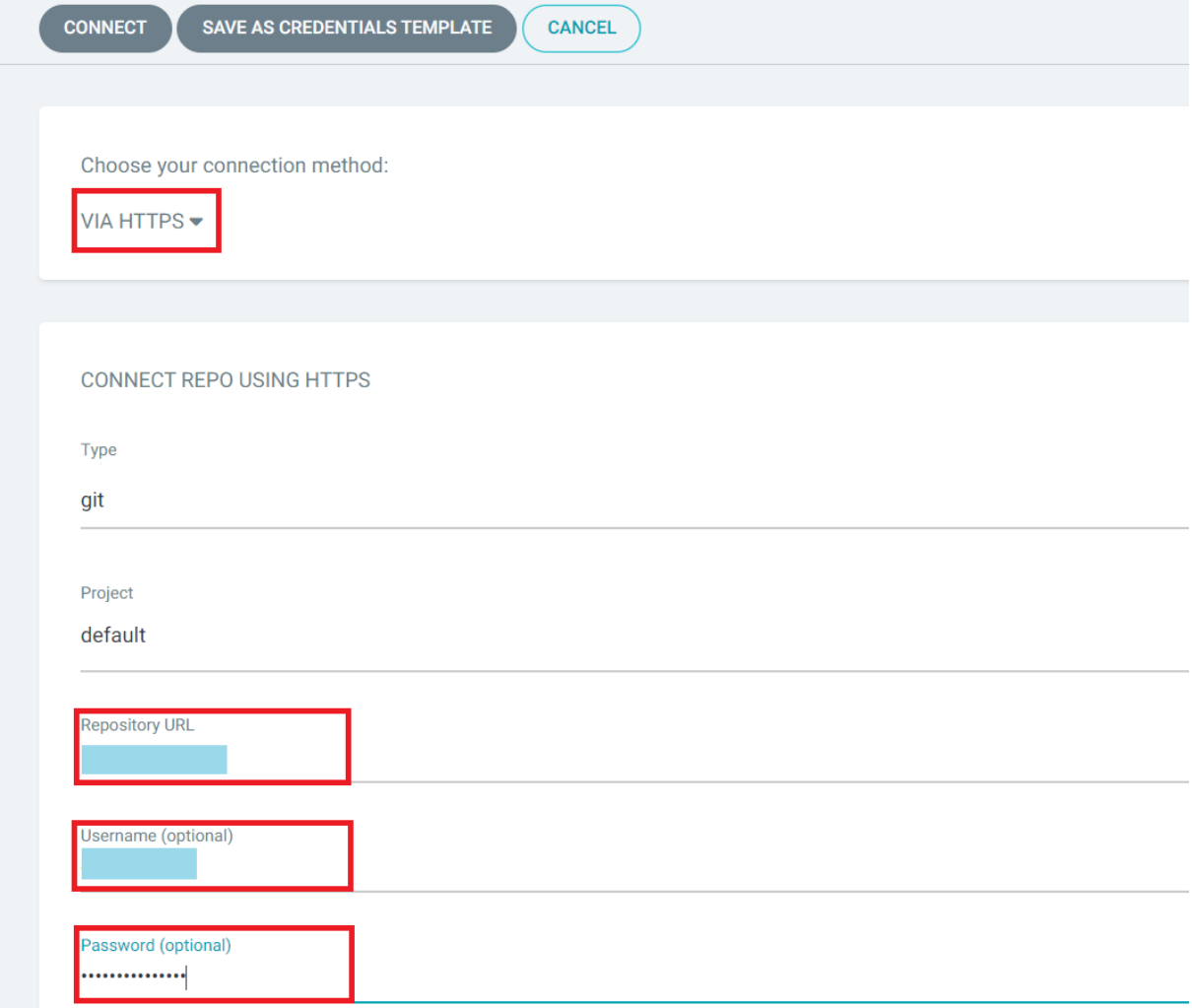
```
  selector:
    {{- include "backend.selectorLabels" . | nindent 4 }}
```

4.2.2. application 등록

- Git repository 등록





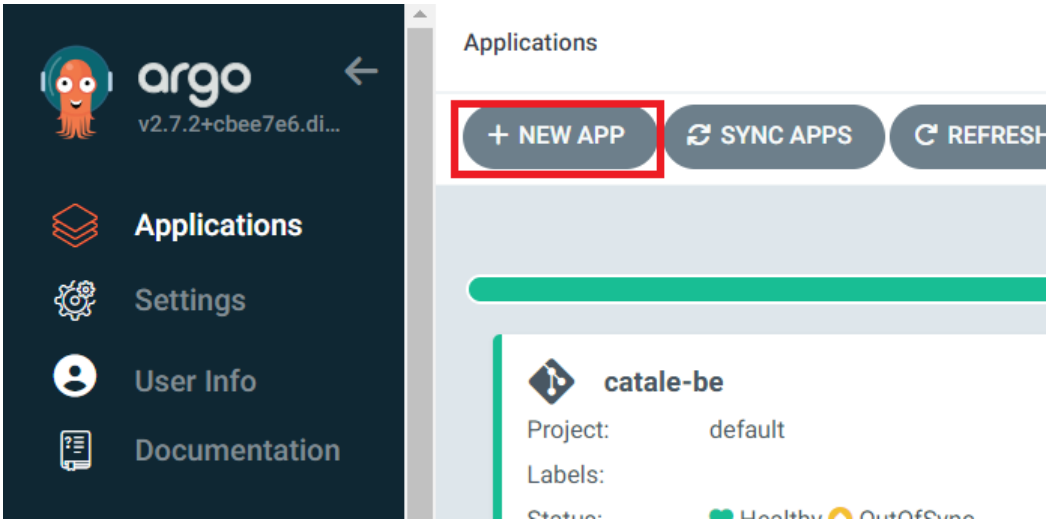


Repository URL : git repo 주소 입력

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

5. Dobie Install

5.1. Shell Script 다운로드 및 실행

Dobie를 설치, 실행하기 위해 다음 명령어를 실행하세요

```
# shell script 다운로드
wget https://raw.githubusercontent.com/eunnseok/dobie-deploy/main/install-dobie.sh

# shell script 실행 권한 부여
chmod +x [install-dobie.sh](http://install-dobie.sh/)

# shell script 실행
sh install-dobie
```

5.2. 관리페이지 접속

```
http://{사용자 Server Public IP or 도메인}:3333
```