

Constelink

PORTING MANUAL

목 차

I. 개발환경	2
1. 프로젝트 기술 스택	2
2. 설정 파일 목록과 프로젝트내 경로	3
공통:	3
Frontend:	3
AuthServer:	3
ConstelinkBeneficiary:	3
ConstelinkFundraising:	4
ConstelinkMember:	4
ConstelinkNotice:	4
ConstelinkFile:	4
3. Kubernetes Manifest	4
4. 설정 파일 및 환경 변수 정보	20
공통:	20
Frontend:	27
AuthServer:	27
ConstelinkBeneficiary:	28
ConstelinkFundraising:	29
ConstelinkMember:	29
ConstelinkNotice:	32
ConstelinkFile:	33
II. 빌드 및 배포	34
1. Podman 설치	34
2. SSL 인증서 발급	35
3. Kubernetes 설치	35
4. Kubernetes 세팅	38
5. DB 배포	43
6. Jenkins CI	43
7. Argo CD	46
III. 외부 서비스	51
1. Google Cloud Storage	51

I. 개발환경

1. 프로젝트 기술 스택

Server : AWS EC2 Ubuntu 20.04 LTS

Visual Studio Code : 1.75.1

IntelliJ IDEA : 2022.3.1 (Ultimate Edition) 17.0.5+1-b653.23 amd64

JVM : OpenJDK 17

Spring Boot: 3.0.4

Gradle: 7.6.1

gRPC: 1.52.1

Node.js : 18.15.0

TypeScript: 4.9.5

React: 18.2.0

Redux: 1.9.3

MariaDB : 10.11.2

Redis : 7.0.10

Kubernetes: 1.26.2

CRI-O: 1.26.1

Nginx Ingress Controller: v1.6.4

Jenkins : 2.397

ArgoCD: v2.6.7+5bcd846

2. 설정 파일 목록과 프로젝트내 경로

공통:

- **Jenkinsfile** : /

Frontend:

- **Dockerfile** : /Frontend
- **.env** : /Frontend

AuthServer:

- **Dockerfile** : /Backend/AuthServer
- **application.yml** : /Backend/AuthServer/src/main/resources

ConstelinkBeneficiary:

- **Dockerfile** : /Backend/ConstelinkBeneficiary
- **application.yml** : /Backend/ConstelinkBeneficiary/src/main/resources

ConstelinkFundraising:

- **Dockerfile** : /Backend/ConstelinkFundraising
- **application.yml** : /Backend/ConstelinkFundraising/src/main/resources

ConstelinkMember:

- **Dockerfile** : /Backend/ConstelinkMember
- **application.yml** : /Backend/ConstelinkMember/src/main/resources

ConstelinkNotice:

- **Dockerfile** : /Backend/ConstelinkNotice
- **application.yml** : /Backend/ConstelinkNotice/src/main/resources

ConstelinkFile:

- **Dockerfile** : /Backend/ConstelinkFile
- **application.yml** : /Backend/ConstelinkFile/src/main/resources

3. Kubernetes Manifest

jenkins-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      securityContext:
        fsGroup: 0
        runAsUser: 0
      containers:
        - name: jenkins
          image: jenkins/jenkins:jdk17
          securityContext:
            privileged: true
          env:
            - name: TZ
              value: Asia/Seoul
          ports:
            - containerPort: 8080
            - containerPort: 50000
          volumeMounts:
            - name: jenkins-home
              mountPath: /var/jenkins_home
      nodeSelector:
        node-role.kubernetes.io/control-plane: ""
      volumes:
```

```
- name: jenkins-home
  hostPath:
    path: /var/jenkins_home
    type: Directory
```

ingress-nginx.yaml:

```
curl -o ingress-nginx.yaml
https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.6.4/deploy/static/provider/baremetal/deploy.yaml

# NodePort에 80, 443 포트 할당
vi ingress-nginx.yaml
```

```
# Service 부분에 nodePort 추가
~~~
ports:
  - appProtocol: http
    name: http
    port: 80
    protocol: TCP
    targetPort: http
    nodePort: 80
  - appProtocol: https
    name: https
    port: 443
    protocol: TCP
    targetPort: https
    nodePort: 443
~~~

# 마스터노드에 생성하려면 nodeSelector에 다음 내용 추가
~~~
spec:
  nodeSelector:
    node-role.kubernetes.io/control-plane: ""
~~~
```

jenkins-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
spec:
  selector:
    app: jenkins
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 8093
  type: NodePort
```

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins-jnlp
spec:
  selector:
    app: jenkins
  ports:
    - protocol: TCP
      port: 50000
      targetPort: 50000
      nodePort: 50000
  type: NodePort
```

mariadb-beneficiary.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb-beneficiary
spec:
  replicas: 1
```

```

selector:
  matchLabels:
    app: mariadb-beneficiary
template:
  metadata:
    labels:
      app: mariadb-beneficiary
  spec:
    nodeSelector:
      node-role.kubernetes.io/control-plane: ""
    containers:
      - name: mariadb
        image: mariadb:latest
        env:
          - name: MYSQL_ROOT_PASSWORD
            value: "루트 비밀번호"
          - name: MYSQL_USER
            value: "유저 이름"
          - name: MYSQL_PASSWORD
            value: "유저 비밀번호"
          - name: MYSQL_USER_HOST
            value: "%"
        ports:
          - containerPort: 3306
        volumeMounts:
          - name: mariadb-data
            mountPath: /var/lib/mysql
    volumes:
      - name: mariadb-data
        hostPath:
          path: /home/ubuntu/mariadb/databases-beneficiary
          type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: mariadb-beneficiary
spec:
  selector:
    app: mariadb-beneficiary
  ports:
    - protocol: TCP

```



```
port: 3306
targetPort: 3306
nodePort: 3325
type: NodePort
```

mariadb-fundraising.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb-fundraising
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mariadb-fundraising
  template:
    metadata:
      labels:
        app: mariadb-fundraising
    spec:
      nodeSelector:
        node-role.kubernetes.io/control-plane: ""
      containers:
        - name: mariadb-fundraising
          image: mariadb:latest
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "루트 비밀번호"
            - name: MYSQL_USER
              value: "유저 이름"
            - name: MYSQL_PASSWORD
              value: "유저 비밀번호"
            - name: MYSQL_USER_HOST
              value: "%"
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mariadb-data
              mountPath: /var/lib/mysql
```

```

    volumes:
      - name: mariadb-data
        hostPath:
          path: /home/ubuntu/mariadb/databases-fundraising
          type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: mariadb-fundraising
spec:
  selector:
    app: mariadb-fundraising
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
      nodePort: 3326
  type: NodePort

```

mariadb-member.yaml:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb-member
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mariadb-member
  template:
    metadata:
      labels:
        app: mariadb-member
    spec:
      nodeSelector:
        node-role.kubernetes.io/control-plane: ""
      containers:
        - name: mariadb-member

```

```

    image: mariadb:latest
    env:
      - name: MYSQL_ROOT_PASSWORD
        value: "루트 비밀번호"
      - name: MYSQL_USER
        value: "유저 이름"
      - name: MYSQL_PASSWORD
        value: "유저 비밀번호"
      - name: MYSQL_USER_HOST
        value: "%"
    ports:
      - containerPort: 3306
    volumeMounts:
      - name: mariadb-data
        mountPath: /var/lib/mysql
  volumes:
    - name: mariadb-data
      hostPath:
        path: /home/ubuntu/mariadb/databases-member
        type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: mariadb-member
spec:
  selector:
    app: mariadb-member
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
      nodePort: 3324
  type: NodePort

```

mariadb-notice.yaml:

```

apiVersion: apps/v1
kind: Deployment
metadata:

```

```

  name: mariadb-notice
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mariadb-notice
  template:
    metadata:
      labels:
        app: mariadb-notice
    spec:
      nodeSelector:
        node-role.kubernetes.io/control-plane: ""
      containers:
        - name: mariadb-notice
          image: mariadb:latest
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "루트 비밀번호"
            - name: MYSQL_USER
              value: "유저 이름"
            - name: MYSQL_PASSWORD
              value: "유저 비밀번호"
            - name: MYSQL_USER_HOST
              value: "%"
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mariadb-data
              mountPath: /var/lib/mysql
      volumes:
        - name: mariadb-data
          hostPath:
            path: /home/ubuntu/mariadb/databases-notice
            type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: mariadb-notice
spec:
  selector:

```

```
app: mariadb-notice
ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
    nodePort: 3327
type: NodePort
```

redis.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      nodeSelector:
        node-role.kubernetes.io/control-plane: ""
      containers:
        - name: redis
          image: redis:latest
          args: ["--requirepass", "비밀번호"]
          ports:
            - containerPort: 6379
          volumeMounts:
            - name: redis-data
              mountPath: /data
      volumes:
        - name: redis-data
          hostPath:
            path: /home/ubuntu/redis/data
            type: Directory
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
      nodePort: 6379
  type: NodePort
```

constelink-front.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: docker.io/sadoruin/constelink-front:v1
          ports:
            - containerPort: 3000
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: frontend
  namespace: default
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: ClusterIP
```

constelink-authserver.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-authserver
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-authserver
  template:
    metadata:
      labels:
        app: constelink-authserver
    spec:
      containers:
        - name: constelink-authserver
          image: docker.io/sadoruin/constelink-authserver:v1
          ports:
            - containerPort: 8080
  ---
apiVersion: v1
kind: Service
metadata:
```

```
name: constelink-authserver
namespace: default
spec:
  selector:
    app: constelink-authserver
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
  type: ClusterIP
```

constelink-beneficiary.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-beneficiary
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-beneficiary
  template:
    metadata:
      labels:
        app: constelink-beneficiary
    spec:
      containers:
        - name: constelink-beneficiary
          image: docker.io/sadoruin/constelink-beneficiary:v1
          ports:
            - containerPort: 8080
            - containerPort: 9090
---
apiVersion: v1
kind: Service
metadata:
  name: constelink-beneficiary
```



```
namespace: default
spec:
  selector:
    app: constelink-beneficiary
  ports:
    - name: springboot
      protocol: TCP
      port: 8080
      targetPort: 8080
    - name: grpc
      protocol: TCP
      port: 9090
      targetPort: 9090
  type: ClusterIP
```

constelink-file.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-file
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-file
  template:
    metadata:
      labels:
        app: constelink-file
    spec:
      containers:
        - name: constelink-file
          image: docker.io/sadoruin/constelink-file:v1
          ports:
            - containerPort: 8080
  ---
apiVersion: v1
```

```
kind: Service
metadata:
  name: constelink-file
  namespace: default
spec:
  selector:
    app: constelink-file
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
  type: ClusterIP
```

constelink-fundraising.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-fundraising
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-fundraising
  template:
    metadata:
      labels:
        app: constelink-fundraising
    spec:
      containers:
        - name: constelink-fundraising
          image: docker.io/sadoruin/constelink-fundraising:v1
          ports:
            - containerPort: 8080
  ---
apiVersion: v1
kind: Service
metadata:
```

```
name: constelink-fundraising
namespace: default
spec:
  selector:
    app: constelink-fundraising
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
  type: ClusterIP
```

constelink-member.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-member
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-member
  template:
    metadata:
      labels:
        app: constelink-member
    spec:
      containers:
        - name: constelink-member
          image: docker.io/sadoruin/constelink-member:v1
          ports:
            - containerPort: 8080
  ---
apiVersion: v1
kind: Service
metadata:
  name: constelink-member
  namespace: default
```

```
spec:
  selector:
    app: constelink-member
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
  type: ClusterIP
```

constelink-notice.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: constelink-notice
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: constelink-notice
  template:
    metadata:
      labels:
        app: constelink-notice
    spec:
      containers:
        - name: constelink-notice
          image: docker.io/sadoruin/constelink-notice:v1
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: constelink-notice
  namespace: default
spec:
  selector:
```

```
app: constelink-notice
ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
type: ClusterIP
```

4. 설정 파일 및 환경 변수 정보

공통:

- Jenkinsfile :

```
pipeline {
  agent any

  tools {
    nodejs "nodejs"
  }

  stages {
    stage('Project Build') {
      steps {
        script {
          if(env.BRANCH_NAME == 'dev-front') {
            echo "Front Project Build Step"
          } else if(env.BRANCH_NAME ==
'feature-back/auth-server') {
            echo "Auth Server Project Build Step"
            dir('Backend/AuthServer') {
              sh 'chmod +x gradlew'
              sh './gradlew clean build -x test'
            }
          } else if(env.BRANCH_NAME == 'feature-back/member') {
            echo "ConstelinkMember Project Build Step"
            dir('Backend/ConstelinkMember') {
              sh 'chmod +x gradlew'
              sh './gradlew clean build -x test'
            }
          } else if(env.BRANCH_NAME ==
```

```

'feature-back/beneficiary') {
    echo "ConstelinkBeneficiary Project Build Step"
    dir('Backend/ConstelinkBeneficiary') {
        sh 'chmod +x gradlew'
        sh './gradlew clean build -x test'
    }
} else if(env.BRANCH_NAME ==
'feature-back/fundraising') {
    echo "ConstelinkFundraising Project Build Step"
    dir('Backend/ConstelinkFundraising') {
        sh 'chmod +x gradlew'
        sh './gradlew clean build -x test'
    }
} else if(env.BRANCH_NAME == 'feature-back/file') {
    echo "ConstelinkFile Project Build Step"
    dir('Backend/ConstelinkFile') {
        sh 'chmod +x gradlew'
        sh './gradlew clean build -x test'
    }
} else if(env.BRANCH_NAME == 'feature-back/notice') {
    echo "ConstelinkNotice Project Build Step"
    dir('Backend/ConstelinkNotice') {
        sh 'chmod +x gradlew'
        sh './gradlew clean build -x test'
    }
}
}
}
}
stage('Image Build') {
    environment {
        PATH = "/busybox:/kaniko:$PATH"
    }
    steps {
        script {
            podTemplate(yaml: """
                kind: Pod
                metadata:
                    name: kaniko
                spec:
                    containers:
                        - name: kaniko

```

```

        image: gcr.io/kaniko-project/executor:debug
        imagePullPolicy: Always
        command:
        - sleep
        args:
        - 99d
        volumeMounts:
        - name: shared-workspace
          mountPath: /workspace
        - name: docker-config
          mountPath: /kaniko/.docker
        tty: true
    nodeSelector:
        node-role.kubernetes.io/control-plane: ""
    volumes:
    - name: shared-workspace
      hostPath:
        path: ${WORKSPACE}
        type: Directory
    - name: docker-config
      secret:
        secretName: regcred
        items:
        - key: .dockerconfigjson
          path: config.json
    """) {
    node(POD_LABEL) {
        container(name: 'kaniko', shell: '/busybox/sh')
    }

    if(env.BRANCH_NAME == 'dev-front') {
        echo "Front Image Build Step"
        sh ""#!/busybox/sh
        /kaniko/executor
--context=/workspace/Frontend --dockerfile=/workspace/Frontend/Dockerfile
--destination=sadoruin/constelink-front:${env.BUILD_NUMBER}
        ""
    } else if(env.BRANCH_NAME ==
'feature-back/auth-server') {
        echo "Auth Server Image Build Step"
        sh ""#!/busybox/sh
        /kaniko/executor
--context=/workspace/Backend/AuthServer

```

```

--dockerfile=/workspace/Backend/AuthServer/Dockerfile
--destination=sadoruin/constelink-authserver:${env.BUILD_NUMBER}
    ""
} else if(env.BRANCH_NAME ==
'feature-back/member') {
    echo "ConstelinkMember Image Build
Step"
    sh ""#!/busybox/sh
    /kaniko/executor
--context=/workspace/Backend/ConstelinkMember
--dockerfile=/workspace/Backend/ConstelinkMember/Dockerfile
--destination=sadoruin/constelink-member:${env.BUILD_NUMBER}
    ""
} else if(env.BRANCH_NAME ==
'feature-back/beneficiary') {
    echo "ConstelinkBeneficiary Image Build
Step"
    sh ""#!/busybox/sh
    /kaniko/executor
--context=/workspace/Backend/ConstelinkBeneficiary
--dockerfile=/workspace/Backend/ConstelinkBeneficiary/Dockerfile
--destination=sadoruin/constelink-beneficiary:${env.BUILD_NUMBER}
    ""
} else if(env.BRANCH_NAME ==
'feature-back/fundraising') {
    echo "ConstelinkFundraising Image Build
Step"
    sh ""#!/busybox/sh
    /kaniko/executor
--context=/workspace/Backend/ConstelinkFundraising
--dockerfile=/workspace/Backend/ConstelinkFundraising/Dockerfile
--destination=sadoruin/constelink-fundraising:${env.BUILD_NUMBER}
    ""
} else if(env.BRANCH_NAME ==
'feature-back/file') {
    echo "ConstelinkFile Image Build Step"
    sh ""#!/busybox/sh
    /kaniko/executor
--context=/workspace/Backend/ConstelinkFile
--dockerfile=/workspace/Backend/ConstelinkFile/Dockerfile
--destination=sadoruin/constelink-file:${env.BUILD_NUMBER}
    ""

```



```

        } else if(env.BRANCH_NAME ==
'feature-back/notice') {
            echo "ConstelinkNotice Image Build
Step"
            sh """#!/busybox/sh
            /kaniko/executor
--context=/workspace/Backend/ConstelinkNotice
--dockerfile=/workspace/Backend/ConstelinkNotice/Dockerfile
--destination=sadoruin/constelink-notice:${env.BUILD_NUMBER}
            """
        }
    }
}
stage('Deploy') {
    steps {
        script {
            dir('/git') {
                git branch: 'main',
                    credentialsId: 'gitlab-account',
                    url: 'Manifest 레포지토리 주소'
                sh 'git config --global user.email "이메일"'
                sh 'git config --global user.name "이름"'

                if(env.BRANCH_NAME == 'dev-front') {
                    echo "Front Deploy Step"
                    sh """
                        sed -i
's/constelink-front:\([^:]*\)/constelink-front:${env.BUILD_NUMBER}/g'
manifests/constelink-front.yaml
                        git add manifests/constelink-front.yaml
                        git commit -m 'Update constelink-front tag
to ${env.BUILD_NUMBER}'
                        """
                } else if(env.BRANCH_NAME ==
'feature-back/auth-server') {
                    echo "Auth Server Deploy Step"
                    sh """
                        sed -i

```

```

's/constelink-authserver:\\([[:^:]]*\\)/constelink-authserver:${env.BUILD_NUMB
ER}/g' manifests/constelink-authserver.yaml
        git add
manifests/constelink-authserver.yaml
        git commit -m 'Update constelink-authserver
tag to ${env.BUILD_NUMBER}'
        ""
    } else if(env.BRANCH_NAME == 'feature-back/member')
{
        echo "ConstelinkMember Deploy Step"
        sh ""
        sed -i
's/constelink-member:\\([[:^:]]*\\)/constelink-member:${env.BUILD_NUMBER}/g'
manifests/constelink-member.yaml
        git add manifests/constelink-member.yaml
        git commit -m 'Update constelink-member tag
to ${env.BUILD_NUMBER}'
        ""
    } else if(env.BRANCH_NAME ==
'feature-back/beneficiary') {
        echo "ConstelinkBeneficiary Deploy Step"
        sh ""
        sed -i
's/constelink-beneficiary:\\([[:^:]]*\\)/constelink-beneficiary:${env.BUILD_NU
MBER}/g' manifests/constelink-beneficiary.yaml
        git add
manifests/constelink-beneficiary.yaml
        git commit -m 'Update
constelink-beneficiary tag to ${env.BUILD_NUMBER}'
        ""
    } else if(env.BRANCH_NAME ==
'feature-back/fundraising') {
        echo "ConstelinkFundraising Deploy Step"
        sh ""
        sed -i
's/constelink-fundraising:\\([[:^:]]*\\)/constelink-fundraising:${env.BUILD_NU
MBER}/g' manifests/constelink-fundraising.yaml
        git add
manifests/constelink-fundraising.yaml
        git commit -m 'Update
constelink-fundraising tag to ${env.BUILD_NUMBER}'
        ""
    }
}

```

- **Dockerfile :**

```
FROM node:18.15-alpine
WORKDIR /app
COPY ./package* ./
RUN npm install
COPY ./ ./
EXPOSE 3000
CMD ["npm", "run", "start"]
```

- **.env :**

```
REACT_APP_MM_PRIVATE_KEY="MetaMask 컨트랙트 계정 Private Key"
WDS_SOCKET_PORT=0
```

AuthServer:

- **Dockerfile :**

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- **application.yml :**

```
spring:
  data:
    redis:
      host: redis
      port: 6379
      password: 비밀번호

  jwt:
    secret: 비밀키값
```

ConstelinkBeneficiary:

- Dockerfile :

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080 9090
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- application.yml :

```
spring:
  datasource:
    url: jdbc:mariadb://mariadb-beneficiary:3306/constelink_beneficiary
    driver-class-name: org.mariadb.jdbc.Driver
    username: 계정명
    password: 비밀번호
  jpa:
    open-in-view: false
    generate-ddl: true
    show-sql: true
    hibernate:
      ddl-auto: none
```

ConstelinkFundraising:

- Dockerfile :

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- application.yml :

```
spring:
  datasource:
    url: jdbc:mariadb://mariadb-fundraising:3306/constelink_fundraising
    driver-class-name: org.mariadb.jdbc.Driver
    username: 계정명
    password: 비밀번호
  jpa:
    open-in-view: false
    generate-ddl: true
    show-sql: true
    hibernate:
      ddl-auto: none
```

ConstelinkMember:

- Dockerfile :

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- application.yml :

```
springdoc:
  version: v1.0.0
  api-docs:
    path: /api-docs
  default-consumes-media-type: application/json
  default-produces-media-type: application/json
  swagger-ui:
    operations-sorter: alpha
    tags-sorter: alpha
    path: /swagger-ui.html
    disable-swagger-default-url: true
    display-query-params-without-oauth2: true
```

```

spring:
  data:
    redis:
      host: redis
      port: 6379
      password: 비밀번호

  datasource:
    url: jdbc:mariadb://mariadb-member:3306/constelink_member
    username: 계정명
    password: 비밀번호
    driver-class-name: org.mariadb.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        default_batch_fetch_size: 100

  security:
    oauth2:
      client:
        registration:
          google: # /oauth2/authorization/google SpringSecurity OAuthLogin
Request Path
          client-id: 클라이언트키
          client-secret: 비밀키값
          redirect-uri:
https://j8a206.p.ssafy.io/member/oauth2/callback/google
          scope:
            - email
            - profile

          # SpringSecurity doesn't provide KAKAO OAuth2 Login
          # add Provider to use KAKAO OAuth2 Login on SpringSecurity
        kakao:
          client-id: 클라이언트키
          redirect-uri: https://도메인/member/oauth2/callback/kakao
          client-authentication-method: POST

```

```
    authorization-grant-type: authorization_code
    scope:
      - profile_nickname
      - profile_image
      - account_email
    client_name: kakao

  provider:
    kakao:
      authorization-uri: https://kauth.kakao.com/oauth/authorize
      token-uri: https://kauth.kakao.com/oauth/token
      user-info-uri: https://kapi.kakao.com/v2/user/me
      user-name-attribute: id

server:
  port: 8080

logging:
  level:
    root: INFO

jwt:
  secret: 비밀키값
  access: 1800
  refresh: 604800

kakao:
  admin:
    key: 클라이언트키
  base:
    fail-url: https://j8a206.p.ssafy.io/kakaoFail
    cancel-url: https://j8a206.p.ssafy.io/kakaoFail
    success-url: https://j8a206.p.ssafy.io/kakao

login:
  redirect-url: https://j8a206.p.ssafy.io/login
```

ConstelinkNotice:

- Dockerfile :


```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- application.yml :

```
springdoc:
  version: v1.0.0
  api-docs:
    path: /api-docs
  default-consumes-media-type: application/json
  default-produces-media-type: application/json
  swagger-ui:
    operations-sorter: alpha
    tags-sorter: alpha
    path: /swagger-ui.html
    disable-swagger-default-url: true
    display-query-params-without-oauth2: true

spring:
  datasource:
    url: jdbc:mariadb://mariadb-notice:3306/constelink_notice
    username: 계정명
    password: 비밀번호
    driver-class-name: org.mariadb.jdbc.Driver
```

ConstelinkFile:

- Dockerfile :

```
FROM openjdk:17-jdk-slim
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

- **application.yml :**

```
springdoc:
  version: v1.0.0
  api-docs:
    path: /api-docs
  default-consumes-media-type: application/json
  default-produces-media-type: application/json
  swagger-ui:
    operations-sorter: alpha
    tags-sorter: alpha
    path: /swagger-ui.html
    disable-swagger-default-url: true
    display-query-params-without-oauth2: true

spring:
  servlet:
    multipart:
      max-file-size: 50MB
      max-request-size: 50MB
  gcp:
    config:
      file: constelink-config.json
    project:
      id: constelink
    bucket:
      id: carrot_box555
    dir:
      name: img
```

II. 빌드 및 배포

1. Podman 설치

- Ubuntu 20.10 버전 미만에서의 설치

```
. /etc/os-release
echo "deb
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbu
ntu_${VERSION_ID}/ /" | sudo tee
```

```
/etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
curl -L
"https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu_${VERSION_ID}/Release.key" | sudo apt-key add -

# 20.10버전 이상에서는 아래 커맨드만 실행
sudo apt update
sudo apt -y install podman
```

- sudo podman [COMMAND] 에서 에러 발생시]

```
# 다음 에러 발생
ERRO[0000] Error loading CNI config list file
/etc/cni/net.d/200-loopback.conflist: error parsing configuration list: no
name

# /etc/cni/net.d/200-loopback.conflist 파일 내용 수정
sudo vi /etc/cni/net.d/200-loopback.conflist

# /etc/cni/net.d/200-loopback.conflist
{
    "cniVersion": "1.0.0",
    "name": "loopback",
    "plugins": [
        {
            "type": "loopback"
        }
    ]
}
```

2. SSL 인증서 발급

- 초기 인증서 발급 (/etc/letsencrypt 에 발급)

```
podman run -it --rm --name certbot \
-p 80:80 \
-v '/etc/letsencrypt:/etc/letsencrypt' \
-v '/var/lib/letsencrypt:/var/lib/letsencrypt' \
certbot/certbot certonly -d '서버도메인' --standalone \
```

```
--server https://acme-v02.api.letsencrypt.org/directory
```

- 재발급

```
podman run -it --rm --name certbot \  
-p 80:80 \  
-v '/etc/letsencrypt:/etc/letsencrypt' \  
-v '/var/lib/letsencrypt:/var/lib/letsencrypt' \  
certbot/certbot renew --standalone \  
--server https://acme-v02.api.letsencrypt.org/directory
```

3. Kubernetes 설치

- Ubuntu 방화벽 사용시 아래 포트들 허용

```
sudo ufw allow 80  
sudo ufw allow 443  
sudo ufw allow 179  
sudo ufw allow 4798  
sudo ufw allow 5473  
sudo ufw allow 6443  
sudo ufw allow 2379  
sudo ufw allow 2380  
sudo ufw allow 10250  
sudo ufw allow 10251  
sudo ufw allow 53
```

- CRI-O 설치

```
# .conf 파일을 만들어 부팅 시 모듈을 로드한다  
  
# Controller / Worker  
  
cat <<EOF | sudo tee /etc/modules-load.d/crio.conf  
overlay  
br_netfilter
```

EOF

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

요구되는 sysctl 파라미터 설정, 이 설정은 재부팅 간에도 유지된다.

Controller / Worker

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cni.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
```

```
sudo sysctl --system
```

Controller / Worker

```
sudo -i
```

```
export OS=xUbuntu_20.04 # OS 버전
```

```
export VERSION=1.26 # cri-o 버전
```

```
echo "deb
```

```
http://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/ /" >
```

```
/etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.list
```

```
curl -L
```

```
https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/Release.key | apt-key add -
```

```
apt-get update
```

```
apt-get install cri-o cri-o-runc
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable crio --now
```

```
sudo systemctl status crio
```

```
# CRI-O는 기본적으로 systemd cgroup 드라이버를 사용한다.
```

- kubeadm 설치

```
# curl 설치 (보통 설치되어 있으므로 pass)
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl

# 구글 클라우드의 공개 사이닝 키를 다운
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

# 쿠버네티스 apt 리포지토리를 추가
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

# apt 패키지 색인을 업데이트하고, kubelet, kubeadm, kubectl을 설치하고 해당
버전을 고정
sudo apt-get update
sudo apt-get install -y kubelet=1.26.2-00 kubeadm=1.26.2-00
kubectl=1.26.2-00
sudo apt-mark hold kubelet kubeadm kubectl

# Kubernetes CRI-O 구성
sudo vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

## 다음 내용 추가
[Service]
Environment="KUBELET_EXTRA_ARGS=--container-runtime=remote
--cgroup-driver=systemd --runtime-request-timeout=15m
--container-runtime-endpoint='unix:///var/run/crio/crio.sock'"

# kubectl alias에 관한 shell 설정 추가
echo "alias k='kubectl'" >> ~/.bashrc
source ~/.bashrc
```

- 유용한 플러그인 kubectl, kubens 설치

```
sudo git clone https://github.com/ahmetb/kubectx /opt/kubectx
sudo ln -s /opt/kubectx/kubectx /usr/local/bin/kubectx
sudo ln -s /opt/kubectx/kubens /usr/local/bin/kubens
```

4. Kubernetes 세팅

- kubeadm으로 마스터 노드 초기화

```
# pod-network-cidr: 내부네트워크 범위
sudo kubeadm init --pod-network-cidr=192.168.0.0/16

# To start using your cluster, you need to run the following as a regular
user:
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Alternatively, if you are the root user, you can run:
export KUBECONFIG=/etc/kubernetes/admin.conf

# 마스터노드에 pod생성 허용
## Taints 확인
kubectl describe node <master-node-name> | grep Taints

## Taints가 존재할 경우 제거
kubectl taint node <master-node-name>
node-role.kubernetes.io/control-plane:NoSchedule-
```

- 워커노드 생성(다른 서버가 있을 경우)

```
# 마스터 노드에서 join 커맨드 출력
kubeadm token create --print-join-command

# 다른 기기에서 워커노드 생성
```

```
kubeadm join <control-plane-host>:<control-plane-port> --token <token>
--discovery-token-ca-cert-hash sha256:<hash>
```

- Calico CNI 설치

```
# Manifest
curl
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml -O

vi calico.yaml
```

```
~~~
livenessProbe:
  exec:
    command:
      - /bin/calico-node
      - -felix-live
      - -bird-live # --> 이 부분을 삭제합니다.
    periodSeconds: 10
    initialDelaySeconds: 10
    failureThreshold: 6
  readinessProbe:
    exec:
      command:
        - /bin/calico-node
        - -felix-ready
        - -bird-ready # --> 이 부분을 삭제합니다.
    ~~~
kind: ConfigMap
apiVersion: v1
metadata:
  name: calico-config
  namespace: kube-system
data:
  # Typha is disabled.
  typha_service_name: "none"
  # 여기를 vxlan으로 수정합니다.
  calico_backend: "vxlan"
```



```

~~~
# Enable IPIP
- name: CALICO_IPV4POOL_IPIP
  value: "Never" # IP-IP 모드 비활성화
- name: CALICO_IPV4POOL_VXLAN
  value: "Always"
~~~

```

```

# calicoctl 설치
cd /usr/local/bin/

sudo curl -O -L
https://github.com/projectcalico/calicoctl/releases/download/v3
.20.6/calicoctl

sudo chmod +x calicoctl

export DATASTORE_TYPE=kubernetes

export KUBECONFIG=~/.kube/config

```

- 인증서 적용

```

# 아래 출력 내용 각각 복사
sudo cat /etc/letsencrypt/live/j8a206.p.ssafy.io/fullchain.pem | base64
sudo cat /etc/letsencrypt/live/j8a206.p.ssafy.io/privkey.pem | base64

# secret.yaml 생성
vi secret.yaml

```

```

# secret.yaml 내용
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: kubernetes.io/tls
data:
  tls.crt: <복사한 fullchain base64값>
  tls.key: <복사한 privkey base64값>

```

```
# secret을 적용
kubectl apply -f secret.yaml
```

- NodePort 할당 가능 범위 변경

```
sudo vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
# 다음 내용 추가
~~~
spec:
  containers:
    - command:
      - --service-node-port-range=1-65535
  ~~~

# kube-system 네임스페이스의 pod를 삭제하면 다시 생성하면서 수정내용 반영
```

- Nginx Ingress Controller 설치

```
curl -o ingress-nginx.yaml
https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.6.4/deploy/static/provider/baremetal/deploy.yaml

# manifest 수정
vi ingress-nginx.yaml
```

```
# Service 부분에 nodePort 추가
~~~
ports:
  - appProtocol: http
    name: http
    port: 80
    protocol: TCP
    targetPort: http
    nodePort: 80
  - appProtocol: https
```

```

    name: https
    port: 443
    protocol: TCP
    targetPort: https
    nodePort: 443
~~~

# 마스터노드에 생성하려면 nodeSelector에 다음 내용 추가
~~~
spec:
  nodeSelector:
    node-role.kubernetes.io/control-plane: ""
~~~

```

```

# ingress-nginx.yaml 적용
kubectl apply -f ingress-nginx.yaml

```

- Ingress 리소스 생성 및 설정 : 서비스를 프록시 하기 위한 적절한 Ingress 설정

예시)

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  tls:
    - hosts:
        - 서버도메인
      secretName: my-secret
  rules:
    - host: 서버도메인
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:

```

```
name: my-service
port:
  number: 80
```

5. DB 배포

- 각 API에 맞는 DB Manifest 생성 : **I-3 Kubernetes Manifest**의 mariadb 항목들 참조
- Manifest 적용

```
kubectl apply -f <mariadb yaml>
```

6. Jenkins CI

- Plugin 설치 : NodeJS, GitLab, Kubernetes 플러그인 설치
- Credential 설정
 1. GitLab API Token : GitLab에서 액세스토큰을 발급받아서 등록
 2. GitLab Account : 종류를 Username with password로 하고 GitLab의 아이디, 비밀번호 입력
 3. Kubeconfig : Kind를 Secret file로 하고 서버에 있는 ~/.kube/config 파일을 등록
- Kubernetes에 Docker Hub Secret 등록

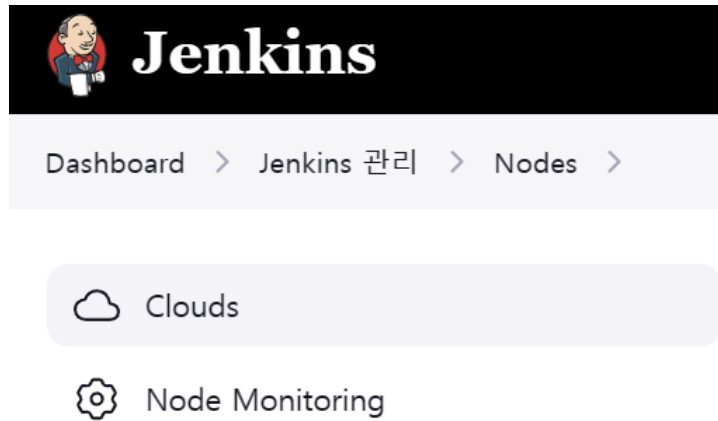
```
kubectl create secret docker-registry [secret name] \
--docker-username="[Docker Hub 계정]" \
--docker-password="[Docker Hub 패스워드]" \
--docker-server=https://index.docker.io/v1/
```

- 스프링부트 프로젝트들의 build.gradle에 다음 내용 추가

```
jar{
    enabled = false
}
```

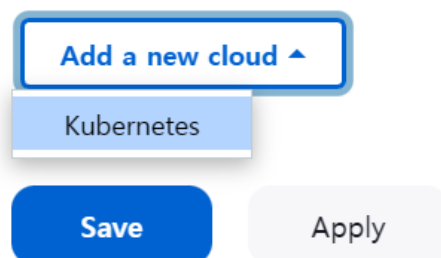
- Jenkins와 Kubernetes Cluster를 연동

1. Jenkins 관리 - Nodes and Clouds - 왼쪽 Clouds 메뉴



2. Add a new cloud - Kubernetes

Clouds



3. 다음과 같이 설정

Clouds

The screenshot shows a configuration window for a Kubernetes cloud. It has a title bar with a hamburger menu icon and the text 'Kubernetes'. Below the title bar, there is a 'Name' field with a question mark icon, containing the text 'kubernetes'. Below this is a section for 'Kubernetes Cloud details' with an expand/collapse arrow and an 'Edited' status. Underneath is a 'Kubernetes URL' field with a question mark icon, containing the text 'https://kubernetes.default.svc'. Below that is a 'Credentials' section with a dropdown menu showing 'config.yaml' and an 'Add' button. At the bottom, there is a status message 'Connected to Kubernetes v1.26.3' and a 'Test Connection' button.

- **Name** : 임의로 설정
- **Kubernetes URL** : https://kubernetes.default.svc
- **Credentials** : 등록된 Kubeconfig 파일


4. Test Connection을 눌러서 연결 확인

- Multibranch Pipeline 생성


1. 새로운 Item - Multibranch Pipeline 선택

Enter an item name


» This field cannot be empty, please enter a valid name




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.




Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특정 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.



Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

2. Branch Sources에 Repository와 Credentials 등록

Branch Sources

Git

Project Repository ?

Credentials ?

Add

Behaviours

Discover branches ?

Add

Property strategy

Add property

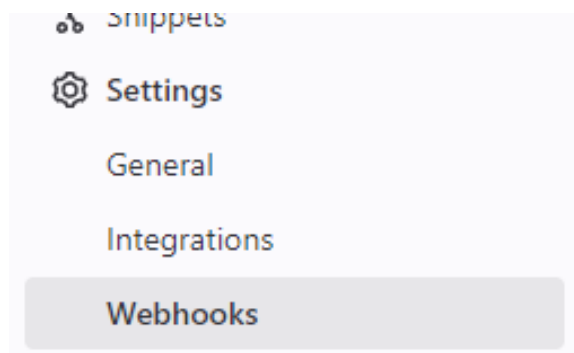
3. Jenkinsfile이 있는 브랜치는 다음과 같이 감지

Branches (18)

S	W	Name ↓
✓	☀	dev-front
✓	☀	feature-back/auth-server
✓	☀	feature-back/beneficiary
✓	☀	feature-back/file
✓	☀	feature-back/fundraising
✓	☀	feature-back/member
✓	☀	feature-back/notice

- GitLab Webhook 설정

1. GitLab 레포지토리 - Settings - Webhooks



2. 다음과 같이 설정

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

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

Secret token

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

Trigger

☒ Push events

Push to the repository.

☐ Tag push events
A new tag is pushed to the repository.

☐ Comments
A comment is added to an issue or merge request.

☐ Confidential comments

7. Argo CD

- Argo 설치

```
kubectl apply -n argocd -f  
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

- NodePort 설정

1. 서버 설정 열기

```
kubectl edit svc argocd-server -n argocd
```

2. type을 NodePort로 변경하고 https에 사용할 포트를 nodePort로 추가

```
~~~  
spec:  
  ports:  
    - name: http  
      port: 80  
      protocol: TCP  
      targetPort: 8080  
    - name: https  
      nodePort: 사용할 포트  
      port: 443  
      protocol: TCP
```

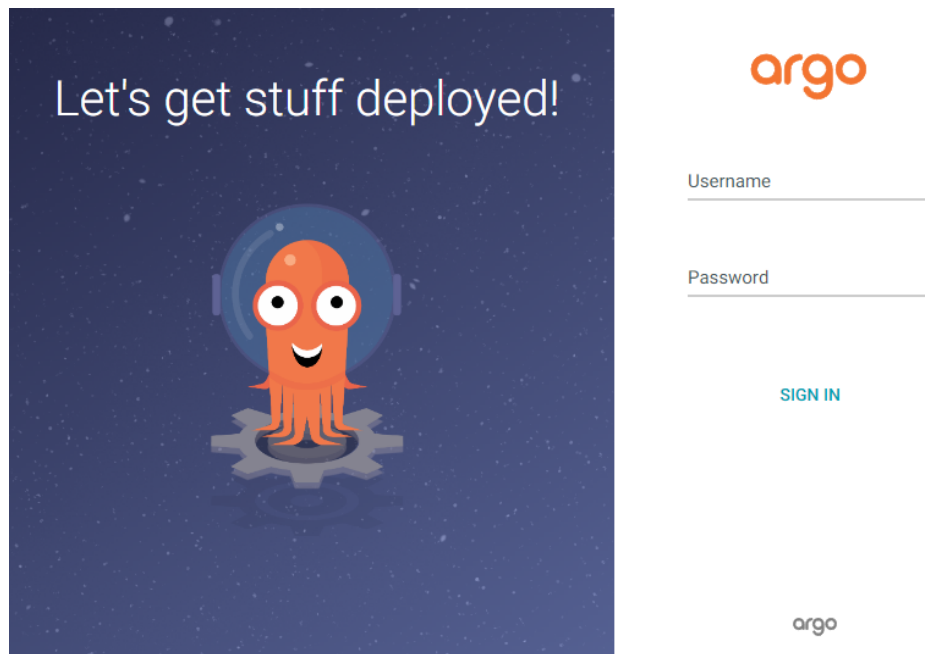
```
targetPort: 8080
selector:
  app.kubernetes.io/name: argocd-server
sessionAffinity: None
type: NodePort
~~~
```

- Argo CD 로그인

1. 초기 비밀번호 복사

```
kubectl -n argocd get secret argocd-initial-admin-secret -o
jsonpath="{.data.password}" | base64 -d; echo
```

2. 설정한 포트로 접속해서 로그인 (username : admin)



- Manifest들을 푸시한 레포지토리 준비 (**I-3 Kubernetes Manifest**의 constelink-*.yaml 파일들)

- Settings - Repository - CONNECT REPO 설정

CONNECT

SAVE AS CREDENTIALS TEMPLATE

CANCEL

Choose your connection method:

VIA HTTPS ▼

CONNECT REPO USING HTTPS

Type

git

Project

default

Repository URL

https://[redacted]-manifest.git

Username (optional)

Password (optional)

- **Project** : 기본으로 default 선택가능
- **Repository URL** : Manifest를 올린 레포지토리 주소
- **Username** : 레포지토리 계정 아이디
- **Password** : 레포지토리 계정 비밀번호

- Applications - NEW APP 설정

The screenshot shows the 'NEW APP' configuration form with the 'GENERAL' tab selected. At the top, there are 'CREATE' and 'CANCEL' buttons. The form contains the following fields:

- Application Name:** constelink
- Project Name:** default
- SYNC POLICY:** Automatic
- PRUNE RESOURCES:** ☒ (with a help icon)
- SELF HEAL:** ☒ (with a help icon)

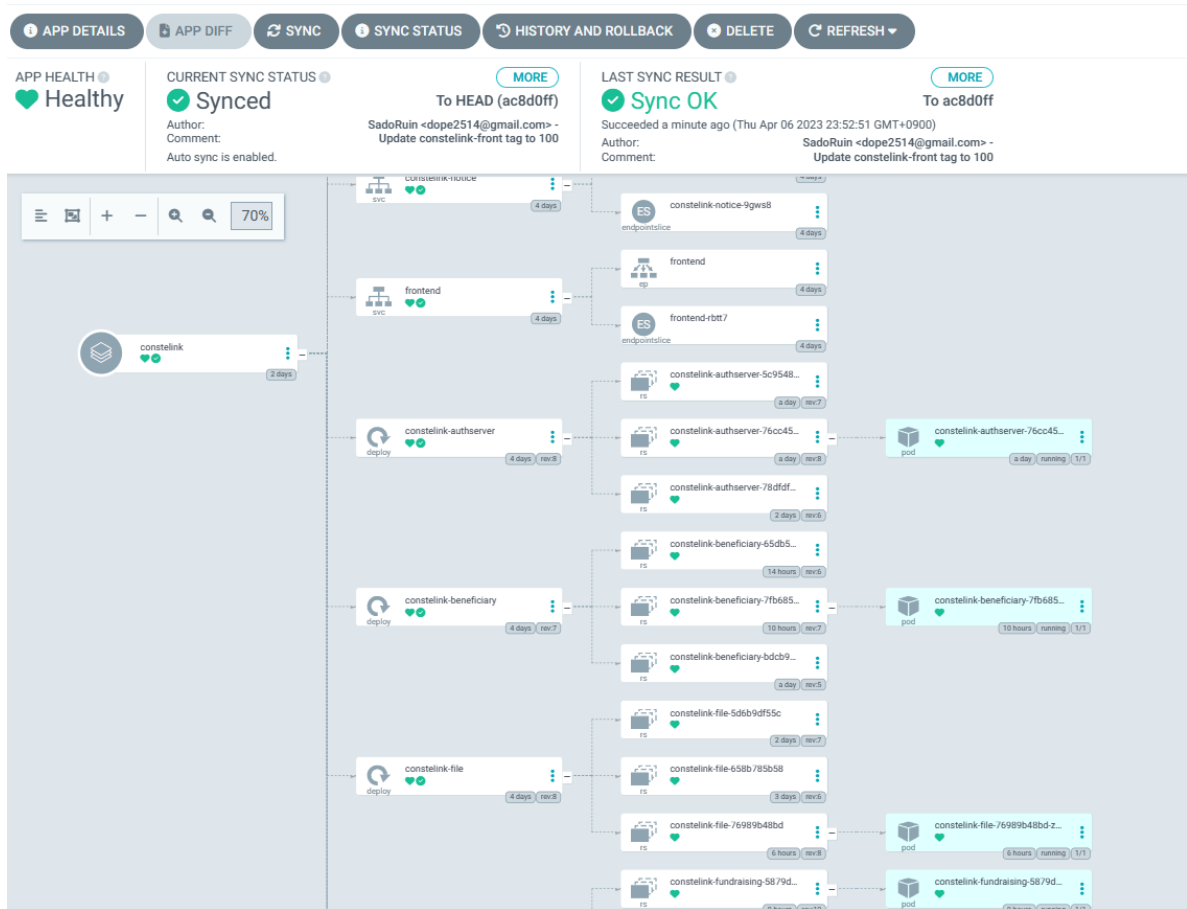
- **Application Name** : 임의로 설정
- **Project Name** : 기본으로 default 선택 가능
- **SYNC POLICY** : Automatic
- **PRUNE RESOURCES** : 체크
- **SELF HEAL** : 체크

The screenshot shows the 'NEW APP' configuration form with the 'SOURCE' tab selected. The form contains the following fields:

- Repository URL:** https://[repository icon]-manifest.git
- Revision:** HEAD
- Path:** manifests

- **Repository URL** : 아까 설정한 주소 선택
- **Path** : 레포지토리내에 manifest들이 들어있는 디렉토리명

- 리소스 한눈에 확인가능 및 젠킨스 빌드 이후 자동으로 배포

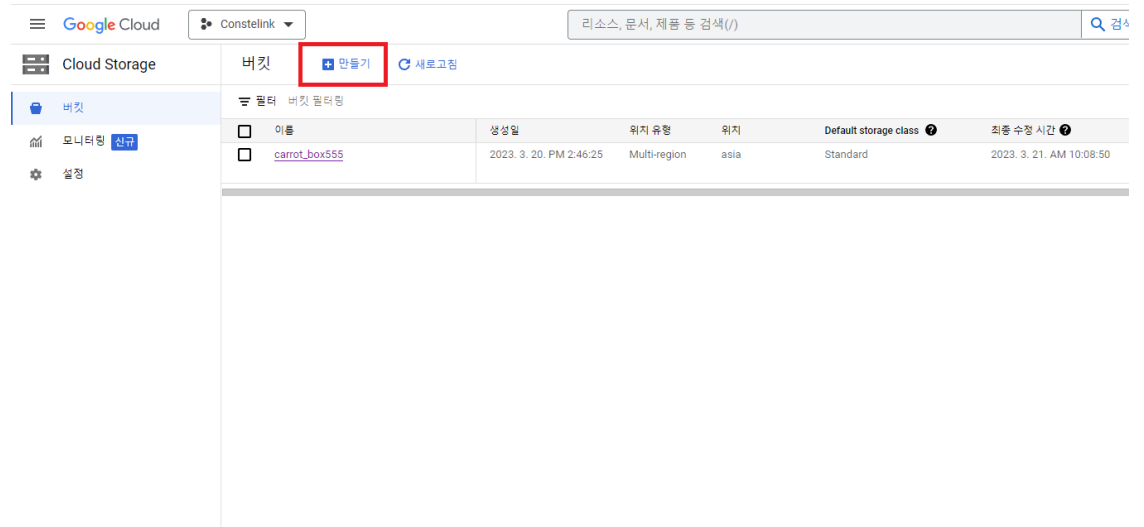


III. 외부 서비스

1. Google Cloud Storage

- Google Cloud Platform 3개월 무료 라이선스 발급
- Google Cloud Platform 프로젝트 생성
- GCS 버킷 생성
 1. Google Cloud Storage 서비스로 이동

2. 버킷 생성 클릭



3. 버킷 이름 및 리전 설정

버킷 이름 지정

전역적으로 고유하고 영구적인 이름을 선택하세요. [이름 지정 가이드라인](#)

test-bucket-5512

팁: 민감한 정보를 포함하면 안 됩니다.

라벨(선택 사항)

계속

데이터 저장 위치 선택

선택사항에 따라 데이터의 지리적 위치가 정의되고 비용, 성능, 가용성이 영향을 받습니다. 나중에 변경할 수 없습니다. [자세히 알아보기](#)

위치 유형

Multi-region

폭넓은 지역에서 가장 높은 가용성

Dual-region

리전 2곳에서 고가용성 및 짧은 지연 시간

Region

단일 리전 내에서 가장 짧은 지연 시간

asia (아시아의 멀티 리전)

계속

54

© 2023. 팀 正육점 all right reserved.

4. 버킷 액세스 제어 방식 선택

- 객체 액세스를 제어하는 방식 선택

공개 액세스 방지

인터넷을 통해 공개적으로 데이터에 액세스할 수 없도록 제한합니다. 이 버킷이 웹 호스팅에 사용되지 않게 합니다. [자세히 알아보기](#)

☒ 이 버킷에 공개 액세스 방지 적용

액세스 제어

☐ 균일한 액세스 제어

버킷 수준 권한(IAM)만 사용하여 버킷의 모든 객체에 대한 균일한 액세스 권한을 가지도록 합니다. 90일이 지나면 이 옵션이 영구적으로 적용됩니다. [자세히 알아보기](#)

☒ 세분화된 액세스 제어

버킷 수준 권한(IAM) 외에도 객체 수준 권한(ACL)을 사용하여 개별 객체에 대한 액세스 권한을 지정합니다. [자세히 알아보기](#)

[계속](#)

5. 버킷 생성 완료 후 우측 메뉴에서 액세스 수정 선택

6. 주 구성원 클릭 후 버킷 액세스 권한 설정

"carrot_box555"에 대한 액세스 권한 부여

주 구성원에게 이 리소스에 대한 액세스 권한을 부여하고 역할을 추가하여 주 구성원이 수행할 수 있는 작업을 지정합니다. 필요에 따라 특정 기준을 충족하는 경우에만 주 구성원에게 액세스 권한을 부여하는 조건을 추가합니다. [IAM 조건 자세히 알아보기](#)

리소스

carrot_box555

주 구성원 추가

주 구성원은 사용자, 그룹, 도메인 또는 서비스 계정입니다. [IAM의 주 구성원 자세히 알아보기](#)

새 주 구성원

allUsers

역할 지정

역할은 권한 집합으로 구성되며, 주 구성원이 이 리소스로 수행할 수 있는 작업을 결정합니다. [자세히 알아보기](#)

역할 * 저장소 개체 뷰어

GCS 개체에 대한 읽기 액세스 권한입니다.

[+ 다른 역할 추가](#)

IAM 조건(선택사항)

IAM 조건 사용 중지됨

-
7. IAM 및 관리자 탭에서 서비스 계정 선택
 8. 생성되어있는 서비스 계정 클릭
 9. 상단에 키 탭 선택 후 키 생성 클릭
 10. JSON 형태의 파일로 다운
 11. 해당 JSON 파일을 프로젝트 폴더 안에 위치시켜 사용 (상세 설정은 application.yml
참고)