

Cloud Z CP CICD (AMF ver.)

2021. 02.

Cloud 그룹

- MTF Team -

I. 개요

II. CI/CD 설정

III. Pipeline 작성 및 배포 실행

IV. Rolling Update

V. AMF Pipeline

본 문서의 목적, 범위, 전제 및 고려사항은 다음과 같음

목적

- Public ZCP Add-on CICD 교육 목적으로 사용자들이 빌드/배포 Hands-On이 가능할 수 있도록 하기 위해서 작성

범위

- Public ZCP Add-on CICD Add-on

전제 및 고려사항

- ZCP Add-on 구축이 된 클러스터

참조 문서

- <https://z-docs.github.io/zcp/cicd/>

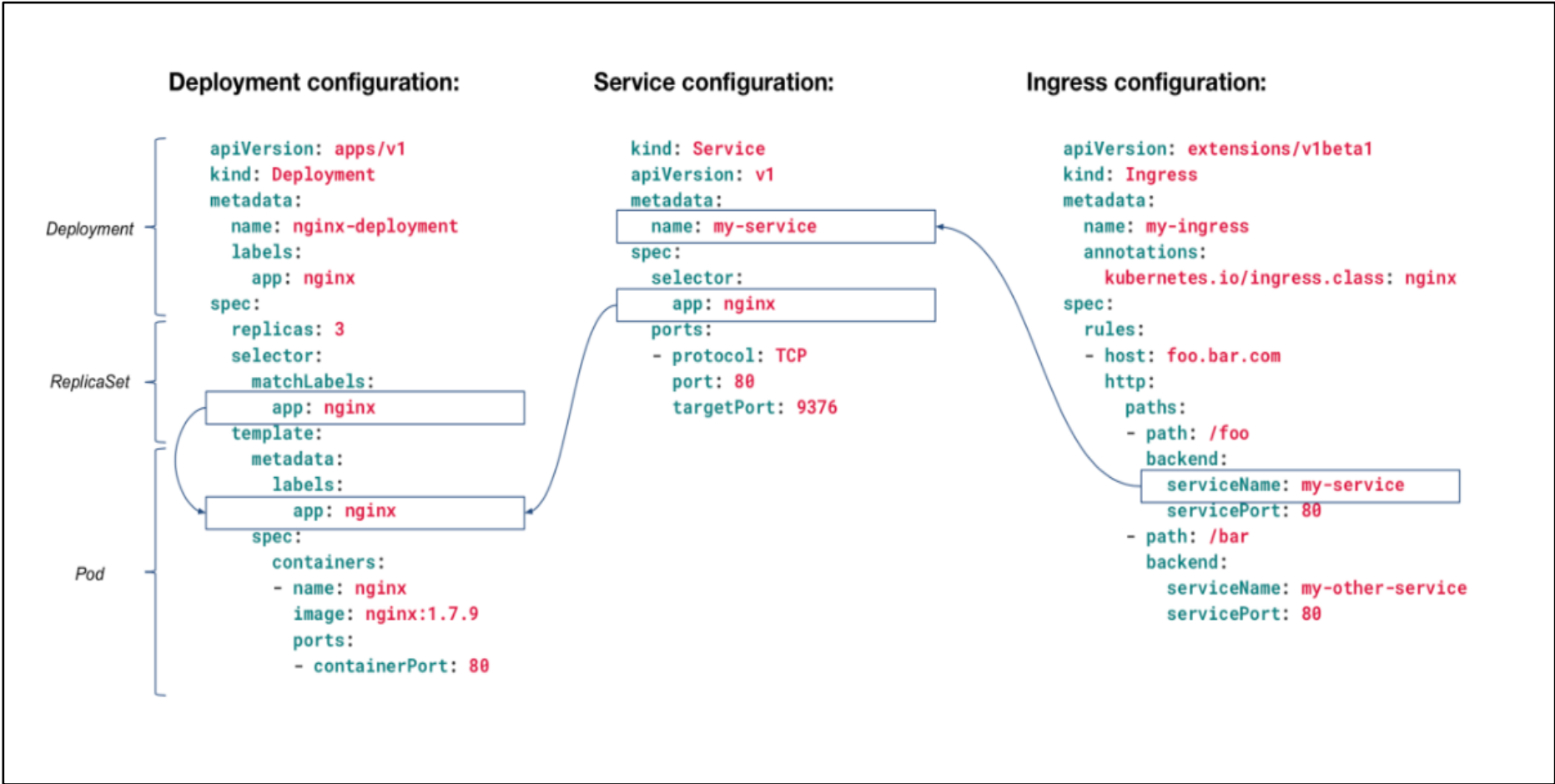
Problems

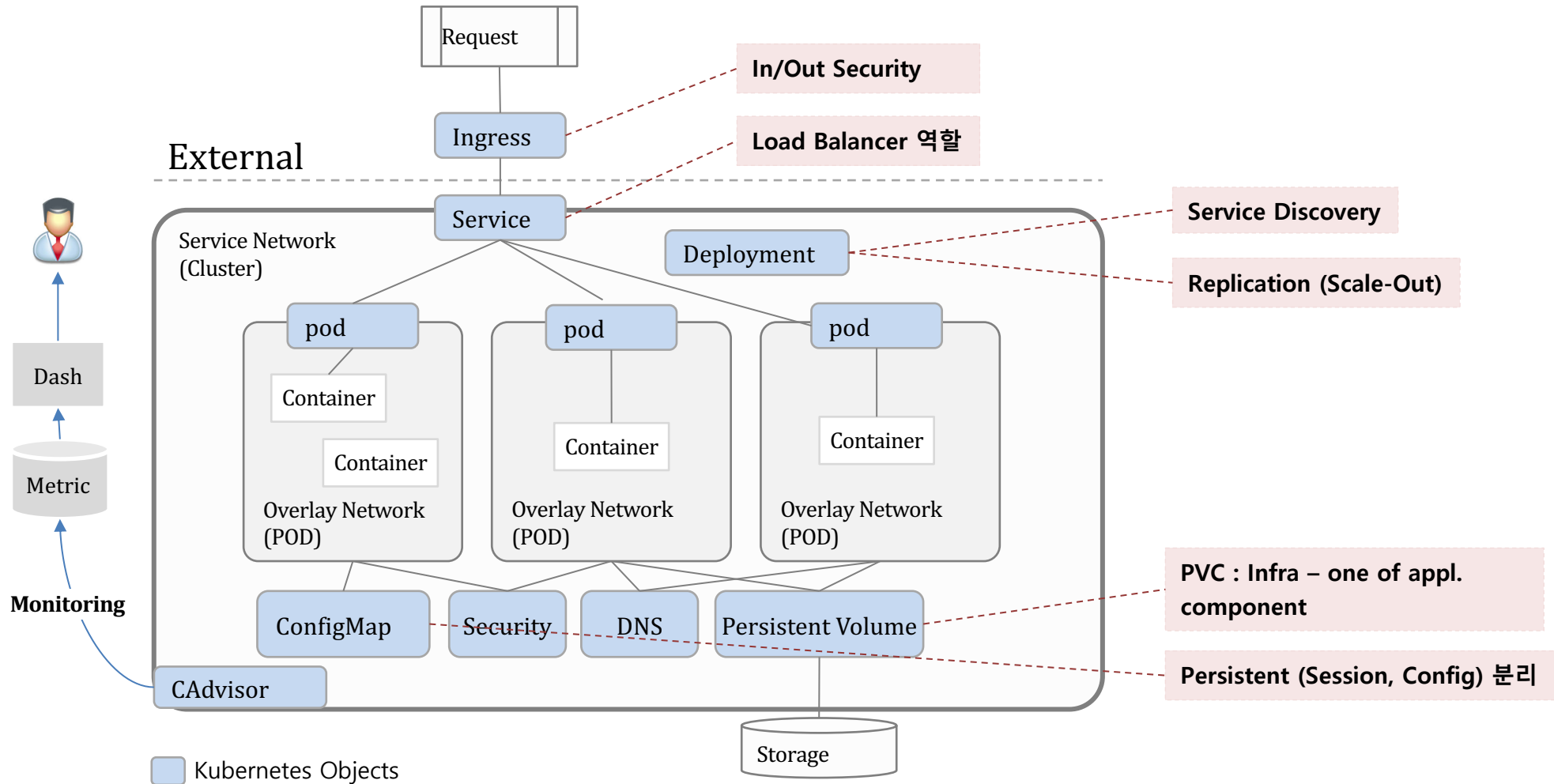
- Business Agility 달성을 위해 Application의 빠르고 지속적인 배포에 대한 요구 증가
- 개발과 운영 조직이 분리
- 배포로 인해 장애 발생 시 운영 조직의 책임 편중
- 배포 주기 길어짐, 배포 한번을 위해 배포 계획 수립 및 검증 작업으로 최소 몇 일 소요됨
- Application의 복잡도 증가
- 관리 대상 서비스/인스턴스 증가

Solution

- 운영환경과 동일한 Infrastructure/환경에서 테스트 수행
- 반복적인 테스트 수행 및 자동화
- Code 기반으로 Development/Staging/Production 환경에 적용
- 빠른 배포 환경
- 표준화를 통한 빌드/배포 프로세스 자동화

Configuration : Ingress <-> Service <-> Deployment 상호 연관 관계
[Key]:[Value] 형태로 자유롭게 선언하여, 자원 선택시 Filter로 사용됨





I. 개요

II. CICD 설정

III. Pipeline 작성 및 배포

IV. Rolling Update

V. AMF Pipeline

II. CI/CD 설정

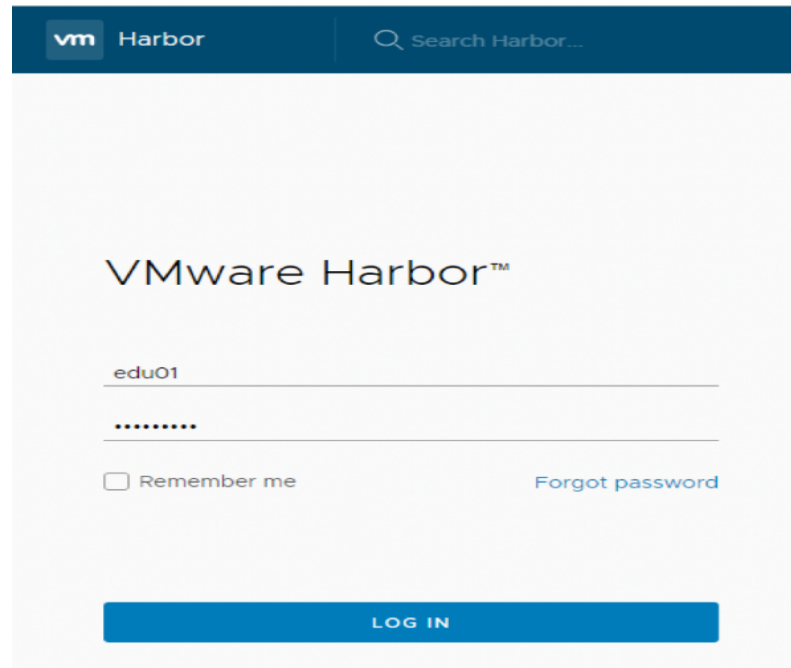
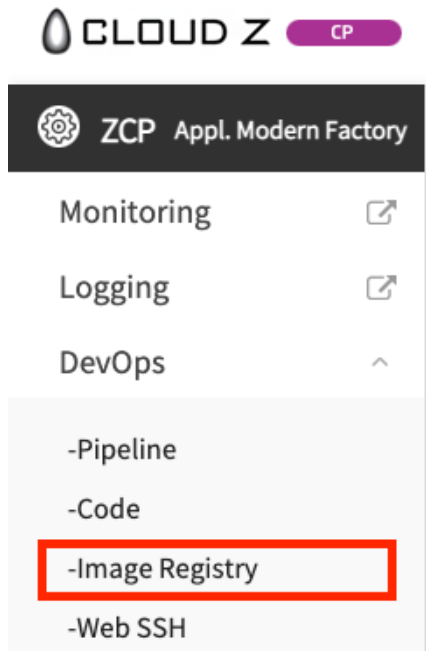
1. Create Harbor Project
2. Create Kubernetes Secret
3. Set up SCM
4. Set up Jenkins

1. Create Harbor Project

What is Harbor?

Harbor는 역할 기반 접근 제어, 이미지 취약점 스캐닝, 이미지 서명 등의 기능을 갖춘 오픈소스 컨테이너 이미지 레지스트리입니다. CNCF Incubating 프로젝트인 Harbor는, Kubernetes와 Docker와 같은 클라우드 네이티브 플랫폼에서 이미지를 안전하고 일관적으로 관리할 수 있는 컴플라이언스와 성능, 상호 운영성을 제공합니다

1. Left menu > Image Registry 클릭
2. Harbor Login(계정이 없을 경우 생성)



Sign Up

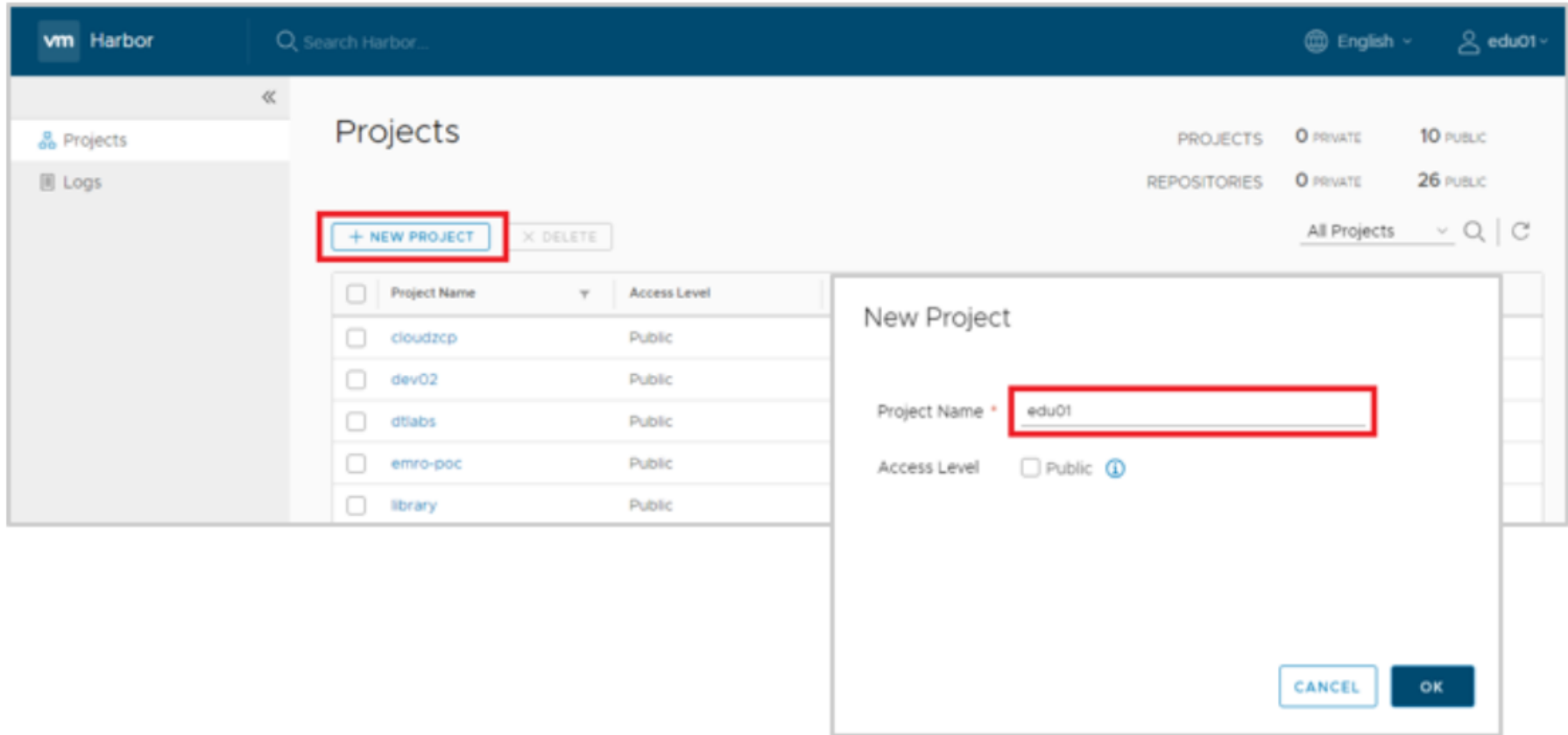
Username *	<input type="text" value="edu01"/>
Email *	<input type="text" value="edu01@sk.com"/> <small>Email is only used for resetting your password.</small>
First and last name *	<input type="text" value="edu01"/>
Password *	<input type="password" value="....."/> <small>8-20 characters long with 1 uppercase, 1 lowercase and 1 number</small>
Confirm Password *	<input type="password" value="....."/>
Comments	<input type="text"/>

CANCEL

SIGN UP

1. Create Harbor Project

3. 좌측 메뉴에서 Project > NEW PROJECT 클릭
4. Project Name 입력 후 OK 클릭 (Project명은 Docker Image 명의 Namespace로 사용됨.)



2. Create Kubernetes Secret

What is kubernetes secret?

K8S Secret Harbor로부터 Docker Image를 받기 위해 필요한 권한 Token임.

1. Left menu > 네임스페이스 > 자신의 namespace Click

2. Secret 탭 선택 > Secret 추가 버튼 클릭

Cloud Z CP

Cloud:

Namespace: 전체

edu01 ●

생성일시: 2018-06-01 00:33:55

Resource 구성 멤버관리 Secret

Secret 정보를 관리합니다.

검색어를 입력하세요.

Secret 목록

Secret Name	Secret Type	생성 일시	Label	관리
데이터가 없습니다.				

총 건수: 0

5

<< < > >>

-Namespaces

2. Create Kubernetes Secret

3. Secret Name 입력, Secret Type Docker Registry Click
4. Docker-server, docker-username, docker-password, docker-email 입력 후 등록

Secret 추가

×

새로운 Secret 정보를 생성합니다.

Secret Name *	harbor-secret
Secret Type	Docker Registry ▾
Label	


docker-server *	asf-registry.cloudzcp.io
docker-username *	edu01
docker-password *
docker-email	


취소등록










3. Setup SCM

II. CI/CD 설정 3. SCM






1. 계정 생성

 CLOUD Z CP

 ZCP VUP

- Clusters 
- Monitoring 
- Logging 
- Service Mesh 
- ZDB 
- Serverless 
- Catalog 
- Notifications 
- DevOps 

- Pipeline
- Code**
- Image Registry
- Web SSH

     로그인

Login with existing credentials to link your existing account to this account. Or, sign up for a new one

사용자 / 비밀번호

사용자 이름이나 이메일 * edu99

비밀번호 *


로그인 [비밀번호를 잊으셨나요?](#)

가입하기

사용자명 * edu99

이메일 * edu99@sk.com

비밀번호 *

재입력 * 




계정 만들기



3. Setup SCM

II. CI/CD 설정


3. SCM

2. 계정 연결 : console 계정 연결 설정

 홈  탐색  도움말

 가입하기  로그인

사용자 / 비밀번호

 OpenID

사용자 / 비밀번호

사용자 이름이나 이메일 *

edu99

비밀번호 *

.....


☐ 자동 로그인

로그인

비밀번호를 잊으셨나요?

계정이 필요하신가요? 지금 가입하세요.




다음을 통해 로그인






3. Setup SCM




3. 저장소 생성







Fork sample URL : <https://asf-git.cloudzcp.io/modern/sam-zcp-lab>




 홈  탐색  도움말


 가입하기  로그인




 modern / sam-zcp-lab










 보기 2  좋아요 0  포크 0

 코드  이슈 0  풀 리퀘스트 0  릴리즈 0  위키  활동

 1 커밋  1 브랜치  1.8MB

 브랜치: master ▼

 HTTPS [https://asf-git.cloudzcp.io/modern/sam-](https://asf-git.cloudzcp.io/modern/sam-zcp-lab)  

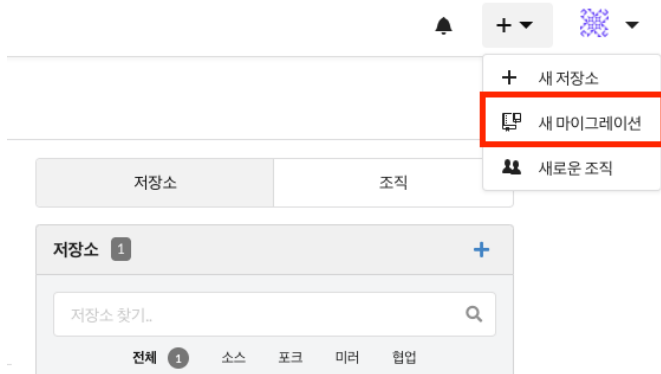
 zzonsang dd9f706f64 init	1분 전
 jenkins-pipeline	init 1분 전
 k8s	init 1분 전
 src	init 1분 전
 Dockerfile	init 1분 전
 README.md	init 1분 전
 mvnw	init 1분 전
 mvnw.cmd	init 1분 전
 pom.xml	init 1분 전

3. Setup SCM

4. 저장소 마이그레이션

' 새 마이그레이션 ' 선택 후 예제 코드를 클론 대상으로 설정하여 생성

1



2

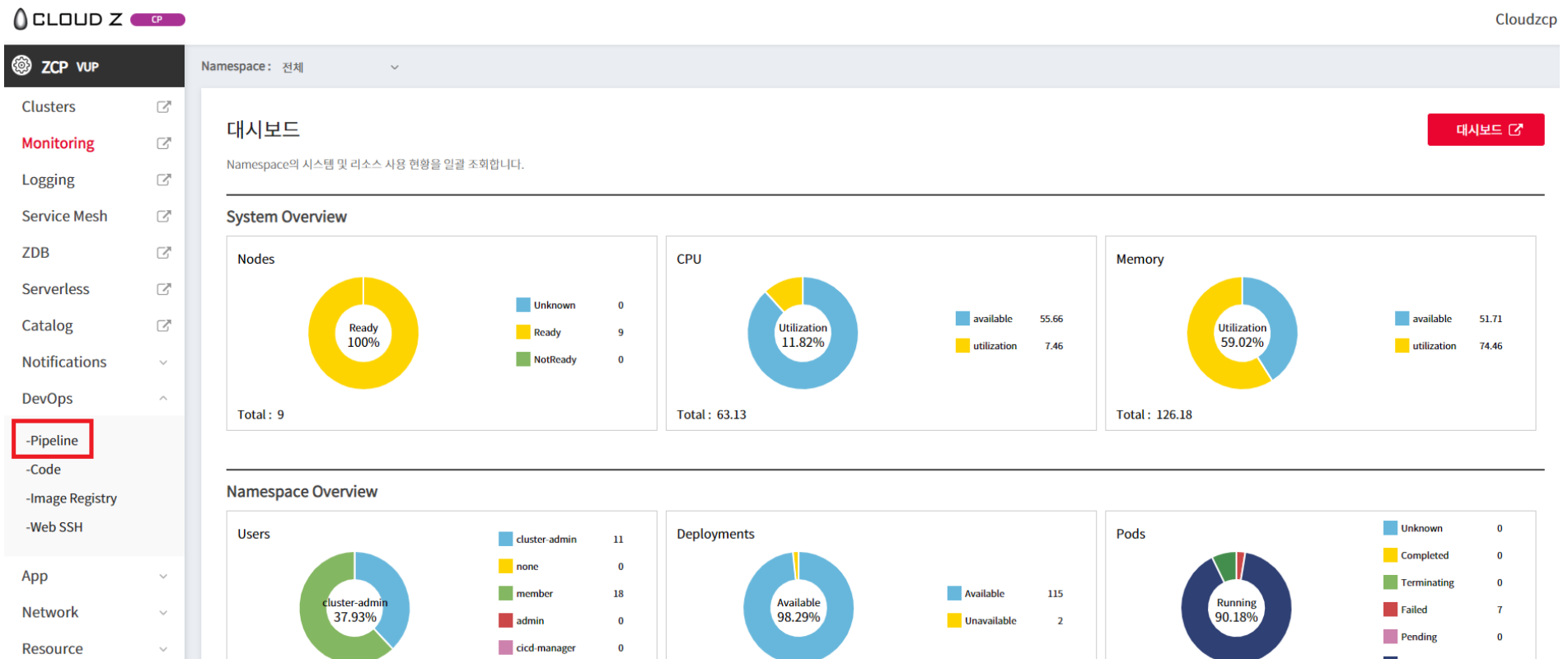
The screenshot shows the '새 마이그레이션' (New Migration) form. At the top, the title '새 마이그레이션' is displayed. Below the title, there's a red rectangular box highlighting the 'URL로 부터 마이그레이트 / 클론' (Migrate from URL / Clone) section. This section contains a text input field with the URL 'https://asf-git.cloudzcp.io/modern/sam-zcp-lab'. Below the URL field, there's a note: 'The HTTP(S) or Git 'clone' URL of an existing repository. When migrating from github, input a username and migration options will be displayed. Mirroring LFS objects is not supported - use 'git lfs fetch --all' and 'git lfs push --all' instead.' Below the note, there's a section labeled '클론시 인증' (Authentication during cloning). This section contains a '소유자' (Owner) dropdown menu with 'edu99' selected, a '저장소 이름' (Repository name) text input field with 'sam-zcp-lab' entered, and two checkboxes: '가시성' (Visibility) with '개인 저장소로 만들기' (Make private repository) and '마이그레이션 유형' (Migration type) with '이 저장소는 미러가 됩니다.' (This repository will be a mirror). Below these fields, there's a '설명' (Description) text area. At the bottom of the form, there are two buttons: '저장소 마이그레이션' (Repository Migration) and '취소' (Cancel).

4. Set up Jenkins

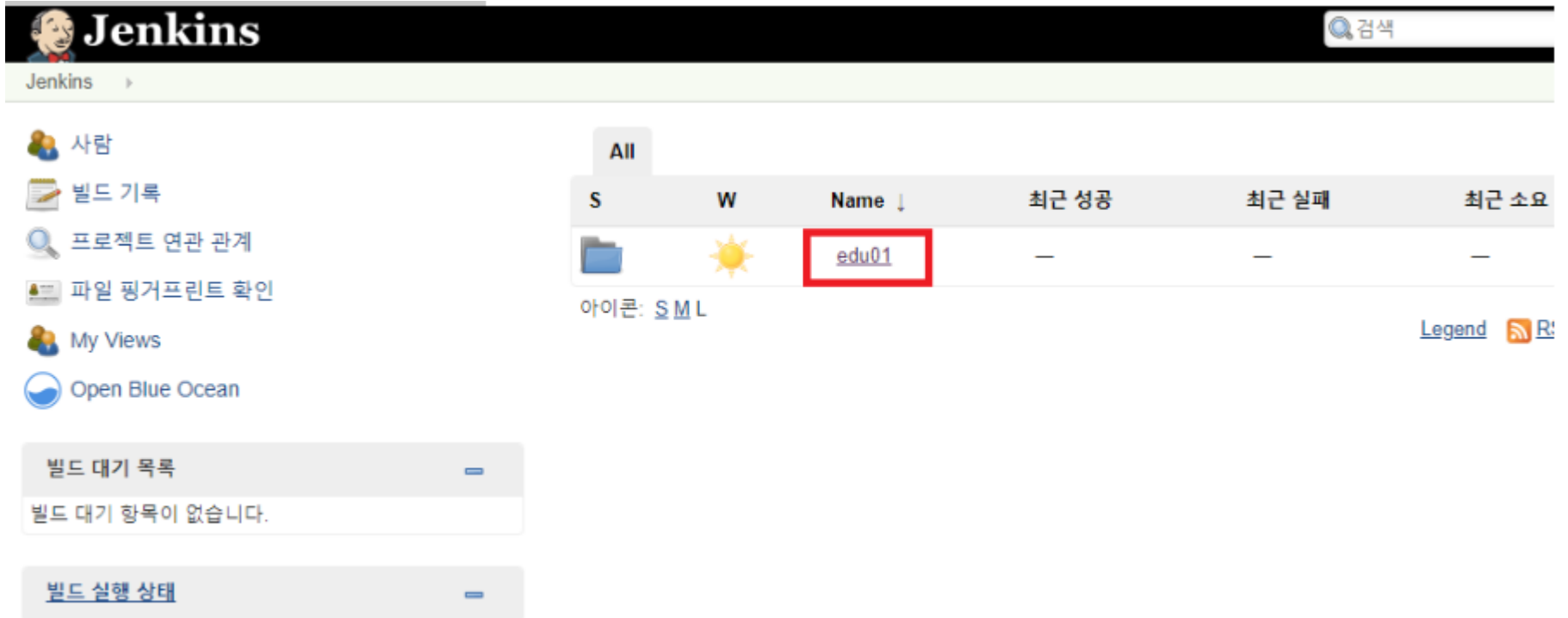
Jenkins 에서 빌드/배포가 실행되면서 필요한 권한 설정과, Kubernetes를 위한 Pipeline 작성 법을 설명함. Jenkins는 ZCP에서 관리하는 사용자그룹(Namespace)과 권한이 적용됨.

사용되는 정보는 User ID = edu01 , Namespace = edu01



1. Jenkins : Web Console에서, DevOps > Pipeline




2. Namespace와 동일한 폴더 Click



The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with navigation links: 사람 (People), 빌드 기록 (Build History), 프로젝트 연관 관계 (Project Relationships), 파일 핑거프린트 확인 (Check File Fingerprints), My Views, and Open Blue Ocean. Below these are sections for '빌드 대기 목록' (Build Queue) and '빌드 실행 상태' (Build Execution Status). The main area displays a table of items under the 'All' tab. The table has columns: S, W, Name ↓, 최근 성공 (Last Success), 최근 실패 (Last Failure), and 최근 소요 (Last Duration). A single item named 'edu01' is listed, with its 'Name' cell highlighted by a red box. Below the table, there are links for '아이콘: S M L' and 'Legend' with an RSS icon.

S	W	Name ↓	최근 성공	최근 실패	최근 소요
		edu01	—	—	—

아이콘: S M L

Legend 

4. Set up Jenkins

3. Jenkins 왼쪽 메뉴에서 *Credentials* > edu01(in *Stores scoped to edu001*) > *Global credentilas* Click

The screenshot shows the Jenkins web interface. On the left sidebar, the 'Credentials' menu item is highlighted with a red box. The main content area displays the 'Credentials' page for the 'edu01' store. A table titled 'Stores scoped to edu01' lists the available credential stores. The 'edu01' store is highlighted with a red box. Below this, a 'Folder' section shows a table with one entry: 'Global credentials (unrestricted)'. This entry is also highlighted with a red box. The description for this folder is 'Credentials that should be available irrespective of domain specification to requirements matching'.

T	P	Store	Domain	ID	Name
아이콘: S M L					
Stores scoped to edu01					
P	Store	Domains			
	edu01	/global			

Domain	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching

아이콘: S M L

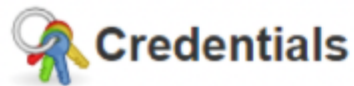
4. Set up Jenkins

4. 왼쪽 메뉴에서 Add Credentials Click
5. Git/Harbor Username과 Password를 입력함
 - ID: GIT_CREDENTIALS
 - ID: HARBOR_CREDENTIALS

The image displays two screenshots of the Jenkins web interface. The left screenshot shows the 'Add Credentials' button in the left sidebar, which is highlighted with a red box. The right screenshot shows the 'Add Credentials' form, which is also highlighted with a red box. The form contains the following fields:

Field	Value
Kind	Username with password
Username	edu01
Password	*****
ID	GIT_CREDENTIALS
Description	GIT_CREDENTIALS

An 'OK' button is visible at the bottom of the form.



T	P	Store ↓	Domain	ID	Name
		edu01	(global)	GIT_CREDENTIALS	edu01/*****
		edu01	(global)	HARBOR_CREDENTIALS	edu01/*****

아이콘: [S](#) [M](#) [L](#)

Stores scoped to [edu01](#)

P	Store ↓	Domains
	edu01	(global)

Table of Contents

I. 개요

II. CICD 설정

III. Pipeline 작성

IV. Rolling Update

V. AMF Pipeline

III. Pipeline 작성

- 1. Get Sample Application Source**
- 2. Create Pipeline**
 - 2-1. Development pipeline**
 - 2-2. Script 작성법**
 - 2-3. 배포실행 및 확인**

1. Get Sample Applicaion Source

1. Get Sample Application Source

1. Open browser and go <https://asf-git.cloudzcp.io/modern/sam-zcp-lab>
2. 예제 프로젝트 Checkout
 - *Clone or download* > *Copy* click

modern / sam-zcp-lab

보기 2 좋아요 0 포크 0

코드 이슈 0 풀리퀘스트 0 릴리즈 0 위키 활동

1 커밋 1 브랜치 1.8MB 복사

브랜치: master

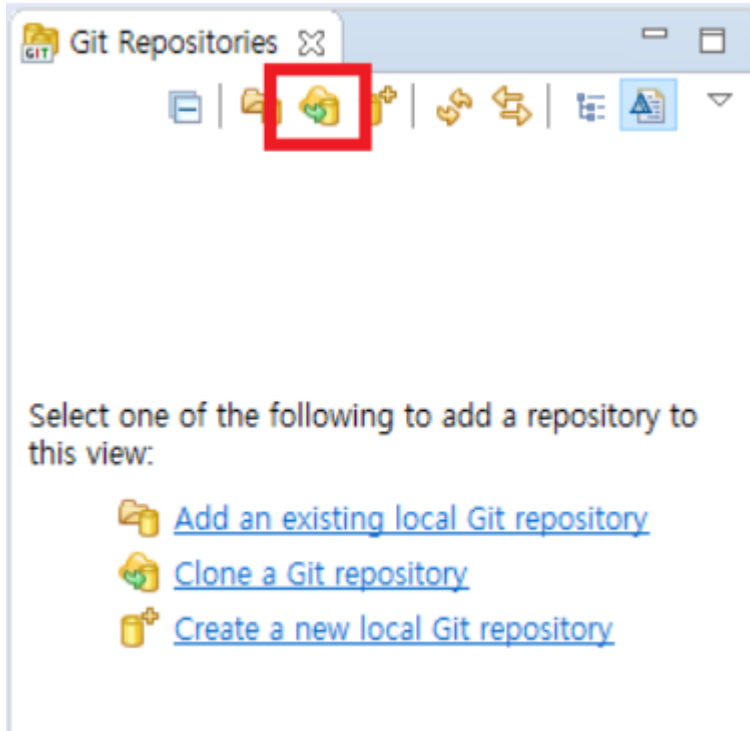
HTTPS SSH <https://asf-git.cloudzcp.io/modern/sam-zcp-lab>

File Name	Commit Hash	Initials	Time
zonsang	dd9f706f64	init	15 분 전
jenkins-pipeline		init	15 분 전
k8s		init	15 분 전
src		init	15 분 전
Dockerfile		init	15 분 전
README.md		init	15 분 전
mvnw		init	15 분 전
mvnw.cmd		init	15 분 전
pom.xml		init	15 분 전

1. Get Sample Applicaion Source

1. Get Sample Application Source

- Eclipse > Git Repository > Clone a Git repository 클릭



1. Get Sample Applicaion Source

1. Get Sample Application Source

- URI 입력 후 Next > Next > Finish

<https://asf-git.cloudzcp.io/modern/sam-zcp-lab>

Clone Git Repository

Source Git Repository
Enter the location of the source repository.

Location

URI: Local File...

Host:

Repository path:

Connection

Protocol: ▼

Port:

Authentication

User:

Password:

☒ Store in Secure Store

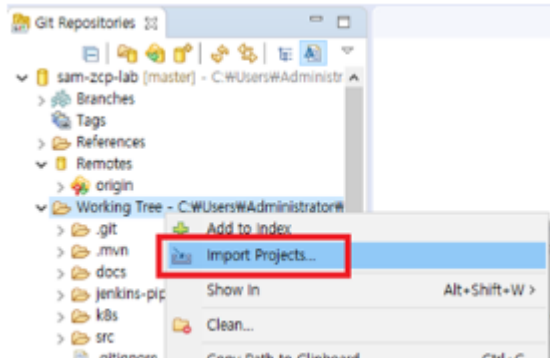
? < Back Next > Finish Cancel

1. Get Sample Applicaion Source

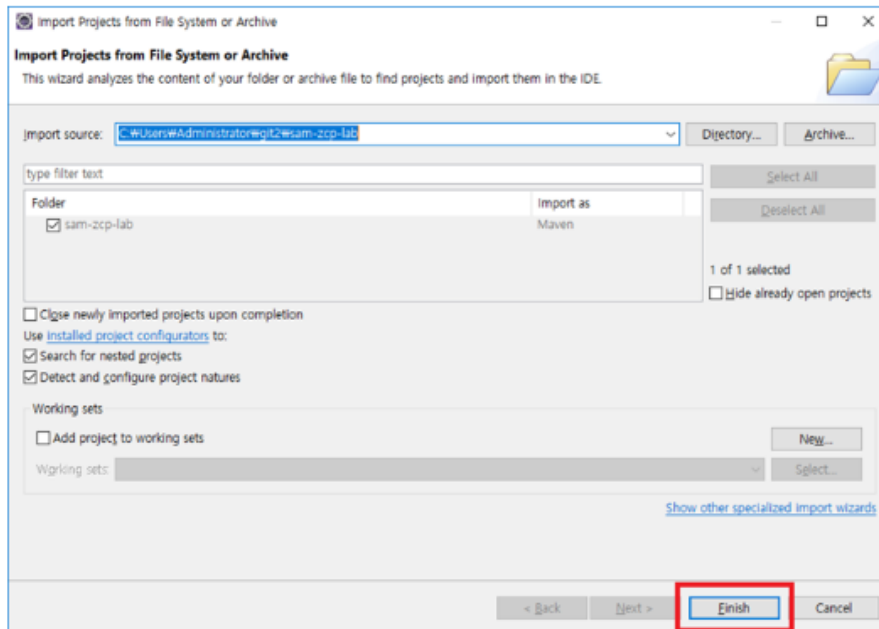
III. Pipeline 작성 1. Get Sample Application Source

3. Import Project

- Working Tree 선택 후 팝업 메뉴에서 Import Projects 선택



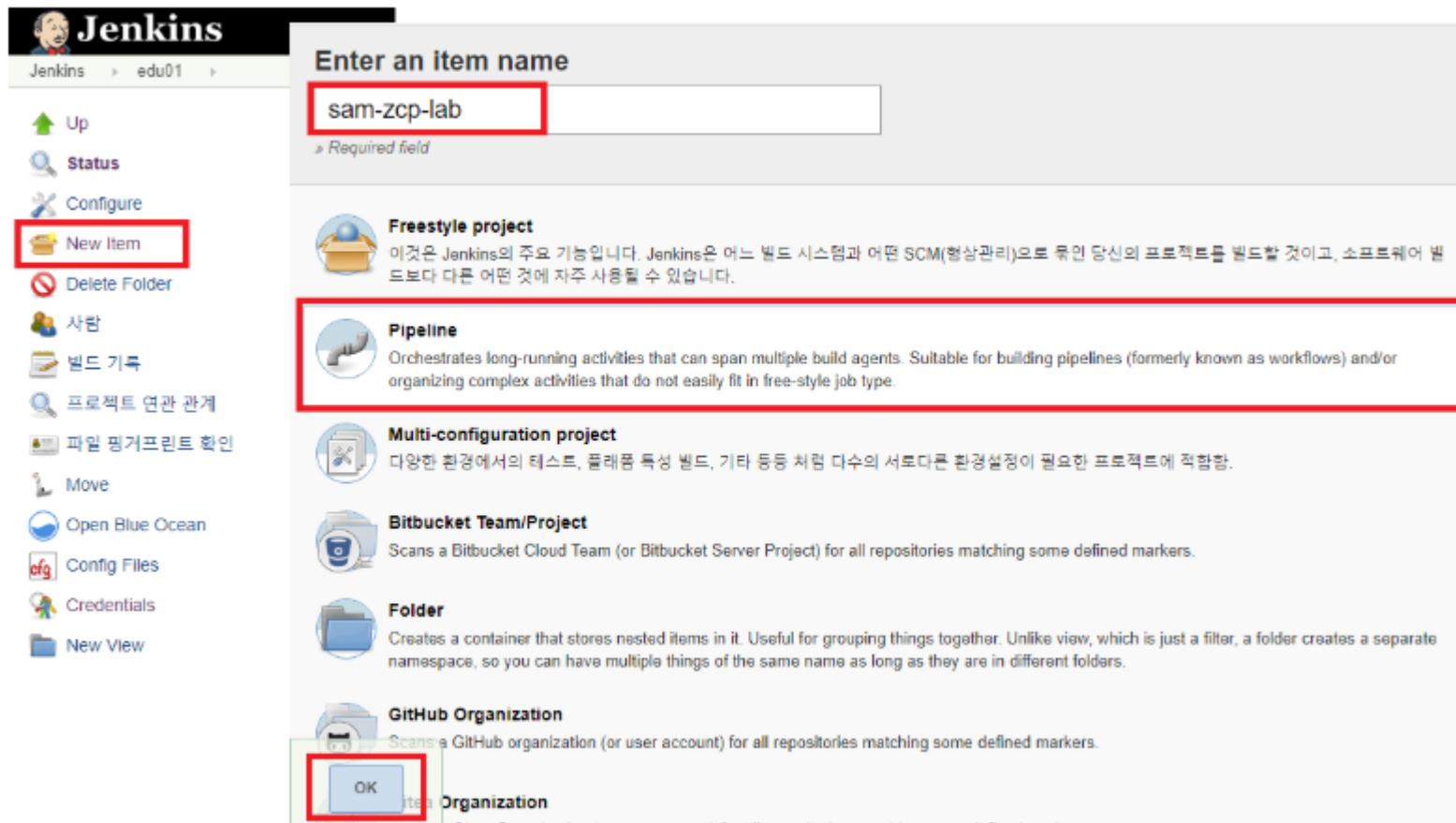
- Finish 클릭



2-1. Development pipeline

III. Pipeline 작성 2. Create Pipeline

1. edu01 폴더 Click
2. 왼쪽메뉴에서 *New Item* Click
3. Inputbox에 sam-zcp-lab(jenkins jobname) 입력
4. Pipeline 선택



5. Pipeline에 필요한 정보 입력: Pipeline section으로 이동(Scroll down)

- Definition 선택 : *Pipeline script from SCM*
- SCM 선택: *Git*
- Repositories
 - Repository URL 입력: *https://asf-git.cloudzcp.io/[edu01]/sam-zcp-lab.git*
 - Credentials 선택: *edu01/...(GIT CREDENTIALS)*
- Branch to build 입력 : **/master*
- Repository browser 선택 : *Gitea*
 - URL 입력: *https://asf-git.cloudzcp.io/[edu01]/sam-zcp-lab* ('.git' 제거, browser url)
- Script Path 입력 : *jenkins-pipeline/deploy-pipeline* (Git프로젝트 Root Path기준 상대 경로)
- 저장

2-1. Development pipeline

III. Pipeline 작성 2. Create Pipeline

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition Pipeline script from SCM

SCM Git

Repositories

Repository URL `https://asf-git.cloudzcp.io/edu99/sam-zcp-lab.gi`

Credentials `edu99/***** (GIT_CREDENTIALS)`

[Add](#) [고급...](#) [Add Repository](#)

Branches to build

Branch Specifier (blank for 'any') `*/master` [Add Branch](#)

Repository browser `gogs`

URL `https://asf-git.cloudzcp.io/edu99/sam-zcp-lab`

Additional Behaviours [Add](#)

Script Path `Jenkinsfile-pipeline/deploy-pipeline`

Lightweight checkout ☒

[Pipeline Syntax](#)

[저장](#) [Apply](#)

edu99 -> 개인별 계정 정보로 대체

구성

Block	내용
변수정의	내장 변수 및 Job에 필요한 기본 변수 선언
환경구성	내부에서 사용할 Resource에 필요한 pod 정의
Volume 선언	각각의 pod에서 사용할 저장소 설정
Job 선언	Git Checkout, Source Build, Docker Image build, Deploy

K8s manifest

1. k8s/deployment.yaml 파일 수정

```
1  ...
2      containers:
3      - name: spring-boot-cicd-demo
4        image: vup-registry.cloudzcp.io/edu01/spring-boot-cicd-demo:develop
5        ports:
6        - containerPort: 8080
7          name: tomcat
8  ...
```

2. k8s/ingress.yaml 작성

```
1  apiVersion: extensions/v1beta1
2  kind: Ingress
3  metadata:
4    name: spring-boot-cicd-demo
5  spec:
6    rules:
7    - host: edu01.cloudzcp.io
8      http:
9        paths:
10       - path: /
11         backend:
12           serviceName: spring-boot-cicd-demo
13           servicePort: 80
14
```

변수정의

jenkins-pipeline/deploy-pipeline 파일 수정

- label: 내부에서 사용하는 UUID
- DOCKER_IMAGE: Pipeline에서 사용할 이름. [Registry URL]/[Repository Name]/[Image Name]. Tag명은 생략하고 정의 함. Tag는 변수로 입력 받거나, 자동할당됨
- ZCP_USERID : 배포 시 사용 할 ZCP 사용자 계정.
- K8S_NAMESPACE: 배포영역의 Namespace 이름
- VERSION: 개발 단계는 develop으로 고정처리함.

```
1 // Jenkins Shared Library 적용
2 @Library("retort-lib") _
3 // Jenkins slave pod에 uuid 생성
4 def label = "Jenkins-${UUID.randomUUID().toString()}"
5 def ZCP_USERID = 'edu01'
6 def DOCKER_IMAGE = 'edu01/spring-boot-cicd-demo' // Harbor Project Name : edu01
7 def K8S_NAMESPACE = 'edu01'
8 def VERSION = 'develop'
```

Volume 선언

```
volumes: [  
    persistentVolumeClaim(mountPath: '/root/.m2', claimName: 'zcp-jenkins-mvn-repo-namespace 명')  
])
```

BUILD DOCKER IMAGE

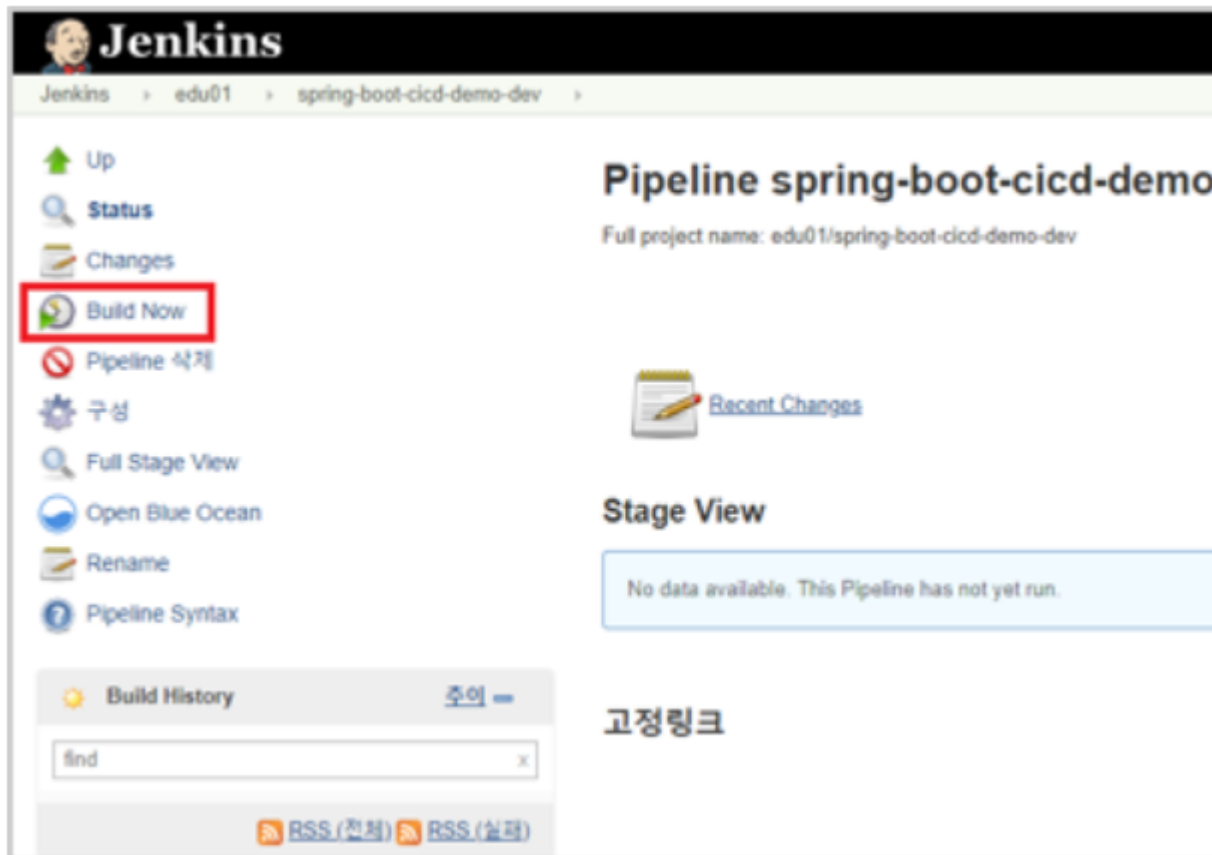
```
stage('BUILD DOCKER IMAGE') {  
    container('docker') {  
        dockerCmd.build tag: "${HARBOR_REGISTRY}/${DOCKER_IMAGE}:${VERSION}"  
        dockerCmd.push registry: HARBOR_REGISTRY, imageName: DOCKER_IMAGE, imageVersion: VERSION, credentialsId: "HARBOR_CREDENTIALS"  
    }  
}
```

Deploy

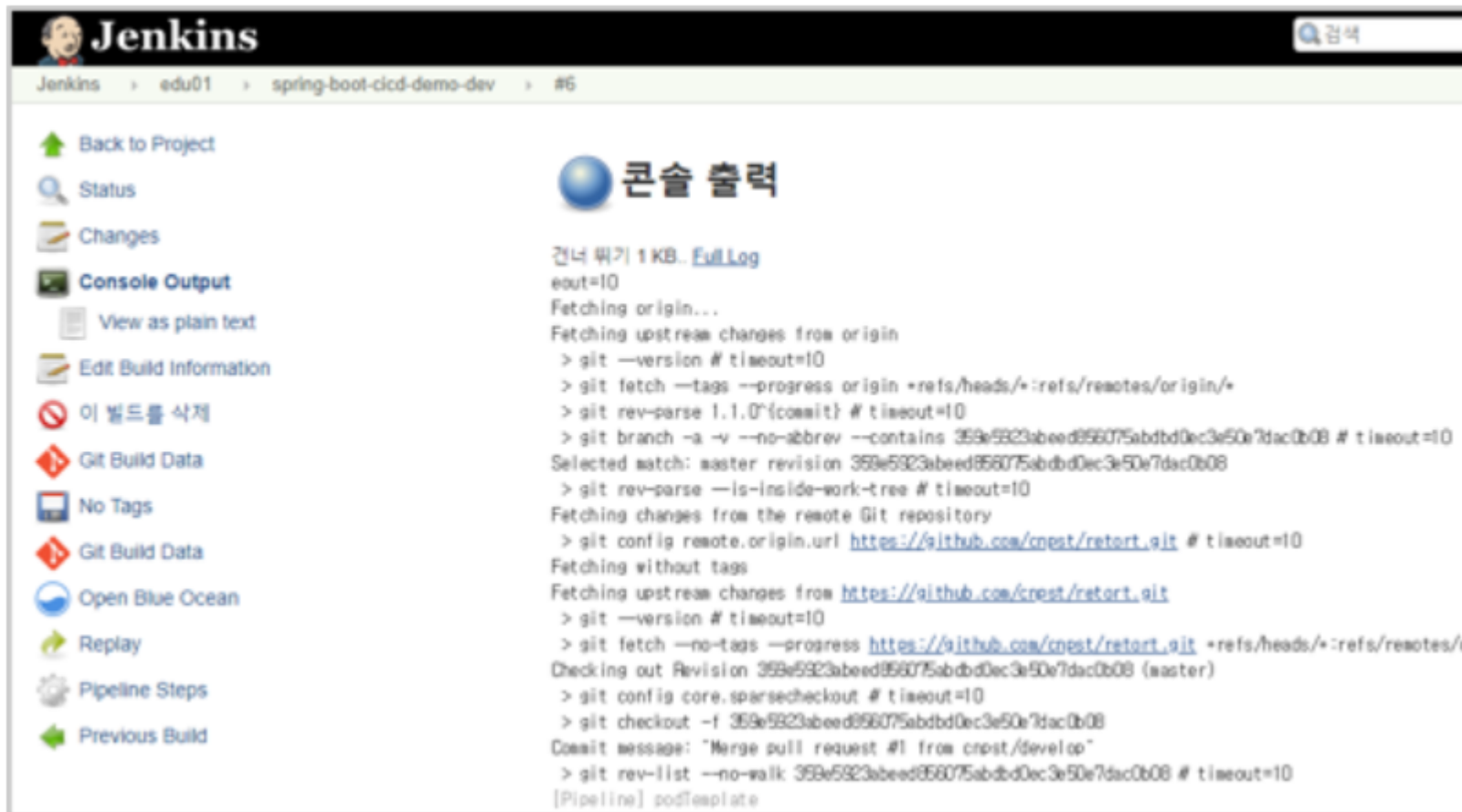
```
stage('DEPLOY') {  
    container('kubectrl') {  
        kubeCmd.apply file: 'k8s/service.yaml', namespace: K8S_NAMESPACE  
        kubeCmd.apply file: 'k8s/ingress.yaml', namespace: K8S_NAMESPACE  
        kubeCmd.apply file: 'k8s/deployment.yaml', namespace: K8S_NAMESPACE, wait: 300  
        //TODO: kubectrl command block  
        // deploy service, ingress, deployment  
    }  
}
```

2-3. 배포 실행 및 확인

1. Git : Staged > Commit > Push
2. 실행 : Job Menu > *Build Now* Click



3. 확인 : 콘솔 출력



4. Open Browser : <http://edu01.cloudzcp.io>

Table of Contents

I. 개요

II. CICD 설정

III. Pipeline 작성

IV. Rolling Update

V. AMF Pipeline

IV. Rolling Update

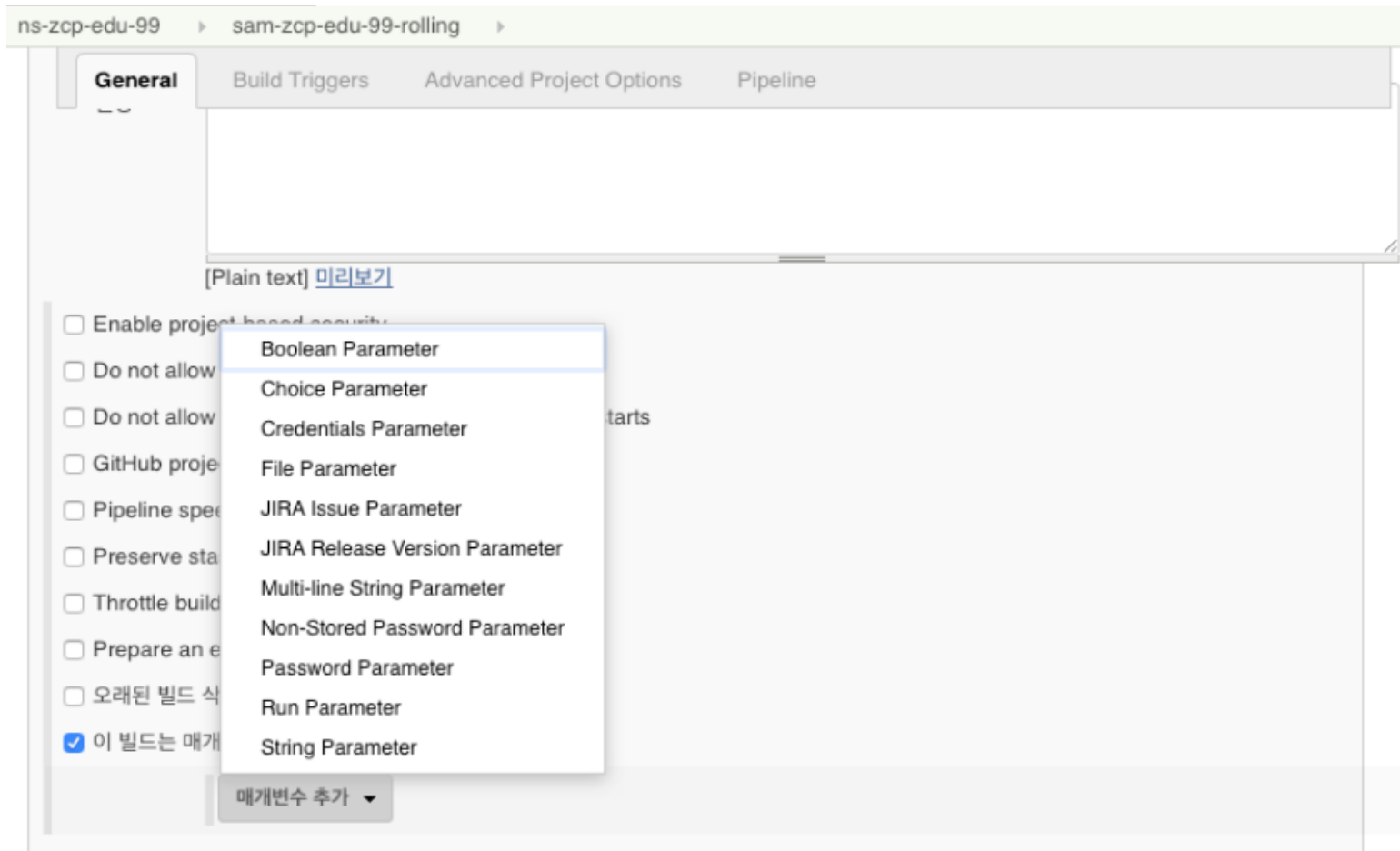
1. Deployment 수정
2. Pipeline 작성
3. 소스변경
4. Rolling update 실행

- spec.strategy.type: Rollingupdate # 배포방식 설정
- spec.strategy.rollingUpdate.maxSurge: 1 # up pod 최대 단위
- spec.strategy.rollingUpdate.maxUnavailable: 1 # down pod 최대 단위

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: spring-boot-cicd-demo
5    labels:
6      app: spring-boot-cicd-demo
7  spec:
8    replicas: 5
9    selector:
10     matchLabels:
11       app: spring-boot-cicd-demo
12    strategy:
13     type: RollingUpdate
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 1
17    ....
```

sam-zcp-lab-rolling 이름으로 Pipeline작성

1. Pipeline 설정에서 Parameter 설정 : VERSION 처리를 위해 Pipeline에 변수 추가
2. General 영역에서 *이 빌드는 매개변수가 있습니다.* 체크
3. 매개변수 추가 Click > String Parameter
 1. 매개변수명 : VERSION
 2. Default Value: develop



GeneralBuild TriggersAdvanced Project OptionsPipeline

☐ GitHub project

☐ Pipeline speed/durability override

☐ Preserve stashes from completed builds

☐ Throttle builds

☐ Prepare an environment for the run

☐ 오래된 빌드 삭제

☒ 이 빌드는 매개변수가 있습니다

String Parameter

매개변수 명VERSION

Default Valuedevelop

설명

[Plain text] [미리보기](#)

☐ Trim the string

매개변수 추가

X

?

?

?

?

?

?

?

?

4. Pipeline 영역

1. Script Path : Production용 파일로 변경. jenkins-pipeline/rolling-pipeline
5. Git project의 jenkins-pipeline/rolling-pipeline 파일 편집
6. VERSION 변수선언 주석 처리

```
1 // def VERSION = 'develop'
```

7. Job설정의 Deploy 변경

```
kubeCmd.apply file: 'k8s/service.yaml', namespace: K8S_NAMESPACE  
yaml.update file: 'k8s/deployment.yaml', update: ['.spec.template.spec.containers[0].image':  
"${HARBOR_REGISTRY}/${DOCKER_IMAGE}:${VERSION}"]  
kubeCmd.apply file: 'k8s/deployment.yaml', wait: 300, recoverOnFail: false, namespace: K8S_NAMESPACE
```

1. Open file(./src/main/resource/static/css/style.css)
2. 47 line의 색상 값 변경 (background-color: #157ed2; --> background-color: red)
3. Stage & Commit
4. Push to origin/master(Git Server)

4. Rolling update 실행

IV. Rolling update 4. Rolling update 실행

1. Click : Build with Parameters
2. VERSION 입력 : rolling-1

The image shows the Jenkins web interface for a pipeline named 'spring-boot-c'. The left sidebar contains navigation links: Up, Status, Changes, Build with Parameters (highlighted with a red box), Pipeline 삭제, 구성, Full Stage View, Open Blue Ocean, Rename, and Pipeline Syntax. The main area displays the pipeline name and full project name. Below this, there's a 'Recent Changes' section and a 'Stage View' section which currently shows 'No data available'. A '고정링크' (Fixed Link) section is also visible. A modal window titled 'Pipeline spring-boot-cicd-demo-dev-rolling' is open, showing a message '매개변수가 필요한 빌드입니다.' (Build requires parameters). It has a 'VERSION' input field with the value 'rolling-1' (highlighted with a red box) and a '빌드하기' (Build) button.

4. Rolling update 실행

IV. Rolling update 4. Rolling update 실행

pod 상태 확인 : `kubectl get deploy -n edu01`

```
$ kubectl get deploy -n edu01
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
spring-boot-cicd-demo	5	5	5	5	1h

```
$ kubectl get po -n edu01
```

NAME	READY	STATUS	RESTARTS	AGE
spring-boot-cicd-demo-54657b864c-62lv9	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-fq2qc	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-fwqgr	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-gjnfd	1/1	Terminating	0	3h
spring-boot-cicd-demo-54657b864c-l47fk	1/1	Running	0	3h
spring-boot-cicd-demo-f98f5bf6f-d4d9x	0/1	Pending	0	0s
spring-boot-cicd-demo-f98f5bf6f-fns7l	0/1	ContainerCreating	0	0s

```
$ kubectl get po -n edu01
```

NAME	READY	STATUS	RESTARTS	AGE
spring-boot-cicd-demo-54657b864c-62lv9	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-fq2qc	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-fwqgr	1/1	Running	0	3h
spring-boot-cicd-demo-54657b864c-l47fk	1/1	Terminating	0	3h
spring-boot-cicd-demo-f98f5bf6f-5kqst	0/1	ContainerCreating	0	0s
spring-boot-cicd-demo-f98f5bf6f-d4d9x	1/1	Running	0	3s
spring-boot-cicd-demo-f98f5bf6f-fns7l	0/1	ContainerCreating	0	3s

- maxUnavailable이 1이므로 5개의 Pod 중 4개가 Running

- maxSurge가 1이므로 Pod의 개수 6개 (Terminating 제외)

- 마찬가지로 Pod의 총 개수, Running 중인 Pod의 개수 유지

- Pod의 AGE를 개수를 비교해보면 이전 버전(Blue)의 Pod가 점차적으로 종료되고 신규(Green) Pod가 생성됨

Table of Contents

I. 개요

II. CICD 설정

III. Pipeline 작성

IV. Rolling Update

V. AMF Pipeline

IV. AMF Pipeline

1. Pipeline 변경 사항
2. 고도화된 배포 전략
3. 기타

1. Pipeline 변경 사항

Kustomize 기반으로 kubernetes에 배포할 리소스(Deployment, Service, Configmap 등)를 자동화하여 이를 기반으로 배포하는 단계 추가


- 1) containerTemplate에 kustomize 활용을 위한 Container 추가
- 2) stage('Build K8S YAML') 로 Kubernetes 배포 리소스 생성 과정 단계 추가

```
12 timestamps {
13     podTemplate(label: label,
14         serviceAccount: "zcp-system-sa-${ZCP_USERID}",
15         containers: [
16             containerTemplate(name: 'maven', image: 'maven:3.5.2-jdk-8-alpine', ttyEnabled: true, command: 'cat'),
17             containerTemplate(name: 'docker', image: 'docker:17-dind', ttyEnabled: true, command: 'dockerd-entrypoint.sh', privileged: true),
18             containerTemplate(name: 'kubectl', image: 'lachlanevenson/k8s-kubectl:v1.18.2', ttyEnabled: true, command: 'cat'),
19             containerTemplate(name: 'kustomize', image: 'gauravgagani/k8s-kustomize:1.1.0', ttyEnabled: true, command: 'cat')
20         ],
```


```
47     stage('BUILD K8S YAML') {
48         container('kustomize') {
49             sh 'kustomize build --load_restructor none --enable_kyaml=false appstore/awesome-shopping/config-base-chaining/overlay/v1/dev/acc
50         }
51     }
52
53     stage('DEPLOY') {
54         container('kubectl') {
55             kubeCmd.apply file: 'deploy.yaml', wait: 300, recoverOnFail: false, namespace: K8S_NAMESPACE
56         }
57     }
```


2. 고도화된 배포 전략






Jenkins 기반의 기본 배포 전략 외에 Rollback, Bluen/Green, Canary 전략 등은 추가적인 개선이 필요함

 브랜치: master ▼

sam-zcp-lab / jenkins-pipeline

 **zzonsang** dd9f706f64 [init](#)

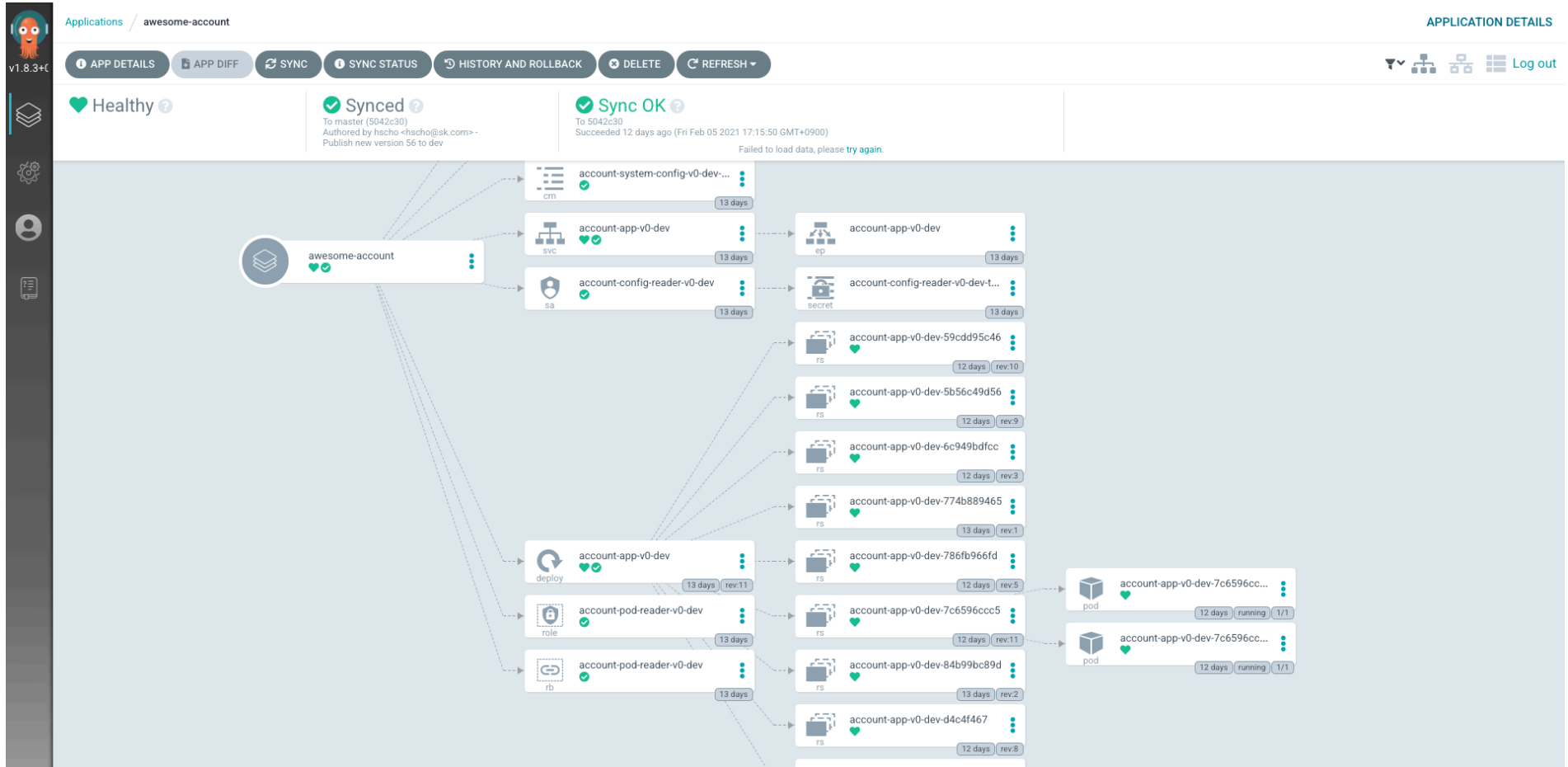
 ..

 canary-adjustment-pipeline	init
 canary-pipeline	init
 deploy-pipeline	init
 rollback-pipeline	init
 rolling-pipeline	init

3. 기타

Jenkins 기반에서 보다 선언적 배포 방식으로의 변화

- 1) Zcp 1.x 버전에서는 Jenkins 기반으로 CI/CD 제공
- 2) Zcp 2.x 버전에서는 Tekton, Argo CD 기반으로 CI/CD **준비 중..**



마지막 페이지