

# Service CI/CD pipeline 배포 설정 가이드

🕒 생성 일시	@2025년 1월 9일 오전 11:46
👤 작성자	① lev

서비스 Local Test 완료 후, Tencent Cloud의 TKE k8s에 배포하기 위한 배포 가이드입니다.

## 1. git-repository 설정

- dockerfile : Docker 이미지 빌드 과정을 자동화하는 설정 파일
- jenkinsfile\_prod : 각 환경에 맞춰서 prod/dev/beta로 네이밍
- \* 아래 jenkinsfile의 빨간색 글씨 수정

```
node {
    def app
    def project = "sl-bo-prod"
    def builder = "${currentBuild.getBuildCauses()[0].shortDescription} / ${currentBuild.getBuildCauses()[0].userId}"
    echo "PROJECT: ${project}"
    echo "BUILDER: ${builder}"
    echo "VERSION: ${env.VERSION}"

    notifyBuild('STARTED', project, builder)
    try {
        stage('Clone repository') {
            checkout scm
        }

        stage('Build image') {
            String currentDirectory = pwd()
            sh "sed -i 's+VERSION=.*+VERSION=${env.VERSION}+g' /var/jenkins_home/env/sl-bo/.env.production.local"
            sh "echo current directory = ${currentDirectory}"
            sh "cp /var/jenkins_home/env/sl-bo/.env.production.local ${current
```

```

Directory}"/"
    app = docker.build("ow-tcr.tencentcloudcr.com/ow-tke-tcr/${project}", "--build-arg NODE_ENV=production --build-arg PROJECT=${project}")
}

stage('Test image') {
    app.inside {
        sh 'echo "-----Tests passed-----"'
    }
}

stage('Push image') {
    docker.withRegistry('http://ow-tcr.tencentcloudcr.com', 'tencent-tcr') {
        app.push("${env.VERSION}")
    }
}

stage('Trigger ManifestUpdate') {
    echo "triggering update-manifest job"
    build job: 'update-manifest', parameters: [string(name: 'VERSION', value: env.VERSION), string(name: 'BUILDER', value: builder)]
}

currentBuild.result = 'SUCCESS'
} catch (Exception e) {
    currentBuild.result = 'FAILURE'
    throw e
} finally {
    notifyBuild(currentBuild.result, project, builder)
}
}

def notifyBuild(String buildStatus, String project, String builder) {
    buildStatus = buildStatus ?: 'UNKNOWN'
    def message = "*Project:* ${project}\n*Builder:* ${builder}\n*Version:* ${env.VERSION}\n*Status:* ${buildStatus}"
}

```

```

if (buildStatus == 'STARTED') {
    // 빌드 시작 알림
    slackSend(channel: '#25_cicd_alarm', color: 'warning', message: "Build started for ${project} by ${builder}")
} else if (buildStatus == 'SUCCESS') {
    // 빌드 성공 알림
    slackSend(channel: '#25_cicd_alarm', color: 'good', message: message)
} else if (buildStatus == 'FAILURE') {
    // 빌드 실패 알림
    slackSend(channel: '#25_cicd_alarm', color: 'danger', message: message)
}
}

```

## 2. jenkins web ui 설정

### Step.1 폴더 생성 (서비스명)





Dashboard > All > New Item

### New Item

Enter an item name

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

Select an item type

- 
**Freestyle project**  
 Classic, general-purpose job type that checks out from u  
 post-build steps like archiving artifacts and sending email
- 
**Pipeline**  
 Orchestrates long-running activities that can span multiple  
 known as workflows) and/or organizing complex activities
- 
**Multi-configuration project**  
 다양한 환경에서의 테스트, 플레폼 특성 빌드, 기타 등등 처럼 다수의
- 
**Folder**  
 Creates a container that stores nested items in it. Useful  
 a folder creates a separate namespace, so you can have  
 different folders.

### Step.2 서비스-deploy // update-manifest 생성

sl-bo-prod

상세 내용 입력

All +

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
✓	☀	sl-bo-prod-deploy	11 min #10	—	1 min 14 sec
✓	☀	update-manifest	9 min 54 sec #10	—	5.3 sec

The screenshot shows a table with columns S, W, Name, and 최근 성공. The first row is 'sl-bo-prod-deploy' with a green checkmark and a sun icon, and a success time of 28 min. The second row is 'update-manifest' with a green checkmark and a sun icon. A context menu is open over the 'sl-bo-prod-deploy' row, showing options: Changes, 파라미터와 함께 빌드, 구성 (highlighted), Pipeline 삭제, Move, Full Stage View, Stages, Rename, and Pipeline Syntax. Below the table, there are filters for 'icon: S M L'.

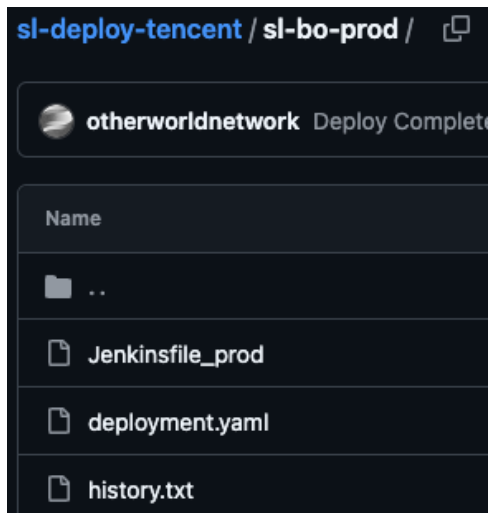
\* 각 설정 내역의 "config 구성" 을 보고 환경에 맞춰서 동일하게 설정 (추후에 gitops 코드로 변경 예정)

### 3. argo git-repository 설정

<https://github.com/ow-develop/sl-deploy-tencent>

- repository에 신규 설정할 서비스 생성

\* 신규 디렉토리 생성 및 아래와 같이 파일



```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

node {
    def app
    echo "BUILDER: ${env.BUILDER}"
    echo "VERSION: ${env.VERSION}"
    def TARGET = "sl-bo-prod"

    stage('Clone repository') {
        checkout scm
    }

    stage('Update GIT') {
        script {
            try {
                def date = new Date()
                def dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss (z Z)")
                def time = TimeZone.getTimeZone("Asia/Seoul")
                dateFormat.setTimeZone(time)
                today = dateFormat.format(date)
                echo today
            }
        }
    }
}
```

```

        withCredentials([usernamePassword(credentialsId: 'dev-git', passwordVariable: 'GIT_PASSWORD', usernameVariable: 'GIT_USERNAME')]) {
            sh "git config user.email admin@otherworld.network"
            sh "git config user.name otherworldnetwork"
            sh "cat './${TARGET}/deployment.yaml'"
            sh "sed -i 's+${TARGET}:+*+${TARGET}:${VERSION}+g' './${TARGET}/deployment.yaml'"
            sh "cat './${TARGET}/deployment.yaml'"
            sh "echo 'TIMESTAMP: ${today}, VERSION: ${VERSION}, BUILD_NUMBER: ${env.BUILD_NUMBER}, BUILDER: ${BUILDER}' >> './${TARGET}/history.txt'"
            sh "git add ."
            sh "git commit -m 'Deploy Complete [${TARGET}]: ${env.BUILD_NUMBER}'"
            sh "git push https://${GIT_USERNAME}:${GIT_PASSWORD}@github.com/ow-develop/sl-deploy-tencent.git HEAD:master"
        }
    } catch (err) {
        def errMsg = err.toString()

        echo errMsg

        throw err
    }
}
}
}
}

```

- kubernetes에 pod 배포할 deployment 수정
- \* 아래 빨간색 변경!! 필요에 따라 옵션 추가 및 변경 가능

```

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: sl-bo-prod
  name: sl-bo-prod

```

```

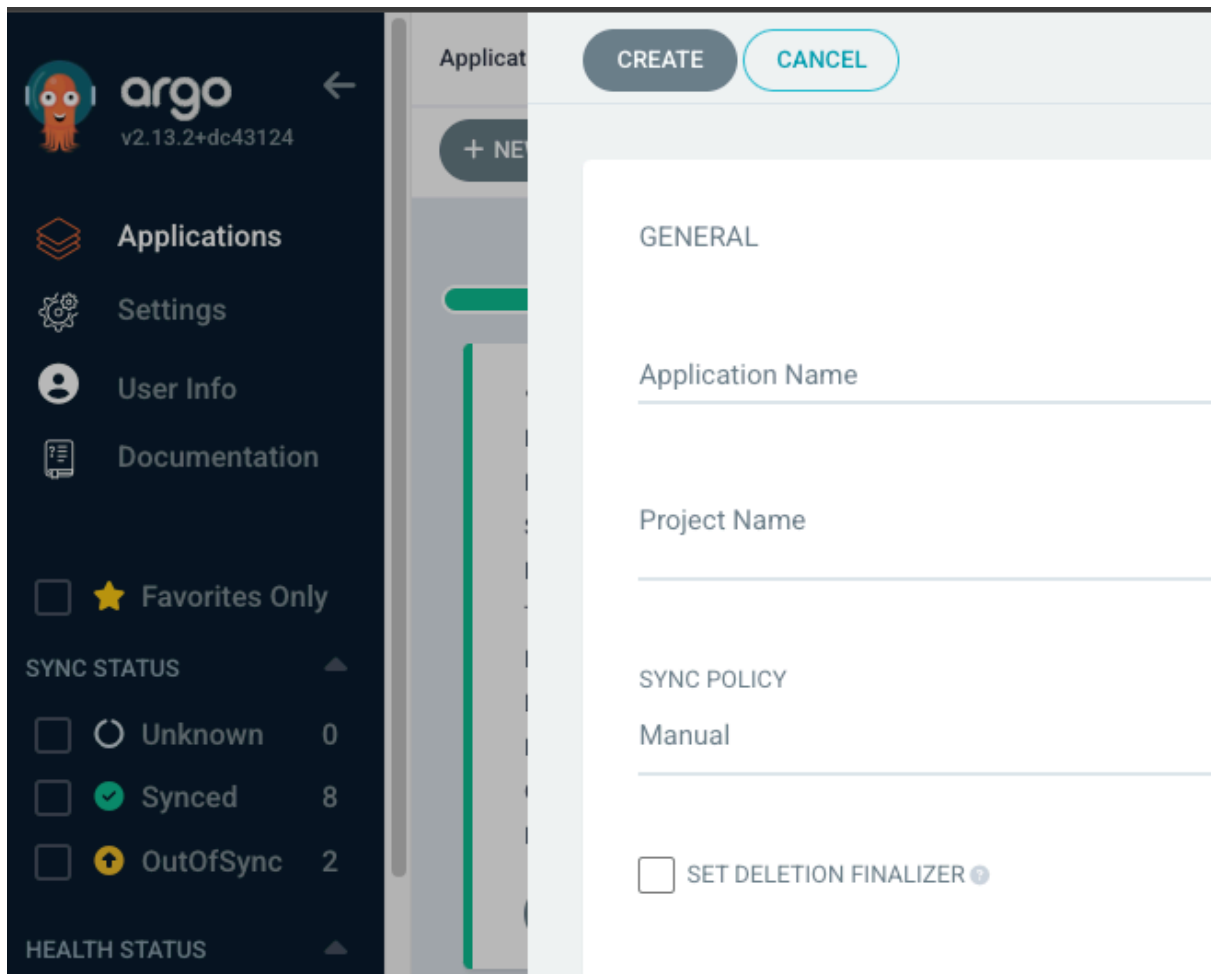
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sl-bo-prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
  template:
    metadata:
      labels:
        app: sl-bo-prod
    spec:
      containers:
        - image: ow-tcr.tencentcloudcr.com/ow-tke-tcr/sl-bo-prod:0.0.1 #처
          name: sl-bo-prod
          resources:
            limits:          # resource limit 설정
              cpu: 1
              memory: 2Gi
            requests:        # resource 정의
              cpu: 0.5
              memory: 1Gi
          ports:
            - containerPort: 3000 #실제 서비스 포트
      imagePullSecrets:
        - name: ow-tcr
status: {}
---
apiVersion: v1
kind: Service
metadata:
  name: sl-bo-prod-service
spec:
  ports:
    - port: 80

```

```
targetPort: 3000 #실제 서비스 포트
selector:
  app: sl-bo-prod
type: NodePort
```

#### 4. argocd web ui 설정

- New app을 눌러서 신규 생성 진행 (추후에 gitops 코드로 변경 예정)



The screenshot shows the ArgoCD web UI interface. On the left is a dark sidebar with the Argo logo and version 'v2.13.2+dc43124'. Below the logo are links for 'Applications', 'Settings', 'User Info', and 'Documentation'. Further down is a 'Favorites Only' toggle and a 'SYNC STATUS' section with three items: 'Unknown' (0), 'Synced' (8), and 'OutOfSync' (2). At the bottom is a 'HEALTH STATUS' section. The main area shows a 'CREATE' modal for a new application. The modal has a 'CREATE' button and a 'CANCEL' button. It contains fields for 'Application Name' and 'Project Name'. Below these is a 'SYNC POLICY' section with a 'Manual' option. At the bottom of the modal is a checkbox for 'SET DELETION FINALIZER'.

- application Name : 서비스명 (prod-sl-bo)
- Project Name : default (클릭하면 자동으로 보일것)
- Repository URL :  
[https://ghp\\_rVpZT4MG3IX3aaNrL0pxNwGvmVNFib1UfpEw@github.com/ow-develop/sl-deploy-tencent.git](https://ghp_rVpZT4MG3IX3aaNrL0pxNwGvmVNFib1UfpEw@github.com/ow-develop/sl-deploy-tencent.git)
- Path : ./path (sl-deploy-tencent 서비스 생성한 경로)
- Cluster URL : <https://kubernetes.default.svc> (클릭하면 자동으로 보일것)
- Namespace : default



## 5. CI/CD pipeline 배포 진행

- jenkins에 접근 후, 파라미터(version)를 이용해서 빌드 시작 (오류 발생 시, 콘솔 Log를 확인해서 대응)
- argocd에 접근 후, refresh → sync를 이용해서 배포 시작 (오류 발생 시, Log를 확인해서 대응)

## 6. Tencent cloud ingress

- TKE에 pod가 정상적으로 올라가면 ingress를 통해서 외부에서 접근 할 수 있도록 해줘야합니다.

경로 : Tencent Login → TKE → Cluster → service and route → Create → 신규 ingress 설정

Cluster-(Seoul) / sl-space-tke / Create Service and route

---

**Basic information**

Ingress name

Enter the ingress name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description

Up to 1000 characters

Namespace

default

Ingress type

Application CLB

Ngix Ingress Controller

Others

Detailed comparison

Application load balancer (supporting HTTP/HTTPS)

---

**CLB configurations**

Network type

Public network

Private network

Load Balancer

Automatic creation

Use existing

Automatically create a CLB for public/private network access to the service. The lifecycle of the CLB is managed by TKE. Do not manually modify the CLB listener created by TKE. [Learn more](#)

IP version

IPv4

IPv6 NAT64

The IP version cannot be changed later.

ISP type

BGP

\* 이걸 devops에게 요청! 실제 개발자가 작업하기엔 변수가 많음  
변수 → 도메인, SSL 인증서, CLB 설정, Security group 등 설정 추가 및 DNSpod Cname 등록