

CI/CD Pipeline Script Using Docker for Build and Deployment

Table of Contents

- [Description](#)
- [Problem Statement](#)
- [Prerequisites](#)
 - [Software Requirement](#)
 - [Hardware Requirement](#)
- [Implementation Steps](#)
 - [Step-1: Kubernetes Deployment with deployment.yaml](#)
 - [Step-2: Deploy to Kubernetes Cluster](#)
 - [Step-3: Verify Deployment](#)
- [References](#)

Description

This document explains how to create a **CI/CD pipeline** script using Docker. The script will automate the process of building, testing, and deploying a Dockerized application like **TodoApp** using common CI/CD tools.

Problem Statement

Manually building and deploying Docker applications can lead to human errors and delays. A well-designed CI/CD pipeline automates these tasks, ensuring that new code changes are automatically built, tested, and deployed in a consistent and reliable way.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-09) is required before proceeding with Lab Guide-10.

Software Required

- **Docker:** To containerize your application.
- **Kubernetes:** A running Kubernetes cluster.
- **kubectl:** To manage your Kubernetes clusters.
- **Git:** Version control system to store and trigger the CI/CD pipeline.
- **CI/CD Tools:**
 - **GitLab CI, GitHub Actions, or Jenkins** (select based on preference).
 - **A remote Docker registry** (such as Docker Hub or GitLab's built-in registry).
- **TodoAPP_MYSQL:** To download the source folder [click here](#)

Hardware Requirement

- Minimum of 4 GB RAM
- At least 2 cores in the processor

Implementation Steps

Step-1: Kubernetes Deployment with deployment.yaml

This section explains how to create and understand a `deployment.yaml` file for Kubernetes. This file defines your Kubernetes resources like pods and deployments.

Example `deployment.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todoapimysql
spec:
  replicas: 3
  selector:
    matchLabels:
      app: todomysql
  template:
    metadata:
      labels:
        app: todomysql
    spec:
      containers:
        - name: todoapimysql
          image: myusername/todoapimysql:1.0
          env:
            - name: MYSQL_HOST
              value: svcmysql.default.svc.cluster.local
          ports:
            - containerPort: 80
```

```
apiVersion: v1
kind: Service
metadata:
  name: svctodomysqlapi
spec:
  type: NodePort
  selector:
```

```

    app: todomysql
  ports:
    - port: 9098
      targetPort: 80

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql
          env:
            - name: MYSQL_DATABASE
              value: tododb
            - name: MYSQL_ROOT_PASSWORD
              value: P@ssw0rd
          ports:
            - containerPort: 3306

---

apiVersion: v1
kind: Service
metadata:
  name: svcmysql
spec:
  type: ClusterIP
  selector:
    app: mysql
  ports:
    - port: 3306
      targetPort: 3306

```

Note: Username should be replaced with docker hub username

Breakdown of `deployment.yaml`:

1. Todo API Deployment (`todoapimysql`):

- **apiVersion:** `apps/v1` – Specifies the API version for creating a deployment.
- **kind:** `Deployment` – Creates a deployment resource that ensures a specified number of pod replicas are running.
- **metadata:**
 - **name:** `todoapimysql` – The name of this deployment.
- **spec:**
 - **replicas:** `3` – This specifies that three replicas (pods) of the Todo API will be running.
 - **selector:**
 - **matchLabels:** `app: todomysql` – Identifies the pods to associate with this deployment using the `app: todomysql` label.
 - **template:** Defines the pod configuration.
 - **metadata:**
 - **labels:** `app: todomysql` – Labels the pod with `app: todomysql`, which must match the selector.
 - **spec:**
 - **containers:** Defines the container configuration.
 - **name:** `todoapimysql` – The name of the container.
 - **image:** `username/todoapimysql:1.0` – The container image to be used (in this case, the `username/todoapimysql` image with tag `1.0`).
 - **env:**
 - **name:** `MYSQL_HOST` – Sets an environment variable in the container to point to the MySQL host (`svcmysql.default.svc.cluster.local`), which is the MySQL service inside the Kubernetes cluster.
 - **ports:**
 - **containerPort:** `80` – The port on which the API container listens.

2. Todo API Service (`svctodomysqlapi`):

- **apiVersion:** `v1` – Specifies the API version for creating a service.
- **kind:** `Service` – Creates a service resource to expose the Todo API deployment.
- **metadata:**
 - **name:** `svctodomysqlapi` – The name of this service.
- **spec:**
 - **type:** `NodePort` – The service will be accessible via a node port (accessible outside the cluster).
 - **selector:**
 - **app:** `todomysql` – This service is linked to the pods that have the `app: todomysql` label.
 - **ports:**
 - **port:** `9098` – The port exposed by the service.
 - **targetPort:** `80` – The port on the container where traffic is forwarded (matches the Todo API container's `containerPort`).

3. MySQL Deployment (mysql):

- **apiVersion:** `apps/v1` – Specifies the API version for creating the deployment.
- **kind:** `Deployment` – Creates a deployment resource for the MySQL database.
- **metadata:**
 - **name:** `mysql` – The name of the deployment.
- **spec:**
 - **replicas:** `1` – This specifies a single MySQL pod replica.
 - **selector:**
 - **matchLabels:** `app: mysql` – Selects pods labeled with `app: mysql`.
 - **template:**
 - **metadata:**
 - **labels:** `app: mysql` – Labels the MySQL pod as `app: mysql`.
 - **spec:**
 - **containers:** Defines the container configuration.
 - **name:** `mysql` – The name of the MySQL container.
 - **image:** `mysql` – Specifies the MySQL image to be used.
 - **env:** Sets the environment variables for the MySQL container.
 - **MYSQL_DATABASE:** `tododb` – The name of the database to be created.
 - **MYSQL_ROOT_PASSWORD:** `P@ssw0rd` – The root password for the MySQL database.
 - **ports:**
 - **containerPort:** `3306` – The port the MySQL container listens on (standard MySQL port).

4. MySQL Service (svcmysql):

- **apiVersion:** `v1` – Specifies the API version for creating a service.
- **kind:** `Service` – Defines a service resource for the MySQL deployment.
- **metadata:**
 - **name:** `svcmysql` – The name of the service for MySQL.
- **spec:**
 - **type:** `ClusterIP` – Makes the MySQL service available only within the cluster.
 - **selector:**
 - **app:** `mysql` – Targets pods with the `app: mysql` label (the MySQL deployment).
 - **ports:**
 - **port:** `3306` – The port exposed by the service.
 - **targetPort:** `3306` – The port on the MySQL container.

Step-2: Deploy to Kubernetes Cluster

To deploy your application to a Kubernetes cluster using the `deployment.yaml` file:

```
kubectl apply -f deployment.yaml
```

or non-interactively:

```
kubectl apply -y deployment.yaml
```

```
Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ kubectl apply -f deployment.yaml
deployment.apps/todoapp-deployment created
service/todoapp-service created
```

Step-3: Verify Deployment

To verify your application to a Kubernetes cluster using the `deployment.yaml` file:

```
kubectl get pods
```

```
Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mysql-9d4fddb9f-thzwd	0/1	ContainerCreating	0	16s
todoapimysql-5fb66c66b-7f9rt	0/1	ContainerCreating	0	16s
todoapimysql-5fb66c66b-8ffkw	1/1	Running	0	16s
todoapimysql-5fb66c66b-qh5fv	1/1	Running	0	16s

```
kubectl get svc
```

```
Administrator@8305453e5e15576 MINGW64 ~/Downloads/ToDoApp_MySQL-main/ToDoApp_MySQL-main
$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	328d
svcmysql	ClusterIP	10.99.112.159	<none>	3306/TCP	39s
svctodomysqlapi	NodePort	10.101.158.83	<none>	9098:31811/TCP	39s
todoapp-service	LoadBalancer	10.111.53.76	localhost	8081:32359/TCP	10m

References

- Docker Documentation: <https://docs.docker.com/>
- GitLab CI Documentation: <https://docs.gitlab.com/ee/ci/>
- Docker Hub: <https://hub.docker.com/>
- Jenkins Documentation: <https://www.jenkins.io/doc/>