Integrating Jenkins with Kubernetes (K8s)

Integrating Jenkins with Kubernetes (K8s) enables powerful CI/CD pipelines by allowing Jenkins to orchestrate and automate the deployment of applications within a Kubernetes cluster. This integration helps manage containerized applications efficiently, enabling continuous integration and continuous delivery workflows.

Overview

Jenkins: Jenkins is an open-source automation server that helps automate parts of software development, including building, testing, and deploying applications.

Kubernetes (K8s): Kubernetes is an open-source container orchestration platform that automates deploying, scaling, and managing containerized applications. It provides a platform for automating deployment, scaling, and operations of application containers across clusters of hosts.

By integrating Jenkins with Kubernetes, you can:

- Dynamically provision Jenkins agents as Kubernetes pods.
- Deploy applications to Kubernetes directly from Jenkins pipelines.
- Manage Kubernetes resources using Jenkins jobs.

Step-by-Step Setup

1. Prerequisites

- **Ubuntu machine**: A server or VM with Ubuntu installed.
- **Jenkins installed**: Jenkins must be installed and running. (Refer to the previous setup if Jenkins is not installed).
- **Kubernetes cluster**: A running Kubernetes cluster. Minikube or MicroK8s can be used for local development, while managed Kubernetes services (like EKS, AKS, GKE) can be used for production.

2. Install Jenkins on Ubuntu

If Jenkins is not installed, you can follow these instructions:

1. **Update System Packages**:
   ```
   sudo apt update
   sudo apt upgrade -y
   ```

2. **Install Java**:
   Jenkins requires Java to run. Install OpenJDK:
   ```
   sudo apt install openjdk-11-jdk -y
   ```

3. **Add Jenkins Repository and Install Jenkins**:
   ```
   wget -q -O -
   https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
   apt-key add -
   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
   binary/ > /etc/apt/sources.list.d/jenkins.list'
   sudo apt update
   sudo apt install jenkins -y
   ```

4. **Start and Enable Jenkins Service**:
   ```
   sudo systemctl start jenkins
   sudo systemctl enable jenkins
   ```

5. **Access Jenkins**:
   Open your web browser and go to
   `http://your_server_ip_or_domain:8080` to complete the Jenkins setup.

3. Install and Set Up Kubernetes on Ubuntu

For local testing and development, you can use Minikube or MicroK8s.

Install Minikube

1. **Install Minikube**:

   ```
   curl -LO
   https://storage.googleapis.com/minikube/releases/latest/min
   ikube-linux-amd64
   sudo install minikube-linux-amd64 /usr/local/bin/minikube
   ```

2. **Start Minikube**:
   ```
   minikube start --driver=docker
   ```

3. **Verify Kubernetes Cluster**:
   ```
   kubectl get nodes
   ```

4. Install and Configure `kubectl`

`kubectl` is the command-line tool for interacting with Kubernetes clusters.

1. **Install `kubectl`**:

   ```
   sudo apt-get update

   sudo apt-get install -y apt-transport-https gnupg2

   curl -s
   https://packages.cloud.google.com/apt/doc/apt-key.gpg |
   sudo apt-key add -

   echo "deb https://apt.kubernetes.io/ kubernetes-xenial
   main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

   sudo apt-get update

   sudo apt-get install -y kubectl
   ```

2. **Test `kubectl` Installation**:
   ```
   kubectl version --client
   ```

5. Install Jenkins Kubernetes Plugin

1. **Install the Kubernetes Plugin**:
   - Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins.
   - In the "Available" tab, search for "Kubernetes" and install the "Kubernetes" plugin.

2. **Restart Jenkins**:
   After the plugin is installed, restart Jenkins to apply the changes.
   `sudo systemctl restart jenkins`

6. Configure Jenkins to Use Kubernetes

1. **Configure Kubernetes Cloud in Jenkins**:
   - Go to Jenkins Dashboard → Manage Jenkins → Configure System.
   - Scroll down to "Cloud" and click "Add a new cloud" → Select "Kubernetes".
   - Configure the Kubernetes plugin:
     - **Name**: Give your Kubernetes cloud a name (e.g., "K8s").
     - **Kubernetes URL**: This is the URL of your Kubernetes cluster (if using Minikube, this can be obtained using `kubectl cluster-info`).
     - **Kubernetes Namespace**: Specify the namespace (default is `default`).
     - **Credentials**: Add Kubernetes credentials if required (for Minikube, it's usually not necessary).
2. **Test Connection**:
   - Click "Test Connection" to ensure Jenkins can connect to your Kubernetes cluster.
3. **Configure Kubernetes Pod Templates**:
   - Under the "Kubernetes" section, click "Add Pod Template".
   - **Name**: Give the pod template a name.
   - **Labels**: Specify a label that can be used to select this pod template (e.g., `jenkins-agent`).
   - **Containers**: Configure containers for your Jenkins agents. Add a container for Jenkins agent:
     - **Name**: e.g., `jnlp`.
     - **Docker Image**: e.g., `jenkins/inbound-agent`.
     - **Command to run**: `jenkins-slave`.
     - **Arguments to pass to the command**: `${computer.jnlpmac} ${computer.name}`.

7. Create a Jenkins Pipeline to Deploy on Kubernetes

1. **Create a New Pipeline Job**:

- Go to Jenkins Dashboard → New Item → Pipeline.
- Enter a name for the pipeline job and select "Pipeline".

2. **Define the Pipeline Script**:
   Here's a simple pipeline script that runs a command in a Kubernetes pod:

```
pipeline {
    agent {
        kubernetes {
            label 'jenkins-agent'
            yaml """
            apiVersion: v1
            kind: Pod
            metadata:
              labels:
                some-label: some-label-value
            spec:
              containers:
              - name: busybox
                image: busybox
                command:
                - cat
                tty: true
            """
        }
    }
    stages {
        stage('Run') {
            steps {
                container('busybox') {
                    sh 'echo Hello from Kubernetes pod!'
                }
            }
        }
    }
}
```

3. **Save and Run the Pipeline**:
    - Save the pipeline configuration.
    - Click "Build Now" to run the pipeline.

8. Verify the Integration

- **Check Pod Creation**: Use `kubectl get pods` to verify that a new pod was created in Kubernetes for your Jenkins agent.
- **Check Logs**: Review Jenkins logs to ensure the pipeline ran successfully and the pod executed the commands as expected.

Integrating Jenkins with Kubernetes on Ubuntu allows you to leverage Kubernetes' powerful orchestration capabilities for your CI/CD pipelines. By dynamically provisioning Jenkins agents on Kubernetes, you can efficiently manage resources and scale your builds. This setup is ideal for environments that heavily utilize containerized applications and need robust automation for continuous integration and deployment workflows.