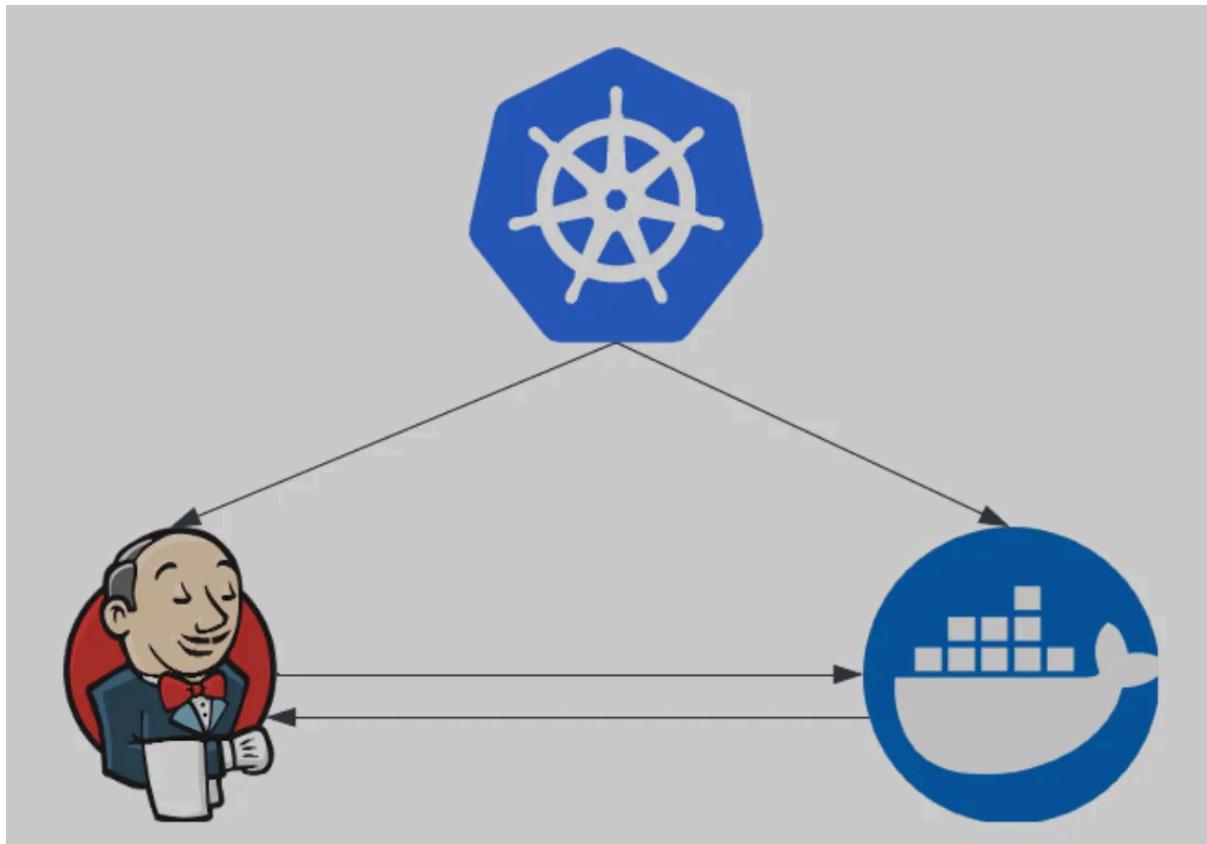


Deploying a Flask Application with Jenkins to a Kubernetes Cluster: CI/CD Pipeline Setup



Introduction

In this guide, you will learn how to automate the deployment of a Flask application using Jenkins, Docker, and Kubernetes in a CI/CD (Continuous Integration and Continuous Deployment) pipeline. This step-by-step tutorial covers everything from setting up Jenkins and Docker to configuring Jenkins, pushing the Dockerized Flask application to DockerHub, and deploying it on Kubernetes.

Prerequisites:

- A server with Ubuntu 24.04
- Basic knowledge of Linux commands is required
- Access to a Kubernetes cluster (you can use Minikube, Kind, or DigitalOcean Kubernetes).

Step 1: Installing Jenkins

First, install Jenkins on your Ubuntu server.

Install Java:

```
sudo apt update -y
sudo apt install openjdk-17-jdk -y
```

Install Jenkins:

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo "deb
[signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y
```

Start and Enable Jenkins:

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

Access Jenkins: Open your browser and navigate to <http://your-server-ip:8080>. Unlock Jenkins using the initial admin password:

The screenshot shows the Jenkins 'Unlock Jenkins' setup page. At the top left, there's a 'Getting Started' link. The main title is 'Unlock Jenkins'. Below it, a message states: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server: `/var/lib/jenkins/secrets/initialAdminPassword`'. It instructs the user to copy the password from either location and paste it into the 'Administrator password' field, which contains a series of dots ('.....'). A 'Continue' button is located at the bottom right.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Complete the setup by installing suggested plugins and creating the first admin user.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

The setup prompts to either **Install suggested plugins** or **Select plugins to install**. It's fine to simply install the suggested plugins

Getting Started

Getting Started

Folders Plugin	Formatter	Ant	Gradle
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle
✓ Pipeline	✓ GitHub Branch Source	✓ Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View
✓ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	✓ Mailer	✓ Folders Plugin
✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	✓ Timestamper

ep
** Pipeline: Declarative
Pipeline
** Java JSON Web Token (JWT)
** OkHttp
** GitHub API
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
** Pipeline Graph Analysis
** Pipeline: REST API
Pipeline: Stage View
Git
** - required dependency

The next step is to the **Create First Admin User**. Enter the credentials you want to use for your Jenkins administrator, then click **Save and Continue**.

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.452.1

Skip and continue as admin

Save and Continue

Once you specify the Jenkins URL, click **Save and Finish**

Getting Started

Instance Configuration

Jenkins URL:



The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

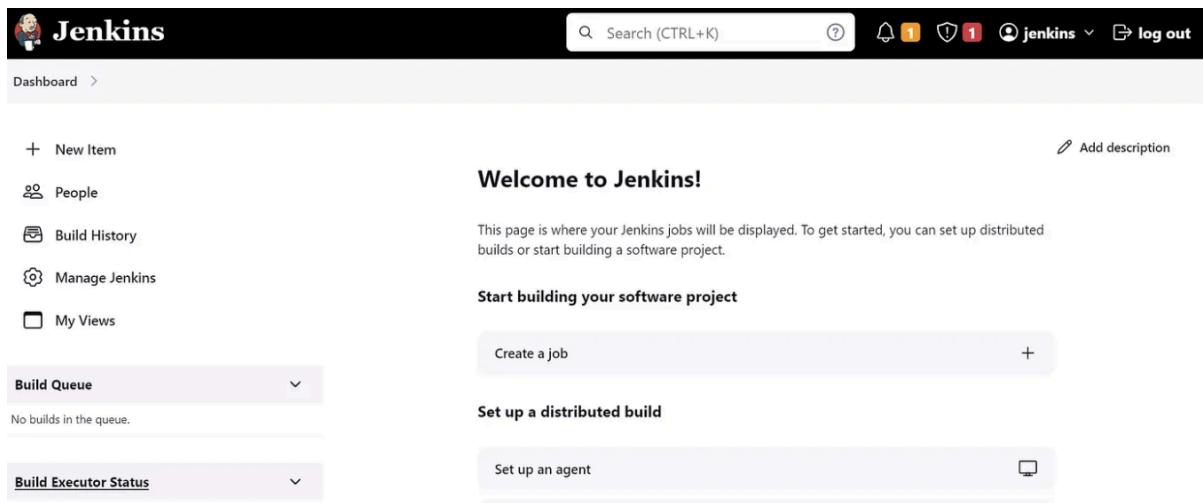


Jenkins 2.426.2

Not now

Save and Finish

You should see a page that says **Jenkins is ready!** Click **Start using Jenkins** to open the Jenkins dashboard



Step 2: Installing Docker

Next, we need Docker to build and manage our containerized applications.

Install Docker:

```
sudo apt-get update -y
sudo apt-get install ca-certificates curl -y
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo \
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update -y
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin -y
```

Add Jenkins User to Docker Group:

```
sudo systemctl enable docker
sudo usermod -aG docker jenkins
```

Step 3: Configuring Jenkins for Docker and Kubernetes

Now that Jenkins and Docker are installed, let's configure Jenkins for Docker and Kubernetes deployments.

Install Required Plugins:

Go to **Manage Jenkins > Manage Plugins** and install the following plugins:

- **Docker Pipeline**
- **Kubernetes Plugin**
- **Kubernetes Pipeline**
- **Git Plugin**

The screenshot shows the Jenkins Manage Plugins interface with a search bar at the top containing 'kubernetes'. Below the search bar, there is a table listing several Jenkins plugins. The table has two columns: 'Name' and 'Enabled'. The 'Name' column lists the plugin names, and the 'Enabled' column shows toggle switches indicating whether each plugin is currently enabled or disabled. Plugins listed include 'Kubernetes :: Pipeline :: DevOps Steps', 'Kubernetes CLI Plugin', 'Kubernetes Client API Plugin', 'Kubernetes Credentials Plugin', and 'Kubernetes plugin'. The 'Kubernetes :: Pipeline :: DevOps Steps' and 'Kubernetes plugin' entries are highlighted with red boxes around their respective rows.

Name	Enabled
Kubernetes :: Pipeline :: DevOps Steps 1.6	<input checked="" type="checkbox"/> Report an issue with this plugin
Kubernetes CLI Plugin 1.12.1	<input checked="" type="checkbox"/> Report an issue with this plugin
Kubernetes Client API Plugin 6.10.0-240.v57880ce8b_0b_2	<input checked="" type="checkbox"/> Report an issue with this plugin
Kubernetes Credentials Plugin 189.v90a_488b_d1d65	<input checked="" type="checkbox"/> Report an issue with this plugin
Kubernetes plugin 4285.v50ed5f624918	<input checked="" type="checkbox"/> Report an issue with this plugin

The screenshot shows the Jenkins Manage Plugins interface with a search bar at the top containing 'git'. Below the search bar, there is a table listing Jenkins plugins related to Git. The table has two columns: 'Name' and 'Enabled'. The 'Name' column lists the plugin names, and the 'Enabled' column shows toggle switches indicating whether each plugin is currently enabled or disabled. Plugins listed include 'Git plugin' and 'Git server Plugin'. Both are shown as enabled with blue toggles.

Git plugin 5.3.0	<input checked="" type="checkbox"/> Report an issue with this plugin
Git server Plugin 126.v0d945d8d2b_39	<input checked="" type="checkbox"/> Report an issue with this plugin

The screenshot shows the Jenkins Manage Plugins interface with a search bar at the top containing 'docker'. Below the search bar, there is a table listing Jenkins plugins related to Docker. The table has two columns: 'Name' and 'Enabled'. The 'Name' column lists the plugin names, and the 'Enabled' column shows toggle switches indicating whether each plugin is currently enabled or disabled. Plugins listed include 'Docker Commons Plugin' and 'Docker Pipeline'. The 'Docker Pipeline' entry is highlighted with a red box around its row.

Docker Commons Plugin 443.v921729d5611d	Provides the common shared functionality for various Docker-related plugins. Report an issue with this plugin
Docker Pipeline 580.vc0c340686b_54	Build and use Docker containers from pipelines. Report an issue with this plugin

Configure Docker Hub Credentials in Jenkins:

Go to **Manage Jenkins > Credentials > System > Global credentials (unrestricted)**

Select **Add Credentials** button

Kind: Username with password

Username: hbayraktar (your dockerhub username)

Password: Dockerhub password

New credentials

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
hbayraktar

Treat username as secret ?

Password ?
.....

ID ?
dockerhub-cred

Description ?

Create

Configure Kubernetes:

Generate secret for Kubernetes service account

Create a kubernetes service account

```
kubectl create serviceaccount jenkins
```

Create a role binding based on the permission needed by the application using below

```
cat <<EOF | kubectl create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins-integration
  labels:
    k8s-app: jenkins-image-builder
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
kind: ClusterRole
name: cluster-admin
subjects:
- kind: ServiceAccount
  name: jenkins
  namespace: default
EOF
```

Extract Service account token using kubectl

```
kubectl get secrets $(kubectl get serviceaccounts jenkins -o jsonpath='{.secrets[].name}') -o jsonpath='{.data.token}' | base64 -d
```

Create a secret for service account jenkins

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: jenkins
  annotations:
    kubernetes.io/service-account.name: jenkins
EOF
```

```
kubectl get secrets jenkins -o jsonpath='{.data.token}' | base64 -d
```

Add Kubernetes service account secret in Jenkins Credentials

Manage Jenkins > Credentials > then click on **global** and click on add credentials

Under **Kind**, scroll on the drop-down list and then choose **Secret text** Under secret, copy the Kubernetes token we generated earlier and paste it there. Then enter ID and description

The screenshot shows the Jenkins 'New credentials' configuration page. The 'Kind' dropdown is set to 'Secret text'. The 'Secret' field contains a long string of characters. The 'ID' field is set to 'SECRET_TOKEN'. The 'Description' field is empty. A 'Create' button is at the bottom.

Integrate Remote Kubernetes Cluster with Jenkins

Go to **Manage Jenkins > Clouds > Select New cloud** type it cloud name:

kubernetes-cloud press the create button

Name: kubernetes-cred

Kubernetes URL: Open your config file and enter server URL

Kubernetes URL: <https://db25c80a-3584-4759-9e48-baa6c16374a8.k8s.ondigitalocean.com>

Kubernetes server certificate key: You need to generate this key using this command

Convert kubernetes server certificate key to base64 format using below command

cat ~/.kube/config

```
-----CERTIFICATE-----
(base) flask-monitoring ➔ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVjUSUZJQ0FURS0tLS0tCk1JSURKeKDQwCrZ0F3SUJBZ01DQm5Vd0RRwUp
Lb1pJaHJzTKFRRUxCUUF3TxPvK1CTUdBMVFQ2hNTVJHbG4KVhSaGJF0WpaV0Z1TVJvd0dBWRWUVFERXhGck9ITmhZWE1nUTJ4MMMzUi
xjaUJEVRBZU3MH10REE0TwpFdwpOVEkxTLrsYUZ3MD80REE0TwpFd05USTFOVGxhTURNeEZU0VRCZ05WQkfFvVERFUuBaMmwWVd4UFky/
mhiakvhCk1CZ0dBMVFQXhNuMF6aHpZv0Z6SUV0c2RYTjBaWElnUTBFd2dnRwlNQTBHQ1NxR1NjYjNEUUVCQVFVQUE0SUIKRhdBd2dnRutB
b01CQVFDNG05WVY2MndSNUpFMFRwbXpNZ1duTFRtWedIZzRtbHdSTn1laU4zSu9hL0Rxcc3050gpOU2pneXZScXkwbgduVuU2Tugy0E1hREJ
3UWdwCHJQZWJENG1qZ05RUHNSm2pFYzZCS3rZkdZWG5sYWNUUUnaCkg4a3h6ZVJN0Wt5VnZYMtJUWg1U283bXlpendzNGtWv2xubkrUyK
NWSmPyZxLTBN0PTDzwK0hFZXR2K0NwVEYKYK5YUzB3S0EzZ1VweFN40FJ3MmwzS1E2RGYwL1pqvaE1wXQzNmVcbEY1L3kyeUhHzmQyMmhkZ
XRSWHJ2T3BIdApINUZSmvEWk5zdHNPaUYrUE8wZnBMcMNRnZJWhlyYkForlgxeFN3SVVvc0ZRRDh1WC9QZ2FYaUxJRG1NsjdYCh3TFIk
ZzNtQ1pDdmIzSE0xRlf1cGN4ZglT3BHVEhoxMNrxZkFnTUJBQUDqUlRCRE1BNEdBMVVKrHdFQi93UUUKQxdJQmhqQVNCZ05WSFJNQkFmOE/
DREFH0VFILOFnRUFN0jBHOTFVZEZRnUVdc0LE1NDdmXBRowXZwaLFHrmN3eOpJZ1U50zZPS0d60U5CZ2txaGtPzrL3MEJBUXNGQUFP00FRRU
FBShVNd2R3TUDi0nq4V1L50ENMcjbLUU9NwmdXcmZouVNkbVA4VDR1NLRncEk1WMN5b2x1TzhShnBHdjk5MjBzbFvxVUlhTnRnUzhwJm1R
GLERFVTZko5ewlBew8KZ3ZVWhVoNDI4VTRGeTRZ0LV2Zj0Q2Rga1ZY0upzdzFIV2ozWFpRa2UrNktGa1Q2T2lXNFcvRn1L13VLS3Q20QpB
ZnVsN0tjbkJJc3g3bkJUeXpGVXN3K2zWmjFJcHBMVWh0SDg5bxBbd1pZ2xPcdJ2Vm0bm9CSGE5MmRpVjdYCKJ6U0ViRG5yT0pHcmdJNER
4d2VuWnBxMkdEZ3BXZ25SalYwdzBYkv5c1JaVStNRkdUy3Vzd3BDV2gwMnpZT2QTKfEczlvakZqZjk2WUVTL2VrbUhXaGg1aEJlRloVZc
x5VlFB0VpSbfEem1VV EhCb1V0ZHf2dsxZz09C10tLS0tRu5E1EnFu1RJrk1D0VRFLS0tLS0K
  server: https://db25c80a-3584-4759-9e48-baa6c16374a8.k8s.ondigitalocean.com
  name: do-fra1-ibb-tech
contexts:
- context:
  cluster: do-fra1-ibb-tech
  user: do-fra1-ibb-tech-admin
  name: do-fra1-ibb-tech
  current-context: do-fra1-ibb-tech
```

```
echo -n
<contents_of_the_certificate-authority-data_entry_of_my_kubeconfig_file> | base64 --decode
```

```
(base) flask-monitoring ➜ echo "LS0tLS1CRUJDJtBDRVJUSUZJQ0FURS0tLS0tCk1JSURKekNDQWcrZ0f3SUJBZ0lD0m5v0d0RRwUpLb1pJaHzjTkFRRUxCUUF3TxpFvk1CTUdBMVFQ2hNTVJHbG4VKhSaGF0wpv0z1TVjvd0dWbJURwIvFPERXhck9ITmhZwE1nTJAmmMzlmxjaUjeJUVRBZU3MH10RE0tWpFdwpOVEkxT1RsYUZ3MD80RE0tWpFd05USTF0VGxhTURNeZL0VRCC05W0kFVERFUnBmwmwVd4UfkVmhniakvhCk1Cz0dBMVVFOXhNUmF6aHpfZV0Z6SuVoc2cRVTyBaWE1nUTBfd2dnRwIN0TBHQ1NkR1NjyjneUUVCQVFVQUE0sUIKRhdld2dnRUBb0lCQVFDMG05WVY2MndSNUpFMFrwPZN1zRtEd1zRtBhdStN1laU4zsu9hL0RpXc050gpQp2pneXZScXkb6duVnU2Tug0y0E1hREJ3UdwchJ02WJENG1qZ05RUHJSN2pFYzZCS3BrZkdZWG5sYNUUENaCkg4a3h6ZVJN0Wt5VnZMytWxubkruYXNWmpyZLTN09PTDwK0fFXR2K0NvYeKYKkSYUzB3S0E21Wefn40FJ3Mmwz51E2RGYwL1pqaVe1WXQzInVcbey1L3keyeHh1zQmhmhKZXRShjZdhNpaUyEBwZnBMcN0RnZJH1yKtRCE1BNEdBMVvKRhdFo193uuUK0xdJ0mhbQNCW5WSFJN0kFm0EVDFRFH0VFI0fRnuFnQjBH0TFVZERnUvdC0L1NdmsXBRwXwaiFHrmnRzeOpJ21U50ZPS0d60U5CZ2txaCtpRz13MEJBUJXNGQUPf0FRRFBShVwd2R3Tudio0qAV1l050EMcjbUu9WmdXcmZoUWnkbaV4VDRjN1Rncek1Wn5b2x17zsbnBhdjk5MjBzbfVxVUHTnRnUzwhUm1RGLERFV7ZkoewLpW8KZ3ZWhVoND120V4TRGETrZ0V1L2V2mJQ2RgaLZY0UpzdF2v0zWFpRa2UrnktGa1Q2T2LXNFcvRn1lIL30200pBzVsNtjbkJJc3g3bkJueXpGVXN3K2ZMjFjChBMWhOSdg5xb8Bd1pFZzXpCD2Vm0bm9CSGe5MnpVjyDCKJ6U0v1Rg5yT0phcmdJNER4d2vWnBxMkdE23BXZ55aLywdzbYbVks1JaVstNrkduY3Vzdb3BDV2gwMnpZT2QKTKFEczlvakZqZjk2UvTL2rVbuhxaG1eJ1RloVZkx5VLFBQvp5bfPfEem1VVHcjhV0ZHF2d2szz09C10tL50tRUE5IEINFU1RJRKdQVRLS0tL50K" | base64 --decode
-----BEGIN CERTIFICATE-----
MIIDJzCCAg+gAwIBAgICBnUwQYJKoZtHvcNA0ELBQAwMzEVMBMGa1UEChMMRGl
aXRhbE9jZWFmRowGAYDVQDDExFr0HNhYXMyQ2x1c3RlcIBDQTAeFw0yNDA4MjEw
NT1NTlaFw00ND4MjEwNT1NlaMDmxFTATBgvBaOTDrpZ2l0YWxPyVbhbEa
MBggA1UEAxMhZyWfz1ENSdXN0ZXIgQ0EwggEiMA0GCSqGSIb3DQEBAQJAA4IB
DwAwggEKAoIBAC04mUF62wR5JE0TpzmQgWnLTMxGh4mlWnVu1l3ToaQ-Dqst9B
PSigvYRqy0lgnVu6MH23MaBwQpprPebD4mjgNQPr7jcc6BkpkfGYXnlaTPCZ
H8kxeZ0N9kyVx2kIoh55o7myizws4kWlnmDnasVjirye5500L6p+Heetv+CpTf
bNxs0KA3gUVx8Rw213kQ60f0/Zj1o5Yt1febtF5/y2Hgf2dhetRxv0pHt
e5FwJe0Nts0ifP00fpLqnFvI7rbAhF1pD8X/Pgx1l1DmJ7X
HwLR1g3mCZcb3HM1F0upcdxm0pGTHh1skFAGjRTBDM4A41udwEB/wQE
AwIBhASBgnVHRMBAf8EcDAGQAH/AgEMB0gV1A1dgb0WBQ547fIpQyppj0GFcwY
IgU9C60KGzANBqkghkIG9w0BA0sFAAOCAQEAHuMwdMGbBtWYR8Clr0e0OMZg
fhQSdmP8T4e67gp15cyoLu08RnpGv9920yLuq0lGntgSpRs5D10dUSfJ9yiayo
gvUxuh42814Fy49YwbfcdFjVx93u1HjwZkZ0ke+6FKT601w4/FyhUkKt69
AfU15kcnBIsx7nBtzFUsw-fp21IppLuhNH9mpAwZog10p2vKtnoBha92diV7X
BzSebDnOrJGrgI4DxwenZpq2GdpqwlgnjV0w0XmYsRzU-MFGTcuspwpCWh02zYod
NADs9ojFj96YES/ekmHhn5hBeFz/flyVQAAZR1ZDzmUTHBn5tdqvwk1g=
-----END CERTIFICATE-----
```

Copy this base64 certificate to jenkins

kubernetes server certification key: Copy to echo result Kubernetes Namespace:default

Cloud kubernetes-cred Configuration

Name:

Kubernetes URL:

Kubernetes server certificate key:

Use Jenkins Proxy: ?

Disable https certificate check: ?

Kubernetes Namespace:

Credentials: secret_token

Press the **Test Connection** button for check kubernetes cluster connection

If your connection is successfully you should press the **Save** button

Disable https certificate check ?

Kubernetes Namespace
 default

Agent Docker Registry ?

Inject restricted PSS security context in agent container definition ?

Credentials
 secret_token

Connected to Kubernetes v1.30.2

WebSocket ?
 Direct Connection ?

⚠ 'TCP port for inbound agents' is disabled in Global Security settings. Connecting Kubernetes agents will not work without this or WebSocket mode!

Jenkins URL ?

Jenkins tunnel ?

Step 4: Flask-monitoring Application Structure

You can fork or clone this repository <https://github.com/hakanbayraktar/flask-monitoring.git>
git clone <https://github.com/hakanbayraktar/flask-monitoring.git>

```
flask-monitoring/
├── templates
│   └──index.html
├── app.py
├── Dockerfile
├── requirements.txt
├── Jenkinsfile
├── deployment.yaml
├── jenkins.sh
└── service.yaml
└── README.md
```

Dockerfile:

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
```

```
COPY . .
ENV FLASK_RUN_HOST=0.0.0.0
EXPOSE 5000
CMD [ "flask", "run"]
```

app.py

```
import psutil
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    cpu_metric = psutil.cpu_percent()
    mem_metric = psutil.virtual_memory().percent
    Message = None
    if cpu_metric > 70 or mem_metric > 70:
        Message = "High CPU or Memory Detected, scale up!!!"
    return render_template("index.html",
                           cpu_metric=cpu_metric, mem_metric=mem_metric,
                           message=Message)

if __name__=='__main__':
    app.run(debug=True, host = '0.0.0.0')
```

requirements.txt

```
Flask==2.2.3
MarkupSafe==2.1.2
Werkzeug==2.2.3
itsdangerous==2.1.2
psutil==6.0.0
plotly==5.5.0
tenacity==8.0.1
boto3==1.34.151
kubernetes==10.0.1
```

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>System Monitoring</title>
    <script
src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <style>
        .plotly-graph-div {
            margin: auto;
            width: 50%;
            background-color: rgba(151, 128, 128, 0.688);
            padding: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>System Monitoring</h1>
        <div id="cpu-gauge"></div>
        <div id="mem-gauge"></div>
        {% if message %}
            <div class="alert alert-danger">{{ message }}</div>
        {% endif %}
    </div>
    <script>
        var cpuGauge = {
            type: "indicator",
            mode: "gauge+number",
            value: {{ cpu_metric }},
            gauge: {
                axis: { range: [null, 100] },
                bar: { color: "#1f77b4" },
                bgcolor: "white",
                borderwidth: 2,
                bordercolor: "#ccc",
                steps: [
                    { range: [0, 50], color: "#d9f0a3" },
                    { range: [50, 85], color: "#ffeb84" },

```

```
        { range: [85, 100], color: "#ff5f5f" }
    ],
    threshold: {
        line: { color: "red", width: 4 },
        thickness: 0.75,
        value: {{ cpu_metric }}
    }
}
};

var memGauge = {
    type: "indicator",
    mode: "gauge+number",
    value: {{ mem_metric }},
    gauge: {
        axis: { range: [null, 100] },
        bar: { color: "#1f77b4" },
        bgcolor: "white",
        borderwidth: 2,
        bordercolor: "#ccc",
        steps: [
            { range: [0, 50], color: "#d9f0a3" },
            { range: [50, 85], color: "#ffeb84" },
            { range: [85, 100], color: "#ff5f5f" }
        ],
        threshold: {
            line: { color: "red", width: 4 },
            thickness: 0.75,
            value: {{ mem_metric }}
        }
    }
};

var cpuGaugeLayout = { title: "CPU Utilization" };
var memGaugeLayout = { title: "Memory Utilization" };
```

```

        Plotly.newPlot('cpu-gauge', [cpuGauge],
cpuGaugeLayout);

        Plotly.newPlot('mem-gauge', [memGauge],
memGaugeLayout);

    </script>
</body>
</html>

```

Jenkinsfile

```

pipeline {
    agent any

    stages {
        stage('Git Cloning') {
            steps {
                echo 'Cloning git repo'
                git url:
'https://github.com/hakanbayraktar/flask-monitoring.git', branch: 'main'
            }
        }
        stage('Build Docker Image') {
            steps {
                echo 'Building the image'
                sh 'docker build -t flask-monitoring .'
            }
        }
        stage('Push to Docker Hub') {
            steps {
                echo 'Pushing to Docker Hub'
                withCredentials([usernamePassword(credentialsId:
'dockerhub-cred', usernameVariable: 'USER', passwordVariable: 'PASS')]) {
                    sh ''
                    echo "${PASS}" | docker login --username "${USER}"
--password-stdin
                    docker tag flask-monitoring ${USER}/flask-monitoring:latest
                    docker push ${USER}/flask-monitoring:latest
                    ...
                }
            }
        }
        stage('Integrate Remote k8s with Jenkins') {
            steps {

```

```
        withCredentials([string(credentialsId: 'secret_token', variable: 'KUBE_TOKEN')]) {
            sh '''
                curl -LO
                "https://storage.googleapis.com/kubernetes-release/release/v1.20.5/bin/linux/amd64/kubectl"
                chmod u+x ./kubectl
                export KUBECONFIG=$(mktemp)
                ./kubectl config set-cluster do-fra1-ibb-tech
                --server=https://db25c80a-3584-4759-9e48-baa6c16374a8.k8s.ondigitalocean.com
                --insecure-skip-tls-verify=true
                ./kubectl config set-credentials jenkins
                --token=${KUBE_TOKEN}
                ./kubectl config set-context default
                --cluster=do-fra1-ibb-tech --user=jenkins --namespace=default
                ./kubectl config use-context default
                ./kubectl get nodes
                ./kubectl apply -f service.yaml
                ./kubectl apply -f deployment.yaml
                ...
            '''
        }
    }
}
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-monitoring
  labels:
    app: flask-monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-monitoring
  template:
    metadata:
      labels:
        app: flask-monitoring
    spec:
      containers:
```

```

- name: flask-monitoring
  image: hbayraktar/flask-monitoring:latest
  ports:
    - containerPort: 5000

```

service.yaml

```

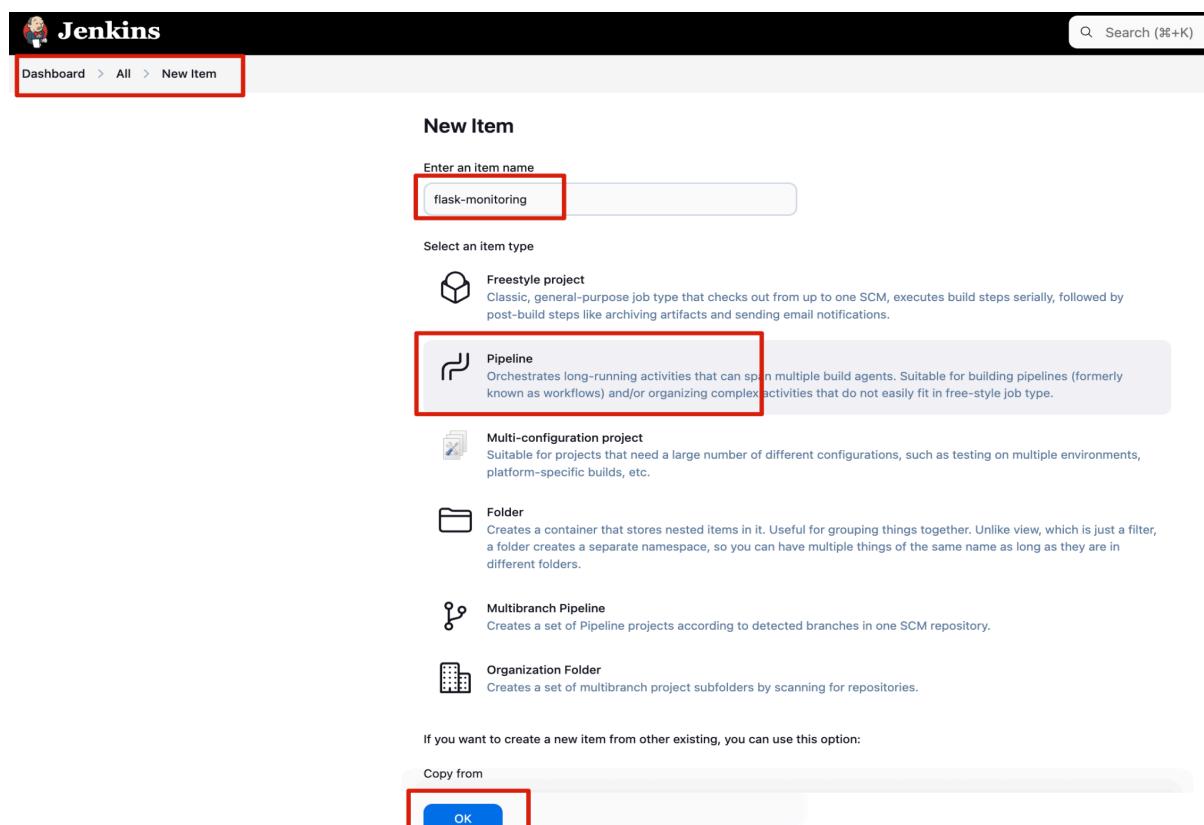
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: LoadBalancer
  selector:
    app: flask-monitoring
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000

```

Step 5: Jenkins Pipeline Configuration

Select **New Item** button for create a new job. **Enter an Item Name:** flask-monitoring

Select **Pipeline** and press the **OK** button



Select **flask-monitoring** job and on the left menu find **Advanced Project Options**:

Definitions: Select pipeline script from SCM

SCM: Git

Repository URL: <https://github.com/hakanbayraktar/flask-monitoring.git>

Credentials: None

The screenshot shows the Jenkins Pipeline configuration page. The 'Definition' dropdown is set to 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git'. The 'Repository URL' field contains 'https://github.com/hakanbayraktar/flask-monitoring.git'. The 'Credentials' dropdown is set to '- none -'. The 'Branch Specifier (blank for "any")' field is empty. At the bottom, there are 'Save' and 'Apply' buttons.

Branch Specifier (blank for 'any'): */main

Script Path: Jenkinsfile

click on **Save** button

The screenshot shows the Jenkins Pipeline configuration page. The 'Branch Specifier (blank for "any")' field contains '*/main'. The 'Script Path' field contains 'Jenkinsfile'. The 'Lightweight checkout' checkbox is checked. At the bottom, there is a large red box highlighting the 'Save' button.

Step 6: Deploying on Kubernetes

The screenshot shows the Jenkins dashboard at <http://167.172.99.134:8080>. The main navigation bar includes links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A search bar and a user account dropdown are also present. The central area displays a table for the 'flask-monitoring' project, which has a status of 'N/A' for both 'Last Success' and 'Last Failure', and 'N/A' for 'Last Duration'. Below the table, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). A large green arrow icon is visible on the right side of the table row.

The second part of the screenshot shows the Jenkins pipeline interface for the 'flask-monitoring' project. The left sidebar contains actions: 'Status' (selected), 'Changes' (disabled), 'Build Now' (highlighted with a red box), 'Configure', 'Delete Pipeline', 'Full Stage View', 'Stages', 'Rename', and 'Pipeline Syntax'. The main content area is titled 'Stage View' and displays a message: 'No data available. This Pipeline has not yet run.' Below this is a 'Permalinks' section. On the far left, a 'Builds' section shows 'No builds'.

Dashboard > flask-monitoring >

flask-monitoring

Status Changes Build Now

Configure Delete Pipeline Full Stage View Stages Rename Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~19s)

Declarative: Checkout SCM	Git Cloning	Build Docker Image	Push to Docker Hub	Integrate Remote k8s with Jenkins
1s	885ms	2s	8s	4s
1s	885ms	2s	8s	4s

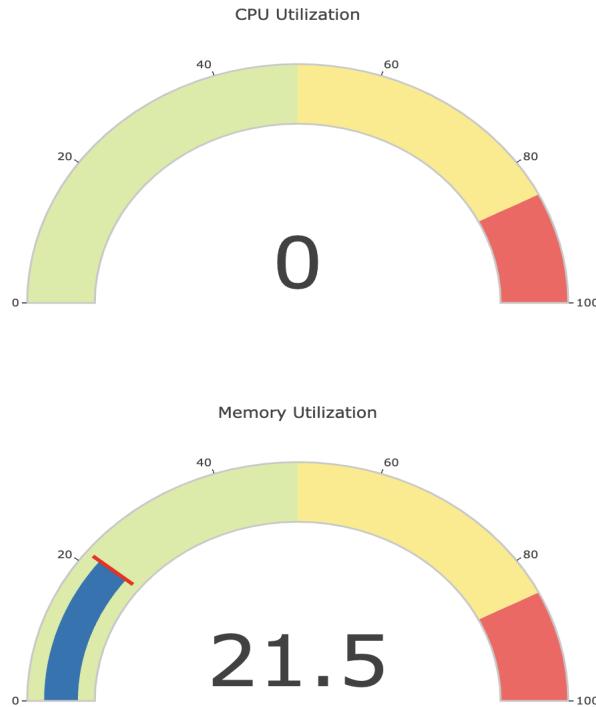
Permalinks

Builds ...

No builds

← → ⚡ Not Secure http://209.38.178.173

System Monitoring



Conclusion

By following this guide, you've set up a CI/CD pipeline with Jenkins, Docker, and Kubernetes, fully automating the process of building, testing, and deploying a Flask application. This setup can be easily adapted for other types of applications or services, making it a powerful tool for DevOps practices.