

DevOps CI/CD Pipeline using Jenkins, Docker, and Kubernetes

Project Objective

Design and automate a robust CI/CD pipeline to deploy a multi-tier web application (frontend and backend) using:

1. **Jenkins** for Continuous Integration and Continuous Deployment
2. **Docker** for containerization
3. **Kubernetes** for orchestration (via **Minikube**)
4. **AWS EC2** (t2.medium) as the host environment
5. **GitHub** as the code repository

The goal is to build a reliable, production-ready DevOps workflow.

Hosting Environment

1. **Cloud Platform:** AWS EC2
2. **Instance Type:** t2.medium (2 vCPU, 4 GB RAM)
3. **Operating System:** Ubuntu 22.04 LTS
4. **Security Group Ports Opened:** 22 (SSH), 8080 (Jenkins), 30080 (Backend), 30081 (Frontend)
5. **Minikube Tunnel:** Enabled using `minikube tunnel --bind-address 0.0.0.0`

Tools & Technologies

Tool	Purpose
GitHub	Version Control
Docker	Containerize frontend/backend apps
Jenkins	Automate CI/CD pipeline
Kubernetes	Manage containerized workloads
Minikube	Local Kubernetes cluster
Nginx	Frontend static file server
Flask	Python backend web server

Project Structure

DevOps-CI-CD-Pipeline-Using-Jenkins-Docker-and-Kubernetes/

```
|— backend/
|   |— Dockerfile
|   |— app.py
|   └— requirements.txt
|— frontend/
|   |— Dockerfile
|   └— index.html
|— k8s/
|   |— backend-deployment.yaml
|   |— frontend-deployment.yaml
|   └— service.yaml
|— Jenkinsfile
|— docker-compose.yml
└— README.md
```

Setup and Installation

Install Required Tools

→ `sudo apt update && sudo apt install -y git curl docker.io`

→ `sudo usermod -aG docker $USER`

Install Minikube:

→ `curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64`

→ `sudo install minikube-linux-amd64 /usr/local/bin/minikube`

Install kubectl:

→ `curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"`

→ `chmod +x kubectl && sudo mv kubectl /usr/local/bin/`

Start Minikube:

→ minikube start --driver=docker

→ minikube tunnel --bind-address 0.0.0.0

Docker: Containerization

Backend Dockerfile

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install Flask==2.0.3 Werkzeug==2.0.3

COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]

Frontend Dockerfile

FROM nginx:alpine

COPY index.html /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

Build & Push Images

→ docker build -t tpathak21/devops-backend:latest ./backend

→ docker build -t tpathak21/devops-frontend:latest ./frontend

→ docker push tpathak21/devops-backend:latest

→ docker push tpathak21/devops-frontend:latest

Kubernetes Deployment

Backend Deployment

- **Kind:** Deployment
- **Name:** devops-backend
- **Replicas:** 2
- **Container Image:** tpathak21/devops-backend:latest
- **Container Port:** 5000
- **Label:** app: devops-backend

Purpose: Launches two backend container replicas managed by Kubernetes

Backend Service

- **Kind:** Service
- **Name:** devops-backend-service
- **Type:** NodePort
- **Selector:** app: devops-backend
- **Ports:**
 - Port: 5000
 - TargetPort: 5000
 - NodePort: 30080

Purpose: Exposes backend app outside the cluster on EC2 instance's IP:30080

Frontend Deployment

- **Kind:** Deployment
- **Name:** devops-frontend
- **Replicas:** 2
- **Container Image:** tpathak21/devops-frontend:latest
- **Container Port:** 80
- **Label:** app: devops-frontend
- **Purpose:** Launches two frontend container replicas

Frontend Service

- **Kind:** Service
- **Name:** devops-frontend-service

- **Type:** NodePort
- **Selector:** app: devops-frontend
- **Ports:**
 - **Port:** 80
 - **TargetPort:** 80
 - **NodePort:** 30081

Purpose: Exposes frontend app outside the cluster on EC2 instance's IP:30081

Apply Resources

→kubectly apply -f k8s/

→kubectly get pods -o wide

→kubectly get svc

Jenkins CI/CD Pipeline

Agent Declaration

- **Type:** any
- **Purpose:** Allows pipeline to run on any available Jenkins agent (node)

Environment Setup

- **DOCKERHUB_CREDENTIALS:**
 - Secure Jenkins credentials (ID: dockerhub-creds) used for DockerHub login

Stages

Stage 1: Checkout Code from GitHub

Clones the main branch from your GitHub repository:

<https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes>

Stage 2: Build Backend Docker Image

- Builds Docker image:
- **Image:** tpathak21/devops-backend:latest
- **Context:** ./backend

Stage 3: Build Frontend Docker Image

- Builds Docker image:
 - **Image:** tpathak21/devops-frontend:latest
 - **Context:** ./frontend

Stage 4: Login to DockerHub

- Authenticates Jenkins with DockerHub using securely stored credentials

Stage 5: Push Docker Images

- Pushes both frontend and backend images to DockerHub:
 - tpathak21/devops-backend:latest
 - tpathak21/devops-frontend:latest

Stage 6: Deploy to Kubernetes

- Uses kubectl to apply manifests for:
 - **Backend Deployment**
 - **Frontend Deployment**
 - **NodePort Services**
- Uses the Kubeconfig file located at:
/var/lib/jenkins/.kube/config

End Result

- Fully automated CI/CD pipeline from GitHub to DockerHub to a live Kubernetes cluster
- Runs seamlessly inside an AWS EC2 instance with Jenkins managing every stage

Output Validation

curl Test

→curl http://<EC2-PUBLIC-IP>:30080 # Backend

→curl http://<EC2-PUBLIC-IP>:30081 # Frontends

Jenkins Pipeline

All stages: Clone > Build > Push > Deploy should show as green

Outputs Images

```

ubuntu@ip-172-31-13-88:~/DevOps-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ cd k8s
ubuntu@ip-172-31-13-88:~/DevOps-Project$ kubectl apply -f k8s/backend-deployment.yaml
error: the path "k8s/backend-deployment.yaml" does not exist
ubuntu@ip-172-31-13-88:~/DevOps-Project$ ls
deployment.yaml service.yaml
ubuntu@ip-172-31-13-88:~/DevOps-Project$ cd ..
ubuntu@ip-172-31-13-88:~/DevOps-Project$ ls
Jenkinsfile README.md backend docker-compose.yml frontend k8s
ubuntu@ip-172-31-13-88:~/DevOps-Project$ cd ..
ubuntu@ip-172-31-13-88:~$ ls
DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes DevOps-Project minikube-linux-amd64
ubuntu@ip-172-31-13-88:~$ cd DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ ls
Jenkinsfile README.md backend docker-compose.yml frontend k8s
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED            STATUS            PORTS
8901042dff54       gcr.io/k8s-minikube/kicbase:v0.0.47   "/usr/local/bin/entr..." 4 hours ago       Up 4 hours       127.0.0.1:32768->2272/tcp, 127.0.0.1:32769->2376/tcp, 127.0.0.1:3
2770-50800/tcp, 127.0.0.1:32771->8443/tcp, 127.0.0.1:32443/tcp   minikube
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tpathak21/devops-frontend   latest             8303479b241d       20 minutes ago     48.2MB
tpathak21/devops-backend    latest             04bb30740f0d       2 hours ago        136MB
gcr.io/k8s-minikube/kicbase  v0.0.47            795ea6a69c6e       2 weeks ago        1.31GB
hello-world           latest             74cc54e27dc4       4 months ago       10.1kB
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ kubectl apply -f k8s/backend-deployment.yaml
kubectl apply -f k8s/service.yaml
deployment.apps/devops-backend created
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ kubectl apply -f k8s/frontend-deployment.yaml
deployment.apps/devops-frontend created
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ kubectl apply -f k8s/service.yaml
service/devops-backend-service created
service/devops-frontend-service created
ubuntu@ip-172-31-13-88:~/DevOps-CD-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes$ minikube service devops-frontend-service
|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|
| default | devops-frontend-service | 80 | http://192.168.49.2:31204 |
|-----|
Opening service default/devops-frontend-service in default browser...

```

Status

Changes

Console Output

Edit Build Information

Delete build '#11'

Timings

Git Build Data

Pipeline Overview

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

✓

Console Output

Download

Copy

View as plain text

```

Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/devops-pipeline
[Pipeline] {
[Pipeline] withCredentials
Masking supported pattern matches of $DOCKERHUB_CREDENTIALS or $DOCKERHUB_CREDENTIALS_PSW
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout Code from GitHub)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/devops-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes.git # timeout=10
Fetching upstream changes from https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
> git rev-parse refs/remotes/origin/main^[commit] # timeout=10
Checking out Revision d89d8d61990f4817ed0f481cad1a4088c62afb11 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f d89d8d61990f4817ed0f481cad1a4088c62afb11 # timeout=10
> git branch --set-upstream-to=origin/main # timeout=10

```

The screenshot displays the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, search bar, and various utility icons. The main header shows the path: EC2 > Security Groups > sg-0d1b339ff5a658469 - launch-wizard-3. A green success banner at the top indicates that the inbound security group rule was successfully modified.

sg-0d1b339ff5a658469 - launch-wizard-3

Details

Security group name launch-wizard-3	Security group ID sg-0d1b339ff5a658469	Description launch-wizard-3 created 2025-06-09T09:28:12.03Z	VPC ID vpc-019357692a9eea8b4
Owner 6205S8997567	Inbound rules count 5 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (5/5)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0e01d513c32a520ce	IPv4	Custom TCP	TCP	30080	0.0.0.0/0
-	sgr-0024bb45d25952dec	IPv4	Custom TCP	TCP	30081	0.0.0.0/0
-	sgr-0926dc1d5375bb9a1	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-092aa37cdbeea1315	IPv4	Custom TCP	TCP	8080	0.0.0.0/0
-	sgr-09681dfbcfb79ec8	IPv4	HTTP	TCP	80	0.0.0.0/0

Dashboard > devops-pipeline > #11

Status

Changes

Console Output

Edit Build Information

Delete build '#11'

Timings

Git Build Data

Pipeline Overview

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

#11 (Jun 9, 2025, 6:46:57 PM)

Started by user admin

This run spent:

- 11 ms waiting;
- 21 sec build duration;
- 21 sec total from scheduled to completion.

Revision: d89d8d61990f4617ed0f481cad1a4088c62afb11

Repository: https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes.git

refs/remotes/origin/main

No changes.

Add description

Keep this build forever

Started 4 min 22 sec ago

Took 21 sec

REST API

Jenkins 2.504.2

aws

Search

[Alt+S]

Asia Pacific (Mumbai)

pathakteja_21

```
=> naming to docker.io/tpathak21/devops-backend:latest

The push refers to repository [docker.io/tpathak21/devops-backend]

fdc4c733acb8: Pushed
f3dfdbd00ad8: Pushed
ac045498dc66: Pushed
7bde2b8801d7: Layer already exists
978f260c1369: Layer already exists
9b5482944372: Layer already exists
9ad43ba78452: Layer already exists
ace34d1d784c: Layer already exists
latest: digest: sha256:8d8edf654bc0e5f2ce74a12baae53431e2c015cae9ae4d53a4895fbc14356fb size: 1990
deployment.apps/devops-backend restarted
ubuntu@ip-172-31-13-88:~/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes/backend$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
devops-backend-56d76f4df4-4tgkl     1/1     Running   0           7s
devops-backend-56d76f4df4-hqjkw     1/1     Running   0           19s
devops-frontend-7c5c67dd-gkdnq      1/1     Running   1 (78m ago) 4h15m
devops-frontend-7c5c67dd-zqmbw      1/1     Running   1 (78m ago) 4h15m
ubuntu@ip-172-31-13-88:~/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes/backend$ kubectl get svc
NAME                                TYPE           CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
backend-service                    NodePort       10.106.218.220 <none>        5000:30080/TCP   4h32m
devops-backend-service             ClusterIP      10.107.155.149 <none>        5000/TCP         4h15m
devops-frontend-service            LoadBalancer  10.109.236.22 <pending>     80:31204/TCP     4h15m
kubernetes                         ClusterIP      10.96.0.1     <none>        443/TCP          8h
ubuntu@ip-172-31-13-88:~/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes/backend$ kubectl get deploy
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
devops-backend  2/2     2             2           4h15m
devops-frontend 2/2     2             2           4h15m
ubuntu@ip-172-31-13-88:~/DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes/backend$ |
```

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

TejPATHAK / DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

DevOps-CI-CD-Pipeline-using-Jenkins-Docker-and-Kubernetes

Public

Pin Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file

Add file

Code

TejPATHAK Update index.html 9796279 · 1 hour ago 16 Commits

backend

Update Dockerfile

yesterday

frontend

Update index.html

1 hour ago

k8s

Update service.yaml

2 hours ago

Jenkinsfile

Update-8 Jenkinsfile

2 days ago

README.md

Update README.md

2 days ago

docker-compose.yml

Initial commit

last week

README

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages


No packages published

Publish your first package


localhost:30081

CI/CD Pipeline Project

Automated Deployment using Jenkins, Docker & Kubernetes

Project Overview

This project demonstrates a fully automated CI/CD pipeline built with Jenkins, Docker, and Kubernetes running on an AWS EC2 instance.

Tech Stack

Jenkins

Docker

Kubernetes


Minikube

AWS EC2

GitHub


HTML

Flask


Live Endpoints

• Frontend: <http://YOUR-EC2-IP:30081>

• Backend: <http://YOUR-EC2-IP:30080>

GitHub Repository

[View on GitHub](#)

Author

Developed by Tejaswi Pathak
tejaswipathak39@gmail.com

© 2025 DevOps Project by Tejaswi Pathak

dockerhub


Explore

My Hub

Search Docker Hub

CtrlK

T

tpathak21
Docker Personal

Repositories

Collaborations

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories

All repositories within the tpathak21 namespace.

Search by repository name

All content

Create a repository

Name	Last Pushed	Contains	Visibility	Scout
tpathak21/devops-frontend	about 4 hours ago	IMAGE	Public	Inactive
tpathak21/devops-backend	about 4 hours ago	IMAGE	Public	Inactive
tpathak21/nginx-proxy	12 days ago	IMAGE	Public	Inactive
tpathak21/nginx-reverse-proxy	about 1 month ago	IMAGE	Public	Inactive
tpathak21/tejaswi_web	4 months ago	IMAGE	Public	Inactive
tpathak21/productpage-app	5 months ago		Public	Inactive
tpathak21/images	about 1 year ago		Public	Inactive

1-7 of 7

Links

GitHub Repository: [DevOps-CI-CD-Pipeline](#)

LinkedIn Profile: [Tejaswi Pathak](#)

DockerHub Profile: [tpathak21](#)

Conclusion

This project is a full-fledged demonstration of how to containerize applications, automate deployments using Jenkins, and orchestrate containers via Kubernetes - all deployed on an AWS EC2 instance. It proves your ability to integrate multiple DevOps tools in a production-like setup.