# DevSecOps CI/CD Pipeline Implementation — Flask + Helm + Jenkins

This document details the setup and implementation of a secure CI/CD pipeline for a Flask-based application.  
It leverages Jenkins, Docker, Helm, and Kubernetes with integrated DevSecOps tools for scanning, testing, and deployment.

## 1. Overview

The pipeline automates build, test, security scanning, and deployment tasks. Key integrations include:  
- TruffleHog: Secret scanning  
- tfsec: Infrastructure-as-Code (IaC) security analysis  
- pytest: Unit testing  
- Trivy: Image vulnerability scanning  
- Helm: Atomic deployment and rollback

**Architecture – Network**

A screenshot of a computer

AI-generated content may be incorrect.

Jenkins Pipeline flow

A diagram of a process

AI-generated content may be incorrect.

### Repository Structure

**flask-cicd/**

**├── app/**

**│ ├── app.py**

**│ ├── requirements.txt**

**│ ├── Dockerfile**

**│ └── tests/**

**│ ├── \_\_init\_\_.py**

**│ └── test\_app.py**

**│**

**├── helm/**

**│ └── py-app/**

**│ ├── Chart.yaml**

**│ ├── values-staging.yaml**

**│ ├── values-production.yaml**

**│ └── templates/**

**│ ├── deployment.yaml**

**│ ├── service.yaml**

**│ └── ingress.yaml**

**│**

**├── Jenkinsfile**

**└── README.md**

**File tree**

**A screenshot of a computer

AI-generated content may be incorrect.**

## 2. Jenkins Pipeline Stages

* Checkout: Pulls source code from Git.
* TruffleHog - Secret Scan: Detects exposed secrets.
* tfsec - IaC Security Scan: Identifies Helm/K8s misconfigurations.
* Unit Tests: Validates Flask app functionality.
* Build Docker Image: Builds and tags the image.
* Trivy Scan: Checks vulnerabilities in image.
* Push to Registry: Uploads image to local Docker registry.
* Helm Deploy (Atomic): Safely deploys to Kubernetes.
* Post Deploy Check: Verifies pod and service status.
* Archive Reports: Stores all scan/test artifacts.

A screenshot of a computer

AI-generated content may be incorrect.

## 4. Security Tools & Purpose

|  |  |  |
| --- | --- | --- |
| Tool | Purpose | Pipeline Stage |
| TruffleHog | Secret scanning | Early stage |
| tfsec | IaC misconfiguration scanning | Before build |
| pytest | Unit testing | Testing stage |
| Trivy | Container image vulnerability scan | Before push |
| Helm | Atomic deployment | Deployment stage |
|  |  |  |

## 4. Jenkinsfile Stages

Git Checkout

A screenshot of a computer

AI-generated content may be incorrect.

Each stage includes corresponding shell commands and security scanning logic.  
Below is an example of how the TruffleHog and tfsec stages are configured.

# TruffleHog – SecretScan

A screenshot of a computer

AI-generated content may be incorrect.

stage('TruffleHog - Secret Scan') {  
 steps {  
 sh '''  
 docker run --rm -v $(pwd):/repo ghcr.io/trufflesecurity/trufflehog:latest filesystem /repo --fail --json > trufflehog-report.json || true  
 '''  
 }  
 post {  
 always {  
 archiveArtifacts artifacts: 'trufflehog-report.json', fingerprint: true  
 }  
 }  
}

# tfsec – Helm chart scanning

A screenshot of a computer

AI-generated content may be incorrect.  
stage('tfsec - IaC Security Scan') {  
 steps {  
 sh '''  
 tfsec helm/py-app --format json --out tfsec-report.json --soft-fail || true  
 '''  
 }  
 post {  
 always {  
 archiveArtifacts artifacts: 'tfsec-report.json', fingerprint: true  
 }  
 }  
}

# Unit Testing

A screenshot of a computer

AI-generated content may be incorrect.

stage('Unit Tests') {

steps {

sh '''

echo ">>> Running unit tests..."

pip install -r app/requirements.txt pytest pytest-cov

export PYTHONPATH=$PYTHONPATH:$(pwd)/app

pytest app/tests --junitxml=pytest-report.xml --cov=app --cov-report=html

'''

}

}

# Building Docker Image

**A screenshot of a computer

AI-generated content may be incorrect.**

stage('Build Docker Image') {

steps {

dir('app') {

sh '''

echo ">>> Building Docker image..."

docker build -t ${REGISTRY}/${IMAGE\_NAME}:${BUILD\_NUMBER} .

'''

}

}

}

# Trivy – Docker Image Scanning

A screenshot of a computer

AI-generated content may be incorrect.

stage('Trivy Scan') {

steps {

sh '''

echo ">>> Running Trivy scan..."

trivy image --severity HIGH,CRITICAL --exit-code 1 ${REGISTRY}/${IMAGE\_NAME}:${BUILD\_NUMBER} || {

echo "Critical vulnerabilities found! Build failed.";

}

'''

}

}

A screenshot of a computer

AI-generated content may be incorrect.

Here in this part, I’ve set the build not fail even if it found vulnerabilities, can add exit 1 code here to skip the next build, but for documentation purposes, ive showing the vulnerability alone and keeping the pipeline as it is to complete entire builds steps.

# Pushing the Image to registry

A screenshot of a computer

AI-generated content may be incorrect.

stage('Push to Registry') {

steps {

sh '''

echo ">>> Pushing Docker image to ${REGISTRY}..."

docker push ${REGISTRY}/${IMAGE\_NAME}:${BUILD\_NUMBER}

'''

}

}

# Deployment via Helm (Atomic)

**Credential**: Created Secret on Jenkins called Kube-jenkins and used that to perform ssh activities to build and run helm charts

**Cluster**:

Rocky 9.3 – Control Plane Node

Rocky 9.3 – Worker Node 1

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

stage('Deploy via Helm (Atomic)') {

steps {

sshagent(['kube-master-ssh']) {

sh '''

echo ">>> Deploying via Helm (Atomic)..."

# Save workspace for reference

WORKSPACE\_DIR=$(pwd)

echo "Current Jenkins workspace: $WORKSPACE\_DIR"

# Copy Helm chart to Kubernetes master

ssh -o StrictHostKeyChecking=no rocky@172.31.86.230 "mkdir -p /home/rocky/deployments"

scp -o StrictHostKeyChecking=no -r $WORKSPACE\_DIR/helm/py-app rocky@172.31.86.230:/home/rocky/deployments/

# Deploy using Helm with atomic mode

ssh -o StrictHostKeyChecking=no rocky@172.31.86.230 "

helm upgrade --install py-app /home/rocky/deployments/py-app \

-n ${DEPLOY\_ENV} \

-f /home/rocky/deployments/py-app/values-${DEPLOY\_ENV}.yaml \

--set image.repository=${REGISTRY}/${IMAGE\_NAME} \

--set image.tag=${BUILD\_NUMBER} \

--atomic \

--wait \

--timeout 5m

"

'''

}

}

}

Deployment is managed using Helm with `--atomic` and `--wait` flags for rollback safety.  
If deployment fails, Helm automatically reverts to the previous working state.

## Artifact Collection

All important build artifacts and reports are archived using Jenkins’ artifact feature.   
This includes JSON, XML, and HTML reports from testing and security scans.

## 7. Reports and Auditability

- TruffleHog → trufflehog-report.json  
- tfsec → tfsec-report.json  
- pytest → Unit test XML report  
These reports can be archived, exported, or integrated into dashboards like Grafana or ELK.  
A screenshot of a computer

AI-generated content may be incorrect.

## 9. Possible Issue and Cause

| **Issue** | **Likely Cause** | **Recommended Fix** |
| --- | --- | --- |
| **TruffleHog taking too long** | Scans entire repo | Limit to key directories: filesystem app/ helm/py-app --fail --json |
| **Email sending blocked** | Org policy / antivirus | Use Slack or Teams notifications instead |
| **Reports not automatically visible** | Jenkins doesn’t parse JSON reports | Add publishHTML plugin or convert reports to HTML |
| **Unit test paths failing initially** | PYTHONPATH mismatch | Fixed, but ensure app/\_\_init\_\_.py exists |
| **tfsec output not used to fail builds** | --soft-fail ignores severity | Use --config-file to enforce severity thresholds |
| **Docker build permission denied** | Jenkins not in docker group | Add Jenkins user to Docker group and restart service |