

DevOps Project Report(Azure)

Project Title: Flask-based Application Deployment using Jenkins, Docker, Kubernetes, and Terraform

Submitted By: Avinash Rajaput

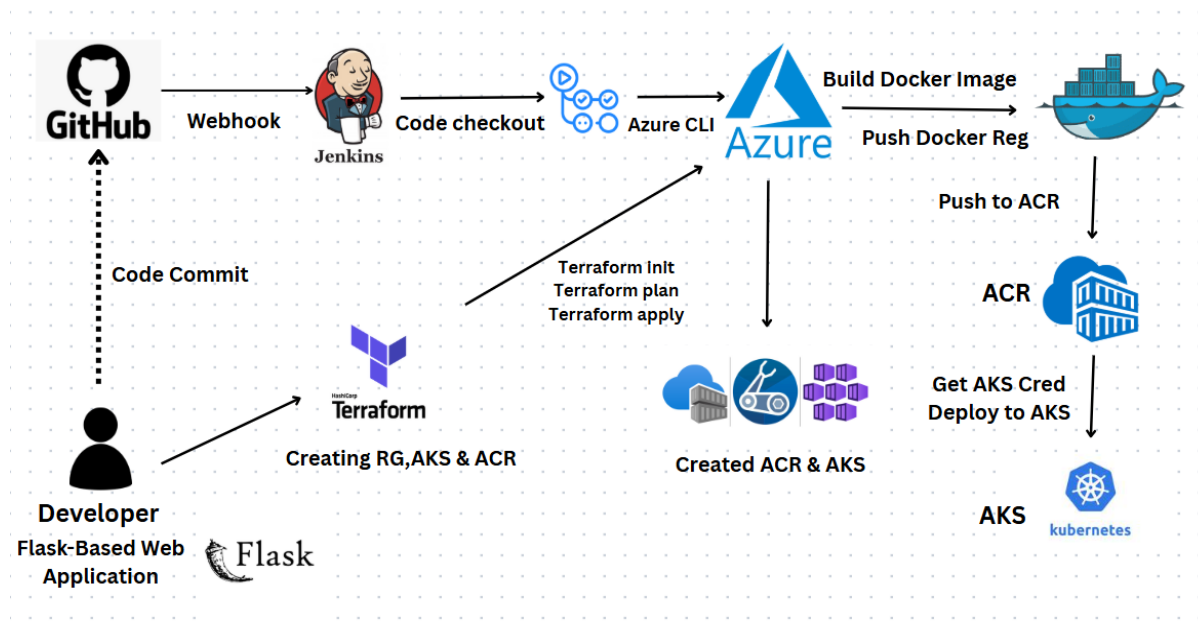
Project Assigned By: Hemanth Kumar

Date: 3 MAY

1. Project Overview :-

This project involves setting up a complete CI/CD pipeline to automate the deployment of a Flask-based web application using Jenkins, Docker, Azure Kubernetes Service (AKS), and Azure Container Registry (ACR). The infrastructure provisioning is handled via Terraform

2. Project Architecture :-



3. Project Objectives :-

- Automate code checkout, build, and deployment process.
- Containerize the Flask application using Docker.
- Push Docker images to Azure Container Registry.
- Deploy application to Azure Kubernetes Service.
- Manage infrastructure using Terraform scripts.

4. Project File Structure :-

File/Folder	Description
app.py	Main Flask application file.
requirements.txt	Python dependencies for the application.
Dockerfile	Instructions to build Docker image for Flask app.
Jenkinsfile	Pipeline script defining stages like build, test, push, Deploy to the container
flask_app.sh	Optional shell script for managing or testing the app.
k8s/	Contains Kubernetes YAML files (Deployment and Services)
terraform/	Contains Terraform configuration files for AKS and ACR provisioning.

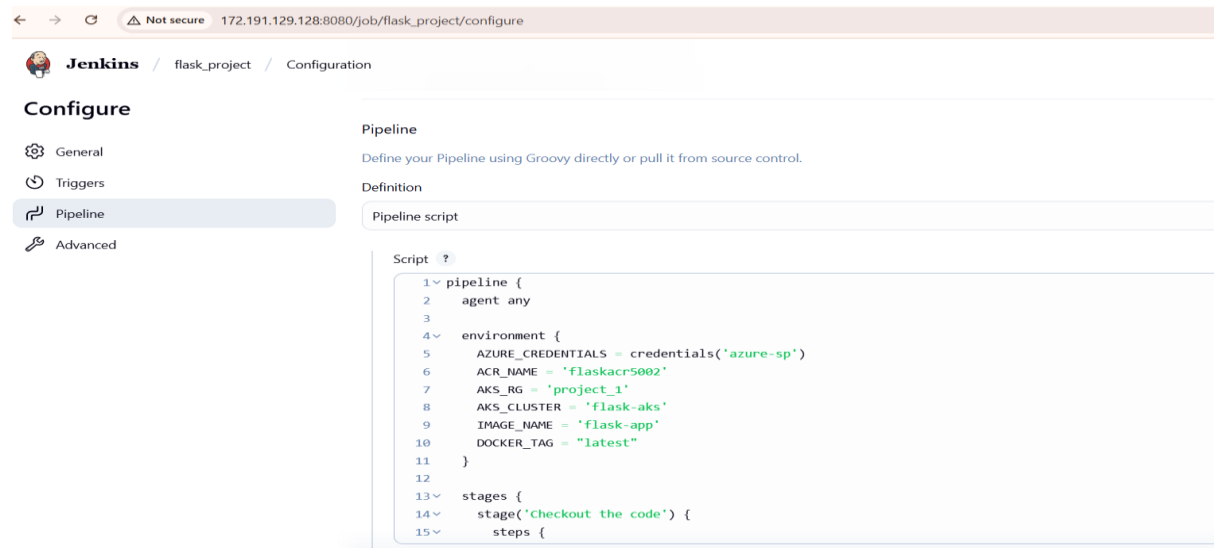
5. Technologies and Tools Used

- **Programming Language:** Python (Flask)
- **CI/CD Tool:** Jenkins
- **Containerization:** Docker
- **Container Registry:** Azure Container Registry (ACR)
- **Orchestration:** Azure Kubernetes Service (AKS)
- **Infrastructure as Code:** Terraform
- **Version Control:** GitHub

6. Pipeline Workflow

1. **Code Checkout** – Jenkins pulls the code from GitHub.
2. **Azure Login** – Authenticate using a Service Principal.
3. **Docker Build** – Build a Docker image of the Flask app.
4. **Push to ACR** – Push the built image to Azure Container Registry.
5. **Get AKS Credentials** – Fetch cluster context.
6. **Deploy to AKS** – Apply Kubernetes manifests to deploy the app.

Figure 1: Jenkins dashboard showing the configured pipeline for the Flask application.



Jenkinsfile

```
pipeline {
  agent any

  environment {
    AZURE_CREDENTIALS = credentials('azure-sp')
    ACR_NAME = 'flaskacr5002'
    AKS_RG = 'project_1'
    AKS_CLUSTER = 'flask-aks'
    IMAGE_NAME = 'flask-app'
    DOCKER_TAG = "latest"
  }

  stages {
    stage('Checkout the code') {
      steps {
        git branch: 'master', url: 'https://github.com/avi913/Flask_App.git'
      }
    }

    stage('Login to Azure') {
      steps {
        withCredentials([string(credentialsId: 'azure-sp', variable: 'AZ_CREDS')]) {
```

```

sh ""

echo $AZ_CREDS > azure.json

az login --service-principal --username $(jq -r .clientId azure.json) \
  --password $(jq -r .clientSecret azure.json) \
  --tenant $(jq -r .tenantId azure.json)

az account set --subscription $(jq -r .subscriptionId azure.json)

""}}}

stage('Build Docker Image') {
  steps {
    sh ""

    az acr login --name $ACR_NAME

    docker build -t $ACR_NAME.azurecr.io/$IMAGE_NAME:$DOCKER_TAG .

    ""}}

stage('Push to ACR') {
  steps {
    sh ""

    docker push $ACR_NAME.azurecr.io/$IMAGE_NAME:$DOCKER_TAG

    ""}}

stage('Get AKS Credentials') {
  steps {
    sh ""

    az aks get-credentials --resource-group $AKS_RG --name $AKS_CLUSTER --
    overwrite-existing" } }

stage('Deploy to AKS') {
  steps {
    sh ""

    kubectl apply -f k8s/deployment.yaml

    kubectl apply -f k8s/service.yaml

    ""}}}

post {
  success {

```

```

    echo 'Deployed successfully to AKS!'
}

failure {
    echo 'Deployment failed!'}}}

```

GitHub Repository :-

Figure: GitHub repository containing the source code and deployment files.

avi913 / Flask_App

Issues Pull requests Actions Projects Wiki Security Insights Settings

Flask_App Public

master 1 Branch 0 Tags

Go to file Add file Code

File	Commit Message	Commit Time
avi913 Update app.py	fd1a4d2 · 41 minutes ago	23 Commits
Terraform	terraform files	yesterday
k8s	Update deployment.yaml	2 days ago
Dockerfile	project files	2 days ago
Jenkinsfile	jenkins file	yesterday
app.py	Update app.py	41 minutes ago
flask_app.sh	script file added	20 hours ago
requirements.txt	adding requirement file	43 minutes ago

Jenkins Pipeline Execution :-

Figure: Jenkins pipeline execution log displaying successful code checkout and build steps.

Jenkins / flask_project / #11

```

[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to AKS)
[Pipeline] sh
+ kubectl apply -f k8s/deployment.yaml
deployment.apps/flask-app created
+ kubectl apply -f k8s/service.yaml
service/flask-service created
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Deployed successfully to AKS!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Jenkins / flask_project / #11

Status

</> Changes

Console Output

Edit Build Information

Delete build '#11'

Timings

Git Build Data

Pipeline Console

Replay

Pipeline Steps

✓ #11 (May 3, 2025, 7:53:30 AM)

Started by user avi

This run spent:

- 5 ms waiting;
- 39 sec build duration;
- 39 sec total from scheduled to completion.

Revision: e8c42ad5ec5997009269b95c50930f8123ffae3e

Repository: https://github.com/avi913/Flask_App.git

- refs/remotes/origin/master

No changes.

Docker Image Build Stage :-

Figure 3: Docker image being built from the Dockerfile using Jenkins.”

Flask_App / Dockerfile

avinash project files

Code Blame 15 lines (11 loc) · 264 Bytes Code 55% faster with GitHub Copilot

```

1  # Used official Python image
2  FROM python:3.9-slim
3
4  # Set working directory
5  WORKDIR /app
6  COPY requirements.txt .
7  RUN pip install -r requirements.txt
8
9  COPY . .
10
11 # Expose port
12 EXPOSE 5000
13
14 # Run the app with gunicorn
15 CMD ["gunicorn", "-b", "0.0.0.0:5000", "app:app"]

```

Cmd :[docker images]

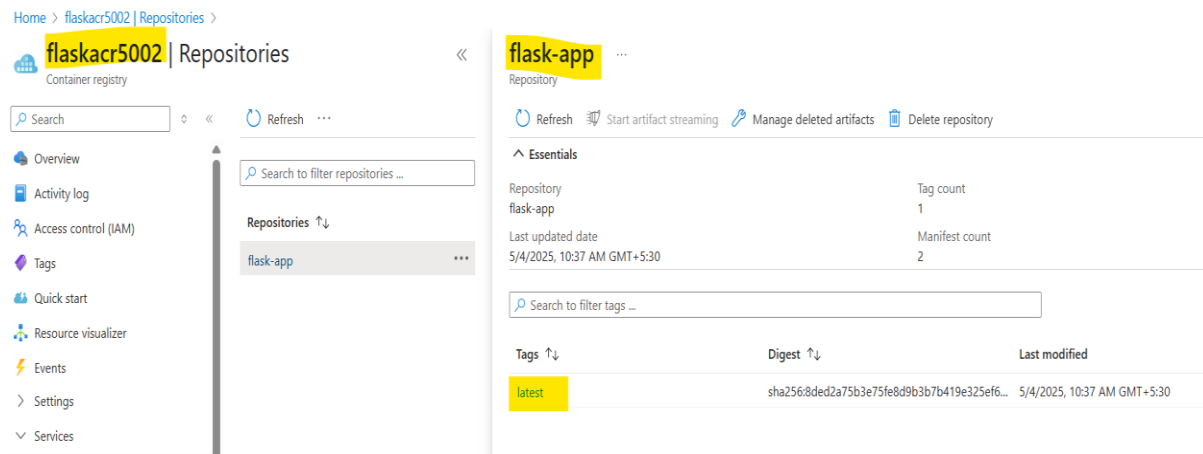
```

Last login: Sun May 4 04:47:52 2025 from 103.215.237.162
avi@master-vm:~$ docker images
REPOSITORY                                TAG          IMAGE ID       CREATED        SIZE
flaskacr5002.azurecr.io/flask-app        latest       9e0db6734110   51 minutes ago 13
sonarqube                                 latest       e3614eb70b40   3 weeks ago   1.
mcr.microsoft.com/azure-cli               latest       4119f014521e   5 weeks ago   74
hello-world                               latest       74cc54e27dc4   3 months ago  10

```

Image Pushed to ACR :-

Docker image successfully pushed to Azure Container Registry



AKS Deployment via Kubectl :-

Figure 4: Application deployed to AKS using Kubernetes manifests.

- Deployment.yaml and service.yaml files

Flask_App / k8s / deployment.yaml

```
avi913 Update deployment.yaml

Code Blame 19 lines (19 loc) · 326 Bytes

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: flask-app
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: flask
10   template:
11     metadata:
12       labels:
13         app: flask
14     spec:
15       containers:
16       - name: flask
17         image: avinashaka3/flaskapp:latest
18         ports:
19         - containerPort: 5000
```

Flask_App / k8s / service.yaml

```
avinash modified deployment and service

Code Blame 14 lines (14 loc) · 207 Bytes

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: flask-service
5    labels:
6      app: flask
7  spec:
8    type: LoadBalancer
9    selector:
10     app: flask
11   ports:
12   - protocol: TCP
13     port: 80
14     targetPort: 5000
```

When we hit that url [http:// 57.151.71.114:80](http://57.151.71.114:80) it will open our web-app

flask-aks | Services and ingresses

Kubernetes service

Search

+ Create Delete Refresh Show labels Give feedback

Overview
Activity log
Access control (IAM)
Tags
Monitor
Diagnose and solve problems
Microsoft Defender for Cloud (preview)
Cost analysis
Resource visualizer
Kubernetes resources

Services Ingresses

Filter by namespace
All namespaces Service name: All Add label filter

Name	Namespace	Status	Type	Cluster IP	External IP	Ports	Age
kubernetes	default	Ok	ClusterIP	10.0.0.1		443/TCP	2 days
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP	2 days
metrics-server	kube-system	Ok	ClusterIP	10.0.60.199		443/TCP	2 days
flask-service	default	Ok	LoadBalancer	10.0.125.1	57.151.71.114	80:31078/TCP	1 hour

Cmd:[kubectl get pods],[kubectl get svc],[kubectl get deployment]

```
avi@master-vm:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
flask-app-7bd4f79657-ln6k8         1/1     Running   0           73m
flask-app-7bd4f79657-lzfk          1/1     Running   0           73m
avi@master-vm:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
flask-service       LoadBalancer  10.0.125.1    57.151.71.114  80:31078/TCP     76m
kubernetes           ClusterIP     10.0.0.1      <none>         443/TCP          43h
avi@master-vm:~$ kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
flask-app  2/2     2            2           76m
avi@master-vm:~$
```

Webhook Setup in GitHub :-

Figure 5: GitHub webhook configured to trigger Jenkins pipeline on code push.

avi913 / Flask_App

Issues Pull requests Actions Projects Wiki Security Insights Settings

General Webhooks Add webhook

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

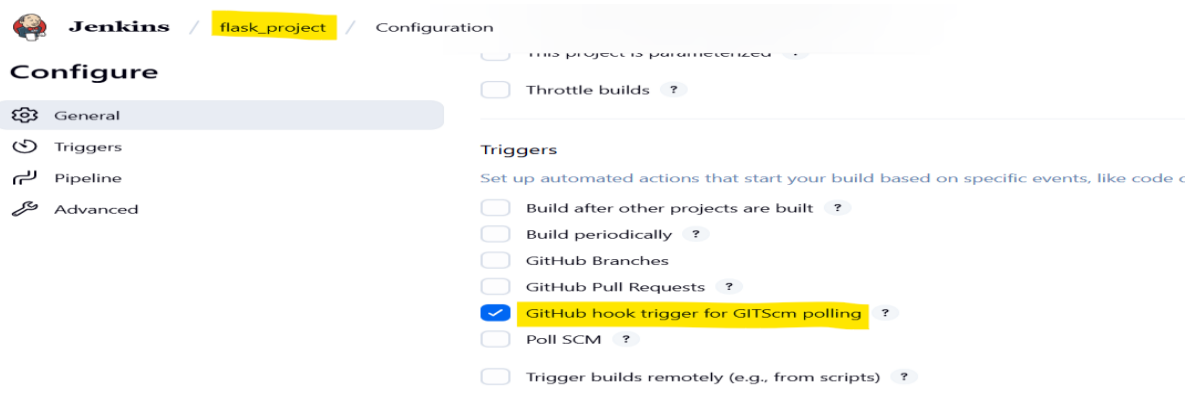
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ http://172.191.129.128:8080/github... (push)

Last delivery was successful.

Edit Delete

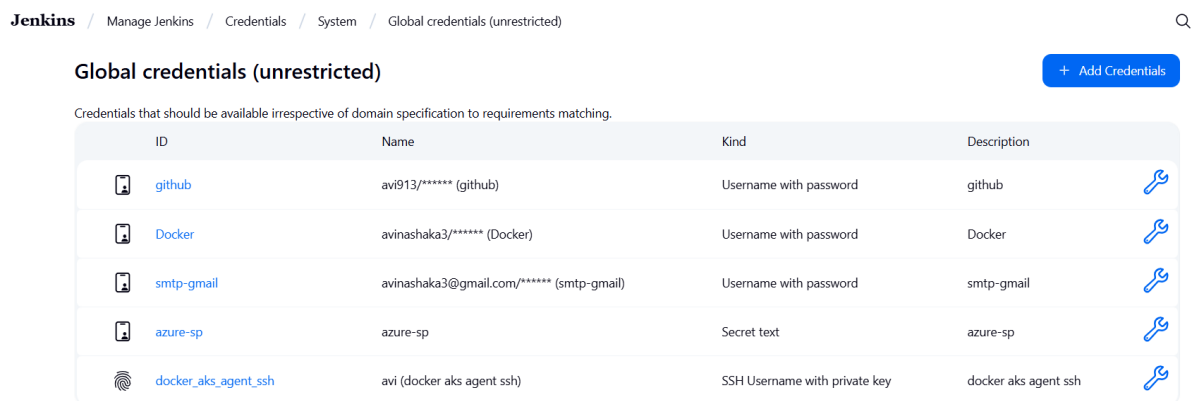
In Jenkins pipeline project we need to select the “GitHub hook trigger for GITScm polling”



Service Principal Credentials in Jenkins :-

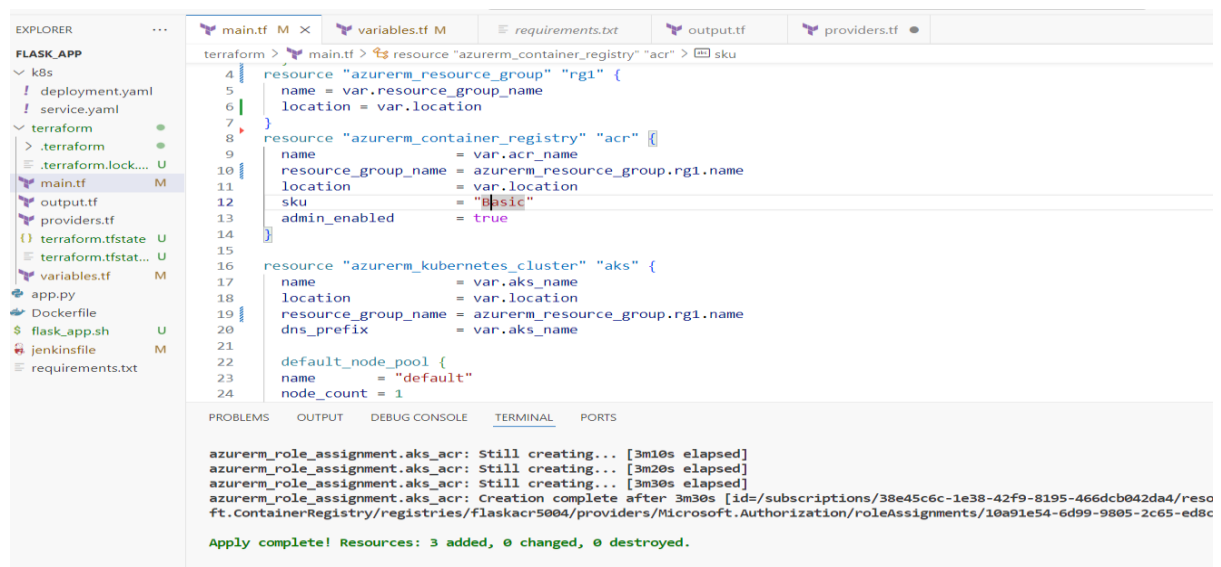
Figure:6 Jenkins credentials configuration for Azure Service Principal authentication.

Goto->manage Jenkins-> select credentials -> add new credentials



Terraform Apply Output :-

Figure 6: Terraform output showing successful provisioning of AKS and ACR.



Jenkins Plugins Used in the Project :-

- 1. Git Plugin**
Used to clone the source code from GitHub.
- 2. Pipeline Plugin**
Allows defining jobs in Jenkinsfile using Declarative or Scripted syntax.
- 3. Blue Ocean Plugin**
Provides a modern UI for visualizing pipeline stages and status.
- 4. Docker Pipeline Plugin**
Used to build and manage Docker containers from the pipeline.
- 5. Credentials Binding Plugin**
Manages and injects credentials like Azure Service Principal or API tokens securely.
- 6. Azure CLI Plugin**
Allows executing az CLI commands directly from Jenkins.
- 7. Kubernetes CLI Plugin (kubectl)**
Used to interact with AKS clusters via kubectl in pipeline steps.
- 8. GitHub Integration Plugin**
Enables webhook triggering and GitHub build status reporting.

Conclusion:

The project successfully demonstrates an end-to-end CI/CD pipeline for deploying a Flask-based web application on Azure Kubernetes Service (AKS) using Jenkins, Docker, Terraform, and Azure Container Registry (ACR). It includes:

- Automated infrastructure provisioning using Terraform (AKS and ACR).
- A Jenkins pipeline that checks out code, builds a Docker image, pushes it to ACR, and deploys it to AKS.
- Integration with GitHub via webhooks for automated deployment.
- Use of key DevOps plugins and Azure authentication via Service Principal