

1) Simple Python App (Flask)

Build a python application and push the source code to a GitHub repository.

2) Dockerize Python App

Create a `Dockerfile` under a `docker/` directory

Build the Docker image

Test the application locally using Docker

3) AWS EC2 Setup Terraform

Terraform will be used to provision AWS infrastructure in an automated and repeatable way. The Terraform setup should create and manage the following resources:

- VPC – A dedicated Virtual Private Cloud for the application environment
- Public Subnet – Subnet to host the EC2 instance with internet access
- Internet Gateway (IGW) – Enables outbound and inbound internet connectivity
- Route Table & Route Association – Routes public traffic (`0.0.0.0/0`) to the Internet Gateway
- Security Group – Controls inbound and outbound traffic
- Key Pair Association – For secure SSH access to the EC2 instance
- EC2 Instance – Ubuntu-based instance to run Dockerized Python application
- User Data Bootstrap – Installs Docker and required packages during instance launch
- Public IP Assignment – Enables public access to the application
- Terraform Outputs – Exposes important values such as:
 - EC2 Public IP
 - EC2 Public DNS
 - VPC ID
 - Security Group ID

4) AWS EC2 Setup

Launch an Ubuntu EC2 instance

Install Docker and required dependencies

5) Jenkins CI/CD (with GitHub)

Install Jenkins (on the same EC2 instance or a separate server)

Install required Jenkins plugins

6) Jenkins Pipeline (GitHub → Build → Push → Deploy)

every push triggers:

- build image
- push to registry (optional)
- deploy to EC2

7) Add Monitoring (Prometheus + Grafana) — simple Docker compose

Create a monitoring stack on the EC2 instance using Docker Compose

Configure Prometheus to collect server and container metrics

Create server and container monitoring dashboards in Grafana