### **CI/CD Deployment Assignment**

You are required to deploy a **Flask backend** and an **Express frontend** on an **Amazon EC2 instance**. Additionally, implement a **CI/CD pipeline** using Jenkins to automate the deployment process.

#### **Part 1: Deploy Flask and Express on a Single EC2 Instance**

1. **Objective**:
   * Deploy both the Flask backend and the Express frontend on a **single Amazon EC2 instance**.
2. **Steps**:
   * **Provisioning the EC2 Instance**:
     + Launch an EC2 instance on AWS (you can use a free-tier eligible instance).
     + SSH into the instance and install the following dependencies:
       - Python for Flask.
       - Node.js for Express.
       - Git for pulling the application code.
   * **Application Setup**:
     + Clone the Flask and Express repositories onto the EC2 instance.
     + Install the required dependencies for both applications using pip and npm.
     + Configure both applications to run on different ports (e.g., Flask on port 5000 and Express on port 3000).
     + Start the applications using process managers like pm2 or systemd to ensure they remain active.
3. **Deliverables**:
   * A running EC2 instance with Flask and Express accessible via the instance's public IP.
   * A description or diagram of the deployed architecture.

#### **Part 2: Implement CI/CD Pipeline Using Jenkins**

1. **Objective**:
   * Automate the deployment of Flask and Express applications using Jenkins.
2. **Steps**:
   * **Install Jenkins**:
     + Install Jenkins on the same EC2 instance or on a separate machine.
     + Configure Jenkins by installing essential plugins like Git, NodeJS, and Python.
   * **Set Up Jenkins Pipeline**:
     + Create two separate Jenkins pipelines for the Flask and Express applications.
     + **Pipeline Steps**:
       - Pull the latest code from the respective Git repositories.
       - Install dependencies for Flask (pip install -r requirements.txt) and Express (npm install).
       - Restart the applications using the process manager (e.g., pm2 restart <app>).
   * **Triggering the Pipeline**:
     + Set up a GitHub webhook to trigger the Jenkins pipeline on every push to the repositories.
   * **Optional Enhancements**:
     + Add testing stages to the pipeline for both applications.
     + Configure environment variables in Jenkins for managing sensitive data like API keys.
3. **Deliverables**:
   * A fully functional CI/CD pipeline that automates the deployment process for Flask and Express.
   * A Jenkins pipeline script (Jenkinsfile) for each application.
   * Evidence of the pipeline working (e.g., screenshots of successful builds and deployments).

EXPLANATION:-

### **Step 1: Launch EC2 Instance**

1. Login to your **AWS Console** → Navigate to **EC2**.
2. Click **Launch Instance** → Choose:  
   * AMI: **Ubuntu 22.04 LTS** (Free-tier eligible)
   * Instance type: **t2.micro**
3. Add a **Security Group**:  
   * Allow inbound traffic for ports:  
     + 22 (SSH)
     + 3000 (Express)
     + 5000 (Flask)
4. Launch and download the **.pem key** for SSH.

### **Step 2: Connect to EC2**

From your terminal:

chmod 400 your-key.pem

ssh -i "your-key.pem" ubuntu@<EC2-Public-IP>

### **Step 3: Install Dependencies**

sudo apt update -y

sudo apt install -y python3 python3-pip git nodejs npm

Check installations:

python3 --version

node -v

npm -v

### **Step 4: Clone Repositories**

(Use your own GitHub repos)

cd /home/ubuntu

git clone https://github.com/yourusername/flask-app.git

git clone https://github.com/yourusername/express-app.git

### **Step 5: Setup Flask Backend**

cd flask-app

pip3 install -r requirements.txt

Run Flask:

python3 app.py

(Default: runs on port 5000)

### **Step 6: Setup Express Frontend**

cd ../express-app

npm install

Run Express:

node server.js

(Default: runs on port 3000)

### **Step 7: Keep Apps Running (using pm2)**

sudo npm install -g pm2

pm2 start /home/ubuntu/flask-app/app.py --interpreter=python3 --name flask

pm2 start /home/ubuntu/express-app/server.js --name express

pm2 startup

pm2 save

**Now visit:**

* Flask → http://<EC2-Public-IP>:5000
* Express → http://<EC2-Public-IP>:3000

### **Architecture Description**

User

│

▼

AWS EC2 Instance

├── Flask (Python) → Port 5000

└── Express (Node.js) → Port 3000

**Part 2: Implement CI/CD Pipeline Using Jenkins Objective**

Automate deployment of both Flask & Express apps using **Jenkins**.

### **Step 1: Install Jenkins**

On the same EC2 instance:

sudo apt update

sudo apt install -y openjdk-17-jre

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install -y jenkins

sudo systemctl start jenkins

sudo systemctl enable jenkins

### **Step 2: Access Jenkins**

Open in browser:

http://<EC2-Public-IP>:8080

Get admin password:

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

Install **suggested plugins** → Create your admin user.

### **Step 3: Install Required Plugins**

In Jenkins Dashboard → **Manage Jenkins → Plugins → Available Plugins** Install:

* **Git Plugin**
* **NodeJS Plugin**
* **Python Plugin**
* **Pipeline Plugin**

Then configure Node.js and Python tools under  
 **Manage Jenkins → Global Tool Configuration**.

### **Step 4: Create Pipelines**

#### **A. Flask Pipeline**

1. Create a new **Pipeline job** → Name it flask-pipeline.
2. Under “Pipeline script”, use:

pipeline {

agent any

stages {

stage('Checkout Code') {

steps {

git 'https://github.com/yourusername/flask-app.git'

}

}

stage('Install Dependencies') {

steps {

sh 'pip3 install -r requirements.txt'

}

}

stage('Restart Flask App') {

steps {

sh 'pm2 restart flask || pm2 start app.py --interpreter=python3 --name flask'

}

}

}

}

#### **B. Express Pipeline**

1. Create another **Pipeline job** → Name it express-pipeline.
2. Add script:

pipeline {

agent any

stages {

stage('Checkout Code') {

steps {

git 'https://github.com/yourusername/express-app.git'

}

}

stage('Install Dependencies') {

steps {

sh 'npm install'

}

}

stage('Restart Express App') {

steps {

sh 'pm2 restart express || pm2 start server.js --name express'

}

}

}

}

### **Step 5: Configure GitHub Webhooks**

In your GitHub repositories:

* Go to **Settings → Webhooks → Add Webhook**
* Payload URL: http://<Jenkins-Public-IP>:8080/github-webhook/
* Content type: application/json
* Trigger: **Just the push event**