**Jenkins Multibranch pipeline project using Jenkins Agent**

**For the code you can refer this repository ( https://github.com/rohit808077/two-tier-flask-app )**

**Steps Overview**

**Step 1: EC2 Instances Setup**

**\*\*Create an AWS Account:\*\***

- Go to the AWS website and sign in or create an account.

**\*\*Launch EC2 Instance:\*\***

- Go to the EC2 dashboard in the AWS Management Console.

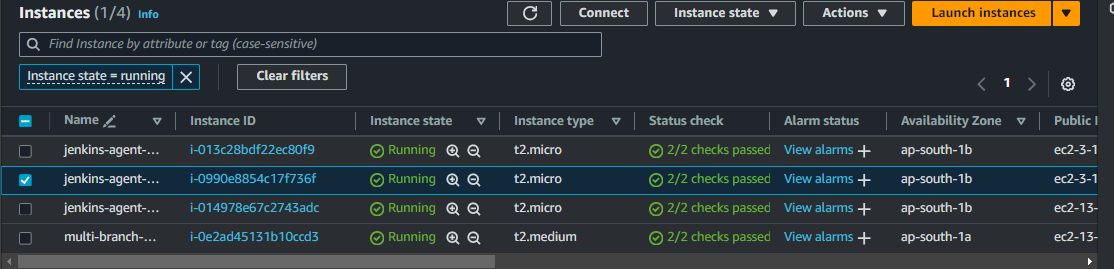
- Click "Launch Instance" and choose an Amazon Machine Image (AMI) (e.g., Amazon Linux, Ubuntu).

- Select an instance type, configure instance details (like network settings, security groups, SSH key pairs), and launch the instance.

**- \*\*Connect to EC2 Instance:\*\***

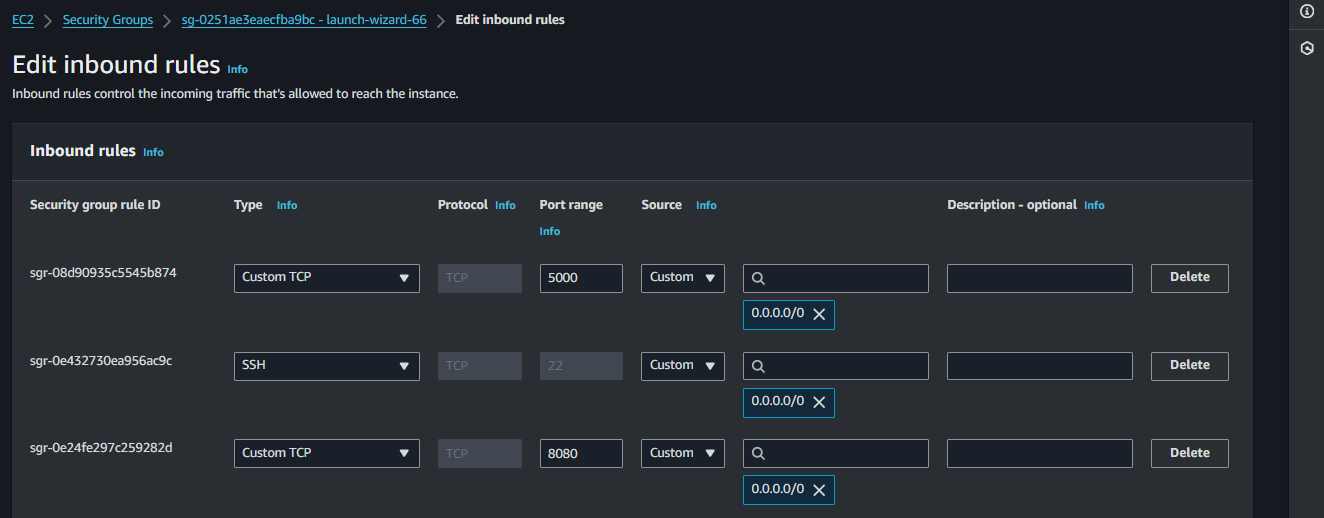
- Use SSH to connect to your EC2 instance. For example: `ssh -i your-key.pem ec2-user@your-instance-ip`

**Jenkins Master Instance:**



1)Install Jenkins, Java, Docker, and Docker Compose.

2)Grant Docker access to the ubuntu user.

3)Open port 8080 for Jenkins access. 

**Set Up Jenkins:**

- \*\*Install Jenkins:\*\*

- Follow the official Jenkins installation guide for your operating system: [Jenkins Installation Guide]( <https://www.jenkins.io/doc/book/installing/linux/> ).

**4)Configure Jenkins admin user and initial setup.**

Access Jenkins in a web browser (`http://your-instance-ip:8080`) and follow the setup wizard to unlock Jenkins and install recommended plugins.

**To access the password to unlock Jenkins use following command**

sudo cat /var/lib/Jenkins/secrets/initialAdminPassword



**Set-up docker - Install Docker: follow Below commands**

#sudo apt-get install docker.io -y

#sudo apt-get install docker-compose -y

give access to user for docker

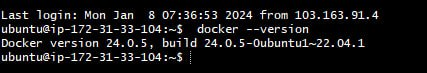
#sudo usermod -aG docker ubuntu

reboot your system

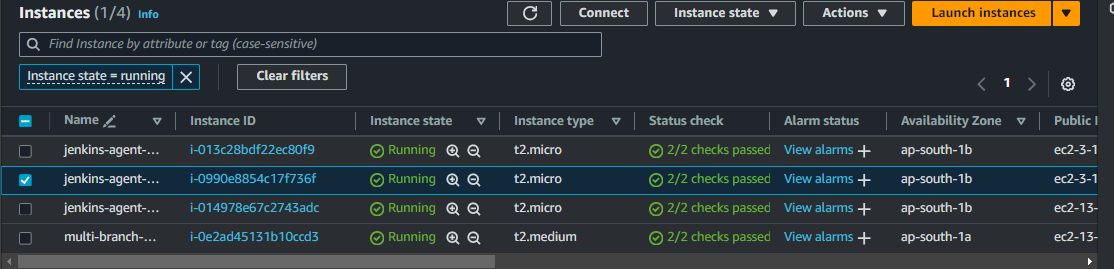
#sudo rebbot

To verify that you have docker permission

#docker ps



**Jenkins Agent Instances: dev-server, stg-server, and prd-server**



1)Create dev-server, stg-server, and prd-server EC2 instances.

2)Install Java, Docker, and Docker Compose on each instance.

3)Grant Docker access to the ubuntu user.

**\*\*Install Java\*\***

- Follow the official Jenkins installation guide for your operating system: [Java Installation Guide] ( <https://www.jenkins.io/doc/book/installing/linux/> ).

**Set-up docker on prd -server-agent and stg-sever**

- Install Docker: follow Below commands

docker installation command

#sudo apt-get install docker.io -y

#sudo apt-get install docker-compose -y

give access to user for docker

#sudo usermod -aG docker ubuntu

reboot your system

#sudo rebbot

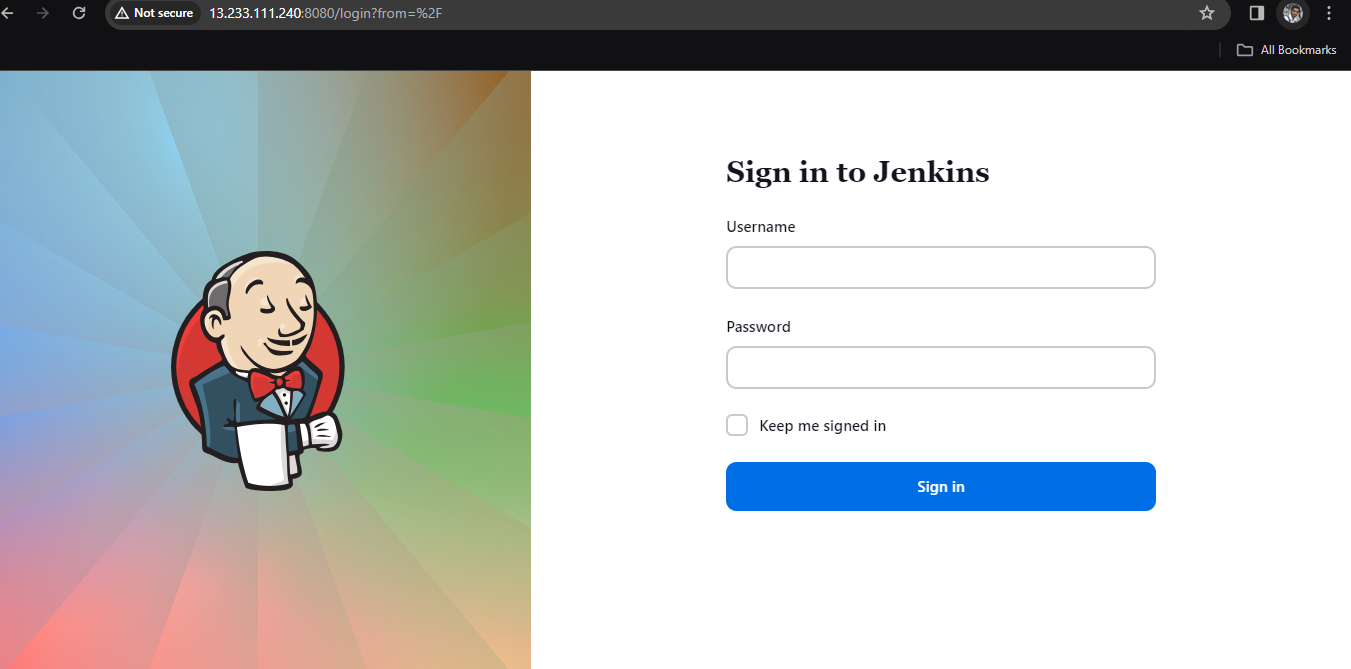
To verify that you have docker permission

#docker ps

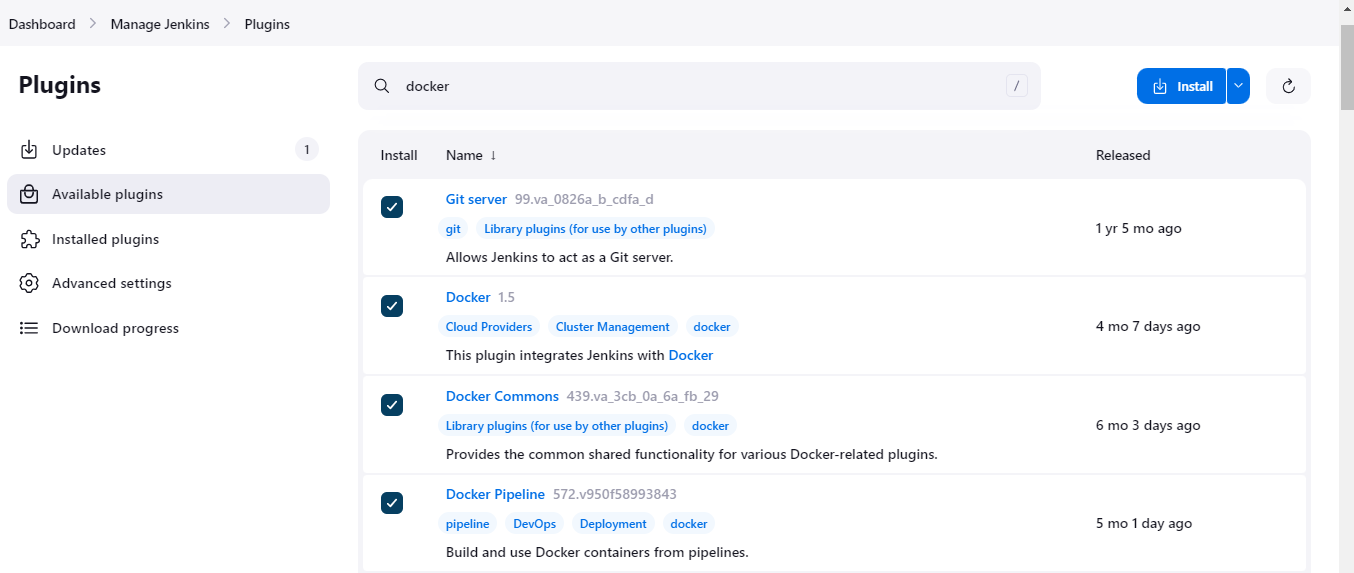
**Step 2: Jenkins Multibranch Pipeline**

Access Jenkins Master:

Use public IP and port 8080 to access Jenkins.

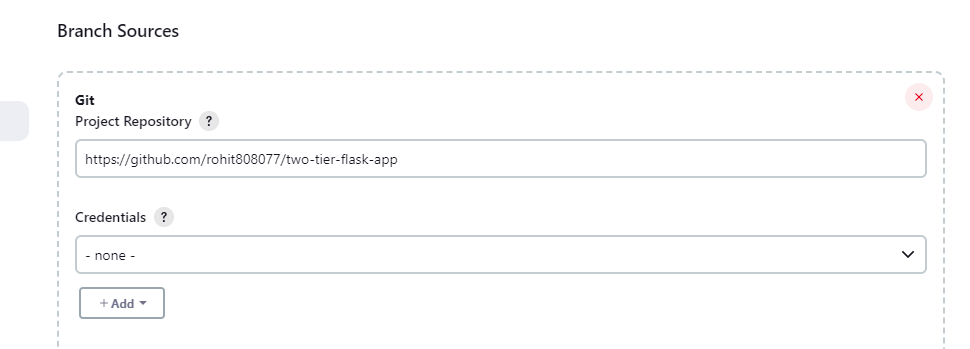


**Install necessary plugins: Git, Docker, Docker Pipeline.**

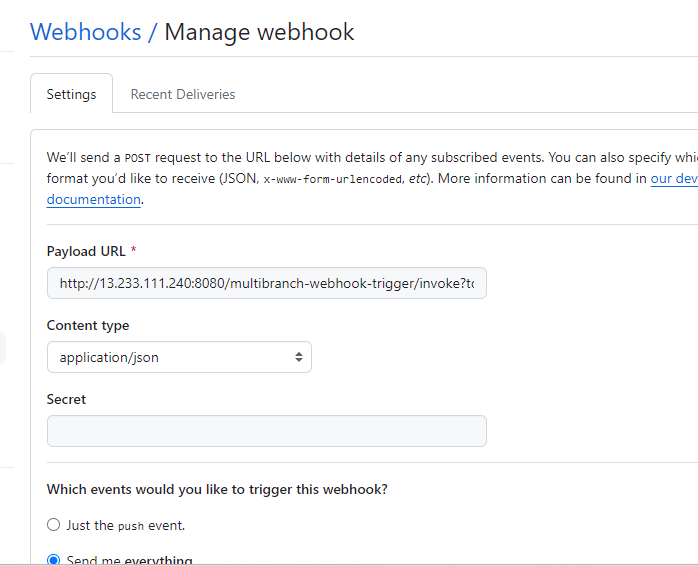


**Create Multibranch Pipeline:**

Create a Multibranch Pipeline linked to the GitHub repository.



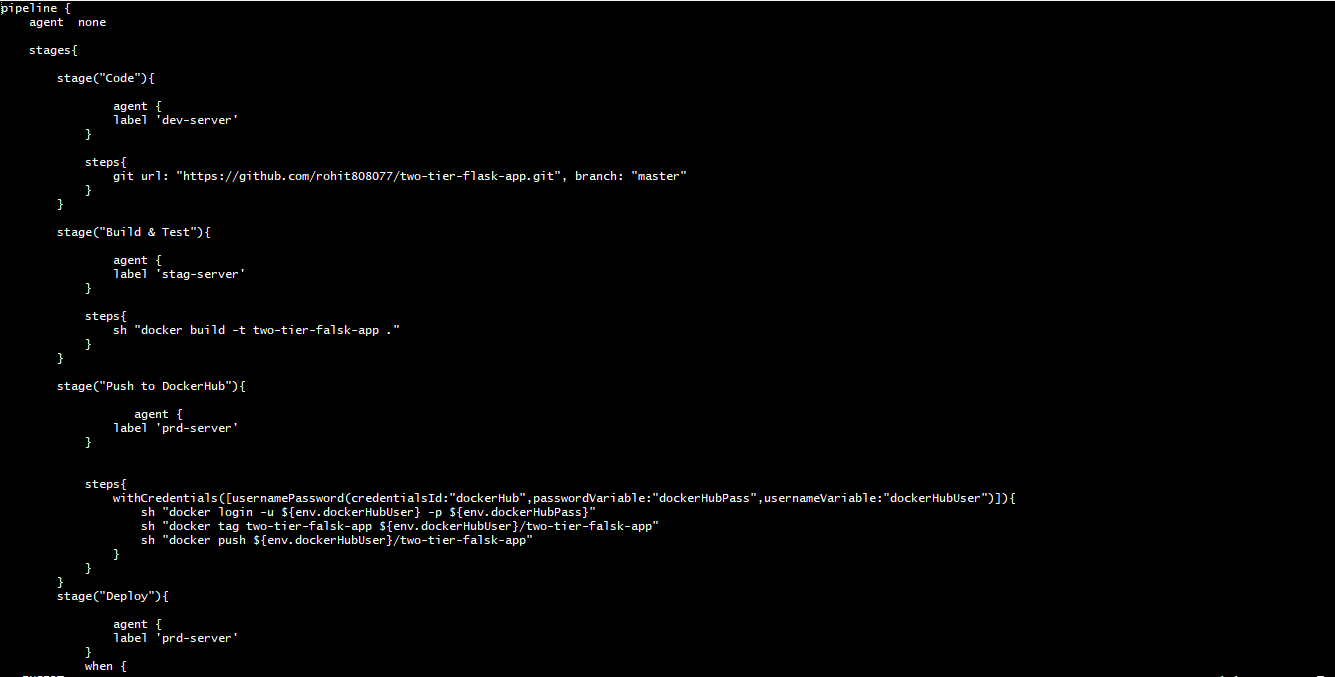
**Configure a webhook for automatic triggering.**

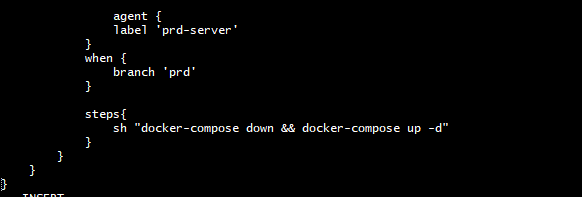


**Jenkins file:**

Create a Jenkins file with stages for code clone, build and test, Docker Hub, and deploy.

Define agent labels for dev-server, stage-server, and prd-server.





**Step 3: Set Up Jenkins Agents**

Generate SSH Keys:

Generate an SSH key pair on the Jenkins master.

Goto .ssh file **Cd .ssh**

Create key pair run command **ssh-keygen**

Copy the public key.

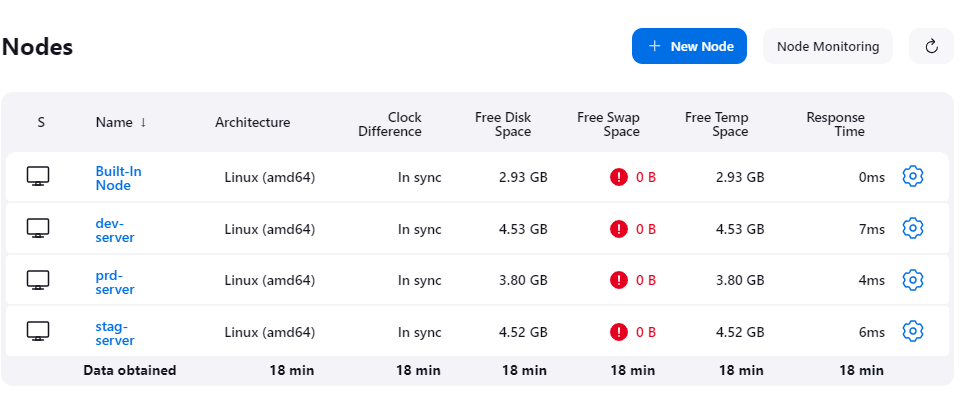
**Jenkins Agent Configuration:**

Install Jenkins agent Java on each EC2 instance.

Configure agents to connect to the Jenkins master using the SSH key.

Connect Agents to Jenkins Master:

Add new nodes for dev-server, stg-server, and prd-server in Jenkins.

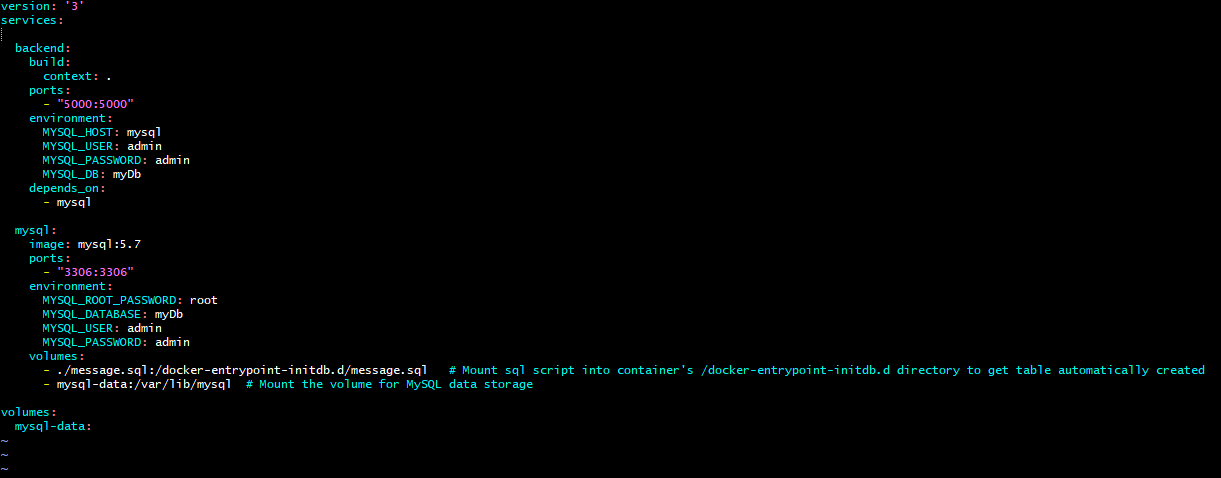


Configure nodes to use the SSH key for authentication.

**Step 4: Docker Compose Deployment**

Docker Compose File:

Create a docker-compose.yml file specifying containers, networks, and ports.

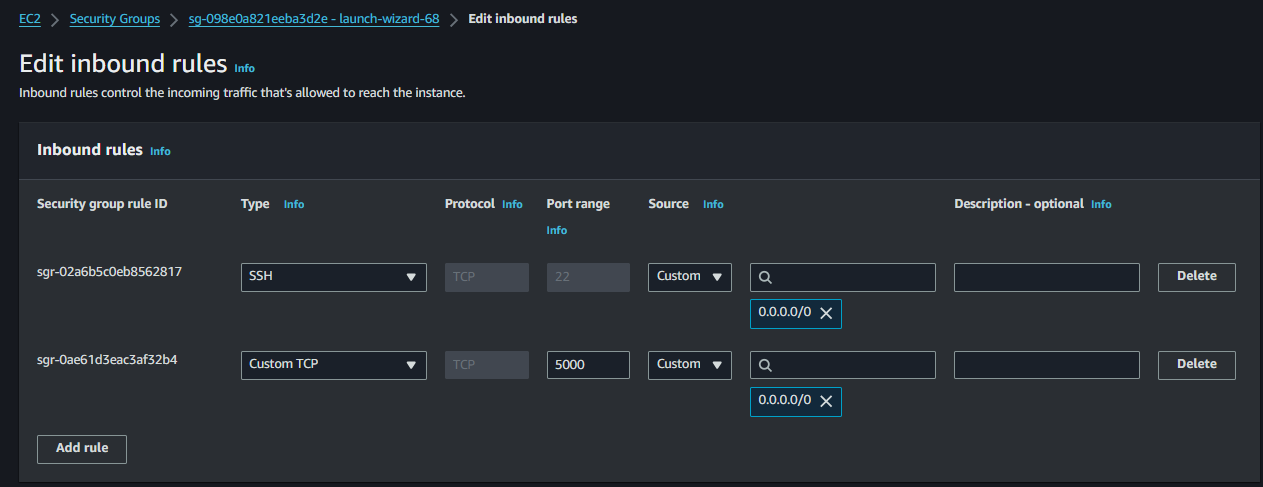


Integration with Jenkins file:

Modify Jenkins file to include a stage for deploying using Docker Compose.

**Step 5: Application Access**

Expose Port on prd-server:



Ensure Docker Compose file exposes the required port (e.g., 5000) on prd-server.

**Access Application:**

Access deployed application on prd-server using public IP and specified port.

