**Final Project Report: CI/CD Pipeline Setup using Ansible, Docker, and Jenkins**  
**By: D Praveen**

**1. Introduction**

This project focuses on automating software deployment using Ansible, Docker, and Jenkins on AWS EC2 instances. It covers setting up a DevOps pipeline to deploy a web application.

DevOps is a combination of development and operations that focuses on continuous integration and continuous deployment (CI/CD). The main goal is to shorten the software development lifecycle and increase efficiency by automating repetitive tasks.

**2. Tools Used in the Project**

**Ansible**

Ansible is an open-source automation tool that enables configuration management, application deployment, and task automation. It is agentless, meaning it does not require any software to be installed on the target machines. Ansible uses simple YAML-based playbooks to define configurations and automate tasks.

**Docker**

Docker is a platform that allows developers to package applications and their dependencies into containers. These containers can be run consistently across different environments, eliminating the "works on my machine" problem. Docker ensures faster application deployment and better resource utilization.

**Jenkins**

Jenkins is an open-source automation server that enables continuous integration and continuous deployment (CI/CD). It helps automate the process of building, testing, and deploying applications. Jenkins integrates with various tools, including Ansible and Docker, to create robust DevOps pipelines.

**Git**

Git is a distributed version control system that allows multiple developers to collaborate on projects efficiently. It enables tracking changes in code, maintaining different versions, and integrating with CI/CD pipelines to automate software deployment. GitHub, GitLab, and Bitbucket are popular platforms that host Git repositories and support collaboration in DevOps environments.

**3. Project Setup**

**Step 1: Launching EC2 Instances**

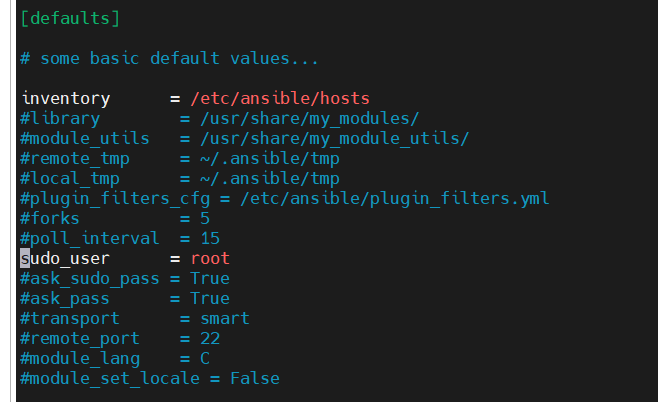
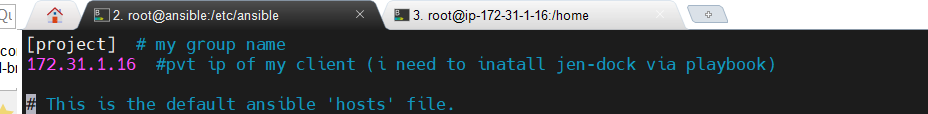
* Created two AWS EC2 instances:
  + One for **Ansible** (Master)
  + One for **Docker & Jenkins** (Node)

**4. Installing Ansible**

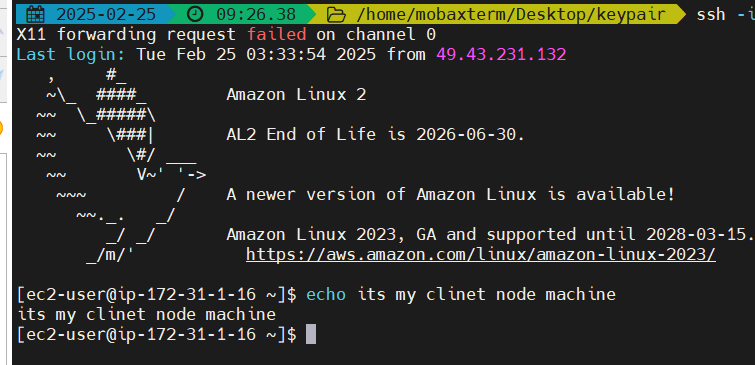
**Bash Script to Install Ansible**

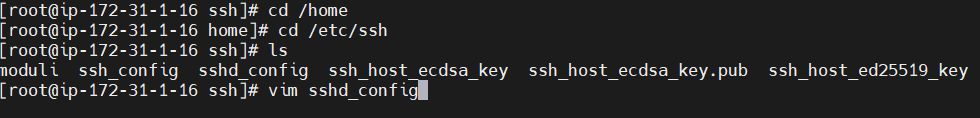
**Filename:** installansible.sh

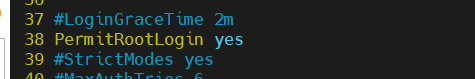
|  |
| --- |
| sudo amazon-linux-extras install epel  sudo yum update -y  sudo yum install ansible -y  ansible --version |

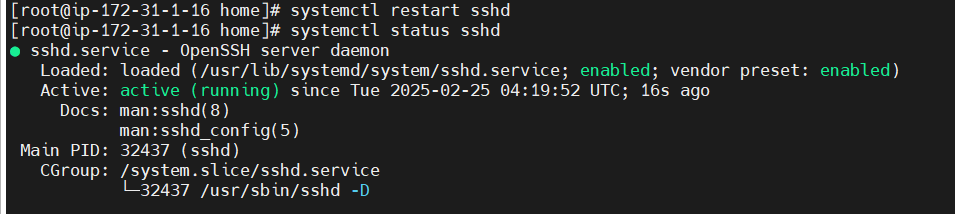
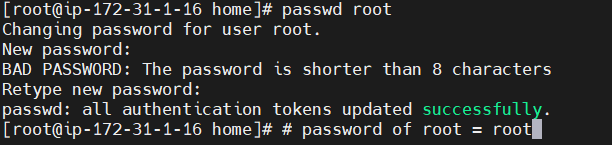
* Moved to Ansible configuration directory: cd /etc/ansible
* [root@ansible ansible]# ls
* ansible.cfg hosts roles
* [root@ansible ansible]# vi ansible.cfg
* [root@ansible ansible]# pwd
* /etc/ansible
* [root@ansible ansible]# ls
* ansible.cfg hosts roles
* [root@ansible ansible]# vi hosts

**5. SSH Configuration & Key Exchange**

* Enabled **Password Authentication** on the Node server (/etc/ssh/sshd\_config).
* Am using the MobaXteram



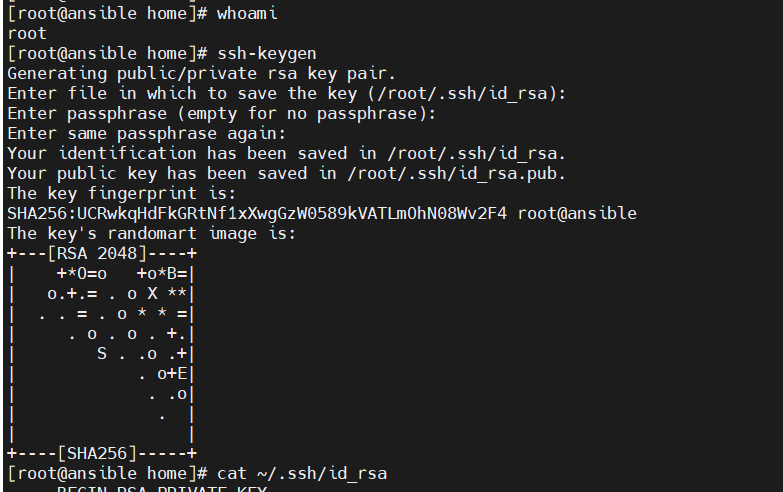
* do the following steps
* NOW WE HAVE TO SAY YES TO PASSWORD AUTHNETICATION

* change the password authentication from no to yes
*  STEP- \* RESTART SSHD (systemctl restart sshd )
* SET A PASSWORD TO USER IN ROOT SERVER (passwd root)
* Restarted SSH service:

systemctl restart sshd

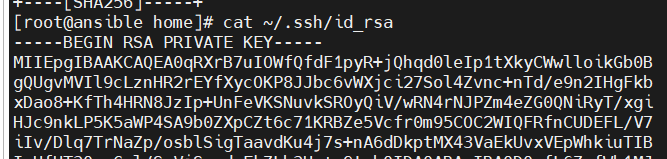
* Set root user password:
* passwd root

# Connect the MASTER To NODE server SSH

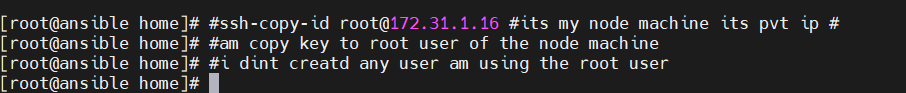
* Generated an SSH key on the Master server: Generate the SSH key == #
* am generating the key as root user

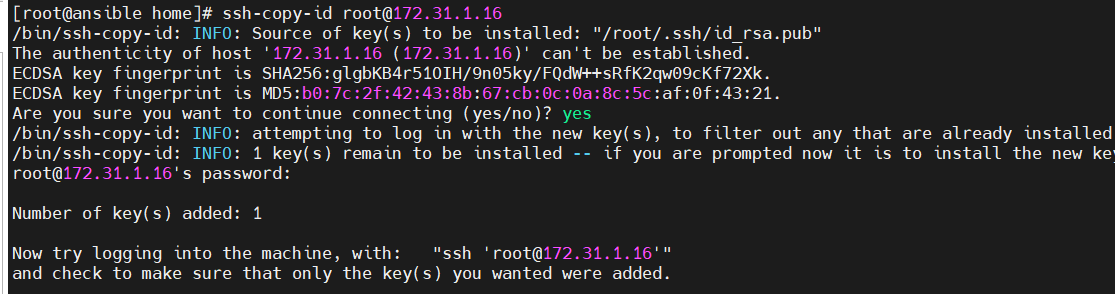
Check the at the location of the ssh key is generated are not to verify use cat

cat ~/.ssh/id\_rsa



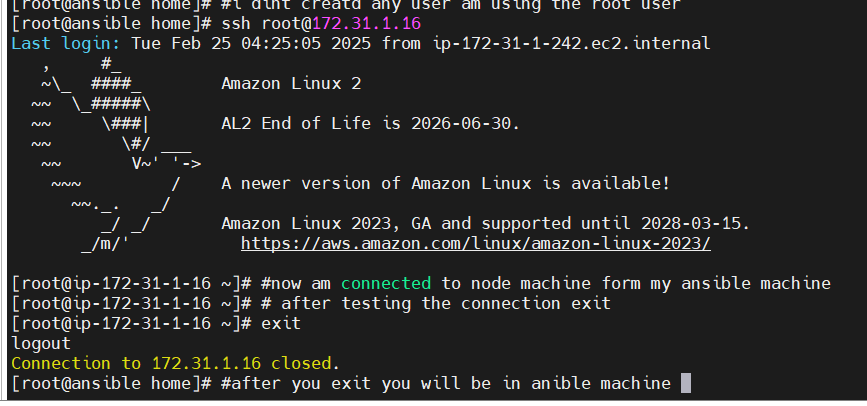
* ssh-keygen
* Copied the SSH key to the Node server:
* ssh-copy-id root@<NODE\_PRIVATE\_IP>
* Verified the connection:
* ssh root@<NODE\_PRIVATE\_IP>

ssh-copy-id root@IP address (pvt IP) of Node server---execute in Master server

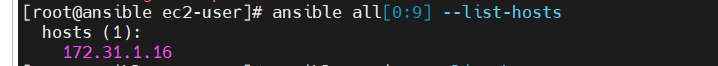
copy the SSHkey === [root@ansible home]# ssh-copy-id root@172.31.1.16

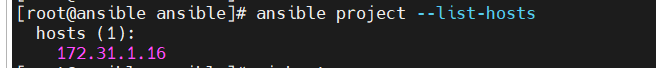
Test Connection: Verify the connection from the control node to the managed node:

ssh user@remote\_host

Now try logging into the machine, with: "ssh 'root@172.31.1.16'"

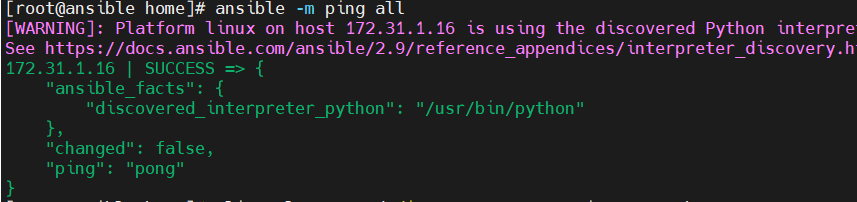
To the HOSTS in Environment

ansible all --list-hosts ---🡪all servers

ansible project --list-hosts -----🡪 all the servers of project environment #[project] is my group name in hosts file

ansible all[0:9] --list-hosts ---🡪top 10 servers

now test the ping between the host

ansible -m ping all

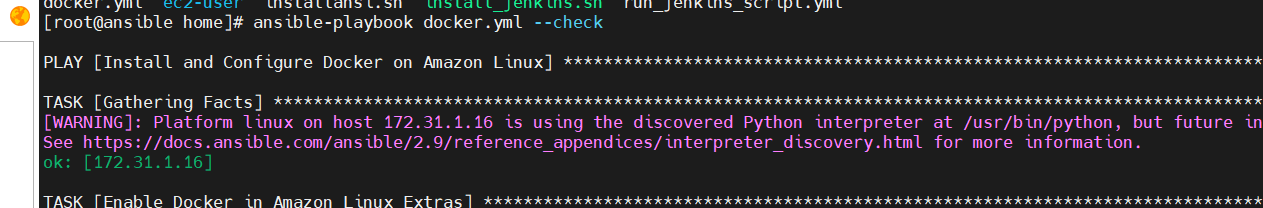
**6. Configuring Ansible Hosts**

* Added the Node server to the Ansible inventory file (/etc/ansible/hosts).
* Tested connectivity:
* ansible -m ping all

**7. Installing Docker using Ansible Playbook**

**Filename:** docker.yml

|  |
| --- |
| ---  - name: Install and Configure Docker on Amazon Linux  hosts: all  become: true  become\_user: root  gather\_facts: true  tasks:  - name: Enable Docker in Amazon Linux Extras  command: amazon-linux-extras enable docker  - name: Install Docker  yum:  name: docker  state: present  - name: Update all packages  yum:  name: "\*"  state: latest  - name: Start Docker service  systemd:  name: docker  state: started  enabled: yes |



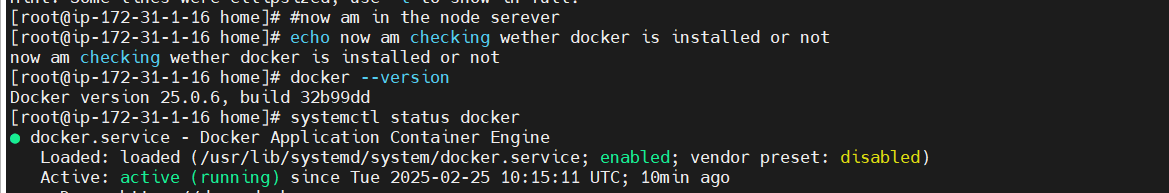
=====================================================================

Now am checking the node server installed of docker

=====================================================================

The output I check in node machine I checked the version and the status of the docker is installed

* Verified installation:
* docker --version
* systemctl status docker

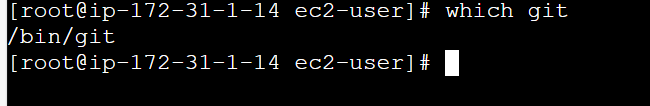


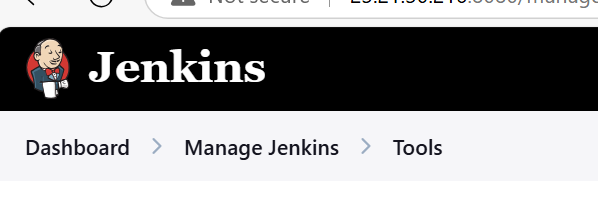
**8. Installing Git using Ansible Playbook**

**Filename:** git-install.yml

|  |
| --- |
| --- #install the git into nodes  - name: Install Git    hosts: all    become: yes    tasks:     - name: Install Git       yum:         name: git         state: latest |

* Verified Git installation:
* which git
* And past in the jenkinks



go to



**9.0 installing Jenkins using playbook**

|  |
| --- |
| **---**  **- name: Install and configure Jenkins on Amazon Linux 2**  **hosts: all**  **become: yes**  **tasks:**  **- name: Update system packages**  **command: yum update -y**  **- name: Add Jenkins repository**  **command: wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo**  **- name: Import Jenkins repository key**  **command: rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key**  **- name: Upgrade system packages**  **command: yum upgrade -y**  **- name: Install Java 17 (Amazon Corretto)**  **command: yum install java-17-amazon-corretto -y**  **- name: Install Jenkins**  **command: yum install jenkins -y**  **- name: Enable Jenkins service**  **command: systemctl enable jenkins**  **- name: Start Jenkins service**  **command: systemctl start jenkins** |

**OR**

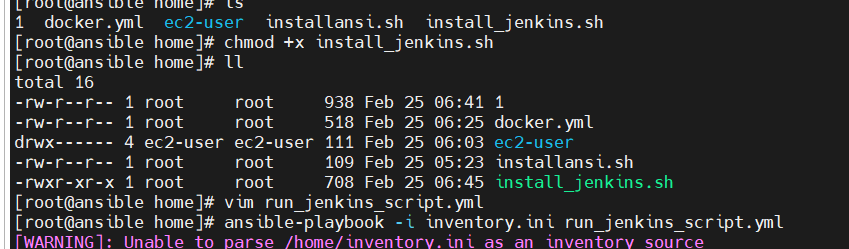
**9.1. Installing Jenkins using Bash Script**

**Filename:** install\_jenkins.sh

|  |
| --- |
| #!/bin/bash  set -e # Exit on any error  # Update system packages  sudo yum update -y  # Add Jenkins repository  sudo wget -O /etc/yum.repos.d/jenkins.repo \  https://pkg.jenkins.io/redhat-stable/jenkins.repo  sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key  # Upgrade system packages  sudo yum upgrade -y  # Install Java 17 (Amazon Corretto)  sudo yum install -y java-17-amazon-corretto  # Install Jenkins  sudo yum install -y jenkins  # Enable and start Jenkins service  sudo systemctl enable jenkins  sudo systemctl start jenkins  # Print Jenkins status  sudo systemctl status jenkins |

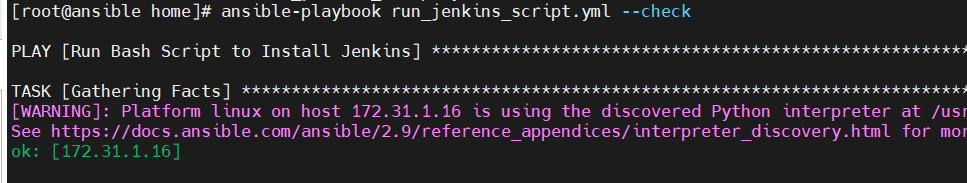
* Made the script executable:
* Ensure the script is **executable**:

|  |
| --- |
| chmod +x install\_jenkins.sh |

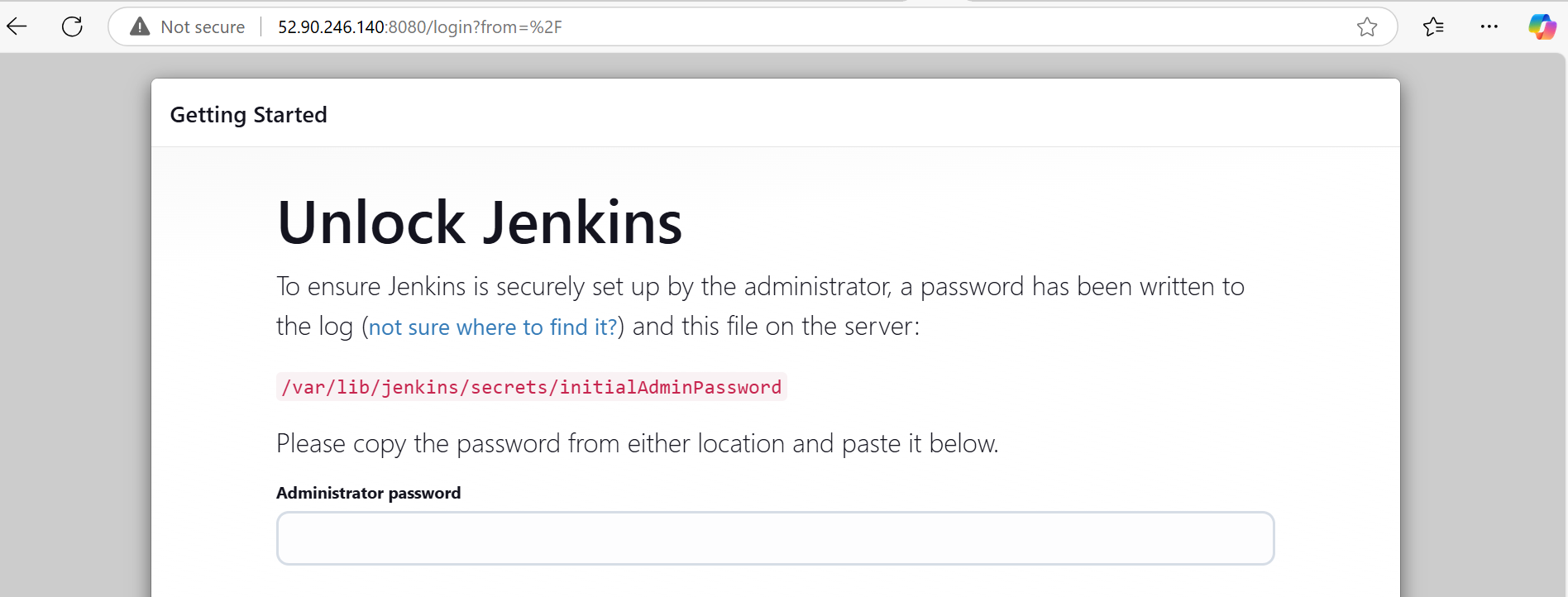


* Copied and executed it via Ansible playbook (run\_jenkins\_script.yml).
* **run\_jenkins\_script.yml == = =**🡺**my file name**

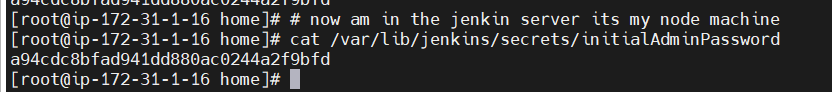
|  |
| --- |
| **---**  **- name: Run Bash Script to Install Jenkins**  **hosts: all**  **become: true**  **tasks:**  **- name: Copy Bash Script to Remote Host**  **copy:**  **src: install\_jenkins.sh #give file location of the file**  **dest: /tmp/install\_jenkins.sh**  **mode: '0755'**  **- name: Execute Bash Script on Remote Host**  **command: bash /tmp/install\_jenkins.sh** |

**The playbook is Successfully Executed**

**10. Configuring Jenkins**

* Accessed Jenkins via:
* http://<NODE\_PUBLIC\_IP>:8080
* Retrieved admin password:
* cat /var/lib/jenkins/secrets/initialAdminPassword
* No go to the node server where the Jenkins is installed and get the AdminPassword

cat /var/lib/jenkins/secrets/initialAdminPassword



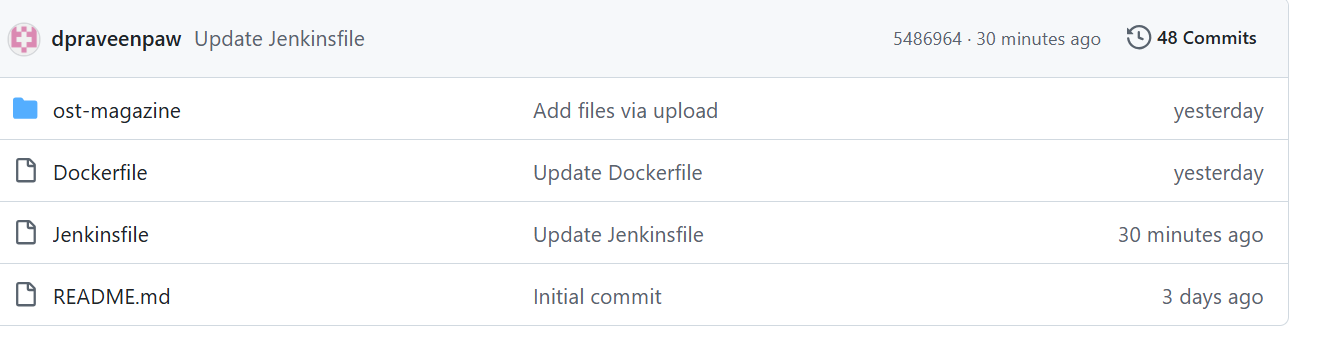
now create admin passwd admin

**11.** Configured **Pipeline Script from SCM** and added GitHub repository.

Now am in git repo create one repo and add the templates & Dockerfile , Jenkinsfile

* I has my Jenkinsfile and the dockerfile in the git repo

<https://www.free-css.com/assets/files/free-css-templates/download/page247/journey.zip>

* Extract it and upload to git repo 

**=====================================================================**

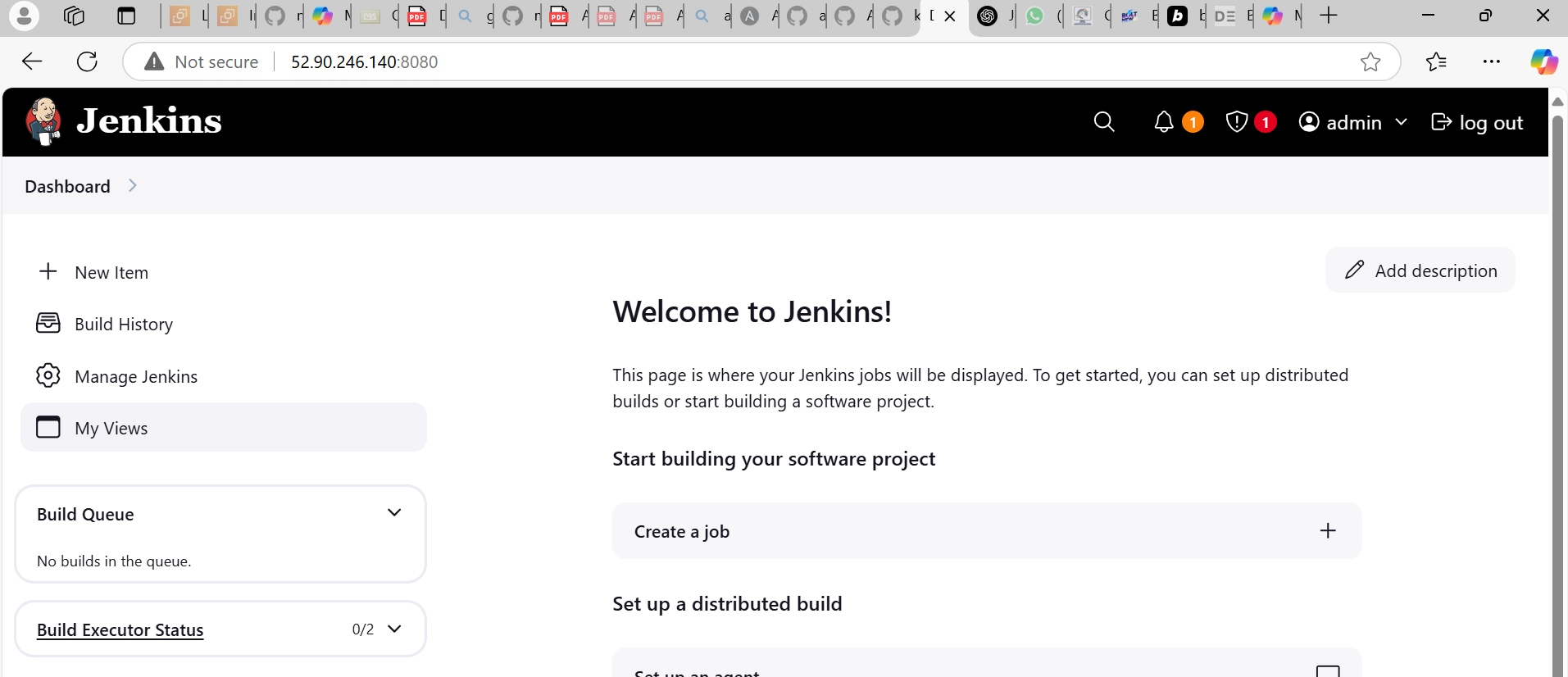
**Jenkinsfile ----**🡪 **make sure the file name in repo**

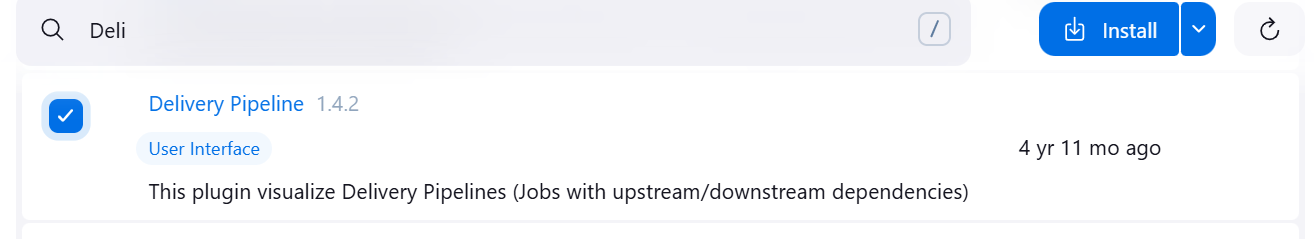
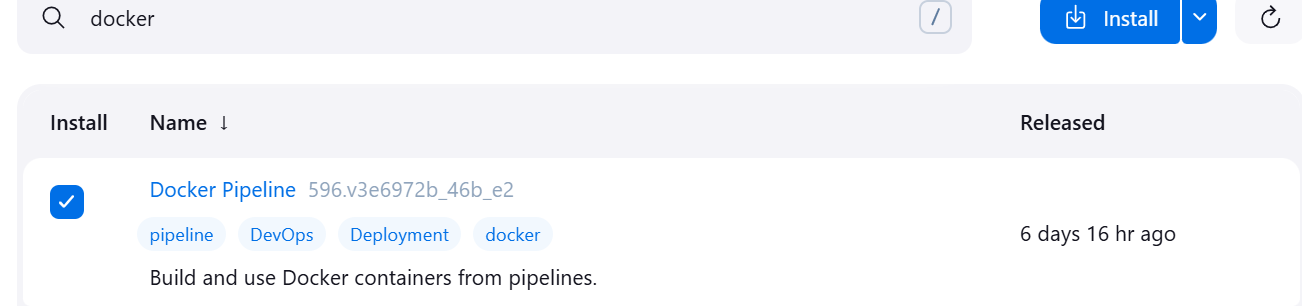
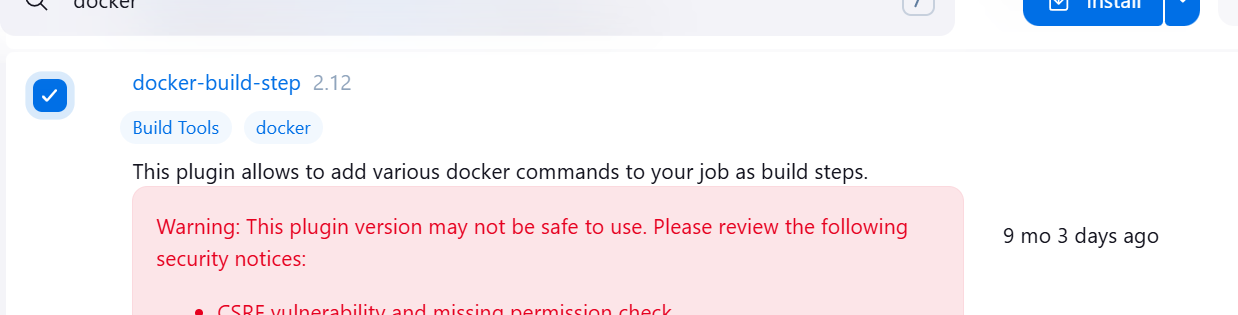
|  |
| --- |
| **pipeline {**  **agent any**    **stages {**  **stage('Checkout') {**  **steps {**  **// Ur branch URL generate from pipeline Syntax**  **git branch: 'main', url: 'https://github.com/dpraveenpaw/major-project.git'**  **checkout scm**  **}**  **}**    **stage('Build Docker Image') {**  **steps {**  **script {**  **// Build the Docker image**  **sh 'docker build -t my-nginx-app .'**  **}**  **}**  **}**    **stage('Run Container') {**  **steps {**  **script {**  **// Stop and remove existing container if it exists**  **sh 'docker rm -f my-nginx-container || true'**    **// Run the new container**  **sh 'docker run -d -p 80:80 --name my-nginx-container my-nginx-app'**  **}**  **}**  **}**    **stage('Verify') {**  **steps {**  **// Simple verification that container is running**  **sh 'docker ps | grep my-nginx-container'**  **}**  **}**  **}**  **}** |

* **Dockerfile ==🡺the file name**

|  |
| --- |
| **FROM nginx:alpine**  **# Remove default nginx static assets**  **RUN rm -rf /usr/share/nginx/html/\***  **# Copy static files from a remote URL**  **COPY ost-magazine . /usr/share/nginx/html/ #the source code is avil from git repo its**  **#scm also in my git repo it’s the path .**  **EXPOSE 80**  **CMD ["nginx", "-g", "daemon off;"]** |

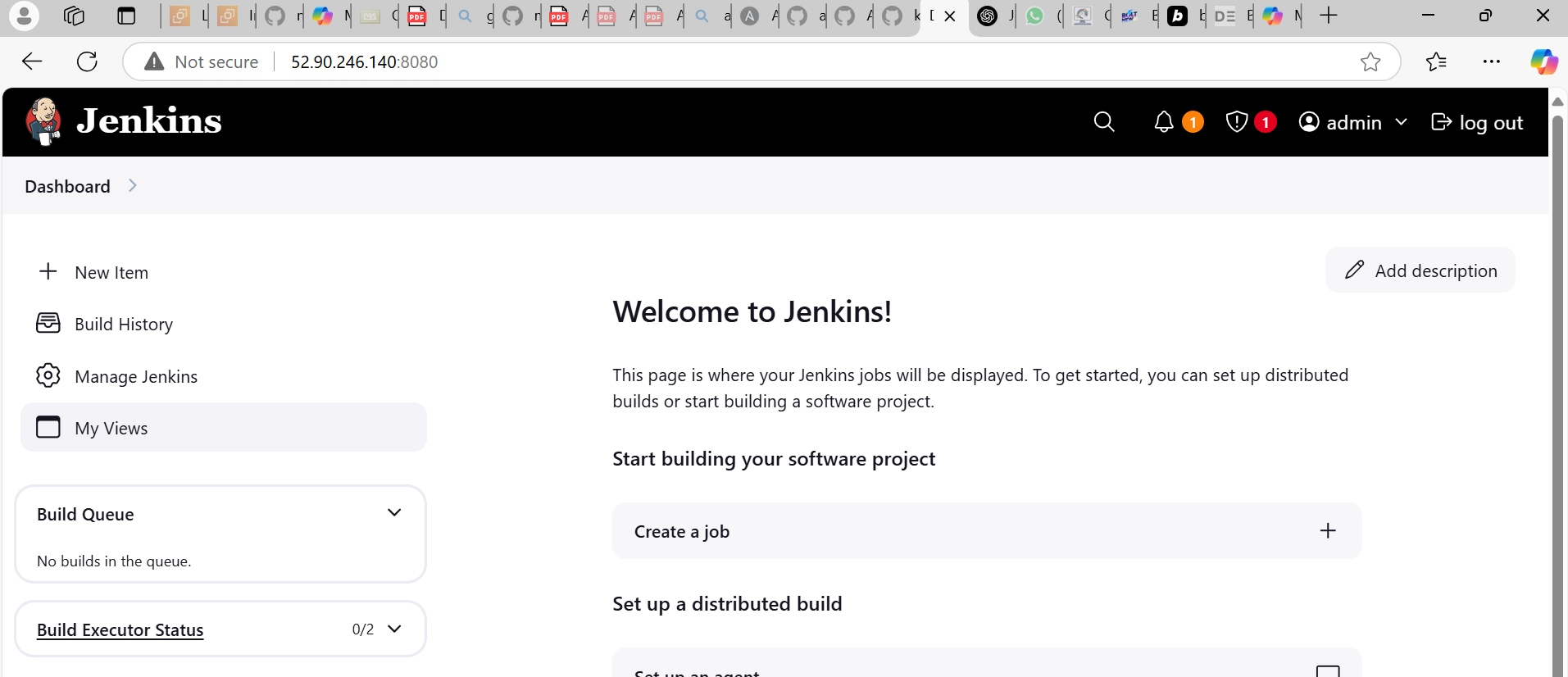
**12: Install the some Jenkins Plugins:**

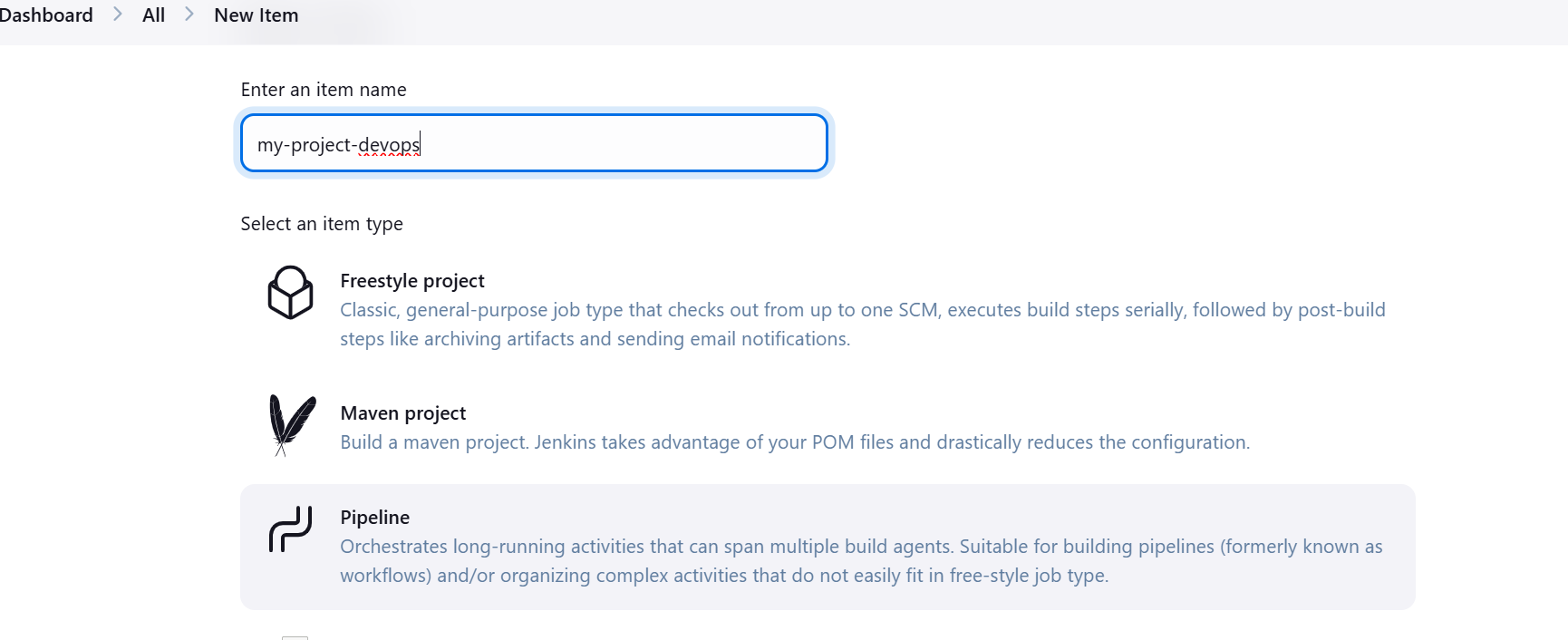


* Installed required Jenkins plugins:
  + **Delivery Pipeline**
  + **Docker Pipeline**
  + **docker-build-step**

**13**: **Build Pipeline Job**

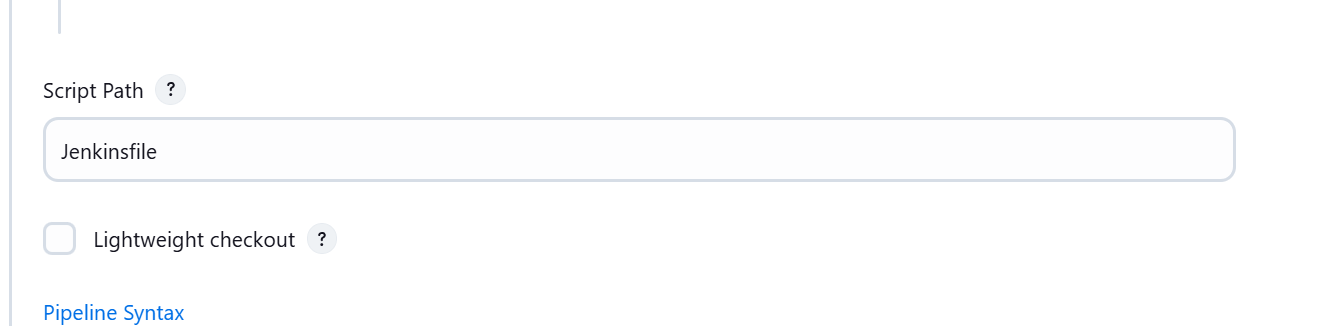
* Configured **Pipeline Script from SCM** and added GitHub repository.

===========================================================================



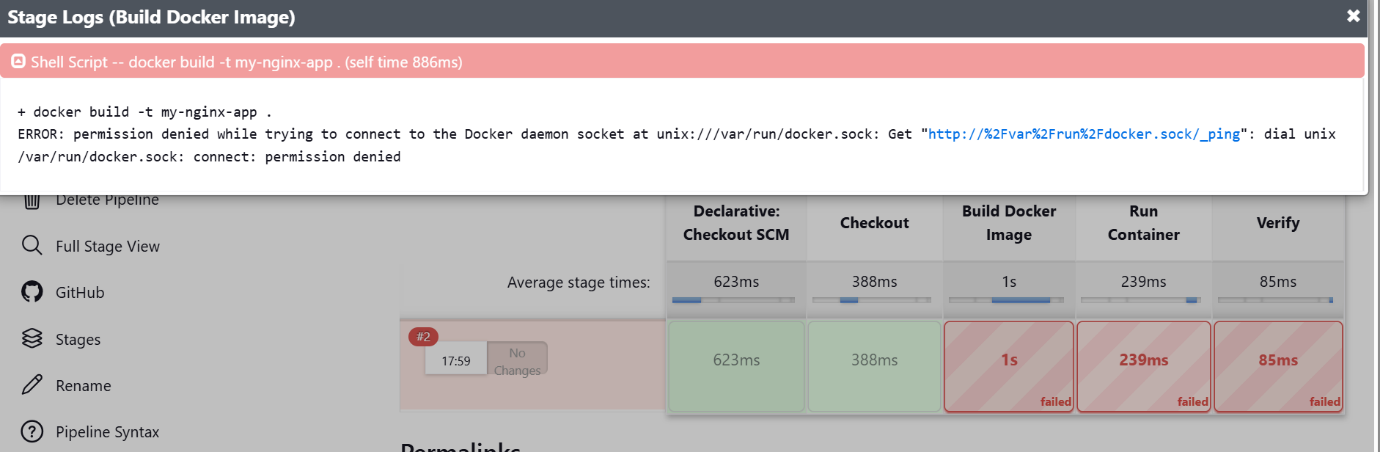
* copy the project url from the git hub
* I written the scrip and I kept in the git repo
* So am selecting the pipeline script from SCM

Script Path I selected Jenkinsfile I has the file in the my git repo



* I written the scrip and I kept in the git repo
* So am selecting the pipeline script from SCM
* check the branch of ur SCM Branch url

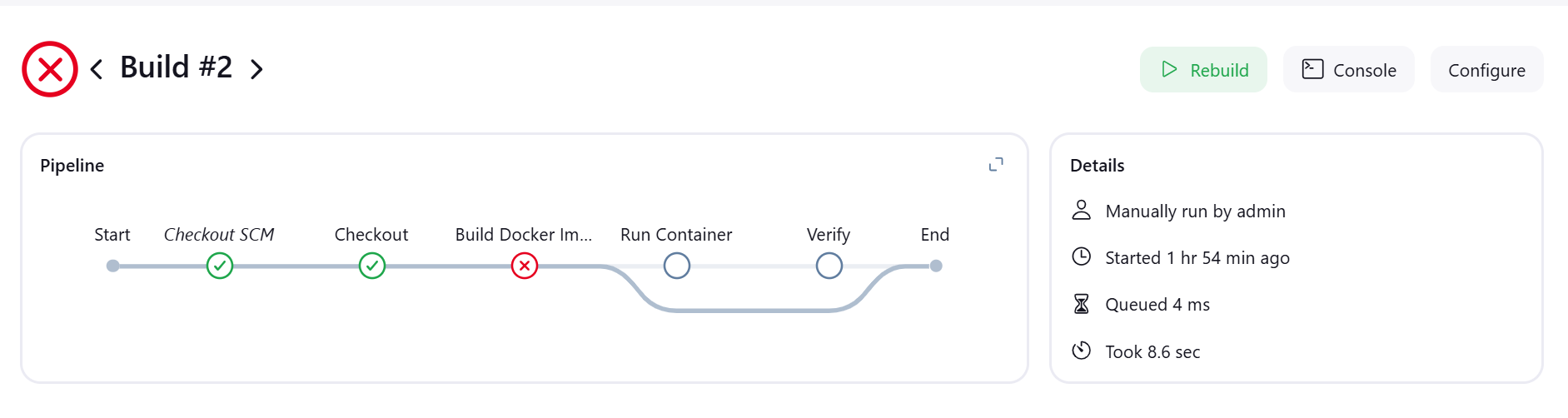
**14. Final Testing**

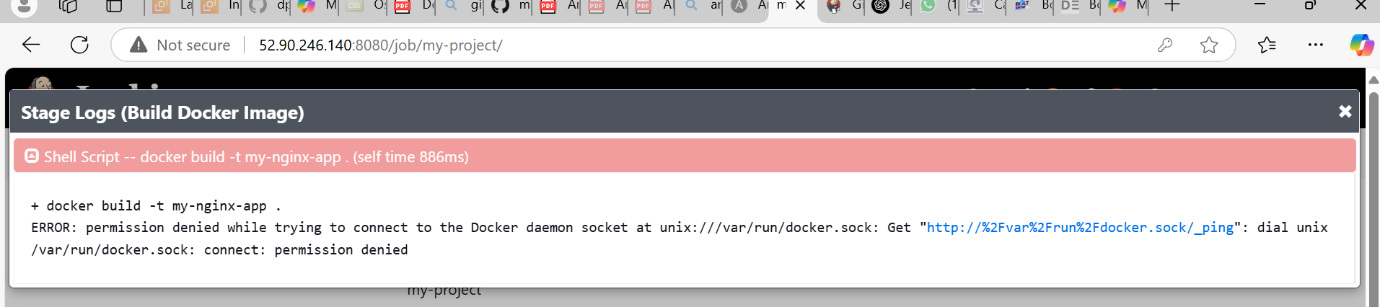
**N**ow build the job

If you get the Build Failure in this steps

Do the some changes in node server where you installed Jenkins and docker

I installed the docker and Jenkins in one instance



****

1. Add the Jenkins user to the Docker group on your Jenkins server. Run these commands on your Jenkins server as root or with sudo:

|  |
| --- |
| sudo usermod -aG docker jenkins |

1. Verify the permissions on the Docker socket:

|  |
| --- |
| sudo chmod 666 /var/run/docker.sock |

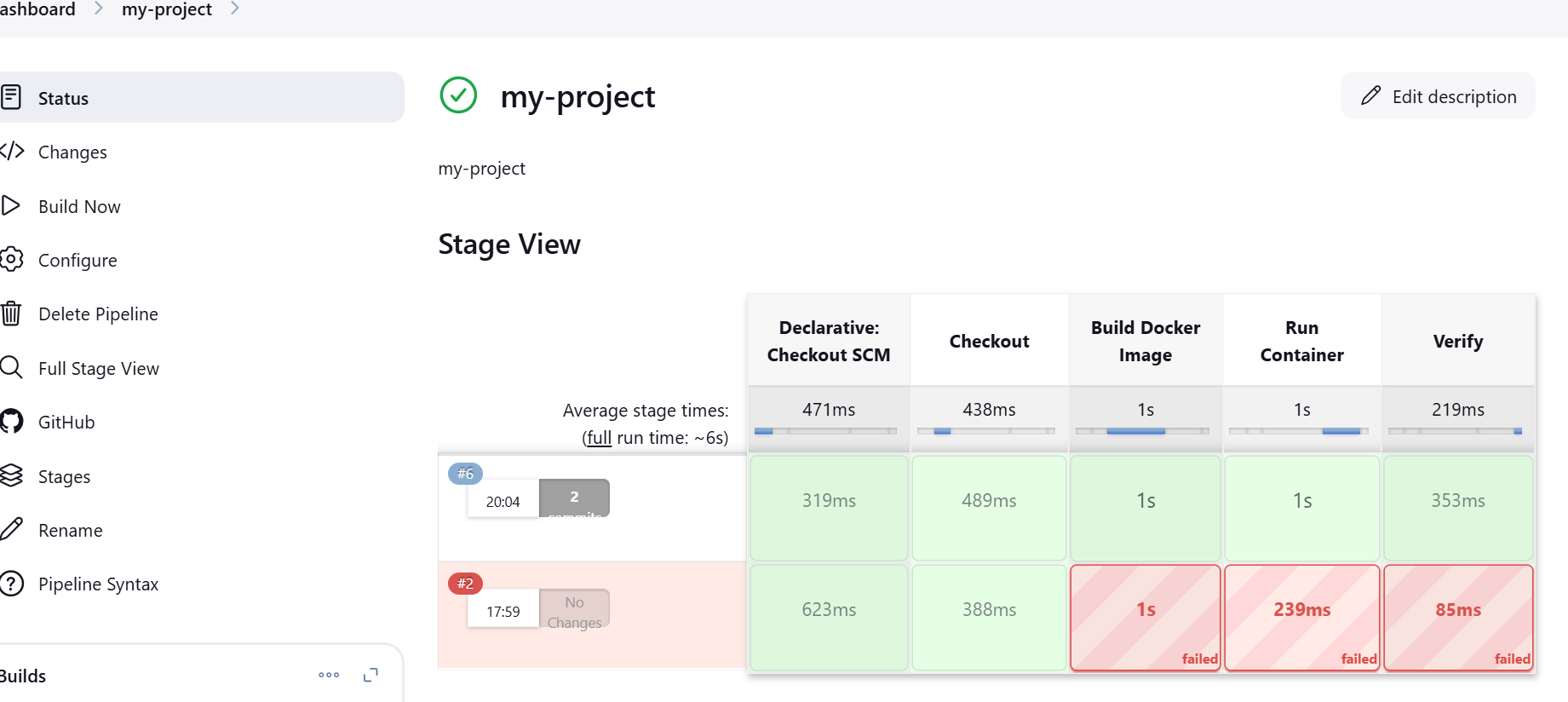
1. After making these changes, you'll need to restart both Jenkins and Docker services:

|  |
| --- |
| sudo systemctl restart docker  sudo systemctl restart jenkins |

IMPORTANT SECURITY NOTE: Adding Jenkins to the Docker group effectively gives Jenkins

===========================================================================

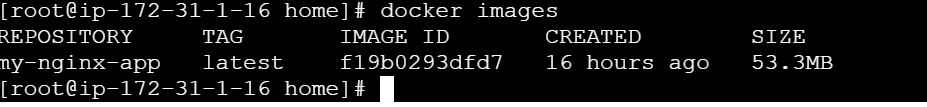
After doing this changes in node server now rebuild the jobs in Jenkins



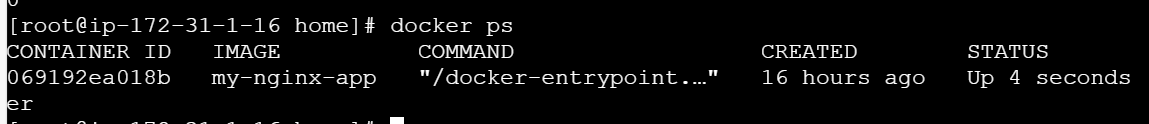
Now after the build success

After job build the docker image and container is automatically run in to port you assigned hear my port number is 80

Am verifying that image and container is available are not

go to the node check the docker images is available

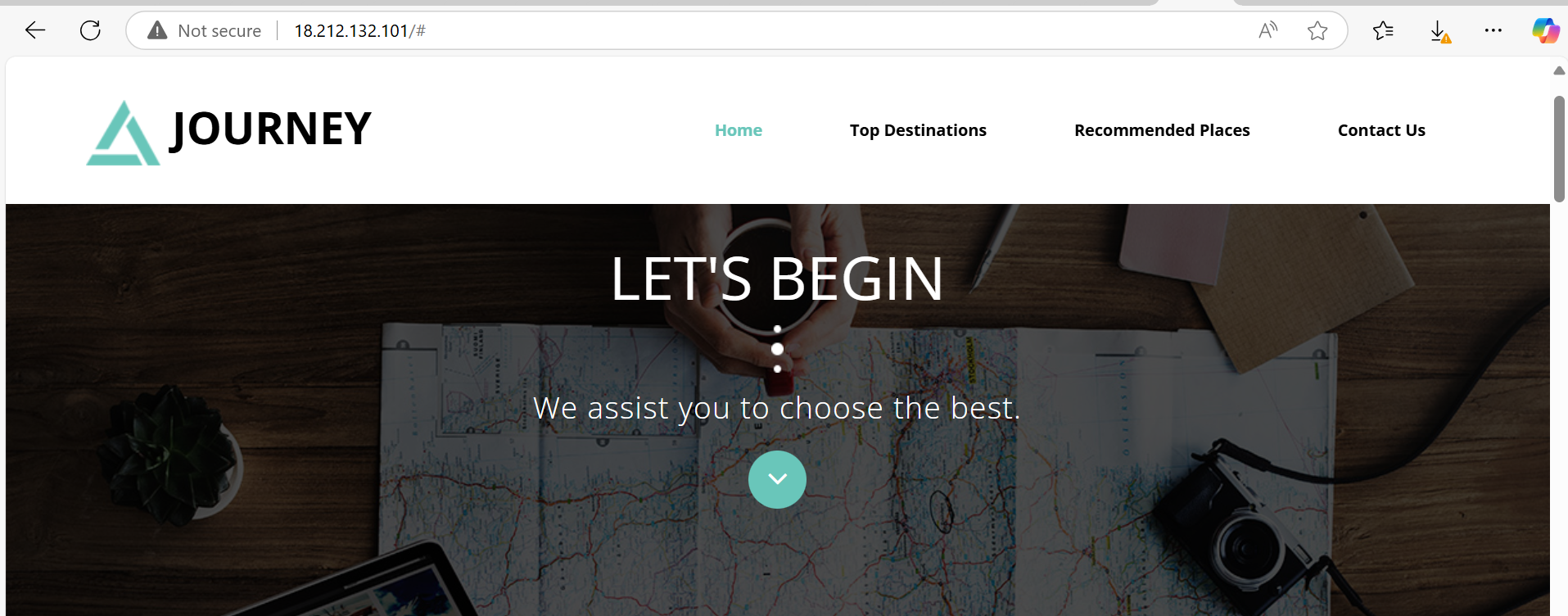
now check the docker container is created and running



Copy the public ip of the node server and search in the browser

52.90.246.140:80

The output



* Triggered a **Jenkins Build**.
* Checked the logs for success/failure.
* Accessed the deployed website at:
* http://<NODE\_PUBLIC\_IP>:80
* Verified that the web page was loading successfully.

**15. Conclusion**

In this project, I successfully automated the deployment of a web application using Ansible, Docker, and Jenkins on AWS. The pipeline ensured efficient CI/CD, reducing manual intervention. By leveraging Infrastructure as Code (IaC) principles, I streamlined the setup and deployment process. This project improved my practical understanding of DevOps methodologies and AWS infrastructure management.

Reference

<https://github.com/dpraveenpaw/major-project/tree/main>