**DevOps Lifecycle Implementation for Abode Software**

**Project Overview**

The goal of this project is to implement a DevOps lifecycle for **Abode Software**. The objective is to automate the build, test, and deployment processes for the product by utilizing a combination of configuration management, Git workflows, containerization, continuous integration (CI), and continuous delivery (CD). This setup will include tools like Ansible, Jenkins, Docker, and GitHub.

**GitHub Repository:** <https://github.com/hshar/website.git>

**1. Install Necessary Software with Ansible**

**Objective:**

Install the required software (Docker, Docker Compose, Jenkins) on all machines using a configuration management tool, Ansible.

**Tool:**

* **Ansible**

**Steps:**

**1.1. Create Ansible Playbook:**

Create a file named **setup.yml** to install Docker, Docker Compose, and Jenkins on all target machines.

- hosts: all

  become: yes

  tasks:

    - name: Install Docker

      apt:

        name: docker.io

        state: present

    - name: Install Docker Compose

      get\_url:

        url: https://github.com/docker/compose/releases/download/1.29.2/docker-compose-{{ ansible\_system }}-x86\_64

        dest: /usr/local/bin/docker-compose

        mode: 'u+x'

    - name: Download Jenkins key

      get\_url:

        url: https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

        dest: /usr/share/keyrings/jenkins-keyring.asc

    - name: Add Jenkins repository to sources list

      lineinfile:

        path: /etc/apt/sources.list.d/jenkins.list

        line: "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/"

        create: yes

    - name: Update package index

      apt:

        update\_cache: yes

    - name: Install Jenkins

      apt:

        name: jenkins

        state: present

    - name: Ensure Jenkins is started

      service:

        name: jenkins

        state: started

        enabled: yes

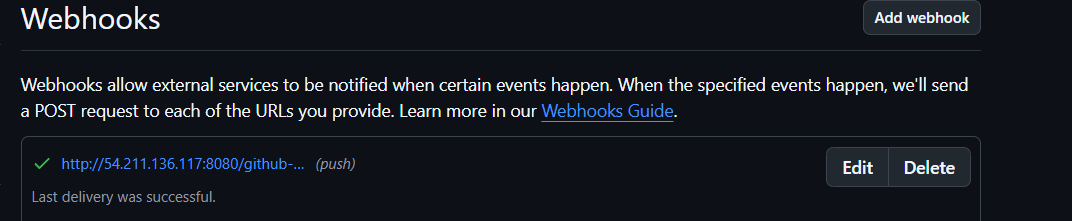
1. **Run the Playbook:**

**ansible-playbook setup.yml**

**2. Implement Git Workflow with Webhooks**

Set up a Git webhook to trigger Jenkins jobs on commits to master or develop branches.

1. **Create a Webhook in GitHub:**
   * Go to your GitHub repository.
   * Navigate to **Settings** > **Webhooks**.
   * Click **Add webhook**.
   * Enter your Jenkins server URL and specify the payload URL, e.g., http://your-jenkins-server/github-webhook/.
   * Select **Just the push event**.
   * Click **Add webhook**.



1. **Configure Jenkins:**
   * Install the GitHub plugin if not already installed.
   * Go to **Manage Jenkins** > **Configure System**.
   * Scroll to **GitHub** section and add your GitHub server details.

**3. Containerize the Application**

Make sure your Dockerfile is in the repository root. Use the pre-built container hshar/webapp and ensure the code is placed in /var/www/html within the Docker container.

**Sample Dockerfile:**

FROM httpd:2.4

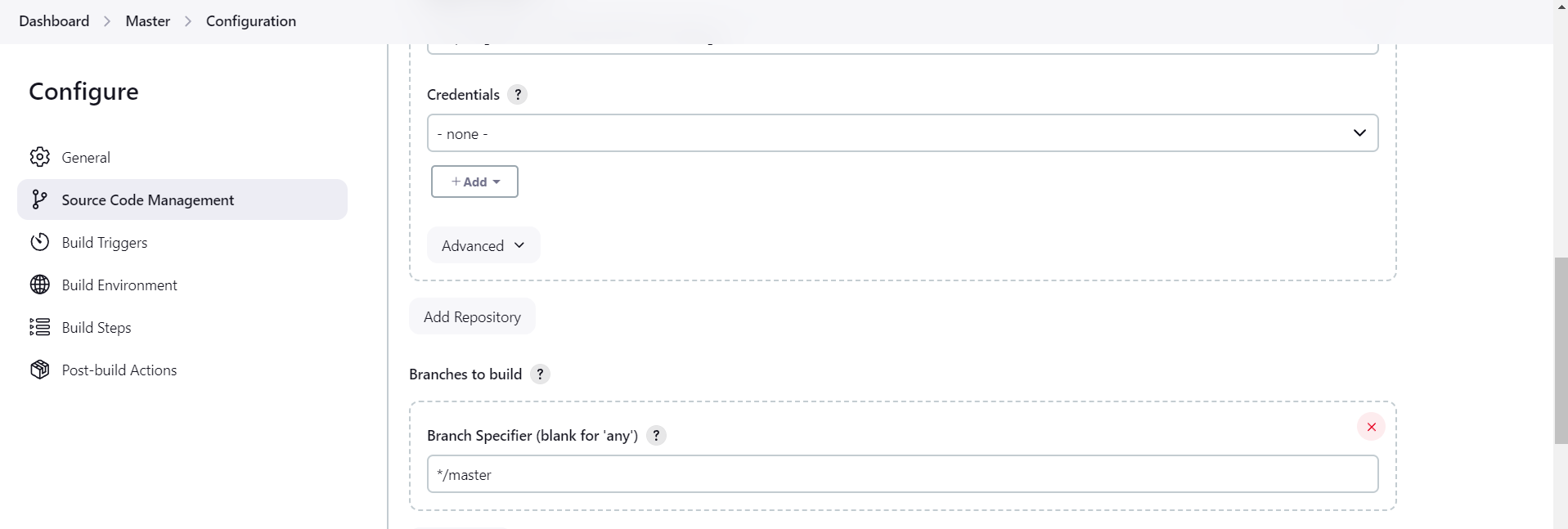
COPY ./website/ /usr/local/apache2/htdocs/

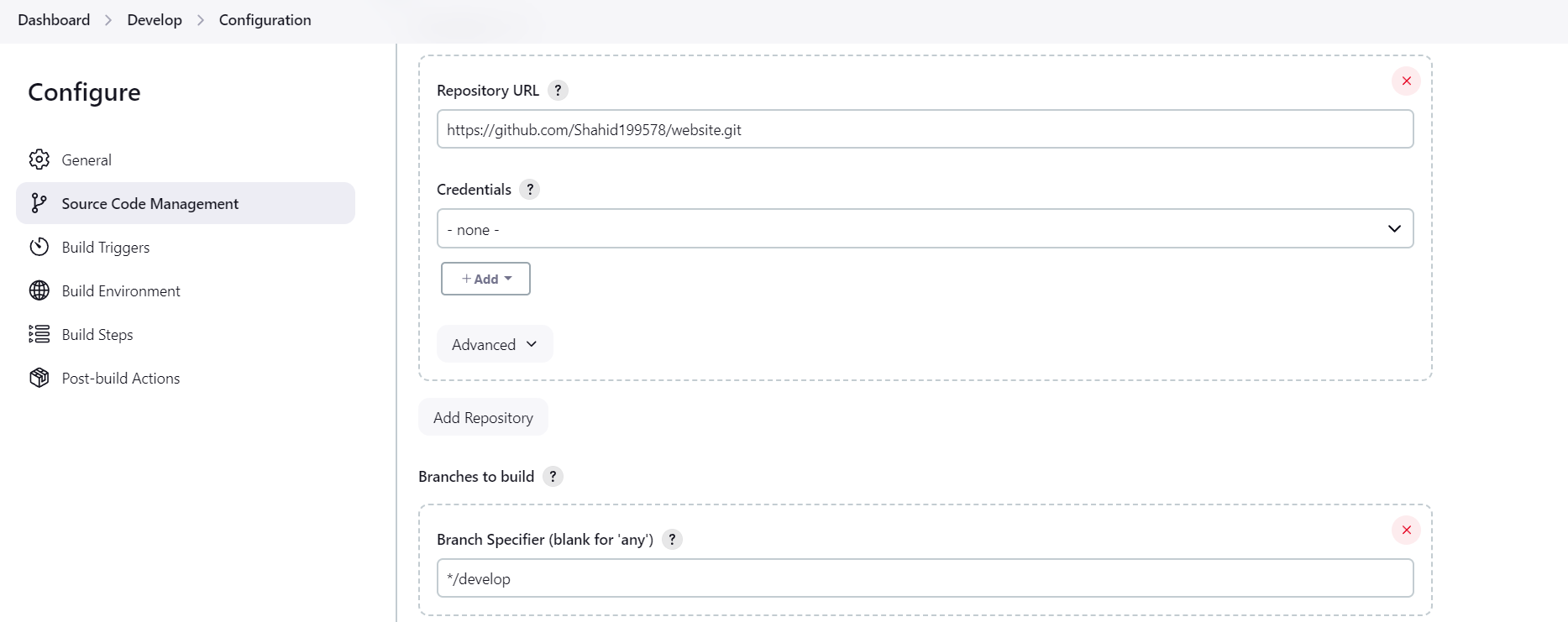
EXPOSE 80

**4. Continuous Integration with Jenkins**

**Step 1: Configure SCM Polling in Jenkins**

1. Open the **Jenkins Dashboard**.
2. Create a new **Freestyle Project**:
   * Click **New Item**.
   * Choose **Freestyle project** and give it a name (e.g., "Master and Develop").
3. Configure **Source Code Management (SCM)**:
   * In the project configuration, go to the **Source Code Management** section.
   * Select **Git** and enter the repository URL (e.g., https://github.com/Shahid199578/website.git).
   * Under **Branches to build**, specify \*/master and \*/develop as per the job name.
4. Configure Polling:
   * Go to **Build Triggers**.
   * Check the option **GitHub hook trigger for GITScm polling**.





1. **Configure Polling**:
   * Scroll to the "Build Triggers" section.
   * Check " GitHub hook trigger for GITScm polling "

**Step 2: Configure Build Job**

1. **Add Build Step**:
   * Scroll to the "Build" section.
   * Click "Add build step" and choose "Execute shell."
   * Enter the following script to build the Docker image:

**#!/bin/bash**

**docker build -t hshar/webapp -f Dockerfile .**

1. **Save Configuration**:
   * Click "Save" to save the configuration for the build job.

**Step 3: Configure Test Job**

1. **Add Test Job**:
   * Create a new Freestyle project for testing (e.g., "Test Job").
   * Configure SCM as described earlier, if it’s the same repository.
2. **Add Build Step**:
   * Scroll to the "Build" section.
   * Click "Add build step" and choose "Execute shell."
   * Enter your test commands (e.g., echo 'Running tests...' or actual test scripts).
3. **Save Configuration**:
   * Click "Save" to save the configuration for the test job.

**Step 4: Configure Deployment Job**

1. **Add Deployment Job**:
   * Create another Freestyle project for deployment (e.g., "Deploy to Prod").
   * Configure SCM as described earlier, if needed.
2. **Add Build Step**:
   * Scroll to the "Build" section.
   * Click "Add build step" and choose "Execute shell."
   * Enter the following script to deploy the Docker container:

**#!/bin/bash**

**# Remove any existing container**

**docker rm -f $(docker ps -q)**

**# Run the new Docker container**

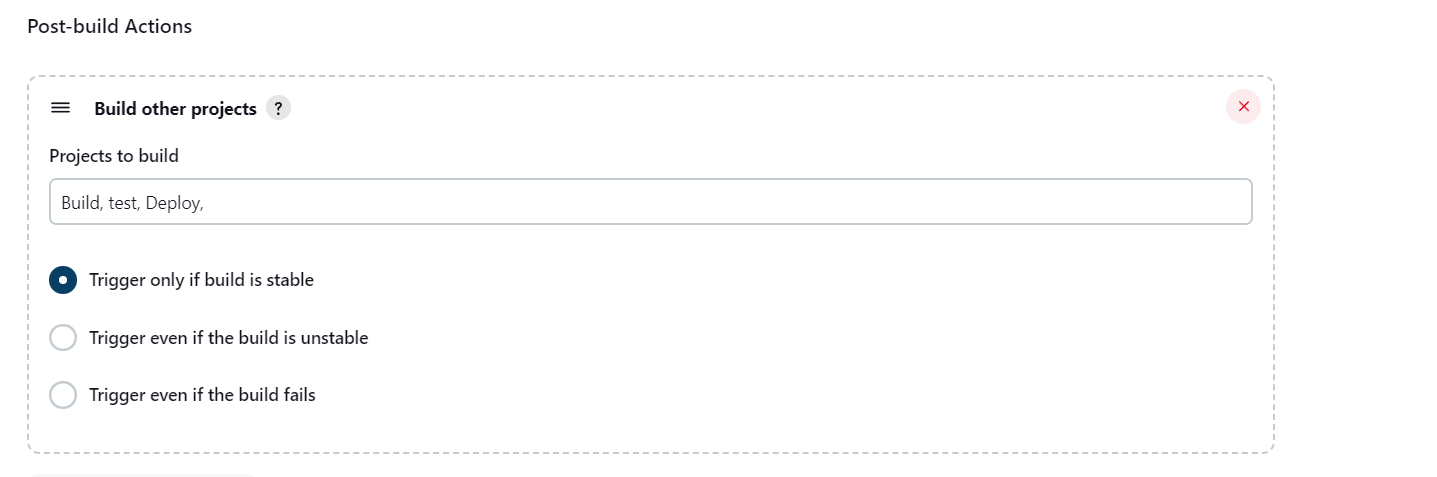
**docker run -d --name webapp -p 80:80 hshar/webapp**

1. **Save Configuration**:
   * Click "Save" to save the configuration for the deployment job.

**5. Configure Build Pipeline and Job Dependencies**

**Step 5.1: Configure Master Job**

1. **In the main project configuration page, go to the Post-build Actions section.**
2. **Add Build other projects and configure it to trigger the "Build", "Test Job", and "Deploy" jobs.**



**Step 5.2: Configure Develop Job**

1. **In the main project configuration page, go to the Post-build Actions section.**
2. **Add Build other projects and configure it to trigger the "Build" and "Test" jobs.**

