|  |
| --- |
| **Problem Statement**  AppleBite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd-party vendors.  The company’s goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers.  As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:  • Building Complex builds is difficult  • Incremental builds are difficult to manage, and deploy  To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:  Git – For version control for tracking changes in the code files  Jenkins – For continuous integration and continuous deployment  Docker – For deploying containerized applications  Ansible - Configuration management tools  This project will be about how to do deploy code to dev/stage/prod etc, just on a click of button.  Link for the sample PHP application: https://github.com/edureka-devops/projCert.git  ***Business challenge/requirement***  As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software. Post this, the code should be containerized and deployed on the test server.  The deployment should then be built and pushed to the prod server.  All this should happen automatically and should be triggered from a push to the GitHub master branch.  ***Steps for executing the solution:***  • Use the Master VM for Jenkins, Ansible, GIT etc.  • Use the fresh instance for Jenkins Slave Node (Test Server)  • Change the IP address of the VMs accordingly  • Add Build Pipeline Plugin and Post-build task plugin to Jenkins on the master VM  • Install python, openssh-server and git on the slave node manually  • Use the image devopsedu/webapp and add your PHP website to it using a Dockerfile  • Push the PHP website, and the Dockerfile to a git repository  *Below tasks should be automated through Jenkins by creating a pipeline:*  1. Install and configure puppet agent on the slave node (Job 1)  2. Push an Ansible configuration on test server to install docker (Job 2)  3. Pull the PHP website, and the Dockerfile from the git repo and build and deploy your PHP docker container. After. (Job 3)  4. If Job 3 fails, delete the running container on Test Server |

### Implementation Steps

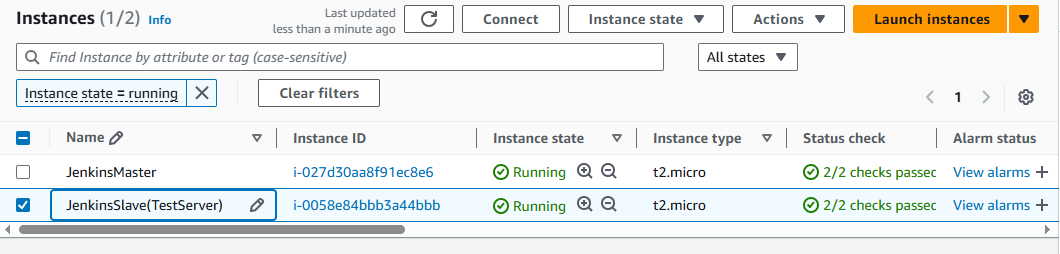
|  |
| --- |
| Step 1: Setting up the Master VM  * **Jenkins** on the Master VM, along with the required **Build Pipeline Plugin** and **Post-build task plugin**. * Configure **Git** for version control, ensuring that Jenkins has access to the Git repository. * **Install Ansible** on the Master VM to manage configurations on the Test Server.  Step 2: Jenkins Slave Node Configuration Install  * Set up a fresh VM for the **Jenkins Slave Node** (Test Server). * Install **Python**, **OpenSSH-server**, and **Git** manually on this server to enable Jenkins to communicate with it. * Configure SSH between Jenkins Master and Slave Node for secure communication.  Step 3: Dockerize the PHP Application  * Use the Docker image devopsedu/webapp as the base for the PHP application. * Create a **Dockerfile** that installs and sets up the PHP application on top of this base image. * Commit the **Dockerfile** and PHP website code to the Git repository.  Step 4: Jenkins Pipeline Jobs  * In Jenkins, create a **multi-step pipeline** with the following jobs:   **Job 1**: Install and Configure Puppet Agent on the Slave Node  This job should use Puppet to install and configure the Puppet agent on the Jenkins Slave Node, enabling consistent configurations.  **Job 2**: Push Ansible Configuration to Install Docker on the Test Server  This job will use Ansible to install Docker on the Test Server. Ansible playbooks should be configured to ensure Docker is installed and configured correctly.  **Job 3**: Deploy the PHP Application in a Docker Container  This job should:   * Pull the latest PHP application code and **Dockerfile** from the Git repository. * Build a Docker image for the PHP application. * Deploy the Docker container on the Test Server. * Configure the job to monitor for failures—if Job 3 fails, execute a rollback by stopping and deleting any running containers from the failed deployment.  Step 5: Automate the CI/CD Pipeline Trigger  * Configure a **Git webhook** to trigger the Jenkins pipeline whenever code is pushed to the master branch. This webhook will initiate the pipeline automatically.  Additional Points  * **Notifications:** Set up notifications to alert the development team if any job fails, making troubleshooting quicker. * **Automated Testing:** Consider adding a testing stage between Job 3 and production deployment to catch issues early. * **Production Deployment:** Once Job 3 passes successfully, configure a separate pipeline step for deploying the application on the production server. |

### **Solution** Implementation**:**

# Step 0: Create 2 Ubuntu VMs in aws: 1 Jenkins Master 2 Jenkins Slave

**Master VM for Jenkins**: This will handle CI/CD and orchestrate jobs for building, testing, and deploying the application.

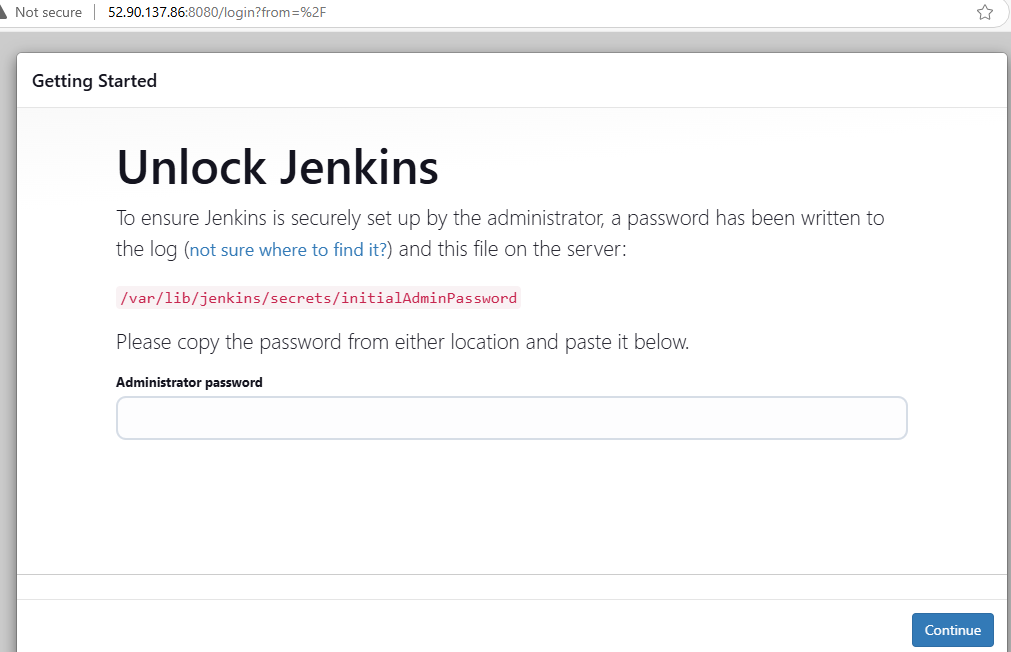
**Jenkins Slave Node (Test Server):** A fresh instance (e.g., an EC2 instance or VM) that will act as the test server.

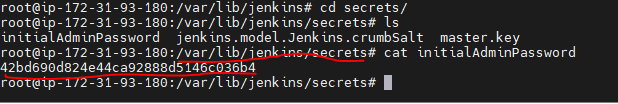


# Step 1: Setting up the Master VM

Install Pre-requisites Java Git ,Jenkins, Ansible in the JenkinsMaster

|  |
| --- |
| 1. ***Install the pre-requisites :***  # Install Jdk:  sudo -i  sudo apt update -y  sudo apt install openjdk-17-jre -y # previous version  java -version      # Install Git  sudo apt install git -y # To install git if required  2. ***Setting Environment Variables***  export JAVA\_HOME=/usr/lib/jvm/java-1.17.0-openjdk-amd64  3. ***Install the Jenkins tool :***  sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key  echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  /etc/apt/sources.list.d/jenkins.list > /dev/null  sudo apt-get update  sudo apt-get install jenkins -y  4. ***Perform Post\_Installation Activities :***  jenkins --version    systemctl status jenkins  systemctl start jenkins  systemctl enable jenkins  /var/lib/jenkins # Default Installation Dir of Jenkin on Linux Box    Enable Port 8080 in Inbound rules:    Open web browser :  http://<Public\_IP\_Address>:8080/ |







|  |
| --- |
| ***7. Install Build Pipeline Plugin and Post-build Task Plugin on the JenkinsMaster VM to create a pipeline.***    8. ***Install the Ansible tool :***  sudo apt update  sudo apt install software-properties-common -y  sudo add-apt-repository --yes --update ppa:ansible/ansible  sudo apt update -y  sudo apt install ansible -y  Ansible is installed in /etc/ansible/ directory:    ***9. Add User in this Ansible Server (i.e JenkinsMaster)***  # Ensure SSH connectivity between the Jenkins Master and Slave by generating and copying SSH keys:  useradd devopsadmin -s /bin/bash -m -d /home/devopsadmin  passwd devopsadmin    su - devopsadmin  ssh-keygen -t ecdsa -b 521    vim /etc/sudoers---add below line  #add the below mentioned line in the file and save it. ( control X control y)    devopsadmin ALL=(ALL) NOPASSWD: ALL |

# STEP 2: JenkinsSlave Node Setup – i.e TEST Server

|  |
| --- |
| 1. ***Install Pre-requisits Java:***  sudo -i  apt update -y  Install Java ::  sudo apt update -y  sudo apt install openjdk-17-jre -y  java -version  2. ***Manually install*** *python****,*** *openssh-server* ***on the*** *Jenkins Slave Node* ***to allow to communicate with the master.***  sudo apt-get update  sudo apt-get install -y python git openssh-server  ***3. Add TestUser in this Test Server (i.e JenkinsSlave)***  useradd testserveruser -s /bin/bash -m -d /home/testserveruser  passwd testserveruser    #Goto:  vi /etc/ssh/sshd\_config  #Enable *Password Authentication to Yes* and save the file    #Execute Below command to update the changes.  service ssh reload    #As a root user edit below file:  vim /etc/sudoers---add below line  #add the below mentioned line in the file and save it. ( control X control y)    testserveruser ALL=(ALL) NOPASSWD: ALL    su - testserveruser  ***4. Copying*** SSH keys ***from JenkinsMaster (AnsibleControllerMachine) to this Test Server (i.e JenkinsSlave)***  #within testserveruser home directory, create .ssh directory  /home/testserveruser  mkdir .ssh  cd /home/testserveruser/.ssh    #paste the id\_ecdsa.pub of devopsadmin user from JenkinsMaster machine (STEP1) to authorized\_keys file in this JenkinsServer (Test Server)      chmod 600 /home/testserveruser/.ssh/\*  ***5. Test*** SSH connection ***from JenkinsMaster (AnsibleControllerMachine) to this Test Server (i.e JenkinsSlave)***  Loginto JenkinsMaster and switch to user “devopsadmin”  ssh [testserveruser@172.31.15.206](mailto:testserveruser@172.31.15.206)    ***6. Configure Jenkins Slave Node by Adding a new node with the its Private IP and specify the launch method as Launch agents via SSH***    #Create tmp directory as workspace directory |

# STEP 3: Dockerize the PHP Application

Use **Docker** to containerize the PHP application and deploy it.

**Steps for Docker setup on the test server**:

|  |
| --- |
| ***1. Create a*** *Dockerfile* ***to build the container for the PHP application.***  Use the **devopsedu/webapp** Docker image and add your PHP website files into it.  # Use the devopsedu/webapp as the base image  FROM devopsedu/webapp  # Set the working directory in the container  WORKDIR /var/www/html  # Install git to clone the repository  RUN apt-get update && apt-get install -y git  # Clone the PHP application repository into the working directory  RUN git clone https://github.com/edureka-devops/projCert.git .  # Optional: Remove the .git folder to clean up the container  RUN rm -rf .git  # Expose port 80 to make the app accessible  EXPOSE 80  # Start the Apache server  CMD ["apachectl", "-D", "FOREGROUND"]    ***2.Push the PHP website, and the Dockerfile to a git repository*** |

# STEP 4: Setting Up Jenkins Pipeline Jobs

#### Job 1: Install and Configure Puppet Agent on the Slave Node

|  |
| --- |
| ***1.Create Job 1 in Jenkins:***  In Jenkins, go to New Item > Freestyle Project, name it Job1 - Install Puppet Agent.  In the Build section, add an Execute shell build step with the following script  ssh [testserveruser@172.31.41.203](mailto:testserveruser@172.31.41.203) << EOF  # Update and install Puppet agent  sudo apt update  sudo apt install puppet -y  EOF      ***2.******Save and Build the Job.***  This job installs the Puppet agent on the Test Server. |

#### Job 2: Install Docker on the Test Server (JenkinsSlave) Using Ansible

#### 

|  |  |
| --- | --- |
| ***1.*** ***Create Job 2 in Jenkins:***  In Jenkins, create a new Freestyle Project named [Job2 - Install Docker with Ansible](http://3.85.36.61:8080/job/Job2%20-%20Install%20Docker%20with%20Ansible/).    ***2.* *Set Up Ansible Playbook for Docker Installation:***  On the Jenkins Master, create an Ansible playbook named ansible-playbook-install\_docker.yml:     |  | | --- | | ---  - name: Install Docker on test server  hosts: testserver  become: yes  tasks:  - name: Update apt and install Docker  apt:  name: docker.io  state: present  update\_cache: yes  - name: Start Docker service  service:  name: docker  state: started |   3. ***Add the test server (i.e JenkinsSlave) IP (***172.31.41.203) ***in Ansible’s inventory file (/etc/ansible/hosts):***      ***4. Run Ansible Playbook in Job 2:***  Below command is added in Build -> Execute shell and Save. Run the Job  ansible-playbook /etc/ansible/ansible-playbook-install-docker.yml    We can see that docker is installed via this Jenkins job in “JenkinsSlave” i.e TestServer by checking its version: |

#### Job 3: Build and Deploy PHP Application Docker Container

|  |  |  |
| --- | --- | --- |
| ***1.*** ***Create Job 3 in Jenkins:***  In Jenkins, create a new Freestyle Project named Job3 - Deploy PHP Application in Docker.  ***2.******In the Build section, add an Execute shell step with the following commands:***   |  | | --- | | ssh [testserveruser@172.31.41.203](mailto:testserveruser@172.31.41.203) << EOF  # Clone the latest code  git clone https://github.com/edureka-devops/projCert.git  cd projCert  # Build the Docker image  sudo docker build -t php\_app:latest .  # Stop any running container with the same name  sudo docker stop php\_app || true  sudo docker rm php\_app || true  # Run the Docker container  sudo docker run -d -p 80:80 --name php\_app php\_app:latest  EOF |   ***3.******Add Post-build Action for Failure Handling:***  In the Post-build Actions section, select Post-build task.   |  | | --- | | ssh [testserveruser@172.31.41.203](mailto:testserveruser@172.31.41.203) << EOF  # Clean up failed container  sudo docker rm -f php\_app || true  EOF | |

Sample Jenkins pipeline script for Job 3:

|  |
| --- |
| pipeline {  agent any  stages {  stage('Build Docker Image') {  steps {  git 'https://github.com/edureka-devops/projCert.git'  script {  sh 'docker build -t my-php-app .'  }  }  }  stage('Deploy Docker Image') {  steps {  script {  sh 'docker run -d -p 80:80 my-php-app'  }  }  }  stage('Test Deployment') {  steps {  script {  sh 'curl -f http://localhost'  }  }  }  }  post {  failure {  script {  sh 'docker rm -f $(docker ps -q --filter "ancestor=my-php-app")'  }  }  }  } |

# STEP 5: Automate the CI/CD Pipeline Trigger

|  |
| --- |
| ***1.*** ***In the GitHub repository settings, configure a Webhook:***  Set the payload URL to http://<jenkins\_ip>:8080/github-webhook/ (i.e http://3.85.36.61:8080/github-webhook/  This will trigger the Jenkins pipeline automatically on every push to the master branch.  ***2.******Pipeline Flow:***  Job1 installs the Puppet agent.  Job2 installs Docker.  Job3 builds and deploys the application. |

# STEP 6: Testing the Pipeline

After successfully testing the deployment on the test server

1. Push a change to the master branch to trigger the webhook.

2. Monitor each Jenkins job for successful completion.

Verify the PHP application deployment by accessing the Test Server’s **Public IP** in a browser (http://<test\_server\_ip>) i.e http:// 34.230.9.254/

