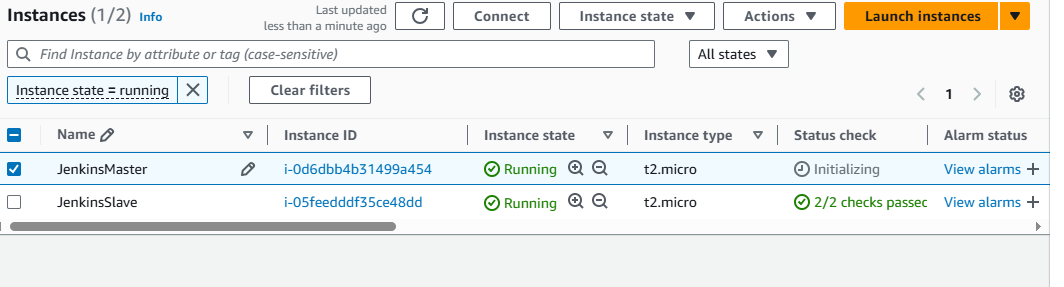
|  |
| --- |
| **Problem Statement**  AppleBite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd-party vendors.  The company’s goal is to deliver the product updates frequently to 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 |

**Solution:**

# Step1: Set Up the Jenkins Environment

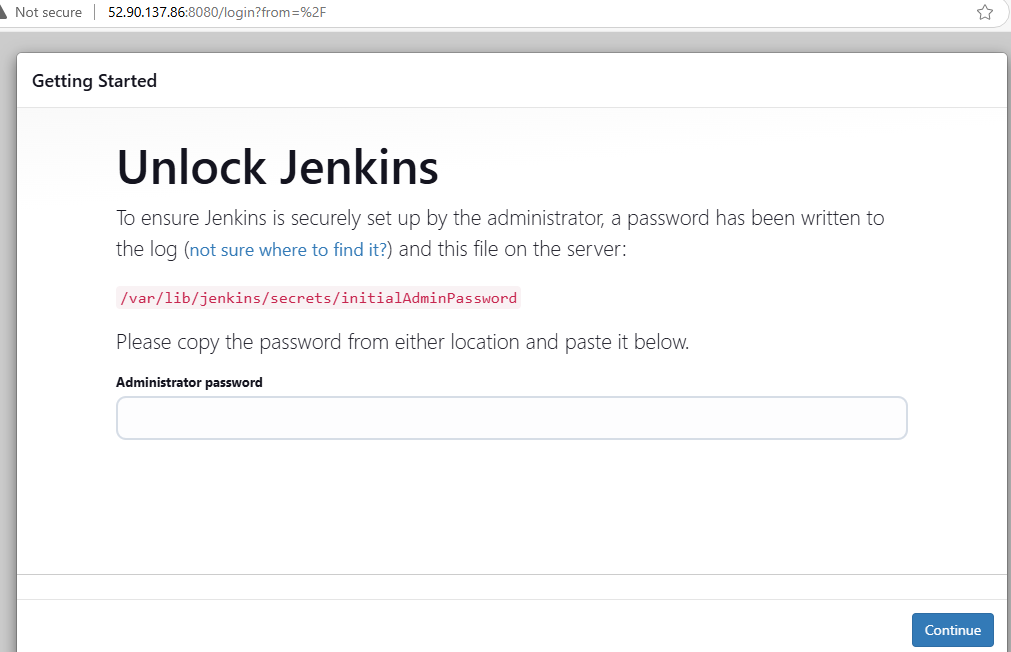
**Master VM for Jenkins**: This will handle CI/CD and orchestrate jobs for building, testing, and deploying the application.

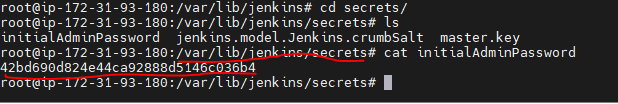
**Jenkins Slave Node (Test Server):** A fresh instance (e.g., an EC2 instance or VM) that will act as the test server.



# Step2: Install Pre-requisites and ***Jenkins*** in the **JenkinsMaster**

|  |
| --- |
| 1. ***Install the pre-requisites :***  # Install Jdk:  sudo -i  sudo apt update -y  sudo apt install git -y # To install git if required  sudo apt install openjdk-17-jre -y # previous version  java -version  2. ***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  3. ***Perform Post\_Installation Activities :***  jenkins --version    systemctl status jenkins  systemctl stop jenkins  systemctl start jenkins  systemctl restart jenkins  systemctl enable jenkins  /var/lib/jenkins # Default Installation Dir of Jenkin on Linux Box    Open web browser :  http://<Public\_IP\_Address>:8080/  E.g.: <http://44.223.65.24:8080/>  5. ***Install Maven - Build Tool :***  https://maven.apache.org/install.html  sudo apt install maven -y  mvn -v #To know version of maven  6. Setting Environment Variables  export JAVA\_HOME=/usr/lib/jvm/java-1.17.0-openjdk-amd64  export Maven\_home=/usr/share/maven |







|  |
| --- |
| 7. Install **Build Pipeline Plugin** and **Post-build Task Plugin** on the JenkinsMaster VM to create a pipeline. |

# STEP 3: JenkinsSlave Node Setup

|  |
| --- |
| sudo -i  apt update -y  Install Java ::  sudo apt update -y  sudo apt install openjdk-17-jre -y  java -version  Install GIT :  sudo apt install git -y  Manually install **python**, **openssh-server**, and **git** on the **Jenkins Slave Node** to allow Jenkins to communicate with the slave.  sudo apt-get update  sudo apt-get install -y python git openssh-server |

# STEP 4: **Docker Configuration**

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

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

* 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.

|  |
| --- |
| FROM devopsedu/webapp  COPY ./your-php-website /var/www/html  EXPOSE 80  CMD ["apache2-foreground"] |

# STEP 5: **Ansible Configuration for Docker Installation**

* **Job 1**: Install and configure the **Puppet Agent** on the slave node.
* **Job 2**: Use **Ansible** to configure the slave node by installing **Docker** on it. This can be done by creating an Ansible playbook to install Docker on the test server.

Example Ansible playbook to install Docker:

|  |
| --- |
| ---  - name: Install Docker on test server  hosts: test-server  become: yes  tasks:  - name: Update apt cache  apt:  update\_cache: yes  - name: Install Docker  apt:  name: docker.io  state: present  - name: Start Docker service  service:  name: docker  state: started |

# STEP 6: Pipeline Creation in Jenkins

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 7 : GitHub Integration with Jenkins

* Create a **GitHub webhook** to trigger the Jenkins pipeline every time code is pushed to the master branch.

In GitHub:

* Go to **Settings** → **Webhooks** → **Add webhook**.
* Set the payload URL to your Jenkins server.
* Set content type as **application/json**.
* Select **Just the push event**.

In Jenkins:

* Set up the GitHub plugin to listen for push events from GitHub and trigger the pipeline.

# STEP8: Deploy to Production

After successfully testing the deployment on the test server, you can push the container to production by adding additional stages in the Jenkins pipeline.