**CSCL1030 Applied Lab 3 - Jenkins CI/CD Pipeline for Python API**

**A black and red text

AI-generated content may be incorrect.**

**Student Name: Vinod Kumar Dhanavath  
Course: CSCL1030 –CloudOps Tools and Techniques  
Instructor: Joba Hassan  
Date: 4-14-2025**

1. **Introduction:**

In this lab, I set up a CI/CD pipeline for a Python API using Jenkins. The process included provisioning infrastructure with Terraform, setting up a Jenkins master and agent, deploying a MariaDB database, and configuring a Python App Server. The application was built and tested in a continuous integration (CI) pipeline, then deployed with a continuous delivery (CD) pipeline using Ansible.

I used the provided GitHub repositories as a base and followed the Class 8 &9 instructions to complete the setup and implementation.

**GitHub Repositories Used**

These are the GitHub repos I forked and worked with during the lab:

* JenkinsComplex
* [ContinuousDelivery](https://github.com/VinodKumarDhanavath/CloudOps_Module4_Class9_Repo3_Python_ContinuousDelivery.git)
* [ProvisionAppServer](https://github.com/VinodKumarDhanavath/CloudOps_Module4_Class9_Repo4_ProvisionAppServer.git)

**Forked Repo Links:**

* https://github.com/VinodKumarDhanavath/CloudOps\_Module4\_Class8\_Repo1\_JenkinsComplex.git
* https://github.com/VinodKumarDhanavath/CloudOps\_Module4\_Class9\_Repo4\_ProvisionAppServer.git
* https://github.com/VinodKumarDhanavath/CloudOps\_Module4\_Class9\_Repo3\_Python\_ContinuousDelivery.git

1. **Infrastructure Provisioning – Terraform Apply Output**

Using Terraform; I deployed the entire environment. This included:

* Jenkins Master
* Python Agent
* MariaDB Database Server
* Python App Server

All resources were successfully created, and the outputs confirmed that the servers were up and accessible. Following is the screenshot for the Jenkins-Master, Python-Node, python\_app\_server & dbServer.

Note: In this exercise, we provisioned the Java-Node, but it was not used as part of the lab tasks.

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

A screen shot of a computer

AI-generated content may be incorrect.

1. **AWS EC2 Instances**

After provisioning, I confirmed the infrastructure by checking the EC2 dashboard in AWS. All servers were listed and running, each with a public IP address. This confirms that the provisioning step completed as expected.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Jenkins Initial Configuration**

After the infrastructure was provisioned:

* The Jenkins master was accessed using its public IP address through a web browser
* Initial setup was completed by creating an administrative user account
* Required plugins were installed, including the Ansible plugin for deployment automation.

1. **Python Agent Setup**

The Python agent node was configured to allow secure communication with the Jenkins master:

* Connected to the agent node via SSH using its public IP
* Generated SSH authentication keys specifically for Jenkins access
* Configured the authorized keys file with proper permissions
* Added the private key to Jenkins' credential store to enable master-agent communication.

1. **Python App Server – SSH and Jenkins Setup**

To enable Jenkins to deploy to the Python App Server, I set up SSH access:

* Connected to the app server
* Switched to the python-app-server user
* Generated an SSH key pair
* Added the public key to authorized\_keys
* Created a new Jenkins credential using the private key

Then, I installed the Ansible plugin in Jenkins and added Ansible under Manage Jenkins > Tools, setting the path based on the output of the which ansible command.

This setup allowed Jenkins to connect securely to the Python App Server and run Ansible playbooks for deployment.

1. **Python\_CI – Continuous Integration Pipeline**

I created a new Jenkins pipeline job named Python\_CI using the **CloudOps\_Module4\_Class9\_Repo3\_Python\_ContinuousDelivery** repository. I set the script path to Jenkinsfile\_Integration to define the CI workflow.

This pipeline does three things:

* Runs pylint for linting the code
* Runs unit tests
* Packages the application into a tar.gz file

The job ran successfully, and all stages completed without issues.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

1. **Python\_CD – Continuous Delivery Pipeline**

Next, I created a second pipeline job named Python\_CD, again using the same repo, but this time using the Jenkinsfile\_Deployment script.

**This pipeline:**

* Transfers the packaged code to the Python App Server
* Executes the python\_deployment.yml Ansible playbook
* Configures and starts the application

The build completed successfully, and the app was deployed to the server as expected.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

1. **Curl Output – Verifying API is Running**

To confirm that the deployment was successful, I logged into the Python App Server and ran a curl command on the localhost. The output showed a valid API response, confirming that the application was running correctly on the server.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Final Notes**

All components were set up successfully:

* Jenkins pipelines for CI and CD
* Infrastructure with Terraform
* Configuration and deployment using Ansible

The app runs as expected and the full CI/CD flow is now automated.

1. **References:**

https://learn.continue.yorku.ca/pluginfile.php/1142964/mod\_lesson/page\_contents/133506/CSCL1030%20Class%208%20CICD%20Tooling%20%20Practices%20-%20Part%202.pdf?time=1709917320660

https://learn.continue.yorku.ca/pluginfile.php/1142964/mod\_lesson/page\_contents/133507/CSCL1030%20Class%209%20CICD%20Tooling%20%20Practices%20-%20Part%203.pdf?time=1709917481128

**Thank you.**