## **Install Python in Ansible Master and Client Instances**

# **sudo -i**

# **apt-get install python-minimal**

# **apt-get install python3**

# check python version with

# python --version

## **Installing Ansible in Ansible Master Instance**

# sudo -i

# apt-get update

# apt-get install software-properties-common

# apt-add-repository ppa:ansible/ansible

# apt-get update

# apt-get install ansible

# check python version with

# ansible --version

# 

# **Deployment**

The most basic continuous delivery pipeline will have, at minimum, three stages which should be defined in a Jenkinsfile: Build, Test, and Deploy. For this section we will focus primarily on the Deploy stage, but it should be noted that stable Build and Test stages are an important precursor to any deployment activity.

## **Stages as Deployment Environments**

One common pattern is to extend the number of stages to capture additional deployment environments, like "staging" or "production",

# **Pipeline**

Jenkins pipelines can be defined using a text file called **JenkinsFile.** You can implement a pipeline as code using JenkinsFile, and this can be defined by using a domain specific language (DSL). With JenkinsFile, you can write the steps needed for running a Jenkins pipeline.

The benefits of using **JenkinsFile are**:

* You can create pipelines automatically for all branches and execute pull requests with just one **JenkinsFile.**
* You can review your Jenkins code on the pipeline
* You can audit your Jenkins pipeline
* This is the singular source for your pipeline and can be modified by multiple users.

JenkinsFile can be defined by either Web UI or with a Jenkins File.

**Declarative versus Scripted pipeline syntax:**

There are two types of Jenkins pipeline syntax used for defining your JenkinsFile.

1. Declarative
2. Scripted

**Declarative:**

Declarative pipeline syntax offers an easy way to create pipelines. It contains a predefined hierarchy to create Jenkins pipelines. It gives you the ability to control all aspects of a pipeline execution in a simple, straight-forward manner.

**Scripted:**

Scripted Jenkins pipeline runs on the Jenkins master with the help of a lightweight executor. It uses very few resources to translate the pipeline into atomic commands. Both declarative and scripted syntax are different from each other and are defined totally differently.

## The pipeline block consists of all the instructions to build, test, and deliver software. It is the key component of a Jenkins Pipeline.

## An agent is assigned to execute the pipeline on a node and allocate a workspace for the pipeline.

## A stage is a block that has steps to build, test, and deploy the application. Stages are used to visualize the Jenkins Pipeline processes.

## A step is a single task to be performed, for example, create a directory, run a docker image, delete a file, etc.

## The Groovy code above, I am using for the JenkinsFile. Any available agent is getting assigned to the pipeline. Then I am defining the Build stage and performing a simple echo step. Then I defined the Test stage where the step asks whether you want to proceed or not. After that, I have created a Deploy stage, which has two more stages in it running in parallel. Deploy start stage has a step with echo command, and Deploying now has a step that pulls a docker image of Nginx on the node. Finally, there is a Prod stage with a simple echo step.

## **Understanding the JenkinsFile.**

1. In case of Nomura, to avoid Permission Issues - Use Linux && Region-as while declaring the node.
2. Adding Parameters, Variables in the Properties of the Node.
3. Once Properties are defined, we declare the first Stage - Checkout SCM. Checking out SCM checks for the latest deployment/commitment in to the git repository.
4. Here we are deploying the Ansible Playbook by Jenkins - Providing all the variables/credentials/basic directory path of the the Playbook and then applying code to run the Playbook.
5. Stage 2 - Deploy the Ansible Book. We use the anisble box and run the Ansbile Playbook. Post Deploy we check for any Pre-Release Checks/ Ansibel Smoke Tests/ API Checks etc.

If any error was caught, the Jenkins deployment is considered as Failure.

We run the Cp-terraform-scripts-to-jumpbox terraform code to create the cloud Inventory using the Terraform code.

1. Now we convert the Terraform template into actual terraform script by replacing variables.
2. Now we execute the Terraform scripts which includes
   1. Deploy Monitoring on Proteus Target Server
   2. Deploy MINIO Monitoring

## **Running Jenkins pipeline**

Click on **Run** to run the Jenkins pipeline.