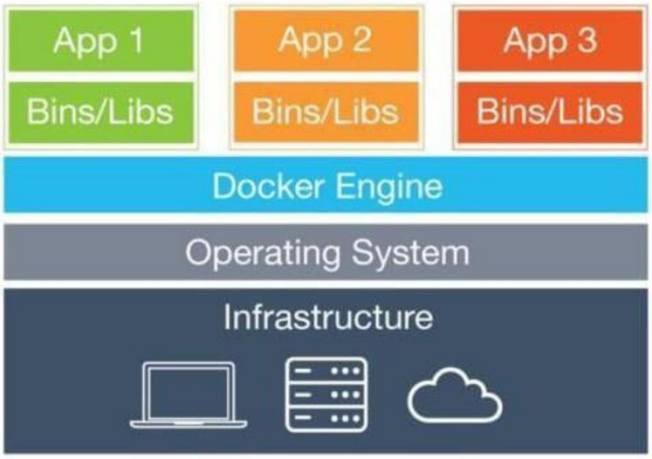
Experiment 4

Aim: Installation and configuration of Docker

Theory:

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. Because all of the containers share the services of a single operating system kernel, they use fewer resources than virtual machines.

Containers isolate applications' execution environments from one another, but share the underlying OS kernel. They're typically measured in megabytes, use far fewer resources than VMs, and start up almost immediately. They can be packed far more densely on the same hardware and spun up and down en masse with far less effort and overhead. Containers provide a highly efficient and highly granular mechanism for combining software components into the kinds of application and service stacks needed in a modern enterprise, and for keeping those software components updated and maintained.



Installing Docker:

1. Update your existing list of packages:

```
$ sudo apt update
```

2. Install a few prerequisite packages which let apt use packages over HTTPS:

```
$ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

3. Add the GPG key for the official Docker repository to your system:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Add the Docker repository to APT sources:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
```

5. Update the package database with the Docker packages from the newly added repo:

```
$ sudo apt update
```

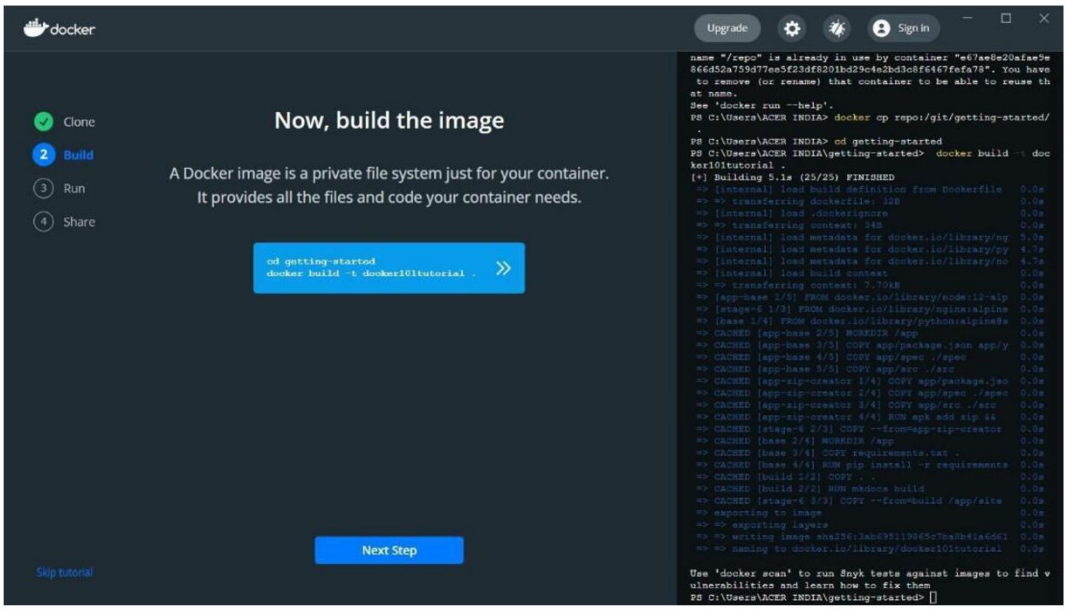
6. Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

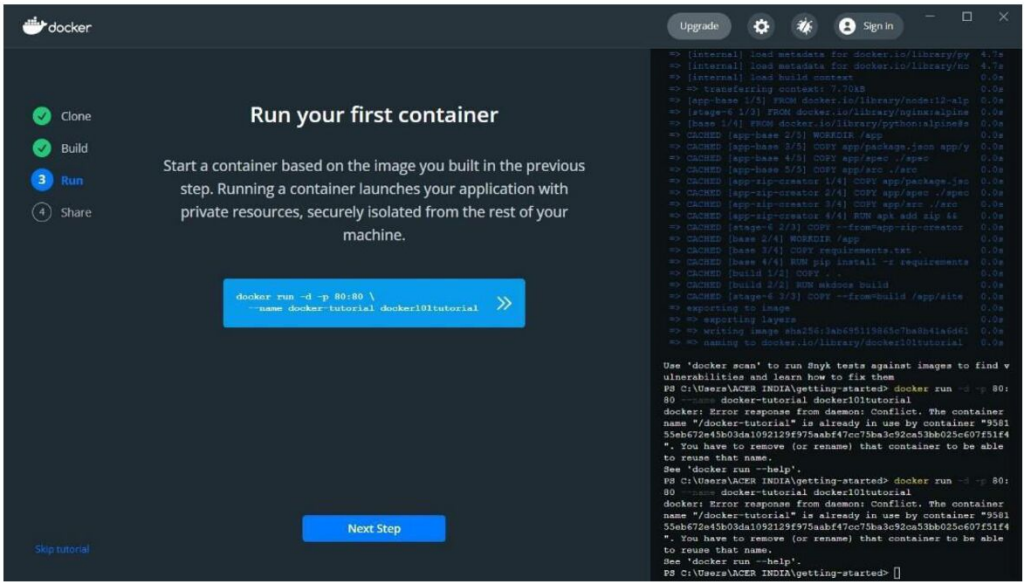
```
$ apt-cache policy docker-ce

docker-ce:
  Installed: (none)
  Candidate: 18.03.1~ce~3-0~ubuntu
  Version table:
     18.03.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64
Packages
```

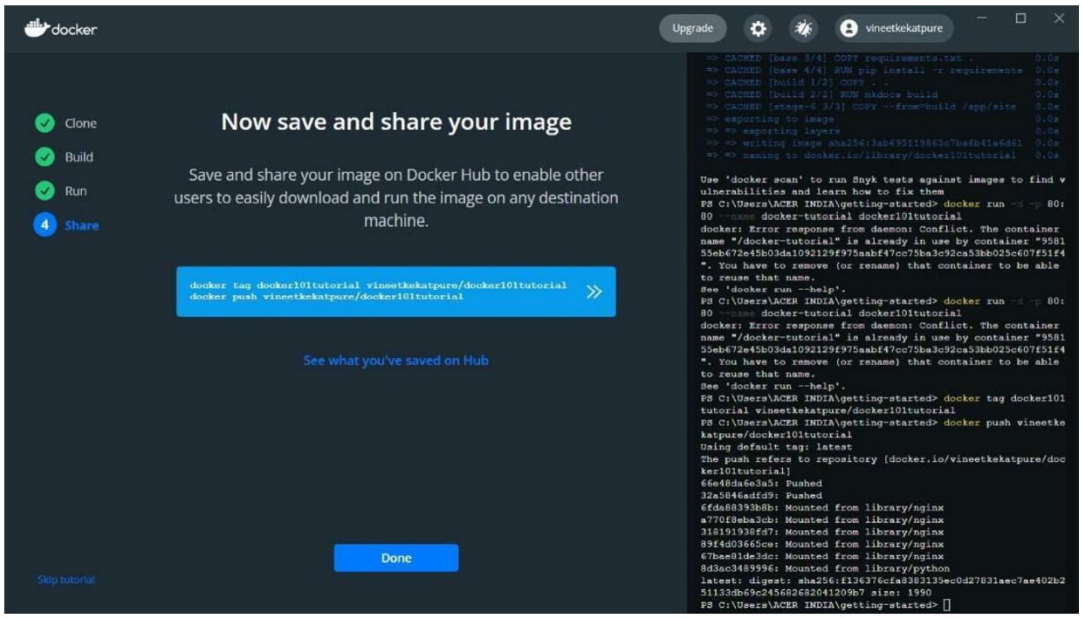
7. Install Docker:

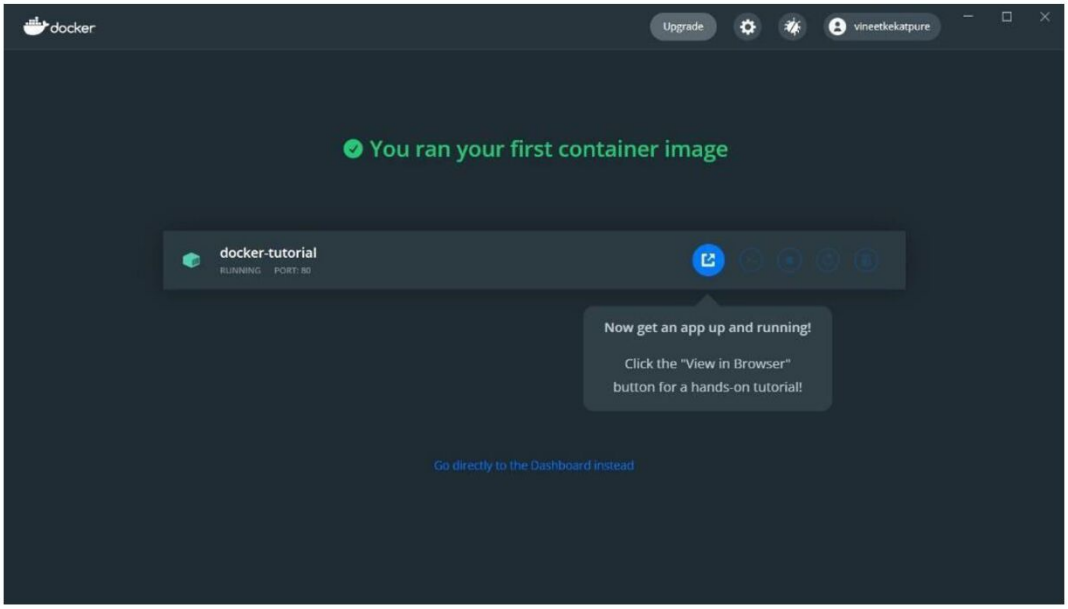
   $ sudo apt install docker-ce

Using Docker:










```
C:\Users\ACER INDIA>docker --version
Docker version 20.10.12, build e91ed57
```

This command shows the version of docker installed

```
C:\Users\ACER INDIA>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: vineetkekatpure
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.c
om/go/access-tokens/
```

This command logs you into docker

```
C:\Users\ACER INDIA>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
08c01a0ec47e: Pull complete
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

This command pulls images from the central Docker Repository

```
C:\Users\ACER INDIA>docker images
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
docker101tutorial                   latest    3ab695119865   21 minutes ago   28.8MB
vineetkekatpure/docker101tutorial   latest    3ab695119865   21 minutes ago   28.8MB
ubuntu                              latest    54c9d81cbb44   2 weeks ago      72.8MB
alpine/git                          latest    c6b70534b534   3 months ago     27.4MB
```

This command shows all the Docker images downloaded on the user's system.

```
C:\Users\ACER INDIA>docker run -it -d ubuntu
9cfee894116275eaca4b84e57977821f9d2ec9528fa6fc7b66e79aadbe09d8ea
```

This command runs containers from their image name.

```
C:\Users\ACER INDIA>docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                NAMES
9cfee8941162   ubuntu             "bash"                   21 seconds ago   Up 20 seconds                        funny_solomon
958155eb672e   docker101tutorial  "/docker-entrypoint.…"   24 minutes ago   Up 24 minutes   0.0.0.0:80->80/tcp   docker-tutorial
```

This command lists all the containers currently running on the system

```
C:\Users\ACER INDIA>docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS                      PORTS                NAMES
9cfee8941162   ubuntu             "bash"                   42 seconds ago   Up 40 seconds                                    funny_solomon
958155eb672e   docker101tutorial  "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes               0.0.0.0:80->80/tcp   docker-tutorial
e67ae8e20afa   alpine/git         "git clone https://g…"   29 minutes ago   Exited (0) 29 minutes ago                        repo
```

Adding "-a" to the above command shows if there are any stopped containers/

```
C:\Users\ACER INDIA>docker exec -it 9cfee8941162 bash
root@9cfee8941162:/#
```

For logging in a container, "exec" command is used.

```
C:\Users\ACER INDIA>docker stop b110e1c732eb6546a6c1a0425bb99671524b27adc5df7894d899080cb698edc2
b110e1c732eb6546a6c1a0425bb99671524b27adc5df7894d899080cb698edc2
```

This command stops the running container.

```
C:\Users\ACER INDIA>docker kill ba817fba58f35cdf0d8afa2fd4f8a87daebf366d7fd093dc83e8ddf74e40be5e
ba817fba58f35cdf0d8afa2fd4f8a87daebf366d7fd093dc83e8ddf74e40be5e
```

This command kills the container by stopping its execution immediately. The difference between 'docker kill' and 'docker stop'. 'docker stop' gives the container time to shutdown gracefully, in situations when it is taking too much time for getting the container to stop, one can opt to kill it

Conclusion: Thus we have successfully installed and configured Docker.

EXPERIMENT – 5

AIM - To Install and Configure Docker for creating Containers of different Operating System Images and running docker file.

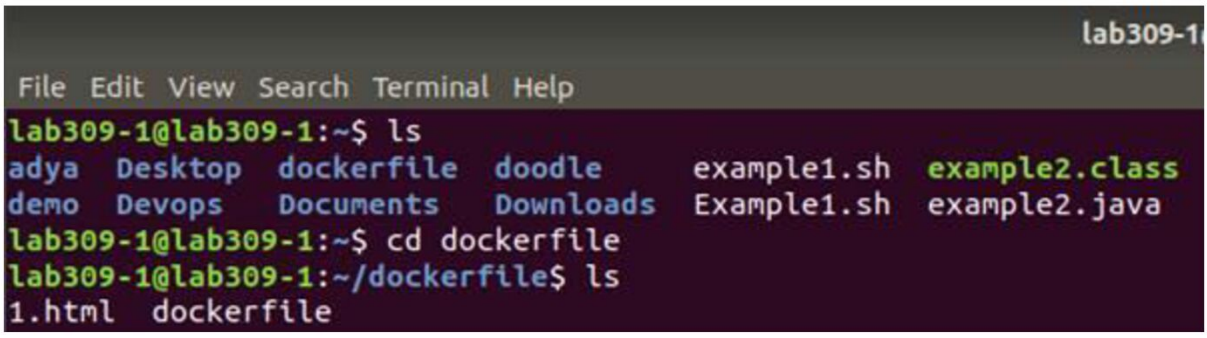
LAB OUTCOME - Analyze & Illustrate the Containerization of OS images and deployment of applications over
Docker

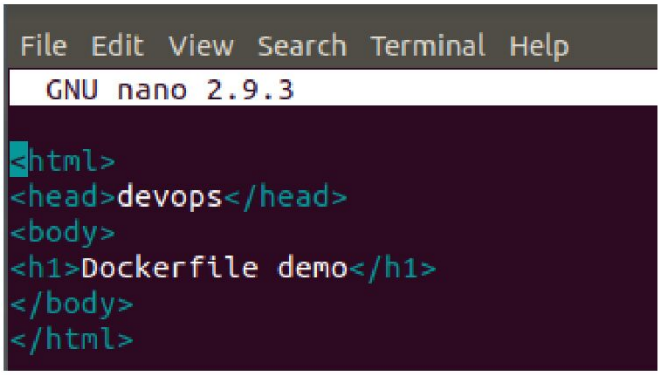
THEORY - Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.

Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides simple commands you can use to build, start, or stop containers.

PROCEDURE -

Steps to create dockerfile





```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                                          dockerfile

FROM ubuntu

ENV TZ=Asia/Dubai
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

RUN apt-get update
RUN apt-get -y install apache2
ADD . /var/www/html
ENTRYPOINT apachect1 -D FOREGROUND
ENV name Intellipaat
```

Steps to build image of dockerfile

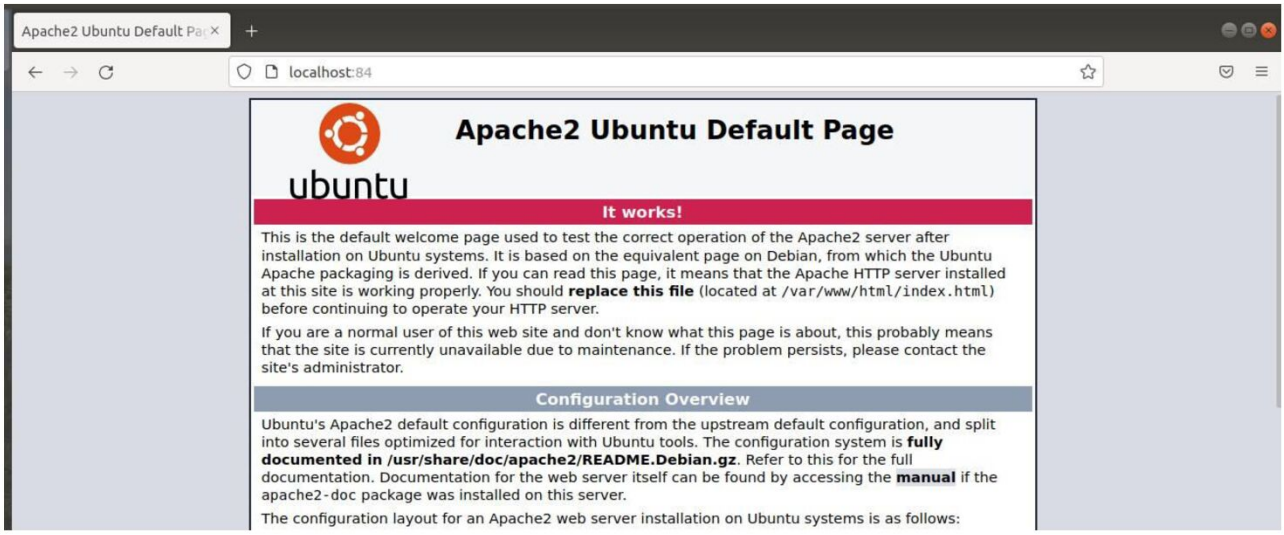
```
lab309-1@lab309-1:~$ sudo docker build dockerfile
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM ubuntu
 ---> 54c9d81cbb44
Step 2/6 : RUN apt-get update
 ---> Running in d56bf6c06e0b
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [982 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [842 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1579 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
```

```
invoke-rc.d: could not determine current runlevel
```

```
File  Edit  View  Search  Terminal  Help
ab309-1@lab309-1:~/dockerfile$ sudo docker images
EPOSITORY       TAG       IMAGE ID       CREATED              SIZE
none>           <none>    41968ef10331   About a minute ago   220MB
none>           <none>    a5e7ef134c92   6 minutes ago        107MB
none>           <none>    d8e7e49c6443   24 minutes ago       107MB
ew_ubuntu       latest    b62931d33d3b   7 days ago           72.8MB
buntu           latest    54c9d81cbb44   3 weeks ago          72.8MB
ockerfile       latest    2c473978044b   24 months ago        189MB
ew_dockerfile   latest    e3d0cfb688ca   24 months ago        189MB
buntu           <none>    72300a873c2c   2 years ago          64.2MB
ello-world      latest    fce289e99eb9   3 years ago          1.84kB
ab309-1@lab309-1:~/dockerfile$ sudo docker run -it -p 84:80 -d dockerfile
177d7874deecf3ac8829479ea33a6d50c51535f34561e84c76d4749ed21b19d
ab309-1@lab309-1:~/dockerfile$
```

```
Step 8/8 : ENV name Intellipaat
 ---> Running in 39fd7a8962e3
Removing intermediate container 39fd7a8962e3
 ---> 41968ef10331
Successfully built 41968ef10331
```

```
lab309-1@lab309-1:~/dockerfile$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS                          PORTS
  NAMES
6177d7874dee   dockerfile       "/bin/sh -c 'apachec…"   28 seconds ago   Up 25 seconds                   0.0.0.0:84->80/tcp, :::84->80/tcp
  romantic_davinci
8db925a214ff   new_dockerfile   "/bin/sh -c 'apachec…"   24 months ago    Restarting (127) 13 seconds ago
  hopeful_babbage
MySQL Workbench
lab309-1@lab309-1:~/dockerfile$ sudo docker exec -it 6177d7874dee bash
root@6177d7874dee:/# exit
exit
lab309-1@lab309-1:~/dockerfile$
```



CONCLUSION – Successfully used Docker for creating Containers of different Operating System Images and running dockerfile.

EXPERIMENT – 6

AIM - To Install and Configure Jenkins for continuous integration purpose.

LAB OUTCOME – Understand the importance of Jenkins to Build, Deploy and Test Software Applications

THEORY -

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making i easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies. To install Jenkins following software packages are required

1) GIT
2) Notepad++
3) Latest Java development kit
4) Jenkins
5) Apache Maven

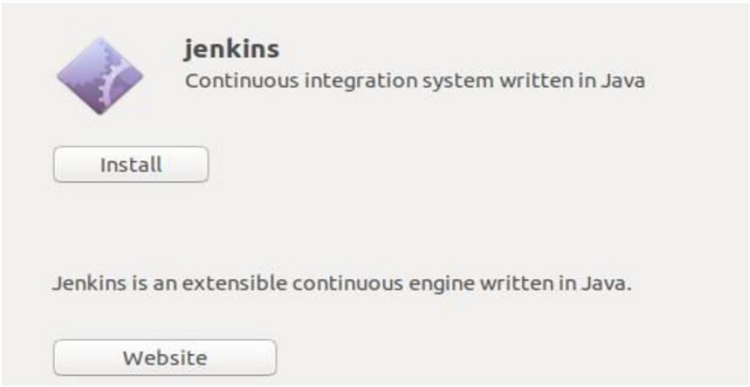
PROCEDURE -

```
lab309-1@lab309-1:~$ sudo apt-get update
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
```

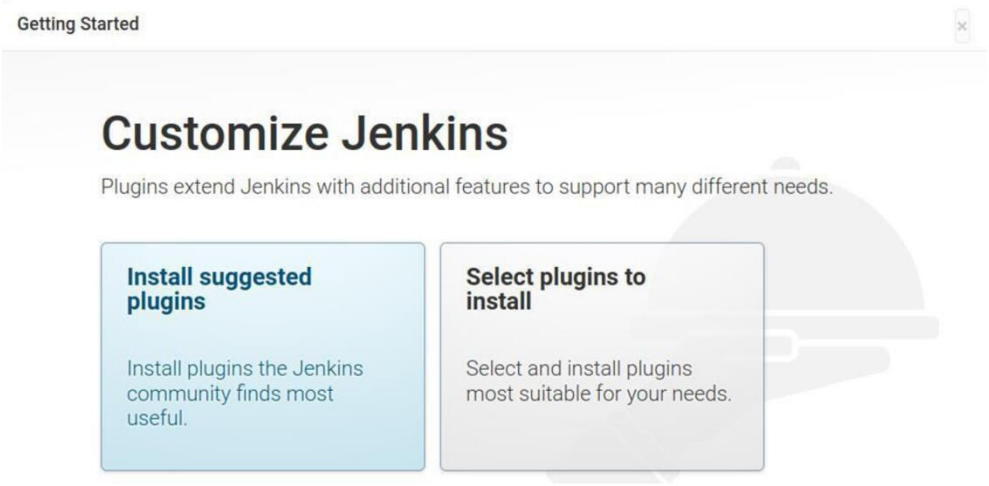
```
lab309-1@lab309-1:~$ javac -version
javac 11.0.14
lab309-1@lab309-1:~$
```

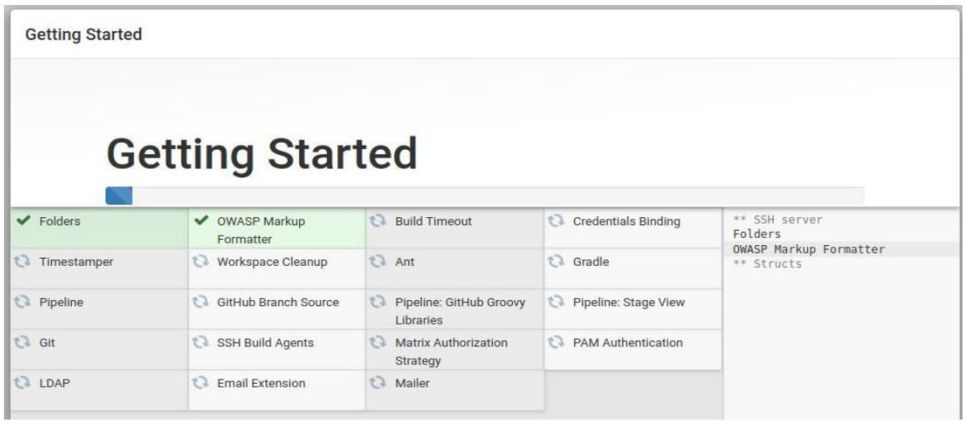
```
lab309-1@lab309-1:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jdk is already the newest version (2:1.11-68ubuntu1~18.04.1).
The following packages were automatically installed and are no longer required:
  libllvm9 linux-headers-4.15.0-163 linux-headers-4.15.0-163-generic
  linux-image-4.15.0-163-generic linux-modules-4.15.0-163-generic
  linux-modules-extra-4.15.0-163-generic python3-click python3-colorama
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
```

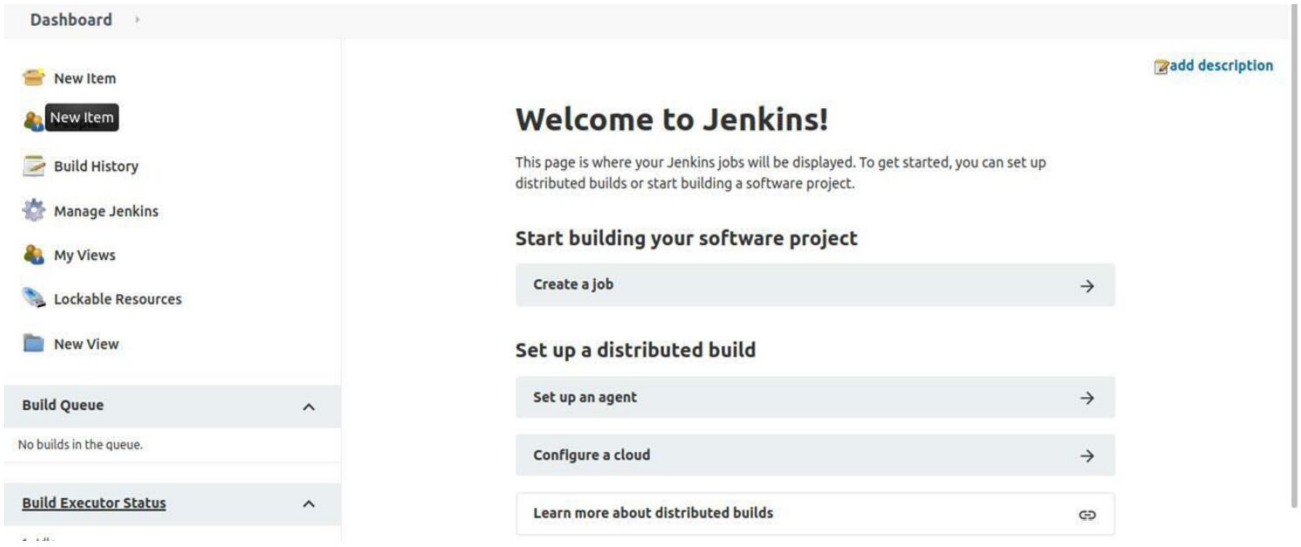
Parent Directory		-
FOOTER.html	2022-02-09 12:16	3.9K
jenkins_1.409.1_all.deb	2011-06-09 00:24	37M
jenkins_1.409.2_all.deb	2011-09-13 16:32	43M
jenkins_1.409.3_all.deb	2011-11-08 20:40	43M
jenkins_1.424.1_all.deb	2011-11-30 21:15	40M
jenkins_1.424.2_all.deb	2012-01-10 23:49	40M
jenkins_1.424.3_all.deb	2012-02-27 20:07	38M




```
lab309-1@lab309-1:~$ sudo cat /var/snap/jenkins/2126/secrets/initialAdminPassword
1406c9a2d418413ca19b366d0c5d31a8
lab309-1@lab309-1:~$
```







CONCLUSION – Successfully installed Jenkins for continuous integration purpose.

Experiment – 7

AIM: Deploying freestyle app in Jenkins.

LO: LO2 – Understand the importance of Jenkins to build, deploy and test software applications.

THEORY:

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible." In simple way, Continuous integration (CI) is the practice of frequently building and testing each change done to your code automatically.

Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
Our first job will execute the shell commands. The freestyle project provides enough options and features to build the complex jobs that you will need in your projects.

PROCEDURE:

Example 1.1 - The Steps for deploying a simple free style project in Jenkins is as follows:-

Step 1-: Click on Create new jobs.
Step 2-: Now Specify name to the project as "Example1", select Option "Free style project" and click on OK button.

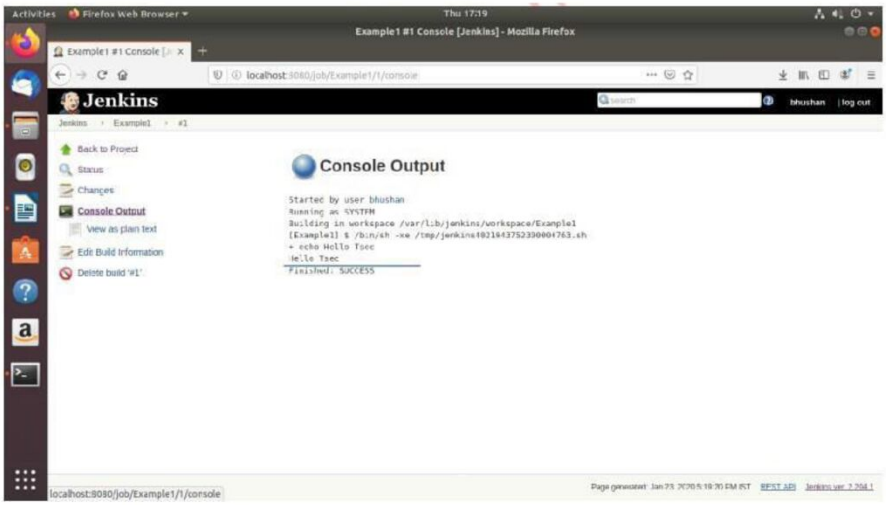
Step 3-: In this project we are going to learn how to run simple shell script on Jenkins. So, Click on Build option select Execute script from dropdown menu.

Step 4-: Now Write a Simple Shell command to print the text as Like given below.

Now click on apply followed by save button.

Step 5-: No Build a project to see the output Click on our first build "1" followed by console output to see the output

Click on our first build "1" followed by console output to see the output



***Example 1.2**:*

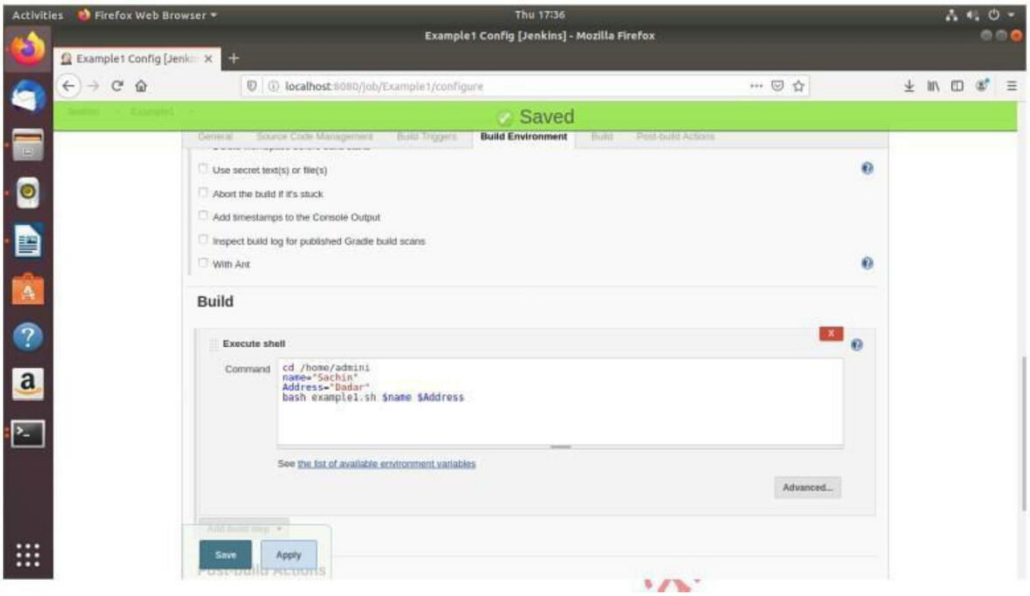
Now let us take parameters through files. So, create a new shell script file in local directory.

Step 1: First run the shell script locally with no parameter, one parameter and two parameters.

Step 2: Let us run it through Jenkins Shell. To change existing program, click on configure option and then modify the script.

<Here, change directory to the path where you have stored your file. Youkan get the location by pwdcommand>

Variation To This Program:



Output

Example 2 - Running a Java Program under Jenkins

Step 1 -: Write a java program and test it locally. Now create freestyle project example2 in Jenkins.

Step 2 -: Go to build option and change path to directory where you have stored java file followed by compile and run program commands.

Step 3 -: Now if your build fails due to permission problem then give write permission to directory underneath it.

Step 4 -: And remove existing .class file from your current directory

Step 5 -: And remove existing .class file from your current directory

Step 6 -: Now after doing this your build gets successful and get results

Example 3 - Parameterize Build

In this program we are going to see how to provide parameters during runtime to your shell script or java program.

Step 1-: Create a free style project example3 by clicking on new item followed by specifying project name and free style project.

Step 2-: Now under general menu, select option this project is parameterize

Step 3-: Select String parameter and specify name as “First-Name”

Step 4-: Again, click on add parameter and select choice parameter Take second parameter as choice box.

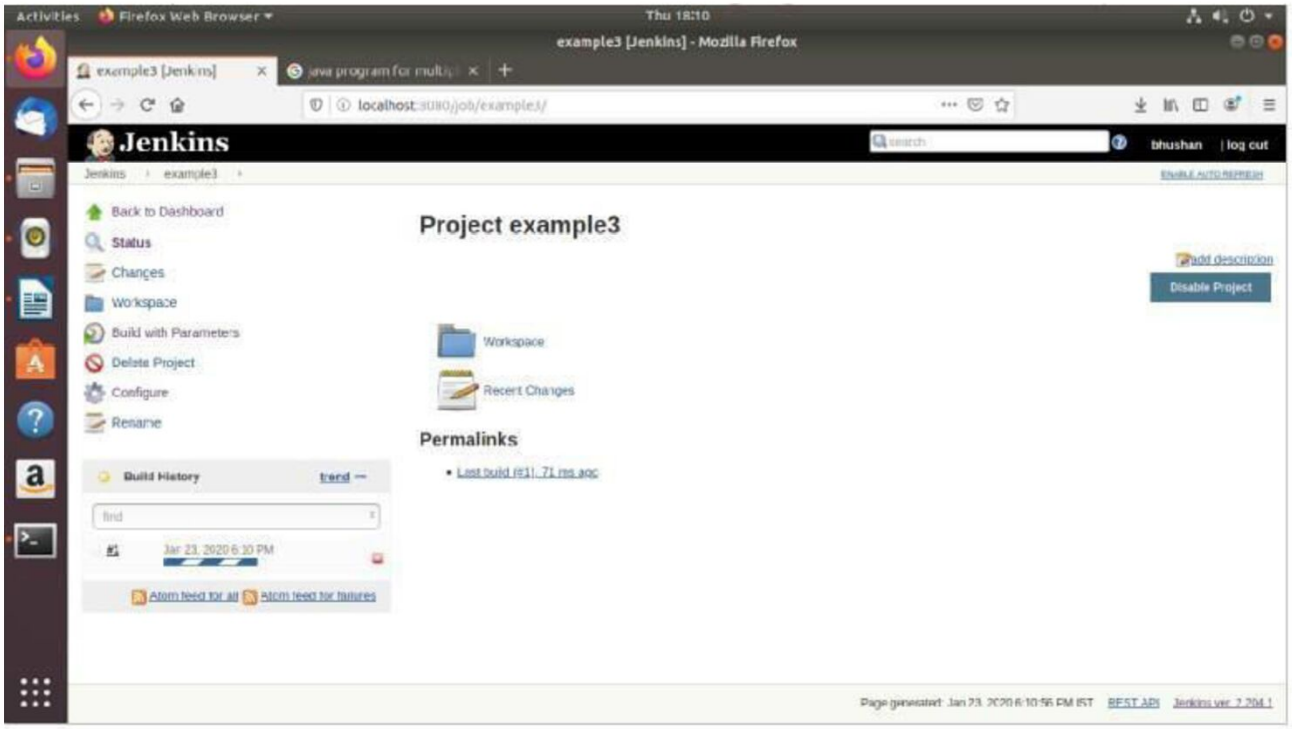
Step 5-: Specify name as “City” and add the choices in each line

Step 6-: Write a shell script that takes 2 parameters with command line arguments name and city.

Step 7-: Now, go back to Jenkins, Select Build option, give the path and write script as shown below

Step 8-: Now click on build with parameters and specify the values. Click on Build.

Step 9-: Go to console to see the output



Example 4 - Running a Java program with parameters

Step 1-: Write a java program for multiplication table with command line arguments and test it locally.

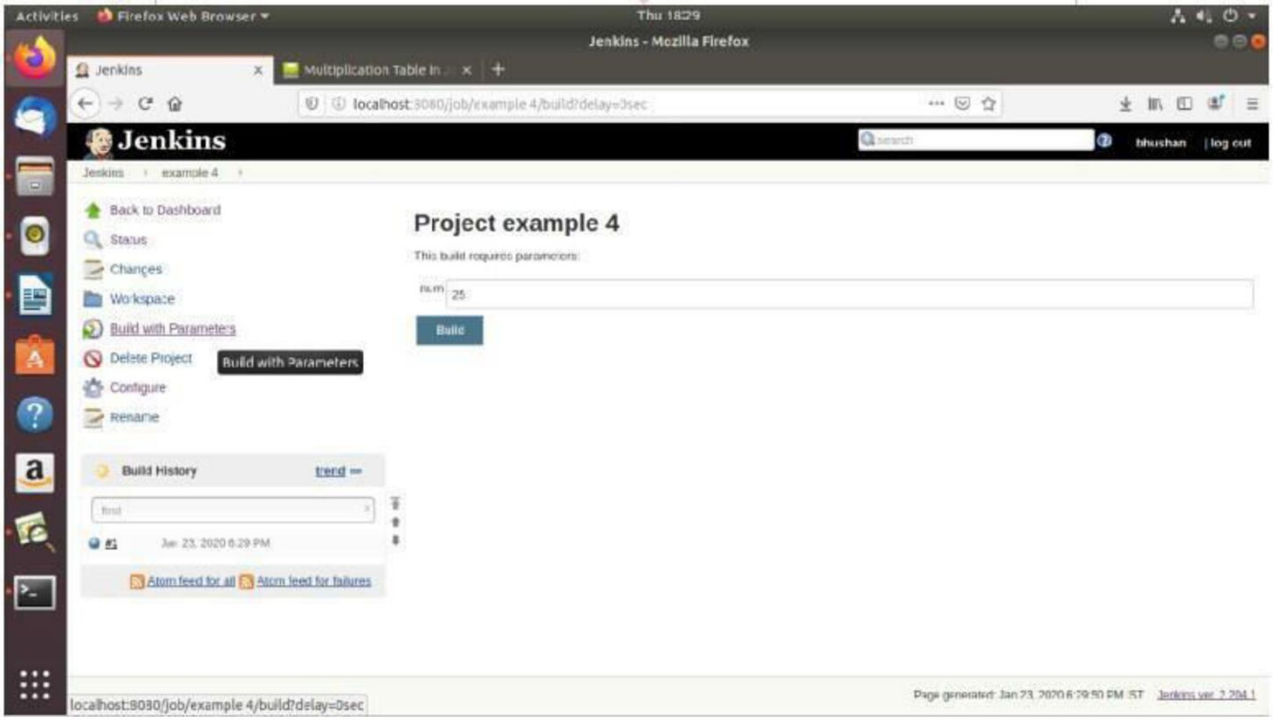
Step 2-: Delete class file and give write permission to program

Step 3-: Now create new Jenkins project “example4”

Step 4-: Go to build option and select execute shell. Write the commands with changing path to directory where you have stored java program as below.

Step 5-: Now click on Save. Select build with parameter option and specify value of “num” whose multiplication tables needs to be displayed.

Step 6-: Click on build to see the output as below.



Example 5 - Running a python program in Jenkins

Step 1-: Write and test python3 program locally

OUTPUT

Step 2-: Now create a new item followed by specifying name “example5” and select freestyle project

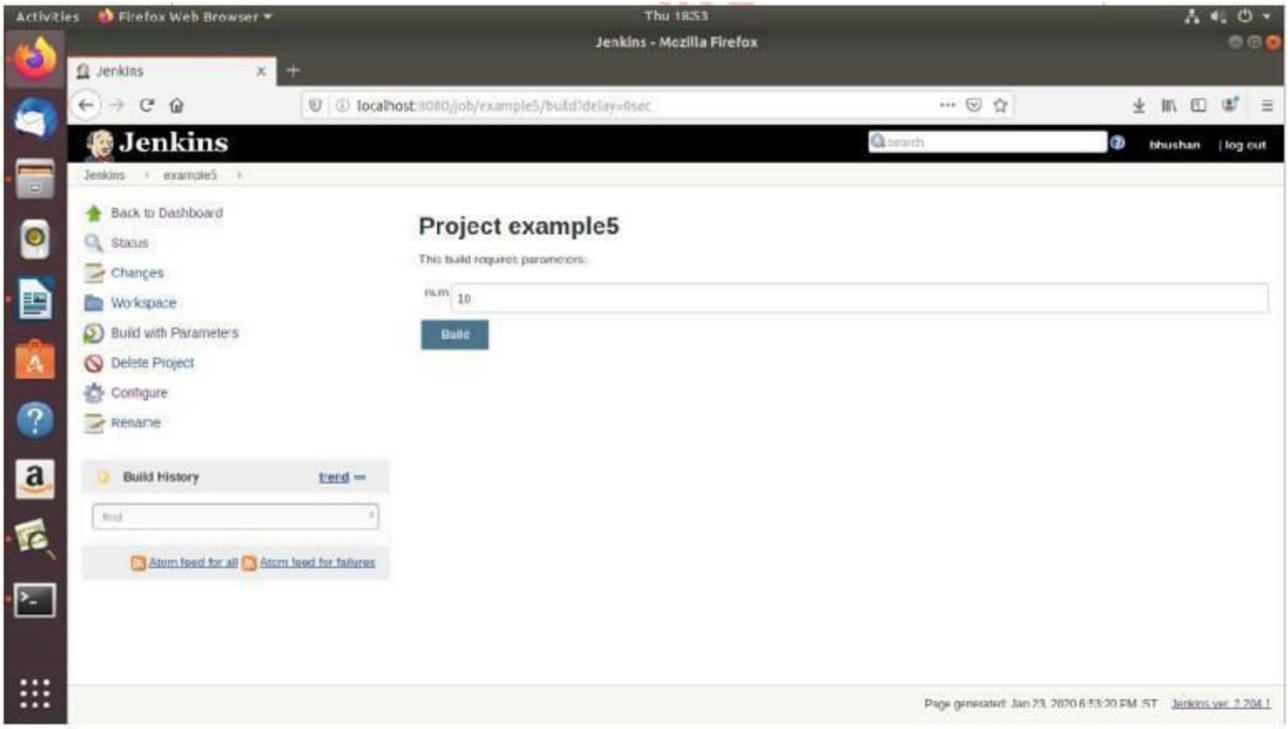
Step 3-: Now select, this project is parameterized, select string parameter and specify parameter name as “num”

Step 4-: Go to build option and write script as follows

Step 5-: Now, select build with parameter and specify num value in textbox, let’s say 10.

Step 6-: Click on build followed by console output to see result.

Step 7-: The Output of program will be shown as below.



CONCLUSION: Hence, we studied and deployed freestyle app in Jenkins.

EXPERIMENT – 8

AIM - To Configure and use Anisible for continuous integration purpose.

LAB OUTCOME – [LO5 LO6] – Summarise the importance of software configuration management in DevOps. Synthesize provisioning using Anisible/Puppet/Saltstack.

THEORY -

Because Ansible requires a Python interpreter (in order to run its modules), we need to install Python as well. For that, issue the command:

```
$ sudo apt-get install python3 -y
```

PROCEDURE - Configure SSH access to the server

Step 1

We need to make it possible for our node to access the Ansible server. We do this via Secure Shell (SSH). Copy the server's SSH public key to the node. If your server doesn't have a key yet, generate one with the command:

```
$ ssh-keygen
```

Step 2

Open your terminal either by using the Ctrl+Alt+T keyboard shortcut or by clicking on the terminal icon and install the openssh-server package by typing:

```
apt update
apt install openssh-server
```

Step 3

Check the status of ssh server using the following command

```
systemctl status ssh
```

Step 4

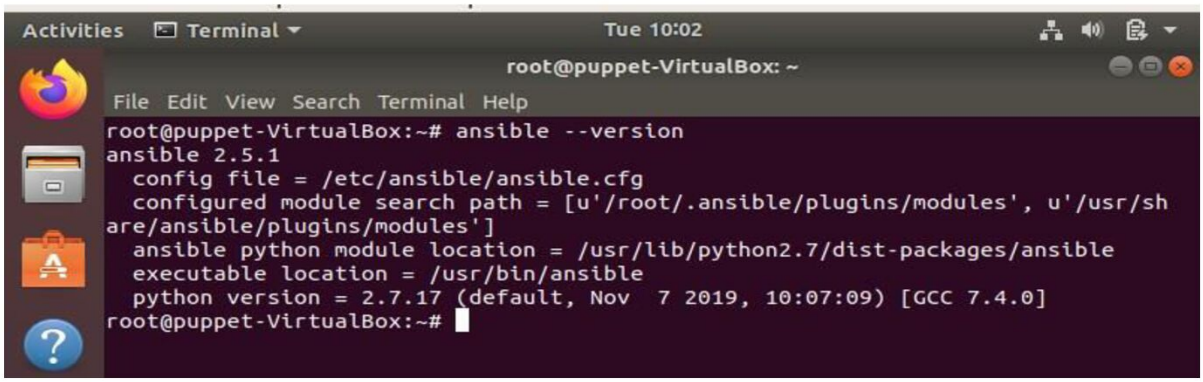
Ubuntu comes with a firewall configuration tool called UFW. If the firewall is enabled on your system, make sure to open the SSH port:

```
$ ufw allow ssh
```

Step 5

Installing Ansible on server. Use the command “sudo apt install ansible”

Confirm the ansible installation by checking its version. Check ansible hosts device by viewing ansible host file. View the ansible inventory.



CONCLUSION – Installed and configured anisible successfully.

Experiment 9

Aim: To install and configure Software Configuration Management using Puppet

Theory:

What is puppet ?

Puppet is a configuration management tool developed by Puppet Labs in order to automate infrastructure management and configuration. Puppet is a very powerful tool which helps in the concept of Infrastructure as code. Puppet follows the client-server model, where one machine in any cluster acts as the server, known as puppet master and the other acts as a client known as a slave on nodes. Puppet has the capability to manage any system from scratch, starting from initial configuration till the end-of-life of any particular machine.

Features of puppet

Following are the most important features of Puppet:

1. *Idempotency* : Puppet supports Idempotency which makes it unique. Idempotency helps in managing any particular machine throughout its lifecycle starting from the creation of machine, configurational changes in the machine, till the end-of-life. Puppet Idempotency feature is very helpful in keeping the machine updated for years rather than rebuilding the same machine multiple times, when there is any configurational change.
2. *Cross-platform:* In Puppet, with the help of Resource Abstraction Layer (RAL) which uses Puppet resources, one can target the specified configuration of system without worrying about the implementation details and how the configuration command will work inside the system, which are defined in the underlying configuration file.

Key Components of puppet

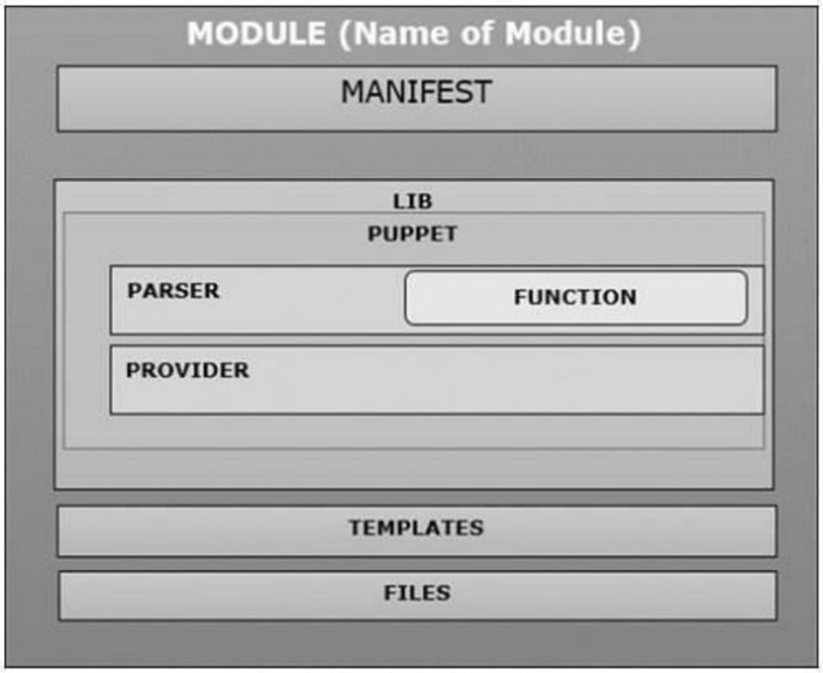


Fig 1. Components of puppet

Puppet Resources

Puppet resources are the key components for modelling any particular machine. These resources have their own implementation model. Puppet uses the same model to get any particular resource in the desired state.

Providers

Providers are basically fulfillers of any particular resource used in Puppet. For example, the package type 'apt-get' and 'yum' both are valid for package management. Sometimes, more than one provider would be available on a particular platform. Though each platform always have a default provider.

Manifest

Manifest is a collection of resources which are coupled inside the function or classes to configure any target system. They contain a set of Ruby code in order to configure a system.

Modules

Module is the key building block of Puppet, which can be defined as a collection of resources, files, templates, etc. They can be easily distributed among different kinds of OS being defined that they are of the same flavour. As they can be easily distributed, one module can be used multiple times with the same configuration.

Templates

Templates use Ruby expressions to define the customized content and variable input.

Static Files

Static files can be defined as a general file which are sometimes required to perform specific tasks. They can be simply copied from one location to another using Puppet. All static files are located inside the files directory of any module. Any manipulation of the file in a manifest is done using the file resource.

Architecture of puppet

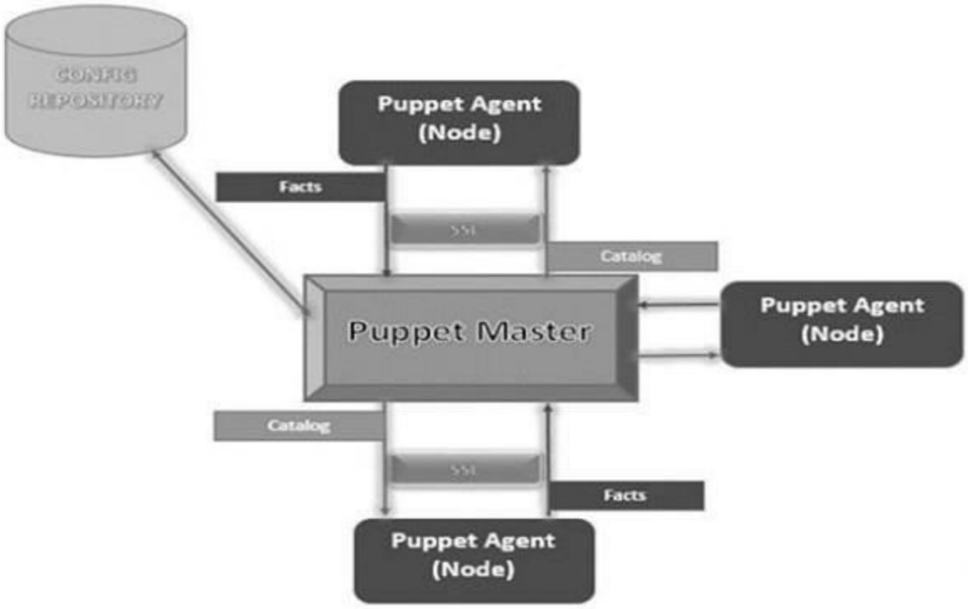


Fig 2. Architecture of puppet

Puppet Master
Puppet Master is the key mechanism which handles all the configuration related stuff. It applies the configuration to nodes using the Puppet agent.

Puppet Agent
Puppet Agents are the actual working machines which are managed by the Puppet master. They have the Puppet agent daemon service running inside them.

Config Repository
This is the repo where all nodes and server-related configurations are saved and pulled when required.

Facts
Facts are the details related to the node or the master machine, which are basically used for analysing the current status of any node. On the basis of facts, changes are done on any target machine. There are pre-defined and custom facts in Puppet.

Catalog
All the manifest files or configuration which are written in Puppet are first converted to a compiled format called catalog and later those catalogs are applied on the target machine.

Procedure
Install puppet agent and puppet master on two separate virtual machines and establish connection between them.
Download Oracle virtualbox Debian Package from its official site and Install Oracle VirtualBox Manager using following command :
ubuntu@ubuntu$ dpkg -i virtualbox.deb

Now your task is to install and configure puppet-master and puppet-agent. Let's proceed with the installation of puppet mater virtual vm on oracle virtualbox.

Step 1: Click on the new option available on the screen. Step 2: Selecting Ubuntu 64 bit as operating system Step 3: Allocating memory of 2GB to the virtual os Step 4: select option virtual hard disk
Step 5: Selecting virtual disk image
Step 6: Allocate file and specify the maximum size vm can take. Step 7 : Allocate file and specify the maximum size vm can take. Step 8: Verify the general settings in the general setting tab.
Step 9: Now go to storage option and click on controller
Step 9: Click on empty option and select disk option at the top right
Step 10: Select ubuntu-18.04 iso file
Step 11: Select the network tab and enable network adapter
Change attached to "Nat Network" from "Nat"
Step 12: Change network options as shown in below image
Step 13: In order to create new nat network click on file -> preferences Step 14: Click on network tab and enter new nat network details
Step 15: Now again go to puppet server and set network

configuration Step 16: Selecting NAT network and choosing
created network.
Step 17: Start the created virtual machine i.e. puppet
Step 18: Create another virtual machine as puppet-agent
Step 19: Allocate 2gb of memory
Step 20: Now click on settings to configure the puppet-agent
Select newly created vm and hit enter key from your keyboard to start puppet-agent.

Conclusion

In this experiment we have learn about puppet which is a configuration management tool. It follows the client-server model, where one machine in any cluster acts as the server, known as puppet master and the other acts as a client known as a puppet agents. We have successfully installed puppet master and puppet agent and have established connection between them.

EXPERIMENT – 10

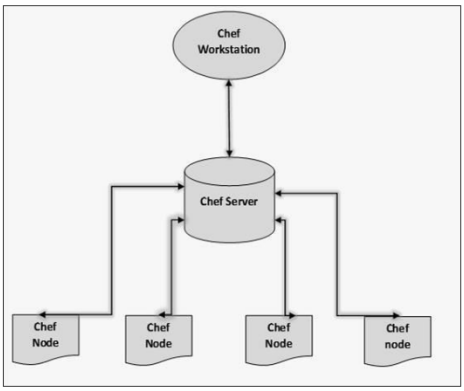
AIM - To Configure and use Chef for continuous integration purpose.

LAB OUTCOME – [LO5 LO6] – Summarise the importance of software configuration management in DevOps. Synthesize provisioning using Anisible/Puppet/Saltstack.

THEORY -

Chef is an open source technology developed by Opscode. Adam Jacob, co-founder of Opscode is known as the founder of Chef. This technology uses Ruby encoding to develop basic building blocks like recipe and cookbooks. Chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

Chef have got its own convention for different building blocks, which are required to manage and automate infrastructure.



Chef works on a three-tier client server model wherein the working units such as cookbooks are developed on the Chef workstation. From the command line utilities such as knife, they are uploaded to the Chef server and all the nodes which are present in the architecture are registered with the Chef server.

Chef Server

This works as a centralized working unit of Chef setup, where all the configuration files are uploaded post development.

Chef Nodes

They are the actual machines which are going to be managed by the Chef server. All the nodes can have different kinds of setup as per requirement.

PROCEDURE -

Using Version Control system is a fundamental part of infrastructure automation. There are multiple kinds of version control system such as SVN, CVS, and GIT. Due to the popularity of GIT among the Chef community, we will use the GIT setup.Note: Don't think of building an infrastructure as a code without a version control system.

On Windows

Step 1: Download the Windows installer from www.git-scm.org and follow the installation steps.

Step 2: Sign up for a central repository on GitHub.

Step 3: Upload the ssh key to the GitHub account, so that one can interact with it easily. For details on ssh key visit the following link https://help.github.com/articles/generating-ssh- keys.
Step 4: Finally create a repo on the github account by visiting https://github.com/new with the name of chef-repo.

Before actually starting to write a cookbook, one can set up an initial GIT repository on the development box and clone the empty repository provided by Opscode.
Step 1: Download Opscode Chef repository empty structure.

Step 2: Extract the tar ball.
Step 3: Rename the directory.
Step 4: Change the current working directory to chef repo.
Step 5: Initialize a fresh get repo.
Step 6: Connect to your repo on the git hub.
Step 7: Push the local repo to github.

By using the above procedure, you will get an empty chef repo in place. You can then start working on developing the recipes and cookbooks. Once done, you can push the changes to the GitHub.

In order to set up on the Linux machine, we need to first get curl on the machine
Step 1: Once curl is installed on the machine, we need to install Chef on the workstation using Opscode's omnibus Chef installer.
Step 2: Install Ruby on the machine.
Step 3: Add Ruby to path variable.

The Omnibus Chef will install Ruby and all the required RubY gems into

/opt/chef/embedded by adding /opt/chef/embedded/bin directory to the .bash_profile file.

If Ruby is already installed, then install the Chef Ruby gem on the machine by running the following command.

Step 1: Download Opscode Chef repository empty structure.

```
$ wget https://github.com/opscode/chef-repo/tarball/master
```

Step 2: Extract the tar ball.

```
$ tar –xvf master
```

Step 3: Rename the directory.

```
$ mv opscode-chef-repo-2c42c6a/ chef-repo
```

Step 4: Change the current working directory to chef repo.

```
$ cd chef-repo
```

Step 5: Initialize a fresh get repo.

```
$ git init .
```

Step 6: Connect to your repo on the git hub.

```
$ git remote add origin git@github.com:vipin022/chef-
```

Step 7: Push the local repo to github.

```
$ git add .
$ git commit –m “empty repo structure added”
$ git push –u origin master
```

CONCLUSION – Successfully installed chef.