BOHARA LOKESH RAMDEV
501812
IT SEM8

Experiment 1

Aim: Introduction to DevOps

Lab Outcome (LO1): Remember the importance of DevOps tools used in software Development life cycle

Theory:

What is DevOps?
DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

How does DevOps work?
The DevOps process flow is all about agility and automation. Each phase in the DevOps lifecycle focuses on closing the loop between development and operations and driving production through continuous development, integration, testing, monitoring and feedback, delivery, and deployment.

What is the difference between Development and Operations?
The Development team works on code which is then sent to the testing team for validation against requirements. Operation team comes in toward the end of the process, where handover of release is given.

What are the stages in a DevOps Lifecycle?
It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring.

- ***Continuous Development –***

  This is the phase that involves ‘planning‘ and ‘coding‘ of the software. The vision of the project is decided during the planning phase and the developers begin developing the code for the application

- ***Continuous Testing –***

  This is the stage where the developed software is continuously tested for bugs. For Continuous testing, automation testing tools like Selenium, TestNG, JUnit, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality. In this phase, Docker Containers can be used for simulating the test environment

- ***Continuous Integration –***
  This stage is the heart of the entire DevOps life cycle. It is a software development practice in which the developers are required to commit changes to the source code more frequently. This may be on a daily or a weekly basis. Every commit is then built and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging

- ***Continuous Deployment –***
  This is the stage where the code is deployed to the production servers. It is also important to ensure that the code is correctly deployed on all the servers

- ***Continuous Monitoring –***
  This is a very crucial stage of the DevOps lifecycle where you continuously monitor the performance of your application. Here vital information about the use of the software is recorded. This information is processed to recognize the proper functionality of the application. The system errors such as low memory, server not reachable, etc are
  resolved in this phase

What is the Software Development Lifecycle?
SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use. Its stages are as follows:

- **Identify the Current Problems**
  This stage of the SDLC means getting input from all stakeholders, including customers, salespeople, industry experts, and programmers. Learn the strengths and weaknesses of the current system with improvement as the goal

- **Plan**
  In this stage of the SDLC, the team determines the cost and resources required for implementing the analyzed requirements. It also details the risks involved and provides sub-plans for softening those risks

- **Design**
  This phase of the SDLC starts by turning the software specifications into a design plan called the Design Specification. All stakeholders then review this plan and offer feedback and suggestions. It’s crucial to have a plan for collecting and incorporating stakeholder input into this document. Failure at this stage will almost certainly result in cost overruns at best and the total collapse of the project at worst

- **Build**
  At this stage, the actual development starts. It’s important that every developer sticks to the agreed blueprint. Also, make sure you have proper guidelines in place about the code style and practices

- **Test**
  In this stage, we test for defects and deficiencies. We fix those issues until the product meets the original specifications. In short, we want to verify if the code meets the defined requirements

- **Software Deployment**
  At this stage, the goal is to deploy the software to the production environment so users can start using the product. However, many organizations choose to move the product through different deployment environments such as a testing or staging environment

What are the tools used in DevOps?
DevOps tools help simplify and accelerate testing, configuration, deployment, and other software-related tasks required to implement DevOps processes. A few tools are as follows:

- **Git:** It is a widely used DevOps tool across the software industry. It’s a distributed SCM (source code management) tool known for its free open source collaboration and planning that is extensively used for tracking the progress of development work by remote teams and open source contributors. It supports most of the version control features including check-in, commits, branches, merging, labels, push and pull to/from GitHub and more
- **Jenkins:** It is an open source solution for continuous integration that orchestrates and automates sequence of actions enabling developers to reliably build, test, and deploy their software. Jenkins is used by DevOps teams for accelerating production rollouts by benefiting from its power of automation. With a large pool of plug-ins available in the Jenkins ecosystem, its capabilities can be expanded to various stages in the DevOps lifecycle
- **Puppet:** It is an open source configuration management tool to automate inspecting, delivering and managing the software across the complete development lifecycle with platform independence. It automates infrastructure management to deliver software quickly and securely
- **Chef:** It is another configuration management tool used to automate and simplify deployment, repair, update and management of application infrastructures. Avoiding manual scrip-based changes, Chef provides a seamless orchestration engine to allow DevOps engineers to ensure continuous delivery of code releases. By treating infrastructure as code, Chef uses pre-built and customizable policies to automatically effect changes to the deployment infrastructure
- **eG Enterprise:** Monitoring is a critical part of software development and deployment. Through all the stages of DevOps, from code build to test and commit to deploy DevOps teams need to understand the impact that their code will have on pre-production and production environments. eG Enterprise is a continuous monitoring tool allows tracking application performance in the context of code changes to understand how they impact performance

Conclusion: Therefore, we have observed an overview of DevOps.

Name: Lokesh Ramdev Bohara

Roll No.: 501812

DevOps Experiment 2

Aim: Version Control System using GIT

Lab Outcome: (LO3) Examine different Version Control Strategies

Theory:

Git is an open-source distributed version control system**.** It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

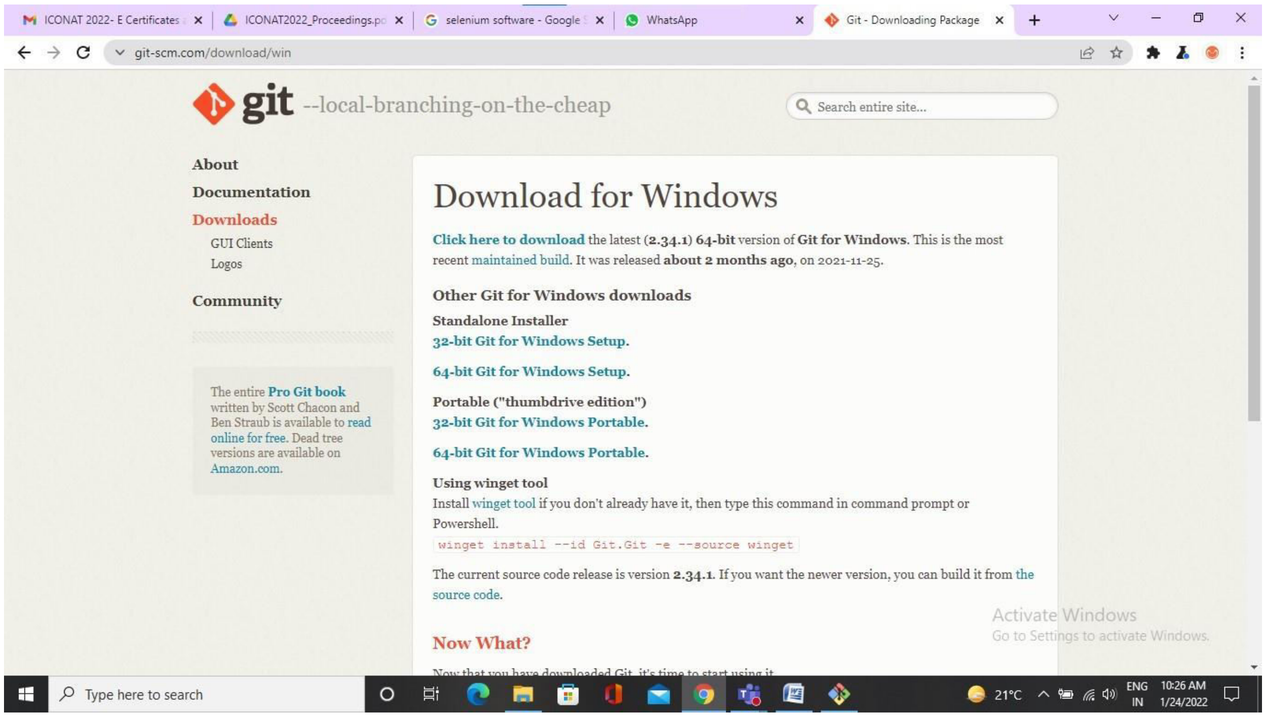
Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

Git was created by Linus Torvalds in 2005 to develop Linux Kernel. It is also used as an important distributed version-control tool for the DevOps**.**

Features of Git:

- Open Source
- Scalable
- Distributed
- Security
- Speed
- Supports non linear development
- Branching and Merging
- Data assurance
- Staging Area
- Maintains a clean history.

Installation of Git Bash for Windows


ICONAT 2022- E Certificates
ICONAT2022_Proceedings.pdf
selenium software - Google
WhatsApp
Git - Downloading Package
git-scm.com/download/win
git --local-branching-on-the-cheap
Search entire site...
About
Documentation
Downloads
GUI Clients
Logos
Community
Activate Windows
Go to Settings to activate Windows.
Type here to search
21°C
ENG IN
10:26 AM
1/24/2022

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Download for Windows

Click here to download the latest (**2.34.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **about 2 months ago**, on 2021-11-25.

Other Git for Windows downloads

Standalone Installer

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

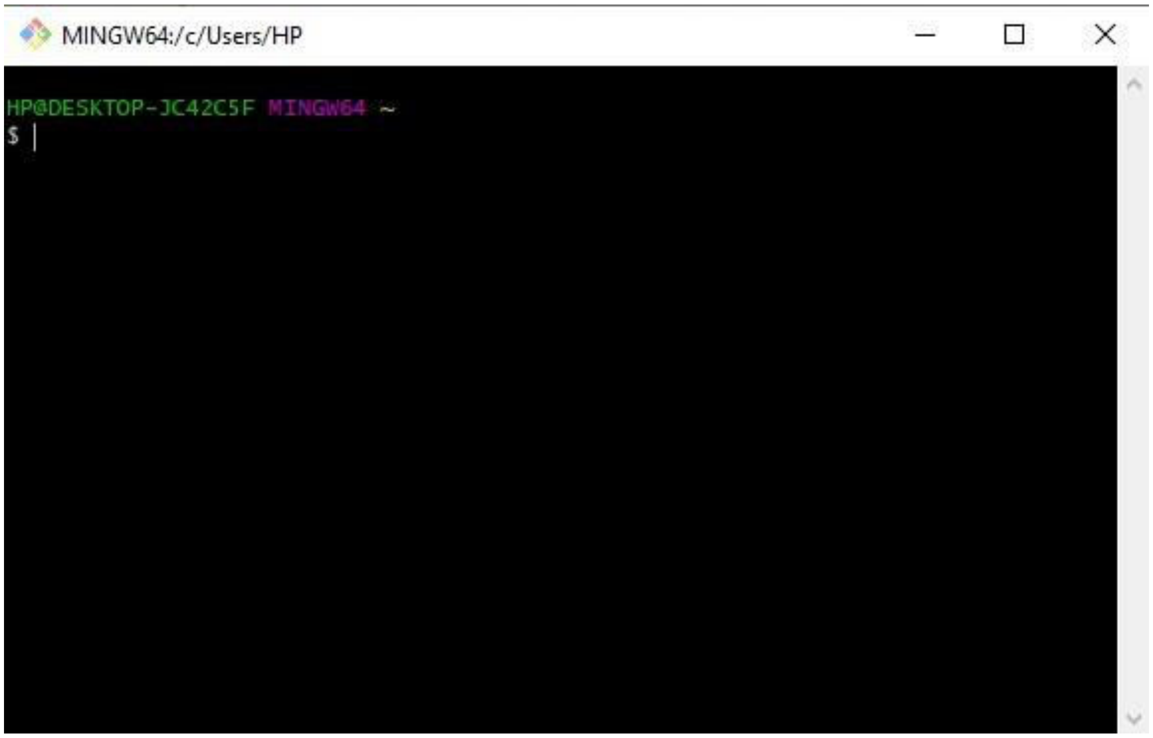
Using winget tool

Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

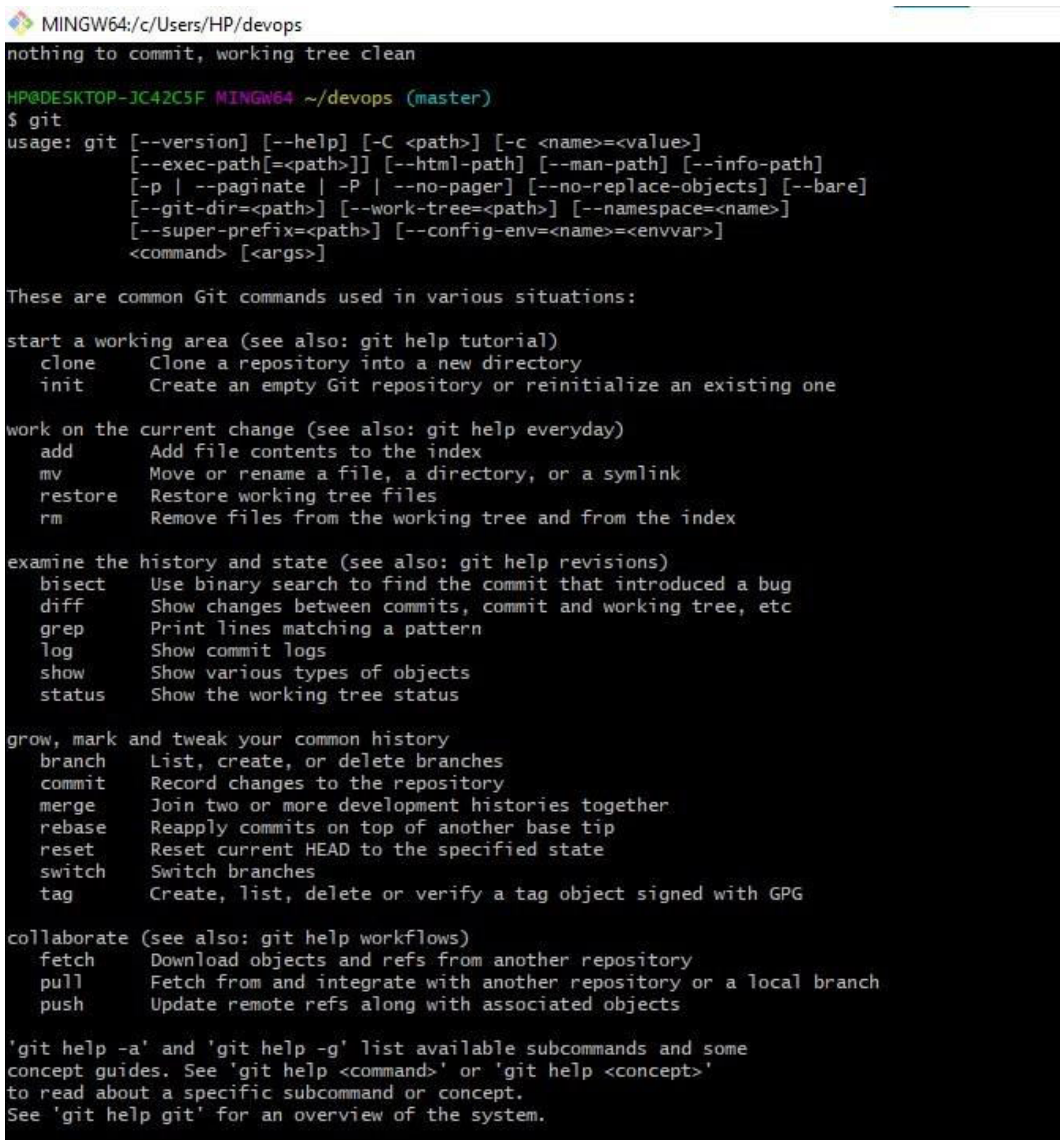
The current source code release is version **2.34.1**. If you want the newer version, you can build it from the source code.

Now What?

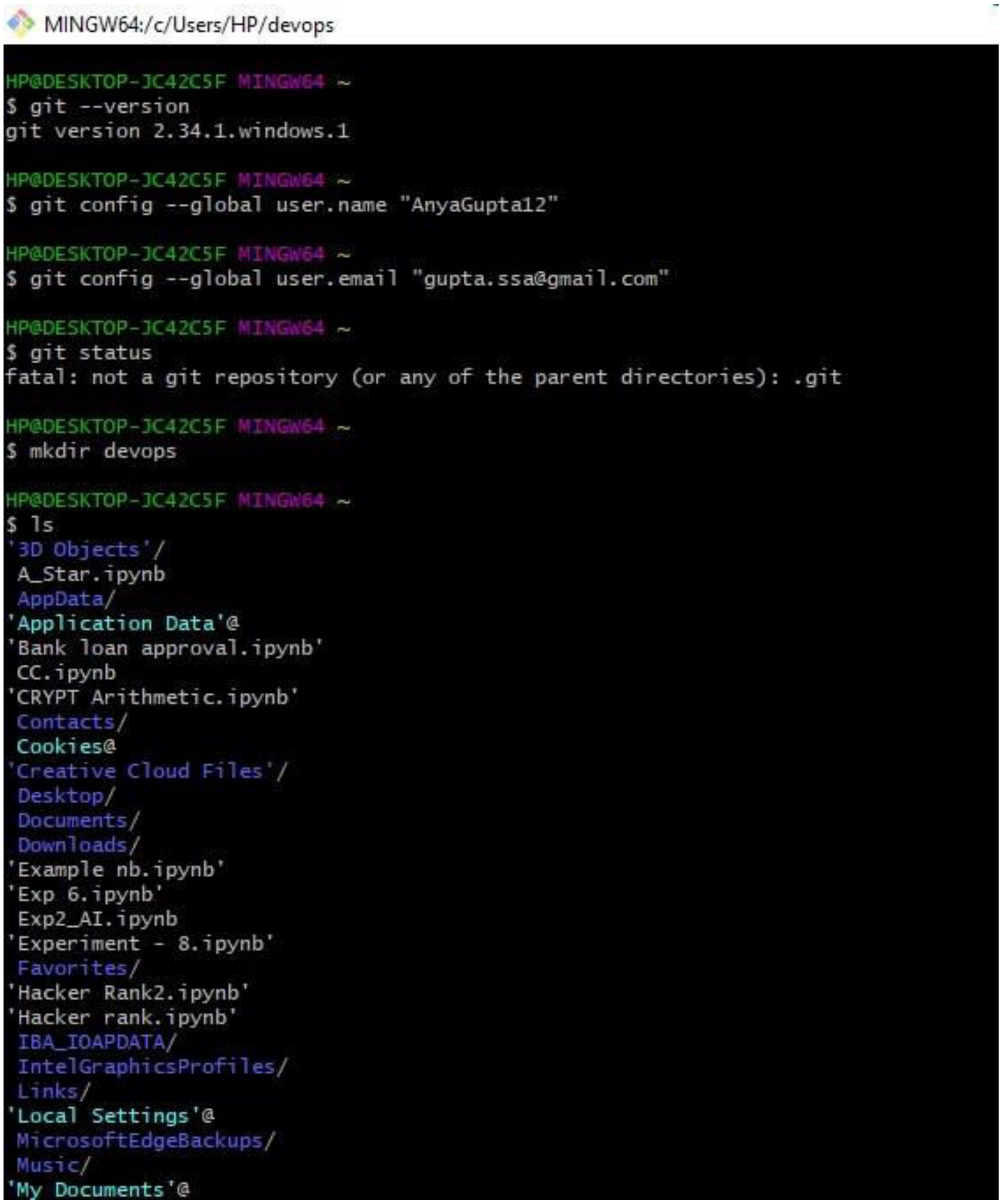


```
HP@DESKTOP-JC42C5F MINGW64 ~
$ |
```

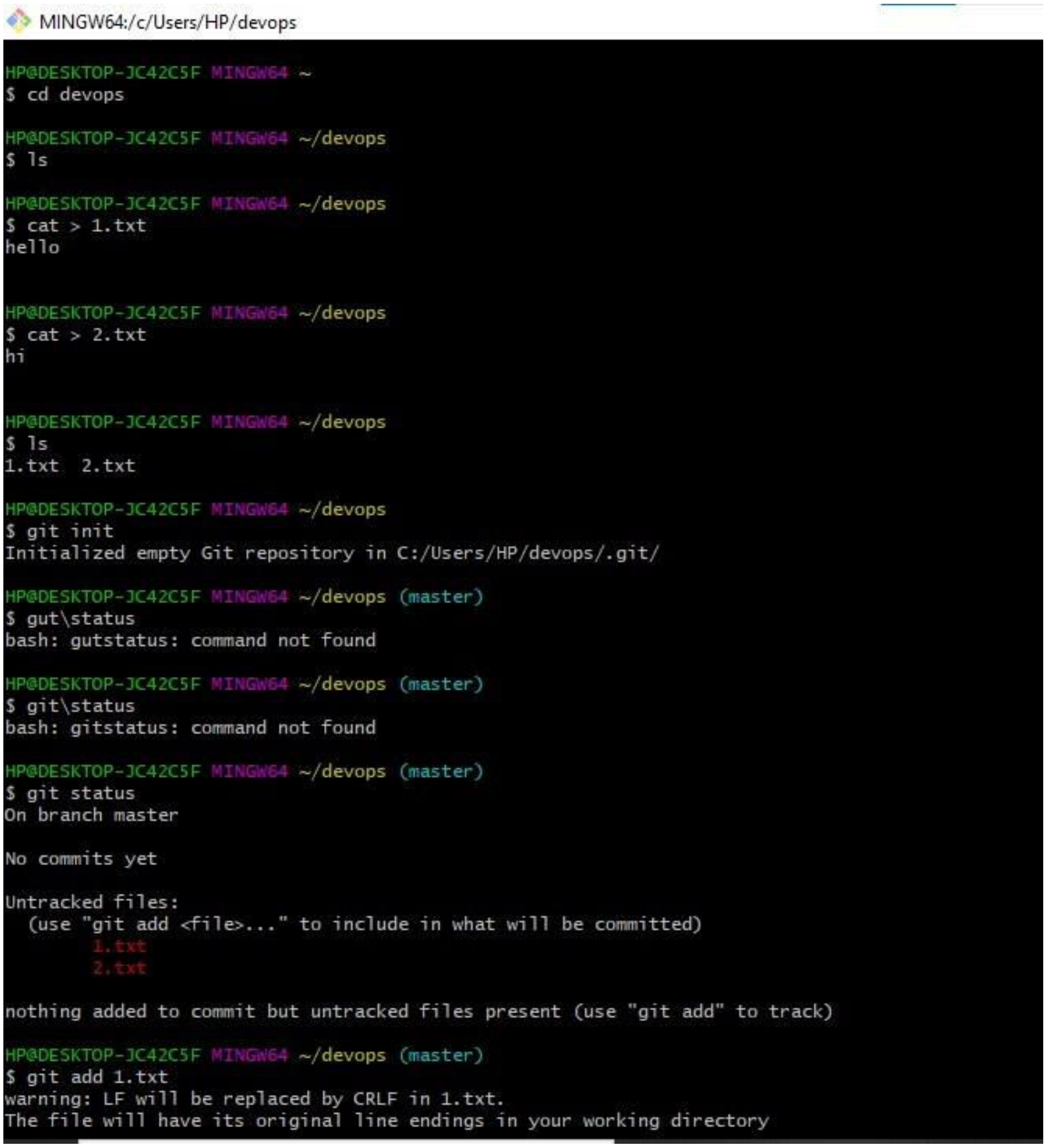
Executing Basic Git commands on GitBash command line


MINGW64:/c/Users/HP/devops

```
nothing to commit, working tree clean

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

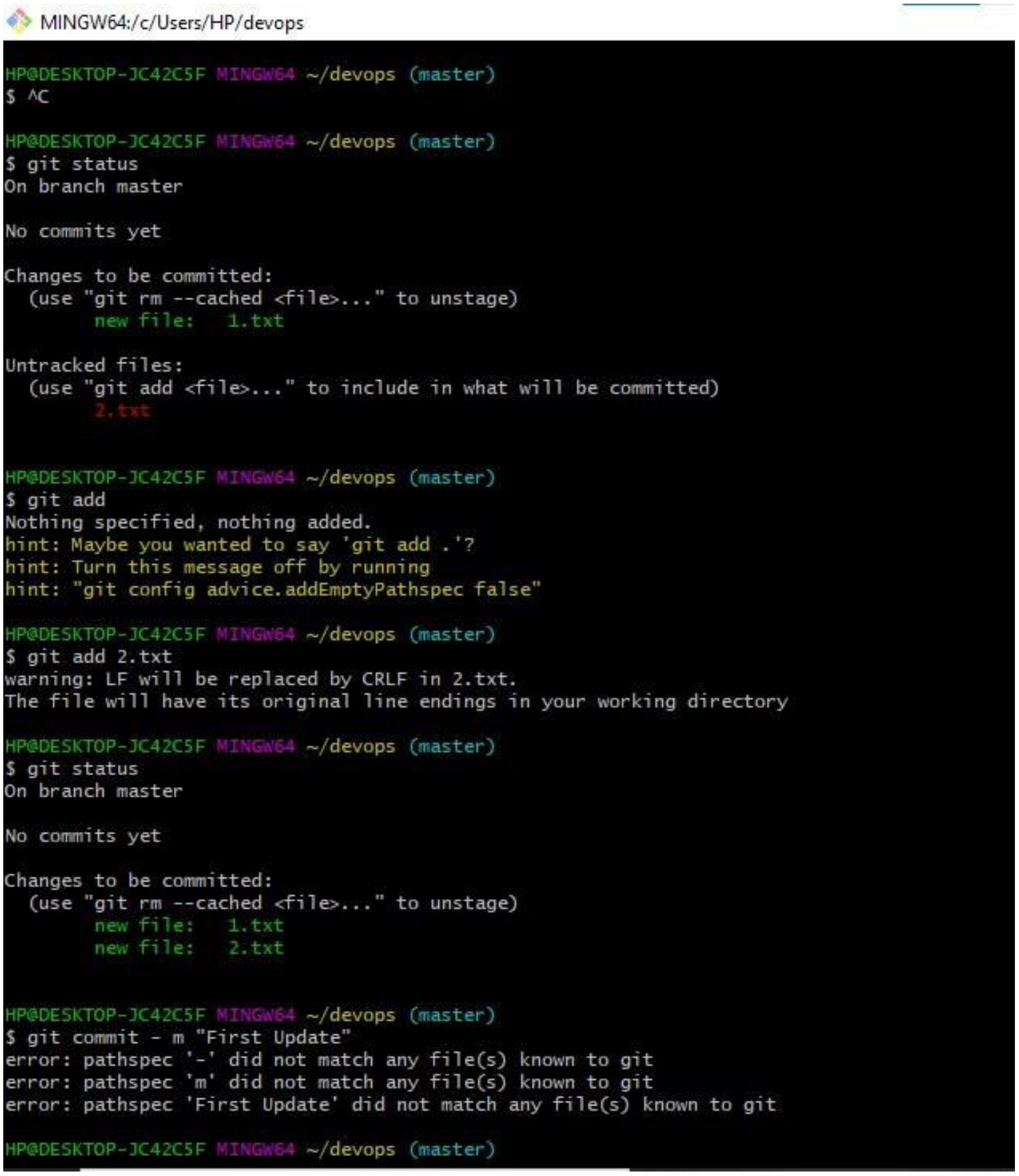

MINGW64:/c/Users/HP/devops

```
HP@DESKTOP-JC42C5F MINGW64 ~
$ git --version
git version 2.34.1.windows.1

HP@DESKTOP-JC42C5F MINGW64 ~
$ git config --global user.name "AnyaGupta12"

HP@DESKTOP-JC42C5F MINGW64 ~
$ git config --global user.email "gupta.ssa@gmail.com"

HP@DESKTOP-JC42C5F MINGW64 ~
$ git status
fatal: not a git repository (or any of the parent directories): .git

HP@DESKTOP-JC42C5F MINGW64 ~
$ mkdir devops

HP@DESKTOP-JC42C5F MINGW64 ~
$ ls
'3D Objects'/
 A_Star.ipynb
 AppData/
'Application Data'@
'Bank loan approval.ipynb'
 CC.ipynb
'CRYPT Arithmetic.ipynb'
 Contacts/
 Cookies@
'Creative Cloud Files'/
 Desktop/
 Documents/
 Downloads/
'Example nb.ipynb'
'Exp 6.ipynb'
 Exp2_AI.ipynb
'Experiment - 8.ipynb'
 Favorites/
'Hacker Rank2.ipynb'
'Hacker rank.ipynb'
 IBA_IOAPDATA/
 IntelGraphicsProfiles/
 Links/
'Local Settings'@
 MicrosoftEdgeBackups/
 Music/
'My Documents'@
```



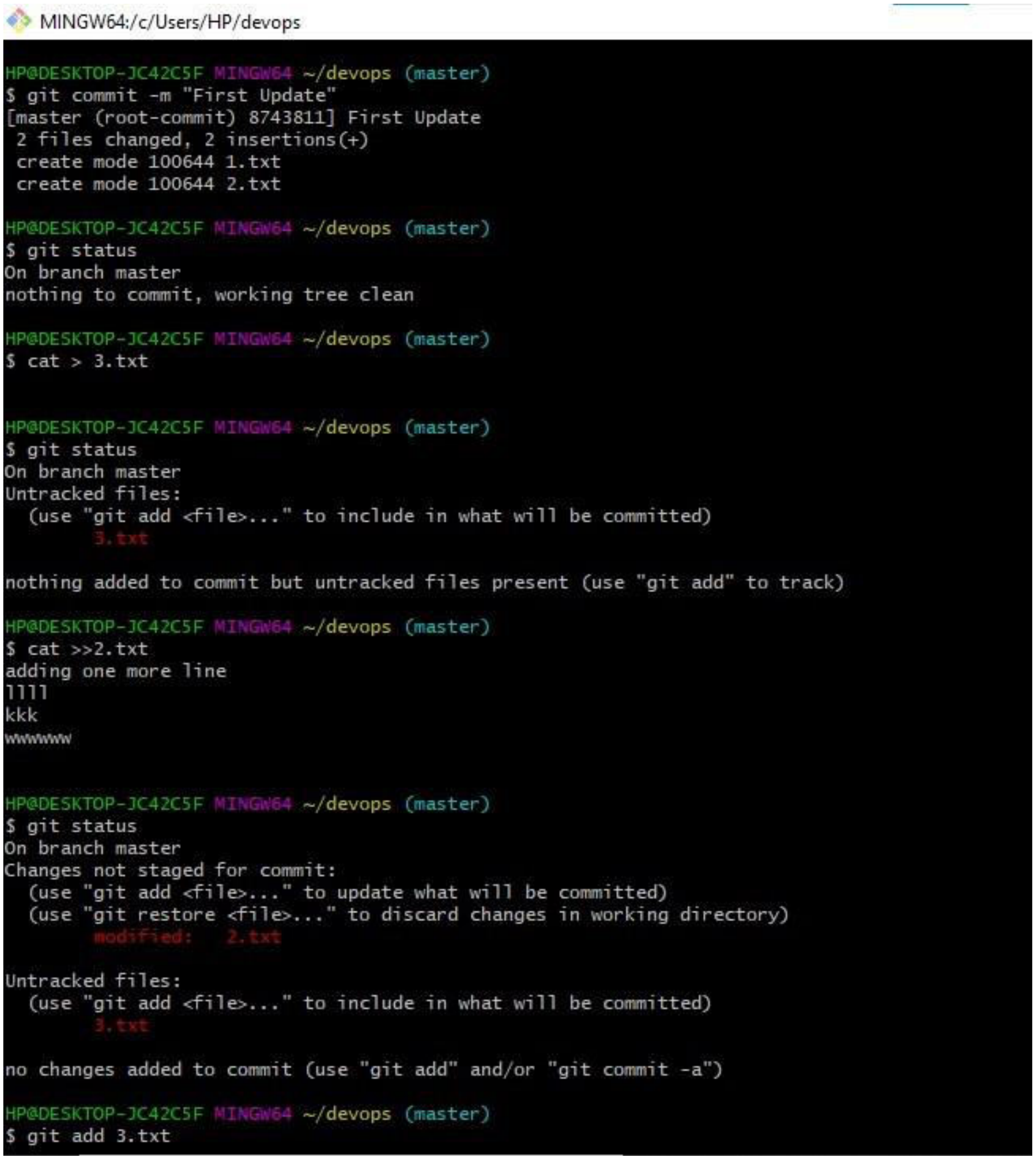
MINGW64:/c/Users/HP/devops

```
HP@DESKTOP-JC42C5F MINGW64 ~
$ cd devops

HP@DESKTOP-JC42C5F MINGW64 ~/devops
$ ls

HP@DESKTOP-JC42C5F MINGW64 ~/devops
$ cat > 1.txt
hello


HP@DESKTOP-JC42C5F MINGW64 ~/devops
$ cat > 2.txt
hi


HP@DESKTOP-JC42C5F MINGW64 ~/devops
$ ls
1.txt  2.txt

HP@DESKTOP-JC42C5F MINGW64 ~/devops
$ git init
Initialized empty Git repository in C:/Users/HP/devops/.git/

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ gut\status
bash: gutstatus: command not found

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git\status
bash: gitstatus: command not found

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        1.txt
        2.txt

nothing added to commit but untracked files present (use "git add" to track)

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add 1.txt
warning: LF will be replaced by CRLF in 1.txt.
The file will have its original line endings in your working directory
```



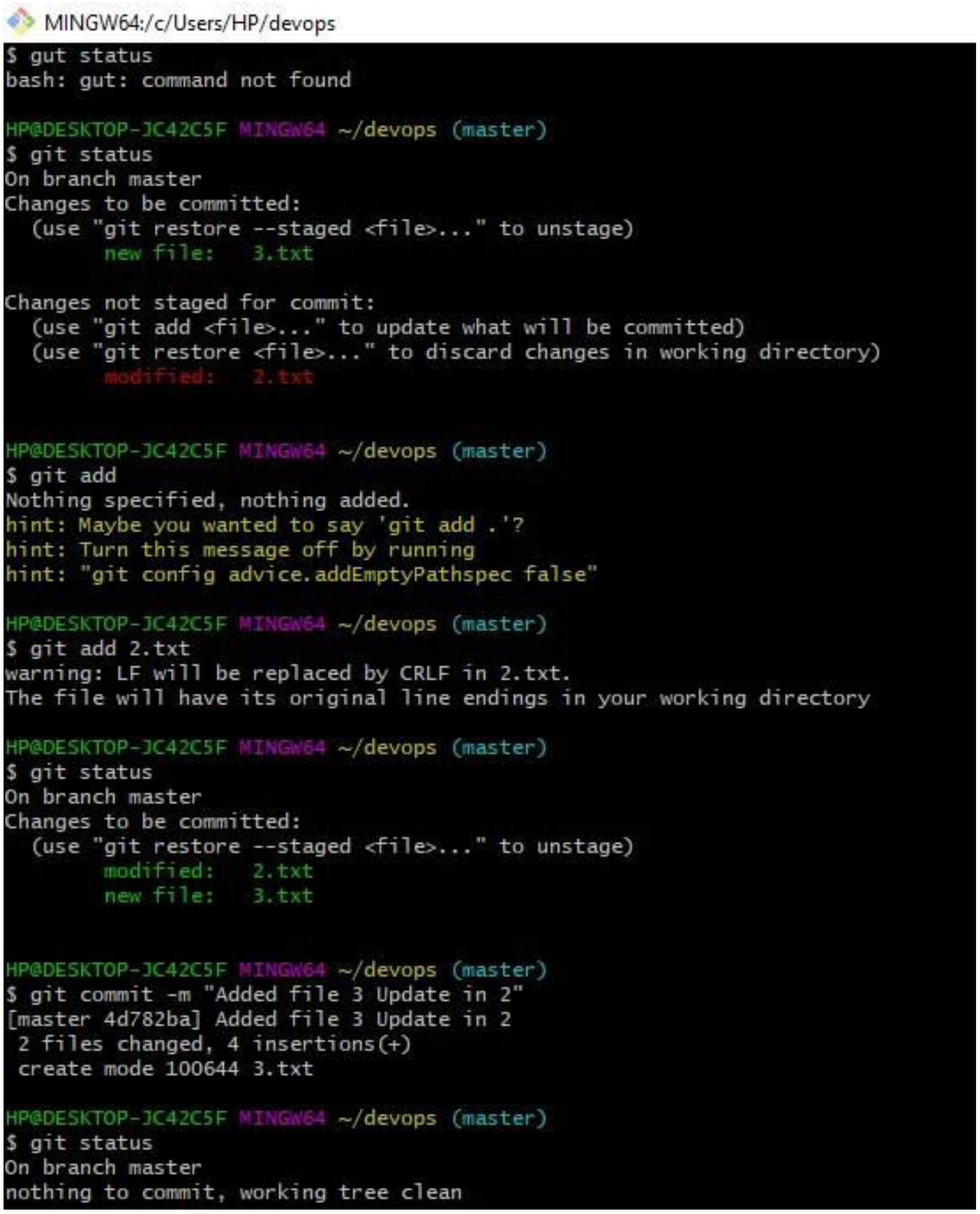
MINGW64:/c/Users/HP/devops

```
HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ ^C

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        2.txt


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add 2.txt
warning: LF will be replaced by CRLF in 2.txt.
The file will have its original line endings in your working directory

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   1.txt
        new file:   2.txt


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git commit - m "First Update"
error: pathspec '-' did not match any file(s) known to git
error: pathspec 'm' did not match any file(s) known to git
error: pathspec 'First Update' did not match any file(s) known to git

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
```



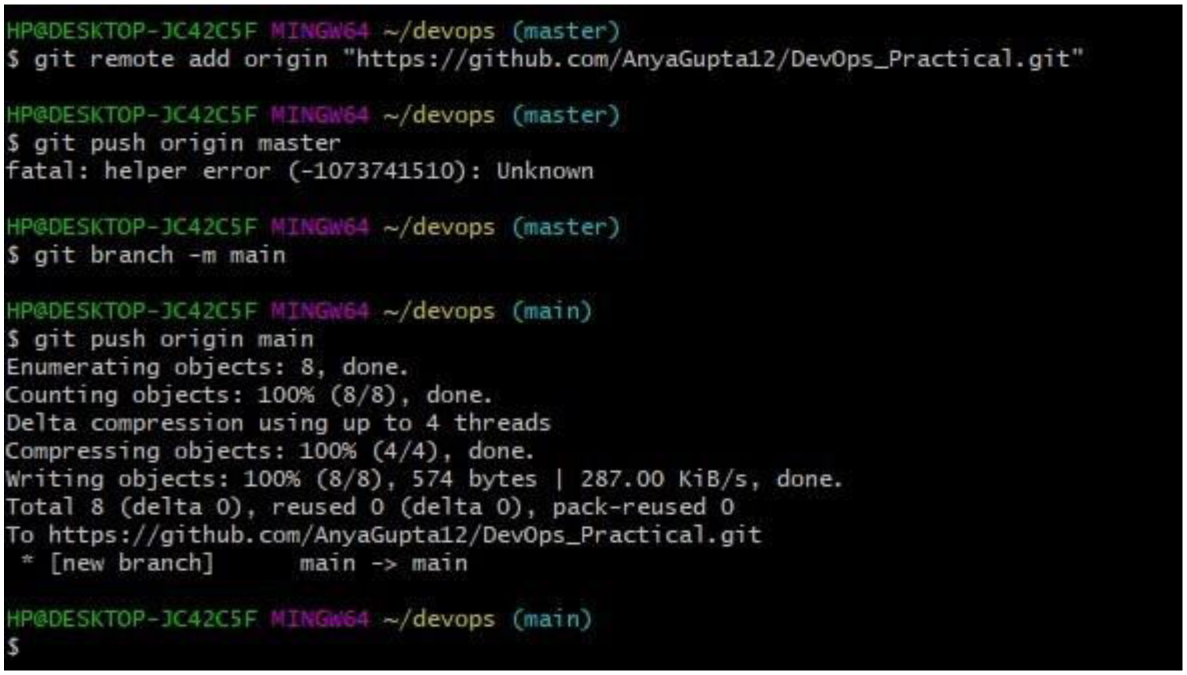
MINGW64:/c/Users/HP/devops

```
HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git commit -m "First Update"
[master (root-commit) 8743811] First Update
 2 files changed, 2 insertions(+)
 create mode 100644 1.txt
 create mode 100644 2.txt

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
nothing to commit, working tree clean

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ cat > 3.txt


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        3.txt

nothing added to commit but untracked files present (use "git add" to track)

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ cat >>2.txt
adding one more line
llll
kkk
wwwwww


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        3.txt

no changes added to commit (use "git add" and/or "git commit -a")

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add 3.txt
```

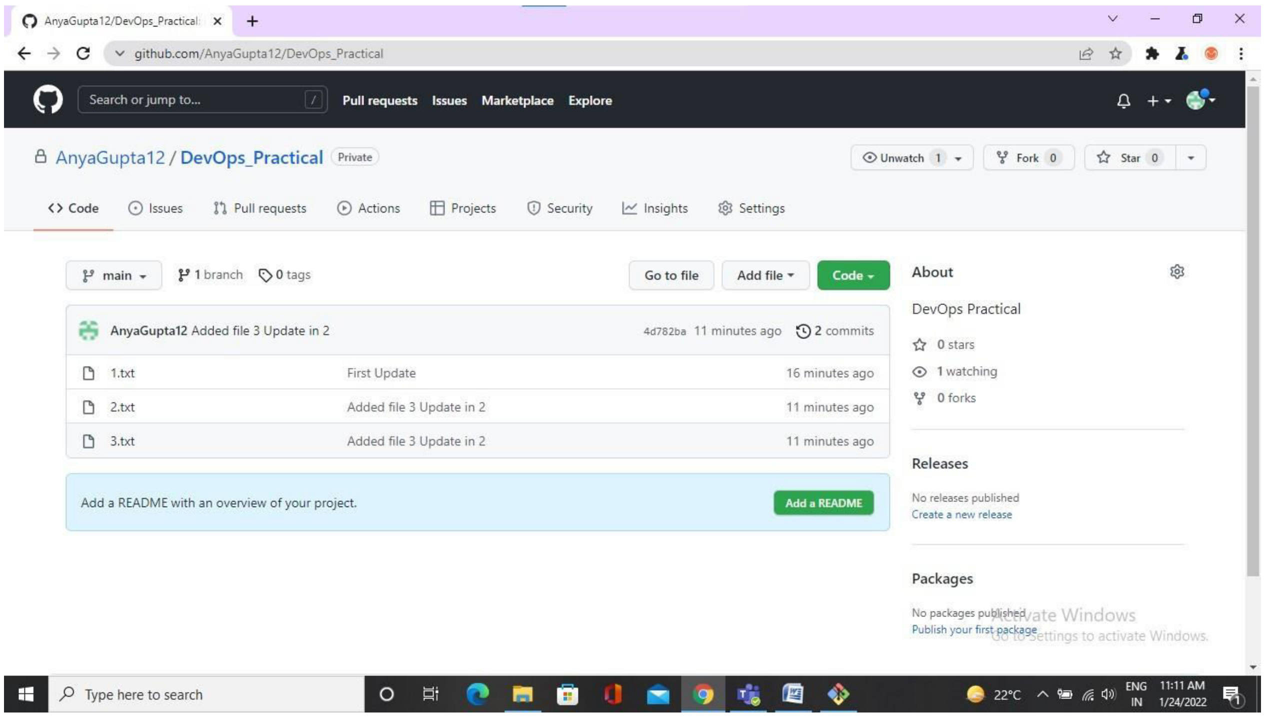


MINGW64:/c/Users/HP/devops

```
$ gut status
bash: gut: command not found

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   3.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git add 2.txt
warning: LF will be replaced by CRLF in 2.txt.
The file will have its original line endings in your working directory

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt
        new file:   3.txt


HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git commit -m "Added file 3 Update in 2"
[master 4d782ba] Added file 3 Update in 2
 2 files changed, 4 insertions(+)
 create mode 100644 3.txt

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git status
On branch master
nothing to commit, working tree clean
```



```
HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git remote add origin "https://github.com/AnyaGupta12/DevOps_Practical.git"

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git push origin master
fatal: helper error (-1073741510): Unknown

HP@DESKTOP-JC42C5F MINGW64 ~/devops (master)
$ git branch -m main

HP@DESKTOP-JC42C5F MINGW64 ~/devops (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 574 bytes | 287.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AnyaGupta12/DevOps_Practical.git
 * [new branch]      main -> main

HP@DESKTOP-JC42C5F MINGW64 ~/devops (main)
$
```



github.com/AnyaGupta12/DevOps_Practical

AnyaGupta12 / DevOps_Practical Private

main · 1 branch · 0 tags

AnyaGupta12 Added file 3 Update in 2 · 4d782ba 11 minutes ago · 2 commits

1.txt	First Update	16 minutes ago
2.txt	Added file 3 Update in 2	11 minutes ago
3.txt	Added file 3 Update in 2	11 minutes ago

Add a README with an overview of your project.

About: DevOps Practical · 0 stars · 1 watching · 0 forks

Releases: No releases published · Create a new release

Packages: No packages published · Publish your first package

Conclusion: Therefore we have successfully studied about Git, Github and how to create a repository and add files to it.

Name: Lokesh Ramdev Bohara

Roll No.: 501812

DevOps Experiment 3

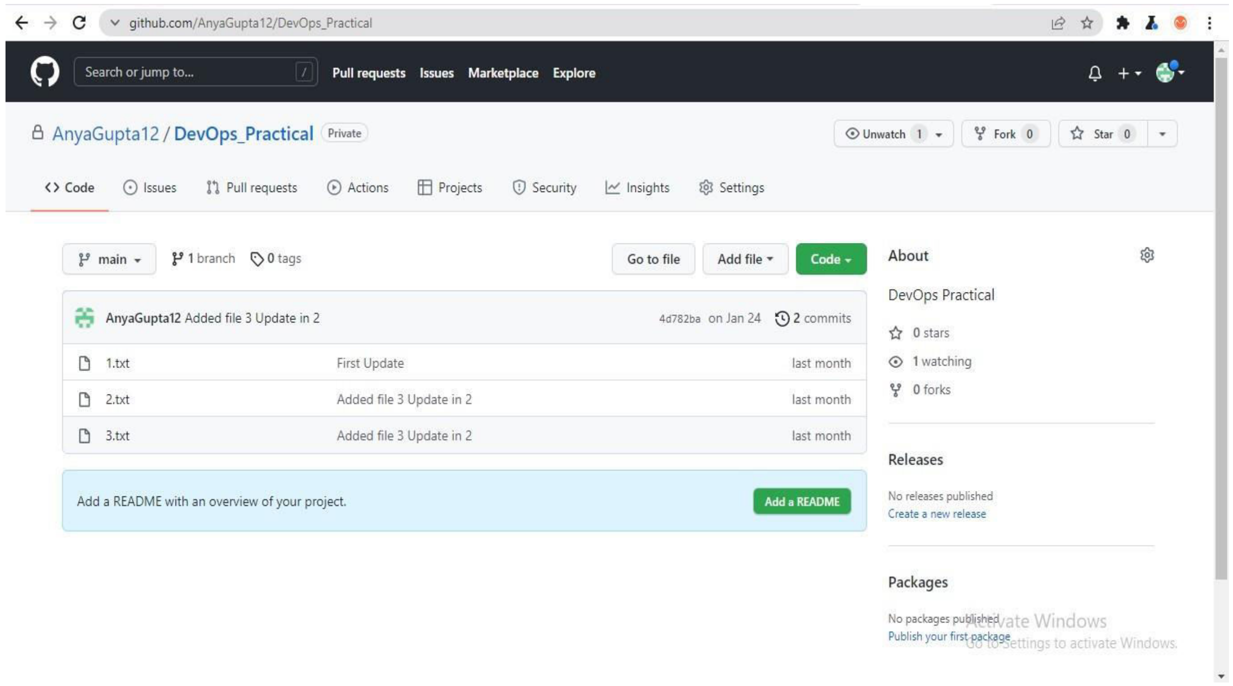
Aim: Cloning and branching Repository using GIT

Theory:

Git is a software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems). As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.Git is free and open-source software distributed under GNU General Public License Version 2.

1. Cloning:

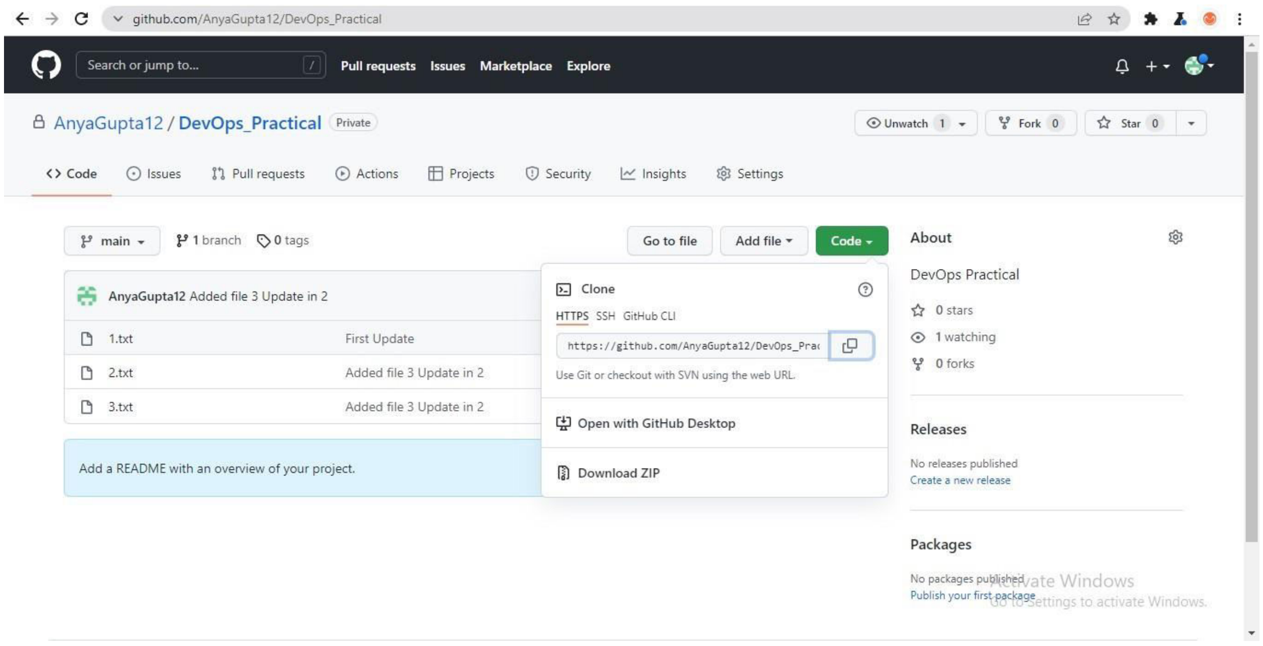
a. Navigate to the repository you want to clone on Github
b. Get the HTTPS url from the 'Code' button in the repo.



AnyaGupta12 Added file 3 Update in 2 4d782ba on Jan 24 2 commits

1.txt	First Update	last month
2.txt	Added file 3 Update in 2	last month
3.txt	Added file 3 Update in 2	last month

Add a README with an overview of your project.

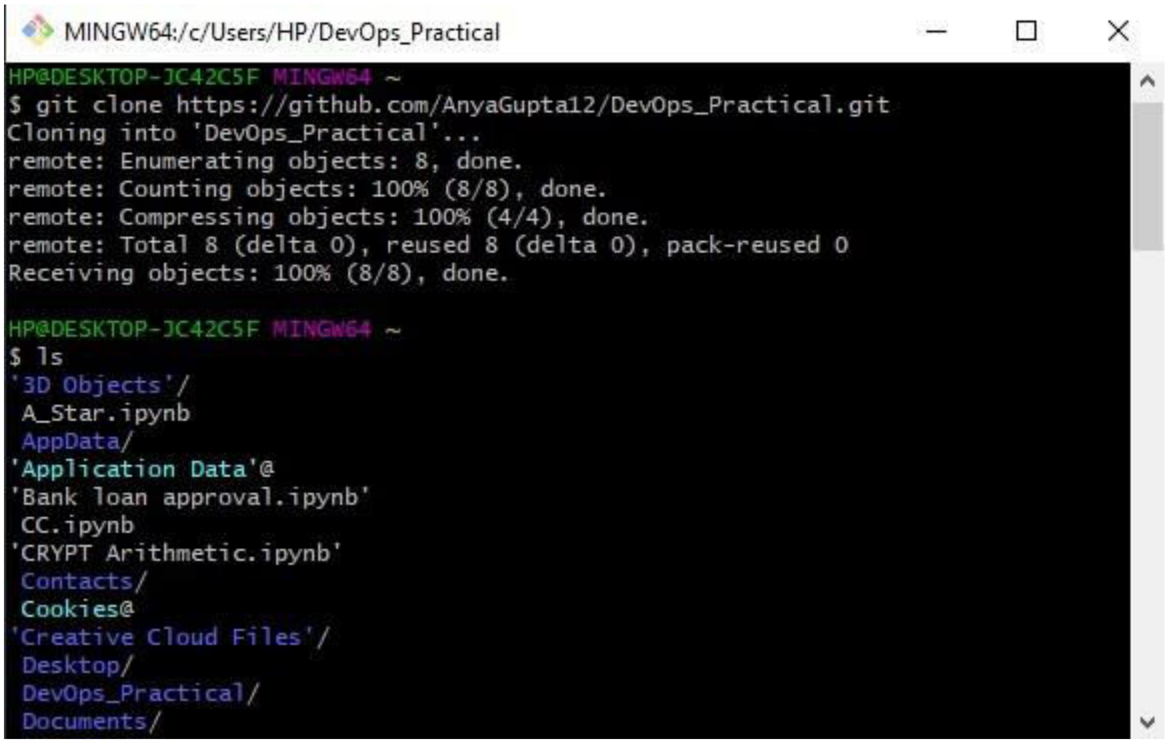


AnyaGupta12 Added file 3 Update in 2	
1.txt	First Update
2.txt	Added file 3 Update in 2
3.txt	Added file 3 Update in 2

Add a README with an overview of your project.

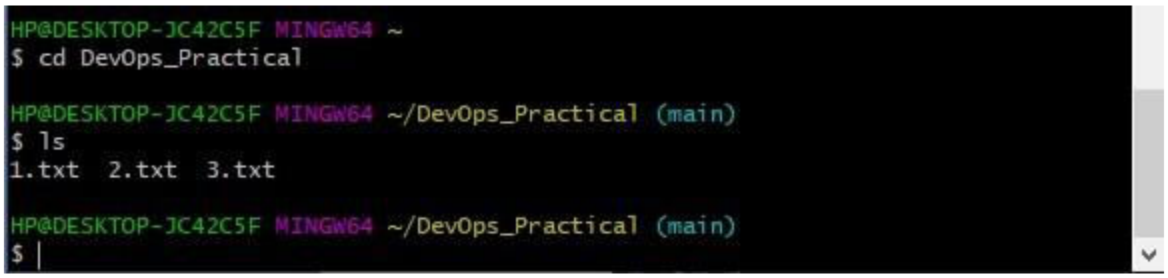
c. Use the following command to download the repository to your local machine. https://github.com/AnyaGupta12/DevOps_Practical.git

d. The files will be saved to a folder named after the repository followed by the branch. In this example, the downloaded files will be stored in “DevOps_Practical”.



MINGW64:/c/Users/HP/DevOps_Practical

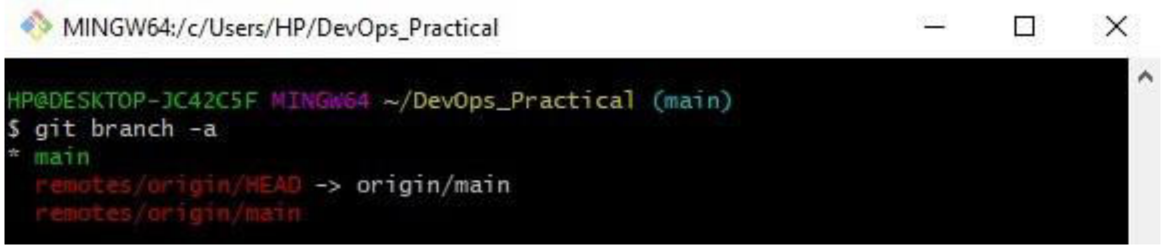
```
HP@DESKTOP-JC42C5F MINGW64 ~
$ git clone https://github.com/AnyaGupta12/DevOps_Practical.git
Cloning into 'DevOps_Practical'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 8 (delta 0), reused 8 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.

HP@DESKTOP-JC42C5F MINGW64 ~
$ ls
'3D Objects'/
 A_Star.ipynb
 AppData/
'Application Data'@
'Bank loan approval.ipynb'
 CC.ipynb
'CRYPT Arithmetic.ipynb'
 Contacts/
 Cookies@
'Creative Cloud Files'/
 Desktop/
 DevOps_Practical/
 Documents/
```



```
HP@DESKTOP-JC42C5F MINGW64 ~
$ cd DevOps_Practical

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ ls
1.txt  2.txt  3.txt

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ |
```

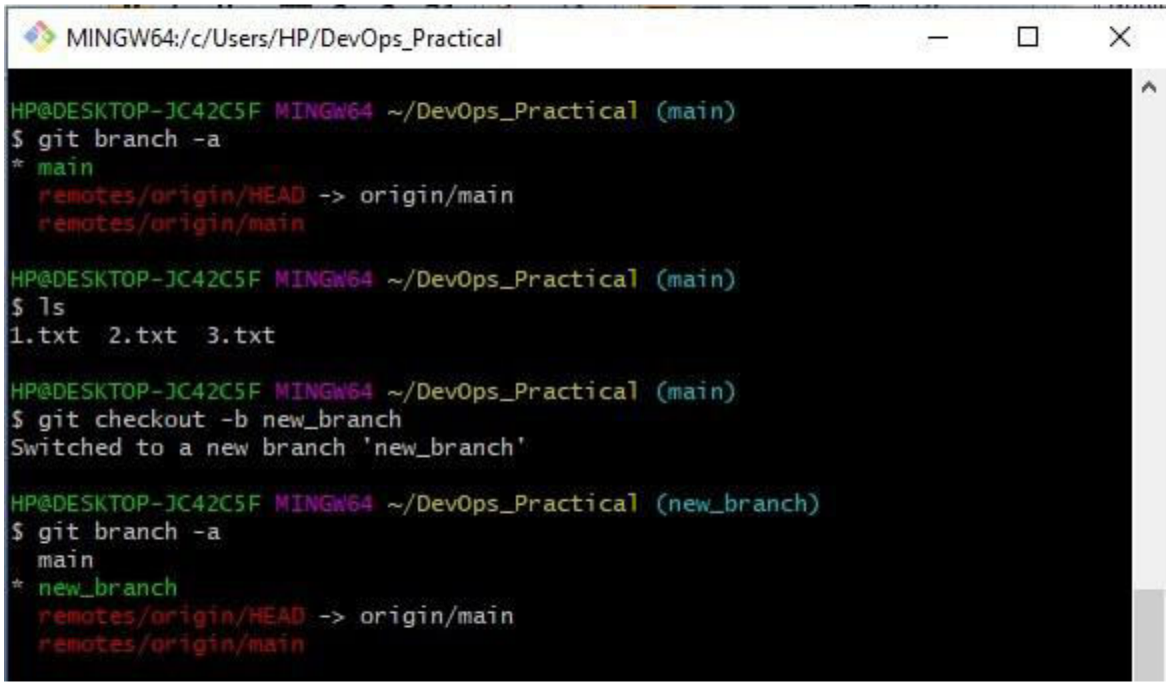
2. Branching:

   a. Listing all branches



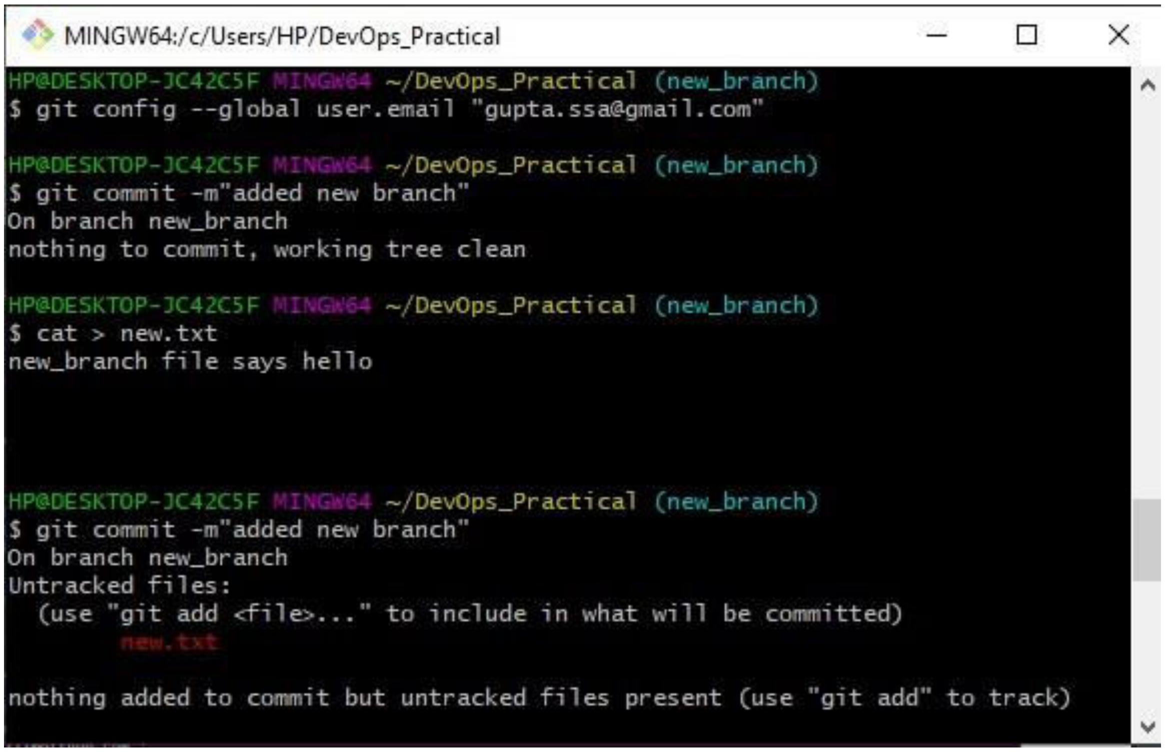
```
HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

   b. Creating a new branch named "new_branch"

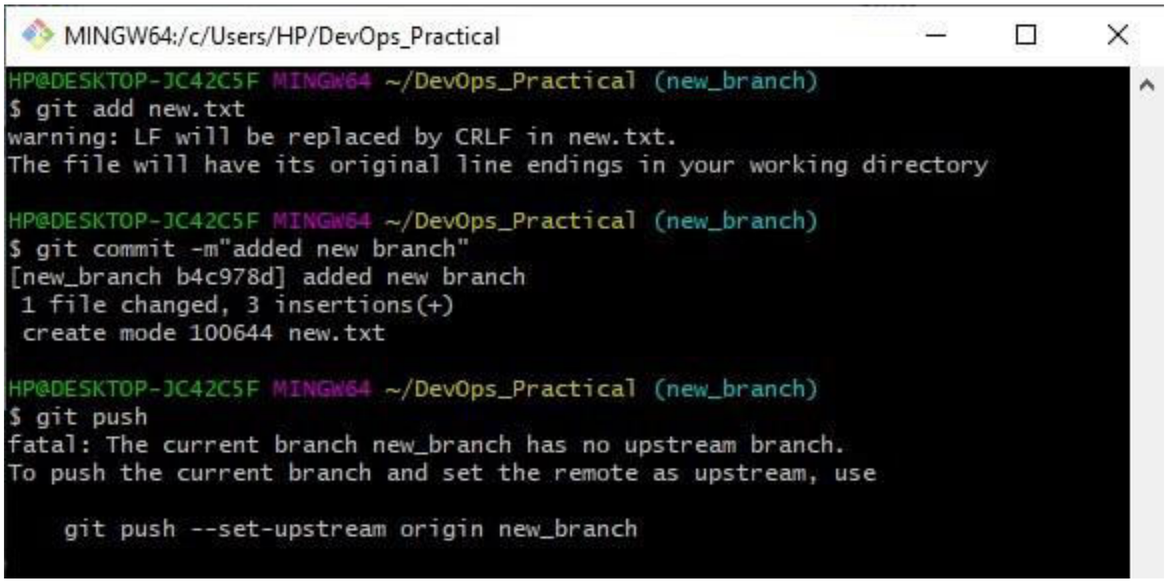


```
HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ ls
1.txt  2.txt  3.txt

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git branch -a
  main
* new_branch
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

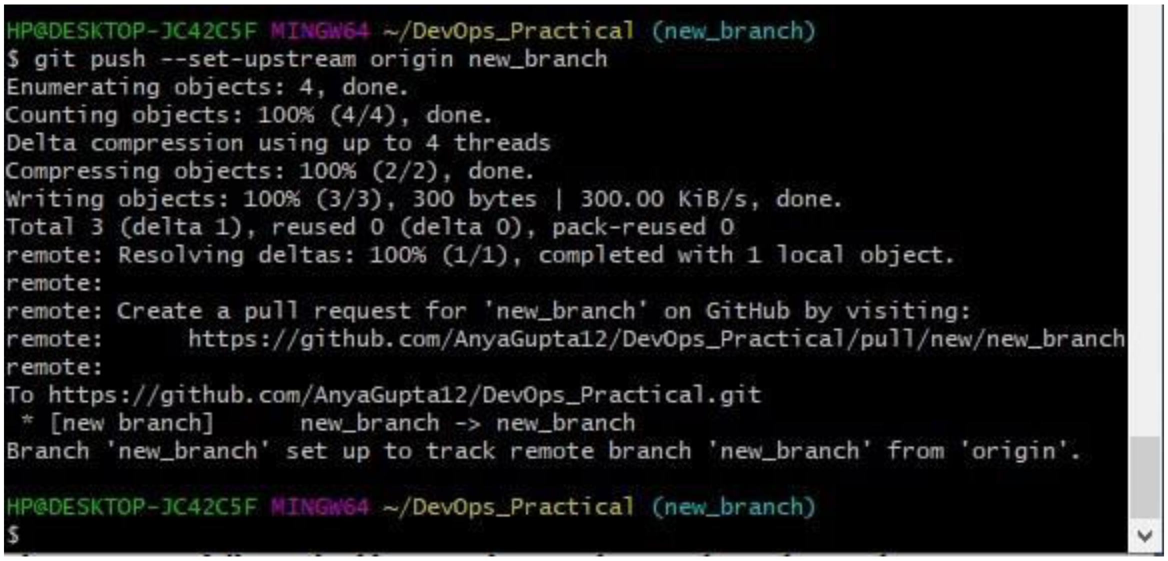
c. Adding files to the new branch



```
MINGW64:/c/Users/HP/DevOps_Practical
HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git config --global user.email "gupta.ssa@gmail.com"

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git commit -m"added new branch"
On branch new_branch
nothing to commit, working tree clean

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ cat > new.txt
new_branch file says hello




HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git commit -m"added new branch"
On branch new_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new.txt

nothing added to commit but untracked files present (use "git add" to track)
```

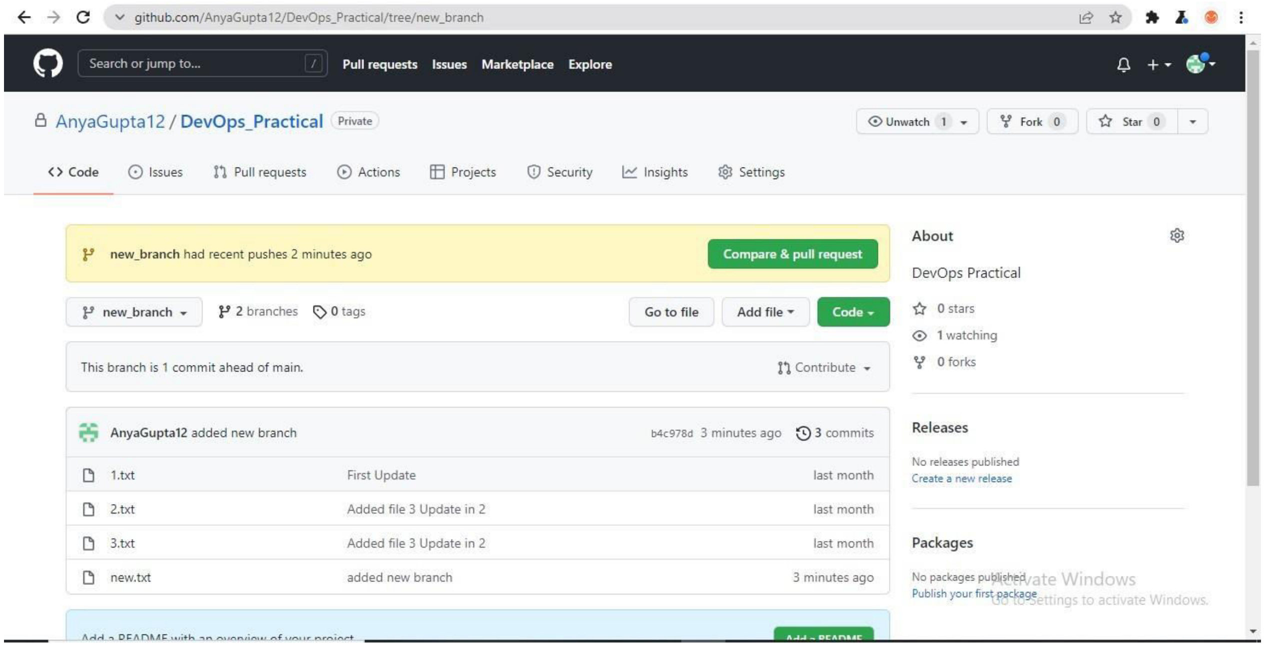


```
MINGW64:/c/Users/HP/DevOps_Practical
HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git add new.txt
warning: LF will be replaced by CRLF in new.txt.
The file will have its original line endings in your working directory

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git commit -m"added new branch"
[new_branch b4c978d] added new branch
 1 file changed, 3 insertions(+)
 create mode 100644 new.txt

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git push
fatal: The current branch new_branch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin new_branch
```



```
HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$ git push --set-upstream origin new_branch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 300.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'new_branch' on GitHub by visiting:
remote:      https://github.com/AnyaGupta12/DevOps_Practical/pull/new/new_branch
remote:
To https://github.com/AnyaGupta12/DevOps_Practical.git
 * [new branch]      new_branch -> new_branch
Branch 'new_branch' set up to track remote branch 'new_branch' from 'origin'.

HP@DESKTOP-JC42C5F MINGW64 ~/DevOps_Practical (new_branch)
$
```

d. Viewing newly added files on Github.



github.com/AnyaGupta12/DevOps_Practical/tree/new_branch

AnyaGupta12 / DevOps_Practical Private

new_branch had recent pushes 2 minutes ago

This branch is 1 commit ahead of main.

AnyaGupta12 added new branch b4c978d 3 minutes ago 3 commits

1.txt	First Update	last month
2.txt	Added file 3 Update in 2	last month
3.txt	Added file 3 Update in 2	last month
new.txt	added new branch	3 minutes ago

About

DevOps Practical

0 stars

1 watching

0 forks

Conclusion : Thus we have successfully studied how to clone and create branches with Git.

Name: Lokesh Ramdev Bohara

Roll No.: 501812

DevOps Experiment 4

Aim: To Install and Configure Docker for creating Containers of different Operating System Images.

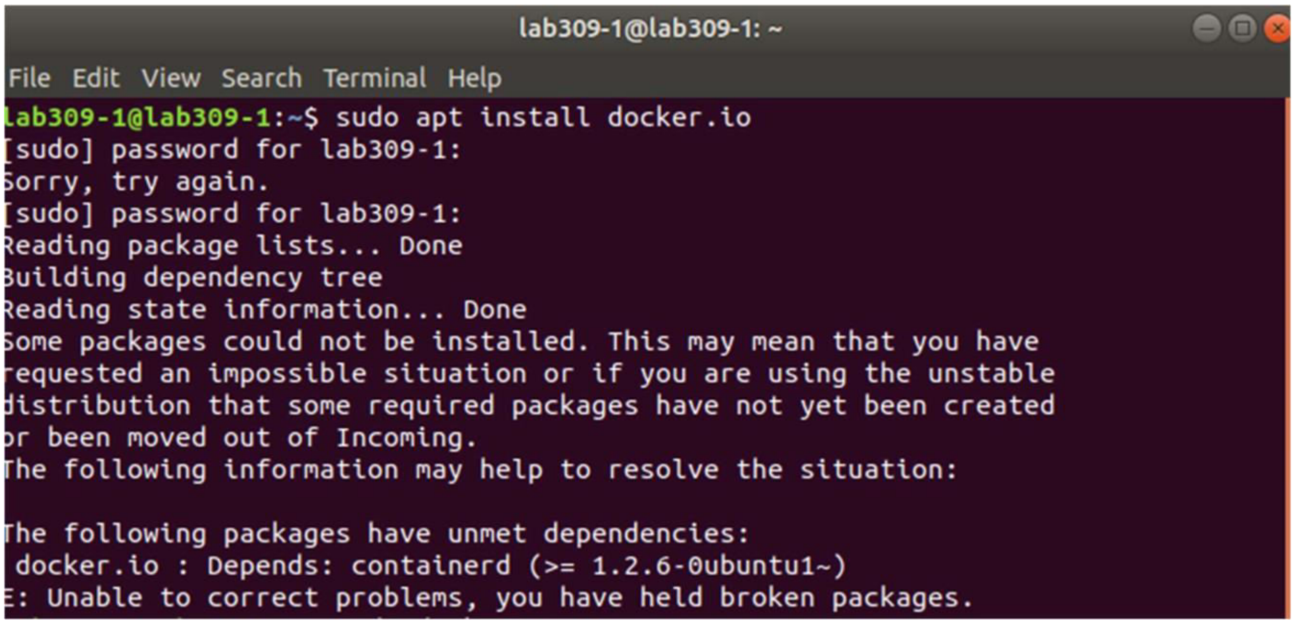
Lab Outcome: – [LO1, LO4] - Remember the importance of DevOps tools used in software development life cycle. Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker.

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.

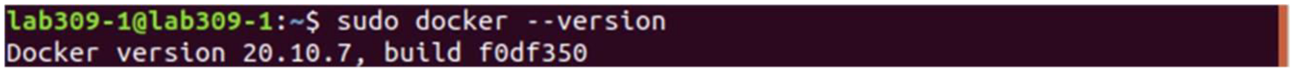
Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides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

Procedure:

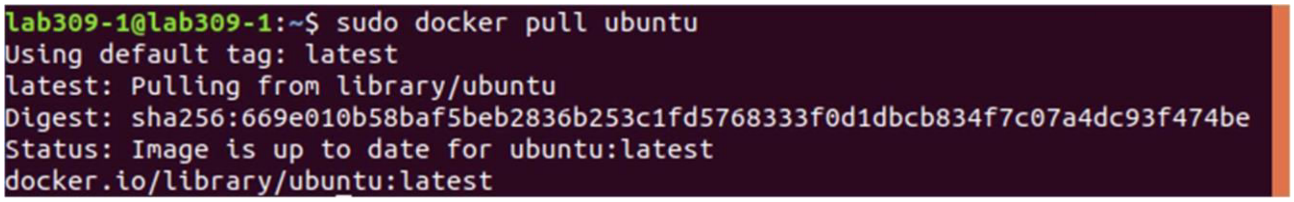
Installing Docker



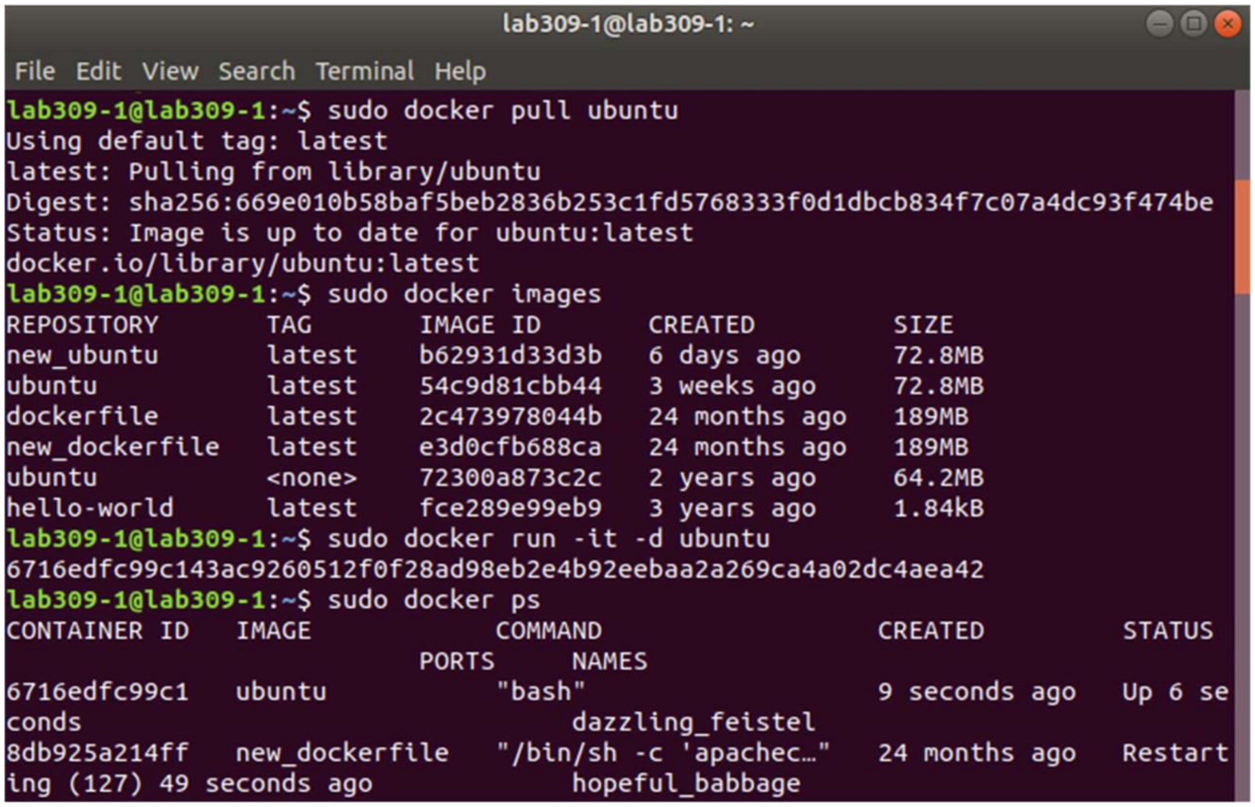
```
lab309-1@lab309-1:~$ sudo apt install docker.io
[sudo] password for lab309-1:
Sorry, try again.
[sudo] password for lab309-1:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 docker.io : Depends: containerd (>= 1.2.6-0ubuntu1~)
E: Unable to correct problems, you have held broken packages.
```



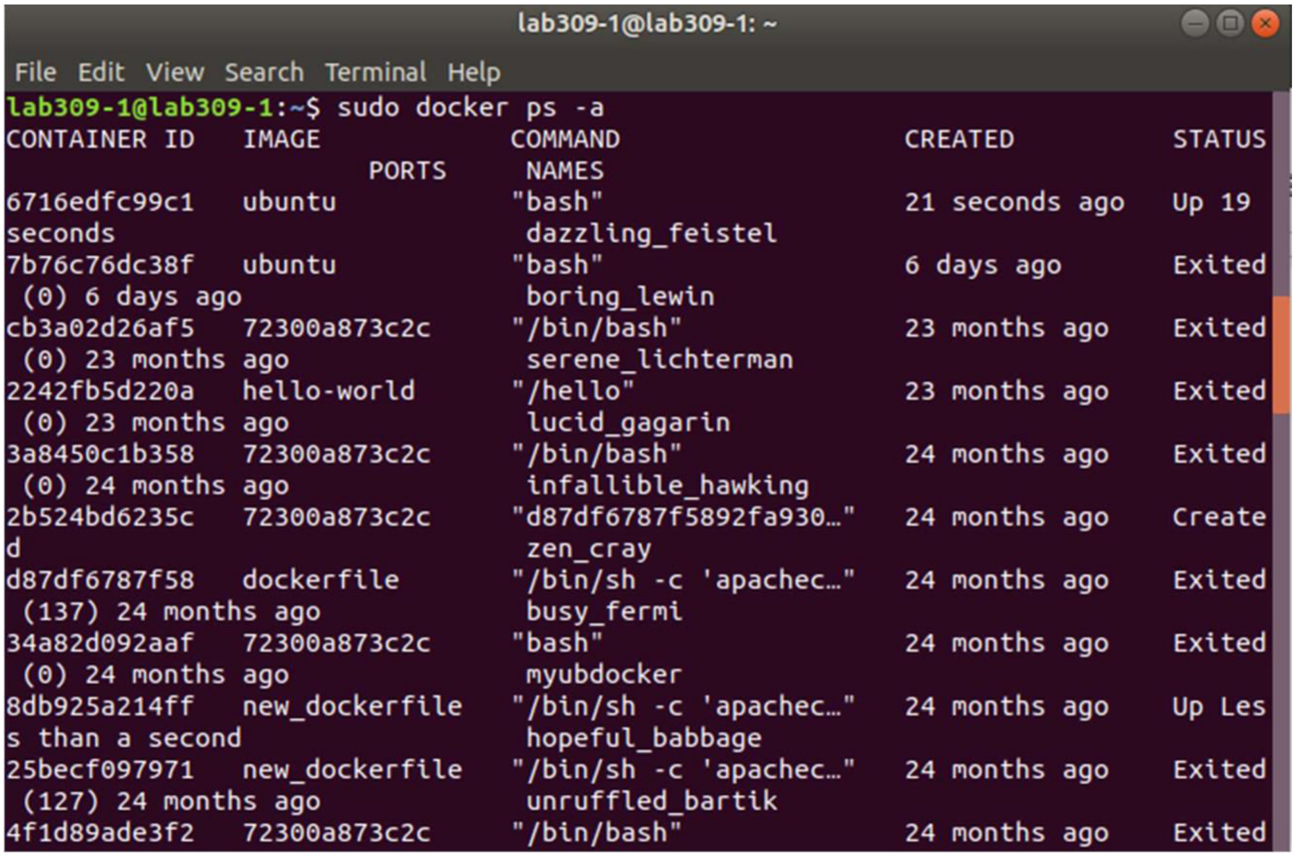
```
lab309-1@lab309-1:~$ sudo docker --version
Docker version 20.10.7, build f0df350
```



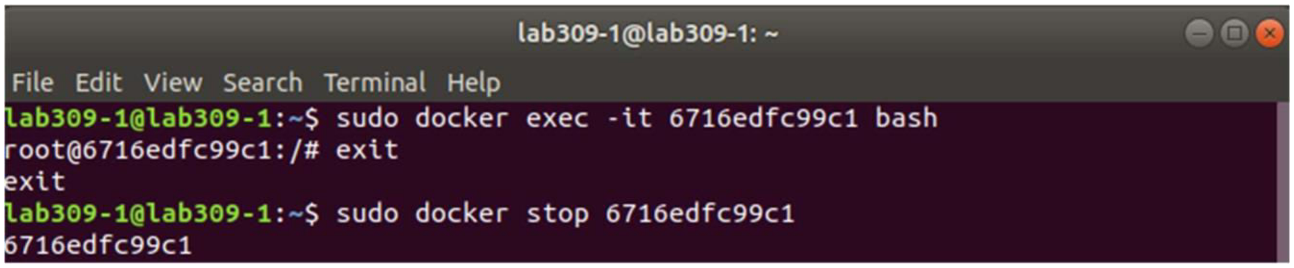
```
lab309-1@lab309-1:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```



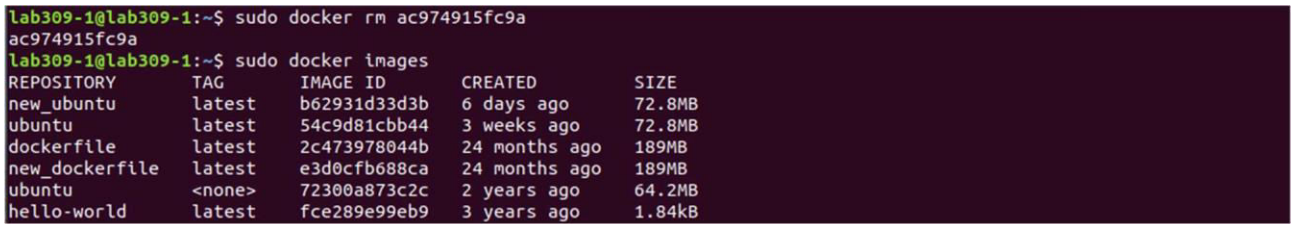
```
lab309-1@lab309-1:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
lab309-1@lab309-1:~$ sudo docker images
REPOSITORY       TAG       IMAGE ID       CREATED         SIZE
new_ubuntu       latest    b62931d33d3b   6 days ago      72.8MB
ubuntu           latest    54c9d81cbb44   3 weeks ago     72.8MB
dockerfile       latest    2c473978044b   24 months ago   189MB
new_dockerfile   latest    e3d0cfb688ca   24 months ago   189MB
ubuntu           <none>    72300a873c2c   2 years ago     64.2MB
hello-world      latest    fce289e99eb9   3 years ago     1.84kB
lab309-1@lab309-1:~$ sudo docker run -it -d ubuntu
6716edfc99c143ac9260512f0f28ad98eb2e4b92eebaa2a269ca4a02dc4aea42
lab309-1@lab309-1:~$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS
                         PORTS     NAMES
6716edfc99c1   ubuntu           "bash"                   9 seconds ago   Up 6 se
conds                              dazzling_feistel
8db925a214ff   new_dockerfile   "/bin/sh -c 'apachec…"   24 months ago   Restart
ing (127) 49 seconds ago           hopeful_babbage
```



```
lab309-1@lab309-1:~$ sudo docker ps -a
CONTAINER ID   IMAGE            COMMAND                    CREATED           STATUS
                      PORTS       NAMES
6716edfc99c1   ubuntu           "bash"                     21 seconds ago    Up 19
seconds                            dazzling_feistel
7b76c76dc38f   ubuntu           "bash"                     6 days ago        Exited
 (0) 6 days ago                    boring_lewin
cb3a02d26af5   72300a873c2c     "/bin/bash"                23 months ago     Exited
 (0) 23 months ago                 serene_lichterman
2242fb5d220a   hello-world      "/hello"                   23 months ago     Exited
 (0) 23 months ago                 lucid_gagarin
3a8450c1b358   72300a873c2c     "/bin/bash"                24 months ago     Exited
 (0) 24 months ago                 infallible_hawking
2b524bd6235c   72300a873c2c     "d87df6787f5892fa930…"     24 months ago     Create
d                                  zen_cray
d87df6787f58   dockerfile       "/bin/sh -c 'apachec…"     24 months ago     Exited
 (137) 24 months ago               busy_fermi
34a82d092aaf   72300a873c2c     "bash"                     24 months ago     Exited
 (0) 24 months ago                 myubdocker
8db925a214ff   new_dockerfile   "/bin/sh -c 'apachec…"     24 months ago     Up Les
s than a second                    hopeful_babbage
25becf097971   new_dockerfile   "/bin/sh -c 'apachec…"     24 months ago     Exited
 (127) 24 months ago               unruffled_bartik
4f1d89ade3f2   72300a873c2c     "/bin/bash"                24 months ago     Exited
```



```
lab309-1@lab309-1:~$ sudo docker exec -it 6716edfc99c1 bash
root@6716edfc99c1:/# exit
exit
lab309-1@lab309-1:~$ sudo docker stop 6716edfc99c1
6716edfc99c1
```



```
lab309-1@lab309-1:~$ sudo docker rm ac974915fc9a
ac974915fc9a
lab309-1@lab309-1:~$ sudo docker images
REPOSITORY       TAG       IMAGE ID       CREATED         SIZE
new_ubuntu       latest    b62931d33d3b   6 days ago      72.8MB
ubuntu           latest    54c9d81cbb44   3 weeks ago     72.8MB
dockerfile       latest    2c473978044b   24 months ago   189MB
new_dockerfile   latest    e3d0cfb688ca   24 months ago   189MB
ubuntu           <none>    72300a873c2c   2 years ago     64.2MB
hello-world      latest    fce289e99eb9   3 years ago     1.84kB
```

Conclusion: Successfully installed and configured Docker for creating Containers of different Operating System Images.

Srividya Subramanian
501865 23/02/22

DevOps Lab

EXPERIMENT – 5

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

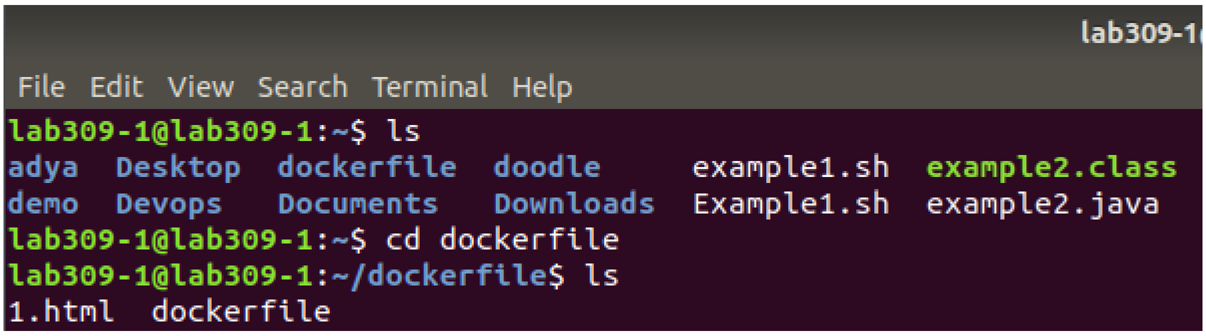
LAB OUTCOME – [LO1, LO4] - Remember the importance of DevOps tools used in software development life cycle. Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker

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.

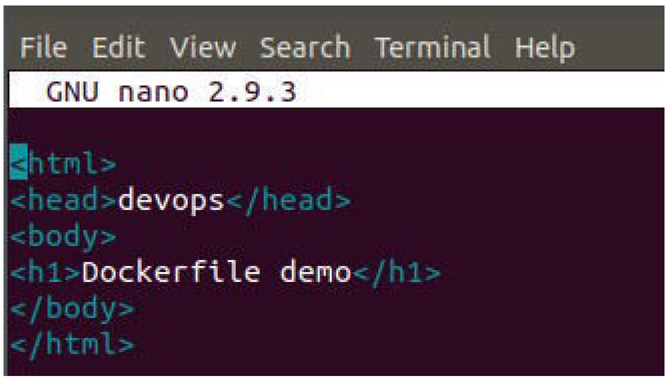
Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides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

PROCEDURE -

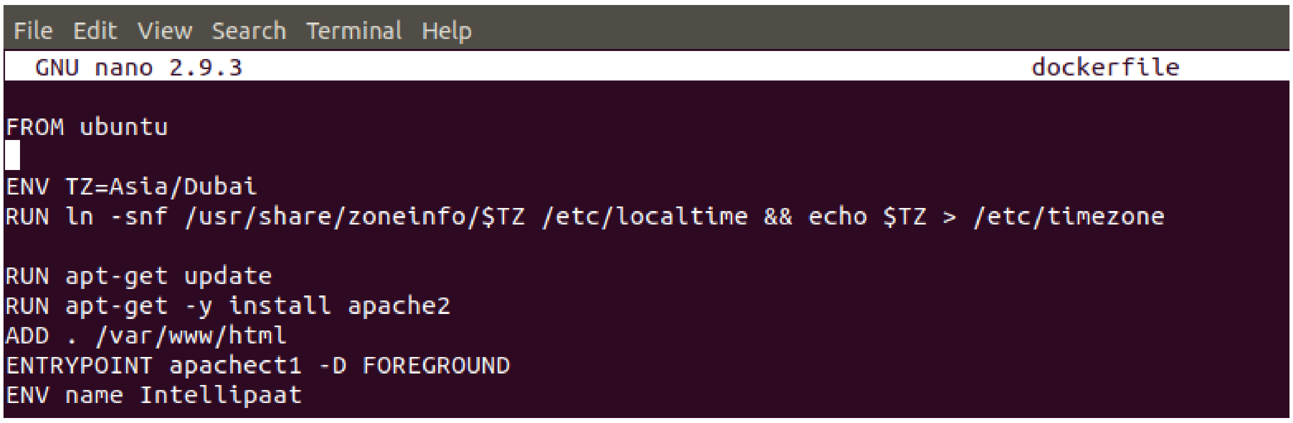
Steps to create dockerfile



```
lab309-1@lab309-1:~$ ls
adya  Desktop  dockerfile  doodle     example1.sh  example2.class
demo  Devops   Documents   Downloads  Example1.sh  example2.java
lab309-1@lab309-1:~$ cd dockerfile
lab309-1@lab309-1:~/dockerfile$ ls
1.html  dockerfile
```

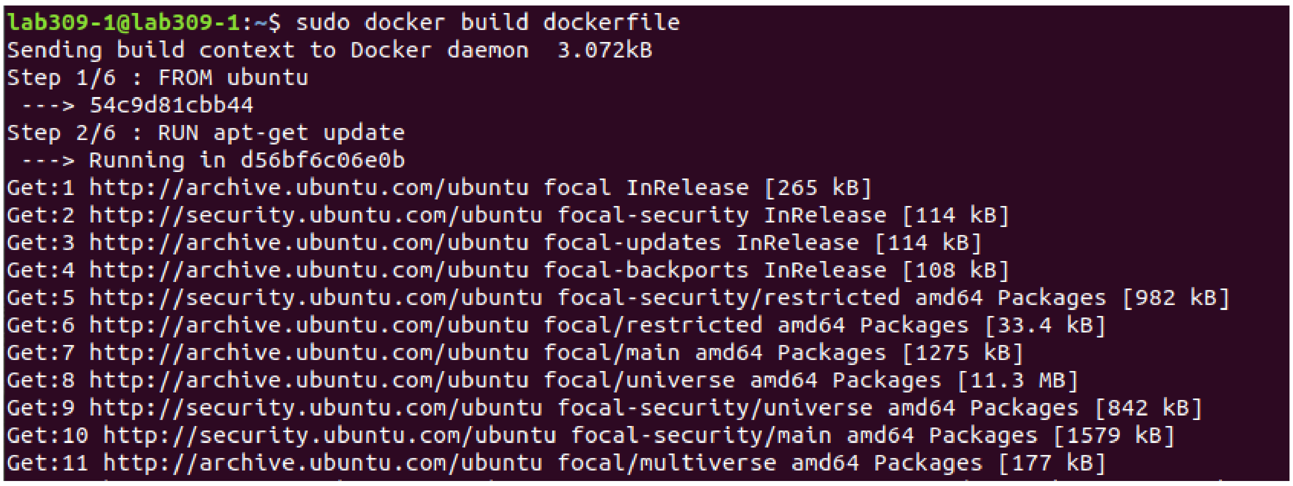


```
<html>
<head>devops</head>
<body>
<h1>Dockerfile demo</h1>
</body>
</html>
```

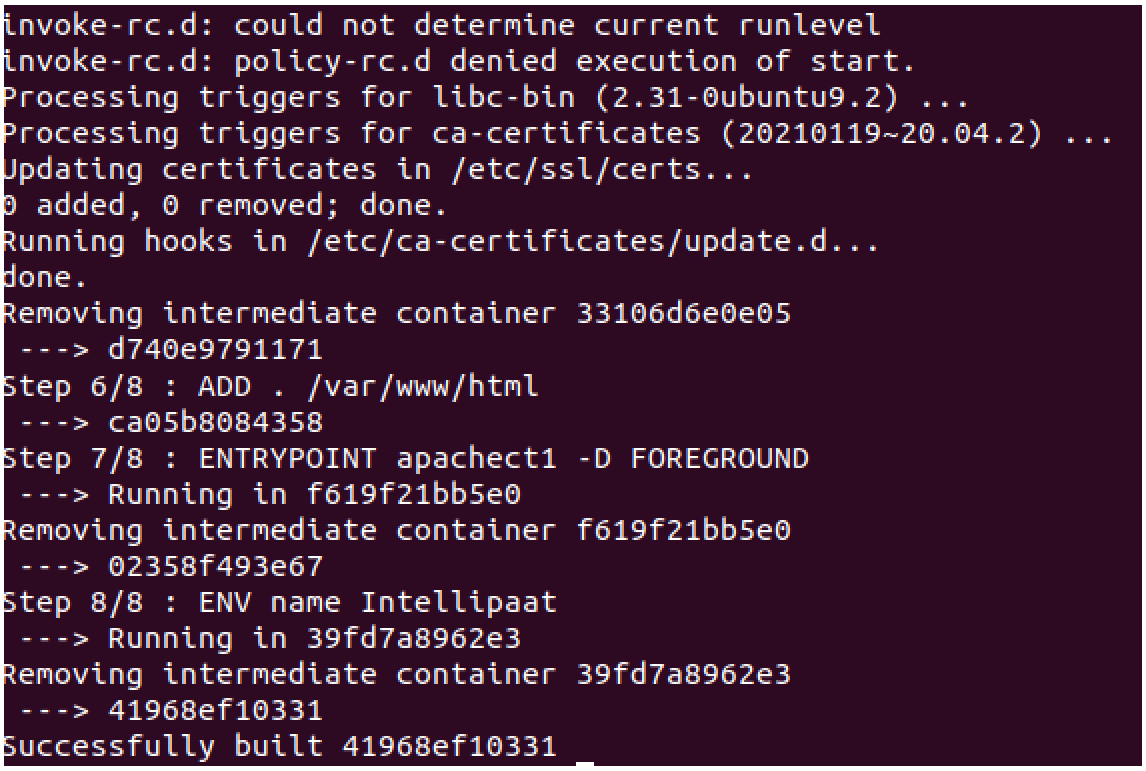


```
  GNU nano 2.9.3                                                    dockerfile

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
```

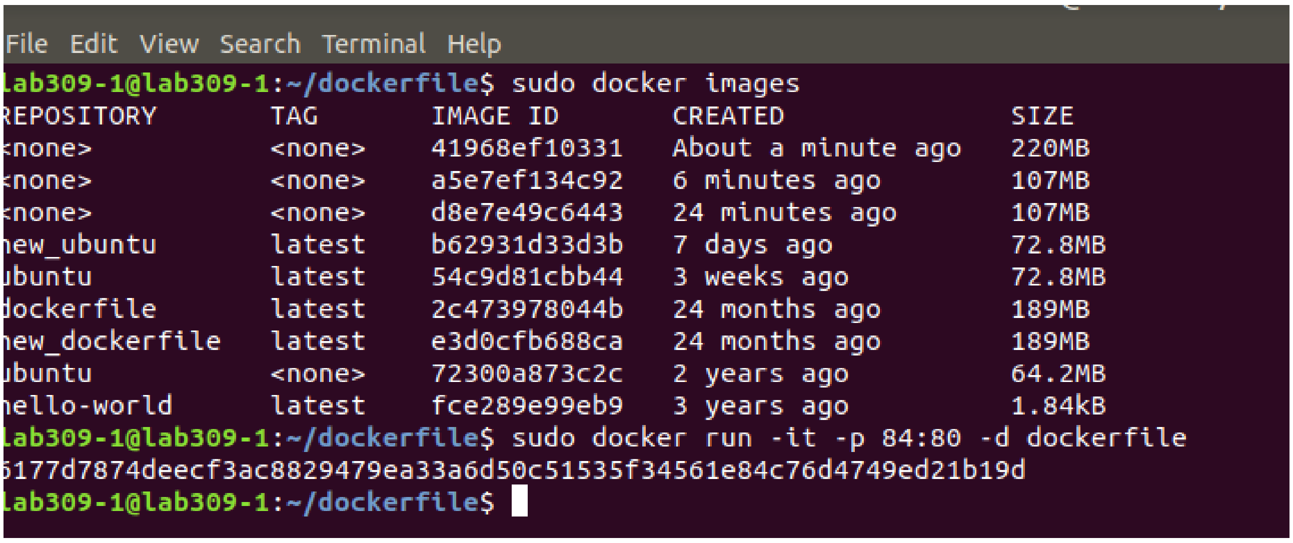
Steps to build image of dockerfile



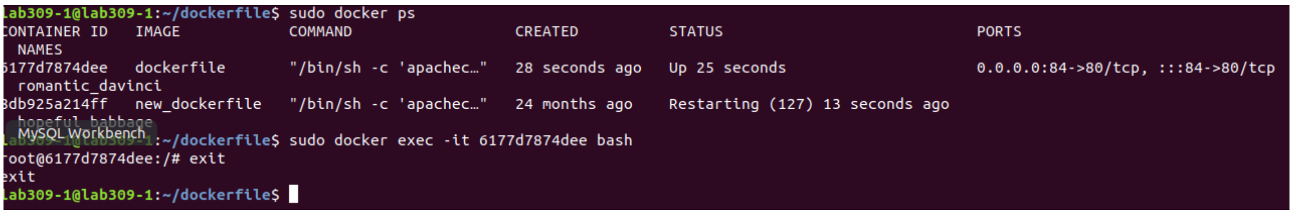
```
lab309-1@lab309-1:~$ sudo docker build dockerfile
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM ubuntu
 ---> 54c9d81cbb44
Step 2/6 : RUN apt-get update
 ---> Running in d56bf6c06e0b
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu 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]
```



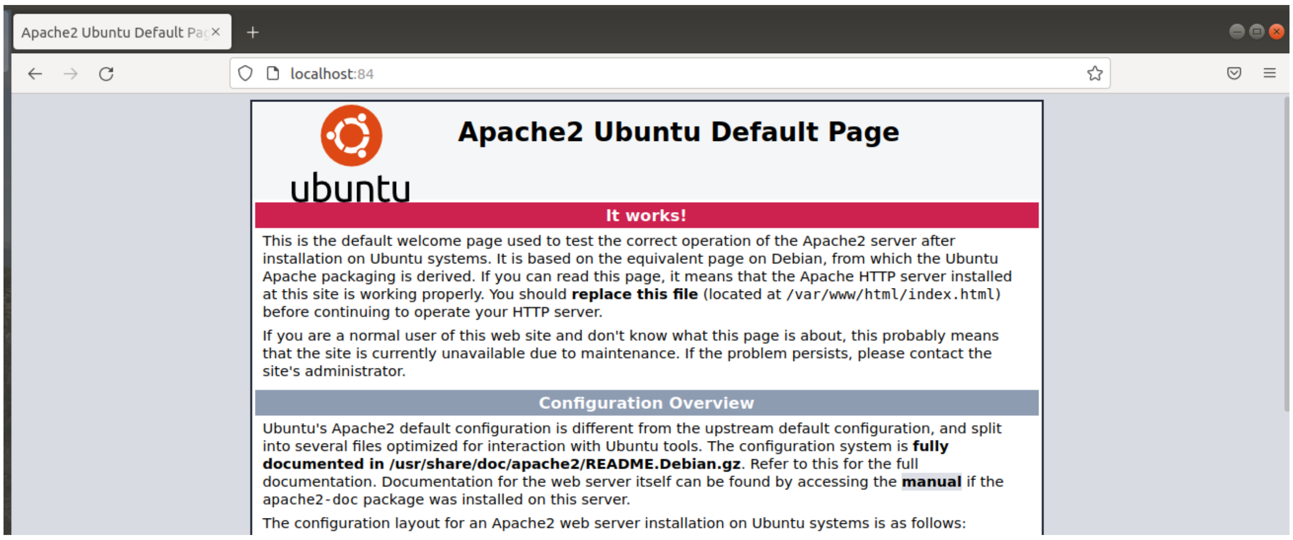
```
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for ca-certificates (20210119~20.04.2) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Removing intermediate container 33106d6e0e05
 ---> d740e9791171
Step 6/8 : ADD . /var/www/html
 ---> ca05b8084358
Step 7/8 : ENTRYPOINT apachect1 -D FOREGROUND
 ---> Running in f619f21bb5e0
Removing intermediate container f619f21bb5e0
 ---> 02358f493e67
Step 8/8 : ENV name Intellipaat
 ---> Running in 39fd7a8962e3
Removing intermediate container 39fd7a8962e3
 ---> 41968ef10331
Successfully built 41968ef10331
```



```
lab309-1@lab309-1:~/dockerfile$ sudo docker images
REPOSITORY       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
new_ubuntu       latest    b62931d33d3b   7 days ago           72.8MB
ubuntu           latest    54c9d81cbb44   3 weeks ago          72.8MB
dockerfile       latest    2c473978044b   24 months ago        189MB
new_dockerfile   latest    e3d0cfb688ca   24 months ago        189MB
ubuntu           <none>    72300a873c2c   2 years ago          64.2MB
hello-world      latest    fce289e99eb9   3 years ago          1.84kB
lab309-1@lab309-1:~/dockerfile$ sudo docker run -it -p 84:80 -d dockerfile
6177d7874deecf3ac8829479ea33a6d50c51535f34561e84c76d4749ed21b19d
lab309-1@lab309-1:~/dockerfile$ 
```



```
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
3db925a214ff   new_dockerfile   "/bin/sh -c 'apachec…"   24 months ago    Restarting (127) 13 seconds ago
  hopeful_babbage
lab309-1@lab309-1:~/dockerfile$ sudo docker exec -it 6177d7874dee bash
root@6177d7874dee:/# exit
exit
lab309-1@lab309-1:~/dockerfile$ 
```



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

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

Srividya Subramanian
501865 9/03/22

DevOps Lab

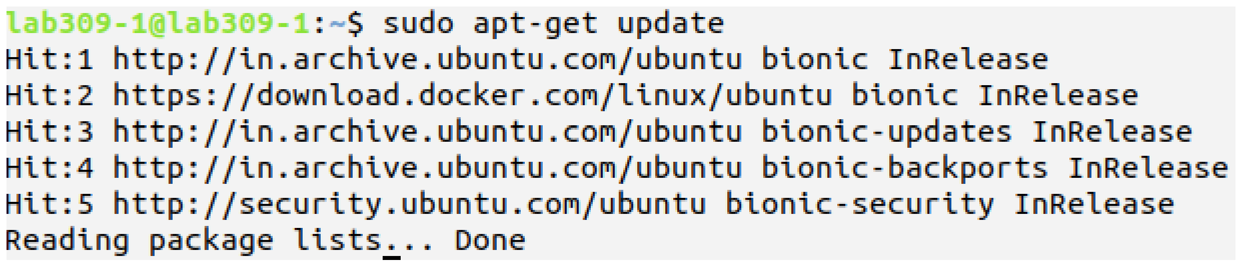
EXPERIMENT – 6

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

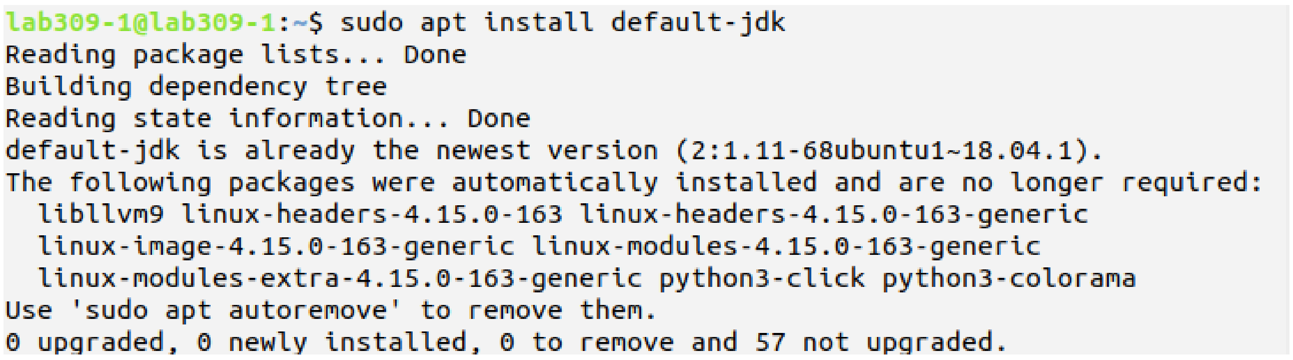
LAB OUTCOME – [LO3] – Examine the different version control strategies.

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

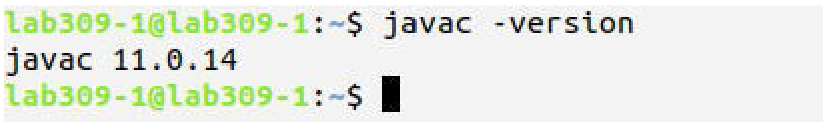
PROCEDURE -



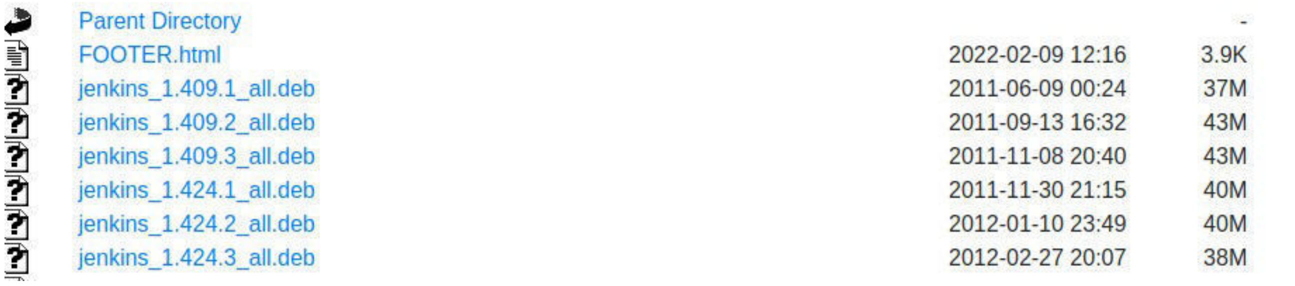
```
lab309-1@lab309-1:~$ sudo apt-get update
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease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



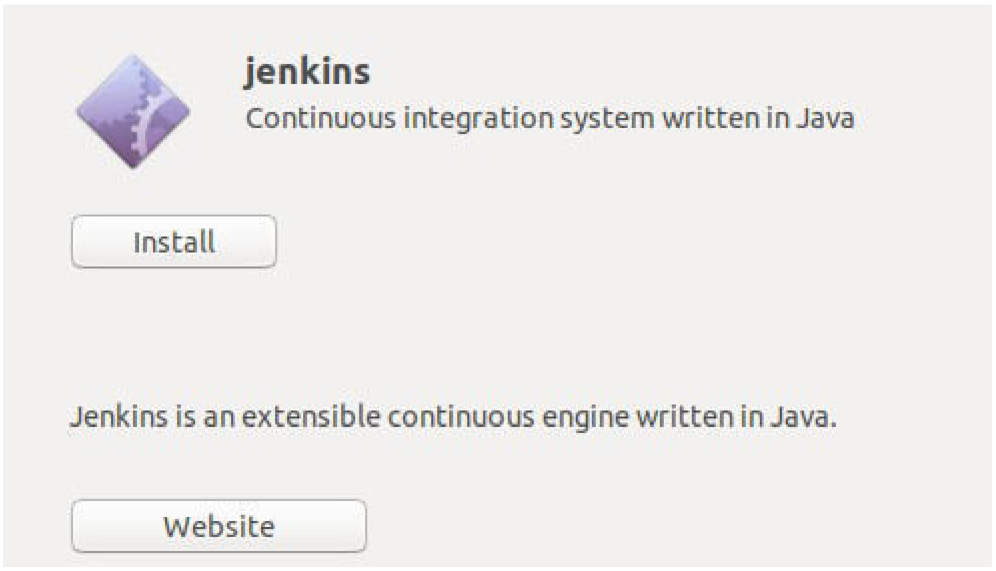
```
lab309-1@lab309-1:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jdk is already the newest version (2:1.11-68ubuntu1~18.04.1).
The following packages were automatically installed and are no longer required:
  libllvm9 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.
```



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



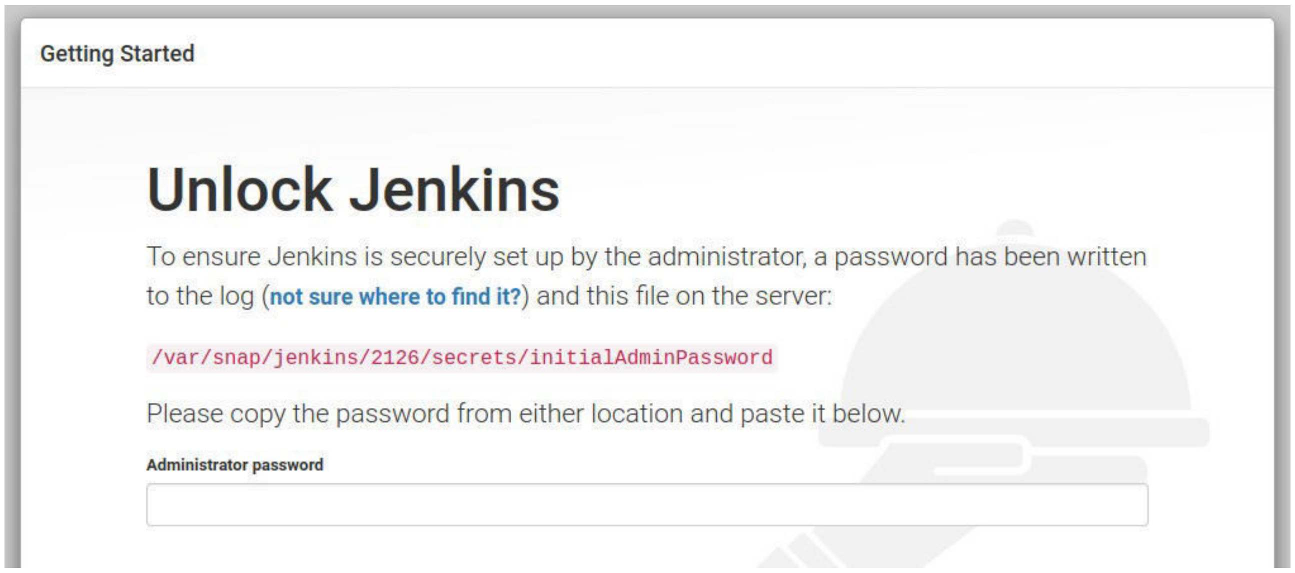
Parent Directory		-
FOOTER.html	2022-02-09 12:16	3.9K
jenkins_1.409.1_all.deb	2011-06-09 00:24	37M
jenkins_1.409.2_all.deb	2011-09-13 16:32	43M
jenkins_1.409.3_all.deb	2011-11-08 20:40	43M
jenkins_1.424.1_all.deb	2011-11-30 21:15	40M
jenkins_1.424.2_all.deb	2012-01-10 23:49	40M
jenkins_1.424.3_all.deb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



jenkins

Continuous integration system written in Java

Jenkins is an extensible continuous engine written in Java.



Getting Started

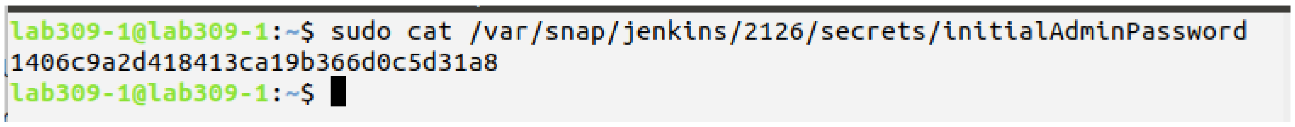
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

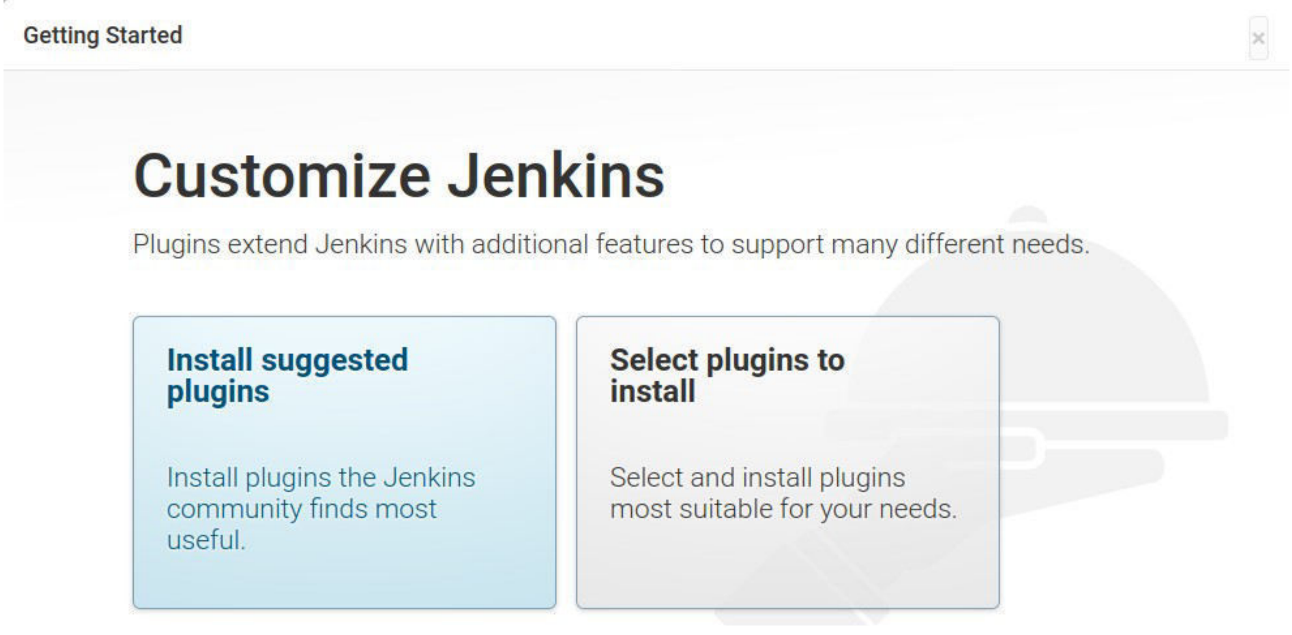
`/var/snap/jenkins/2126/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



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


Getting Started

Customize Jenkins

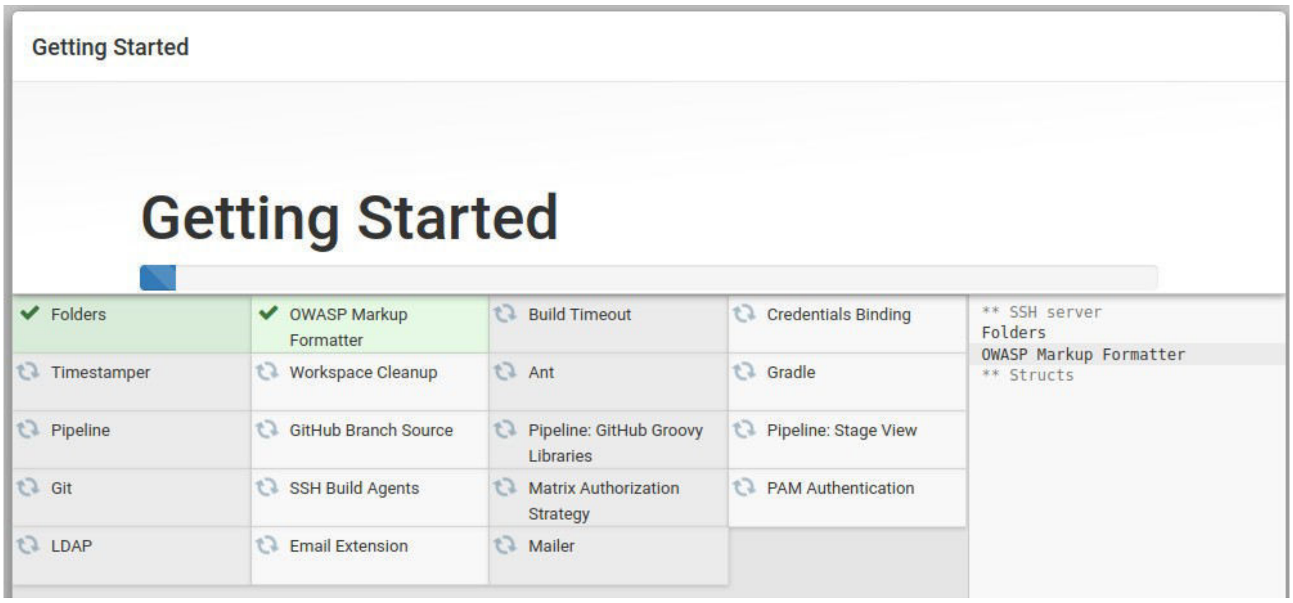
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

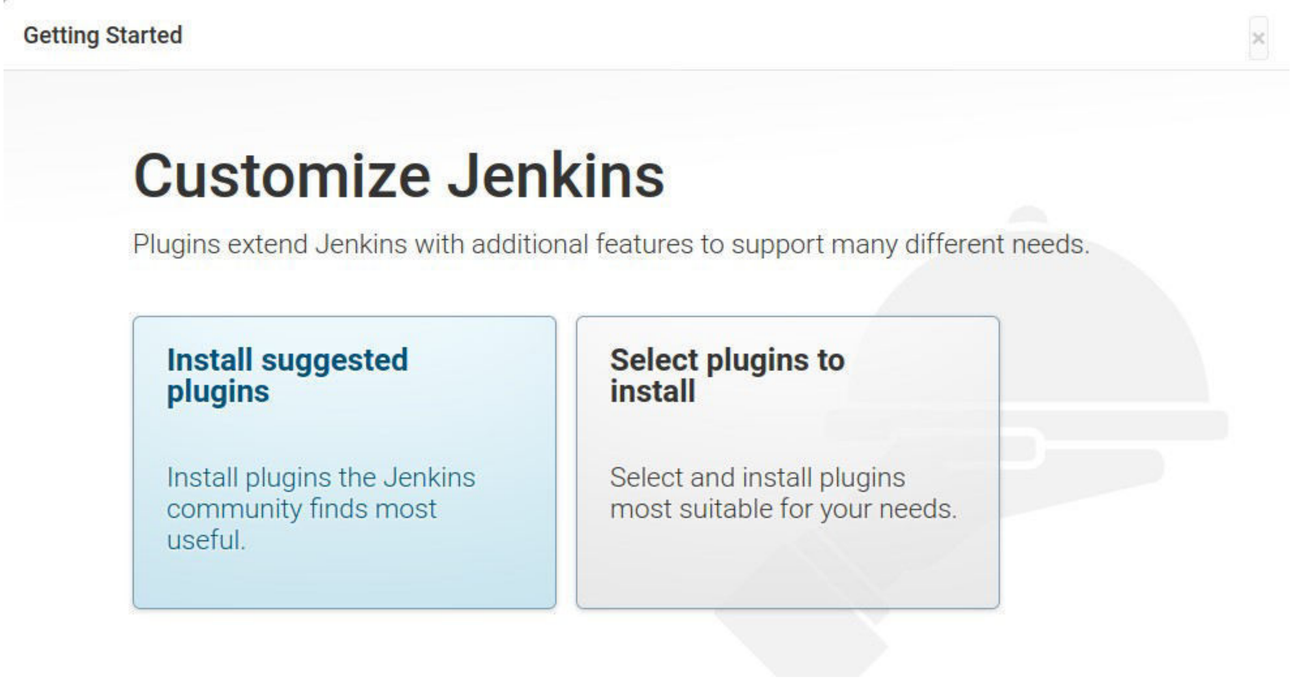
Select and install plugins most suitable for your needs.


Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
** SSH server
Folders
OWASP Markup Formatter
** Structs
```


Getting Started

Customize Jenkins

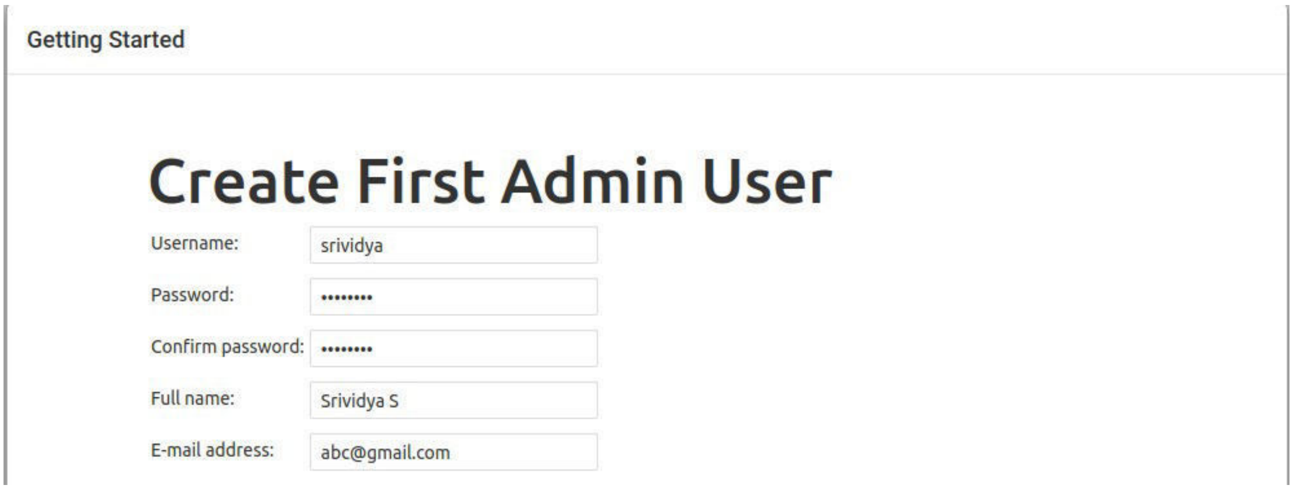
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

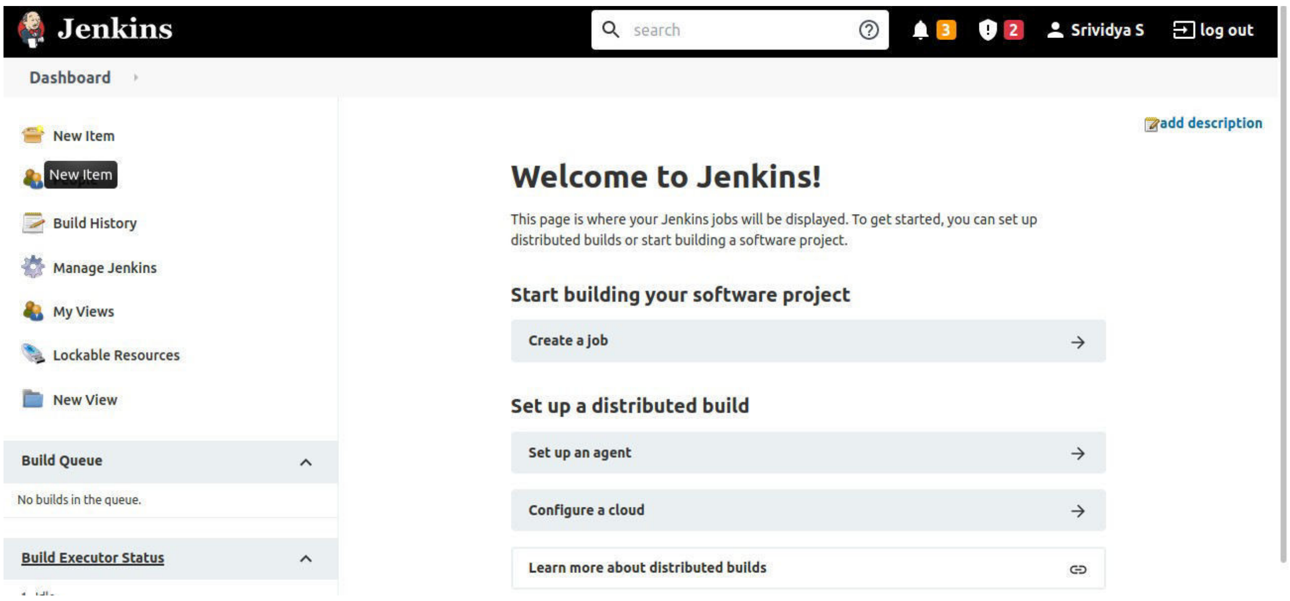
Select plugins to install

Select and install plugins most suitable for your needs.


Getting Started

Create First Admin User

Username:	srividya
Password:	••••••••
Confirm password:	••••••••
Full name:	Srividya S
E-mail address:	abc@gmail.com



No builds in the queue.

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

CONCLUSION – Successfully installed Jenkins for continuous integration purpose.

Srividya Subramanian
501865 23/03/22

DevOps Lab

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

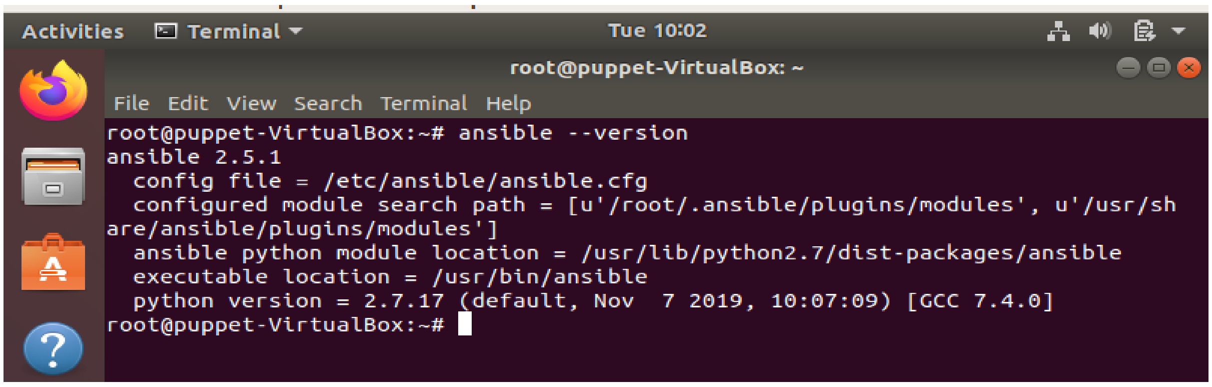
Step 1
We need to make it possible for our node to access the Ansible server. We do this via Secure Shell (SSH). Copy the server's SSH public key to the node. If your server doesn't have a key yet, generate one with the command:
$ ssh-keygen

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.


Activities
Terminal
Tue 10:02
root@puppet-VirtualBox: ~
File Edit View Search Terminal Help

```
root@puppet-VirtualBox:~# ansible --version
ansible 2.5.1
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/sh
are/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Nov  7 2019, 10:07:09) [GCC 7.4.0]
root@puppet-VirtualBox:~#
```

CONCLUSION – Installed and configured anisible successfully.

Anya Gupta
501866 23/03/22

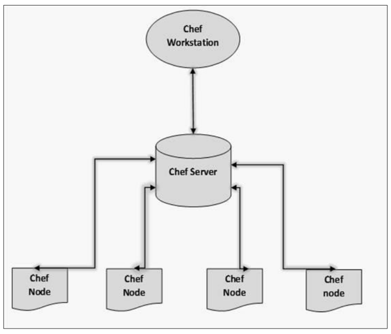
DevOps Lab

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.

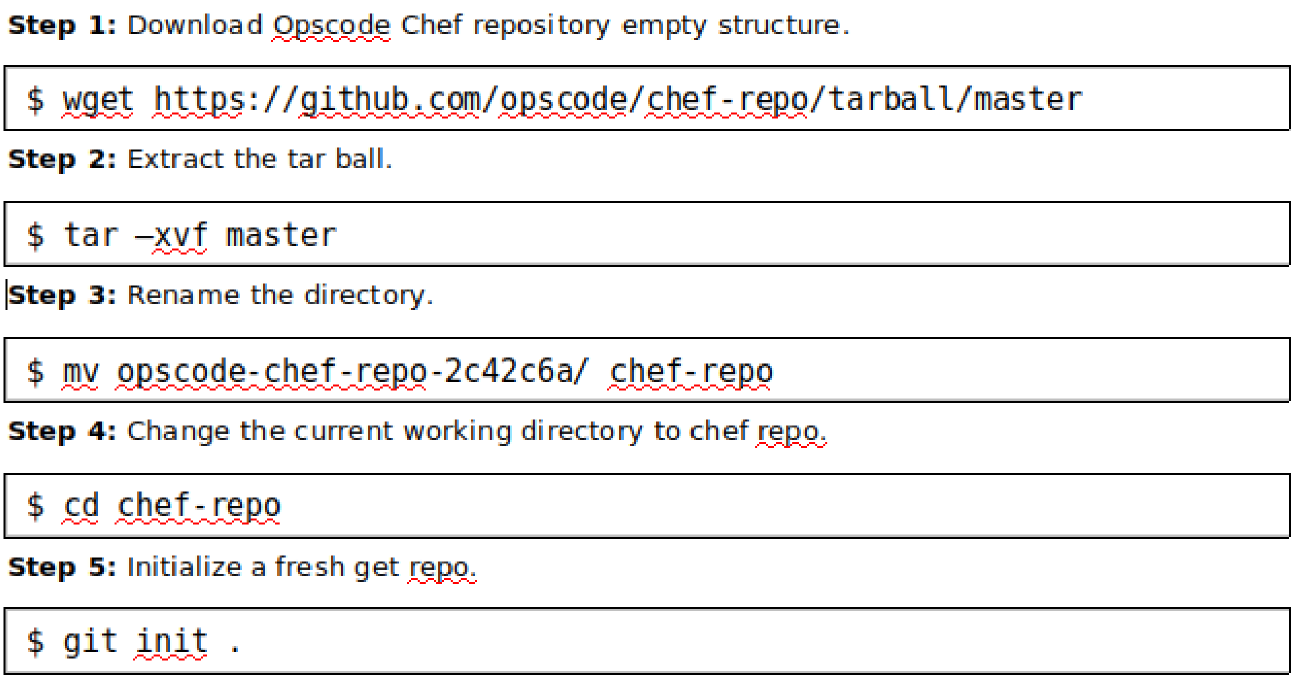
By using the above procedure, you will get an empty chef repo in place. You can then start working on developing the recipes and cookbooks. Once done, you can push the changes to the GitHub.

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.

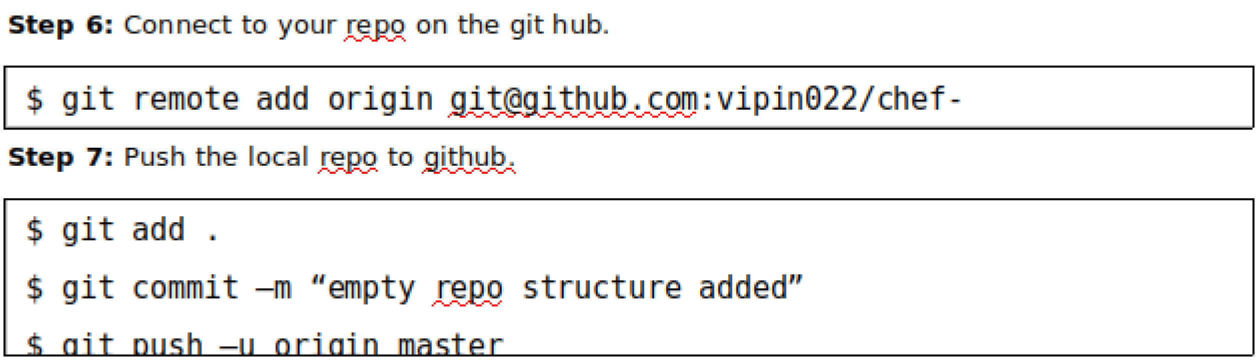
```
$ mv opscode-chef-repo-2c42c6a/ chef-repo
```

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.

Anya Gupta
IT Sem-8 B4 501866

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.

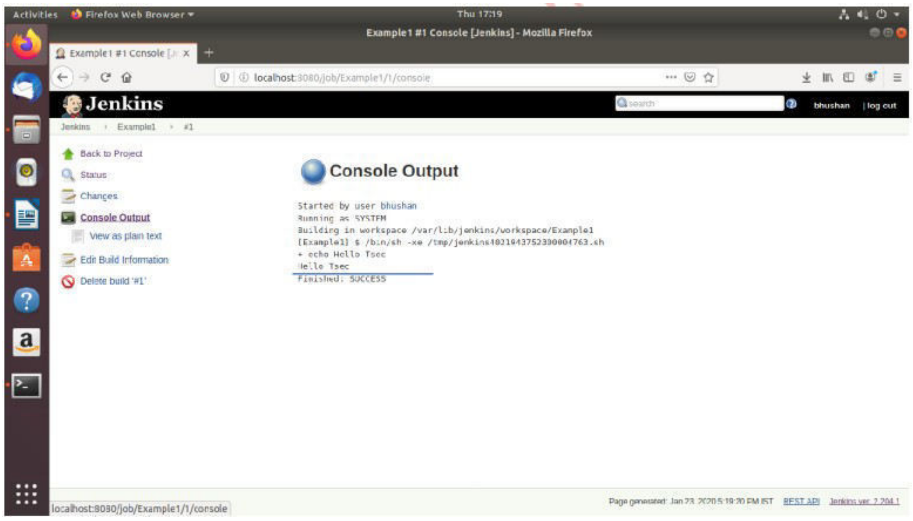
Step 3-: In this project we are going to learn how to run simple shell script on Jenkins. So, Click on Build option select Execute script from dropdown menu.

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



Console Output

```
Started by user bhushan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Example1
[Example1] $ /bin/sh -xe /tmp/jenkins4021943752330004763.sh
+ echo Hello Tsec
Hello Tsec
Finished: SUCCESS
```

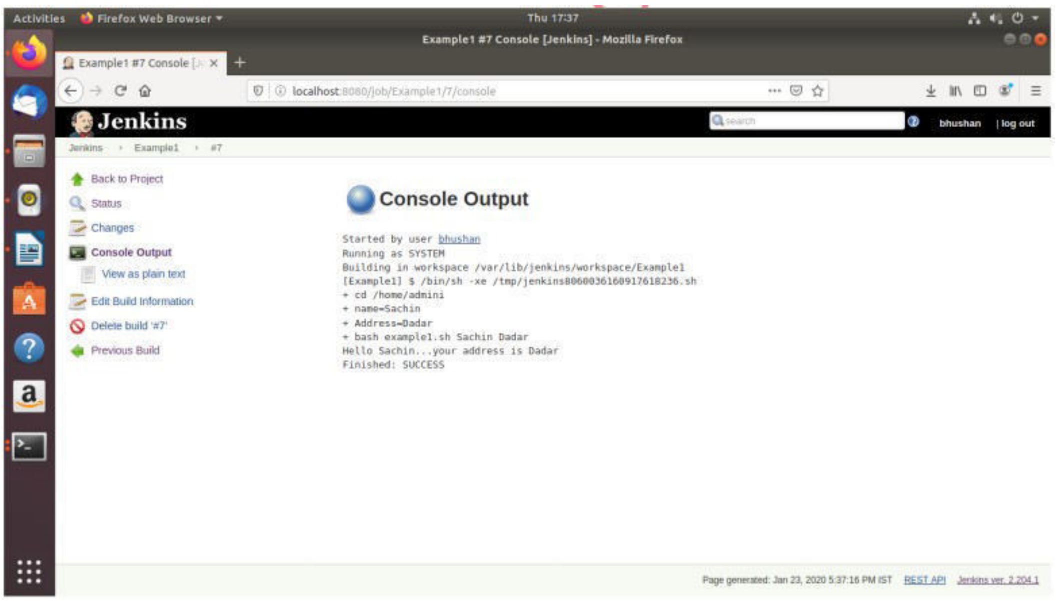
Example 1.2:

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 yourfile..Youcan get the location by pwdcommand>

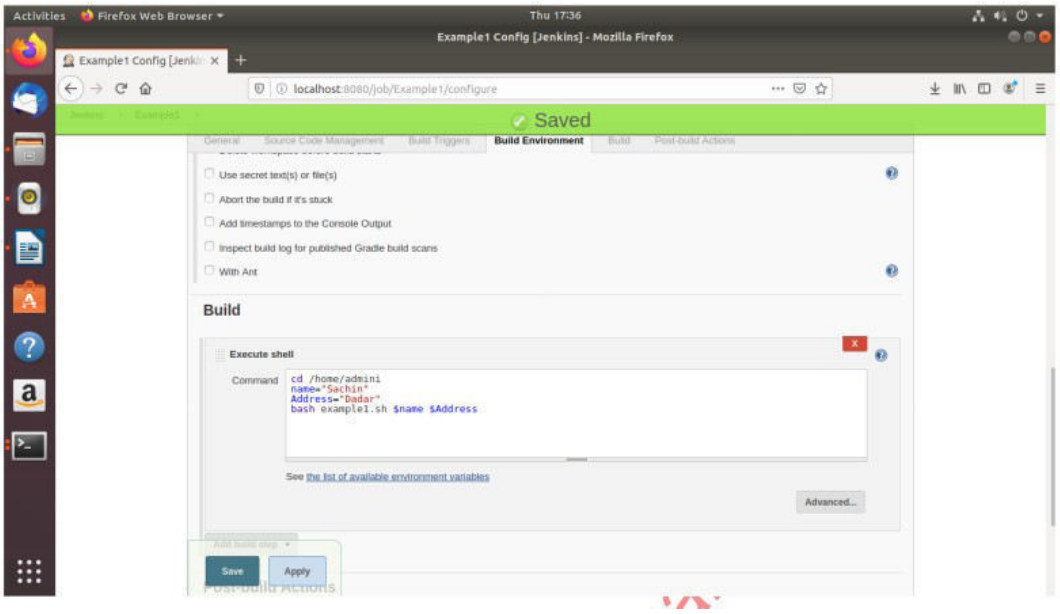


Console Output

```
Started by user bhushan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Example1
[Example1] $ /bin/sh -xe /tmp/jenkins8060036160917618236.sh
+ cd /home/admini
+ name=Sachin
+ Address=Dadar
+ bash example1.sh Sachin Dadar
Hello Sachin...your address is Dadar
Finished: SUCCESS
```

Page generated: Jan 23, 2020 5:37:16 PM IST REST API Jenkins ver. 2.204.1

Variation To This Program:


Example1 Config [Jenkins] - Mozilla Firefox
localhost:8080/job/Example1/configure
Saved
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant
Advanced...
Add build step
Save
Apply

Build

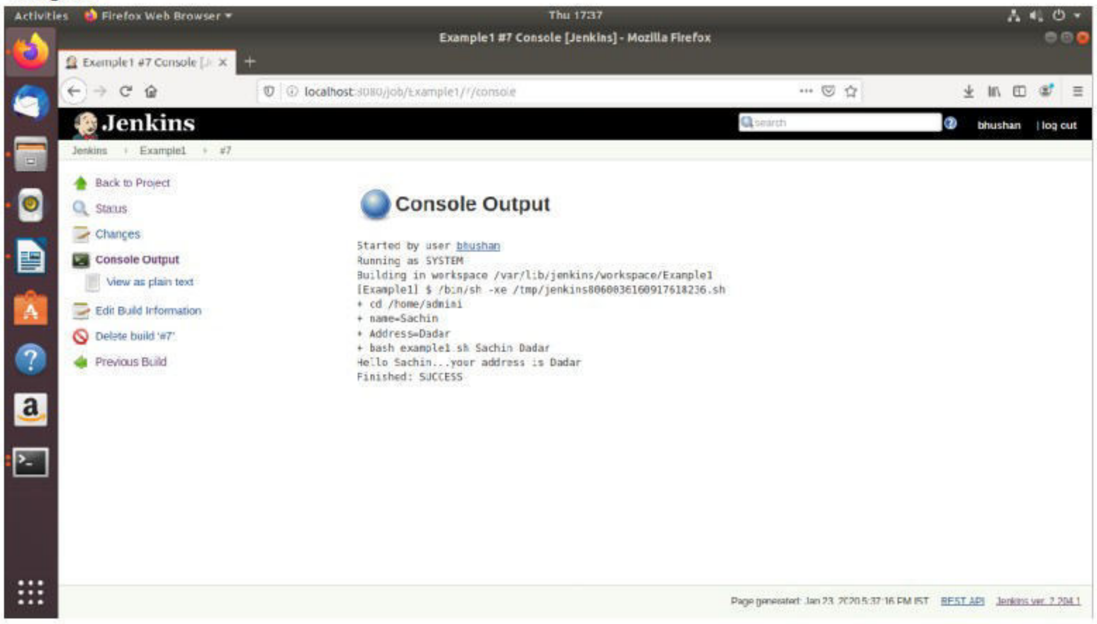
Execute shell

Command

```
cd /home/admini
name="Sachin"
Address="Dadar"
bash example1.sh $name $Address
```

See the list of available environment variables

Output



Console Output

```
Started by user bhushan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Example1
[Example1] $ /bin/sh -xe /tmp/jenkins8060036160917618236.sh
+ cd /home/admini
+ name=Sachin
+ Address=Dadar
+ bash example1.sh Sachin Dadar
Hello Sachin...your address is Dadar
Finished: SUCCESS
```

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.

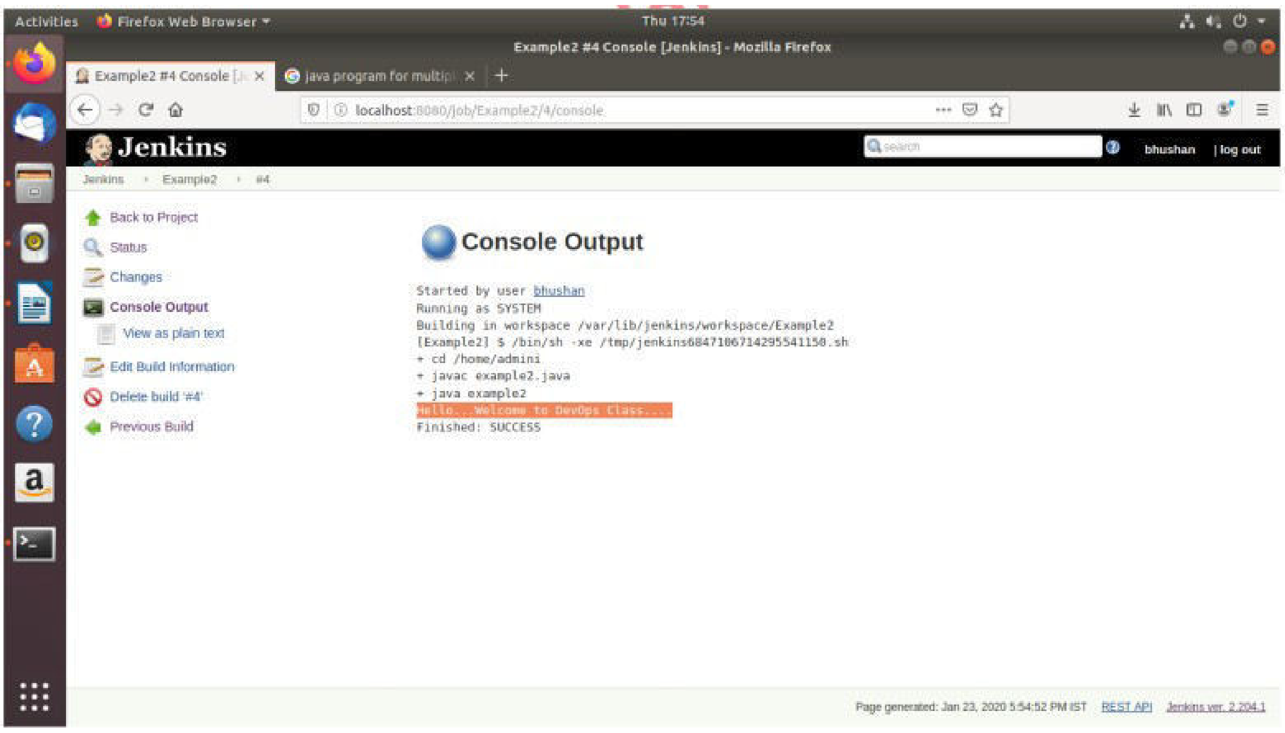
Step 2 -: Go to build option and change path to directory where you have stored java file followed by compile and run program commands.

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



Console Output

```
Started by user bhushan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Example2
[Example2] $ /bin/sh -xe /tmp/jenkins6847106714295541150.sh
+ cd /home/admini
+ javac example2.java
+ java example2
Hello...Welcome to DevOps Class....
Finished: SUCCESS
```

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 folowed 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.

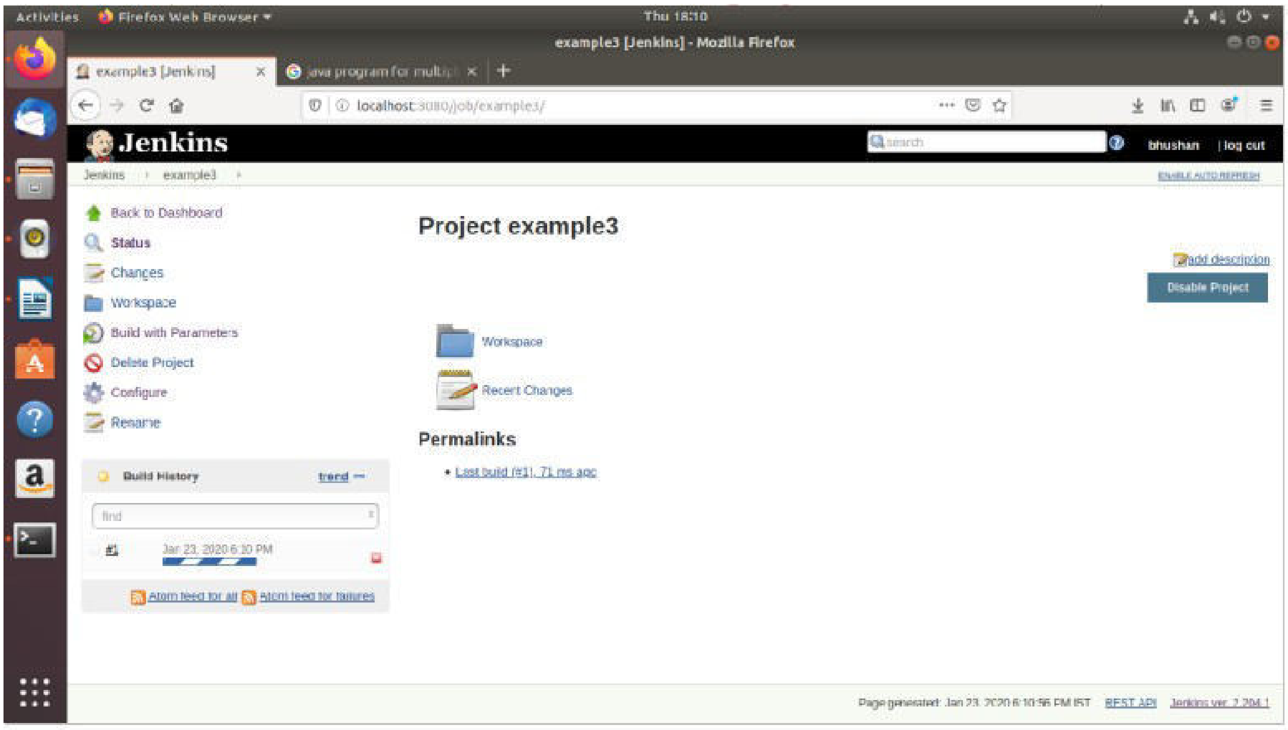
Step 5-: Specify name as “City” and add the choices in each line

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

Step 7-: Now, go back to jenkins, Selct 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.

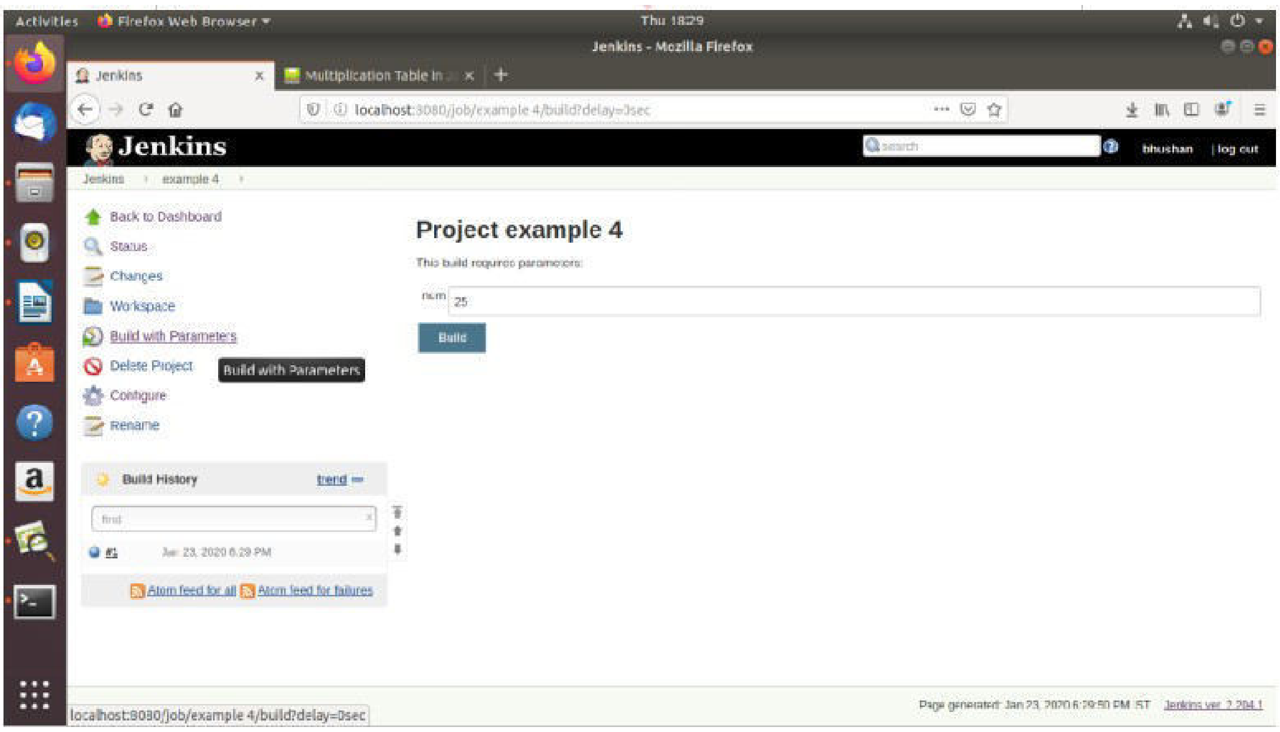
Step 2-: Delete class file and give write permission to program

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.



Project example 4

This build requires parameters:

num 25

#1 Jan 23, 2020 6:29 PM

Page generated: Jan 23, 2020 6:29:50 PM IST Jenkins ver. 2.204.1

localhost:8080/job/example 4/build?delay=0sec

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

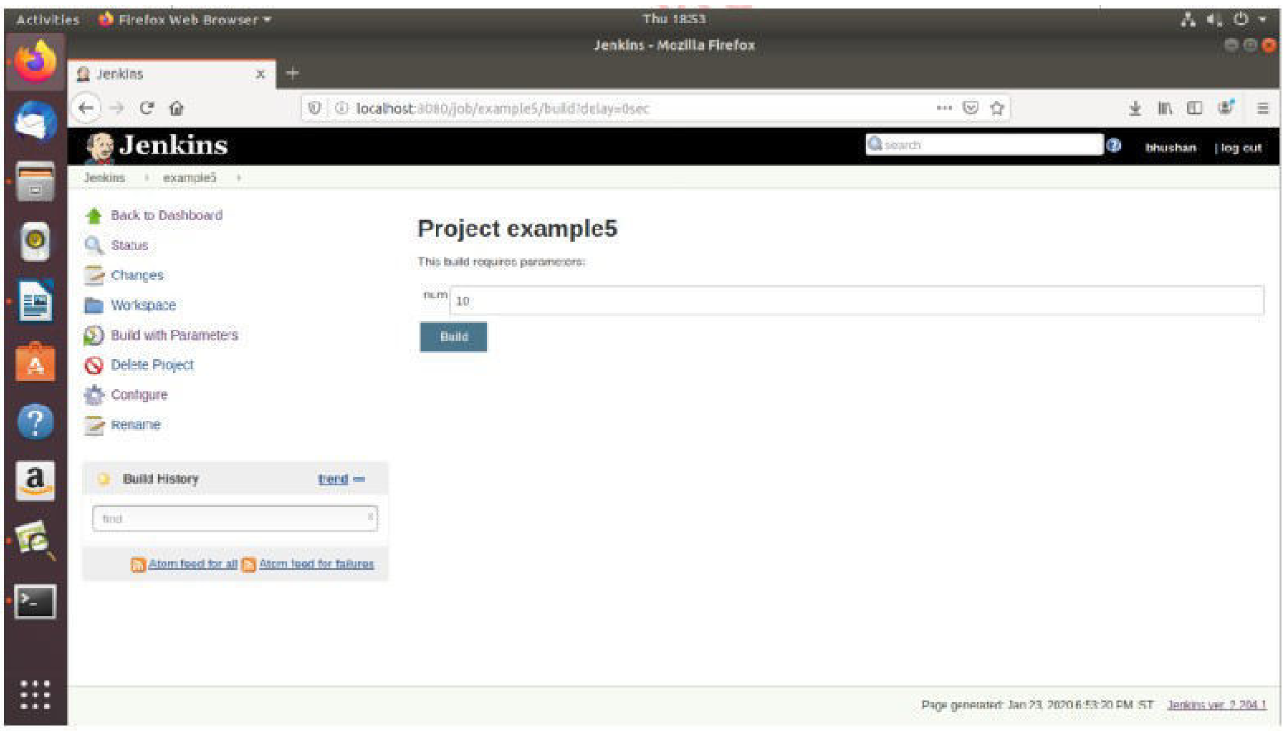
Step 3-: Now select, this project is parameterized, select string parameter and specify parameter name as “num”

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.

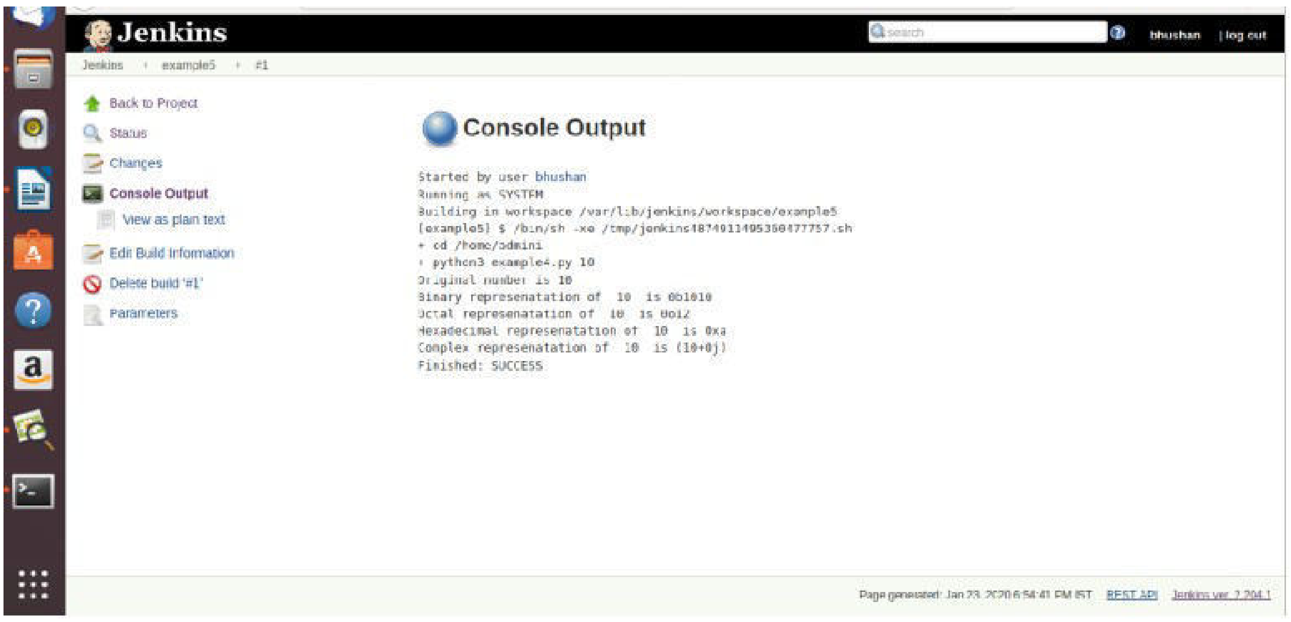


Project example5

This build requires parameters:

num 10

Build



Console Output

```
Started by user bhushan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/example5
[example5] $ /bin/sh -xe /tmp/jenkins1874911495360477757.sh
+ cd /home/admini
+ python3 example4.py 10
Original number is 10
Binary represenatation of  10  is 0b1010
Octal represenatation of  10  is 0o12
Hexadecimal represenatation of  10  is 0xa
Complex represenatation of  10  is (10+0j)
Finished: SUCCESS
```

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

Name: Anya Gupta
Roll no.: 501866
IT Sem-8

DevOps 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

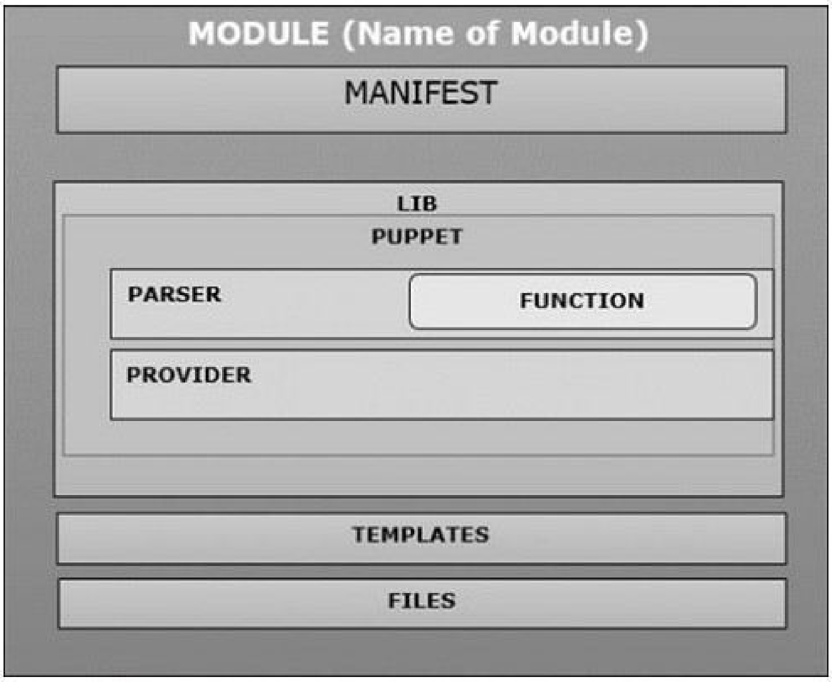


Fig 1. Components of puppet

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

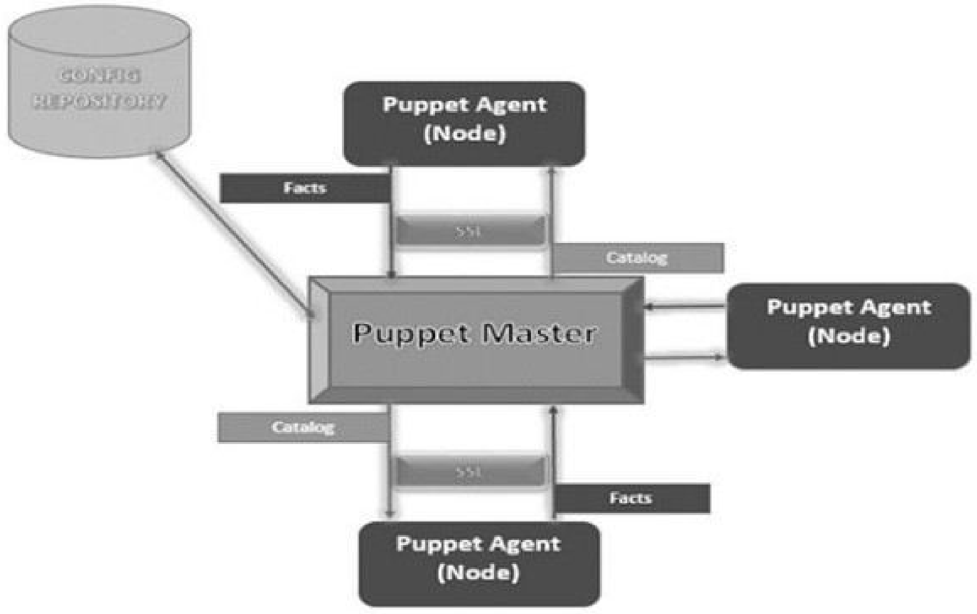


Fig 2. Architecture of puppet

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.

Name: Anya Gupta

Roll no.: 501866

IT Sem – 8

DevOps Experiment 4

Aim: To Install and Configure Docker for creating Containers of different Operating System Images.

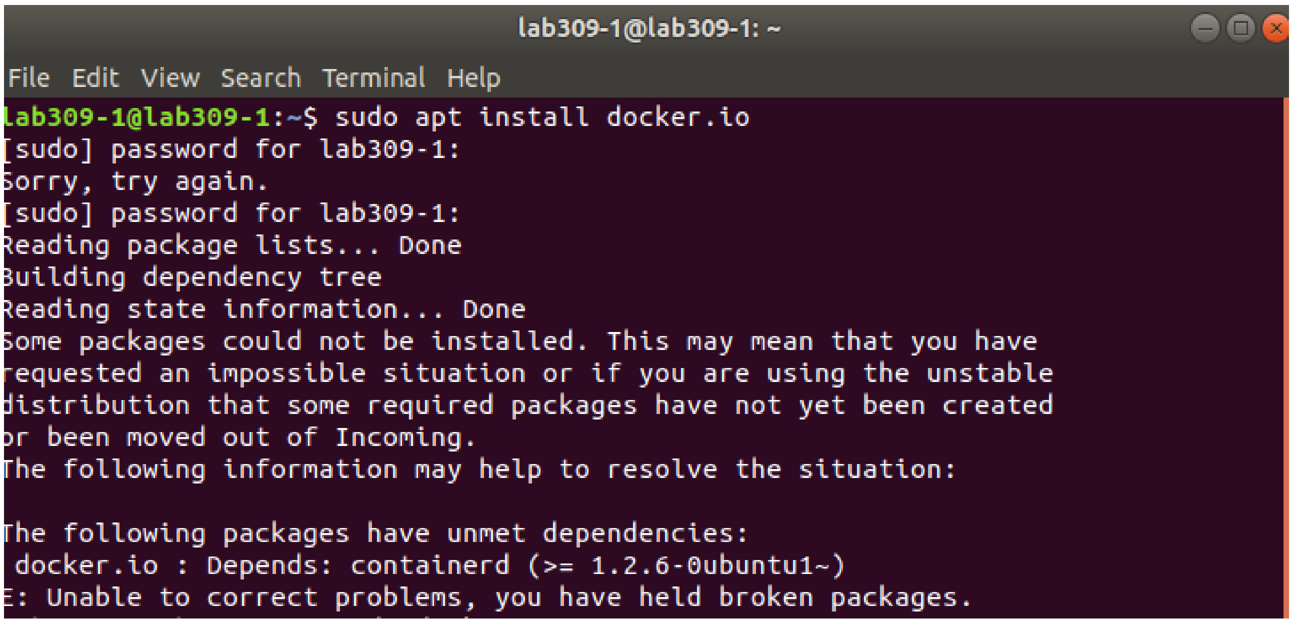
Lab Outcome: – [LO1, LO4] - Remember the importance of DevOps tools used in software development life cycle. Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker.

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.

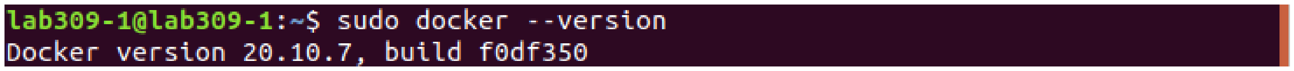
Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides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

Procedure:

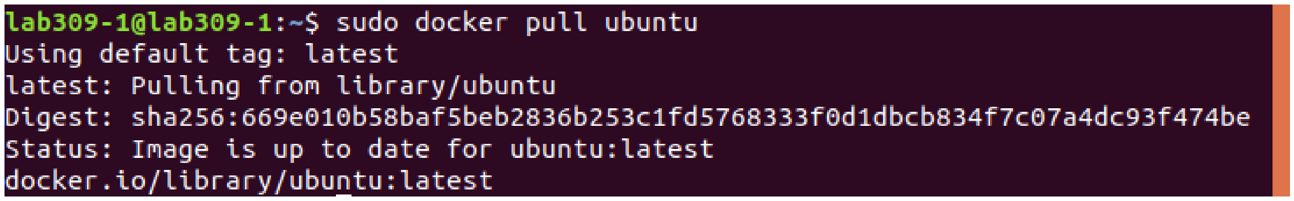
Installing Docker



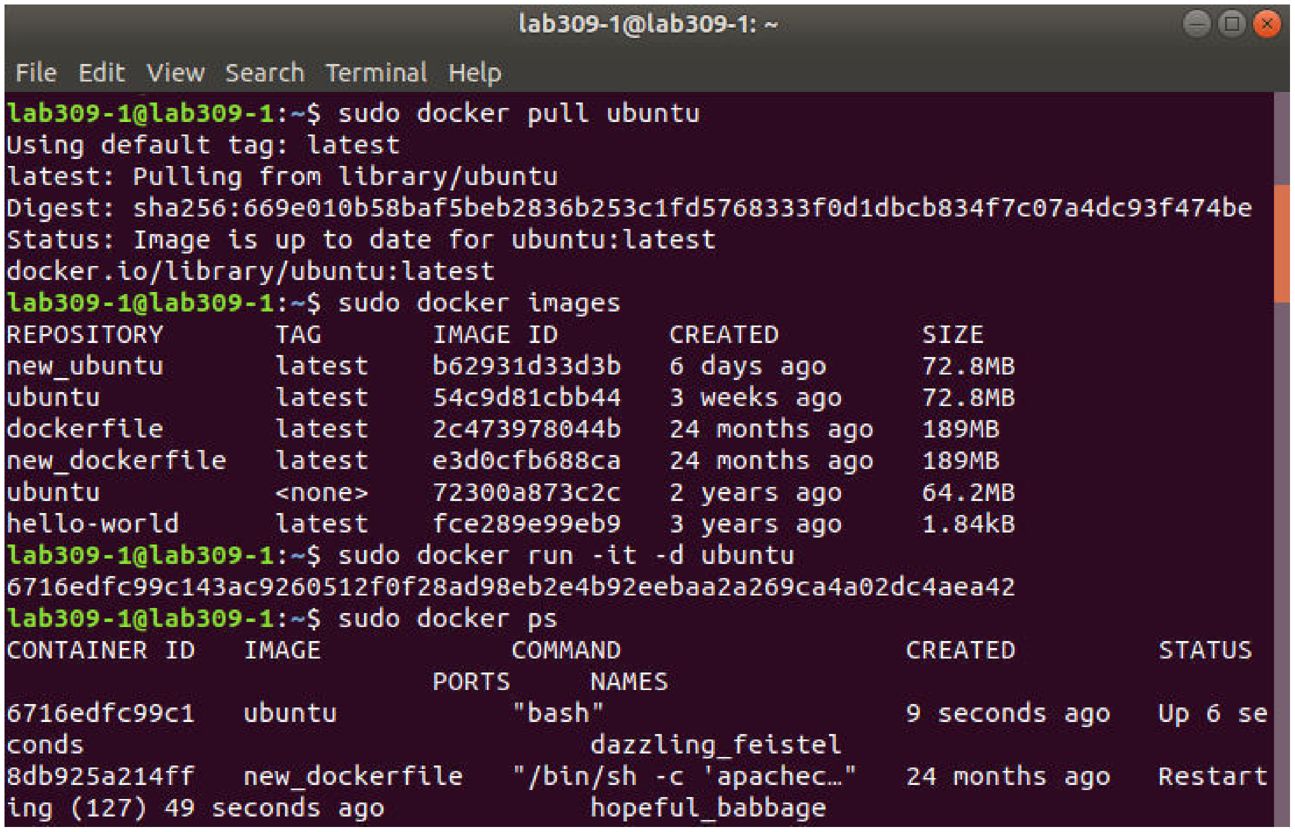
```
lab309-1@lab309-1:~$ sudo apt install docker.io
[sudo] password for lab309-1:
Sorry, try again.
[sudo] password for lab309-1:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 docker.io : Depends: containerd (>= 1.2.6-0ubuntu1~)
E: Unable to correct problems, you have held broken packages.
```



```
lab309-1@lab309-1:~$ sudo docker --version
Docker version 20.10.7, build f0df350
```

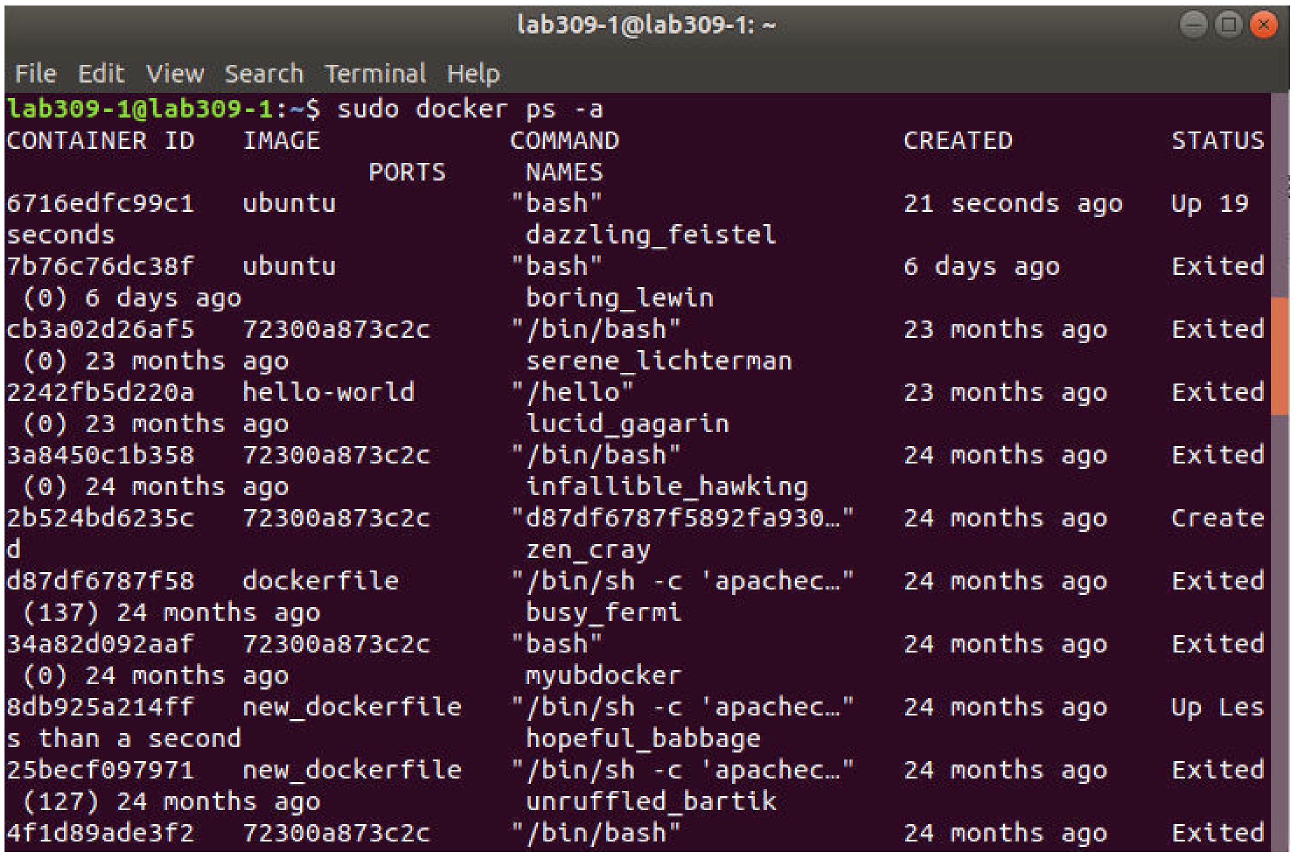


```
lab309-1@lab309-1:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

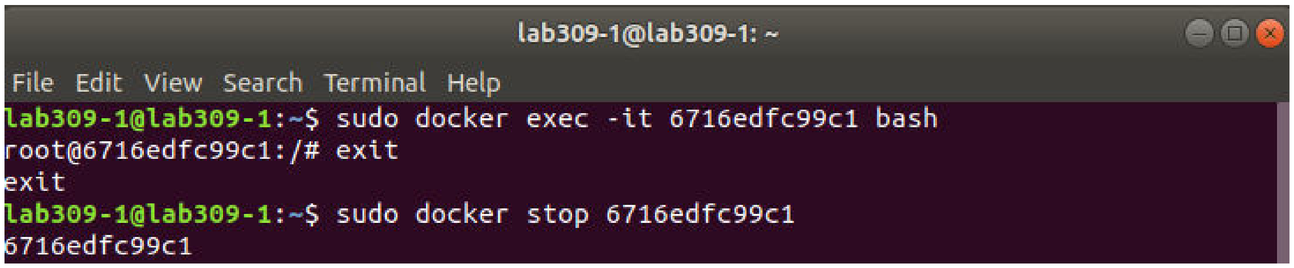

lab309-1@lab309-1: ~

File Edit View Search Terminal Help

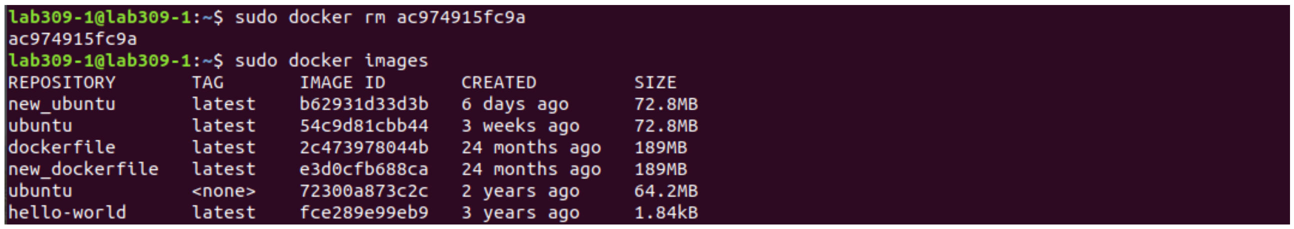
```
lab309-1@lab309-1:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
lab309-1@lab309-1:~$ sudo docker images
REPOSITORY       TAG        IMAGE ID       CREATED         SIZE
new_ubuntu       latest     b62931d33d3b   6 days ago      72.8MB
ubuntu           latest     54c9d81cbb44   3 weeks ago     72.8MB
dockerfile       latest     2c473978044b   24 months ago   189MB
new_dockerfile   latest     e3d0cfb688ca   24 months ago   189MB
ubuntu           <none>     72300a873c2c   2 years ago     64.2MB
hello-world      latest     fce289e99eb9   3 years ago     1.84kB
lab309-1@lab309-1:~$ sudo docker run -it -d ubuntu
6716edfc99c143ac9260512f0f28ad98eb2e4b92eebaa2a269ca4a02dc4aea42
lab309-1@lab309-1:~$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS
                          PORTS     NAMES
6716edfc99c1   ubuntu           "bash"                   9 seconds ago    Up 6 se
conds                               dazzling_feistel
8db925a214ff   new_dockerfile   "/bin/sh -c 'apachec…"   24 months ago    Restart
ing (127) 49 seconds ago            hopeful_babbage
```



```
lab309-1@lab309-1:~$ sudo docker ps -a
CONTAINER ID   IMAGE              COMMAND                    CREATED            STATUS
                      PORTS        NAMES
6716edfc99c1   ubuntu             "bash"                     21 seconds ago     Up 19
seconds                            dazzling_feistel
7b76c76dc38f   ubuntu             "bash"                     6 days ago         Exited
 (0) 6 days ago                    boring_lewin
cb3a02d26af5   72300a873c2c       "/bin/bash"                23 months ago      Exited
 (0) 23 months ago                 serene_lichterman
2242fb5d220a   hello-world        "/hello"                   23 months ago      Exited
 (0) 23 months ago                 lucid_gagarin
3a8450c1b358   72300a873c2c       "/bin/bash"                24 months ago      Exited
 (0) 24 months ago                 infallible_hawking
2b524bd6235c   72300a873c2c       "d87df6787f5892fa930…"     24 months ago      Create
d                                  zen_cray
d87df6787f58   dockerfile         "/bin/sh -c 'apachec…"     24 months ago      Exited
 (137) 24 months ago               busy_fermi
34a82d092aaf   72300a873c2c       "bash"                     24 months ago      Exited
 (0) 24 months ago                 myubdocker
8db925a214ff   new_dockerfile     "/bin/sh -c 'apachec…"     24 months ago      Up Les
s than a second                    hopeful_babbage
25becf097971   new_dockerfile     "/bin/sh -c 'apachec…"     24 months ago      Exited
 (127) 24 months ago               unruffled_bartik
4f1d89ade3f2   72300a873c2c       "/bin/bash"                24 months ago      Exited
```



```
lab309-1@lab309-1:~$ sudo docker exec -it 6716edfc99c1 bash
root@6716edfc99c1:/# exit
exit
lab309-1@lab309-1:~$ sudo docker stop 6716edfc99c1
6716edfc99c1
```



```
lab309-1@lab309-1:~$ sudo docker rm ac974915fc9a
ac974915fc9a
lab309-1@lab309-1:~$ sudo docker images
REPOSITORY        TAG         IMAGE ID        CREATED          SIZE
new_ubuntu        latest      b62931d33d3b    6 days ago       72.8MB
ubuntu            latest      54c9d81cbb44    3 weeks ago      72.8MB
dockerfile        latest      2c473978044b    24 months ago    189MB
new_dockerfile    latest      e3d0cfb688ca    24 months ago    189MB
ubuntu            <none>      72300a873c2c    2 years ago      64.2MB
hello-world       latest      fce289e99eb9    3 years ago      1.84kB
```

Conclusion: Successfully installed and configured Docker for creating Containers of different Operating System Images.