

Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

© CERTIFICATE ©

Certify that Mr./Miss Altaf Alam
of IT Department, Semester V with
Roll No. 02 has completed a course of the necessary
experiments in the subject Devops Lab under my
supervision in the **Thadomal Shahani Engineering College**
Laboratory in the year 20 23 - 20 24

Teacher In-Charge

Head of the Department

Sondal
20/10/23

Date 20/10/23

Principal

CONTENTS

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

Experiment No-1

Aim:- To study Devops: principles, practices & DevOps engineer roles & responsibilities.

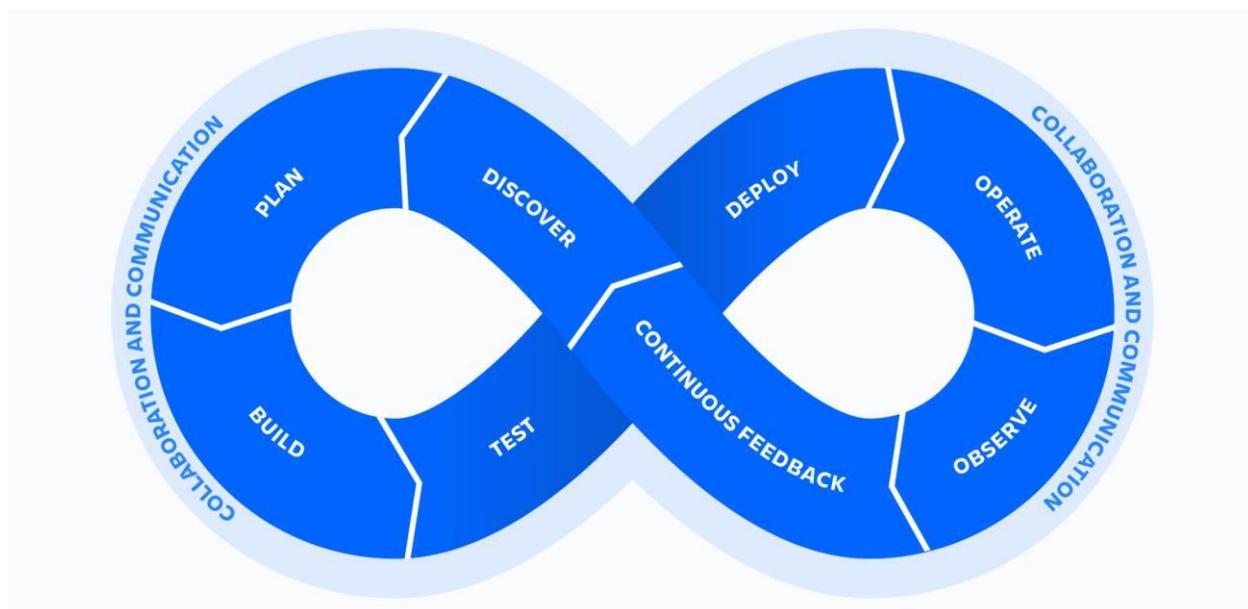
Lab Outcome:- LO1 : To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

Theory:-

What is Devops?

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

Lifecycle:



DevOps is the next evolution of agile methodologies. Because of the continuous nature of DevOps, practitioners use the infinity loop to show how the phases of the DevOps lifecycle relate to each other. Despite appearing to flow sequentially, the loop symbolizes the need for constant collaboration and iterative improvement throughout the entire lifecycle. A cultural shift that brings development and operations teams together. DevOps is a practice that involves a cultural change, new management principles, and technology tools that help to implement best practices.

Regardless of the type of DevOps toolchain an organization uses, a DevOps process needs to use the right tools to address the key phases of the DevOps lifecycle:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

- Discover
- Plan
- Build
- Test
- Monitor
- Operate
- Continuous feedback

Discover:

In the Discover phase, a DevOps team researches and defines the scope of a project. In particular, it involves activities such as user research, establishing goals, and defining success. Tools like Mural and Miro empower the entire software team to gather ideas and conduct research. Jira Product Discovery organizes this information into actionable inputs and prioritizes actions for development teams. As you're prioritizing, you'll also need to keep your backlog of user feedback in mind.

Plan:

Taking a page out of the agile handbook, we recommend tools that allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments. This allows you to learn from users sooner and helps with optimizing a product based on the feedback. Look for tools that provide sprint planning, issue tracking, and allow collaboration, such as Jira.

Build:

While Puppet and Chef primarily benefit operations, developers use open source tools like Kubernetes and Docker to provision individual development environments. Coding against virtual, disposable replicas of production helps you get more work done.

Continuous Delivery:

Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. That way, you automatically detect problems early, fix them when they're easiest to fix, and roll out new features to your users as early as possible. Look for tools that automatically apply your tests to development branches, and give you the option to push to main when branch builds are successful. Along with that, you get continuous feedback through real-time chat alerts from your team with a simple integration.

Test:

Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration. However, for the DevOps toolchain, automation is an essential function. Automated testing pays off over time by speeding up your development and testing cycles in the long run. And in a DevOps environment, it's important for another reason: awareness. Test automation can increase software quality and reduce risk by doing it early and often. Development teams can execute automated tests repeatedly, covering several areas such

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

as UI testing, security scanning, or load testing. They also yield reports and trend graphs that help identify risky areas.

Deploy:

One of the most stressful parts of shipping software is getting all the change, test, and deployment information for an upcoming release into one place. The last thing anyone needs before a release is a long meeting to report on status.

Automated deployment:

There's no magic recipe for automated deployment that will work for every application and IT environment. But converting operations' runbook into a cmd-executable script using Ruby or bash is a common way to start.

Operate:

The keys to unlocking collaboration between DevOps teams is making sure they're viewing the same work. What happens when incidents are reported? Are they linked and traceable to software problems? When changes are made, are they linked to releases?

Nothing blocks Dev's collaboration with Ops more than having incidents and software development projects tracked in different systems. Look for tools that keep incidents, changes, problems, and software projects on one platform so you can identify and fix problems faster.

Observe:

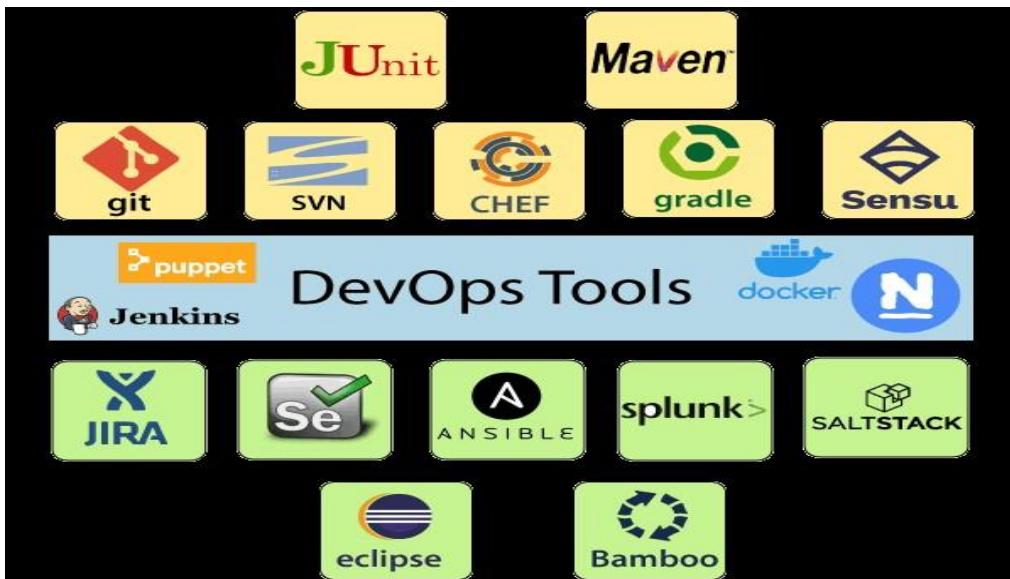
There are two types of monitoring that should be automated: server monitoring and application performance monitoring.

Manually "topping" a box or hitting your API with a test is fine for spot-checking. But to understand trends and the overall health of your application (and environments), you need software that is listening and recording data 24/7. Ongoing observability is a key capability for successful DevOps teams.

Continuous Feedback:

Customers are already telling you whether you've built the right thing – you just have to listen. Continuous feedback includes both the culture and processes to collect feedback regularly, and tools to drive insights from the feedback. Continuous feedback practices include collecting and reviewing NPS data, churn surveys, bug reports, support tickets, and even tweets. In a DevOps culture, everyone on the product team has access to user comments because they help guide everything from release planning to exploratory testing sessions.

DevOps tools:



1) Puppet

Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

2) Ansible

Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity. Ansible is easy to deploy because it does not use any agents or custom security infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

3) Docker

Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

4) Nagios

Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.

5) CHEF

A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

6) Jenkins

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

7) Git

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed, efficiency and co-ordinate the work and track the work among programmers. It is used as a critical distributed version control for the DevOps tool.

8) SALTSTACK

Stackify is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data center.

9) Splunk

Splunk is a tool to make machine data usable, accessible, and valuable to everyone. It delivers operational intelligence to DevOps teams. It helps companies to be more secure, productive, and competitive.

10) Selenium

Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests.

What are the roles and responsibilities of a Devops engineer?

Responsibilities of devops engineer include:

1. Infrastructure Provisioning and Management: Setting up and managing infrastructure, including servers, networks, databases, and cloud services. This involves using infrastructure-as-code tools like Terraform or CloudFormation.
2. Continuous Integration and Continuous Deployment (CI/CD): Implementing and maintaining CI/CD pipelines to automate the build, testing, and deployment processes. This ensures fast and reliable delivery of software.
3. Version Control and Configuration Management: Managing version control systems like Git and ensuring the consistency of configuration files across environments.
4. Automation: Automating repetitive tasks and processes to increase efficiency and reduce manual intervention. This includes tasks like environment provisioning, deployment, and monitoring.
5. Monitoring and Logging: Setting up monitoring and logging tools to track the performance, health, and availability of applications and infrastructure. This helps in detecting issues and troubleshooting problems proactively.
6. Security and Compliance: Ensuring that the systems and applications are secure by implementing security measures and best practices. Also, ensuring compliance with relevant regulations and standards.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

7. Collaboration and Communication: Working closely with development, operations, and other cross-functional teams to foster collaboration and improve communication.
8. Incident Management and Troubleshooting: Responding to incidents, identifying root causes, and resolving issues quickly to minimize downtime.
9. Performance Optimization: Optimizing the performance of applications and infrastructure to achieve scalability and efficiency.
10. Disaster Recovery and High Availability: Implementing disaster recovery strategies and ensuring high availability of critical systems.
11. Capacity Planning: Estimating future resource requirements based on growth and usage patterns to avoid resource shortages.
12. Documentation: Maintaining clear and up-to-date documentation for processes, configurations, and infrastructure.
13. Continuous Learning: Staying updated with the latest DevOps tools, practices, and trends to continuously improve and enhance processes.

Roles of devops engineer include:

1. DevOps Evangelist – The principal officer (leader) responsible for implementing DevOps.
2. Release Manager – The one releasing new features & ensuring post-release product stability.
3. Automation Expert – The guy responsible for achieving automation & orchestration of tools.
4. Software Developer/ Tester – The one who develops the code and tests it.
5. Quality Assurance – The one who ensures the quality of the product confirms to its requirement.
6. Security Engineer – The one always monitoring the product's security & health.

Conclusion:-

I have successfully learnt about devops, roles and responsibilities of devops engineer and also learnt about the various phases and tools used in devops.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

Experiment No 2

Aim : To understand Version Control System, Git installation & Github account.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO2 : To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and Github.

Theory :

What is Version Control?

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

What is Git?

Git is a specific open-source version control system created by Linus Torvalds in 2005. Specifically, Git is a distributed version control system, which means that the entire codebase and history is available on every developer's computer, which allows for easy branching and merging.

Instead, version control lets developers safely work through branching and merging. With branching, a developer duplicates part of the source code (called the repository). The developer can then safely make changes to that part of the code without affecting the rest of the project.

Then, once the developer gets his or her part of the code working properly, he or she can merge that code back into the main source code to make it official.

To understand GitHub, you must first have an understanding of Git. Git is an open-source version control system that was started by Linus Torvalds—the same person who created Linux. Git is similar to other version control systems—Subversion, CVS, and Mercurial to name a few.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

So, Git is a version control system, but what does that mean? When developers create something (an app, for example), they make constant changes to the code, releasing new versions up to and after the first official (non-beta) release.

Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

Similarly, people who have nothing to do with the development of a project can still download the files and use them. Most Linux users should be familiar with this process, as using Git, Subversion, or some other similar method is pretty common for downloading needed files—especially in preparation for compiling a program from source code (a rather common practice for Linux geeks).

Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better.

Git Installation in Windows:

1. Browse to the official Git website: <https://git-scm.com/downloads> 2. Click the download link for Windows and allow the download to complete.

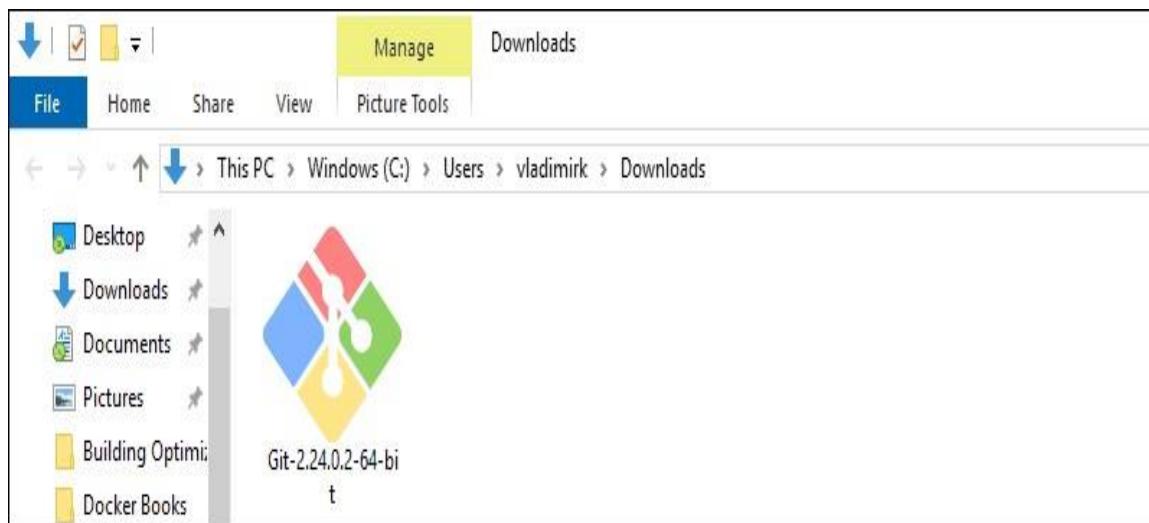


Extract and Launch Git Installer

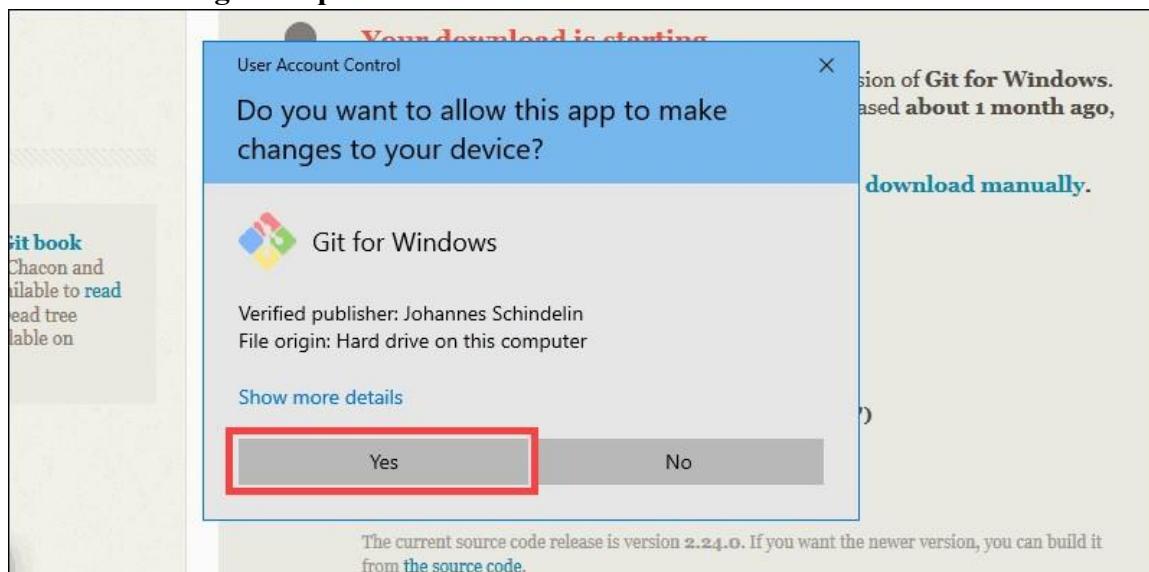
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

3. Browse to the download location (or use the download shortcut in your browser).
Double-click the file to extract and launch the installer.



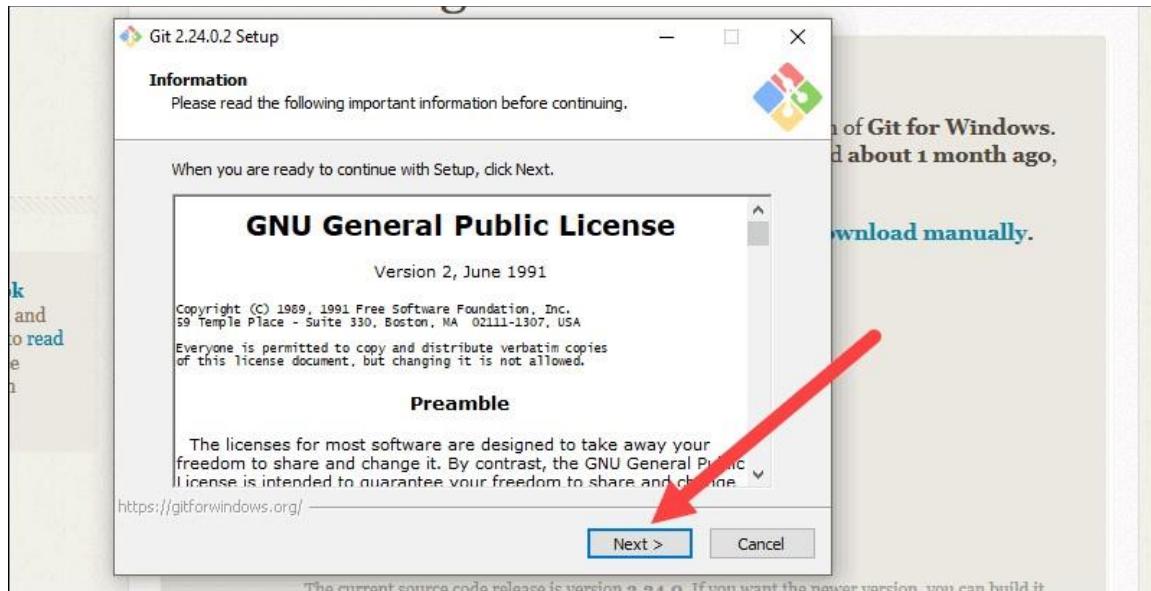
4. Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.



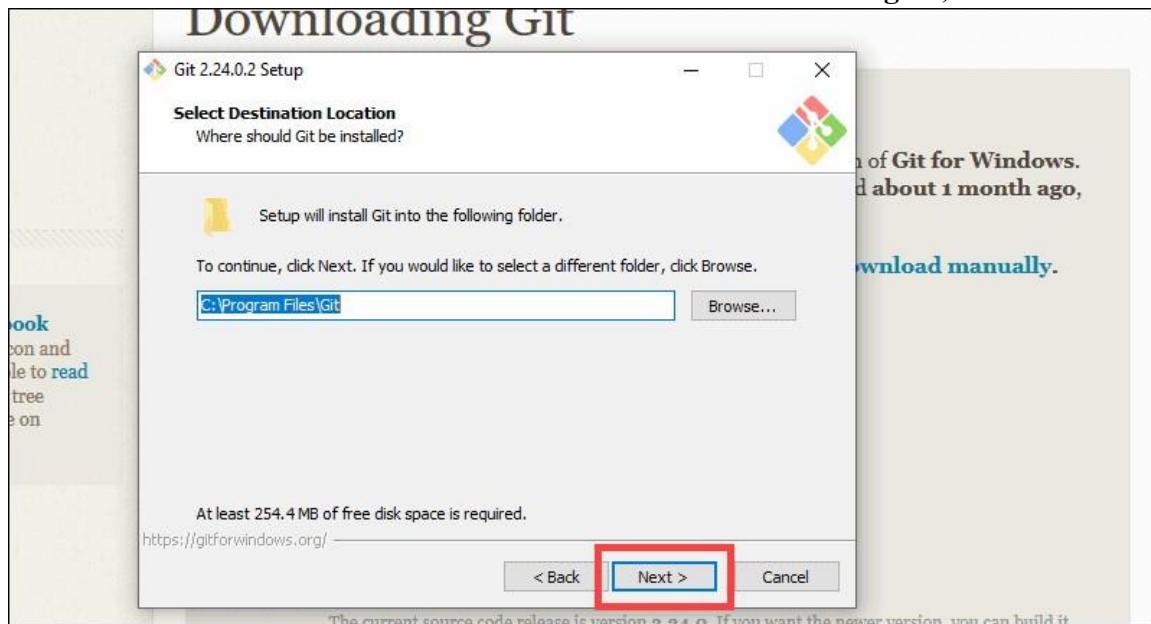
5. Review the GNU General Public License, and when you're ready to install, click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



- 6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click Next.**



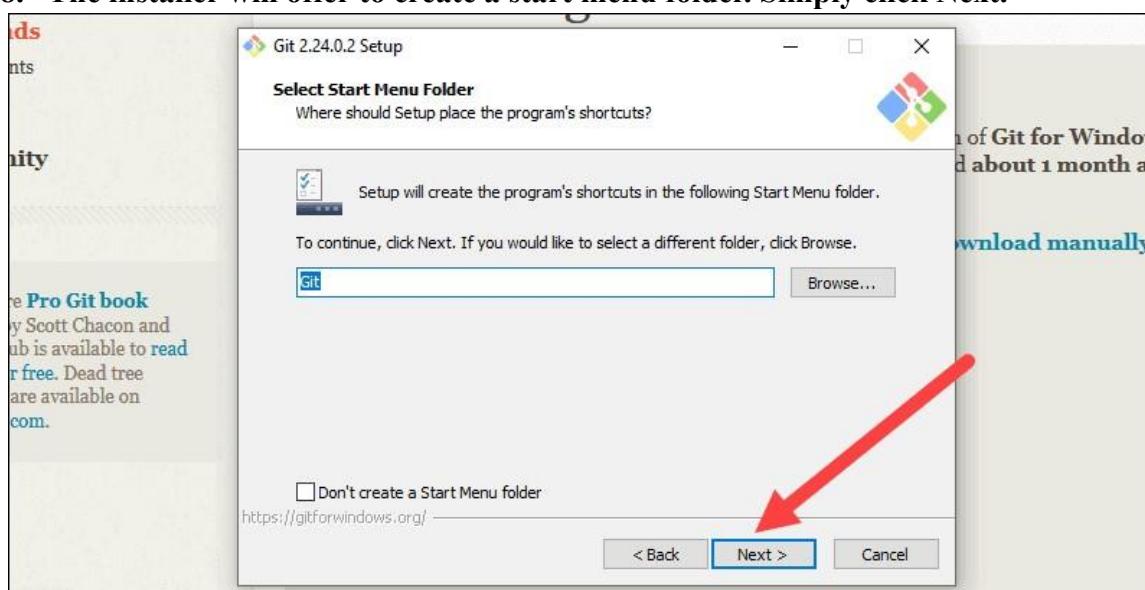
- 7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click next.**

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



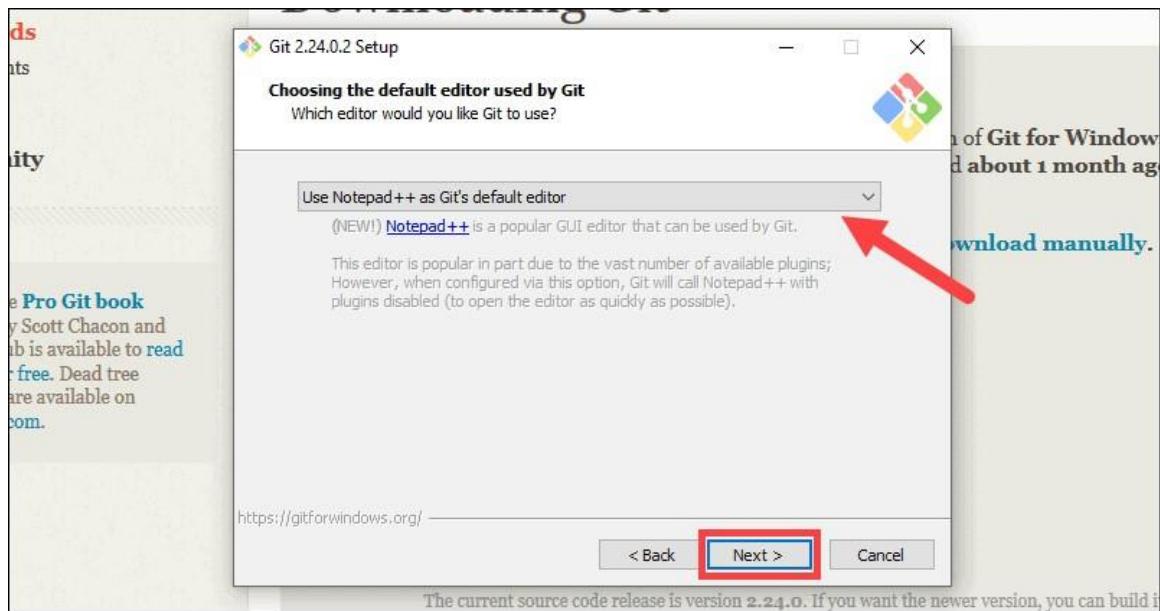
8. The installer will offer to create a start menu folder. Simply click Next.



9. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



10. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click Next.



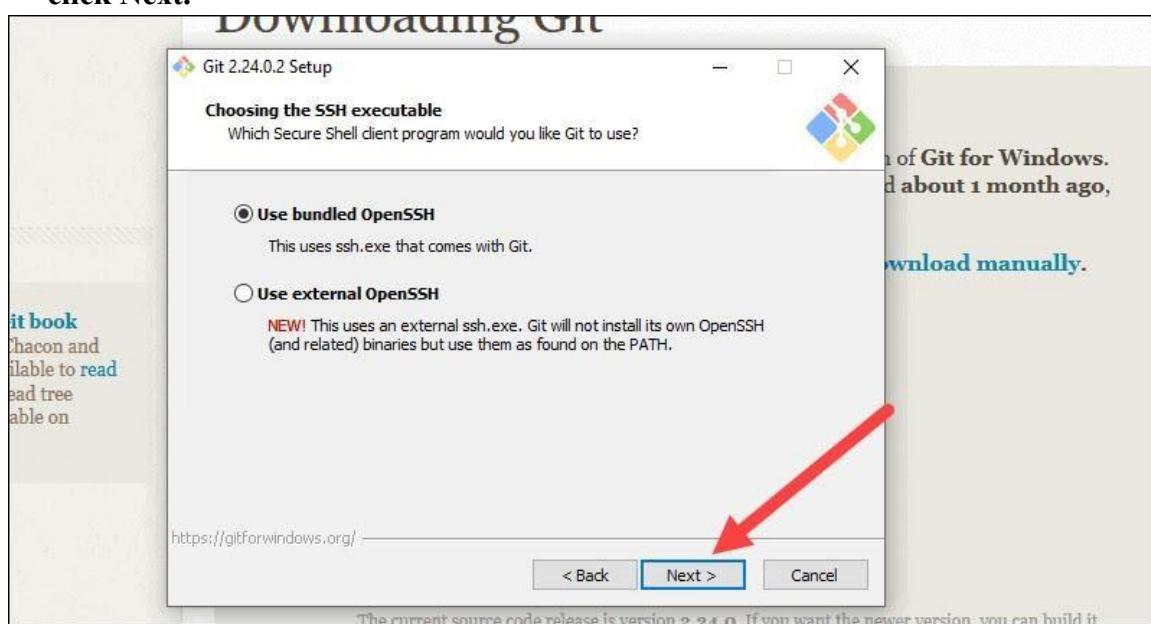
11. This installation step allows you to change the PATH environment. The PATH is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



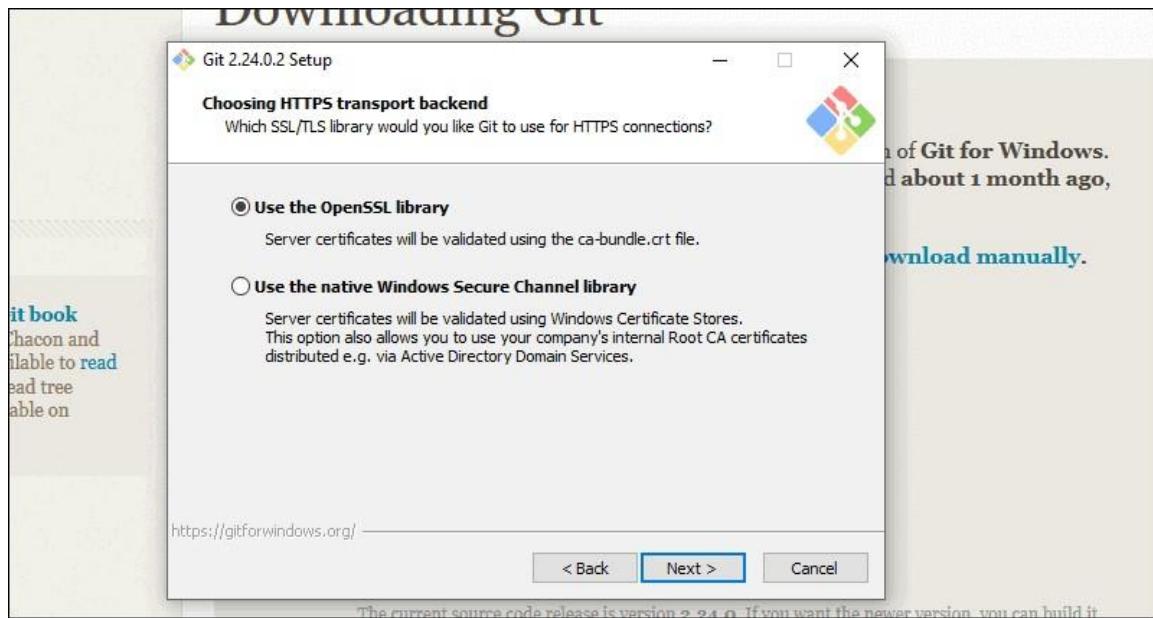
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click Next.



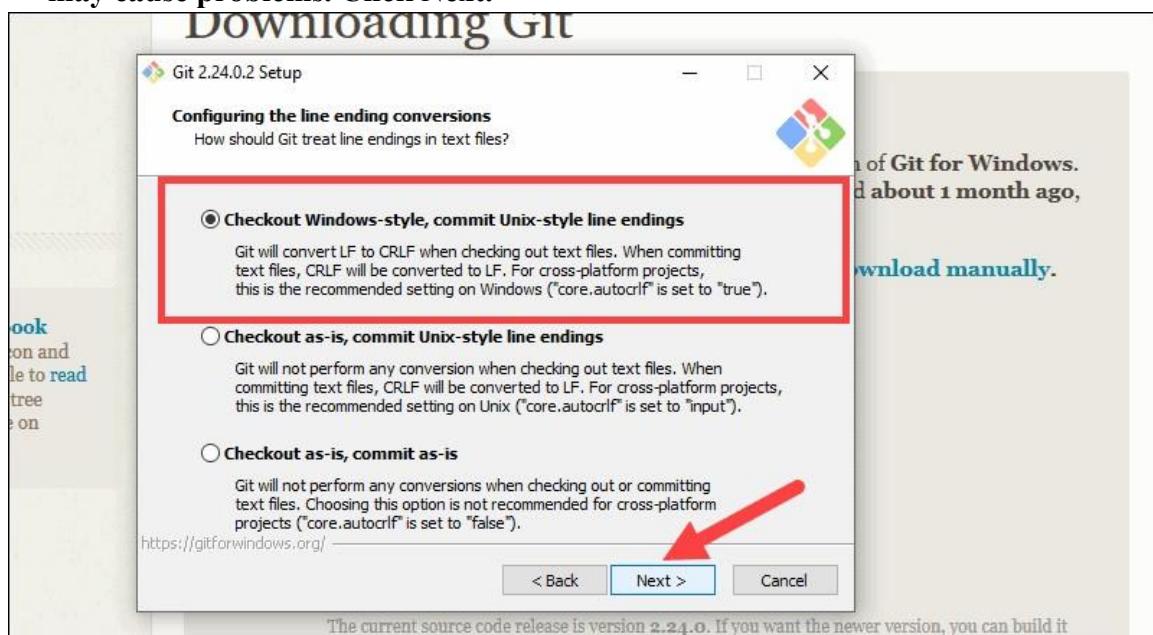
13. The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click Next.



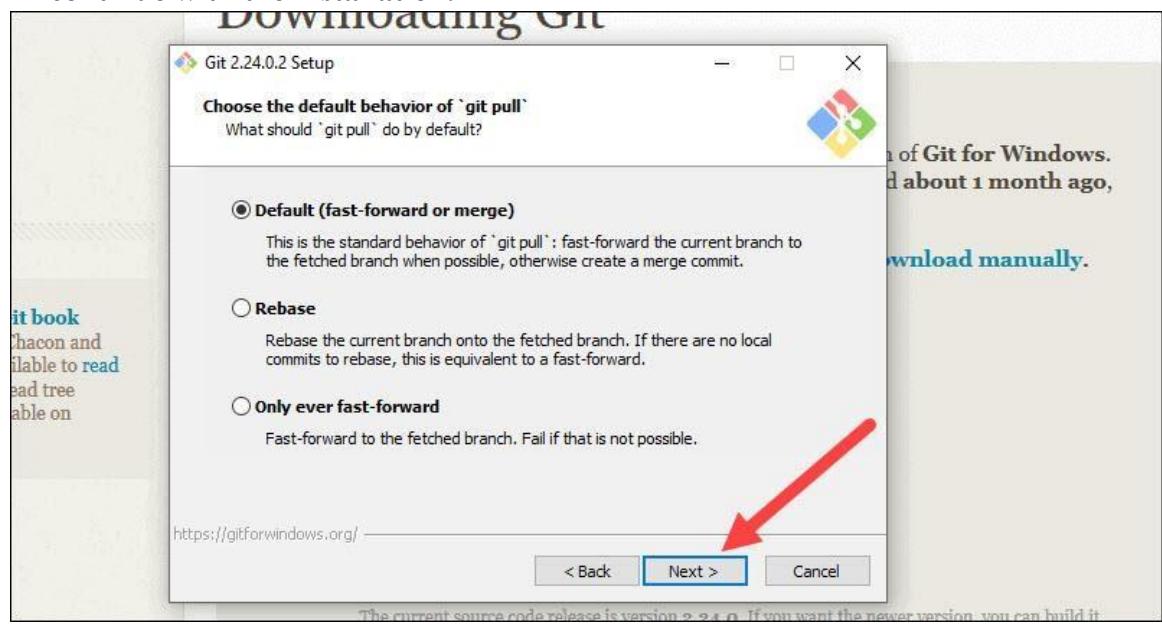
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



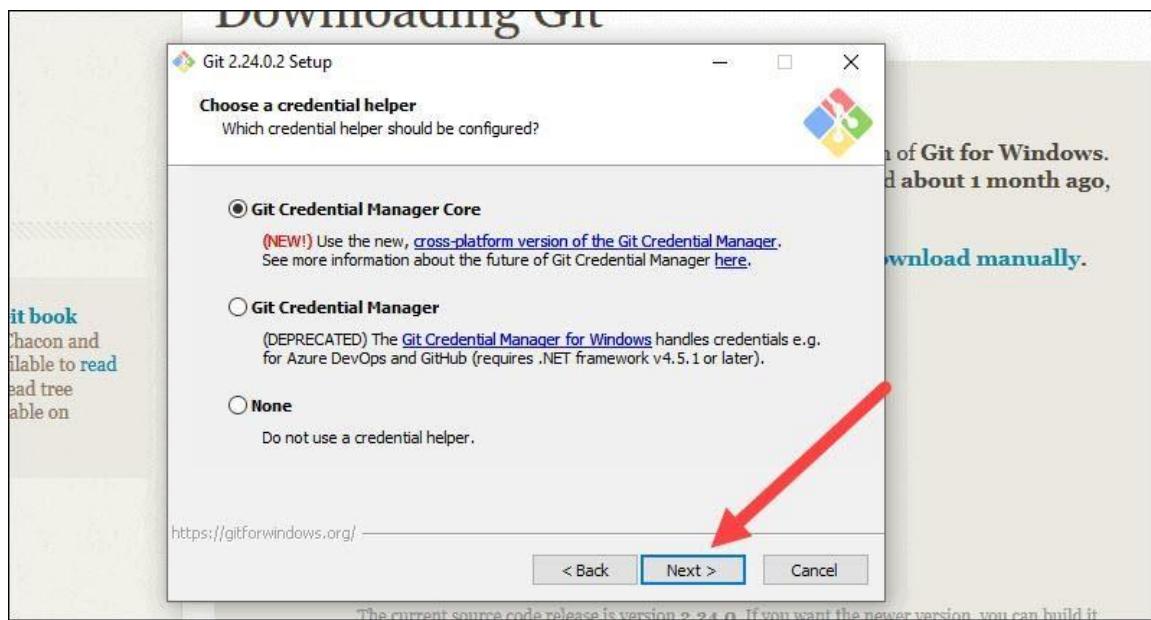
16. The installer now asks what the git pull command should do. The default option is recommended unless you specifically need to change its behavior. Click Next to continue with the installation.



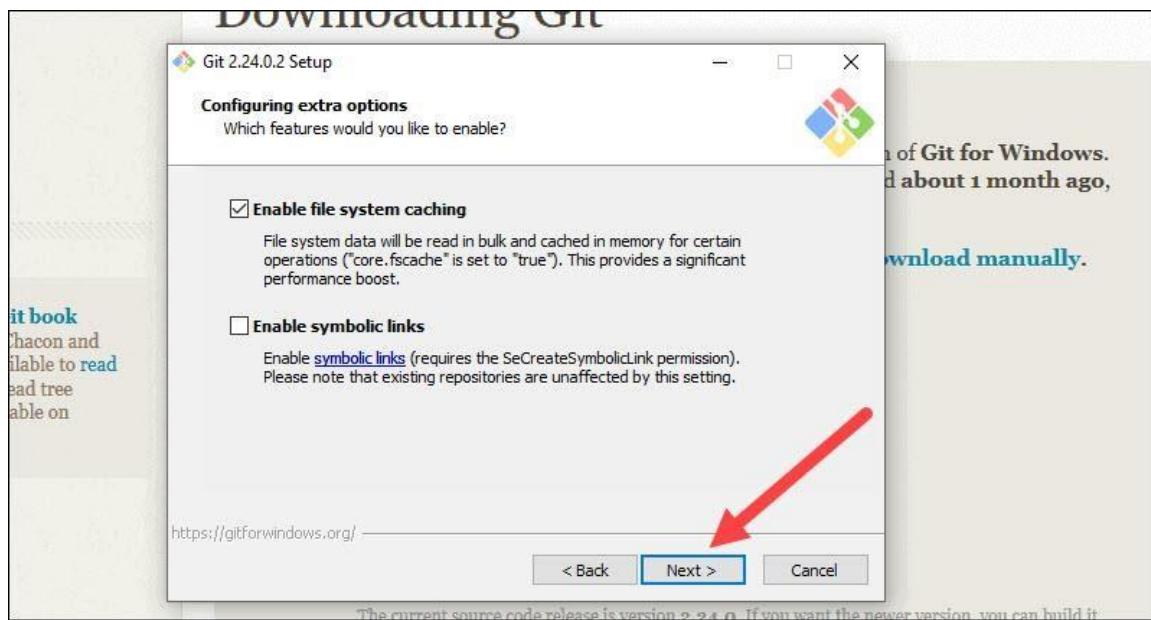
17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click Next.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



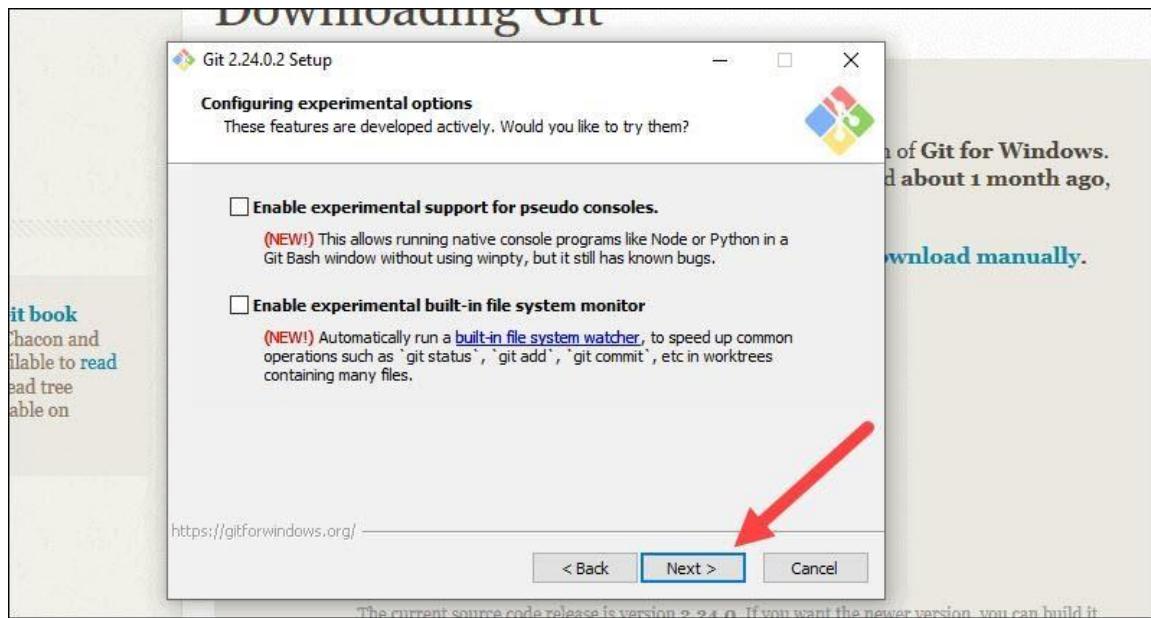
18. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcut



19. Depending on the version of Git you're installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click Install.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23



20. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish.



What is Github?

GitHub is a web-based version control and collaboration platform for software developers. Microsoft, the biggest single contributor to GitHub, acquired the platform for \$7.5 billion in 2018. GitHub, which is delivered through a software as a service (SaaS) business model, was started in 2008. It was founded on Git, an open source code management system created by Linus Torvalds to make software builds faster.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 18/07/23

Altafalalm3 / README.md

Hi 🤝, I'm Altaf Alam 🚀

A FullStack Web developer, Python Developer

Profile views 3,031

FOLLOW @ALTAFO032

- I'm currently learning React + Django & DSA with cpp. Interested in Web dev and python .
- I'm looking for help with Backend Web Development and CP

Connect with me:

Languages and Tools:

Edit profile

GitHub allows developers to change, adapt and improve software from its public repositories for free, but it charges for private repositories, offering various paid plans. Each public and private repository contains all of a project's files, as well as each file's revision history. Repositories can have multiple collaborators and can be either public or private.

GitHub facilitates social coding by providing a hosting service and web interface for the Git code repository, as well as management tools for collaboration. The developer platform can be thought of as a social networking site for software developers. Members can follow each other, rate each other's work, receive updates for specific open source projects, and communicate publicly or privately.

Conclusion :

Thus we have understood about control version, installation of git and creation of github account.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

Experiment No 3

Aim : To perform various Git operations.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO2 : To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and Github.

Theory :

Git operations:

1. `git init`:

Initializes a new Git repository in the current directory, creating the necessary folders and files to track changes.

2. `git clone <repository>`:

Creates a copy of a remote repository on your local machine, allowing you to work with the code and contribute changes.

3. `git add <file>`: Stages a file, preparing it to be committed. Use this command before committing changes.

4. `git commit -m "<message>"`:

Commits the staged changes with a descriptive message, creating a snapshot of the code's current state.

5. `git status`: Shows the status of your repository, including changes that are staged or not yet staged.

6. `git diff`:

Displays the differences between the working directory and the last committed version, helping you see the changes you've made.

7. `git log`:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

Shows a chronological list of commits in the repository, providing information about authors, dates, and commit messages.

8. `git pull`:

Fetches changes from a remote repository and merges them into your current branch, ensuring you have the latest updates.

9. `git push`: Sends your committed changes to a remote repository, updating it with your new code.

10. `git branch`:

Lists all branches in the repository and highlights the branch you're currently on. Use "git branch <name>" to create a new branch.

11. `git checkout <branch>`: Switches to a different branch, allowing you to work on a different part of the codebase.

12. `git merge <branch>`:

Integrates changes from one branch into another, combining the code and resolving conflicts if needed.

13. `git fetch`:

Retrieves changes from a remote repository without automatically merging them, allowing you to review before merging.

14. `git stash`:

Temporarily saves changes that you don't want to commit yet, allowing you to switch branches or pull changes without committing everything.

15. `git reset <file>`: Unstages a file that you previously staged using "git add," reversing the staging process.

16. `git remote -v`: Lists all remote repositories associated with your local repository, showing their URLs.

17. `git rm <file>`: Removes a file from the working directory and stages its deletion for the next commit.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

18. git config: Configures Git settings such as your name, email, and preferred text editor.

19. git remote add <name> <url>:

Adds a new remote repository with a given name and URL. This enables you to fetch and push changes to/from this remote.

20. git remote remove <name>:

Removes a remote repository from your local repository.

21. git remote show <name>: Provides information about a specific remote repository, including fetch and push URLs.

22. git remote rename <old-name> <new-name>: Renames a remote repository from an old name to a new one.

23. git remote set-url <name> <new-url>: Changes the URL of a remote repository.

24. git tag:

Lists existing tags in the repository. Tags are used to mark specific points in the commit history.

25. git tag -a <tag-name> -m "<message>": Creates an annotated tag with a message.

Tags are often used to mark releases.

26. git tag -d <tag-name>: Deletes a local tag.

27. git push <remote> --tags: Pushes tags to a remote repository, making them available to others.

28. git fetch <remote>:

Retrieves changes from a specific remote repository without automatically merging them. It's often followed by a "git merge" to integrate the changes.

29. git merge <branch> --no-ff:

Performs a merge with a specified branch, creating a new merge commit even if the changes could be fast-forwarded.

30. git rebase <branch>:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

Moves your local commits to the tip of a specified branch, resulting in a linear commit history. Use with caution when working with shared branches.

31. git cherry-pick \<commit>: Applies the changes of a specific commit to your current branch.

32. git revert \<commit>: Creates a new commit that undoes the changes introduced by a specified commit.

33. git bisect:

Helps you find the commit that introduced a bug by performing a binary search through the commit history.

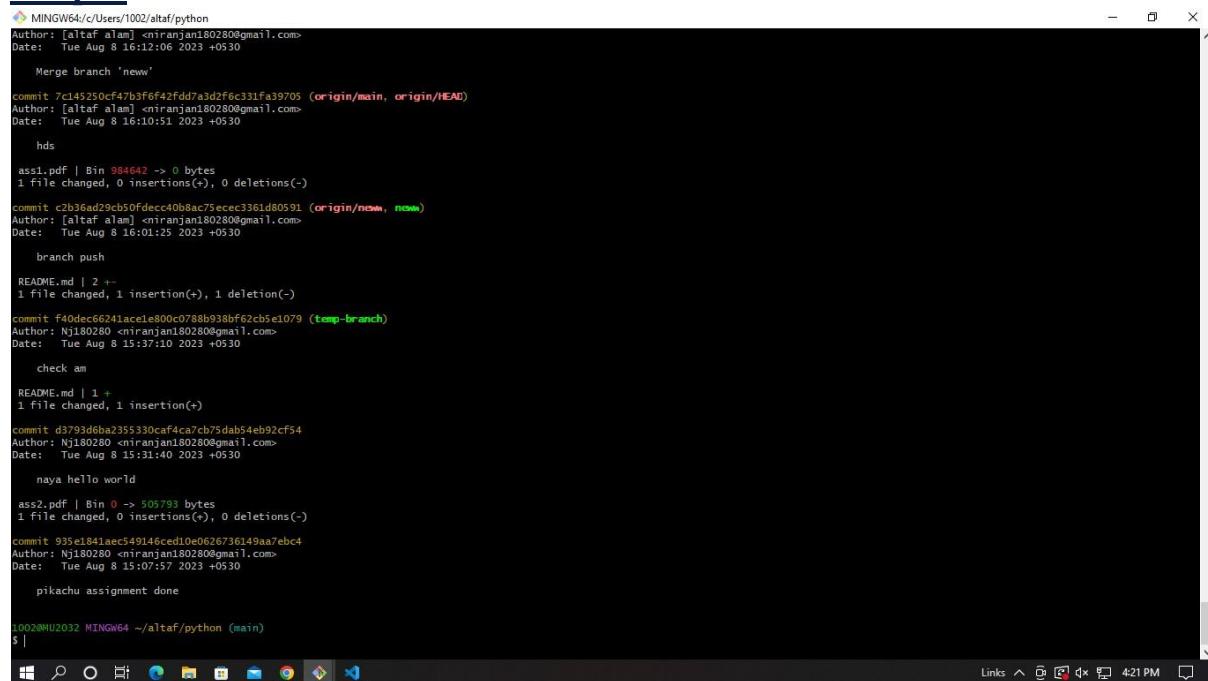
34. git show \<commit>: Displays detailed information about a specific commit, including the changes introduced.

35. git clean -n:

Shows a preview of untracked files and directories that can be removed using "git clean f."

36. git config --global alias.\<shortcut> \<command>: Creates a custom alias to simplify frequently used Git commands.

Output :



```
MINGW64:/c/Users/1002/altaf/python
Author: [altaf alam] <irranjan180280@gmail.com>
Date: Tue Aug 8 16:12:06 2023 +0530
Merge branch 'new'
commit 7c149250cf47b5f6f42fdd7a3d2f6c331fa39705 (origin/main, origin/HEAD)
Author: [altaf alam] <irranjan180280@gmail.com>
Date: Tue Aug 8 16:10:51 2023 +0530
ass1.pdf | Bin 004642 -> 0 bytes
1 file changed, 0 insertions(+), 0 deletions(-)

commit c2b34ad29cb50fdcc40b8ac75ecce3361d80591 (origin/new, new)
Author: [altaf alam] <irranjan180280@gmail.com>
Date: Tue Aug 8 16:01:25 2023 +0530
branch push
README.md | 2 ++
1 file changed, 1 insertion(+), 1 deletion(-)

commit f40dec6241acc1800c0738b938bf62cb5e1079 (temp-branch)
Author: Nj180280 <irranjan180280@gmail.com>
Date: Tue Aug 8 15:37:10 2023 +0530
check an
README.md | 1 +
1 file changed, 1 insertion(+)

commit d3793d6ba2355330ca4ca7cb75dab54eb92cf54
Author: Nj180280 <irranjan180280@gmail.com>
Date: Tue Aug 8 15:31:40 2023 +0530
naya hello world

ass2.pdf | Bin 0 -> 505793 bytes
1 file changed, 0 insertions(+), 0 deletions(-)

commit 935e1841aec49146ced10e0626736149a7ebc4
Author: Nj180280 <irranjan180280@gmail.com>
Date: Tue Aug 8 15:07:57 2023 +0530
pikachu assignment done

1002@NU2032 MINGW64 ~/altaf/python (main)
$ |
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

```
MINGW64:/c/Users/1002/altaf/python
$ Already on 'main'
Your branch is up to date with 'origin/main'.

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git merge [new]
merge: [new] - not something we can merge

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git merge new
Merge made by the 'ort' strategy.
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git log --oneline -M
commit a9241aaa989bf7e08cf063az9c4243666339016 (HEAD -> main)
Merge: 7c14525 c2b36ad
Author: [altaf alam] <riranjan180280@gmail.com>
Date:  Tue Aug 8 16:10:51 2023 +0530

  Merge branch 'new'

commit 7c145250cf47b3f6f42fdd7a3d2f6c331fa39705 (origin/main, origin/HEAD)
Author: [altaf alam] <riranjan180280@gmail.com>
Date:  Tue Aug 8 16:10:51 2023 +0530

  hds

  ass1.pdf | Bin 984642 -> 0 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)

commit c2b36ad29cb50fdec40b8ac75ec3361d80591 (origin/new, new)
Author: [altaf alam] <riranjan180280@gmail.com>
Date:  Tue Aug 8 16:01:25 2023 +0530

  branch push

README.md | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)

commit f40dec66241ace1e800c0788b938bf62cb5e1079 (temp-branch)
Author: NJ180280 <riranjan180280@gmail.com>
Date:  Tue Aug 8 15:37:10 2023 +0530

  check am

README.md | 1 +
 1 file changed, 1 insertion(+)

commit d3793d6ba2355330caf4ca7cb75dab54eb92cf54

MINGW64:/c/Users/1002/altaf/python
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:  ass1.pdf

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git push
Everything up-to-date

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git push -u origin main
Everything up-to-date
branch 'main' set up to track 'origin/main'.

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git pull
Already up to date.

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git add .

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git commit -m "hds"
[main 7c14525] hds
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 ass1.pdf

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 225 bytes | 225.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1); completed with 1 local object.
To https://github.com/AltafAlam5/Python.git
  f40dec6..7c14525  main -> main

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git merge [new]
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

The image shows three vertically stacked terminal windows on a Windows desktop. Each window has a title bar indicating the path as 'MINGW64:/c/Users/1002/altaf/python'. The first window shows a series of commits from August 8, 2023, including changes to README.md, a file named 'ass1.pdf' (which was deleted), and a file named 'Variables.py'. The second window shows a commit on November 8, 2022, adding programs. The third window shows a merge commit on February 25, 2023, merging the 'main' branch from the repository at https://github.com/Altafalalam3/PythonBasics into the current branch.

```
added

commit ed7b9362015da76548fb0fa69c03675680d053e2
Author: Altaf Alam <102013452+Altafalalam3@users.noreply.github.com>
Date:   Fri Jan 27 15:03:13 2023 +0530

Delete Python directory

10020MU2032 MINGW64 ~/altaf/python (main)
$ git log main..new
commit c2b3ad29cb0fdcc40b8ac75ecce3361d80591 (origin/new, new)
Author: [altaf alam] <iranganjan180280@gmail.com>
Date:   Tue Aug 8 16:01:25 2023 +0530

branch push

10020MU2032 MINGW64 ~/altaf/python (main)
$ git diff main..new
diff --git a/README.md b/README.md
index fa19236..9770027 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,2 @@
 # PythonBasics
 #
 \\ No newline at end of file
-Hellooooooooooooooo bachchooooooo
\\ No newline at end of file

10020MU2032 MINGW64 ~/altaf/python (main)
$ git diff new..main

10020MU2032 MINGW64 ~/altaf/python (main)
$ git show [SHA]

10020MU2032 MINGW64 ~/altaf/python (main)
$ git rm ass1.pdf
rm 'ass1.pdf'

10020MU2032 MINGW64 ~/altaf/python (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    ass1.pdf

MINGW64:/c/Users/1002/altaf/python
Commit 17c38502acefd3e2a32839527adc65b25d8a2c5
Author: Altaf Alam <102013452+Altafalalam3@users.noreply.github.com>
Date:   Tue Nov 8 18:55:46 2022 +0530

  update _Variables.py

commit a7a19567248cb76fbffff392f810d1390d064d3fd
Author: Altafalalam3 <altaf.alam0032@gmail.com>
Date:   Tue Nov 8 13:43:44 2022 +0530

  added programs

10020MU2032 MINGW64 ~/altaf/python (main)
$ git log main..new
commit c2b3ad29cb0fdcc40b8ac75ecce3361d80591 (origin/new, new)
Author: [altaf alam] <iranganjan180280@gmail.com>
Date:   Tue Aug 8 16:01:25 2023 +0530

  branch push

commit f40dec66241ace1e800c0788b938bf62cb5e1079 (HEAD -> main, origin/main, origin/HEAD, temp-branch)
Author: Nj180280 <iranganjan180280@gmail.com>
Date:   Tue Aug 8 15:37:10 2023 +0530

  check am

commit d3793d6ba235330cafc4a7cb75dab54eb92cf54
Author: Nj180280 <iranganjan180280@gmail.com>
Date:   Tue Aug 8 15:31:40 2023 +0530

  naya hello world

commit 935e1841aec549146ced10e0626736149aa7ebc4
Author: Nj180280 <iranganjan180280@gmail.com>
Date:   Tue Aug 8 15:07:57 2023 +0530

  pikachu assignment done

commit 6e4417570c12453ac3cd1fd2d27f62fd62321661 (upstream/main)
Author: Altaf Alam <102013452+Altafalalam3@users.noreply.github.com>
Date:   Tue Aug 8 15:01:58 2023 +0530

  Add files via upload

commit c3e68939497b845bffa554da000d1e872e975d2
Merge: e8ac134 ed79936
Author: Altafalalam3 <altaf.alam0032@gmail.com>
Date:   Sat Feb 25 18:24:28 2023 +0530

  Merge branch 'main' of https://github.com/Altafalalam3/PythonBasics
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

```
MINGW64:/c/Users/1002/altaf/python
$ git log neww
commit c2b3ad29cb50fdecc40b8ac75eccc3361d80591 (origin/neww, neww)
Author: [altaf alam] <iranjan180280@gmail.com>
Date: Tue Aug 8 16:01:25 2023 +0530
  branch push

commit f40dec6241ace1800c0788b938bf62cb5e1079 (HEAD -> main, origin/main, origin/HEAD, temp-branch)
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:37:10 2023 +0530
  check am

commit d3793d6ba2355330caf4ca7cb75dab54eb92cf54
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:31:40 2023 +0530
  naya hello world

commit 935e1841ec49146ced10e062673619aa7ebc4
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:07:57 2023 +0530
  pikachu assignment done

commit 6ed4417570c12493ac3cdf1d2d27f62fd62321661 (upstream/main)
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Tue Aug 8 15:01:58 2023 +0530
  Add files via upload

commit c3d689394979845fffc554da000die872e975d2
Merge: 88ac134 cd7b936
Author: Altafalal3 <altaf.alam0032@gmail.com>
Date: Sat Feb 25 18:24:28 2023 +0530
  Merge branch 'main' of https://github.com/Altafalal3/PythonBasics

commit e8ac134957814eccf7e69cf9d88249b217bd57d8
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Sat Feb 25 18:23:58 2023 +0530
  added

commit ed7b9362015da76548fb0fa69c03675680d053e2
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Fri Jan 27 15:03:13 2023 +0530
  Links: ^ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ 4:20 PM
```



```
MINGW64:/c/Users/1002/altaf/python
$ git commit -m "branch push"
[neww c2b3ad29] branch push
 1 file changed, 1 insertion(+), 1 deletion(-)

0028MU2032 MINGW64 ~/altaf/python (neww)
$ git push -u origin neww
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 295 bytes | 295.00 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 'neww' on GitHub by visiting:
remote:   https://github.com/Altafalal3/Python.git
remote: * [new branch]    neww -> neww
branch 'neww' set up to track 'origin/neww'.
0028MU2032 MINGW64 ~/altaf/python (neww)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
0028MU2032 MINGW64 ~/altaf/python (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
0028MU2032 MINGW64 ~/altaf/python (main)
$ git log neww
commit c2b3ad29cb50fdecc40b8ac75eccc3361d80591 (origin/neww, neww)
Author: [altaf alam] <iranjan180280@gmail.com>
Date: Tue Aug 8 16:01:25 2023 +0530
  branch push

commit f40dec6241ace1800c0788b938bf62cb5e1079 (HEAD -> main, origin/main, origin/HEAD, temp-branch)
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:37:10 2023 +0530
  check am

commit d3793d6ba2355330caf4ca7cb75dab54eb92cf54
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:31:40 2023 +0530
  naya hello world

commit 935e1841ec49146ced10e062673619aa7ebc4
Author: Nj180280 <iranjan180280@gmail.com>
Date: Tue Aug 8 15:07:57 2023 +0530
  pikachu assignment done

commit 6ed4417570c12493ac3cdf1d2d27f62fd62321661 (upstream/main)
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Tue Aug 8 15:01:58 2023 +0530
  Add files via upload

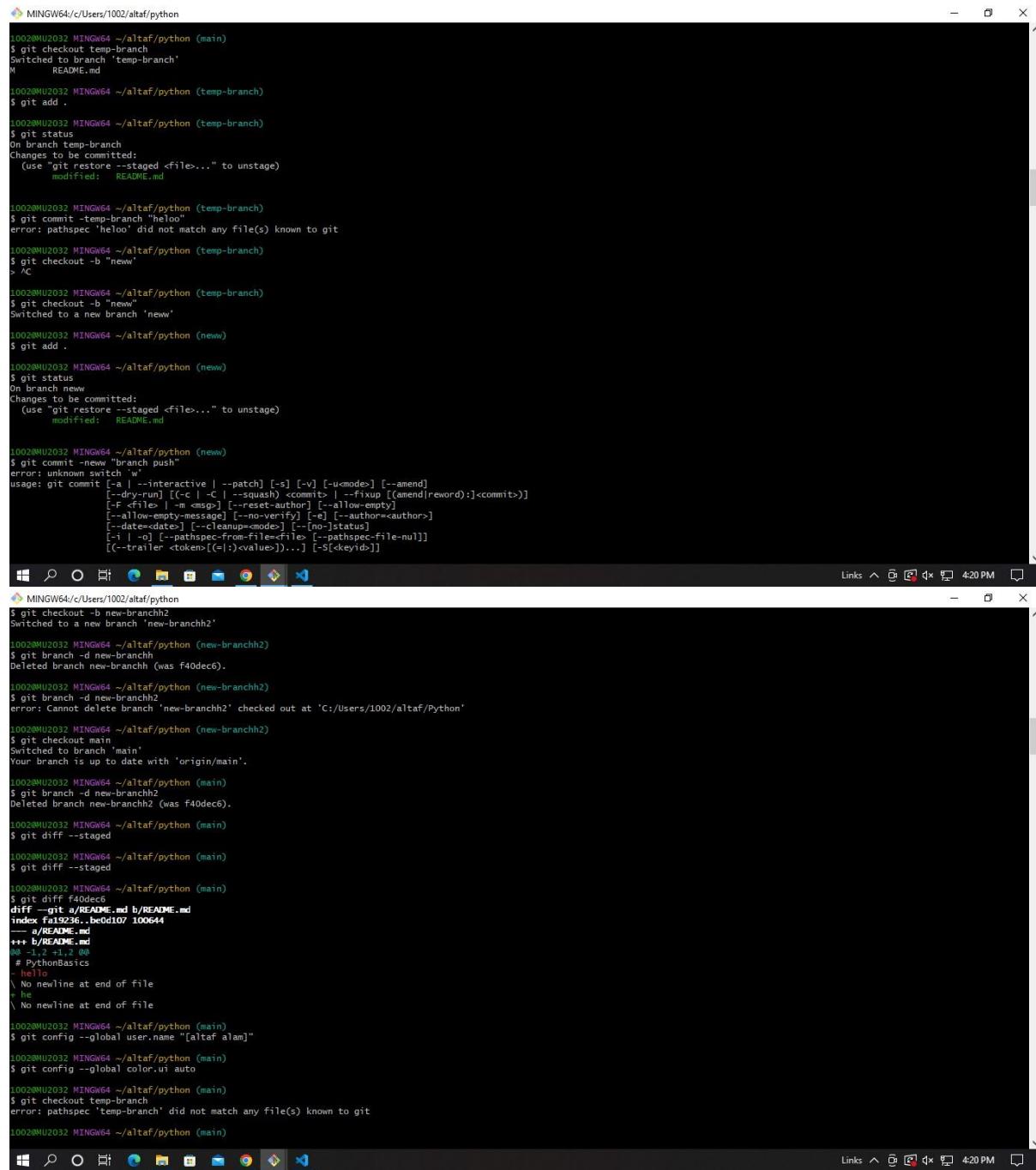
commit c3d689394979845fffc554da000die872e975d2
Merge: 88ac134 cd7b936
Author: Altafalal3 <altaf.alam0032@gmail.com>
Date: Sat Feb 25 18:24:28 2023 +0530
  Merge branch 'main' of https://github.com/Altafalal3/PythonBasics

commit e8ac134957814eccf7e69cf9d88249b217bd57d8
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Sat Feb 25 18:23:58 2023 +0530
  added

commit ed7b9362015da76548fb0fa69c03675680d053e2
Author: Altaf Alam <02013452+Altafalal3@users.noreply.github.com>
Date: Fri Jan 27 15:03:13 2023 +0530
  Links: ^ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ 4:20 PM
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23



The image shows three separate terminal windows side-by-side, all running on a Windows operating system (evident from the taskbar icons at the bottom). Each window has a dark background and white text. The windows are identical in content, displaying a sequence of Git commands and their outputs.

```
MINGW64/c/Users/1002/altaf/python
$ git checkout temp-branch
Switched to branch 'temp-branch'
M README.md

1002@MU2032 MINGW64 ~/altaf/python (temp-branch)
$ git add .
1002@MU2032 MINGW64 ~/altaf/python (temp-branch)
$ git status
On branch temp-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified: README.md

1002@MU2032 MINGW64 ~/altaf/python (temp-branch)
$ git commit -temp-branch "Hello"
error: pathspec 'heloo' did not match any file(s) known to git
1002@MU2032 MINGW64 ~/altaf/python (temp-branch)
$ git checkout -b 'neww'
> ^C
1002@MU2032 MINGW64 ~/altaf/python (temp-branch)
$ git checkout -b 'neww'
Switched to a new branch 'neww'
1002@MU2032 MINGW64 ~/altaf/python (neww)
$ git add .
1002@MU2032 MINGW64 ~/altaf/python (neww)
$ git status
On branch neww
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified: README.md

1002@MU2032 MINGW64 ~/altaf/python (neww)
$ git commit -neww "branch push"
error: unknown switch 'w'
usage: git commit [-a | -interactive | --patch] [-s] [-v] [-u<mode>] [--amend]
  [-dry-run] [[-c | -c | --squash] <commit>] [--fixup [[<amend>|reword>]:<commit>]]
  [-F <file>] [-m <msg>] [--reset-author] [--allow-empty]
  [--allow-empty-message] [--no-verify] [-e] [-author=<author>]
  [--date=<date>] [--cleanup=<mode>] [--no-status]
  [-i | -o] [--pathspec-from-file=<file>] [--pathspec-file-null]
  [--trailer <token>[=(=):<value>]]... [-S[<keyids>]]]

MINGW64/c/Users/1002/altaf/python
$ git checkout -b new-branchh2
Switched to a new branch 'new-branchh2'
1002@MU2032 MINGW64 ~/altaf/python (new-branchh2)
$ git branch -d new-branchh1
Deleted branch new-branchh1 (was f40dec6).
1002@MU2032 MINGW64 ~/altaf/python (new-branchh2)
$ git branch -d new-branchh2
error: Cannot delete branch 'new-branchh2' checked out at 'C:/Users/1002/altaf/Python'

MINGW64 ~/altaf/python (new-branchh2)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

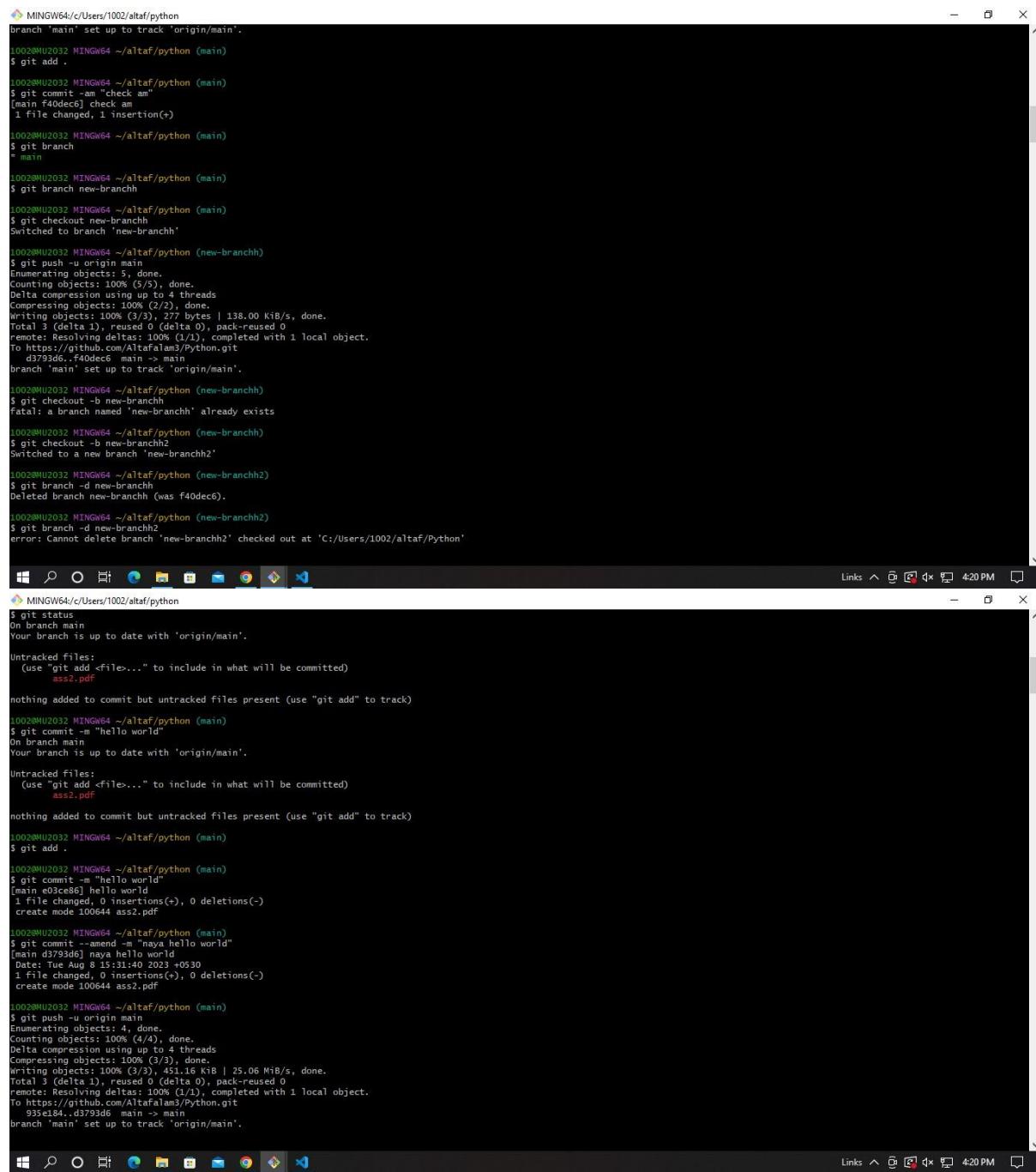
1002@MU2032 MINGW64 ~/altaf/python (main)
$ git branch -d new-branchh2
Deleted branch new-branchh2 (was f40dec6).

MINGW64 ~/altaf/python (main)
$ git diff --staged
1002@MU2032 MINGW64 ~/altaf/python (main)
$ git diff
diff --git a/README.ad b/README.md
index fa19236..be0d107 100644
--- a/README.ad
+++ b/README.md
@@ -1,1 +1,1 @@
 # PythonBasics
 - heloo
 \ No newline at end of file
 + he
 \ No newline at end of file

1002@MU2032 MINGW64 ~/altaf/python (main)
$ git config --global user.name "[altaf alam]"
1002@MU2032 MINGW64 ~/altaf/python (main)
$ git config --global color.ui auto
1002@MU2032 MINGW64 ~/altaf/python (main)
$ git checkout temp-branch
error: pathspec 'temp-branch' did not match any file(s) known to git
1002@MU2032 MINGW64 ~/altaf/python (main)
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23



```
MINGW64:/c/Users/1002/altaf/python
$ git add .
$ git commit -am "check am"
[main f40dec6] check am
1 file changed, 1 insertion(+)
$ git branch
* main
$ git branch new-branch
$ git checkout new-branch
Switched to branch 'new-branch'
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 277 bytes | 138.00 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.
To https://github.com/Altafalalm3/Python.git
 d3793d6..f40dec6 main -> main
branch 'main' set up to track 'origin/main'.

MINGW64:/c/Users/1002/altaf/python (new-branch)
$ git checkout -b new-branchh
Switched to a new branch 'new-branchh'

MINGW64:/c/Users/1002/altaf/python (new-branchh)
$ git branch -d new-branchh
Deleted branch new-branchh (was f40dec6).

MINGW64:/c/Users/1002/altaf/python (new-branchh2)
$ git branch -d new-branchh2
error: Cannot delete branch 'new-branchh2' checked out at 'C:/Users/1002/altaf/Python'

MINGW64:/c/Users/1002/altaf/python
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ass1.pdf

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

MINGW64:/c/Users/1002/altaf/python (main)
$ git commit -m "hello world"
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ass2.pdf

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

MINGW64:/c/Users/1002/altaf/python (main)
$ git add .
$ git commit -m "hello world"
[main e03ce86] hello world
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ass2.pdf

MINGW64:/c/Users/1002/altaf/python (main)
$ git commit -am "naya hello world"
[main d3793d6] naya hello world
Date: Tue Aug 8 15:31:40 2023 +0530
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ass2.pdf

MINGW64:/c/Users/1002/altaf/python (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 451.16 KiB | 25.06 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Altafalalm3/Python.git
 935e184..d3793d6 main -> main
branch 'main' set up to track 'origin/main'.
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

The image shows three separate terminal windows side-by-side, all running on a Windows operating system (evidenced by the taskbar at the bottom).

- Top Terminal:** Shows a series of Git commands being run against a repository named "altaf/python". It includes pushing changes to a remote branch, committing files, and updating the local repository.
- Middle Terminal:** Shows the user navigating through a directory structure under "c:/Users/1002/altaf/Python". It lists several Python files and their commit history, including authors like "Altaf Alam" and dates ranging from November 8, 2022, to November 9, 2022.
- Bottom Terminal:** Shows the user navigating through another directory structure under "c:/Users/1002/altaf/Python". It lists several Python files and their commit history, similar to the middle terminal.

```
MINGW64:/c/Users/1002/altaf/python
$ git pull
From https://github.com/Altafalalm3/Python
 * branch            c3e6893..6ee44175  main      -> origin/main
Already up to date.

MINGW64:/c/Users/1002/altaf/python (main)
$ git push -u origin main
remote: Permission to Altafalalm3/Python.git denied to Altafalalm3.
fatal: unable to access 'https://github.com/Altafalalm3/Python.git/': The requested URL returned error: 403

MINGW64:/c/Users/1002/altaf/python (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3) done.
Writing objects: 100% (3/3) done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1) completed with 1 local object.
To https://github.com/Altafalalm3/Python.git
   6ee44175..935el84  main -> main
branch 'main' set up to track 'origin/main'.

MINGW64:/c/Users/1002/altaf/python (main)
$ git add .

MINGW64:/c/Users/1002/altaf/python (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>" to unstage)
    new file:   ass2.pdf

MINGW64:/c/Users/1002/altaf/python (main)
$ git reset ass2.pdf

MINGW64:/c/Users/1002/altaf/python (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ass2.pdf

MINGW64:/c/Users/1002/altaf/Python
added programs

commit 25aba0f6d78b9dce75bd1ac125a166a838e93354
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:39:11 2022 +0530

  Update 2_Operators.py

commit 49390b59406df2a588045647ead2460020dd577
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:37:21 2022 +0530

  update 2_Operators.py

commit bcd51cfc781a814d28c287a99ad41e2bf68e5e
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:26:31 2022 +0530

  Update 3_Strings.py

commit 233f41e3f615594715491fcecaeafa72b39d6f8F9
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:14:49 2022 +0530

  Update 3_Strings.py

commit fc3477910686fcccc0c21ebeb878459061692b4
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:11:38 2022 +0530

  update 2_Operators.py

commit 4b031137f31ffaa667d268366c52aa0ca424fa56
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 19:06:38 2022 +0530

  update 2_Operators.py

commit 17e38502acefd1cc2a32838527adc65b25d8a2c5
Author: Altaf Alam <02013452+Altafalalm3@users.noreply.github.com>
Date:   Tue Nov 8 18:55:46 2022 +0530

  Update 2_Variables.py

commit a7a19567248cb76fbffff392f810d1390d064d3fd
Author: Altafalalm3 <altaf.alam032@gmail.com>
Date:   Tue Nov 8 13:43:44 2022 +0530

added programs
[END]
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

The image shows three separate terminal windows (cmd.exe) running on a Windows operating system. Each window displays a different stage of a Git workflow:

- Top Terminal:** Shows the initial state where a local 'main' branch is ahead of the remote 'origin/main'. It includes a commit history for a 'pikachu assignment' and a merge from 'origin/main'. A file named 'ass1.pdf' is present in the local repository.
- Middle Terminal:** Shows the result of a 'git push -u origin main' command. It indicates that the local 'main' branch is now fully synchronized with the remote 'origin/main'. The file 'ass1.pdf' has been successfully pushed.
- Bottom Terminal:** Shows the result of a 'git push' command after adding the 'ass1.pdf' file. It shows the file was staged and committed, and then pushed to the remote repository.

```
MINGW64:/c/Users/1002/altaf/Python (main)
$ git push -u origin main
Fatal: User cancelled dialog.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/bin/git-askpass.exe'
Username for 'https://github.com':
commit 935el84laec549146ced10e0626736149aa7ebc4 (HEAD -> main)
Author: Altaf Alam <02013452+AltafAlam3@users.noreply.github.com>
Date:  Tue Aug 8 15:07:57 2023 +0530

  pikachu assignment done

commit 6ed4417570c12493ac3cdff1d2d27f62fd62321661 (upstream/main)
Author: Altaf Alam <02013452+AltafAlam3@users.noreply.github.com>
Date:  Tue Aug 8 15:01:58 2023 +0530

  Add files via upload

commit c3e68939497d845bffa554da000die872e975d2 (origin/main, origin/HEAD)
Merge: e8ac134 ed7b936
Author: AltafAlam3 <altaf.alam0032@gmail.com>
Date:  Sat Feb 25 18:24:28 2023 +0530

  Merge branch 'main' of https://github.com/AltafAlam3/PythonBasics

commit e8ac134957844cccf7e6cef9d88240b217bd57d8
Author: AltafAlam3 <altaf.alam0032@gmail.com>
Date:  Sat Feb 25 18:23:58 2023 +0530

  added

commit ed7b9362015da76548fb0fa69c03675680d053e2
Author: Altaf Alam <02013452+AltafAlam3@users.noreply.github.com>
Date:  Fri Jan 27 15:03:13 2023 +0530

  Delete Python directory

commit 9c2a0823aee6cc93f8e7f7cdb9cae50d1114e6e8
Author: Altaf Alam <02013452+AltafAlam3@users.noreply.github.com>
Date:  Fri Jan 27 15:01:57 2023 +0530

  Add files via upload

commit 1c3bdf8c969be40fdd8a44dba605c0d54851f
... skipping...
commit 935el84laec549146ced10e0626736149aa7ebc4 (HEAD -> main)
Author: Nj180280 <iranganjan180280@gmail.com>
Date:  Tue Aug 8 15:07:57 2023 +0530

Links ^ ⌂ ⌂ ⌂ ⌂ ⌂ 4:19 PM
```



```
MINGW64:/c/Users/1002/altaf/Python (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use 'git push' to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ass1.pdf

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

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git add ass1
fatal: pathspec 'ass1' did not match any files

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git add ass1.pdf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use 'git push' to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   ass1.pdf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git commit -m "pikachu assignment done"
[main 935el84] pikachu assignment done
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ass1.pdf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git push

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git push -u origin main
Fatal: User cancelled dialog.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/bin/git-askpass.exe'
Username for 'https://github.com':
commit 935el84laec549146ced10e0626736149aa7ebc4 (HEAD -> main)
Author: Nj180280 <iranganjan180280@gmail.com>
Date:  Tue Aug 8 15:07:57 2023 +0530

Links ^ ⌂ ⌂ ⌂ ⌂ ⌂ 4:19 PM
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23

```
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ass1.pdf

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

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git add ass1
fatal: pathspec 'ass1' did not match any files

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git add ass1.pdf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git commit -m "pikachu assignment done"
[main 935e184] pikachu assignment done
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ass1.pdf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git push

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git push -u origin main
fatal: User cancelled dialog.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/bin/gi
t-askpass.exe'
Username for 'https://github.com':
commit 935e184aec49146ced10e0626736149aa7ebc4 (HEAD -> main)
Author: Nj180280 <iranjan180280@gmail.com>
Date:  Tue Aug 8 15:07:57 2023 +0530

Links ▲ ↻ 🔍 ⌂ 4:19 PM
```



```
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote add upstream https://github.com/Altafalalm3/Python.git
usage: git remote add [<options>] <name> <url>

-f, --fetch      fetch the remote branches
--tags          import all tags and associated objects when fetching
               or do not fetch any tag at all (-no-tags)
-t, --track <branch> branch(es) to track
-m, --master <branch>
               master branch
--mirror=[(push|fetch)]  set up remote as a mirror to push to or fetch from

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote add upstream https://github.com/Altafalalm3/Python.git
error: remote upstream already exists.

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git fetch upstream
From https://github.com/Altafalalm3/Python
 * [new branch]      main      -> upstream/main

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git fetch upstream
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 391.39 KiB | 666.00 KiB/s, done.
From https://github.com/Altafalalm3/Python
 * [new branch]      main      -> upstream/main

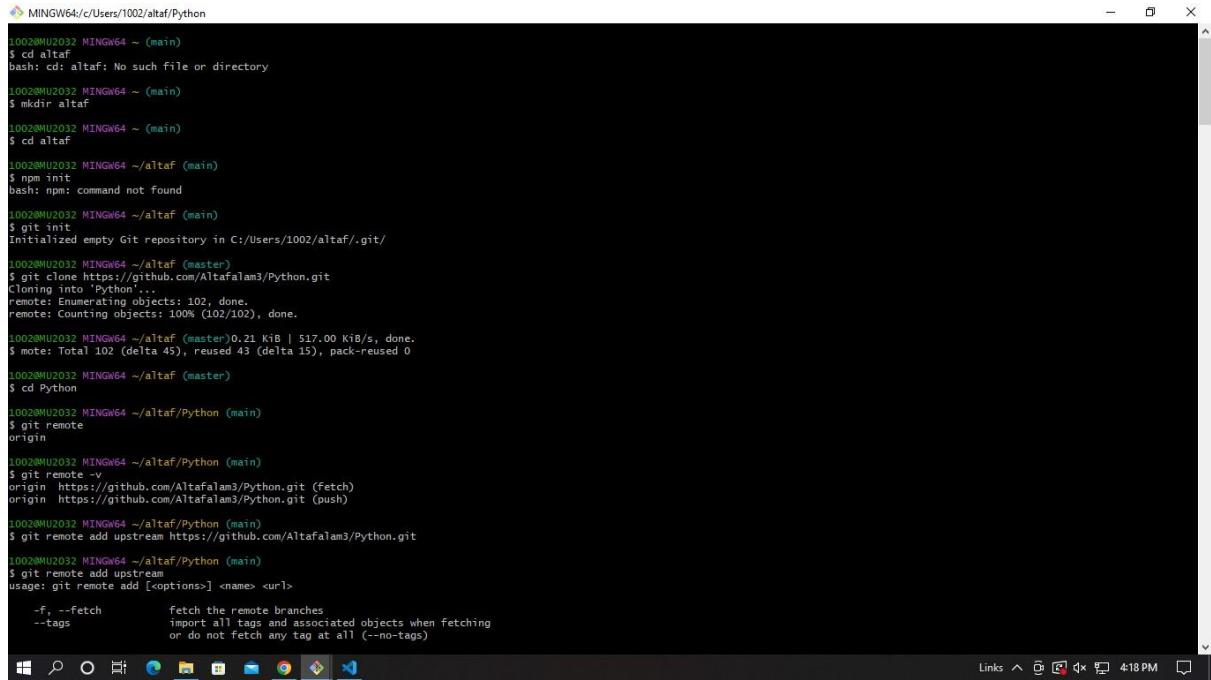
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git merge upstream/main
Updating c3e6893..6e44175
Fast-forward
  Devops_ass1.rtf | 151685 ++++++-----+
  1 file changed, 151685 insertions(+)
 create mode 100644 Devops_ass1.rtf

1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 15/08/23



The screenshot shows a terminal window with a black background and white text. It displays a sequence of Git commands being run in a Windows command prompt (MINGW64). The commands include navigating to a directory, cloning a GitHub repository, and updating remote branches. The terminal window has a standard Windows title bar at the top and a taskbar with various icons at the bottom.

```
MINGW64/c/Users/1002/altaf/Python
1002@MU2032 MINGW64 ~ (main)
$ cd altaf
bash: cd: altaf: No such file or directory
1002@MU2032 MINGW64 ~ (main)
$ mkdir altaf
1002@MU2032 MINGW64 ~ (main)
$ cd altaf
1002@MU2032 MINGW64 ~/altaf (main)
$ npm init
bash: npm: command not found
1002@MU2032 MINGW64 ~/altaf (main)
$ git init
Initialized empty Git repository in C:/Users/1002/altaf/.git/
1002@MU2032 MINGW64 ~/altaf (master)
$ git clone https://github.com/Altafalalm3/Python.git
Cloning into 'Python'...
remote: Enumerating objects: 102, done.
remote: Counting objects: 100% (102/102), done.
1002@MU2032 MINGW64 ~/altaf (master)
$ git status
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote
origin
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote -v
origin https://github.com/Altafalalm3/Python.git (fetch)
origin https://github.com/Altafalalm3/Python.git (push)
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote add upstream https://github.com/Altafalalm3/Python.git
1002@MU2032 MINGW64 ~/altaf/Python (main)
$ git remote add upstream
usage: git remote add [<options>] <name> <url>
      -f, --fetch           fetch the remote branches
      --tags                import all tags and associated objects when fetching
      or do not fetch any tag at all (--no-tags)
```

Conclusion :

Mastering these Git commands empowers effective version control and collaborative coding. Learning them facilitates seamless collaboration and efficient management of code repositories.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

Experiment No 4

Aim : To understand Continous Integration, Jenkins installation.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO3 : To understand the importance of Jenkins to build and deploy software applications on server environment.

Theory :

Steps :

Jenkins is an open-source automation server that facilitates continuous integration and delivery. It automates software building, testing, and deployment processes, enabling teams to achieve faster development cycles and better code quality. Jenkins supports various plugins, allowing integration with version control systems, testing frameworks, and deployment tools. Through pipelines, Jenkins orchestrates workflows, ensuring consistent and automated execution of tasks. Developers receive rapid feedback on code changes, and the system helps identify and address issues early in the development lifecycle. Jenkins promotes collaboration, scalability, and efficiency in software development by automating repetitive tasks and fostering seamless integration within the development ecosystem.

STEPS TO INSTALL

Step 1: Check System Requirements

Ensure your system meets the following requirements:

- Operating System: Windows 7 or later
- Java: Java 8 or Java 11 (required for Jenkins)
- Minimum RAM: 1 GB (2 GB recommended)
- Disk Space: 10 GB free space

Step 2: Install Java

1. Download and install Java Development Kit (JDK) 8 or 11 from the official Oracle website or OpenJDK.
2. Set the 'JAVA_HOME' environment variable to point to the JDK installation directory.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

Step 3: Download Jenkins

1. Visit the Jenkins official website: <https://www.jenkins.io/download/>
2. Download the Windows installer package (`.msi` file).

Step 4: Install Jenkins

1. Double-click the downloaded `.msi` file to start the installer.
2. Choose the installation directory (default is usually fine).
3. Select "Install as a Windows Service" and choose a service name (e.g., "Jenkins").
4. Choose whether Jenkins should run as a Local System account or as a specific user. Using a specific user is recommended for security reasons.
5. Specify the location for Jenkins to store its data (e.g., `C:\Jenkins`).
6. Review your choices and click "Install."

Step 5: Complete Setup

1. Once the installation is complete, a window will open in your default web browser to the Jenkins setup wizard.
2. Retrieve the initial admin password. The password can be found in a file named `initialAdminPassword`, located in the Jenkins installation directory (e.g., `C:\Program Files\Jenkins\secrets\initialAdminPassword`).
3. Copy and paste the password from the file into the setup wizard and click "Continue."

Step 6: Customize Plugins (Optional)

1. Choose either the "Install suggested plugins" option or the "Select plugins to install" option.
2. If you choose the second option, select the plugins you want to install. For starters, you can select the suggested plugins.

Step 7: Create Admin User

1. Enter the required details to create an admin user (username, password, full name, and email).
2. Click "Save and Finish."

Step 8: Instance Configuration (Optional)

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

1. If needed, you can change the Jenkins instance's URL. Otherwise, you can leave it as the default value.
2. Click "Save and Finish."

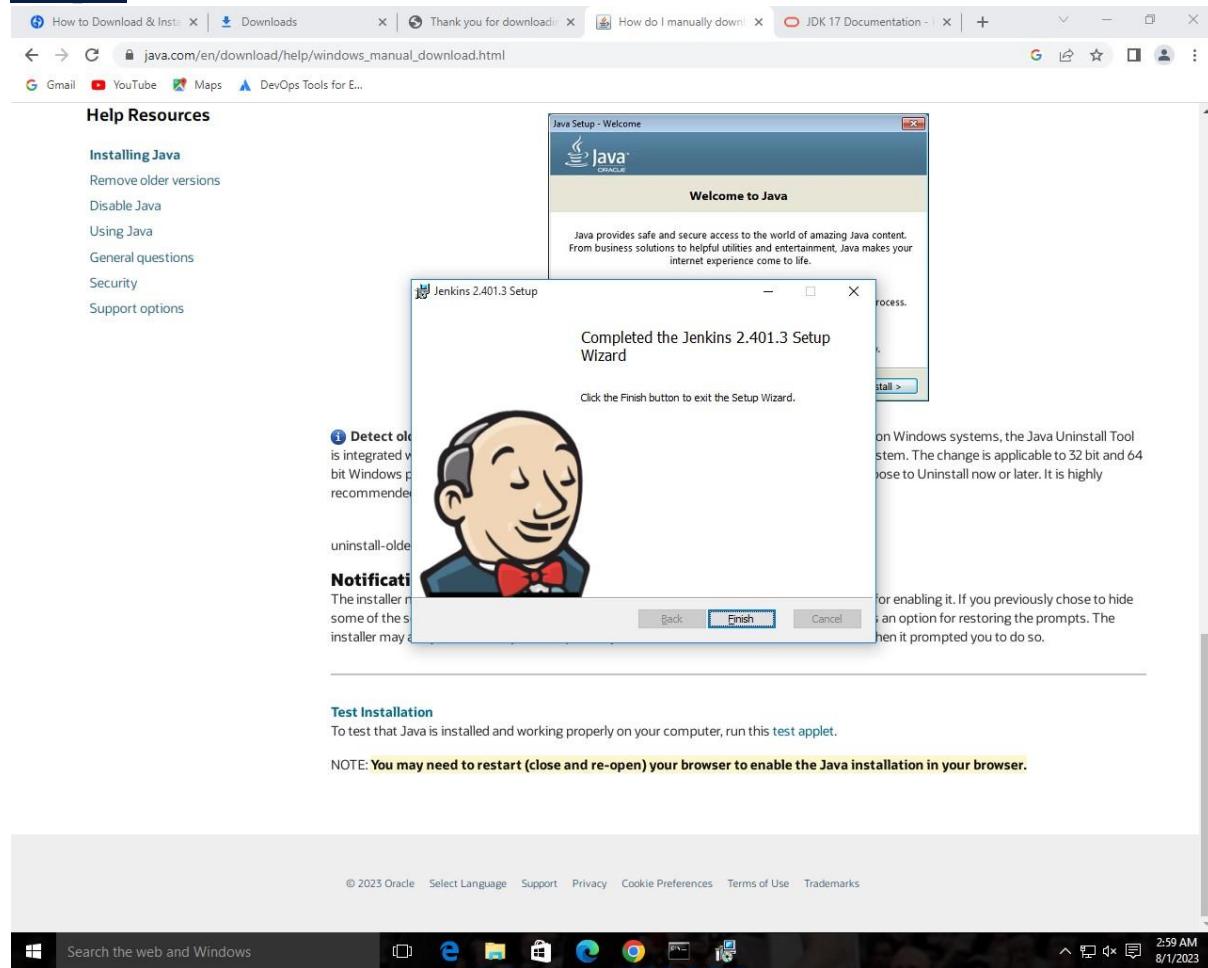
Step 9: Start Jenkins

1. Click "Start using Jenkins."
2. Jenkins will redirect you to the dashboard.

Step 10: Access Jenkins Dashboard

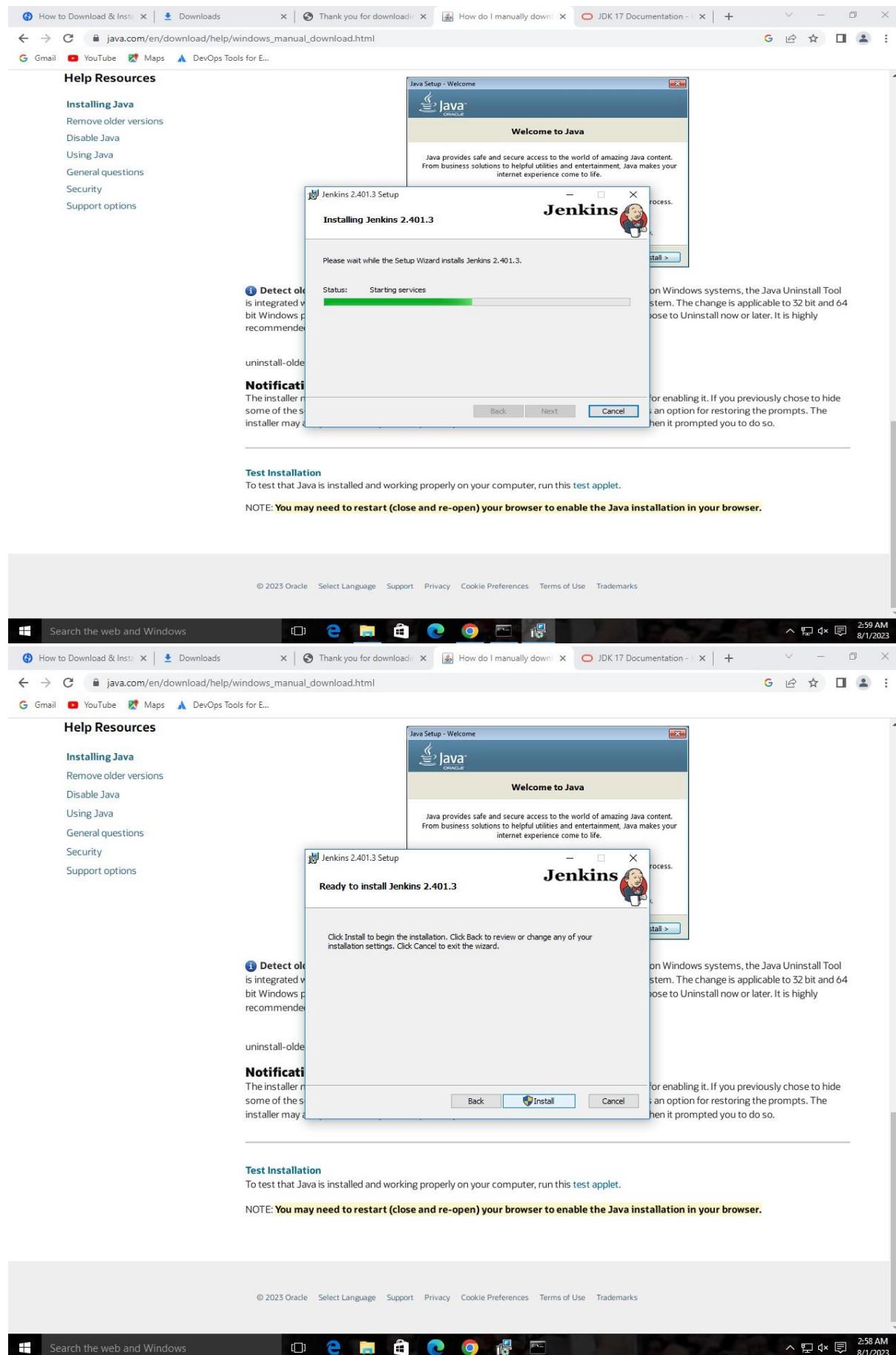
1. Open your web browser and navigate to `http://localhost:8080` (or the custom URL you configured).
2. Log in using the admin credentials you created earlier.

Output :



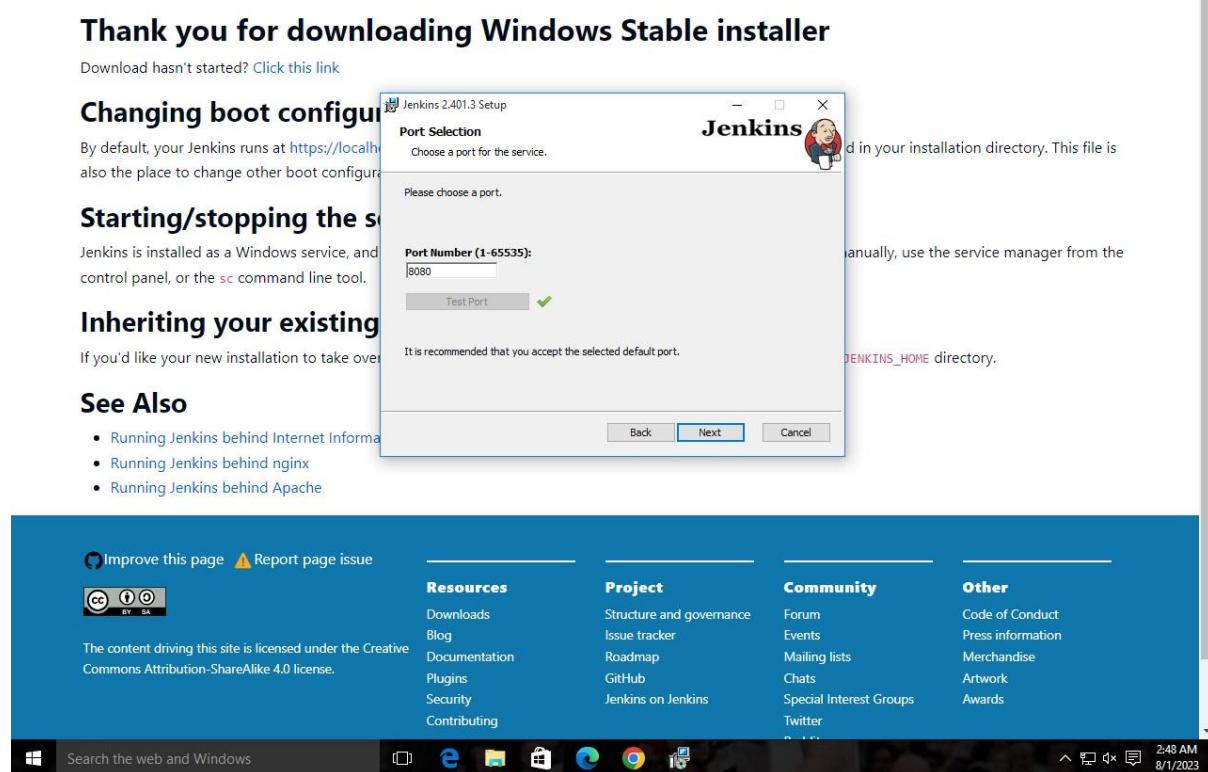
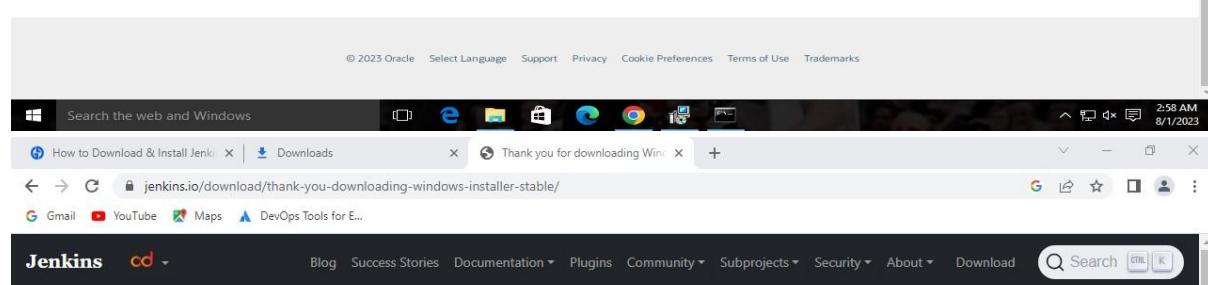
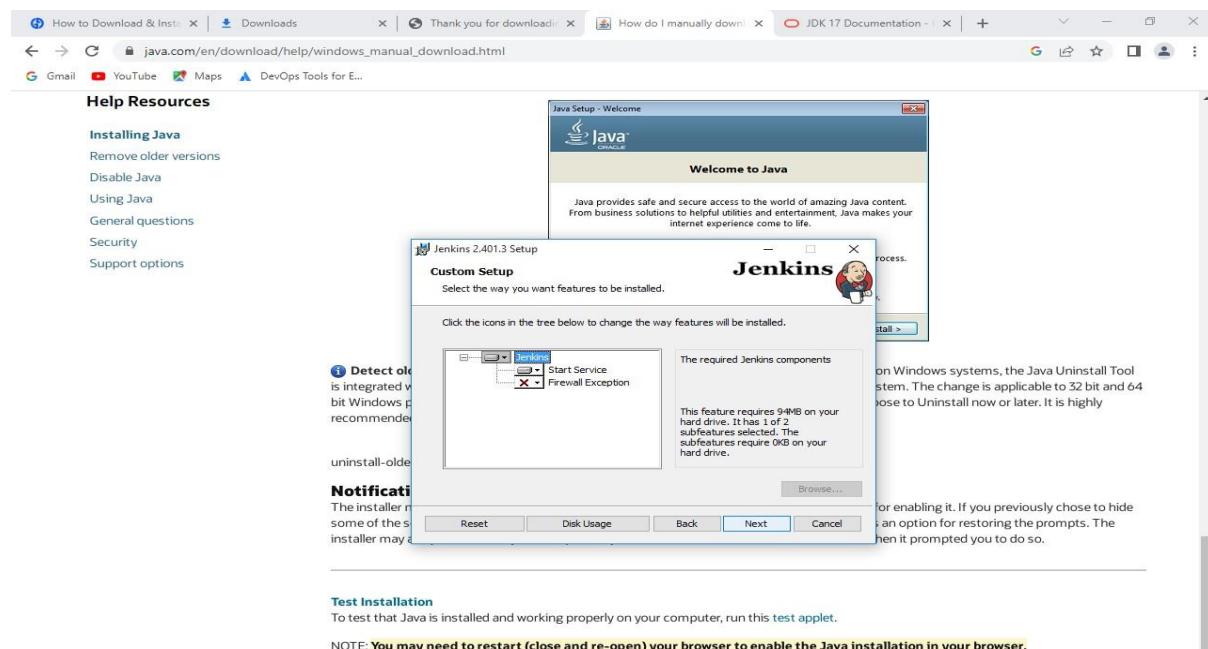
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows a web browser with multiple tabs open. The main tab displays the Jenkins download page, which includes sections for boot configuration, starting/stopping the service, inheriting existing installations, and a 'See Also' sidebar. An overlay window titled 'Jenkins 2.401.3 Setup' is centered over the page, specifically on the 'Service Logon Credentials' step. This window asks for account and password information to run Jenkins as a Windows service. The background page has a blue header bar with Jenkins navigation links like 'Blog', 'Success Stories', 'Documentation', etc.

This screenshot is similar to the one above, showing the Jenkins download page with its various sections and a 'See Also' sidebar. The 'Service Logon Credentials' setup window is again overlaid, this time on the 'Starting/stopping the service' section. The window's title bar says 'Jenkins 2.401.3 Setup' and it contains fields for 'Account' and 'Password'. The background page features a blue header with Jenkins-related links.

This screenshot shows the Jenkins download page with its standard layout. A Jenkins 2.401.3 Setup window is overlaid on the 'Starting/stopping the service' section. The setup window is titled 'Service Logon Credentials' and asks for logon type, account, and password. The 'Install' button at the bottom of the window is highlighted with a red box. To the right of the main content area, there is a sidebar with various Jenkins-related links and a small advertisement for FXM.

Step 5) Click on the Install button.

This screenshot shows the Jenkins 2.129 Setup window, which is part of the download process. It displays the path 'C:\Program Files (x86)\Jenkins\'. Below this, the 'Service Logon Credentials' window is shown, with the 'Install' button highlighted by a red box. The background page shows the Jenkins download page with its various sections and a sidebar.

Step 6) Once install is complete, click Finish.

This screenshot shows the Jenkins 2.129 Setup window again, but this time the 'Completed the Jenkins 2.129 Setup Wizard' message is displayed. The 'Finish' button at the bottom of the window is highlighted with a red box. The background page shows the Jenkins download page with its various sections and a sidebar.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows a Microsoft Edge browser window. The address bar displays 'jenkins.io/download/thank-you-downloading-windows-installer-stable/'. The main content area shows the Jenkins download page with sections like 'Thank you for downloading Windows Stable installer', 'Changing boot configuration', 'Starting/stopping the service', 'Inheriting your existing Jenkins installation', and 'See Also'. Overlaid on the page is the 'Jenkins 2.401.3 Setup' window. The setup window has a title 'Welcome to the Jenkins 2.401.3 Setup Wizard'. It features a cartoon Jenkins head icon and text explaining the setup process. Buttons for 'Back', 'Next', and 'Cancel' are visible at the bottom.

This screenshot is identical to the one above, showing the Jenkins download page and the 'Jenkins 2.401.3 Setup' window overlaid. The setup window is titled 'Welcome to the Jenkins 2.401.3 Setup Wizard' and contains the same text and icons as the previous screenshot.

This screenshot is identical to the ones above, showing the Jenkins download page and the 'Jenkins 2.401.3 Setup' window overlaid. The setup window is titled 'Welcome to the Jenkins 2.401.3 Setup Wizard' and contains the same text and icons as the previous screenshots.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows a web browser window with two side-by-side panels for downloading Jenkins. The left panel is for Jenkins 2.401.3 LTS and the right panel is for Jenkins 2.416. Both panels offer various installation packages including Generic Java package (.war), Docker, Kubernetes, Ubuntu/Debian, Red Hat/Fedora/Alma/Rocky/CentOS, Windows, openSUSE, Arch Linux, FreeBSD, Gentoo, macOS, and OpenBSD. Below each panel is a download link. At the bottom, a download progress bar indicates 'jenkins.msi' is at 93.8/99.7 MB, with 6 secs left. The browser's address bar shows 'jenkins.io/download/'.

Conclusion :

Hence, we understood about the Jenkins and the step by step process to install the Jenkins in the window software and all the requirements needed for it .

Experiment No 5

Aim : To build java programs using Jenkins.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO3 : To understand the importance of Jenkins to build and deploy software applications on server environment.

Theory :

Steps :

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

Building Java programs using Jenkins involves setting up a continuous integration and continuous deployment (CI/CD) pipeline that automates the process of building, testing, and deploying Java applications. Jenkins is a popular open-source automation server that can help achieve this. Here's a step-by-step process to build Java programs using Jenkins:

Step 1: Install Jenkins

1. Install Jenkins on a server or local machine.
2. Access Jenkins through a web browser using the provided URL.
3. Follow the initial setup steps to unlock Jenkins and install recommended plugins.

Step 2: Configure Jenkins

1. Navigate to "Manage Jenkins" > "Global Tool Configuration."
2. Install and configure JDK:
 - Click on "JDK installations..." - Add a new JDK installation and provide the required details.

Step 3: Create a New Jenkins Job

1. From the Jenkins dashboard, click on "New Item."
2. Enter a name for your project and select "Freestyle project." 3. Click "OK" to create the project.

Step 4: Configure Source Code Repository

1. Under "Source Code Management," choose your version control system (e.g., Git).
2. Provide the repository URL and credentials, if required.

Step 5: Configure Build

1. Under the "Build" section, add a build step:
 - Choose "Invoke top-level Maven targets" or "Execute shell" (for Ant or custom build scripts).
 - Configure the build command (e.g., `mvn clean install` for Maven projects). - Specify the JDK version you configured earlier.

Step 6: Configure Post-Build Actions

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

1. Under the "Post-build Actions" section:

- Choose actions like archiving artifacts, triggering other builds, or sending notifications.
- For example, archive the built JAR files.

Step 7: Save and Build

1. Save your Jenkins job configuration.
2. Trigger a manual build to test the setup.
3. Observe the build process, console output, and any issues.

Step 8: Set Up Webhooks or Polling

1. Configure webhooks or polling in your version control system to trigger builds on code changes.

Step 9: Advanced Steps (Optional)

1. Integrate with a testing framework (JUnit, TestNG) to run automated tests.
2. Set up deployment to different environments using plugins or scripts.
3. Use Jenkins pipeline DSL to define complex build and deployment workflows.

Step 10: Monitor and Improve

1. Monitor build status, logs, and performance.
2. Review test results and identify areas for improvement.
3. Adjust your Jenkins setup and build configuration as needed.

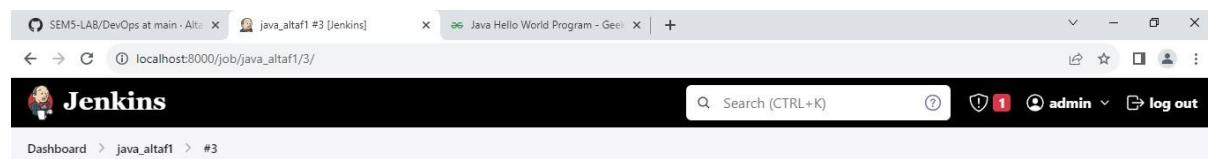
By following these steps, you can create a robust CI/CD pipeline for building Java applications using Jenkins. This process automates the build and deployment steps, ensuring that your Java programs are consistently built and tested whenever changes are made to your codebase.

Output :

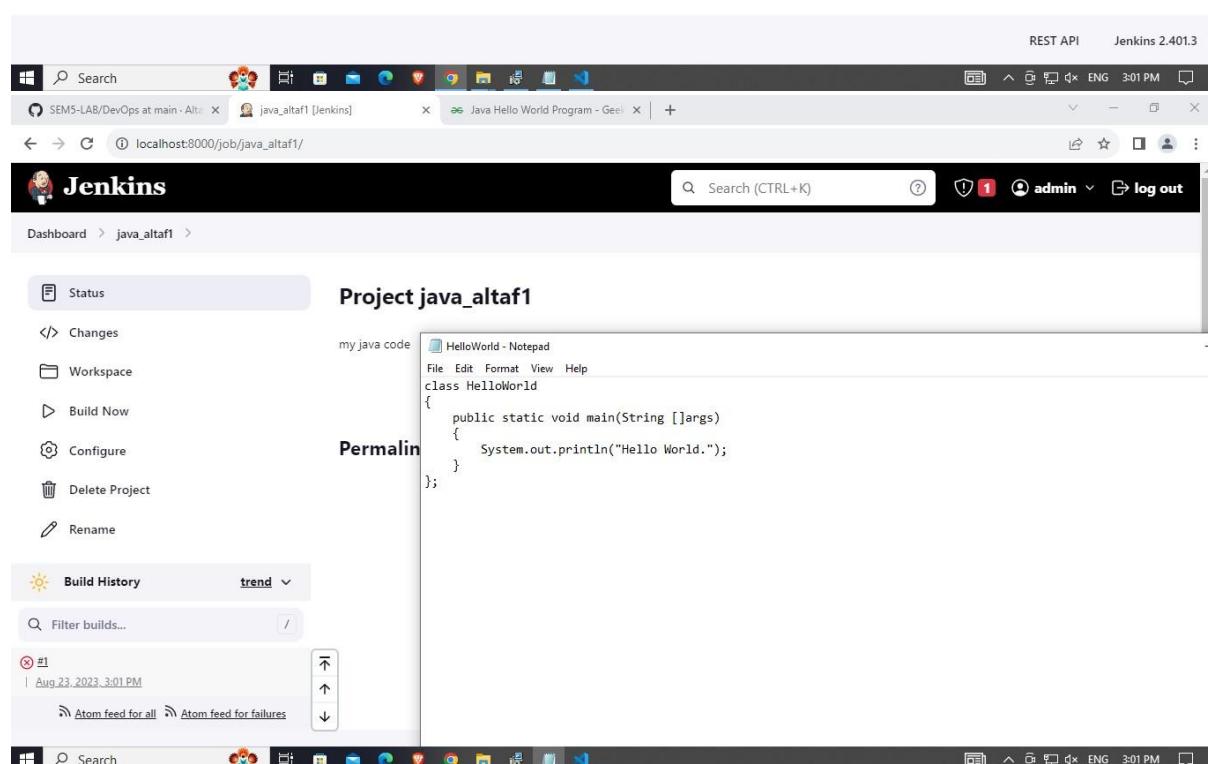
1) Building Java Programs

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



Screenshot of Jenkins showing Build #3 (Aug 23, 2023, 3:01:50 PM). The build status is green (Success). It shows 'No changes' and was started by user 'admin'. The build took 0.67 sec.



Screenshot of Jenkins showing the Project java_altaf1. The workspace contains a file named 'HelloWorld - Notepad' with the following Java code:

```
my java code
File Edit Format View Help
class HelloWorld
{
    public static void main(String []args)
    {
        System.out.println("Hello World.");
    }
};
```

The build history shows one build from Aug 23, 2023, 3:01 PM.

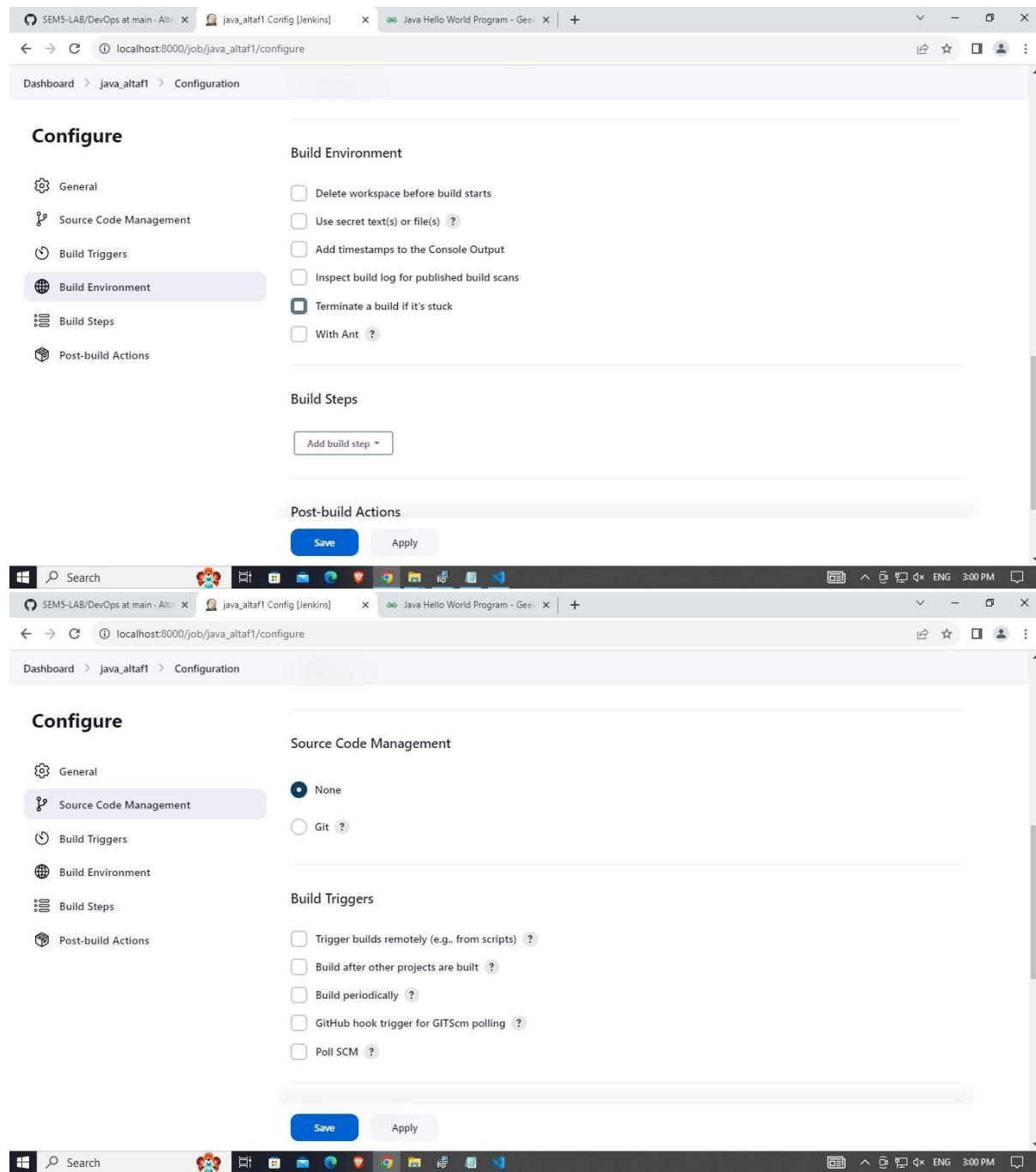
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows two side-by-side browser windows. The left window displays the Jenkins interface for the project 'java_altaf1'. It includes a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main area shows a 'Build History' section with a single build entry labeled 'trend' and a note 'No builds'. The right window shows the configuration page for the same project. The 'Build Steps' section contains a step titled 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'. Below this is a 'Post-build Actions' section with an 'Add post-build action' button. At the bottom of both windows are standard Windows taskbars.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



The screenshot shows the Jenkins configuration page for a job named "Java Hello World Program - Geel". The "Build Environment" section is currently selected. It contains several checkboxes for build-related options:

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

The "Build Steps" section has a button labeled "Add build step ▾".

The "Post-build Actions" section includes "Save" and "Apply" buttons.

The browser window title is "localhost:8000/job/java_althaf1/configure".

The operating system taskbar at the bottom shows various application icons and the date/time as 3:00 PM.

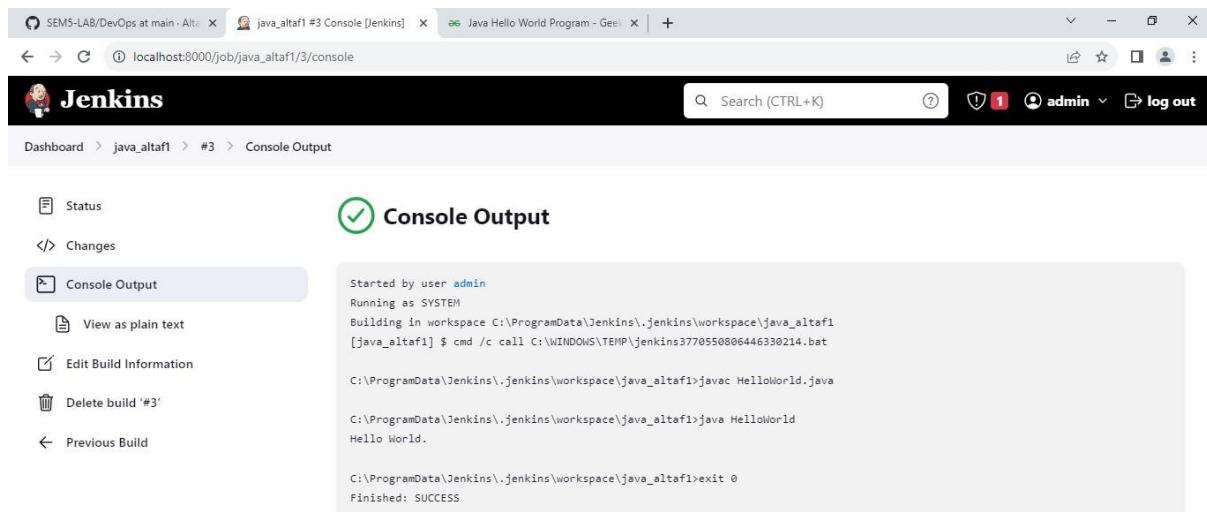
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows two stacked Jenkins interfaces. The top interface is the 'Configure' screen for a job named 'java_althaf1'. It includes a 'General' tab with a description field containing 'my java code', several checkboxes for build options like 'Discard old builds', and buttons for 'Save' and 'Apply'. The bottom interface is the 'New Item' creation screen, showing a form to 'Enter an item name' with 'java_althaf1' entered. It lists three project types: 'Freestyle project', 'Pipeline', and 'Multi-configuration project', each with a brief description and an 'OK' button.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



The screenshot shows a browser window with three tabs open:

- SEMS-LAB/DevOps at main - Altaf Alam
- java_altaf1 #3 Console [Jenkins]
- Java Hello World Program - GeeksforGeeks

The second tab is active and displays the Jenkins interface. The title bar says "localhost:8000/job/java_altaf1/3/console". The main content area is titled "Console Output" with a green checkmark icon. On the left, there's a sidebar with links: Status, Changes, Console Output (which is selected and highlighted in grey), View as plain text, Edit Build Information, Delete build '#3', and Previous Build. The main pane shows the command-line output of the build:

```
Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\java_altaf1
[java_altaf1] $ cmd /c call C:\WINDOWS\TEMP\jenkins3770550806446330214.bat
C:\ProgramData\Jenkins\.jenkins\workspace\java_altaf1>javac HelloWorld.java
C:\ProgramData\Jenkins\.jenkins\workspace\java_altaf1>java HelloWorld
Hello World.

C:\ProgramData\Jenkins\.jenkins\workspace\java_altaf1>exit 0
Finished: SUCCESS
```



2) Building Java Program using Git

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows the Jenkins configuration interface for a job named "java_git2". The "Source Code Management" section is selected. Under "Branches to build", the "Branch Specifier" is set to */main. In the "Build Triggers" section, the "Git" trigger is selected, and the "Repository URL" is set to https://github.com/Altafalam3/Java-Lab.git. The "Save" and "Apply" buttons are visible at the bottom.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows a web browser window with multiple tabs open. The tabs include:

- Altafalam3 (Altaf Alam)
- java.git2 Config [Jenkins]
- Selenium Tutorial - javatpoint
- Java Hello World Program - GeeksforGeeks
- localhost:8000/job/java_git2/configure

The main content area displays the Jenkins configuration for a job named "java_git2". The "General" configuration page is shown, with the "Enabled" checkbox checked. The "Description" field contains the text "Author : Altaf Alam". Below the description, there are several checkboxes for build options:

- Discard old builds
- GitHub project
- This project is parameterized
- Throttle builds
- Execute concurrent builds if necessary

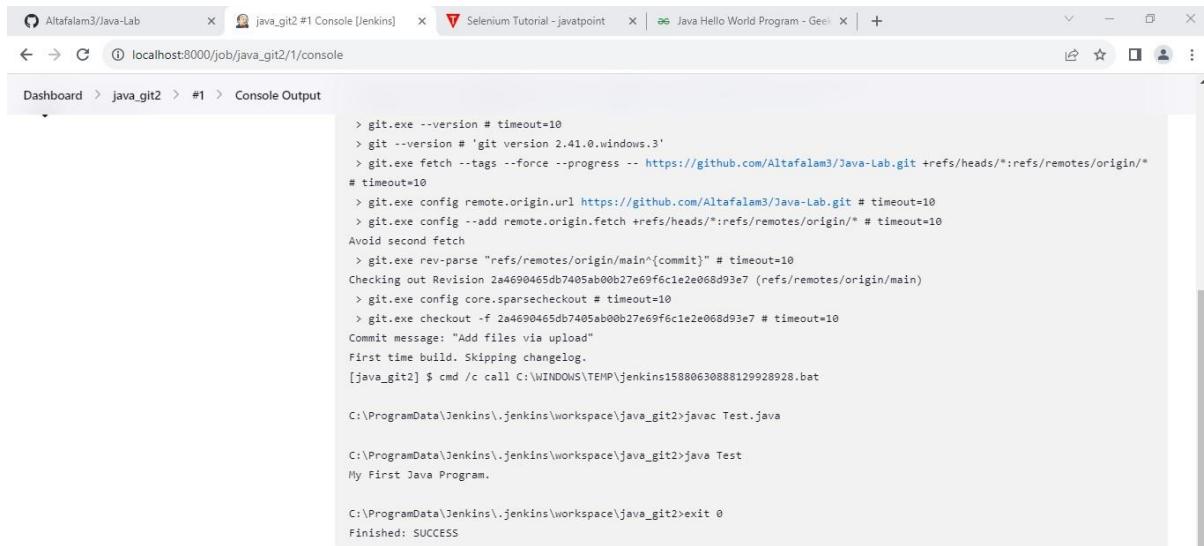
At the bottom of the configuration page are "Save" and "Apply" buttons.

Below the Jenkins configuration, the browser window shows a GitHub repository named "Altafalam3 / Java-Lab". The "Code" tab is selected, displaying the file "Test.java". The code content is:1 class Test
2 {
3 public static void main(String []args)
4 {
5 System.out.println("My First Java Program.");
6 }
7 }

The GitHub interface also includes a "Symbols" panel on the right side, which lists symbols found in the code, such as "class Test" and "func main".

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



The screenshot shows a browser window with four tabs open:

- Altafalalm3/Java-Lab
- java.git2 #1 Console [Jenkins]
- Selenium Tutorial - javatpoint
- Java Hello World Program - GeeksforGeeks

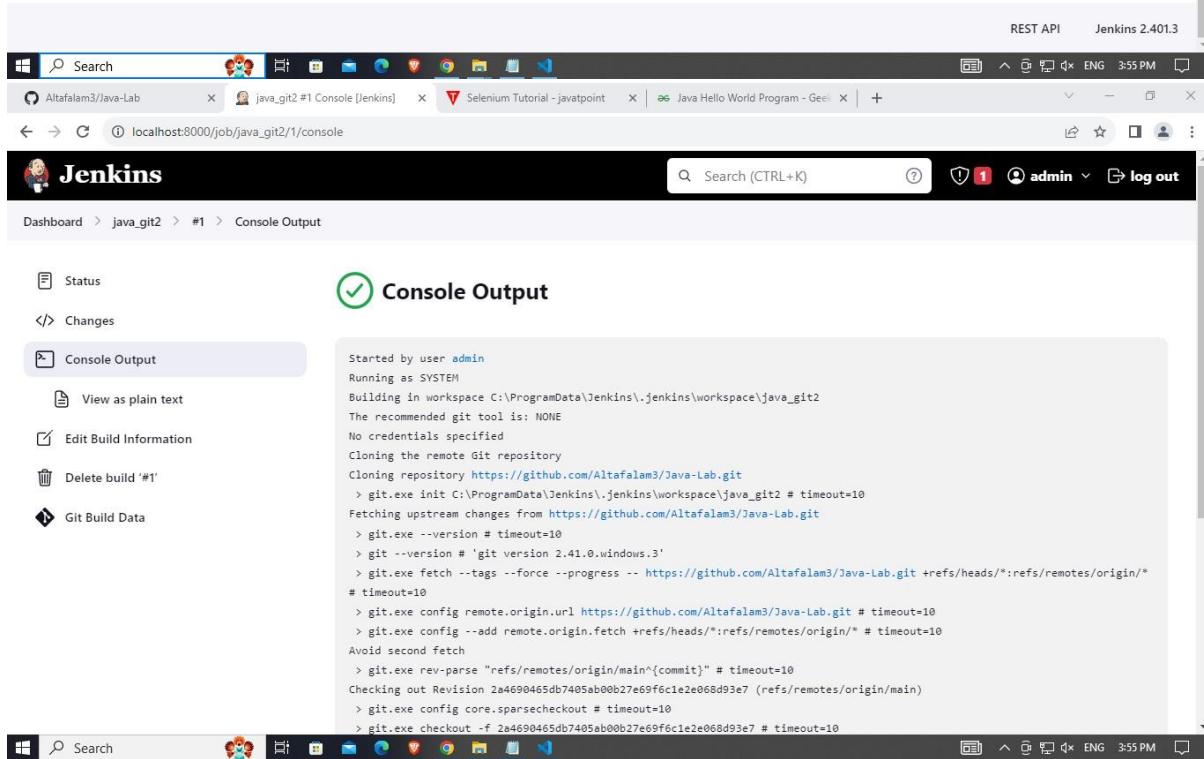
The active tab is "java.git2 #1 Console [Jenkins]". The console output is as follows:

```
> git.exe --version # timeout=10
> git --version # 'git version 2.41.0.windows.3'
> git.exe fetch --tags --force --progress -- https://github.com/Altafalalm3/Java-Lab.git +refs/heads/*:refs/remotes/origin/*
# timeout=10
> git.exe config remote.origin.url https://github.com/Altafalalm3/Java-Lab.git # timeout=10
> git.exe config -add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 2a4690465db7405ab00b27e69f6c1e2e068d93e7 (refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 2a4690465db7405ab00b27e69f6c1e2e068d93e7 # timeout=10
Commit message: "Add files via upload"
First time build. Skipping changelog.
[java_git2] $ cmd /c call C:\WINDOWS\TEMP\jenkins15880630888129928928.bat

C:\ProgramData\Jenkins\.jenkins\workspace\java_git2>javac Test.java

C:\ProgramData\Jenkins\.jenkins\workspace\java_git2>java Test
My First Java Program.

C:\ProgramData\Jenkins\.jenkins\workspace\java_git2>exit 0
Finished: SUCCESS
```



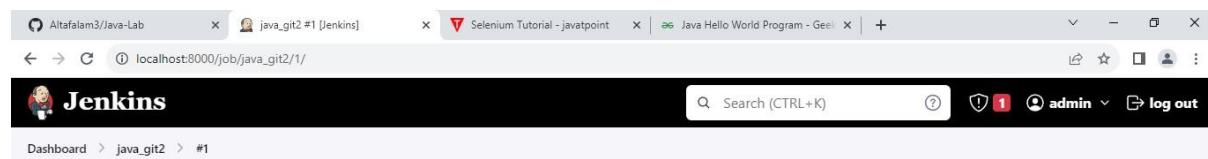
The screenshot shows a browser window with four tabs open:

- Altafalalm3/Java-Lab
- java.git2 #1 Console [Jenkins]
- Selenium Tutorial - javatpoint
- Java Hello World Program - GeeksforGeeks

The active tab is "java.git2 #1 Console [Jenkins]". The Jenkins UI is overlaid on the browser window, showing the "Console Output" section. The output is identical to the one in the previous screenshot.

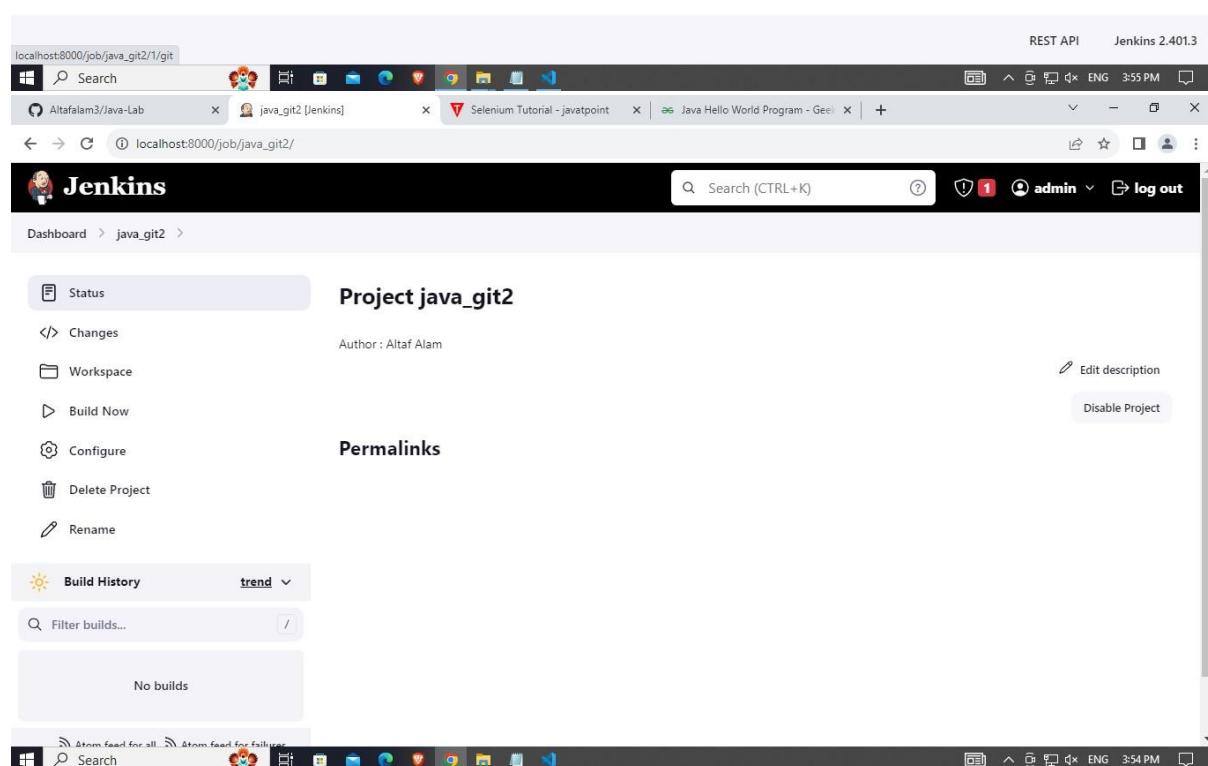
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23



A screenshot of a web browser showing a Jenkins job named "java_git2". The current build is #1, which was successful (indicated by a green checkmark icon). The build was started 13 seconds ago and took 8.8 seconds. The build information shows "No changes" and was started by user "admin". The repository information is listed as follows:

- Revision: 2a4690465db7405ab00b27e69f6c1e2e068d93e7
- Repository: <https://github.com/Altafalam3/Java-Lab.git>
- refs/remotes/origin/main



A screenshot of a web browser showing the Jenkins project "java_git2". The project page displays the following details:

- Project Name:** Project java_git2
- Author:** Altaf Alam
- Actions:** Edit description, Disable Project
- Build History:** Trend dropdown, Filter builds... (No builds)

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 29/08/23

The screenshot shows the Jenkins configuration interface for a job named 'java_git2'. The 'Build Environment' tab is selected. Under 'Build Steps', there is a single step titled 'Execute Windows batch command'. The 'Command' field contains the following text:

```
javac Test.java  
Java Test
```

At the bottom of the screen, the taskbar shows various application icons and the system tray indicates the date and time as 3:54 PM.

Conclusion :

Hence, we understood about Jenkins and how to build programs of java directly and by using git.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

Experiment No 6

Aim : To build pipelines using Jenkins.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO3 : To understand the importance of Jenkins to build and deploy software applications on server environment.

Theory :

Jenkins is a popular open-source automation server used for building, testing, and deploying software. Jenkins Pipelines are a powerful feature that allows you to define and manage your continuous delivery (CD) and continuous integration (CI) processes as code. Here's an in-depth explanation of Jenkins Pipelines:

1. Continuous Delivery Pipelines:

- Continuous Delivery (CD) pipelines are a set of automated steps that code goes through, from development to production, ensuring that it is tested, reviewed, and deployed in a controlled and consistent manner.
- Jenkins Pipelines allow you to define, version, and execute these CD processes in a structured and repeatable way.

2. Jenkins Pipeline:

- A Jenkins Pipeline is a suite of plugins that enable implementing and integrating CD and CI workflows as code. It provides two main types of pipelines: Declarative and Scripted.

3. Declaration and Scripted Pipeline Syntax:

- Declarative Pipeline: This is a simplified and structured way to define pipelines. It uses a predefined set of directives to specify stages, steps, and conditions. Declarative pipelines are easier to read and maintain.
- Scripted Pipeline: This provides more flexibility by allowing you to write custom scripts in Groovy to define your pipeline. It gives you complete control over the workflow but can be more complex.

4. Benefits of Jenkins Pipelines:

- Version Control: Pipelines are defined as code, which means you can store them in version control systems like Git. This enables collaboration and history tracking.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

- Reusability: Pipelines can be reused across different projects, promoting consistency and reducing duplication of effort.
- Visibility: Pipelines provide clear visibility into the entire CD process. You can see where failures occur, making troubleshooting easier.
- Parallel Execution: Pipelines can execute multiple tasks in parallel, improving build and deployment speed.
- Extensibility: You can extend pipelines with custom scripts or integrate with other tools and services.
- Scalability: Pipelines can be distributed across multiple Jenkins agents or nodes, scaling to handle complex workflows and high workloads.

5. How Jenkins Pipelines Work:

- Jenkins Pipelines typically start with a trigger, such as code commits to a version control system.
- The pipeline definition specifies stages, steps, and conditions. Each stage represents a phase in the CD process (e.g., build, test, deploy).
- Jenkins agents (or nodes) execute the defined steps, which can include building, testing, and deploying the application.
- The pipeline can be configured to run automated tests, send notifications, and trigger deployments to various environments (e.g., development, staging, production). - If any step fails, the pipeline can be configured to send alerts or halt further execution.

Jenkins Pipelines provide a structured, code-driven approach to continuous delivery, making it easier to manage, automate, and visualize your software delivery process while promoting collaboration and consistency.

1st script :

Make a pipeline using add item and select pipeline

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'localhost:8000/view/all/newJob'. The page displays a 'Jenkins' header and a 'Dashboard > All >' breadcrumb. A central form is titled 'Enter an item name' with a field containing 'MyFirstPipeline2'. Below this, a note says 'Required field'. There are three job type options: 'Freestyle project' (selected), 'Pipeline', and 'Multi-configuration project'. Each option has a brief description and a 'OK' button. The status bar at the bottom shows a Windows taskbar with various icons and the time '3:21 PM'.

Add description

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'localhost:8000/job/MyFirstPipeline2/configure'. The page displays a 'Jenkins' header and a 'Dashboard > MyFirstPipeline2 > Configuration' breadcrumb. On the left is a sidebar with 'Configure' and 'General' tabs. The 'General' tab is selected, showing a 'Description' section with the text 'Altaf Alam
hello'. Below this is a 'Plain text' link and a preview area. A 'Enabled' toggle switch is turned on. On the right, there are several configuration checkboxes: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', and 'Pipeline speed/durability override'. At the bottom are 'Save' and 'Apply' buttons. The status bar at the bottom shows a Windows taskbar with various icons and the time '3:22 PM'.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

Add scripts to it hello world

The screenshot shows the Jenkins Pipeline configuration page for a job named 'MyFirstPipeline2'. The 'Pipeline' tab is selected in the sidebar. The main area displays a Groovy script for defining a pipeline. The script includes an agent section and a single stage named 'Hello' with an echo step. A dropdown menu next to the stage name is set to 'Hello World'. Below the script, there is a checkbox for 'Use Groovy Sandbox'. At the bottom, there are 'Save' and 'Apply' buttons.

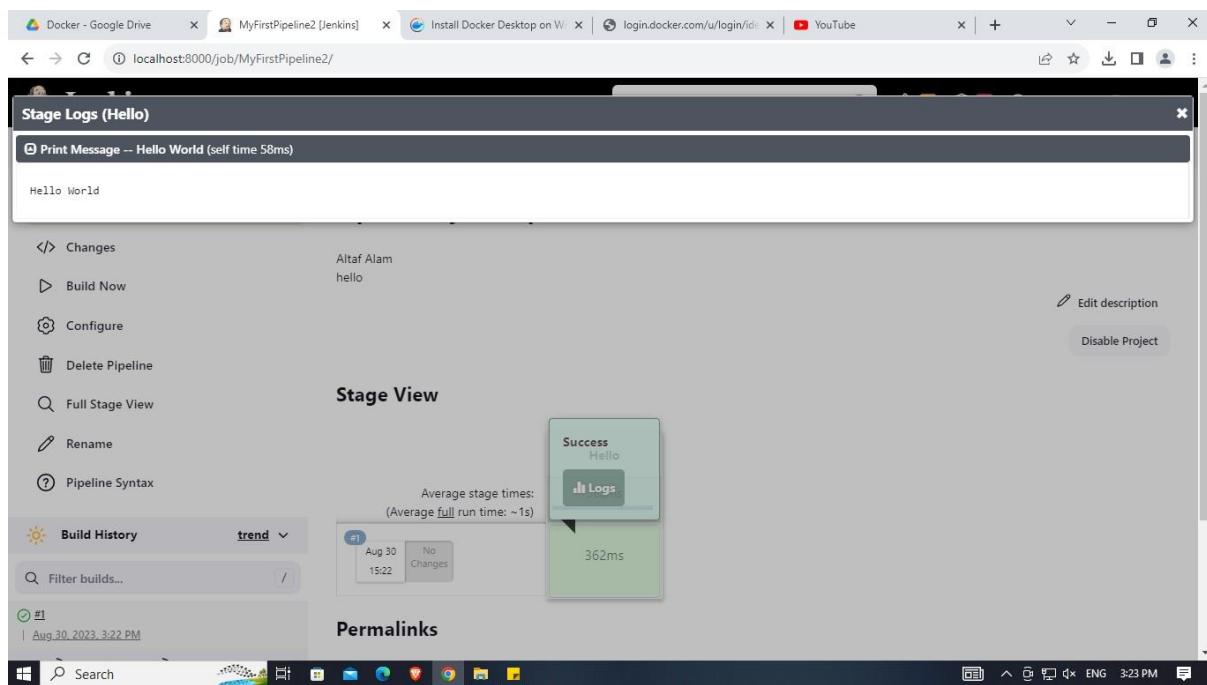
```
1> pipeline {  
2>     agent any  
3>  
4>     stages {  
5>         stage('Hello') {  
6>             steps {  
7>                 echo 'Hello World'  
8>             }  
9>         }  
10>    }  
11> }  
12>
```

Save and build now

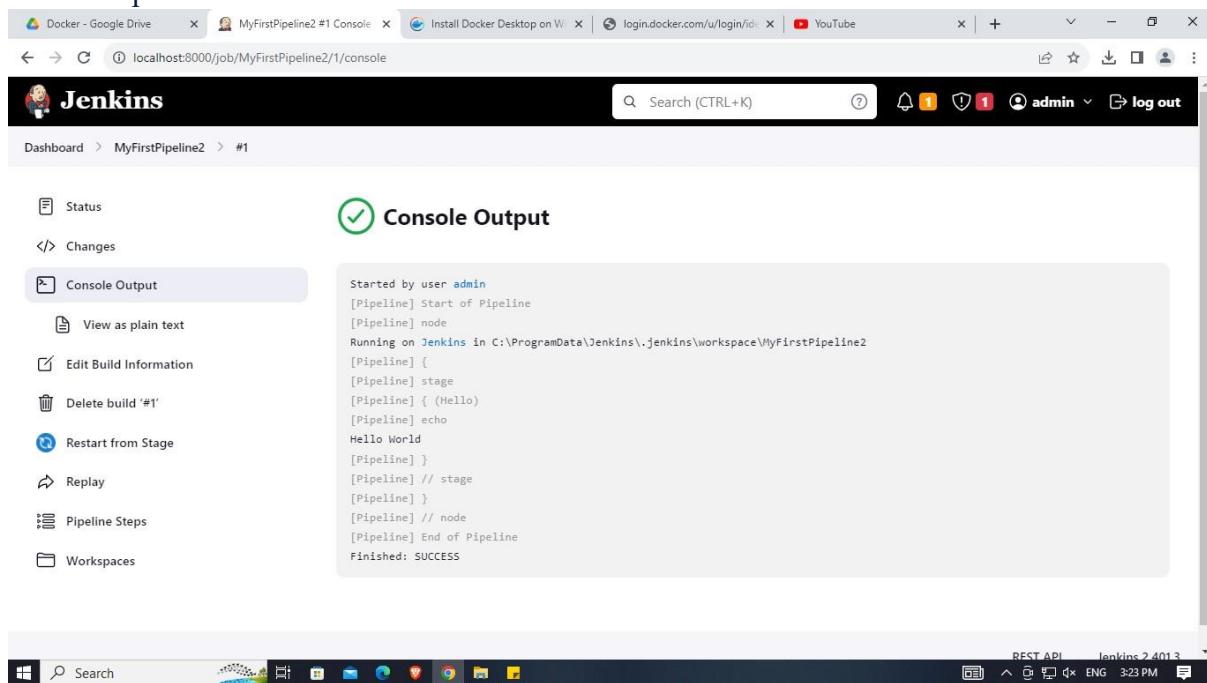
The screenshot shows the Jenkins Pipeline status page for 'MyFirstPipeline2'. The left sidebar has links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area is titled 'Pipeline MyFirstPipeline2' and shows the stage view with a message: 'No data available. This Pipeline has not yet run.' Below this is a 'Permalinks' section and a 'Build History' table which is currently empty. The status bar at the bottom indicates the user is 'admin'.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23



See the output in console



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

2nd Script pipeline:

The screenshot shows the Jenkins 'New Item' creation interface. A modal window titled 'Enter an item name' has 'altaf_pipeline3' entered into the required field. Below it, there are three project type options: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'Freestyle project' option is selected. At the bottom of the modal is an 'OK' button.

The screenshot shows the 'General' configuration page for the 'altaf_pipeline3' job. The 'Configure' tab is active. In the 'Description' section, the text 'Altaf Alam docker' is entered. Under the 'Advanced Project Options' section, the 'Do not allow concurrent builds' checkbox is checked. Other available options include 'Discard old builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterized', and 'Throttle builds'. At the bottom are 'Save' and 'Apply' buttons.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

The screenshot shows the Jenkins Pipeline configuration screen for a job named 'altaf_pipeline3'. The 'Pipeline' tab is selected. The pipeline script is displayed in a code editor:

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Build') {
6             steps {
7                 echo 'Hi, Geekflare. Starting to build the app.'
8             }
9         }
10        stage('Test') {
11            steps {
12                input('Do you want to proceed?')
13            }
14        }
15        stage('Deploy') {
16            parallel {
17                stage('Deploy Start') {
18                    steps {
19                        echo 'Start the deploy..'
20                    }
21                }
22                stage('Deploying now') {
23                    agent {
24                        docker{
25                            reuseNode true
26                            image 'nginx'
27                        }
28                    }
29                }
30            }
31        }
32    }
33}
34
35    stage('Prod'){
36        steps{
37            echo 'App is prod ready'
38        }
39    }
40}
41}
42}
```

Below the code editor are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Pipeline configuration screen for the same job. The 'Pipeline' tab is selected. The pipeline script has been modified:

```
11    steps {
12        input('Do you want to proceed?')
13    }
14}
15    stage('Deploy') {
16        parallel {
17            stage('Deploy Start') {
18                steps {
19                    echo 'Start the deploy..'
20                }
21            }
22            stage('Deploying now') {
23                agent {
24                    docker{
25                        reuseNode true
26                        image 'nginx'
27                    }
28                }
29                steps{
30                    echo 'Docker created'
31                }
32            }
33        }
34    }
35    stage('Prod'){
36        steps{
37            echo 'App is prod ready'
38        }
39    }
40}
41}
42}
```

Below the code editor are 'Save' and 'Apply' buttons.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 01/09/23

The screenshot shows two consecutive screenshots of a Jenkins pipeline interface. Both screenshots are taken from a Windows desktop environment, as indicated by the taskbar at the bottom.

Screenshot 1: Pipeline Overview

- Header:** Jenkins, Search (CTRL+K), Notifications (1), Admin (admin), Log out.
- Breadcrumbs:** Dashboard > altaf_pipeline3 > Pipeline altaf_pipeline3
- Left Sidebar:** Status, Changes (Altaf Alam, docker), Build Now, Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax.
- Right Sidebar:** Edit description, Disable Project.
- Section:** Stage View - No data available. This Pipeline has not yet run.
- Bottom:** Build History (trend dropdown, Filter builds...), Permalinks.

Screenshot 2: Pipeline Console Output

- Header:** Jenkins, Search (CTRL+K), Notifications (1), Admin (admin), Log out.
- Breadcrumbs:** Dashboard > altaf_pipeline3 > #1
- Left Sidebar:** Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete build '#1', Replay, Pipeline Steps, Workspaces.
- Section:** Console Output - Started by user admin
org.codehaus.groovy.control.MultipleCompilationErrorsException: startup failed:
WorkflowScript: 24: Invalid agent type "docker" specified. Must be one of [any, label, none] @ line 24, column 25.
 docker{
 ^
1 error
at org.codehaus.groovy.control.ErrorCollector.failIfErrors(ErrorCollector.java:309)
at org.codehaus.groovy.control.CompilationUnit.applyToPrimaryClassNodes(CompilationUnit.java:1107)
at org.codehaus.groovy.control.CompilationUnit.doPhaseOperation(CompilationUnit.java:624)
at org.codehaus.groovy.control.CompilationUnit.processPhaseOperations(CompilationUnit.java:602)
at org.codehaus.groovy.control.CompilationUnit.compile(CompilationUnit.java:579)
at groovy.lang.GroovyClassLoader.doParseClass(GroovyClassLoader.java:323)
at groovy.lang.GroovyClassLoader.parseClass(GroovyClassLoader.java:293)
at org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.GroovySandbox\$Scope.parse(GroovySandbox.java:163)
at org.jenkinsci.plugins.workflow.cps.CpsGroovyShell.doParse(CpsGroovyShell.java:190)
at org.jenkinsci.plugins.workflow.cps.CpsGroovyShell.reparse(CpsGroovyShell.java:175)
at org.jenkinsci.plugins.workflow.cps.CpsFlowExecution.parseScript(CpsFlowExecution.java:563)
at org.jenkinsci.plugins.workflow.cps.CpsFlowExecution.start(CpsFlowExecution.java:513)

Conclusion :

Hence learn about the pipelines in Jenkins and running the scripts in jenkin.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

Experiment No 7

Aim : To understand Docker architecture and container life cycle, install docker, deploy container in docker.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO5 : To understand the concept of containerization and analyze the containerization of OS images and deployment of applications over Docker.

Theory :

What is Docker ?

Docker is a containerization platform that allows developers to package applications and their dependencies into a standardized unit called a container. These containers can run consistently across various environments, such as development, testing, and production, making it easier to build, ship, and deploy software.

At its core, Docker containers are lightweight, standalone, and executable packages that include everything needed to run an application, including code, runtime, libraries, and system tools. This encapsulation ensures that an application behaves the same way regardless of where it runs, whether it's on a developer's laptop, a testing server, or in the cloud.

Key concepts and components of Docker include:

1. Docker Engine: This is the core component of Docker that manages containers. It includes a server, a REST API, and a command-line interface (CLI). The Docker Engine is responsible for building, running, and managing containers.
2. Images: Docker images are read-only templates that define the application and its dependencies. Images serve as the basis for creating containers. They can be stored in a registry, such as Docker Hub, and shared among developers and teams.
3. Containers: Containers are instances of Docker images that are runnable and isolated from the host system and other containers. They encapsulate an application and its environment, ensuring consistency and portability.
4. Dockerfile: A Dockerfile is a text file that contains instructions for building a Docker image. It specifies the base image, adds application code, sets environment variables, and configures the container.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

5. Docker Compose: Docker Compose is a tool for defining and running multi-container applications. It allows developers to define the services, networks, and volumes in a single YAML file, simplifying the orchestration of complex applications.

Docker offers several advantages, including:

- Portability: Docker containers can run consistently across different environments, reducing "it works on my machine" issues.
- Isolation: Containers provide process and file system isolation, allowing multiple applications to run securely on the same host.
- Efficiency: Containers are lightweight and share the host OS kernel, resulting in minimal resource overhead and quick startup times.
- Scalability: Docker simplifies the scaling of applications by creating multiple containers from the same image.
- Version Control: Docker images are versioned, allowing for easy rollbacks and updates.

Docker Architecture:

Docker's architecture consists of several key components that work together to manage containers efficiently. Understanding this architecture is essential for effectively using Docker in containerization. Here's a simplified explanation:

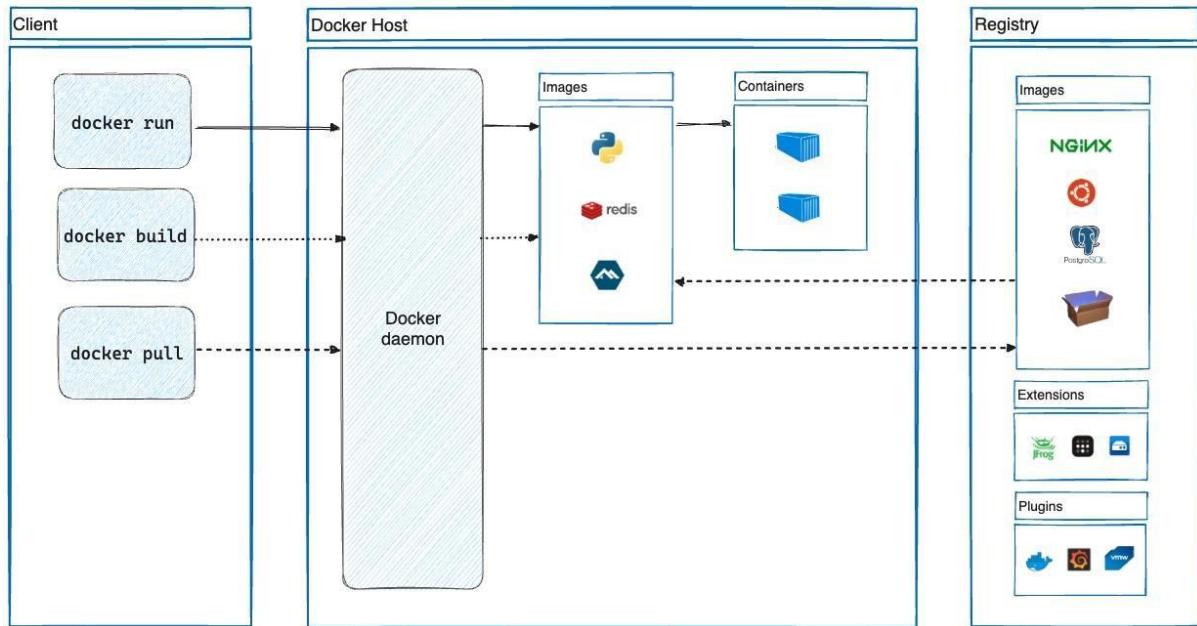
1. Docker Client: The Docker command-line interface (CLI) is the primary way users interact with Docker. It sends commands to the Docker daemon, which in turn communicates with the underlying Docker engine.
2. Docker Daemon: The Docker daemon (also known as the Docker engine) is responsible for managing containers on the host system. It listens for Docker API requests from the Docker client and handles container lifecycle operations.
3. Docker Images: Docker images are read-only templates that serve as the basis for creating containers. They contain the application code, libraries, dependencies, and configurations required to run an application. Images are stored in a local registry on the host system.
4. Docker Registry: Docker images can be stored and shared in registries like Docker Hub or private registries. Registries are repositories where users can upload and access images. When you run a container, Docker pulls the required image from the registry if it's not available locally.
5. Containerization: Containers are instances of Docker images that are isolated from each other and the host system. Each container runs as a separate process with its own file

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

system and network, allowing multiple containers to run on the same host without interfering with one another.

6. Docker Compose: Docker Compose is a tool for defining and managing multi-container applications. It uses a YAML file to define services, networks, and volumes, making it easier to manage complex applications with multiple containers.



Lifecycle of a Docker Container:

The lifecycle of a Docker container can be summarized in several stages:

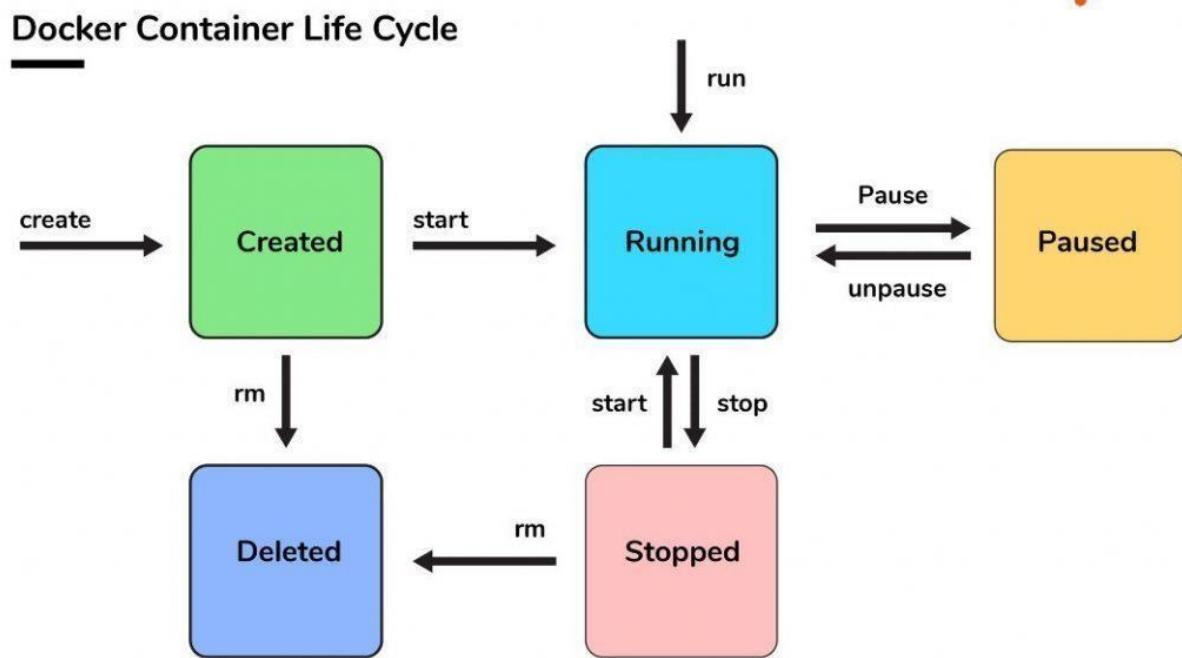
1. Creating a Container: You start by creating a container from a Docker image using the `'docker run'` command. This action creates a writable container layer on top of the image.
2. Running the Container: Once created, you can start the container with the `'docker start'` command. The container runs as a separate process on the host, executing the application defined in the image.
3. Interacting with the Container: You can interact with a running container through the `'docker exec'` command, which allows you to execute commands within the container. This is useful for debugging or configuring the container.
4. Stopping the Container: To stop a running container, you use the `'docker stop'` command. The container is gracefully terminated, and its resources are freed.
5. Restarting the Container: Containers can be restarted with the `'docker start'` command, either manually or automatically based on certain conditions, such as system restarts or failures.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

6. Pausing and Resuming: Docker allows you to pause and resume containers with the `docker pause` and `docker unpause` commands. Paused containers retain their state, including open network connections.

7. Removing the Container: When you're done with a container, you can remove it using the `docker rm` command. This action deletes the container and its associated filesystem.



Output:

Name : Altaf Alam

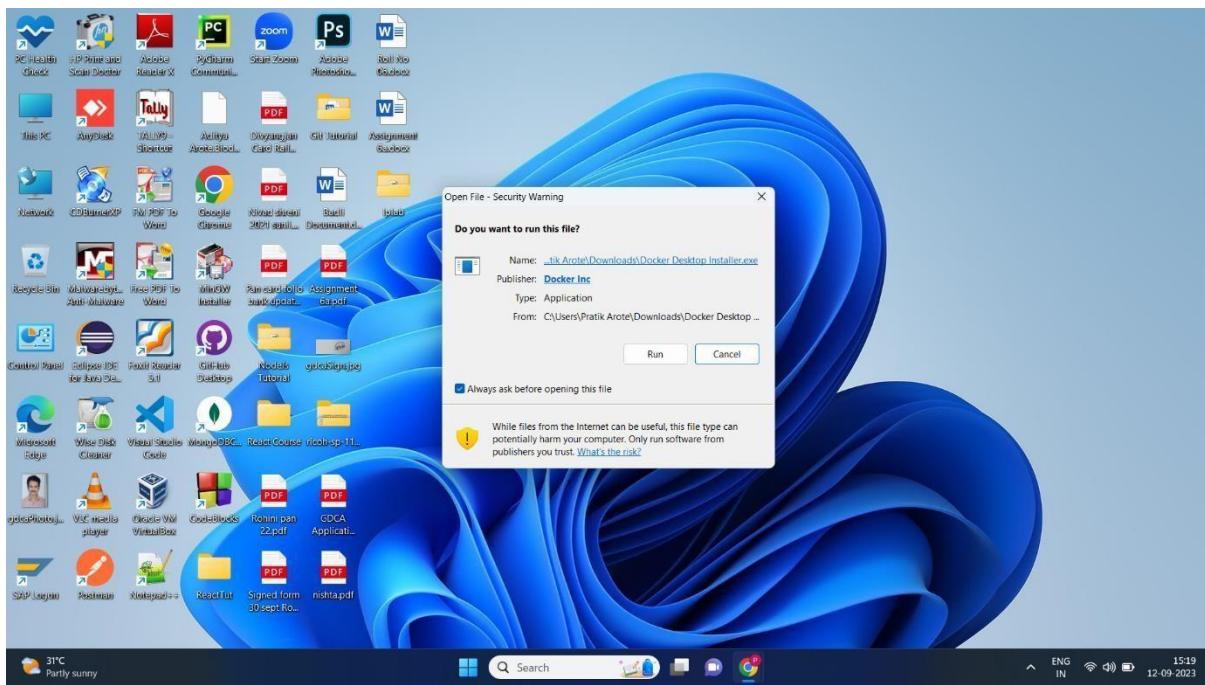
Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

The screenshot shows a web browser window with the URL docs.docker.com/desktop/install/windows-install/. The page title is "Install Docker Desktop on Windows". The left sidebar has a tree view with "Install on Windows" selected. The main content area contains a section titled "Docker Desktop for Windows" with a note about checksums and release notes. A "Docker Desktop terms" section states that commercial use requires a paid subscription. Below this is a "System requirements" section. The bottom of the page includes a cookie consent banner with "Accept All Cookies" and "Reject All" buttons.

The screenshot shows a Windows PowerShell window with the title "Windows PowerShell". It displays logs related to the Windows Subsystem for Linux (WSL). The logs show the user running commands like "wsl --status", "wsl --update", and "wsl -l -v". It also shows the user visiting the Microsoft Store to install distributions. The logs indicate that the WSL 2 kernel file was not found and that the user needs to run "wsl --update" to restore it. The logs also mention that the Windows Subsystem for Linux has no installed distributions and can be installed by visiting the Microsoft Store. The logs end with the message "Windows Subsystem for Linux has been installed". The bottom of the window includes a cookie consent banner with "Accept All Cookies" and "Reject All" buttons.

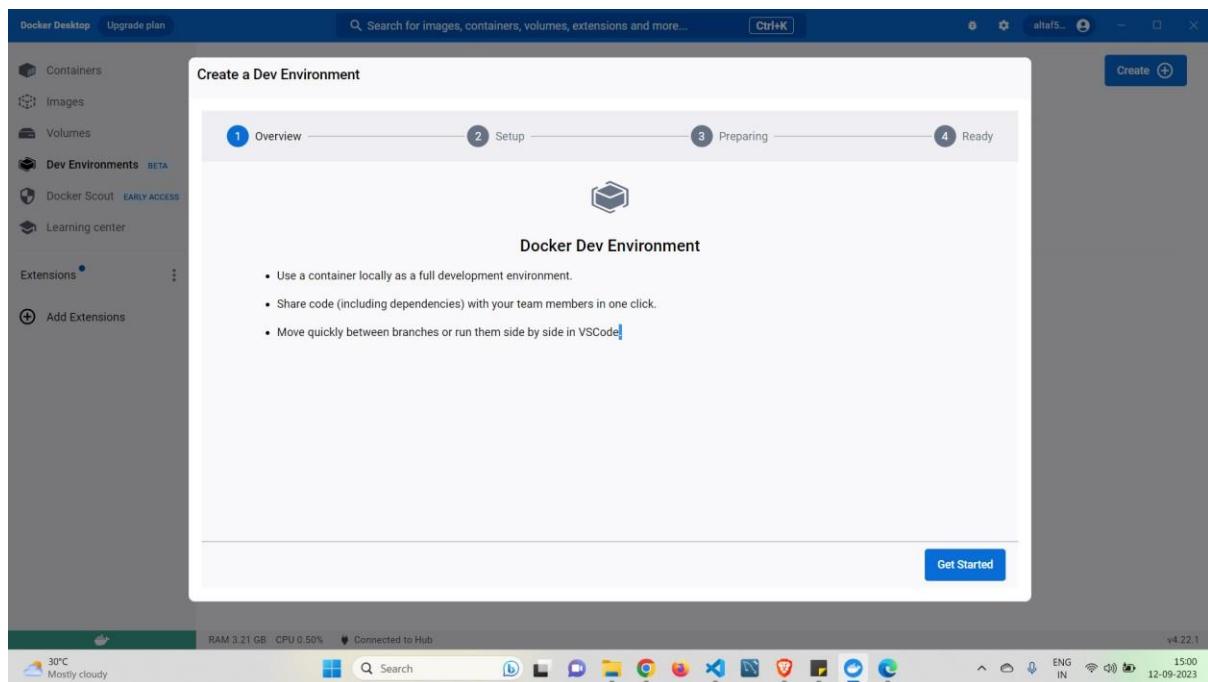
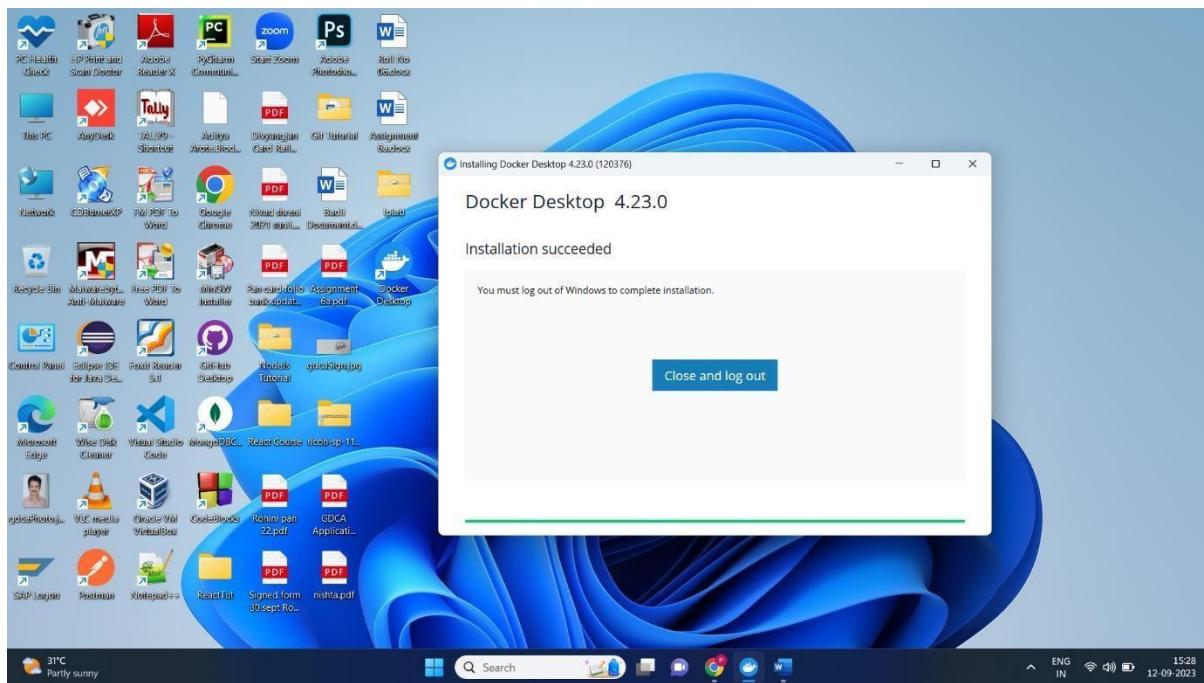
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23



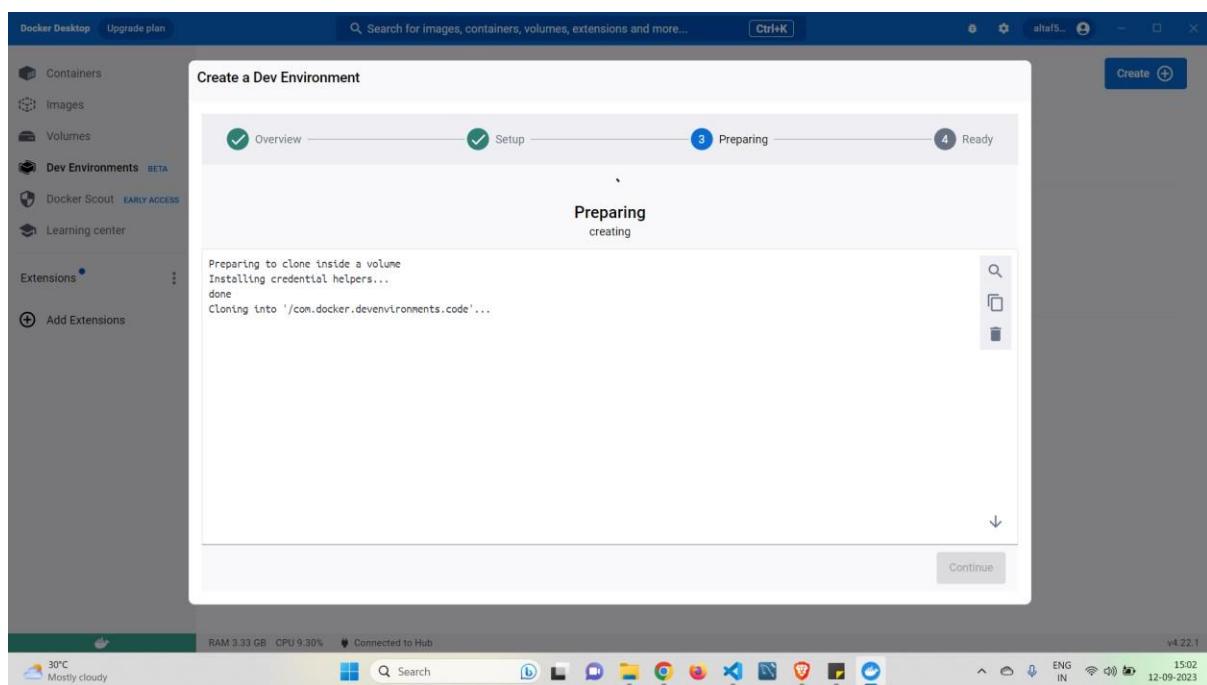
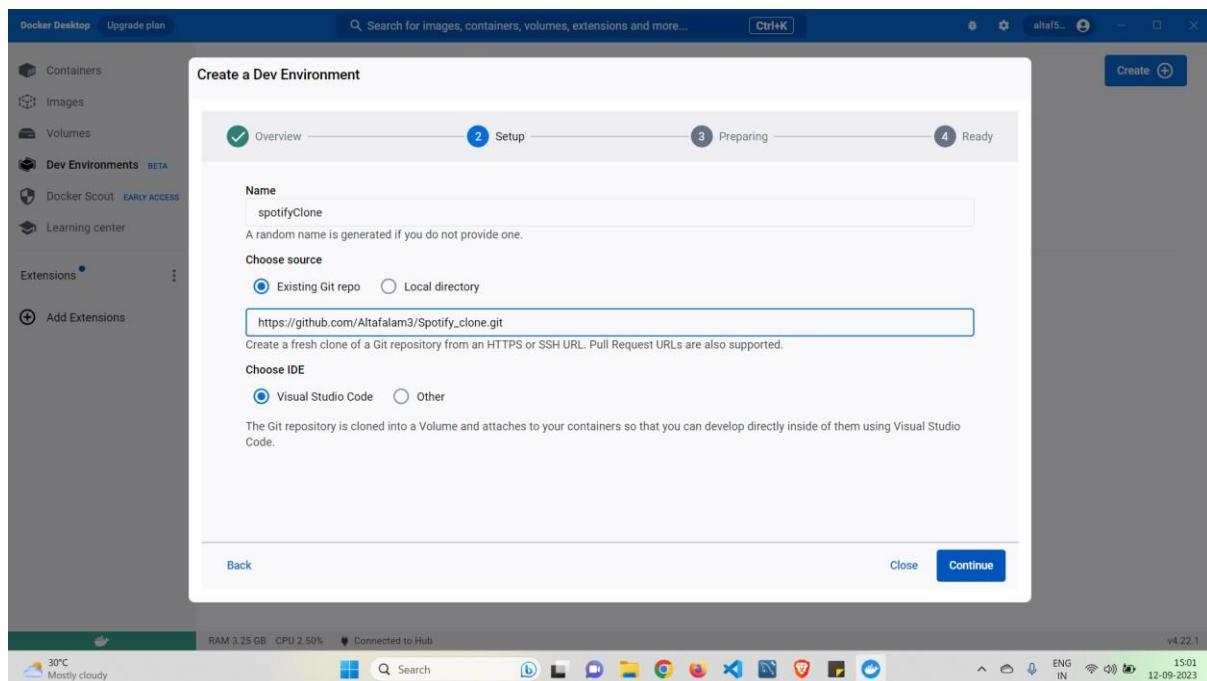
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23



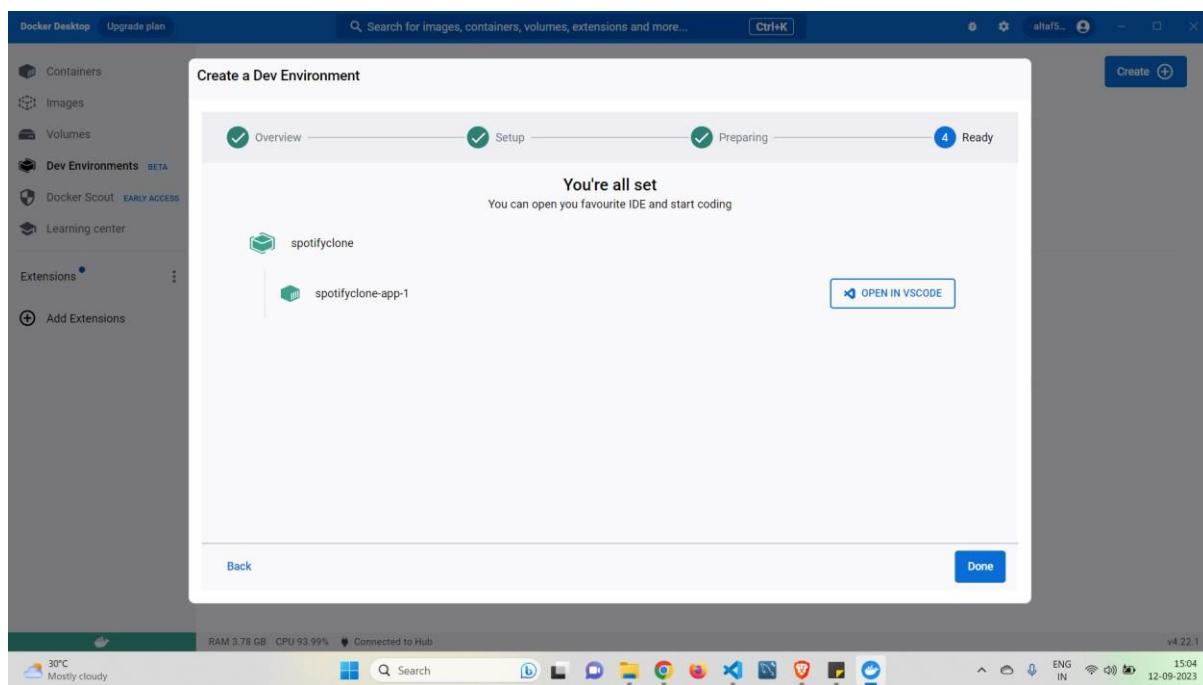
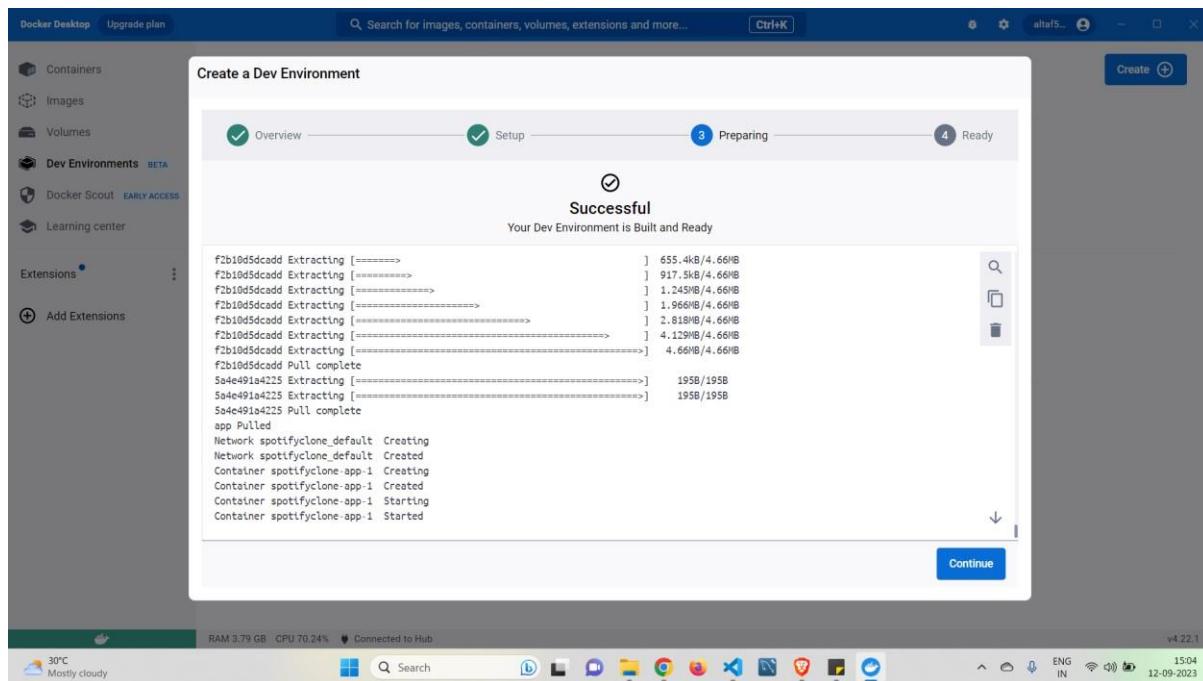
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23



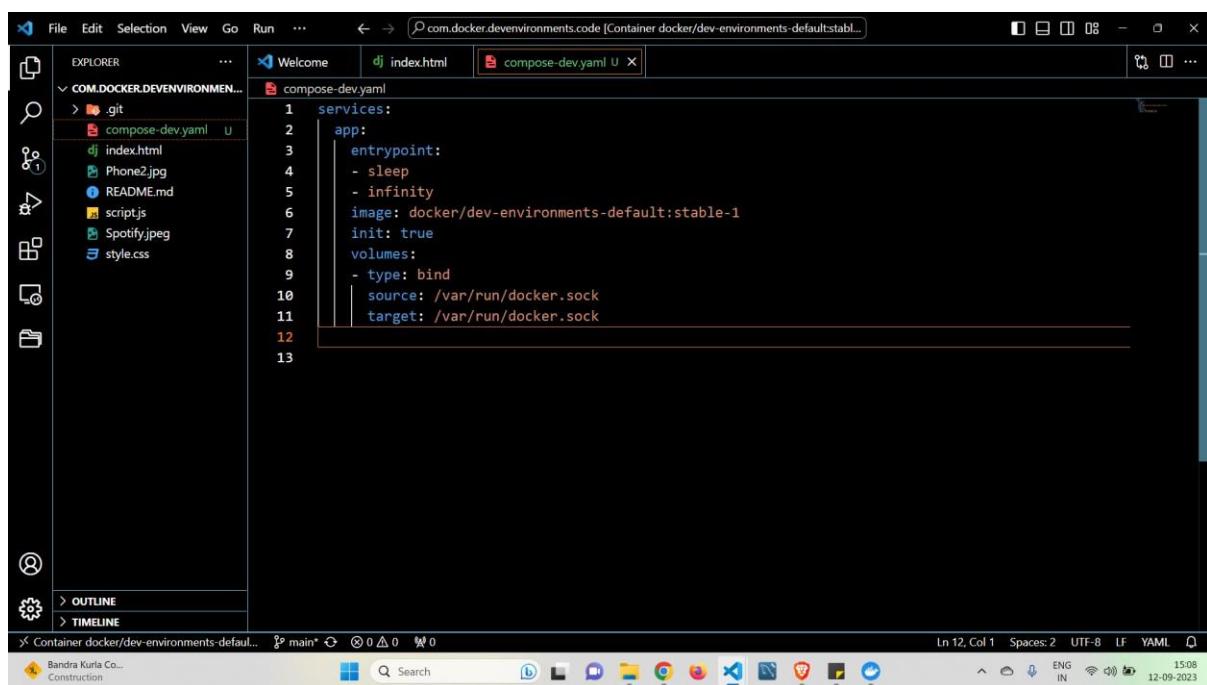
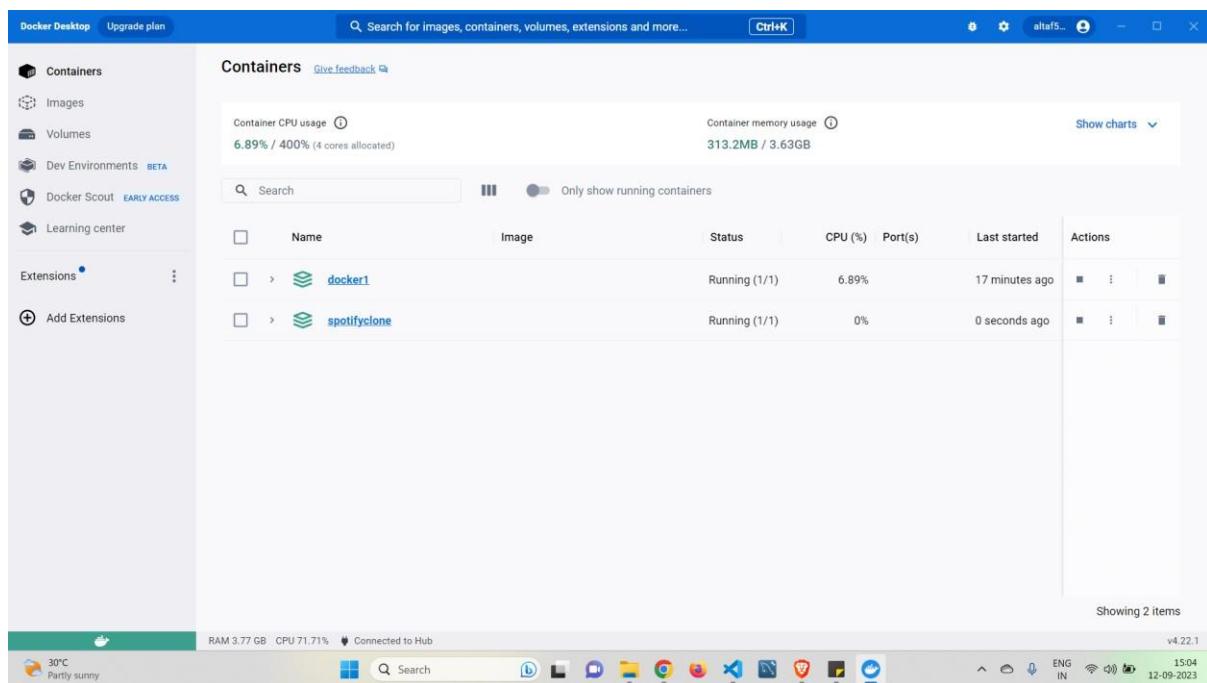
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 23/09/23

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a project structure with files like `compose-dev.yaml`, `index.html`, `Phone2.jpg`, `README.md`, `script.js`, `Spotify.jpeg`, and `style.css`. The main area displays the contents of `compose-dev.yaml`:

```
1 services:
2   app:
3     entrypoint:
4       - sleep
5       - infinity
6     image: docker/dev-environments-default:stable-1
7     init: true
8     volumes:
9       - type: bind
10      source: /var/run/docker.sock
11      target: /var/run/docker.sock
```

Below the code editor is a terminal window showing the output of a `docker pull httpd` command:

```
root@...:/com.docker.devenvironments.code# bash
root@...:/com.docker.devenvironments.code# 3+2
root@...:/com.docker.devenvironments.code# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
360eba32fa65: Pull complete
2852a695827e: Pull complete
b57c1299d233: Pull complete
45a0ea29816d: Pull complete
8c226ac2053e: Pull complete
Digest: sha256:0adbeffbe65176b93e4135f5c133072e2fb963187b00df565e6c73814894af5c
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
root@...:/com.docker.devenvironments.code#
```

The status bar at the bottom indicates the file is `Ln 6, Col 52 (40 selected)`, the encoding is `UTF-8`, and the date is `12-09-2023`.

Conclusion:

Here, we have studied about docker, its architecture and lifecycle of a docker container. Also we have successfully created and run a container in docker.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

Experiment No 8

Aim : To deploy static web application on docker.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO5 : To understand the concept of containerization and analyze the containerization of OS images and deployment of applications over Docker.

Theory :

In the ever-evolving world of web development and deployment, containers have gained immense popularity due to their efficiency, scalability, and portability. Docker, a leading containerization platform, has become the de facto choice for packaging and deploying applications. While Docker is frequently associated with complex microservices and multicontainer applications, it can also be used effectively for deploying simple static web applications. This article will delve into the theory and steps involved in deploying a static web application on Docker.

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications inside lightweight, portable containers. These containers encapsulate the application code, runtime, system tools, and libraries, ensuring that the application runs consistently in any environment.

Before we begin deploying a static web application on Docker, there are a few prerequisites to consider:

1. Docker Installation: Ensure that Docker is installed on your system. Docker provides a user-friendly installation process for various operating systems.
2. Static Web Application: You should have a static web application ready for deployment. A static web application typically consists of HTML, CSS, JavaScript, and other static assets.

Steps to Deploy a Static Web Application on Docker

1. Create a Dockerfile

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

A Dockerfile is a script containing a set of instructions that Docker will use to build a Docker image. For a static web application, the Dockerfile is straightforward. Here's a sample Dockerfile:

In this Dockerfile, we start with a base image that contains Node.js, set the working directory, copy the application files, expose port 80, define an environment variable, and finally, run the static web application.

2. Build a Docker Image

To build a Docker image from the Dockerfile, navigate to the directory containing the Dockerfile and the web application files and execute the following command:

```
```bash docker build -t my-static-
web-app .
```

```

This command will create a Docker image tagged as "my-static-web-app." Make sure to replace it with your preferred image name.

3. Run a Docker Container

Now that we have a Docker image, we can run a Docker container based on that image. Use the following command:

```
```bash docker run -p 8080:80 my-static-
web-app
```
```

This command will start a Docker container from the "my-static-web-app" image, mapping port 8080 on your host machine to port 80 inside the container. You can access your static web application by opening a web browser and navigating to <http://localhost:8080>.

Benefits of Docker for Deploying Static Web Applications

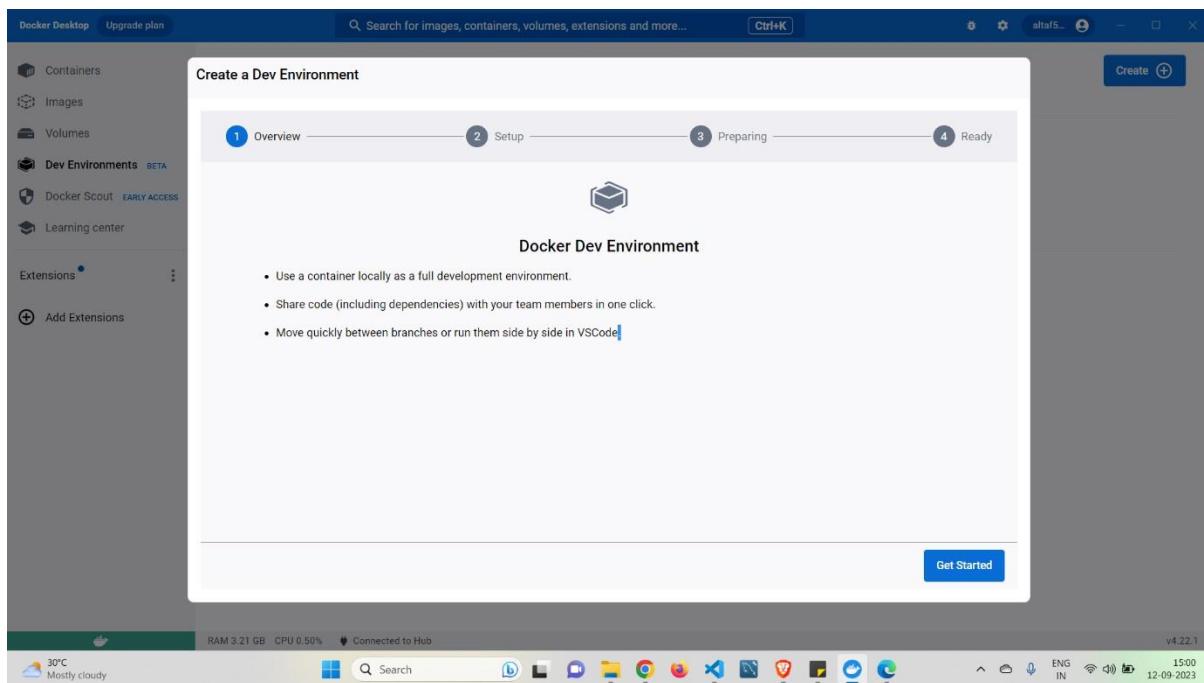
1. Portability: Docker containers encapsulate the application and its dependencies, ensuring consistent behavior across different environments.
2. Isolation: Each container operates independently, avoiding conflicts and ensuring a clean environment.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

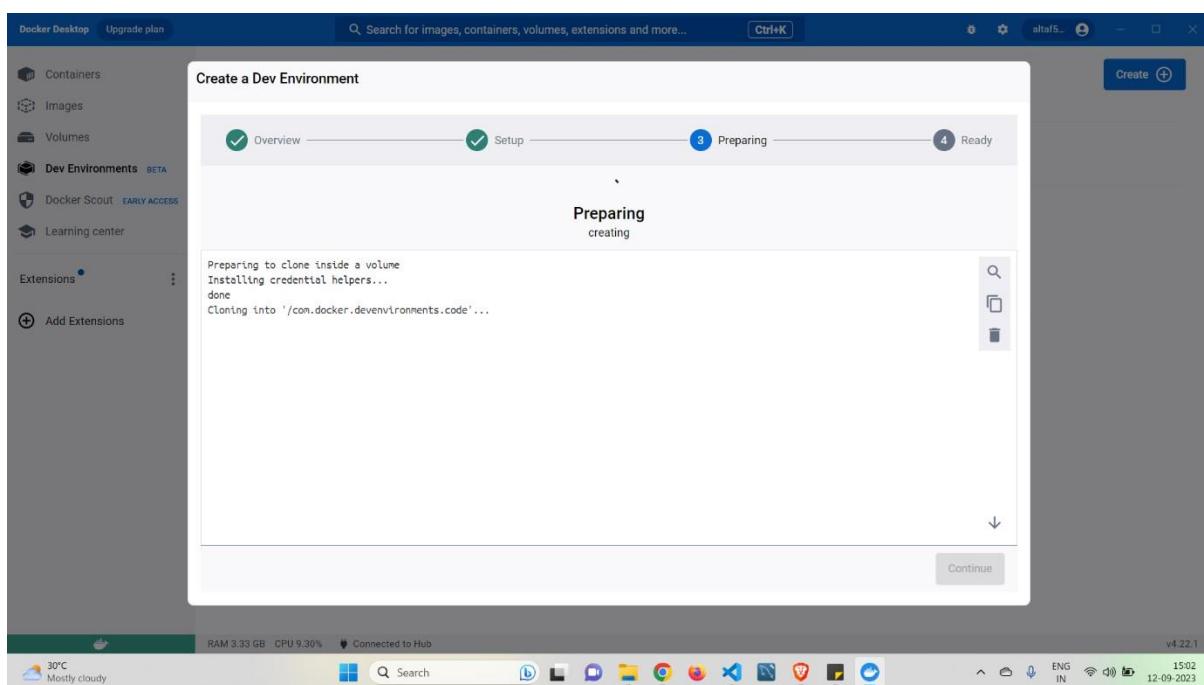
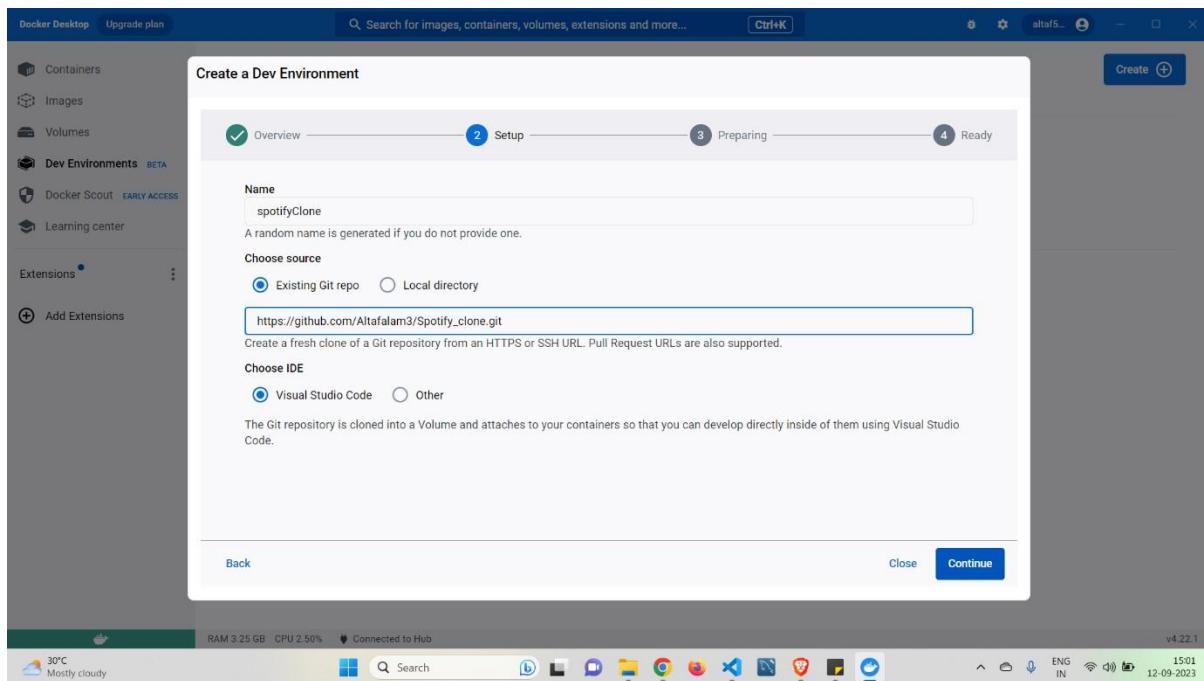
3. Scalability: Docker simplifies scaling by allowing the deployment of multiple containers and load balancing between them.
4. Resource Efficiency: Containers are lightweight and share the host OS kernel, leading to efficient resource utilization.
5. Easy Version Control: Images and containers can be versioned, making it easy to roll back to previous states if needed.
6. Security: Docker provides security features to isolate containers and control access.
7. Community and Ecosystem: Docker has a vast community and ecosystem, with a wealth of pre-built images and tools.

Output:



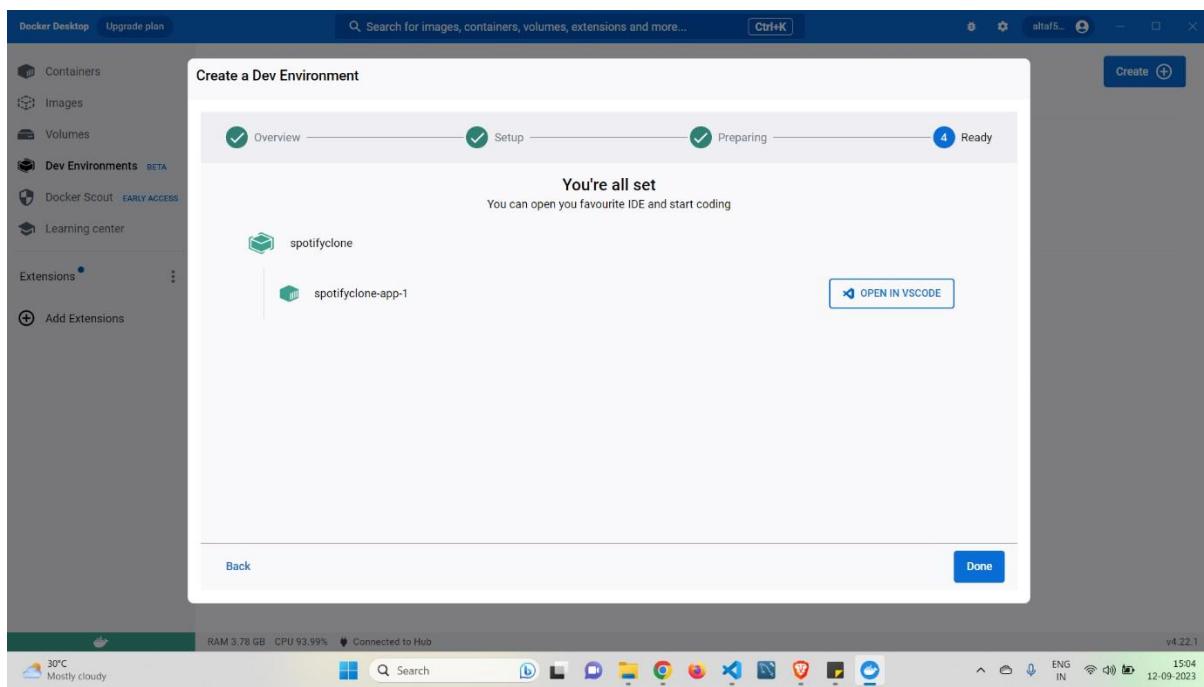
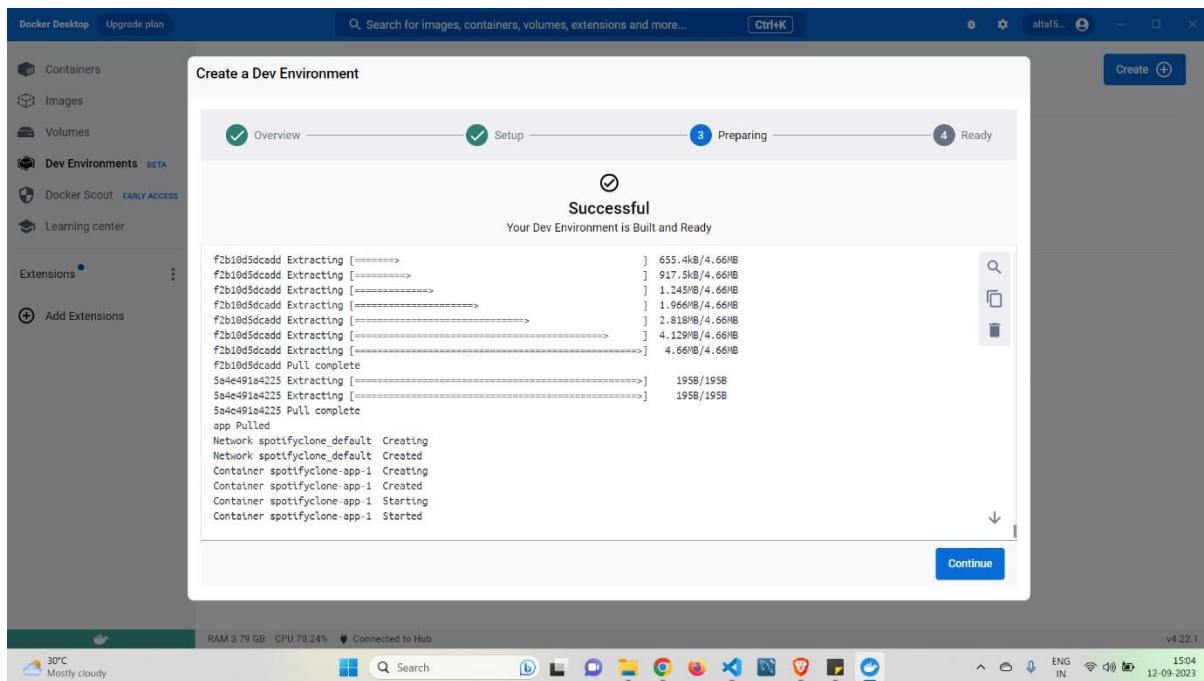
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

The screenshot shows the Docker Desktop interface. On the left sidebar, there are links for Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), and Learning center. Under Dev Environments, two environments are listed: 'spotifyclone' (RUNNING) and 'docker1' (RUNNING). Each environment has a 'Create' button. The Docker Desktop status bar at the bottom indicates RAM 3.78 GB, CPU 47.03%, Connected to Hub, v4.22.1, 30°C Partly sunny, ENG IN, and 15:04 12-09-2023.

The screenshot shows the Docker Desktop interface. On the left sidebar, there are links for Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), and Learning center. Under Containers, a table lists two running containers: 'docker1' (Running (1/1), 6.89% CPU, 17 minutes ago) and 'spotifyclone' (Running (1/1), 0% CPU, 0 seconds ago). The Docker Desktop status bar at the bottom indicates RAM 3.77 GB, CPU 71.71%, Connected to Hub, v4.22.1, 30°C Partly sunny, ENG IN, and 15:04 12-09-2023.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

The screenshot shows the Visual Studio Code interface with the following details:

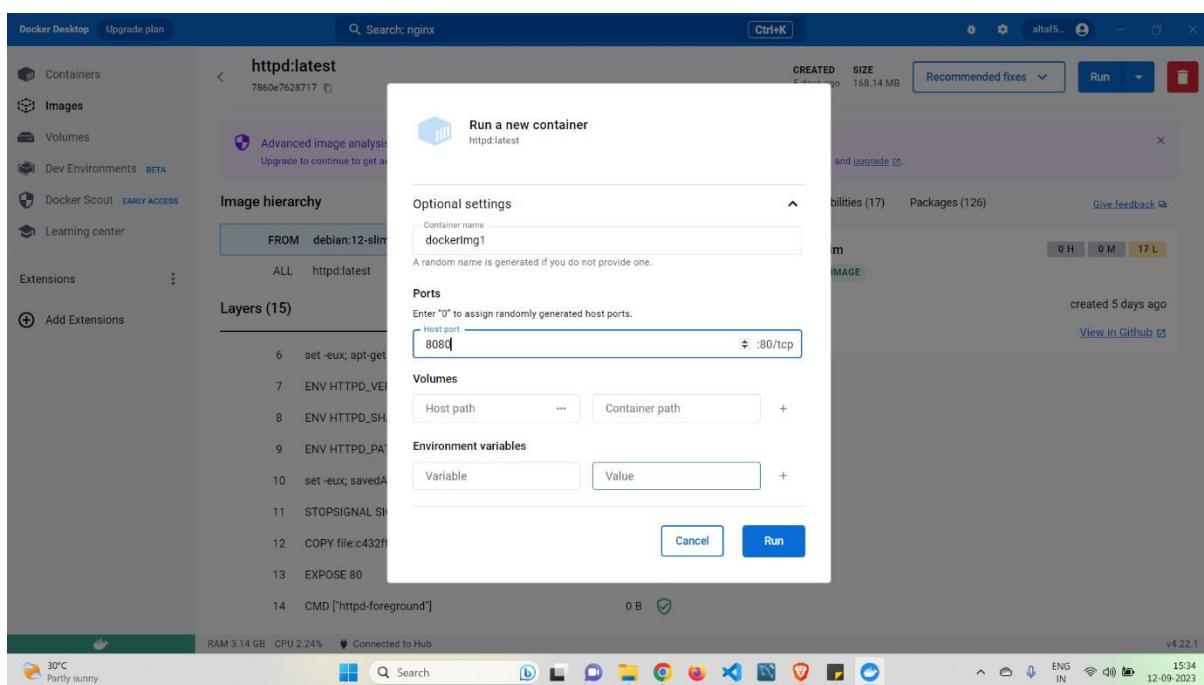
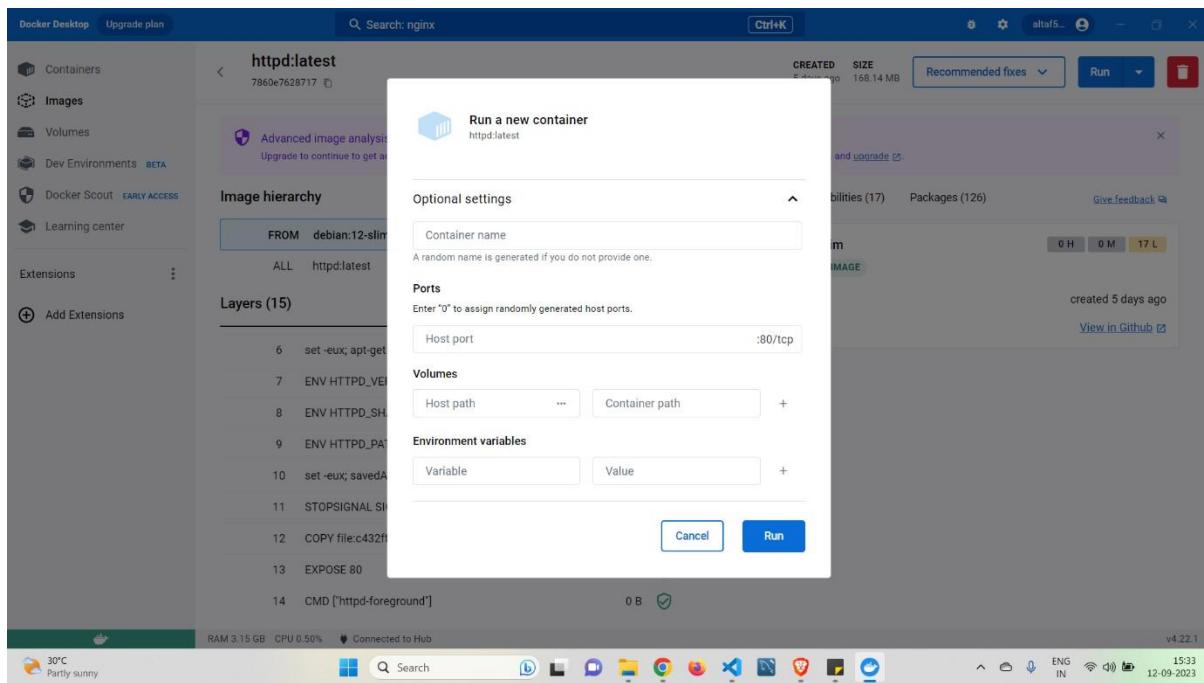
- File Explorer:** Shows a folder named "COM.DOCKER.DEVENVIRONMEN..." containing ".git", "compose-dev.yaml", "index.html", "Phone2.jpg", "README.md", "script.js", "Spotify.jpeg", and "style.css".
- Editor:** Displays the contents of "compose-dev.yaml".
- Terminal:** Shows the command "docker pull httpd" being run.
- Bottom Status Bar:** Includes file paths like "Container docker/dev-environments-default...", line numbers (Ln 12, Col 1), and encoding information (UTF-8, LF, YAML).

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "COM.DOCKER.DEVENVIRONMEN..." containing ".git", "compose-dev.yaml", "index.html", "Phone2.jpg", "README.md", "script.js", "Spotify.jpeg", and "style.css".
- Editor:** Displays the contents of "compose-dev.yaml".
- Terminal:** Shows the command "docker pull httpd" being run, followed by a log of image pull attempts for "httpd".
- Bottom Status Bar:** Includes file paths like "Container docker/dev-environments-default...", line numbers (Ln 6, Col 52), and encoding information (UTF-8, LF, YAML).

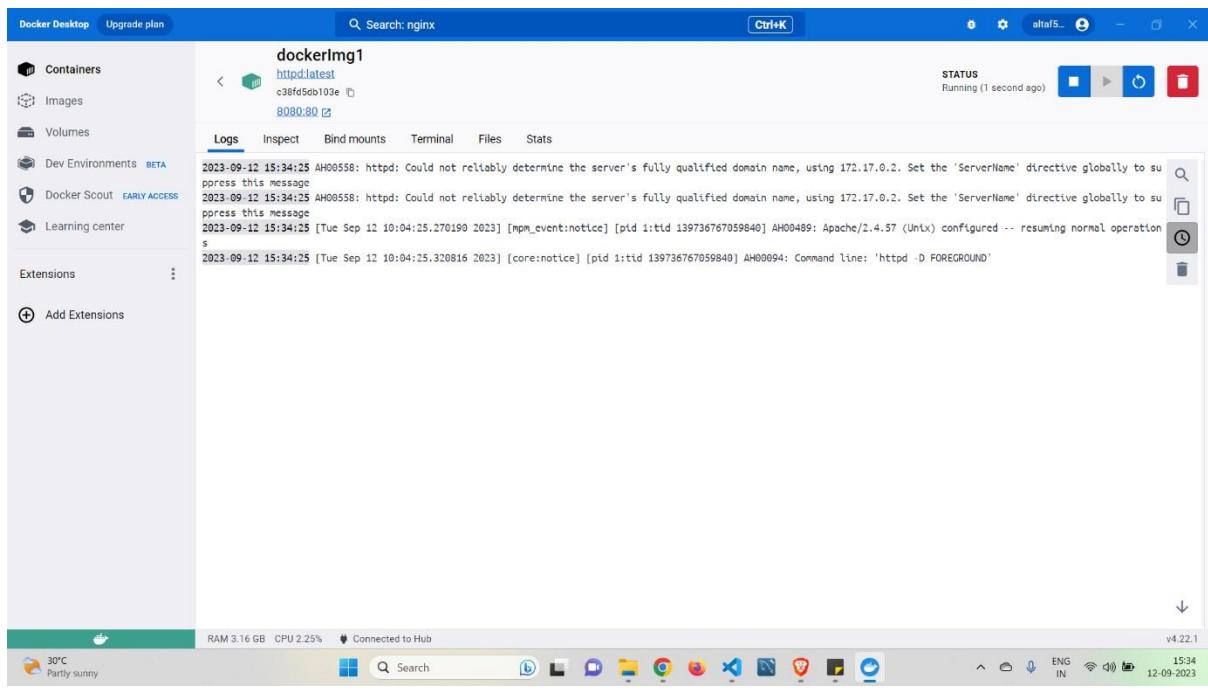
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

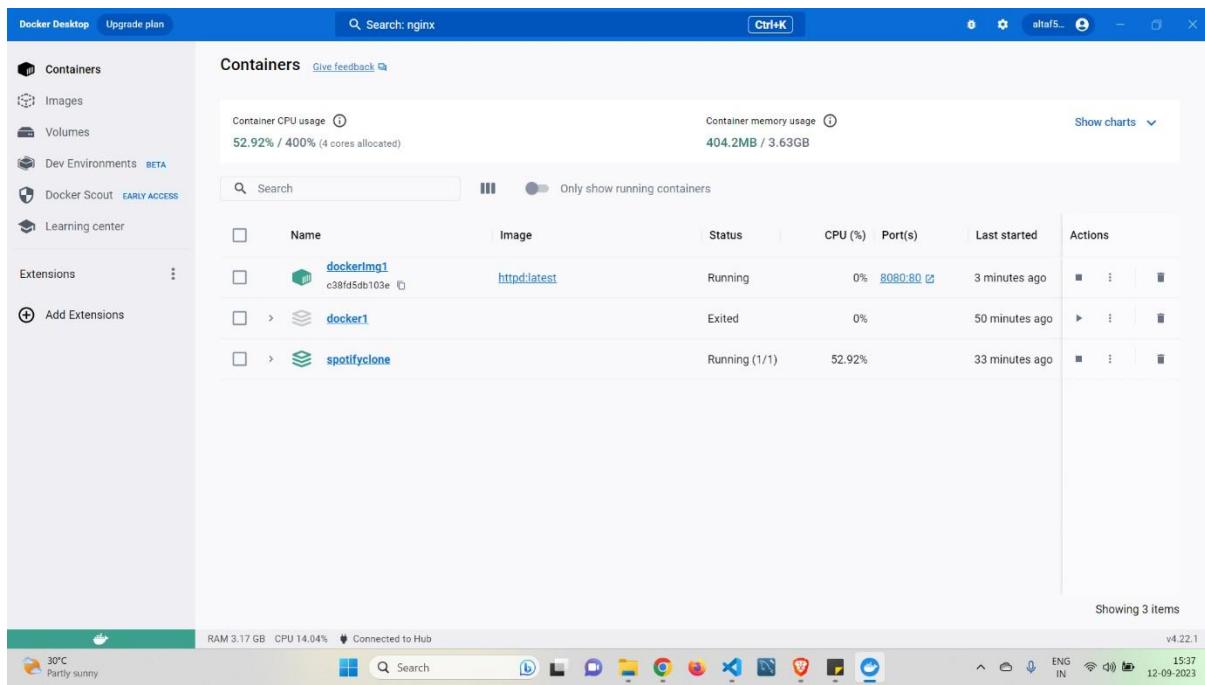


It works!



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Conclusion :

In conclusion, deploying a static web application on Docker is a straightforward yet powerful way to leverage containerization for web development. Docker simplifies the process of packaging and deploying your application, providing the benefits of portability, isolation, scalability, and resource efficiency. With Docker, you can streamline the deployment of static web applications and ensure a consistent experience across different environments.

Experiment No 9

Aim : Installation of nagios on ubuntu system.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO5 : To understand concept of containerization and analyze the containerization of os images and deployment of applications over docker.

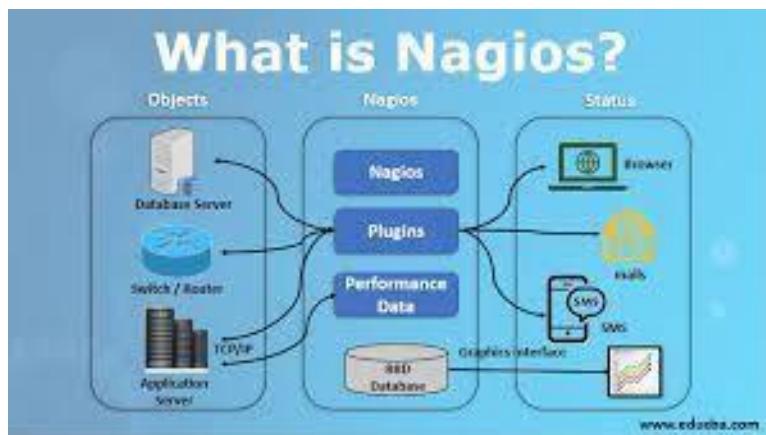
Theory :

What is Nagios?

Nagios is an [open source](#) IT system monitoring tool. It was designed to run on the [Linux](#) operating system and can monitor devices running Linux, Windows and Unix [OSes](#).

Nagios software runs periodic checks on critical parameters of application, network and server resources. For example, Nagios can monitor [memory](#) use, disk use and microprocessor load, as well as the number of currently running [processes](#) and log files. Nagios also can monitor services such as Simple Mail Transfer Protocol ([SMTP](#)), [Post Office Protocol 3](#), Hypertext Transfer Protocol ([HTTP](#)) and other common network protocols. Nagios initiates active checks, while passive checks come from external applications connected to the monitoring tool.

Originally released in 1999 as NetSaint, Nagios was developed by Ethan Galstad and subsequently refined by numerous contributors as an open source project. Nagios Enterprises, a company based around the Nagios Core technology, offers multiple products, such as Nagios XI, Log Server, Network Analyzer and Fusion.



How Nagios works

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23

Users can choose to work in the [command-line interface](#) or select a web-based graphical user interface in some versions of Nagios and from third parties. Nagios' dashboard provides an overview of the critical parameters monitored on assets.

Based on the parameters and thresholds defined, Nagios can send out alerts if critical levels are reached. These notifications can be sent through email and text messages. An authorization system enables administrators to restrict access.

Nagios runs both agent-based and [agentless](#) configurations. Independent agents are installed on any hardware or software system to collect data that is then reported back to the management server. Agentless monitoring uses existing protocols to emulate an agent. Both approaches can monitor file system use, OS metrics, service and process states. Examples of Nagios agents include Nagios Remote Data Processor (NRDP), Nagios Cross Platform Agent and NSClient++.

Nagios plugins

Nagios can also run remote scripts and plugins using the Nagios Remote Plugin Executor (NRPE) agent. NRPE enables remote monitoring of system metrics such as system load, memory and disk use. It consists of the check_nrpe plugin, which is stored on the local monitoring machine, and NRDP, which runs on the remote machine. Nagios uses a plugin to consolidate data from the NRPE agent before it goes to the management server for processing. NRPE can also communicate with Windows agents to [monitor Windows machines](#).

Nagios supports plugins that are stand-alone add-ons and extensions so users can define targets and which target parameters to monitor. Nagios plugins process command-line arguments and communicate commands with Nagios Core.

There are around 50 plugins developed and maintained by Nagios, while there are over 3,000 from the community. These plugins are categorized into lists including hardware, software, cloud, OSes, security, log files and network connections. As an example, when used in conjunction with environmental-sensing systems, a Nagios plugin can share data on environmental variables, such as temperature, humidity or barometric pressure.

Nagios tools

Nagios has proven popular among small and large businesses, as well as [internet service providers](#), educational institutions, government agencies, healthcare institutions, manufacturing companies and financial institutions.

Users can choose among free and paid options, depending on the needed services and support.

Nagios Core

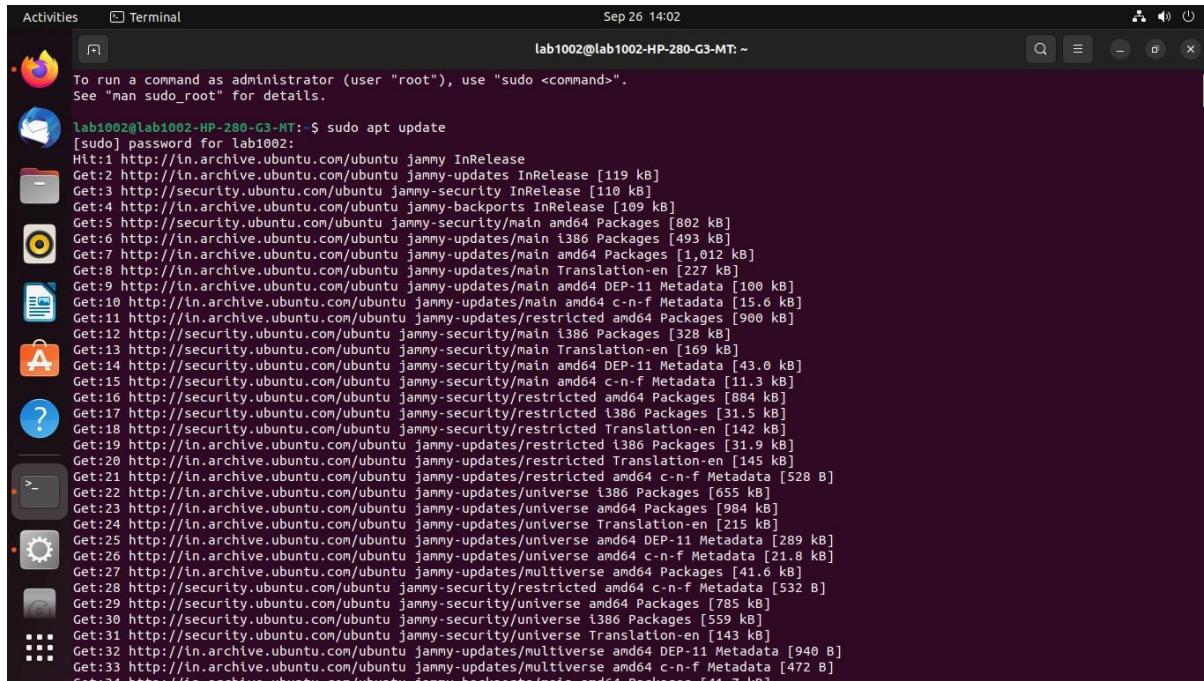
The service that was originally known as Nagios is now referred to as Nagios Core. Core is freely available as an open source monitoring software for IT systems, networks and infrastructure. Core contains a wide array of infrastructure monitoring through allowing plugins to extend its monitoring capabilities. It is the base for paid Nagios monitoring systems.

Nagios Core has an optional web interface, which displays network status, notifications and log files. Core can notify its user when there are server or host issues. Additionally, Core can monitor network services such as SMTP, HTTP and [Ping](#).

Name : Altaf Alam

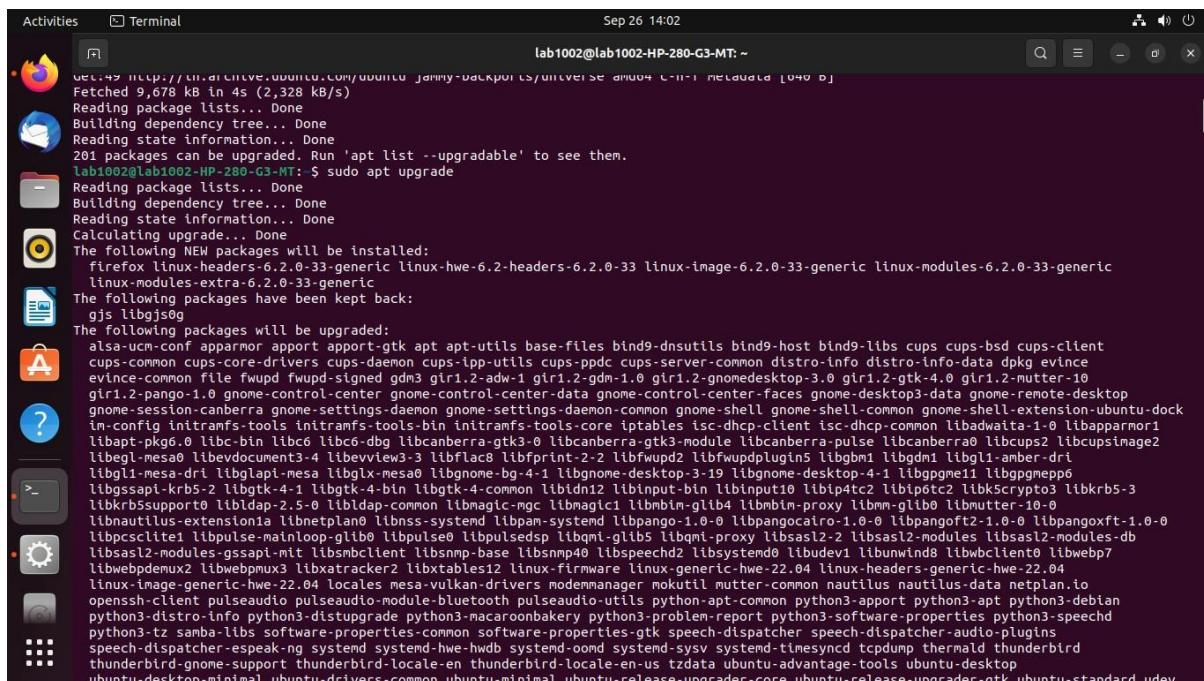
Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23

Installation in Ubuntu:



```
Activities Terminal Sep 26 14:02
lab1002@lab1002-HP-280-G3-MT: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

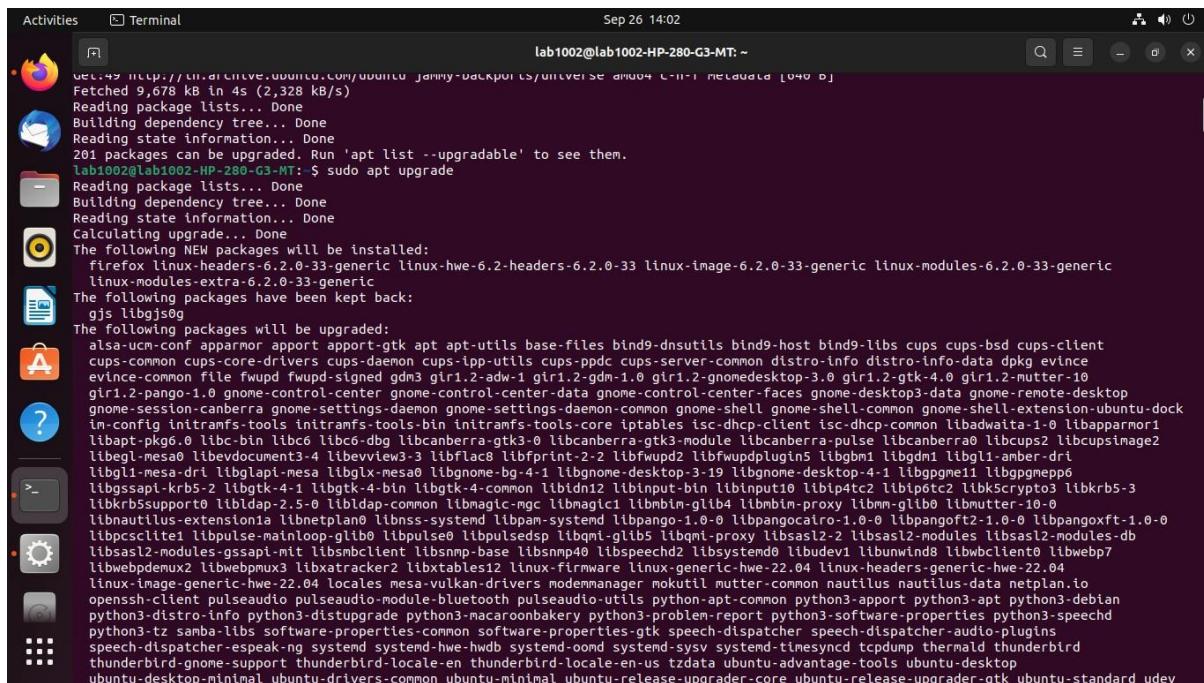
[sudo] password for lab1002:
Hit:: http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get::2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get::3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get::4 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get::5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [802 kB]
Get::6 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [493 kB]
Get::7 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,012 kB]
Get::8 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [227 kB]
Get::9 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [100 kB]
Get::10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [15.6 kB]
Get::11 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [900 kB]
Get::12 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [328 kB]
Get::13 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [169 kB]
Get::14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.0 kB]
Get::15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [11.3 kB]
Get::16 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [884 kB]
Get::17 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [31.5 kB]
Get::18 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [142 kB]
Get::19 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [31.9 kB]
Get::20 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [145 kB]
Get::21 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [528 B]
Get::22 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [655 kB]
Get::23 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [984 kB]
Get::24 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [215 kB]
Get::25 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [289 kB]
Get::26 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [21.8 kB]
Get::27 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [41.6 kB]
Get::28 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [532 B]
Get::29 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [785 kB]
Get::30 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [559 kB]
Get::31 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [143 kB]
Get::32 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get::33 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
```



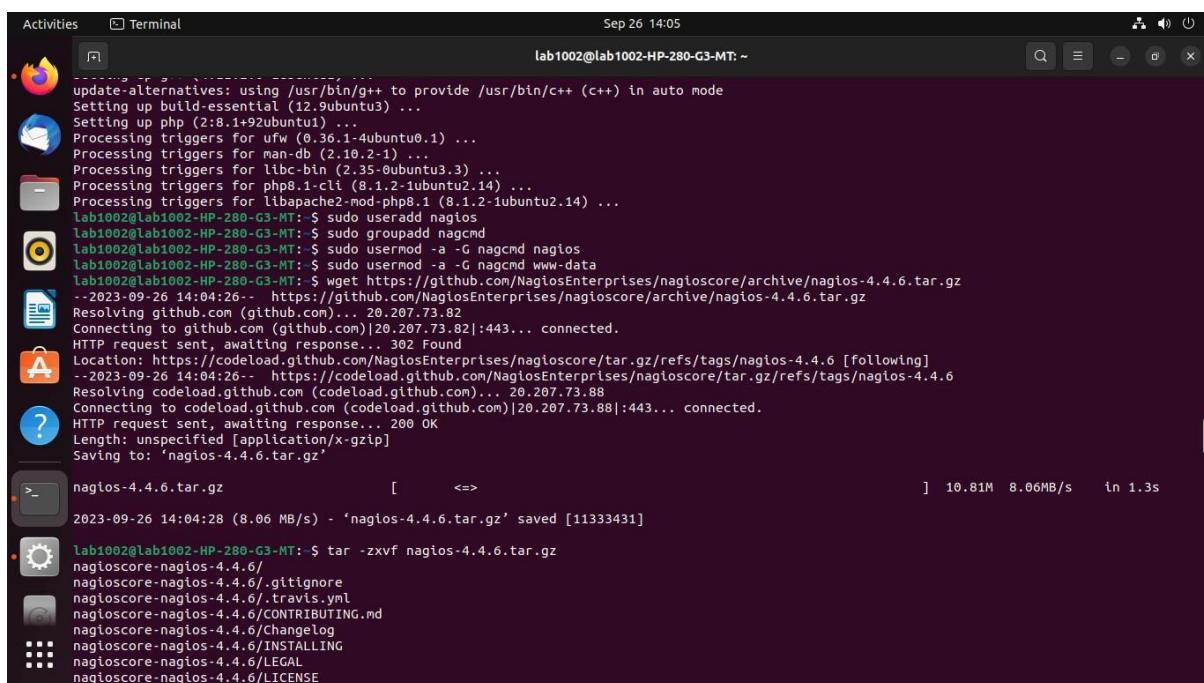
```
Activities Terminal Sep 26 14:02
lab1002@lab1002-HP-280-G3-MT: ~
Fetched 9,678 kB in 4s (2,328 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
201 packages can be upgraded. Run 'apt list --upgradable' to see them.
lab1002@lab1002-HP-280-G3-MT: ~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  firefox linux-headers-6.2.0-33-generic linux-hwe-6.2-headers-6.2.0-33 linux-image-6.2.0-33-generic linux-modules-6.2.0-33-generic
  linux-modules-extra-6.2.0-33-generic
The following packages have been kept back:
  gjs libgjs0g
The following packages will be upgraded:
  alsu-ucm-conf apparmor apt apt-utils base-files bind9-dnsutils bind9-host bind9-libs cups cups-bsd cups-client
  cups-common cups-core-drivers cups-daemon cups-ipp-utils cups-ppdc cups-server-common distro-info distro-info-data dpkg evince
  evince-common file fwupd-signed gdm3 gir1.2-adw-1 gir1.2-gdm-1.0 gir1.2-gnomedesktop-3.0 gir1.2-gtk-4.0 gir1.2-mutter-10
  gir1.2-pango-1.0 gnome-control-center gnome-control-center-data gnome-control-center-faces gnome-desktop3-data gnome-remote-desktop
  gnome-session-canberra gnome-settings-daemon gnome-shell gnome-shell-common gnome-shell-extension-ubuntu-dock
  im-config initramfs-tools initramfs-tools-bin initramfs-tools-core iptables isc-dhcp-client isc-dhcp-common libbadwaiata-1-0 libapparmor1
  libapt-pkg6.0 libc-bin libc6 libc6-0b9 libcanberra-gtk3-0 libcanberra-gtk3-module libcanberra-pulse libcanberra libcupsysimage2
  libegl-mesa0 libevdevdocument3-4 libevdevv3-3 libflac8 libgprint-2-2 libfwupd libfwupdplugin5 libgbm libgl1-amber-dri
  libgbm-mesa-dri libglapi-mesa libgbx-mesa0 libgnome-bg-4-1 libgnome-desktop-3-19 libgnome-desktop-4-1 libgbpm1 libgbpmep0
  libgbssapi-krb5-2 libgbt4-4-1 libgbt4-4-bin libgbt4-4-common libidn12 libinput-bin libinput10 libl10n libl10n-qt libl10n-qt5
  libkrb5support0 libldap-2.5-0 libldap-common libmagic1 libmbin1 libmbin4 libmbin-proxy libmm-glib0 libmutter-10-0
  libnautilus-extension1a libnetplan0 libnss-systemd libpam-systemd libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpangoft2-1.0-0
  libpcsc-lite1 libpulse-mainloop-glib libpulse0 libpulsesdp libqmi-libs libqmi-proxy libssasl2-2 libssasl2-modules libssasl2-modules-db
  libssasl2-modules-gssapi-mit libsmclient libsnmp-base libsnmp40 libspeeched2 libsystemd libudev1 libunwind8 libwbclient0 libwebp7
  libwebpdemux2 libwebpmux3 libxatracker2 libxtables12 linux-firmware linux-generic-hwe-22.04 linux-headers-generic-hwe-22.04
  linux-image-generic-hwe-22.04 locales mesa-vulkan-drivers nodemanager mokutil mutter-common nautilus nautilus-data netplan.io
  openssh-client pulseaudio pulseaudio-module-bluetooth pulseaudio-utility python3-common python3-apport python3-apt python3-debian
  python3-distro-info python3-distroupgrade python3-macaroonbakery python3-problem-report python3-software-properties python3-speechd
  python3-tz samba-libs software-properties-common software-properties-gtk speech-dispatcher speech-dispatcher-audio-plugins
  speech-dispatcher-espeak-ng systemd systemd-hwe-hwdb systemd-oomd systemd-sysv systemd-timesyncd tcpdump thermald thunderbird
  thunderbird-gnome-support thunderbird-locale-en thunderbird-locale-en-us tzdata ubuntu-advantage-tools ubuntu-desktop
  ubuntu-desktop-minimal ubuntu-drivers-common ubuntu-minimal ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk ubuntu-standard udev
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23



```
Activities Terminal Sep 26 14:02
lab1002@lab1002-HP-280-G3-MT: ~
[1]:49 wget://ui.dl.cdnlf.ubuntu.com/ubuntu_jammy-updates/universe_amd64/tarball [0+0 B]
Fetched 9,678 kB in 4s (2,328 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
201 packages can be upgraded. Run 'apt list --upgradable' to see them.
lab1002@lab1002-HP-280-G3-MT: $ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  firefox linux-headers-6.2.0-33-generic linux-hwe-6.2-headers-6.2.0-33 linux-image-6.2.0-33-generic linux-modules-6.2.0-33-generic
  linux-modules-extra-6.2.0-33-generic
The following packages have been kept back:
  gjs libgjs0g
The following packages will be upgraded:
  alsu-ucm-conf apparmor apport apport-gtk apt apt-utils base-files bind9-host bind9-libs cups cups-bsd cups-client
  cups-common cups-core-drivers cups-daemon cups-ipp cups-ppdc cups-server-common distro-info distro-info-data dpkg evince
  evince-common file fwupd fwupd-signed gdm3 gir1.2-adw-1 gir1.2-gdm-1.0 gir1.2-gnomedesktop-3.0 gir1.2-gtk-4.0 gir1.2-mutter-10
  gir1.2-pango-1.0 gnome-control-center gnome-control-center-data gnome-control-center-faces gnome-desktop3-data gnome-remote-desktop
  gnome-session-canberra gnome-settings-daemon gnome-settings-daemon-common gnome-shell gnome-shell-common gnome-shell-extension-ubuntu-dock
  in-config initramfs-tools initramfs-tools-bin initramfs-tools-core iptables isc-dhcp-client isc-dhcp-common libadwaita-1.0 libapparmor1
  libapt-pkg6.0 libc-bin libc6 libc6-dbg libcanberra-gtk3-0 libcanberra-pulse libcanberra0 libcupsys2 libcupsysimage2
  libegl-mesa0 libevedocument3-4 libevview3-3 libflac8 libfprint-2-2 libfwupd2 libfwupdplugin5 libgbm1 libgdi1 libgl1-amber-dri
  libgl1-mesa-dri libglapi-mesa libglx-mesa0 libgnome-bg-4-1 libgnome-desktop-3-19 libgnome-desktop-4-1 libgpgrme1 libgpgrmepp0
  libgssapi-krb5-2 libgtk-4-1 libgtk-4-2 libgtk-4-common libidn12 libinput-bin libinput10 libip4tc2 libip6tc2 libk5crypto3 libkrb5-3
  libkrb5support0 libldap-2.5-0 libldap-common libmagic-mgc libmbin-glib4 libmbin-proxy libmm-glib0 libmutter-10-0
  libnautilus-extension1a libnetplan1 libnss-systemd libpam-systemd libpango-1.0-0 libpangocaliro-1.0-0 libpangoft2-1.0-0 libpangoftx-1.0-0
  libpcsc-lite1 libpulse-mainloop-glib0 libpulsedsp libqmi-glib5 libqmi-proxy libtasl2-2 libtasl2-modules libtasl2-modules-db
  libtasl2-modules-gssapi-mit libusbclient0 libusbnmp40 libspeеч2 libsystemd libubuenv1 libuwind8 libwbcclient0 libwebp7
  libwebpdmux2 libwebpnmux3 libxatracker2 libxtables12 linux-firmware linux-generic-hwe-22.04 linux-headers-generic-hwe-22.04
  linux-image-generic-hwe-22.04 locales mesa-vulkan-drivers modemmanager mokutil mutter-common nautilus nautilus-data tptplan.io
  openssh-client pulseaudio pulseaudio-module-bluetooth pulseaudio-utils python-apt-common python3-apport python3-apt python3-debian
  python3-distro-info python3-distupgrade python3-macarobakery python3-problem-report python3-software-properties python3-speechd
  python3-tz samba-libs software-properties-common software-properties-gtk speech-dispatcher speech-dispatcher-audio-plugins
  speech-dispatcher-espeak-ng systemd systemd-hwe-hwdb systemd-oomd systemd-sysv cpdump thermald thunderbird
  thunderbird-gnome-support thunderbird-locale-en thunderbird-locale-en-us tzdata ubuntu-advantage-tools ubuntu-desktop
  ubuntu-desktop-minimal ubuntu-drivers-common ubuntu-minimal ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk ubuntu-standard udev
```



```
Activities Terminal Sep 26 14:05
lab1002@lab1002-HP-280-G3-MT: ~
[1]: update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.9ubuntu3) ...
Setting up php (2:8.1+92ubuntu1) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libmc-bin (2.35-ubuntu3.3) ...
Processing triggers for php8.1-cl (8.1.2-1ubuntu2.14) ...
Processing triggers for libapache2-mod-php8.1 (8.1.2-1ubuntu2.14) ...
lab1002@lab1002-HP-280-G3-MT: $ sudo useradd nagios
lab1002@lab1002-HP-280-G3-MT: $ sudo groupadd nagcmd
lab1002@lab1002-HP-280-G3-MT: $ sudo usermod -a -G nagcmd nagios
lab1002@lab1002-HP-280-G3-MT: $ sudo usermod -a -G nagcmd www-data
lab1002@lab1002-HP-280-G3-MT: $ wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz
--2023-09-26 14:04:26-- https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://code.load.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6 [following]
--2023-09-26 14:04:26-- https://code.load.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6
Resolving code.load.github.com (code.load.github.com)... 20.207.73.88
Connecting to code.load.github.com (code.load.github.com)|20.207.73.88|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz [=====] 10.81M 8.06MB/s in 1.3s
2023-09-26 14:04:28 (8.06 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333431]
lab1002@lab1002-HP-280-G3-MT: $ tar -xvf nagios-4.4.6.tar.gz
nagioscore-nagios-4.4.6/
nagioscore-nagios-4.4.6/.gitignore
nagioscore-nagios-4.4.6/.travis.yml
nagioscore-nagios-4.4.6/CONTRIBUTING.md
nagioscore-nagios-4.4.6/ChangeLog
nagioscore-nagios-4.4.6/INSTALLING
nagioscore-nagios-4.4.6/LICENSE
nagioscore-nagios-4.4.6/LICENSE
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23

```
Activities Terminal Sep 26 14:07 lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
-----  
HTML URL: http://localhost/nagios/  
CGI URL: http://localhost/nagios/cgi-bin/  
Traceroute (used by WAP):  
  
Review the options above for accuracy. If they look okay,  
type 'make all' to compile the main program and CGIs.  
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ make all  
cd .base && make  
make[1]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/base'  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c  
nagios.c: In function 'main':  
nagios.c:611:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]  
611 |         asprintf(&mc->x[MACRO_PROCESSSTARTTIME], "%llu", (unsigned long long)program_start);  
|  
nagios.c:841:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]  
841 |         asprintf(&mc->x[MACRO_EVENTSTARTTIME], "%llu", (unsigned long long)event_start);  
|  
nagios.c: In function 'nagios_core_worker':  
nagios.c:176:17: warning: ignoring return value of 'read' declared with attribute 'warn_unused_result' [-Wunused-result]  
176 |     read(sd, response + 3, sizeof(response) - 4);  
|  
nagios.c: In function 'test_path_access':  
nagios.c:122:17: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]  
122 |     asprintf(&path, "%s/%s", p, program);  
|  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ..//common/shared.o ..//common/shared.c  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c  
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c  
workers.c: In function 'handle_worker_result':  
workers.c:801:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]  
801 |     asprintf(&error_reason, "timed out after %.2fs", tv_delta_f(&wpres.start, &wpres.stop));  
|  
workers.c:804:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23

```
Activities Terminal Sep 26 14:09
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6

make[1]: Leaving directory '/home/lab1002/nagioscore-nagios-4.4.6'
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cgi
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/swtch.cfg /usr/local/nagios/etc/objects/swtch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files. You'll need to read the documentation for more information on how to actually define services, hosts, etc. to fit your particular needs.

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation

*** Exfoliation theme installed ***
NOTE: Use 'make install-classic' to revert to classic Nagios theme

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$
```

```
Activities Terminal Sep 26 14:11
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6

/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/swtch.cfg /usr/local/nagios/etc/objects/swtch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files. You'll need to read the documentation for more information on how to actually define services, hosts, etc. to fit your particular needs.

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation

*** Exfoliation theme installed ***
NOTE: Use 'make install-classic' to revert to classic Nagios theme

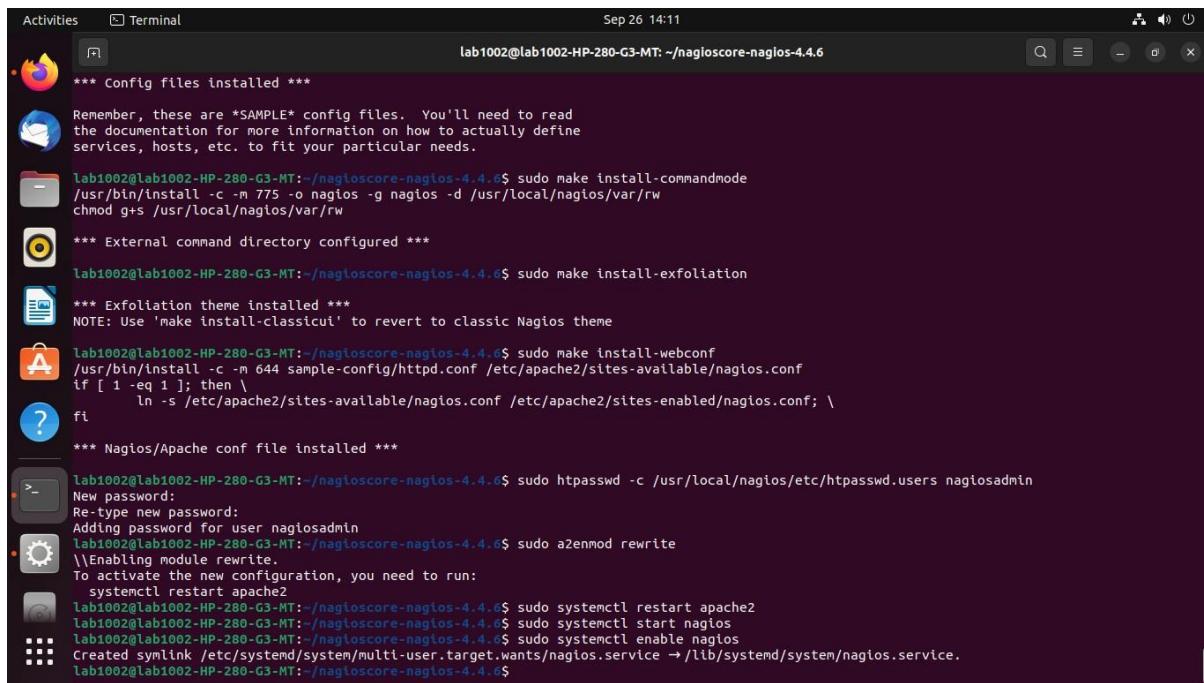
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf
if [ 1 -eq 1 ]; then \
    ln -s /etc/apache2/sites-available/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo a2enmod rewrite
\bEnabling module rewrite.
To activate the new configuration, you need to run:
    systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23



```
*** Config files installed ***
Remember, these are *SAMPLE* config files. You'll need to read the documentation for more information on how to actually define services, hosts, etc. to fit your particular needs.

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

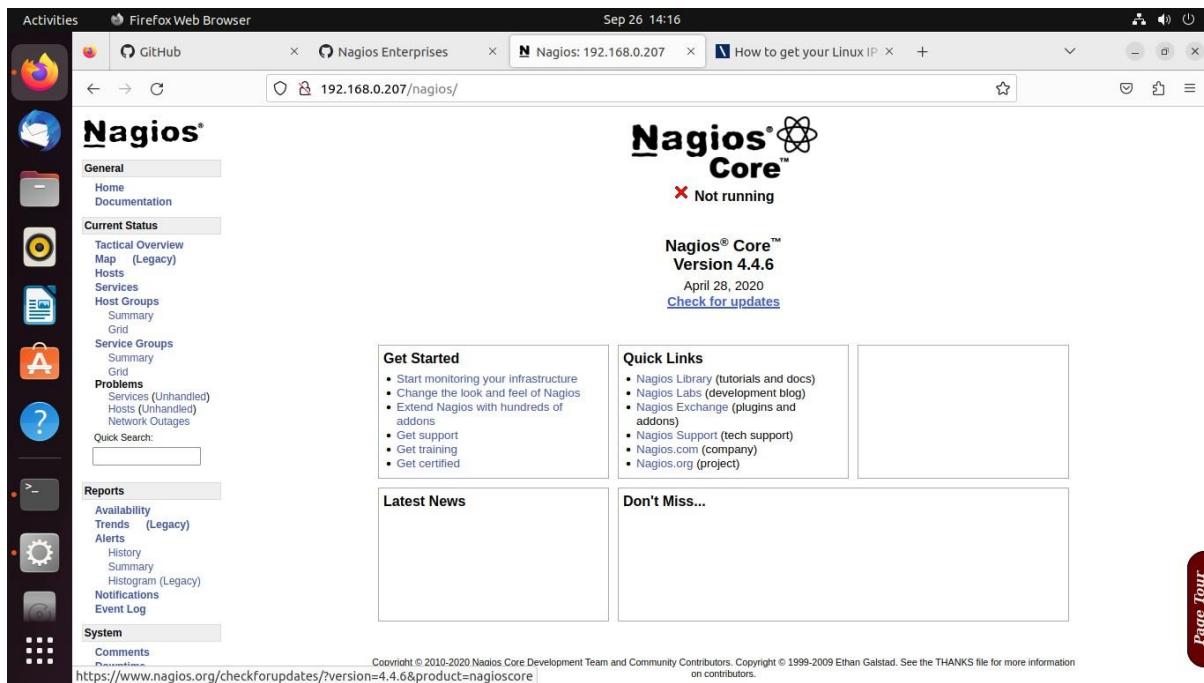
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation

*** Exfoliation theme installed ***
NOTE: Use 'make install-classicul' to revert to classic Nagios theme

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-webconf
/usr/bin/uninstall -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf
if [ 1 -eq 1 ]; then \
    ln -s /etc/apache2/sites-available/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo a2enmod rewrite
\[Enabling module rewrite.
To activate the new configuration, you need to run:
systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl start nagios
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /lib/systemd/system/nagios.service.
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$
```



Nagios® Core™
Version 4.4.6
April 28, 2020
[Check for updates](#)

Not running

Get Started

- Start monitoring your infrastructure
- Change the look and feel of Nagios
- Extend Nagios with hundreds of addons
- Get support
- Get training
- Get certified

Quick Links

- Nagios Library (tutorials and docs)
- Nagios Labs (development blog)
- Nagios Exchange (plugins and addons)
- Nagios Support (tech support)
- Nagios.com (company)
- Nagios.org (project)

Latest News

Don't Miss...

Copyright © 2010-2020 Nagios Core Development Team and Community Contributors. Copyright © 1999-2009 Ethan Galstad. See the THANKS file for more information on contributors.

<https://www.nagios.org/checkforupdates?version=4.4.6&product=nagioscore>

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 26/09/23

The screenshot shows the Nagios Core 4.4.6 dashboard. The top navigation bar includes tabs for GitHub, Nagios, How to get your Linux, and Nagios is active on CLI. The main header features the Nagios logo and the message "Daemon running with PID 49172". On the left, a sidebar menu lists General, Current Status, Reports, and System sections. The Current Status section contains links for Tactical Overview Map (Legacy), Hosts, Services, Host Groups, Service Groups, Problems, and Reports. The Reports section includes Availability, Trends (Legacy), Alerts, History, Summary, Histogram (Legacy), Notifications, and Event Log. The System section has links for Comments, Downtime, and Process Info. The central content area is divided into several panels: "Get Started" (with a list of 7 items), "Quick Links" (with a list of 7 items), "Latest News" (empty), and "Don't Miss..." (empty). A footer note at the bottom left reads: "Copyright © 2010-2020 Nagios Core Development Team and Community Contributors. Copyright © 1999-2009 Ethan Galstad. See the THANKS file for more information on contributors." A "Page Tour" button is located on the right side.

Conclusion :

Thus we have understood the steps of installation of nagios and installed the nagios on ubuntu system.

Experiment No 10

Aim : To study Puppet tools.

Lab Outcome :

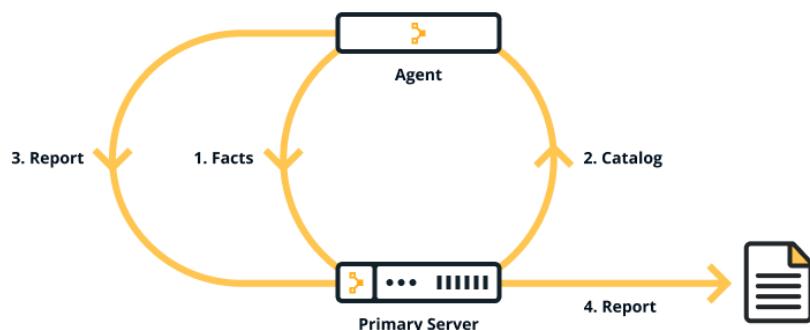
LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO6 : To Synthesize software configuration and provisioning using Ansible.

Theory :

1) What is Puppet ?

Puppet is a powerful and widely-used configuration management and automation tool designed to simplify and streamline the management of IT infrastructure. It falls under the category of DevOps tools, which aim to bridge the gap between development and operations by automating infrastructure provisioning and configuration. Puppet allows you to define and enforce the desired state of your systems, making it easier to manage large-scale, heterogeneous IT environments efficiently.



2) What Puppet Can Do?

Puppet is a versatile tool that offers a wide range of capabilities, making it an essential component in the toolkit of DevOps and system administrators. Here are some key things Puppet can do:

1. Configuration Management

Puppet excels at configuration management. It enables you to define the desired configuration of your systems using code, called Puppet manifests. These manifests describe how each component of your infrastructure should be configured, ensuring consistency across your environment.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

2. Automation

Puppet automates repetitive tasks and workflows. You can use Puppet to deploy applications, manage users and permissions, configure network settings, and perform other routine operations. This automation reduces the risk of human error and frees up valuable time for IT teams.

3. Infrastructure as Code (IaC)

Puppet treats infrastructure as code, allowing you to define, version, and manage your infrastructure using code files. This approach promotes collaboration, code reuse, and version control, similar to software development practices.

4. Scalability

Puppet is designed to handle large and complex infrastructures. Whether you manage a handful of servers or thousands of nodes, Puppet's scalability ensures that you can maintain consistent configurations and apply changes across your entire environment with ease.

5. Cross-Platform Support

Puppet supports various operating systems and platforms, making it suitable for heterogeneous environments. You can manage Linux, Windows, macOS, and other systems using a single Puppet codebase.

6. Reporting and Monitoring

Puppet provides reporting and monitoring features to track the status and compliance of your infrastructure. You can gain insights into configuration changes, troubleshoot issues, and ensure that your systems remain in the desired state.

7. Extensibility

Puppet's ecosystem includes a vast library of pre-built modules and a flexible plugin system. This extensibility allows you to customize Puppet to meet your specific needs and integrate it with other tools in your DevOps toolchain.

3) How Puppet Works?

Puppet is based on a Pull deployment model, where the agent nodes check in regularly after every **1800** seconds with the master node to see if anything needs to be updated in the agent. If anything needs to be updated the agent pulls the necessary puppet codes from the master and performs required actions.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

Example: Master – Agent Setup:

The Master:

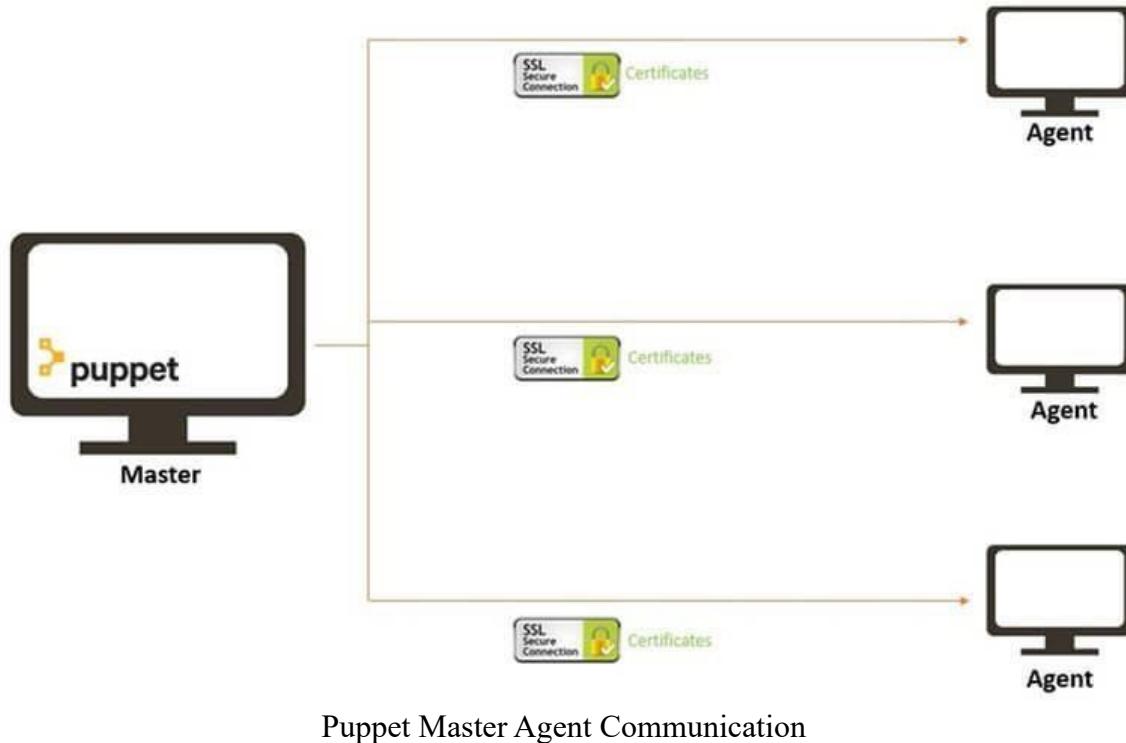
A Linux based machine with Puppet master software installed on it. It is responsible for maintaining configurations in the form of puppet codes. The master node can only be Linux.

The Agents:

The target machines managed by a puppet with the puppet agent software installed on them.

The agent can be configured on any supported operating system such as Linux or Windows or Solaris or Mac OS.

The communication between master and agent is established through secure certificates.

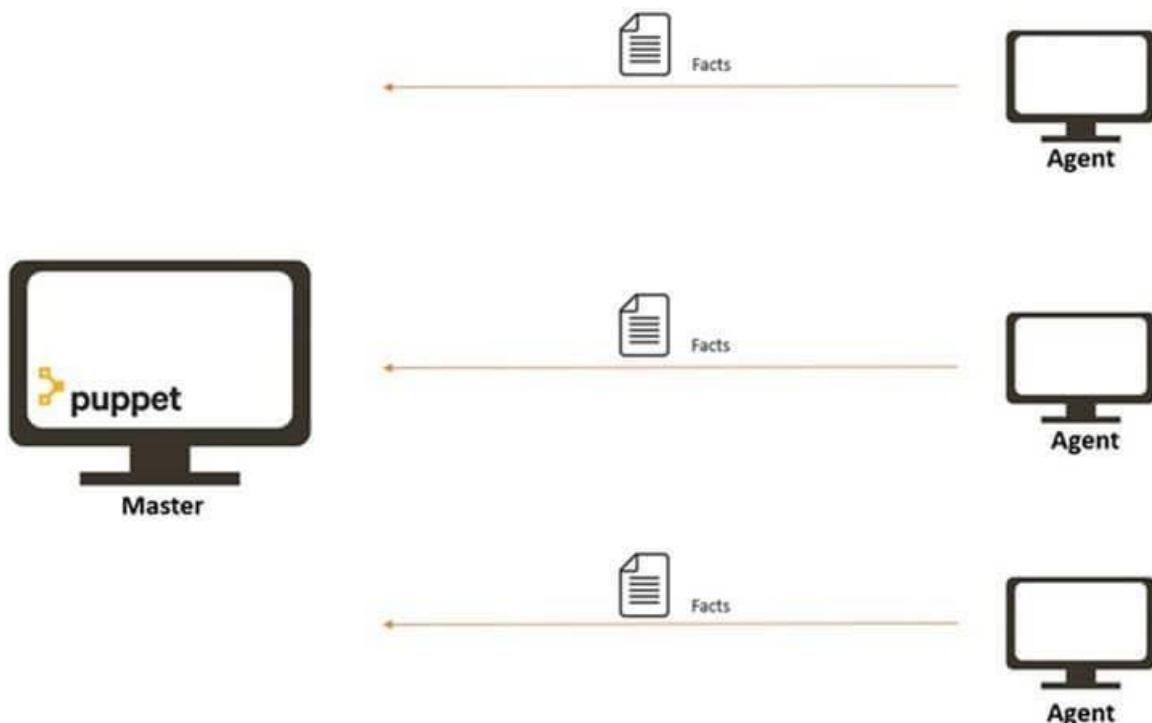


Communication between the Master and the Agent:

Step 1) Once the connectivity is established between the agent and the master, the Puppet agent sends the data about its state to the Puppet master server. These are called Facts: This information includes the hostname, kernel details, IP address, file name details, etc....

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23



Agent Sends Facts to Master

Step 2) Puppet Master uses this data and compiles a list with the configuration to be applied to the agent. This list of configuration to be performed on an agent is known as a **catalog**. This could be changed such as package installation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.



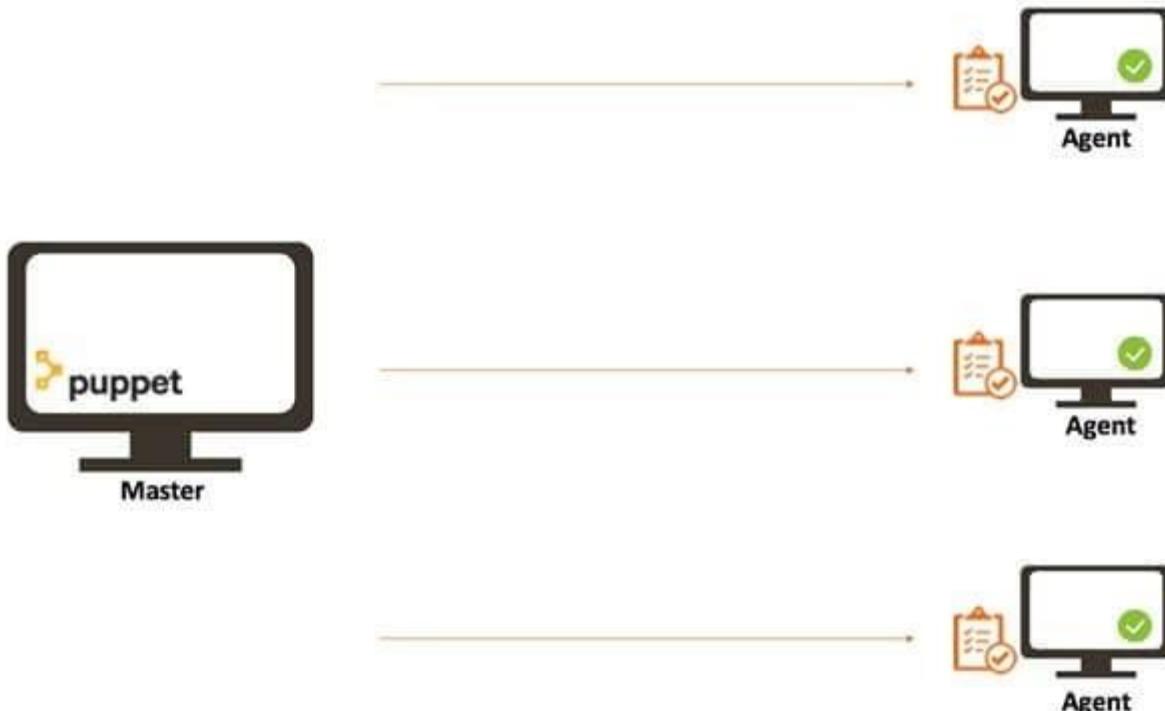
Master sends a catalog to Agent

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

Step 3) The agent uses this list of configuration to apply any required configuration changes on the node.

In case there are no drifts in the configuration, Agent does not perform any configuration changes and leaves the node to run with the same configuration.



Agent applies configuration

Step 4) Once it is done the node reports back to puppet master indicating that the configuration has been applied and completed.

4) Puppet Blocks: Puppet Resources, Puppet Classes, Puppet Manifests, Puppet Modules

To understand how Puppet is organized and how it manages configuration, it's essential to grasp its fundamental building blocks:

1. Puppet Resources

Puppet resources are the basic building blocks of configuration management. A resource represents a specific aspect of a system's configuration, such as a file, package, user, or service. Resources are defined in Puppet manifests and describe what the system should look like.

Example of defining a file resource:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

```
```puppet file {
 '/etc/nginx/nginx.conf':
 ensure => present,
 owner => 'root',
 group => 'root',
 mode => '0644',
 source => 'puppet:///modules/nginx/nginx.conf',
}
````
```

In this example, we're defining a file resource that ensures the presence of the Nginx configuration file with specific ownership, permissions, and content source.

2. Puppet Classes

Puppet classes are used to group related resources and provide structure to your Puppet manifests. Classes make it easier to organize and manage configurations for specific roles or applications. They can be included in manifests to apply the associated configuration to nodes.

Example of defining and including a class:

```
```puppet class
nginx {
 Define resources related to Nginx here
}

node 'webserver.example.com' {
 include nginx
}
````
```

In this example, we've defined an "nginx" class and included it in a specific node's configuration.

3. Puppet Manifests

Puppet manifests are the code files that define your desired system configuration. Manifests contain a collection of resource definitions, classes, and node configurations. Puppet manifests are written in Puppet's declarative language.

Example of a simple Puppet manifest:

```
```puppet file {
 '/etc/motd':
 ensure
 => present,
 content => "Welcome to our server.\n",
}
````
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

}

```
package { 'nginx':  
  ensure => installed,  
}  
...
```

This manifest ensures the presence of a message of the day (motd) file and installs the Nginx package.

4. Puppet Modules

Puppet modules are reusable units of Puppet code that encapsulate configurations, resources, and classes for specific tasks or applications. Modules are stored in a directory structure and can be easily shared and reused across different Puppet environments.

Benefits of using Puppet modules:

- Reusability: Modules can be shared across teams and projects, reducing duplication of effort.
- Modularity: Modules promote organization and separation of concerns in your Puppet code.
- Versioning: Modules can be versioned, making it easier to track changes and updates.
- Community Ecosystem: Puppet Forge hosts a vast collection of community-contributed Puppet modules for various purposes.

5) Benefits of Puppet as a DevOps Tool

Puppet offers several benefits as a DevOps tool, facilitating collaboration between development and operations teams and improving the overall management of IT infrastructure:

1. Consistency and Compliance

Puppet ensures that all systems in your environment adhere to a consistent and compliant configuration. This reduces the risk of configuration drift and security vulnerabilities.

2. Automation

Puppet automates repetitive tasks and workflows, allowing teams to focus on more strategic activities. This increases efficiency and reduces the potential for human error.

3. Scalability

Puppet scales effortlessly, making it suitable for managing both small and large infrastructures. As

your environment grows, Puppet can accommodate your needs.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

4. Infrastructure as Code (IaC)

By treating infrastructure as code, Puppet promotes collaboration, version control, and best practices similar to software development. This aligns with the DevOps philosophy of automating and codifying processes.

5. Reporting and Monitoring

Puppet provides reporting and monitoring capabilities, giving you visibility into your infrastructure's state and helping you identify and resolve issues proactively.

6. Cross-Platform Support

Puppet supports a wide range of operating systems and platforms, simplifying the management of heterogeneous environments.

7. Ecosystem and Community

Puppet has a thriving community and ecosystem, with numerous pre-built modules and resources available on Puppet Forge. This accelerates development and configuration efforts.

6) What are Puppet Manifest Files?

Puppet manifest files are the core configuration files in Puppet. They define the desired state of your systems by declaring resources and their properties. Manifests are written in Puppet's declarative language and are used to manage and configure resources on target nodes.

A Puppet manifest typically has a `'.pp` file extension and contains resource definitions, class declarations, and node configurations.

7) Syntax of a Manifest File

The syntax of a Puppet manifest file is designed to be human-readable and easy to understand. Here's a basic structure of a Puppet manifest:

```
```puppet
Comments start with a " symbol

Resource definition
resource_type { 'resource_name':
 property1 => value1, property2
 => value2,
 ...
}
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

Class declaration class  
class\_name { Class  
content here  
}

Node configuration node  
'node\_name' {  
    Manifest content specific to the node  
}  
...

Key components of the syntax include:

- Resource Definition: Resources are defined with a type (e.g., 'file', 'package') and a name. Properties and values configure the resource.
- Class Declaration: Classes are declared using the 'class' keyword and contain configurations specific to a role or application.
- Node Configuration: Node configurations specify which nodes should apply particular manifests or classes.

## 7) Why Do We Need Puppet Manifest Files?

Puppet manifest files serve several crucial purposes:

1. Configuration as Code: Manifests allow you to describe your infrastructure's desired state in code, making configurations easily reproducible and version-controlled.
2. Automation: Manifests automate the process of configuring and maintaining your systems. Puppet agents apply configurations based on manifest definitions.
3. Consistency: Manifests ensure that all nodes within your environment adhere to the same configuration, reducing configuration drift and errors.
4. Modularity: By organizing configurations into manifests, classes, and modules, you can maintain a modular and organized Puppet codebase, promoting reusability.

## 7) Writing a Basic Manifest File with Examples

Let's create a simple Puppet manifest to illustrate its structure and usage. In this example, we'll ensure the presence of an SSH configuration file and install the Apache web server.

```
```puppet
Puppet manifest to manage SSH configuration and install Apache
```

```
Resource definition for SSH configuration
file { '/etc/ssh/sshd_config': ensure =>
```

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 05/09/23

```
present, owner => 'root', group =>
'root', mode => '0644',
content => template('my_module/sshd_config.erb'),
}
```

Resource definition to install Apache package

```
package { 'apache2': ensure => installed,
}
```

Node configuration for a specific node

```
node 'webserver.example.com' {
  Include the Apache class for this node
  include apache
}
```

...

Here:

- We define a `file` resource to ensure the presence of the SSH configuration file with specific ownership, permissions, and content sourced from a template.
- We define a `package` resource to ensure that the Apache (`apache2`) package is installed on the node.
- We specify a node configuration for a specific node ('webserver.example.com') and include the 'apache' class, which we can define elsewhere in our Puppet codebase.

This basic Puppet manifest demonstrates how to manage configuration files and packages on a node. Puppet will ensure that the SSH configuration and Apache package are in the desired state for the specified node.

Conclusion :

Puppet is a robust DevOps tool that automates and manages your computer systems using special instructions called "manifests." These manifests, acting like a blueprint for your infrastructure, ensure your systems run smoothly and consistently, making Puppet an indispensable solution for system configuration and maintenance.

Written Assignment 1

1. What is test automation or automation testing? What are the advantages of automation testing?

In the realm of software development and quality assurance, test automation, or automation testing, plays a pivotal role in ensuring the reliability and effectiveness of software applications. This process involves the use of specialized software tools and scripts to automate the execution of test cases, thereby reducing manual effort and improving the overall efficiency of the testing process.

Test automation, simply put, is the use of automation tools and frameworks to perform testing tasks, which are otherwise done manually. These testing tasks include the execution of test cases, regression testing, performance testing, load testing, and many more. The fundamental idea behind automation testing is to minimize human intervention while running repetitive and time-consuming tests. Automation can be applied at various levels of software testing, including unit testing, integration testing, functional testing, and user interface testing.

Automation testing offers a multitude of advantages that make it an indispensable part of the software development and quality assurance process. Below are some of the key benefits:

1. Improved Efficiency:

- Automation allows for the quick and efficient execution of test cases, which significantly reduces the time and effort required for testing.
- Test scripts can run tests simultaneously on multiple platforms, browsers, or devices, increasing coverage and speeding up the testing process.

2. Consistency:

- Automated tests are consistent in their execution, ensuring that the same set of test cases is executed with precision every time.
- This consistency is especially valuable in regression testing, where it is crucial to verify that recent code changes have not introduced new defects.

3. Reusability:

- Test scripts can be reused across different test phases, saving time and effort in creating new test cases.
- Reusable test scripts allow for comprehensive test coverage throughout the software development lifecycle.

4. Faster Feedback:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

4. Faster Feedback:

- Automation testing provides rapid feedback to developers, allowing them to identify and rectify issues early in the development process.
- Early bug detection results in reduced debugging and maintenance costs.

5. Increased Test Coverage:

- Automation can execute a large number of test cases in a short time, leading to more comprehensive test coverage.
- This ensures that a wide range of scenarios and edge cases are tested, reducing the likelihood of undiscovered defects in the final product.

6. Regression Testing:

- Automation is highly beneficial for regression testing, as it allows for the repeated execution of test cases after code changes to verify that new features or modifications have not adversely impacted existing functionality.

7. Data-Driven Testing:

- Automation supports data-driven testing, where the same test script is executed with different sets of test data, enhancing the versatility and scalability of test cases.

8. Performance Testing:

- Automation can simulate a large number of virtual users to conduct performance and load testing, providing insights into how an application behaves under various conditions.

9. Cost-Effective:

- While setting up automation initially requires an investment of time and resources, it ultimately leads to cost savings by reducing the need for manual testing efforts.

10. Continuous Integration and Continuous Delivery (CI/CD):

- Automation seamlessly integrates with CI/CD pipelines, allowing for the automatic execution of tests with every code commit, ensuring that only high-quality code is deployed.

- Automation testing tools often provide detailed test reports, logs, and dashboards, which facilitate comprehensive test result analysis and debugging.

11. Enhanced Reporting:

12. Scalability:

- As the size and complexity of software projects increase, automation can easily scale to accommodate the growing number of test cases and variations.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

- Automation testing tools often provide detailed test reports, logs, and dashboards, which facilitate comprehensive test result analysis and debugging.

12. Scalability:

- As the size and complexity of software projects increase, automation can easily scale to accommodate the growing number of test cases and variations.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

13. Documentation:

- Automated test scripts serve as documentation for test cases, making it easier for team members to understand and maintain test procedures.

14. Reduced Human Error:

- Automation eliminates the risk of human error in test case execution, which can be especially crucial in scenarios where precision is paramount.

15. Parallel Testing:

- Automation allows the execution of test cases in parallel, which can significantly reduce testing time, especially for complex applications.

Challenges and Considerations:

While automation testing offers numerous advantages, it also comes with challenges and considerations. Some of these include:

- Initial Setup: Setting up automation frameworks and writing test scripts can be time-consuming, especially for complex applications.
- Maintenance: Test scripts require regular maintenance to adapt to changes in the application's functionality. This can be an ongoing effort.
- Skill Requirements: Automation testing requires testers to have scripting and programming skills, which may not be possessed by all team members.
- Costs: Initially, automation can incur costs in terms of tool licensing, infrastructure, and personnel training.
- Not Suitable for Everything: Not all testing scenarios are suitable for automation, such as usability testing, exploratory testing, or tests that require human judgment.
- Test Data Management: Managing test data for automation can be complex, and ensuring the privacy and security of sensitive data is critical.

In conclusion, automation testing is a critical component of modern software development and quality assurance processes. Its numerous advantages, including improved efficiency, consistency, reusability, and cost-effectiveness, make it an essential tool for ensuring the

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

reliability and quality of software applications. However, it is essential to approach automation with a well-defined strategy, considering the specific needs and challenges of the project, to harness its full potential and maximize the return on investment.

2. What is XPath? Explain the difference between single slash and double slash in XPath.

XPath is a language used for navigating XML documents and is often employed in web scraping, XML processing, and, notably, in locating elements within an HTML document for web automation testing.

XPath Basics:

XPath uses a path-like syntax to pinpoint elements within an XML or HTML document. It consists of a series of location steps separated by slashes ("/"). Each step helps to traverse the document's structure, identifying elements or nodes.

Single Slash ("/"):

When you use a single slash in XPath, it denotes an absolute path. It starts its search from the root node of the document and identifies elements based on the provided criteria. For example, the XPath expression '/html/body/div' will locate the 'div' element that is a direct child of the 'body' element, which, in turn, is a direct child of the 'html' root element.

Double Slash ("//"):

Conversely, a double slash in XPath indicates a relative or partial path. It conducts a search throughout the entire document, not just from the root node. This makes it more flexible for locating elements in a nested structure or finding elements anywhere in the document. For instance, the XPath expression '//div' will locate all 'div' elements, regardless of their nesting or hierarchy.

In summary, XPath is a powerful tool for navigating XML and HTML documents, and it employs both single and double slashes to define paths for locating elements. A single slash signifies an absolute path starting from the root node, while a double slash denotes a relative path, allowing for more flexible element location throughout the document. This flexibility makes XPath an essential tool in web automation testing and web scraping, where element identification is a critical aspect of the testing process.

Written Assignment 2

1. What is Selenium? What are the Selenium suite components?

Selenium is a widely-used open-source framework for automating web browsers, primarily employed for testing web applications but also serving various other purposes in web development and web scraping. Selenium provides a suite of tools and libraries for browser automation and testing, offering cross-browser and cross-platform support. The Selenium project was initiated by Jason Huggins in 2004, and it has since evolved into a powerful and versatile set of tools.

Selenium Suite Components

Selenium is composed of several key components that facilitate different aspects of web automation and testing. These components can be categorized into two primary groups: Selenium WebDriver and Selenium IDE.

1. Selenium WebDriver:

Selenium WebDriver is the most critical and widely used component of the Selenium suite. It provides a programming interface to interact with web browsers and automate user interactions. WebDriver enables developers and testers to write scripts in various programming languages to control web browsers. Key features and components of Selenium WebDriver include:

- Browser Drivers: WebDriver communicates with web browsers through specific browser drivers. Different drivers are available for popular web browsers like Chrome, Safari, Firefox, Safari, and more.
- Programming Language Bindings: Selenium WebDriver supports multiple programming languages, including Java, Python, C#, Ruby, and more. Users can choose the language they are most comfortable with.
- Commands and API: WebDriver provides a rich set of commands and methods for interacting with web elements, such as clicking buttons, filling out forms, and navigating through web pages.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

- Cross-Browser and Cross-Platform Compatibility: One of the strengths of Selenium WebDriver is its ability to work with different web browsers (cross-browser) and on various operating systems (cross-platform). It's great for conducting rigorous web testing or automating web-related tasks.
- Test Framework Integration: WebDriver can be integrated with various testing frameworks, such as JUnit and TestNG, enabling users to organize and manage their test suites effectively.

2. Selenium IDE:

2. Selenium IDE:

Selenium IDE is a simpler, record-and-playback tool for creating test scripts. It's a browser extension that allows testers and developers to record their interactions with a web application and then play them back. While it's less powerful and flexible compared to WebDriver, it's a great tool for quick and easy test script creation. Key features of Selenium IDE include:

- Record and Playback: Selenium IDE records user interactions with a web page and generates test scripts in a user-friendly, script-like format.
- Selenium Script Export: Test scripts created in Selenium IDE can be exported to various programming languages for further customization and execution in WebDriver.
- Limited Scripting Support: While it's not as versatile as WebDriver, Selenium IDE does offer basic scripting capabilities, allowing users to add conditions and loops to their test scripts.
- Rapid Test Script Creation: Selenium IDE is known for its ease of use, making it a great choice for those who need to quickly create test scripts without in-depth programming knowledge.

In summary, Selenium is a powerful and versatile suite of tools used for automating web browsers and testing web applications. Its core components, Selenium WebDriver and Selenium IDE, provide developers and testers with the means to automate user interactions with web applications, ensuring their functionality and performance. Whether you need to conduct rigorous web testing or automate web-related tasks, Selenium's components offer a range of capabilities to suit your needs.

2. What makes Selenium such a widely used testing tool? Give reasons. Why is it advised to select Selenium as a testing tool for web applications or systems?

Selenium is a widely used testing tool for web applications and systems, and its popularity can be attributed to several key factors. Here are the reasons why Selenium is a top choice for web application testing:

1. Open Source and Cost-Effective:

Selenium is an open-source testing framework, which means it is freely available to anyone. This open-source nature significantly reduces testing costs, making it an attractive choice for both large enterprises and small businesses. This cost-effectiveness allows organizations to allocate resources to other critical areas of development and testing.

2. Browser and Platform Independence:

Selenium supports multiple web browsers, including Chrome, Firefox, Internet Explorer, Edge, Safari, and more. This cross-browser compatibility is crucial as it ensures that web applications behave consistently across different browsers and platforms. It allows developers to verify that their applications are accessible and functional for a broad user base.

3. Programming Language Flexibility:

Selenium supports various programming languages, including Java, Python, C#, Ruby, and more. This flexibility allows development and testing teams to work with the language they are most proficient in. They can leverage their existing knowledge and libraries, making Selenium adaptable to a wide range of development environments.

4. Active Community and Documentation:

Selenium has a vibrant and active user community. This community support provides a wealth of online resources, including tutorials, forums, and discussion groups. Users can find answers to their questions, share knowledge, and troubleshoot issues. The extensive documentation helps testers and developers get started and maximize their use of Selenium.

5. Cross-Browser Testing with Selenium Grid:

Selenium Grid, a component of the Selenium suite, enables cross-browser testing by allowing tests to run simultaneously on multiple browser and platform configurations. This reduces the risk of browser-specific issues and improves overall application quality.

6. Reusability and Maintainability:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

ensures that web applications are tested comprehensively across various environments, reducing the risk of browser-specific issues and improving overall application quality.

6. Reusability and Maintainability:

Selenium promotes the creation of reusable test scripts. Test scripts are written to interact with web elements using selectors, and when the structure of a web application changes, only the relevant selectors need to be updated. This reduces the effort required to maintain test suites, especially in dynamic development environments.

7. Integration with Testing Frameworks:

Selenium seamlessly integrates with popular testing frameworks such as TestNG, JUnit, and Cucumber. These frameworks offer advanced features for test management, reporting, and test data management, making it easier to manage and execute tests efficiently.

8. Continuous Integration and Automation:

Selenium plays a vital role in enabling continuous integration and continuous delivery (CI/CD) pipelines. Tests can be automated and executed with each code change, providing early feedback on the quality of the application. Automation helps identify and address issues in real-time, accelerating the release process.

9. Scalability:

Selenium can be used to automate a wide range of test cases, from simple unit tests to complex end-to-end scenarios. Its scalability allows organizations to implement automation at different levels of their testing processes, making it adaptable to the specific needs and maturity of the project.

10. Industry-Wide Acceptance:

Many industry leaders and technology giants rely on Selenium for their web application testing needs. Its widespread adoption demonstrates its credibility and effectiveness in identifying and addressing issues in web applications, making it a trusted choice in the industry.

In summary, Selenium's open-source nature, compatibility with multiple browsers and platforms, programming language flexibility, and extensive community support make it a widely used and advised testing tool for web applications and systems. Its ability to facilitate cross-browser testing, reusability, integration with testing frameworks, and support for continuous integration and automation further strengthen its position as a valuable asset in the world of software testing. Its reliability and industry-wide acceptance make it a top choice for ensuring the quality and reliability of web applications and systems.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 12/10/23

the world of software testing. Its reliability and industry-wide acceptance make it a top choice for ensuring the quality and reliability of web applications and systems.