Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Niranjan Rajesh Joshi

of IT Department, Semester V with

Roll No. 51 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 2023 - 2024

Teacher In-Charge

Sonali
23/10/23

Head of the Department

Date 23/10/23

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1)	To understand DevOps: principles, practices		17/07/23	
	and DevOps engineer role and responsibilities			
2)	To understand version control system. Git		24/07/23	
	installation and Github account			
3)	To perform various Git operations		31/07/23	
4)	To understand continuous integration,		8/08/23	
	jenkins installation			
5)	To build Java program using Jenkins		07/08/23	
6)	To build pipeline using Jenkins		21/08/23	Sondal 23/10/23
7)	To understand Docker architecture and		28/08/23	
	container life cycle, install docker,			
	deploy container in Docker			
8)	To build image for sample web application		11/09/23	
	using Dockerfile			
9)	Installation of Nagios on Ubuntu System		25/09/23	
10)	To study puppet tools		16/10/23	
11)	Assignment-1		03/10/23	Sondal 23/10/23
12)	Assignment-2		03/10/23	

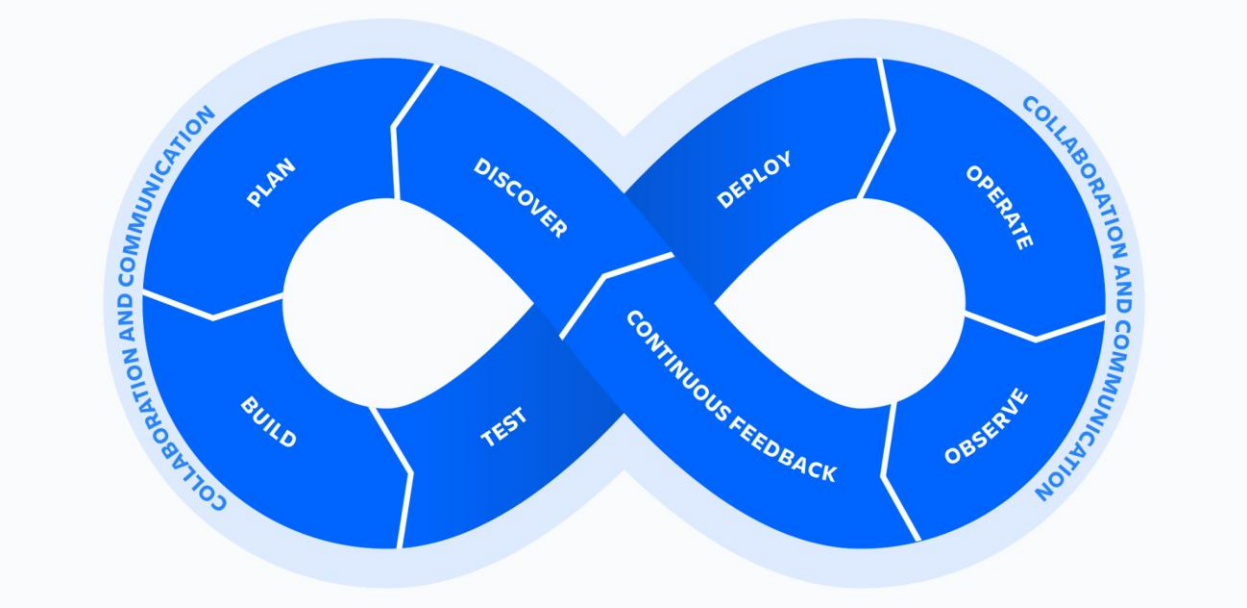
Experiment 1

Aim :- To study Devops:principles,practices,role and responsibilities of devops engineer

What is Devops?

DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams. It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.

The DevOps lifecycle consists of eight phases representing the processes, capabilities, and tools needed for development . Phases include : Discover , Plan , Build , Test , Deploy , Operate , Observe , Continuous feedback



What are tools used in Devops ?

All 8 phases of lifecycle of Devops have different tools , these Include:

1)**Discover Phase**
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

2)**Plan Phase**
tools that allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments such as **Jira , Slack**

3)**Build Phase**
While Puppet and Chef primarily benefit operations, developers use open source tools like **Kubernetes** and **Docker** to provision individual development environments.

4)**Continuous Delivery**
Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. Tools include
Jenkins , Bitbucket , AWS

5)**Test**
Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration.Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration.Tools are : **Xray , Mabl , Snyk**

6)**Deploy**
tools with a single dashboard integrated with your code repository and deployment tools.gives you full visibility on branches, builds, pull requests, and deployment warnings in one place.Tools include **BitBucket , AWSCodePipeline**

7)**Operate**

tools that integrate with your group chat client so alerts go straight to your team's room, or a dedicated room for incidents.Tools include **Slack , Datadog**

8)**Continuous feedback**
applications that integrate your chat tool with your favorite survey platform for NPS-style feedback. Twitter and/or Facebook can also be integrated with chat for real-time feedback. Tools include **GetFeedback , Pendo**

Summary of tools :

1. Git and GitHub – Source code management (Version Control System)
2. Jenkins – Automation server, with plugins built for developing CI/ CD pipelines
3. Selenium – Automation testing
4. Docker – Software Containerization Platform
5. Kubernetes – Container Orchestration tool
6. Puppet – Configuration Management and Deployment
7. Chef – Configuration Management and Deployment
8. Ansible – Configuration Management and Deployment
9. Nagios – Continuous Monitoring

What are roles and responsibilities of Devops engineer ?

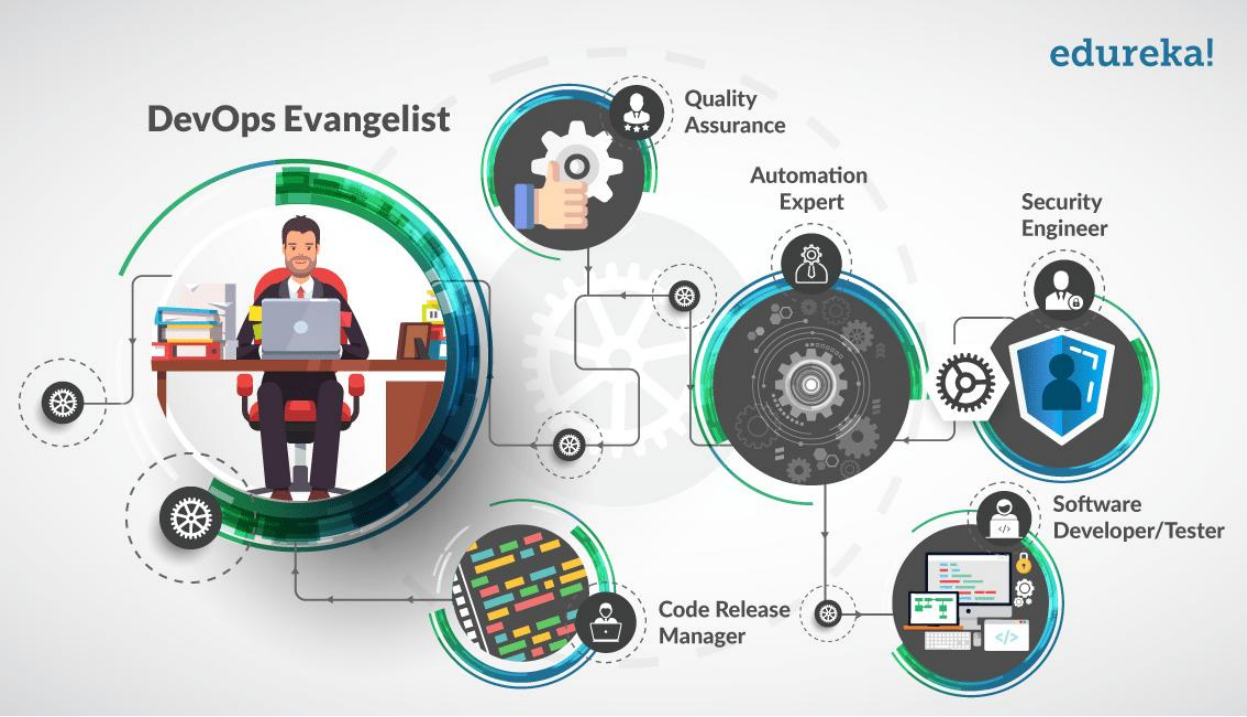
Responsibilities of devops engineer include :

- Understanding customer requirements and project KPIs
- Implementing various development, testing, automation tools, and IT infrastructure
- Planning the team structure, activities, and involvement in project management activities.
- Managing stakeholders and external interfaces
- Setting up tools and required infrastructure
- Defining and setting development, test, release, update, and support processes for DevOps operation
- Have the technical skill to review, verify, and validate the software code developed in the project.
- Troubleshooting techniques and fixing the code bugs

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 various phases and tools used in devops .

LO Mapped : L01

Experiment 2

Aim:-To understand version control system , git installation and github account

Theory:-

What is Git ?

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

What is version control system ?

Version control enables teams to collaborate and streamline development to resolve conflicts and create a centralized location for code.
With version control, every change made to the code base is tracked. This allows software developers to see the entire history of who changed what at any given time — and roll back from the current version to an earlier version if they need to. It also creates a single source of truth.

Version control (or source control or revision control) serves as a safety net to protect the source code from irreparable harm, giving the development team the freedom to experiment without fear of causing damage or creating code conflicts.

Version control system can be of 4 types :

1)Distributed :

A distributed version control system (DVCS) allows users to access a repository from multiple locations. DVCSs are often used by developers who need to work on projects from multiple computers or who need to collaborate with other developers remotely.

2)Centralized :

A centralized version control system (CVCS) is a type of VCS where all users are working with the same central repository. This central repository can be located on a server or on a developer's local machine. Centralized version control systems are typically used in software development projects where a team of developers needs to share code and track changes.

3)Lock Based:

A lock-based version control system uses file locking to manage concurrent access to files and resources. File locking prevents two or more users from making conflicting changes to the same file or resource.

4)Optimistic:

In an optimistic version control system, every user has their own private workspace. When they want to share their changes with the rest of the team, they submit a request to the server. The server then looks at all the changes and determines which ones can be safely merged together.

How to install git ?

Step 1:

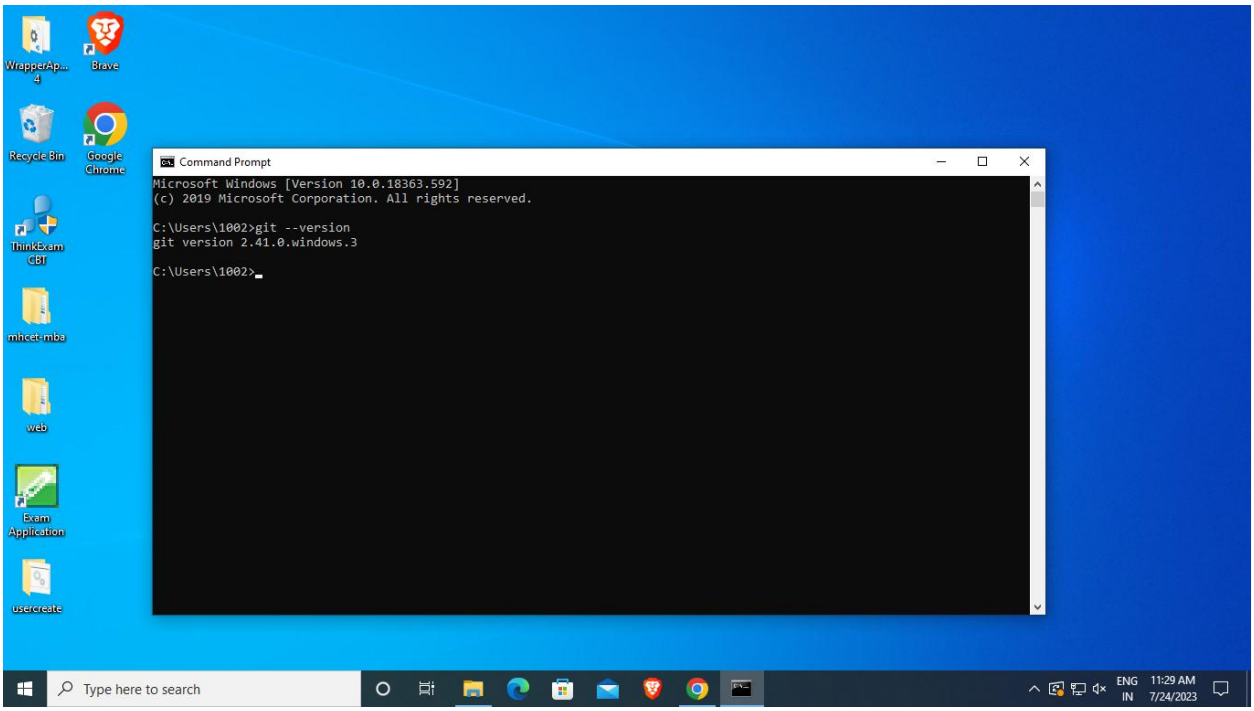
Download the latest version of Git and choose the 64/32 bit version. After the file is downloaded, install it in the system. Once installed, select Launch the Git Bash, then click on finish. The Git Bash is now launched**.**

Step 2:

Check the Git version:

$git --version

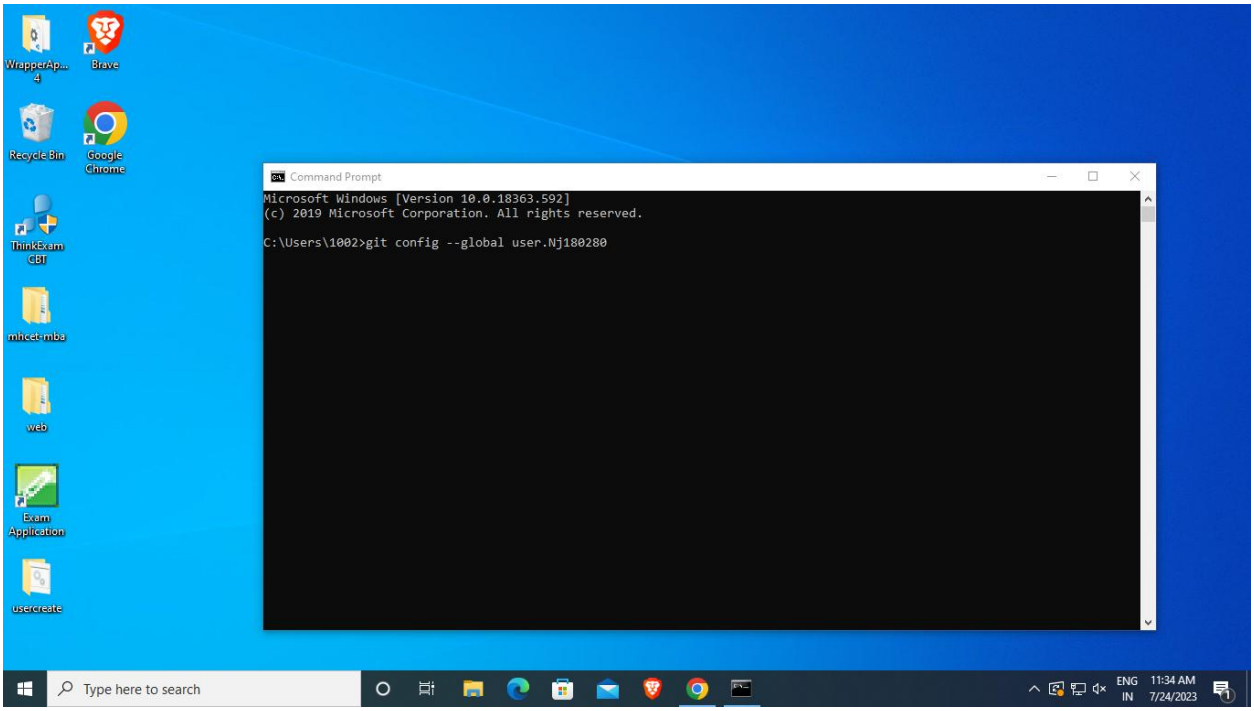
If proper version of git is displayed then git is successfully installed in your computer



Step 3:

Link the Git to a Github Account:

$ git config --global user.username



What is Github and Github account?

GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.GitHub's interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line.

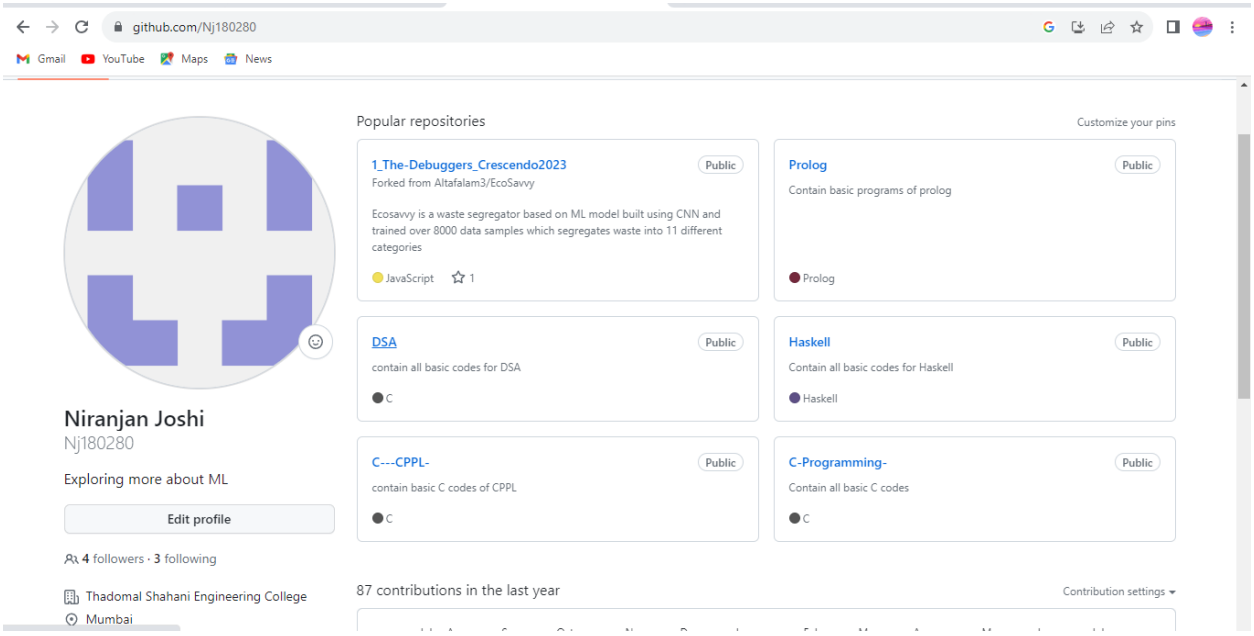
GitHub is so user-friendly, though, that some people even use GitHub to manage other types of projects – like writing books.Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

With GitHub, you can store and collaborate on code. Accounts allow you to organize and control access to that code. There are three types of accounts on GitHub.

- Personal accounts
- Organization accounts
- Enterprise accounts

Every person who uses GitHub signs into a personal account. An organization account enhances collaboration between multiple personal accounts, and an enterprise account allows central management of multiple organizations.

Basic Image of Github account profile is shown below :



Conclusion:-

I have successfully installed git on my desktop and also learnt about version control system , github account and explored various other things related to github account such as repository etc.

LO Mapped : LO1,LO2

Experiment 3

Aim:- To perform various GIT operations

Theory:-

1) git config
Utility : To set your user name and email in the main configuration file.
How to : To check your name and email type in git config --global user.name and git config --global user.email.

2) git init
Utility : To initialise a git repository for a new or existing project.
How to : git init in the root of your project directory.

3) git clone
Utility : To copy a git repository from remote source, also sets the remote to original source so that you can pull again.
How to : git clone <:clone git url:>

4) git status
Utility : To check the status of files you've changed in your working directory, i.e, what all has changed since your last commit.
How to : git status in your working directory. lists out all the files that have been changed.

5) git add
Utility : adds changes to stage/index in your working directory.
How to : git add .

6) git commit
Utility : commits your changes and sets it to new commit object for your remote.
How to : git commit -m"sweet little commit message"

7) git push/git pull
Utility : Push or Pull your changes to remote. If you have added and committed your changes and you want to push them. Or if your remote has updated and you want those latest changes.
How to : git pull <:remote:> <:branch:> and git push <:remote:> <:branch:>

8) git branch

Utility : Lists out all the branches.
How to : git branch or git branch -a to list all the remote branches as well.

9) git checkout
Utility : Switch to different branches
How to : git checkout <:branch:> or **_git checkout -b <:branch:> if you want to create and switch to a new branch.

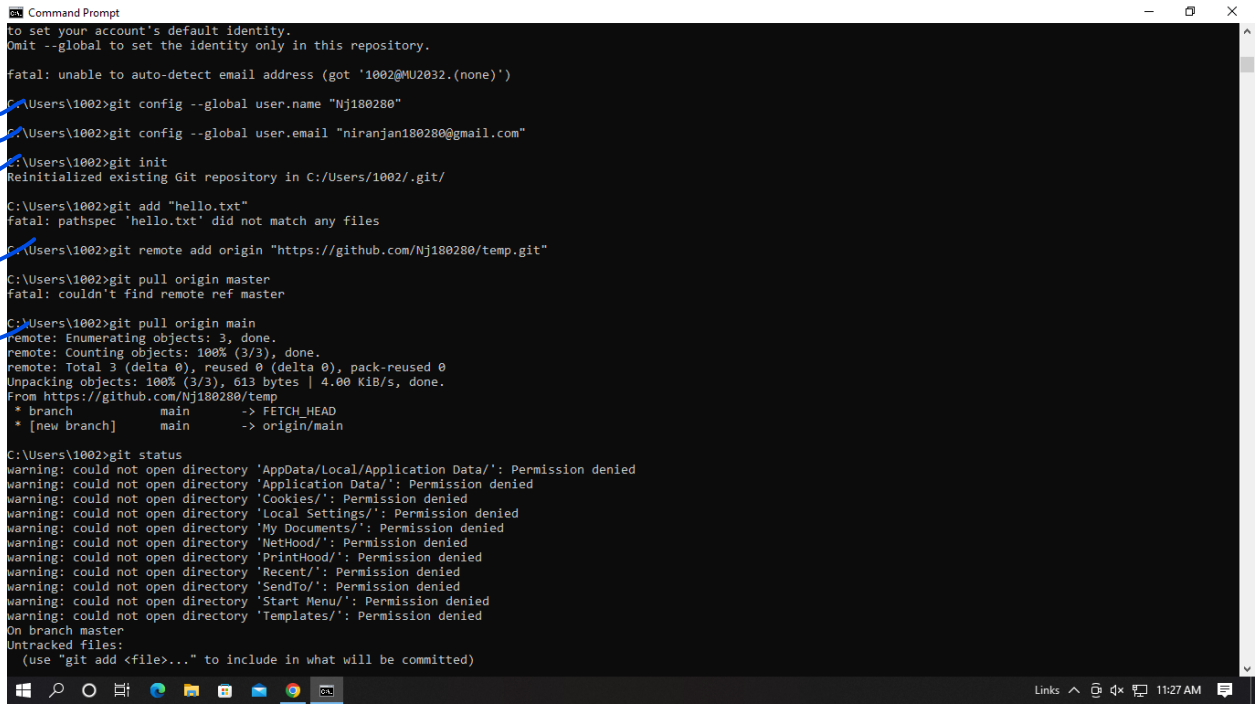
10) git stash
Utility : Save changes that you don't want to commit immediately.
How to : git stash in your working directory. git stash apply if you want to bring your saved changes back.

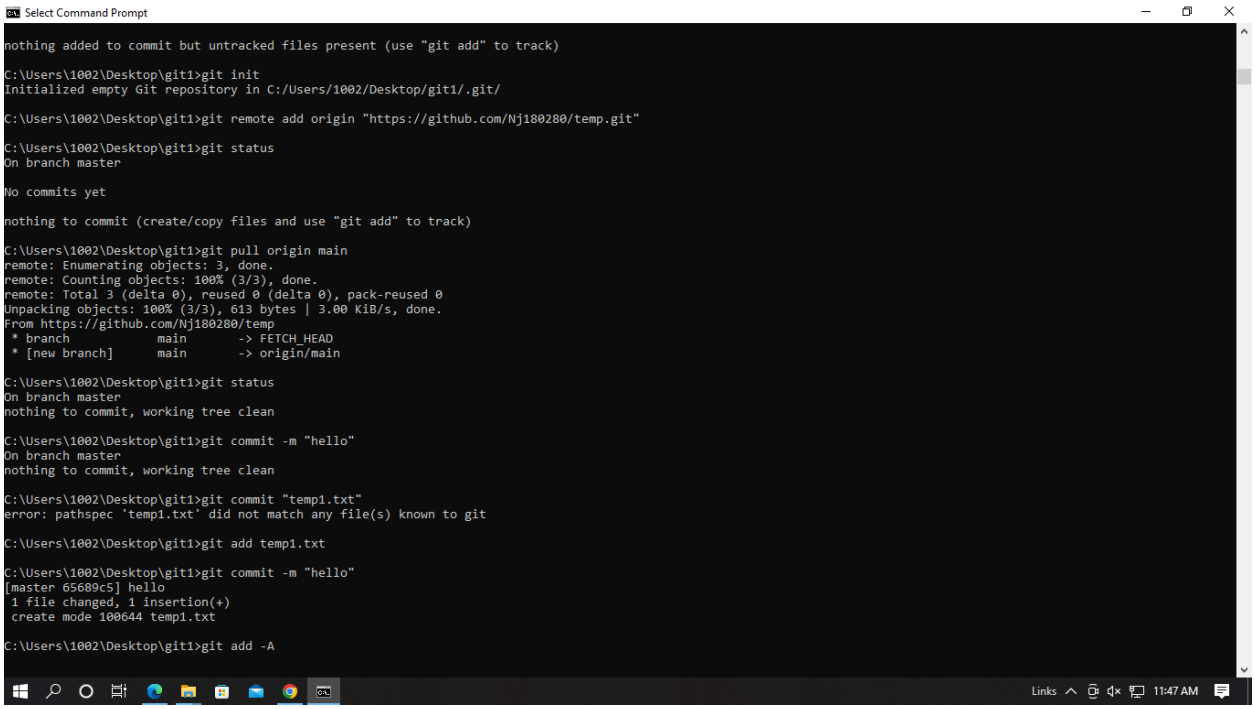
11) git merge
Utility : Merge two branches you were working on.
How to : Switch to branch you want to merge everything in. git
merge <:branch_you_want_to_merge:>

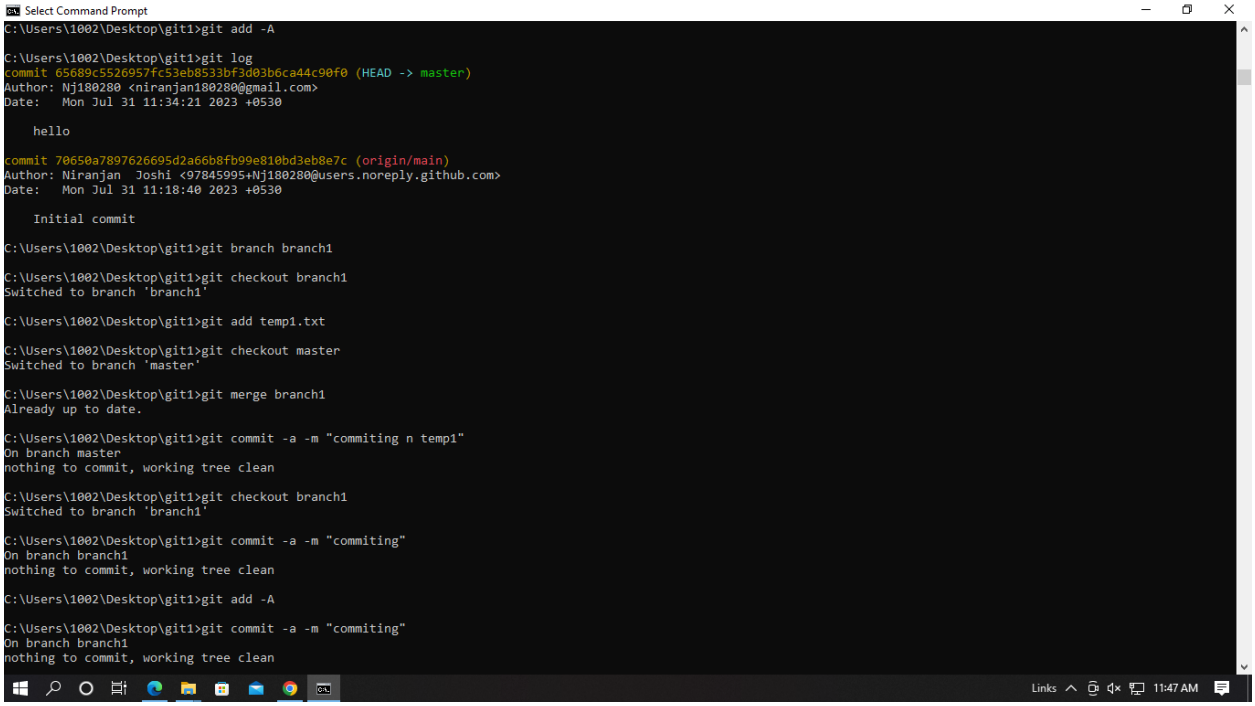
12) git reset
Utility : You know when you commit changes that are not complete, this sets your index to the latest commit that you want to work on with.
How to : git reset <:mode:> <:COMMIT:>

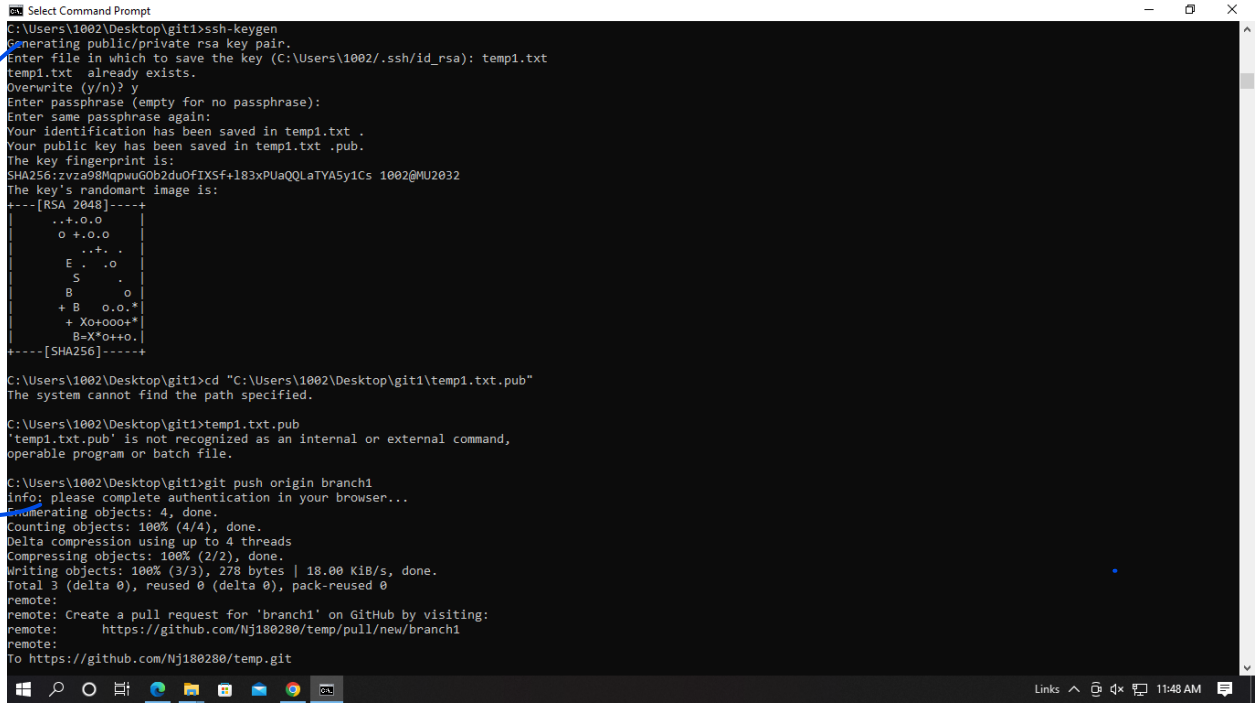
13) git remote
Utility : To check what remote/source you have or add a new remote.
How to : git remote to check and list. And git remote add <:remote_url:>

Output after execution of git commands:-


Command Prompt
to set your account's default identity.
Omit --global to set the identity only in this repository.
fatal: unable to auto-detect email address (got '1002@MU2032.(none)')
C:\Users\1002>git config --global user.name "Nj180280"
C:\Users\1002>git config --global user.email "niranjan180280@gmail.com"
C:\Users\1002>git init
Reinitialized existing Git repository in C:/Users/1002/.git/
C:\Users\1002>git add "hello.txt"
fatal: pathspec 'hello.txt' did not match any files
C:\Users\1002>git remote add origin "https://github.com/Nj180280/temp.git"
C:\Users\1002>git pull origin master
fatal: couldn't find remote ref master
C:\Users\1002>git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 613 bytes | 4.00 KiB/s, done.
From https://github.com/Nj180280/temp
* branch main -> FETCH_HEAD
* [new branch] main -> origin/main
C:\Users\1002>git status
warning: could not open directory 'AppData/Local/Application Data/': Permission denied
warning: could not open directory 'Application Data/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Local Settings/': Permission denied
warning: could not open directory 'My Documents/': Permission denied
warning: could not open directory 'NetHood/': Permission denied
warning: could not open directory 'PrintHood/': Permission denied
warning: could not open directory 'Recent/': Permission denied
warning: could not open directory 'SendTo/': Permission denied
warning: could not open directory 'Start Menu/': Permission denied
warning: could not open directory 'Templates/': Permission denied
On branch master
Untracked files:
(use "git add <file>..." to include in what will be committed)
Links
11:27 AM


Select Command Prompt
nothing added to commit but untracked files present (use "git add" to track)
C:\Users\1002\Desktop\git1>git init
Initialized empty Git repository in C:/Users/1002/Desktop/git1/.git/
C:\Users\1002\Desktop\git1>git remote add origin "https://github.com/Nj180280/temp.git"
C:\Users\1002\Desktop\git1>git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)
C:\Users\1002\Desktop\git1>git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 613 bytes | 3.00 KiB/s, done.
From https://github.com/Nj180280/temp
* branch main -> FETCH_HEAD
* [new branch] main -> origin/main
C:\Users\1002\Desktop\git1>git status
On branch master
nothing to commit, working tree clean
C:\Users\1002\Desktop\git1>git commit -m "hello"
On branch master
nothing to commit, working tree clean
C:\Users\1002\Desktop\git1>git commit "temp1.txt"
error: pathspec 'temp1.txt' did not match any file(s) known to git
C:\Users\1002\Desktop\git1>git add temp1.txt
C:\Users\1002\Desktop\git1>git commit -m "hello"
[master 65689c5] hello
1 file changed, 1 insertion(+)
create mode 100644 temp1.txt
C:\Users\1002\Desktop\git1>git add -A
Links
11:47 AM


Select Command Prompt
C:\Users\1002\Desktop\git1>git add -A
C:\Users\1002\Desktop\git1>git log
commit 65689c5526957fc53eb8533bf3d03b6ca44c90f0 (HEAD -> master)
Author: Nj180280 <niranjan180280@gmail.com>
Date: Mon Jul 31 11:34:21 2023 +0530
hello
commit 70650a7897626695d2a66b8fb99e810bd3eb8e7c (origin/main)
Author: Niranjan Joshi <97845995+Nj180280@users.noreply.github.com>
Date: Mon Jul 31 11:18:40 2023 +0530
Initial commit
C:\Users\1002\Desktop\git1>git branch branch1
C:\Users\1002\Desktop\git1>git checkout branch1
Switched to branch 'branch1'
C:\Users\1002\Desktop\git1>git add temp1.txt
C:\Users\1002\Desktop\git1>git checkout master
Switched to branch 'master'
C:\Users\1002\Desktop\git1>git merge branch1
Already up to date.
C:\Users\1002\Desktop\git1>git commit -a -m "commiting n temp1"
On branch master
nothing to commit, working tree clean
C:\Users\1002\Desktop\git1>git checkout branch1
Switched to branch 'branch1'
C:\Users\1002\Desktop\git1>git commit -a -m "commiting"
On branch branch1
nothing to commit, working tree clean
C:\Users\1002\Desktop\git1>git add -A
C:\Users\1002\Desktop\git1>git commit -a -m "commiting"
On branch branch1
nothing to commit, working tree clean
Links
11:47 AM


Select Command Prompt
C:\Users\1002\Desktop\git1>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\1002/.ssh/id_rsa): temp1.txt
temp1.txt already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in temp1.txt .
Your public key has been saved in temp1.txt .pub.
The key fingerprint is:
SHA256:zvza98MqpwuGOb2duOfIXSf+l83xPUaQQLaTYA5y1Cs 1002@MU2032
The key's randomart image is:
+---[RSA 2048]----+
| ..+.o.o |
| o +.o.o |
| ..+. . |
| E . .o |
| S . |
| B o |
| + B o.o.*|
| + Xo+ooo+*|
| B=X*o++o.|
+----[SHA256]-----+
C:\Users\1002\Desktop\git1>cd "C:\Users\1002\Desktop\git1\temp1.txt.pub"
The system cannot find the path specified.
C:\Users\1002\Desktop\git1>temp1.txt.pub
'temp1.txt.pub' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\1002\Desktop\git1>git push origin branch1
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes | 18.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote: https://github.com/Nj180280/temp/pull/new/branch1
remote:
To https://github.com/Nj180280/temp.git
Links
11:48 AM

```
C:\Users\1002>cd desktop

C:\Users\1002\Desktop>git remote -v
origin  https://github.com/Nj180280/temp.git (fetch)
origin  https://github.com/Nj180280/temp.git (push)

C:\Users\1002\Desktop>git  push origin main
Everything up-to-date
```

```
C:\Users\1002\Desktop\temp>git commit -m "commit message"
[master (root-commit) 4623ba9] commit message
 1 file changed, 1 insertion(+)
 create mode 100644 temp1.txt

C:\Users\1002\Desktop\temp>git commit -am "commit next"
On branch master
nothing to commit, working tree clean

C:\Users\1002\Desktop\temp>git commit --amend -m "new commit"
[master a056b05] new commit
 Date: Mon Aug 7 10:51:46 2023 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 temp1.txt

C:\Users\1002\Desktop\temp>git remote

C:\Users\1002\Desktop\temp>git branch
* master

C:\Users\1002\Desktop\temp>git branch branch2

C:\Users\1002\Desktop\temp>git branch
  branch2
* master

C:\Users\1002\Desktop\temp>git checkout branch2
Switched to branch 'branch2'
```

```
C:\Users\1002\Desktop\temp>git pull
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 859 bytes | 3.00 KiB/s, done.
From https://github.com/Nj180280/temp
 * [new branch]      branch1    -> temp/branch1
 * [new branch]      main       -> temp/main
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.
```

Conclusion:-

Explored various git operations and commands used to perform operation on local git and uploaded all files on git repository created during process

Experiment 4

Aim:-

To understand continuous integration , jenkins installation

LO Mapped :- LO1 , LO3

Theory:-

Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more.
You can set up Jenkins to watch for any code changes in places like GitHub, Bitbucket or GitLab and automatically do a build a with tools like Maven and Gradle. You can utilize container technology such as Docker and Kubernetes, initiate tests and then take actions like rolling back or rolling forward in production.

Jenkins History

The Jenkins project was started in 2004 (originally called Hudson) by Kohsuke Kawaguchi, while he worked for Sun Microsystems. Kohsuke was a developer at Sun and got tired of incurring the wrath of his team every time his code broke the build. He created Jenkins as a way to performcontinuous integration – that is, to test his code before he did an actual commit to the repository,to be sure all was well. Once his teammates saw what he was doing, they all wanted to use Jenkins.Kohsuke open sourced it, creating the Jenkins project, and soon Jenkins usage had spread around the world.

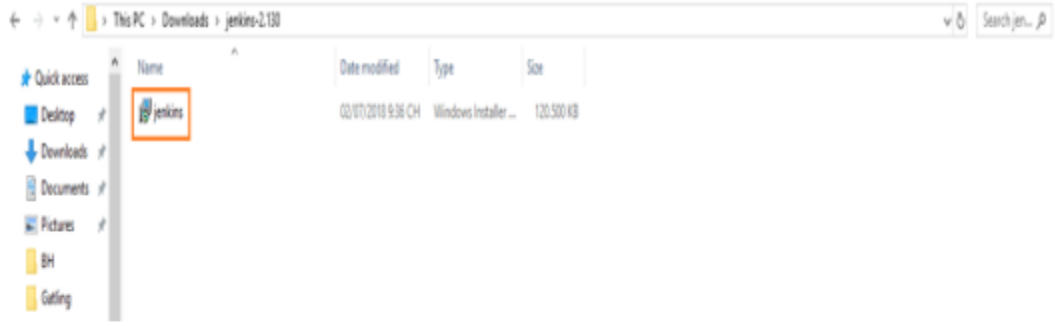
Jenkins currently

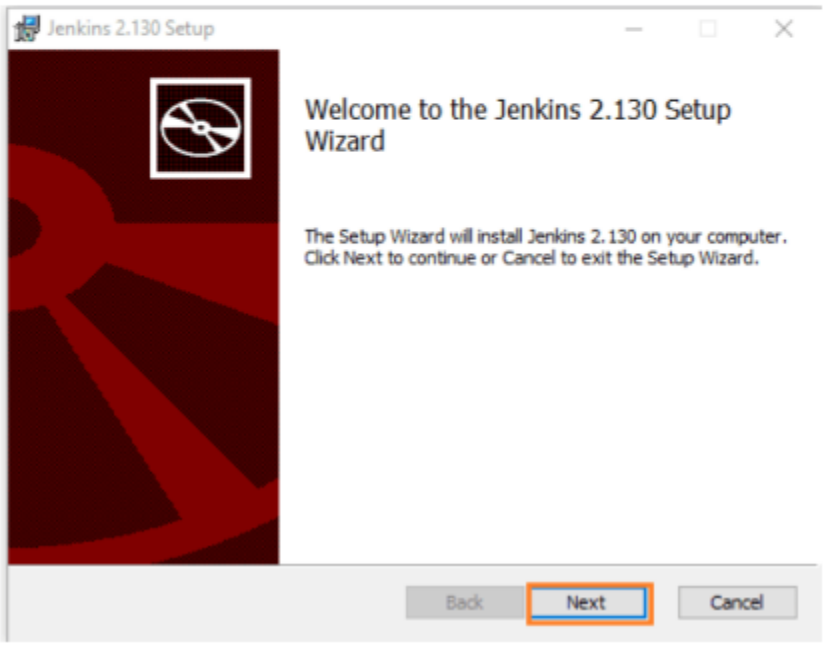
Originally developed by Kohsuke for continuous integration (CI), today Jenkins orchestrates the entire software delivery pipeline – called continuous delivery. For some organizations automationextends even further, to continuous deployment. Continuous delivery (CD), coupled with a DevOps culture, dramatically accelerates the delivery of software.
Jenkins is the most widely adopted solution for continuous delivery, thanks to its extensibility and a vibrant, active community. The Jenkins community offers more than 1,700 plugins that enable Jenkins to integrate with virtually any tool, including all of the best-of-breed solutions used throughout the continuous delivery process. Jenkins continues to grow as the dominant solution for software process automation, continuous integration and continuous delivery and, as of February 2018, there are more than 165,000 active installations and an estimated 1.65 million users around the world. Jenkins offers a simple way to set up a continuous integration or continuous delivery (CI/CD)

environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks. While Jenkins doesn't eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

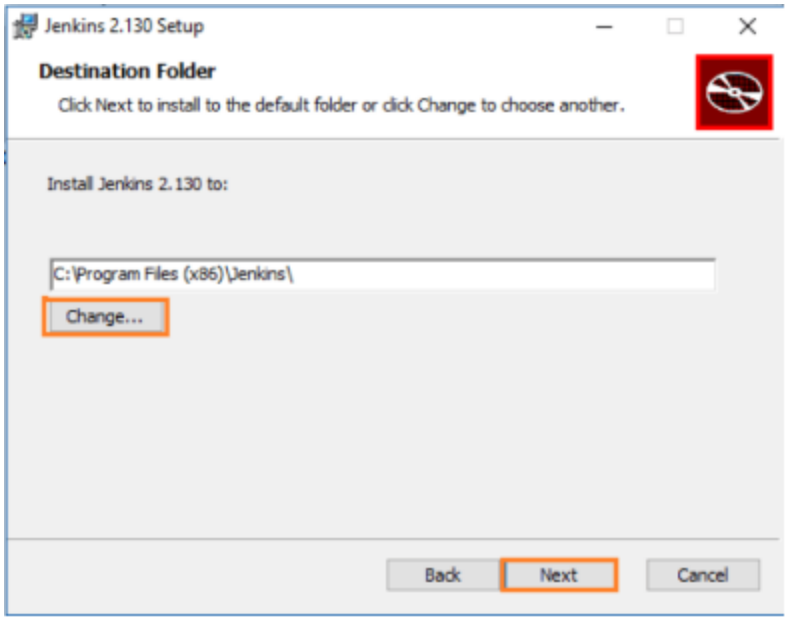
Steps to install and configure Jenkins :-

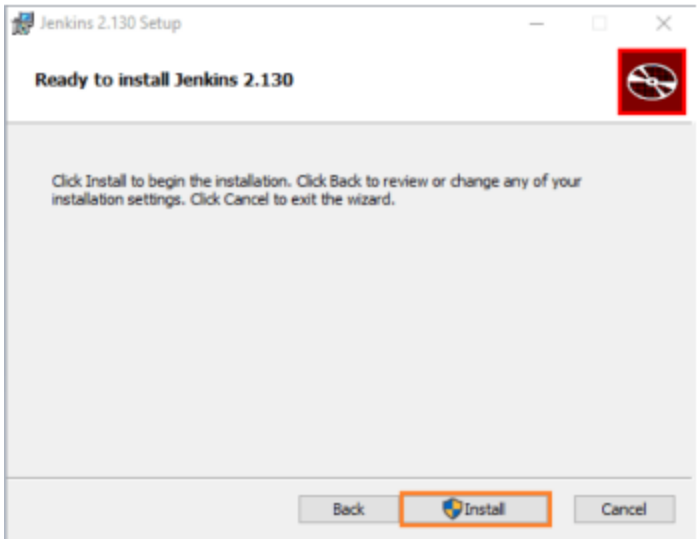
1)Installation of Jenkins in local system , click on setup



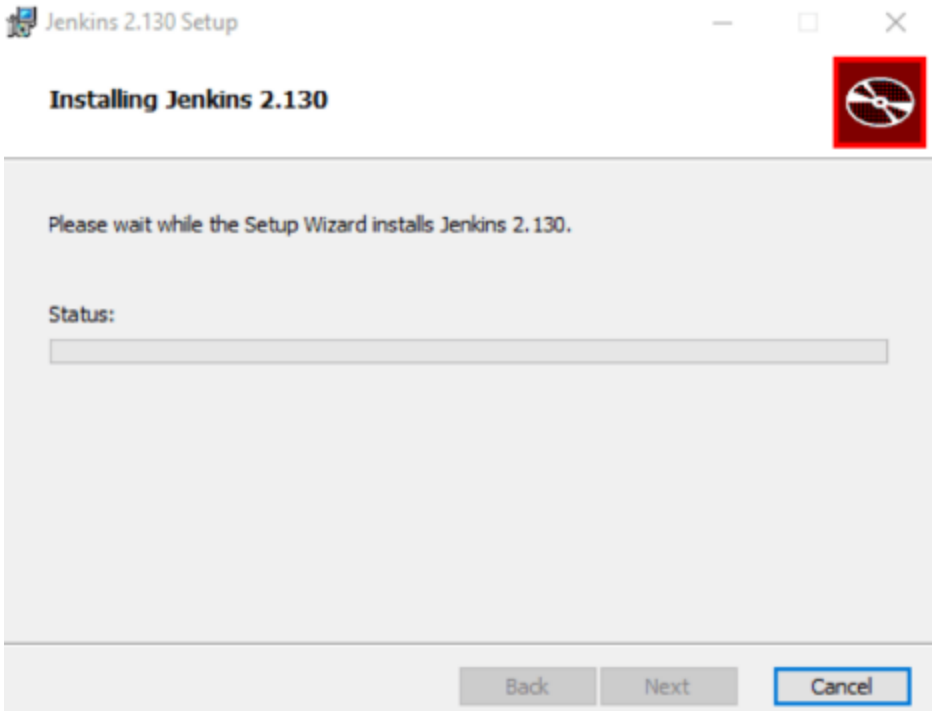


2)Then choose destination folder and click on install

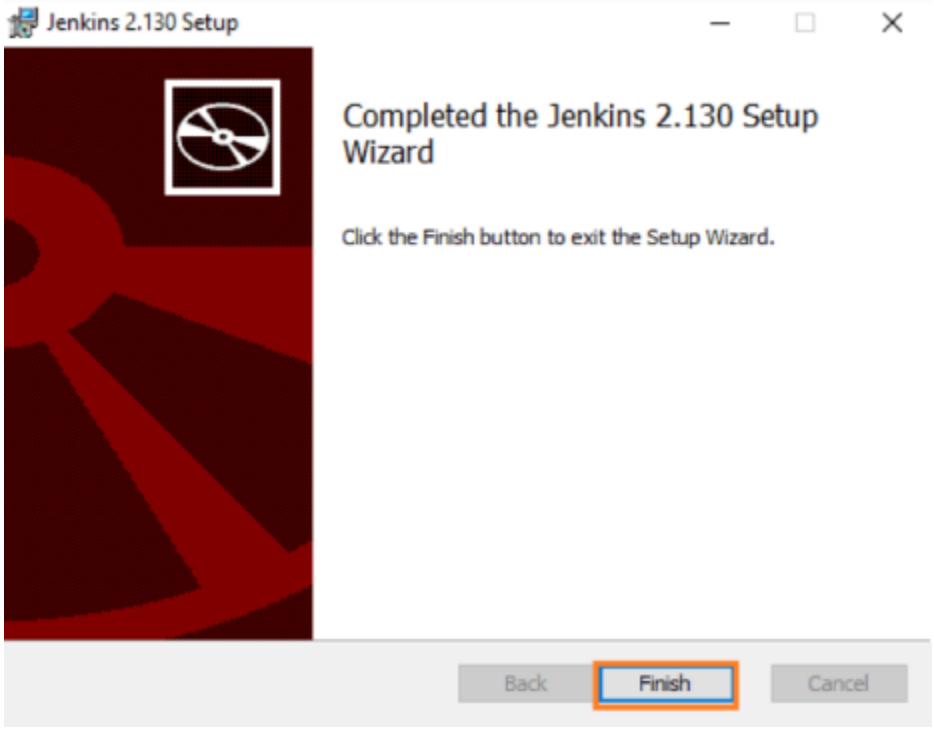




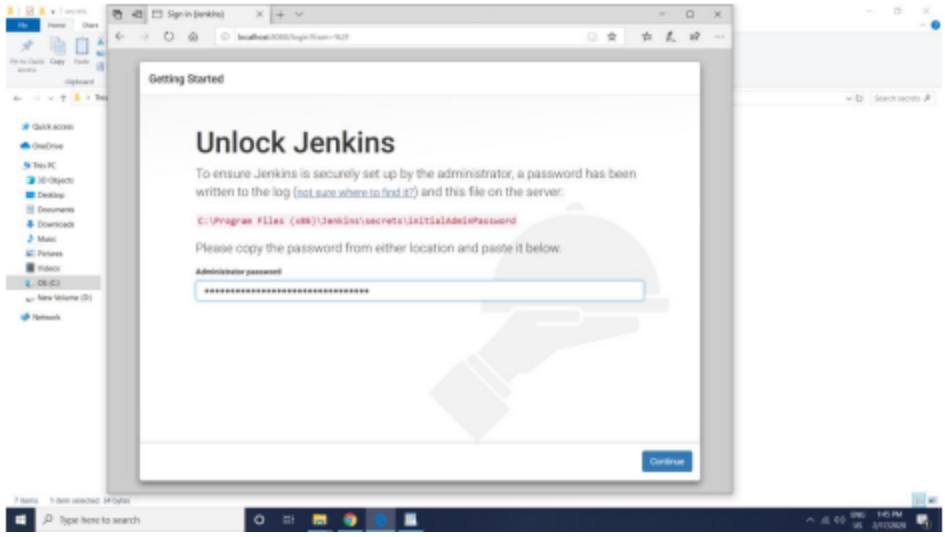
3)Then installation will be proceed further



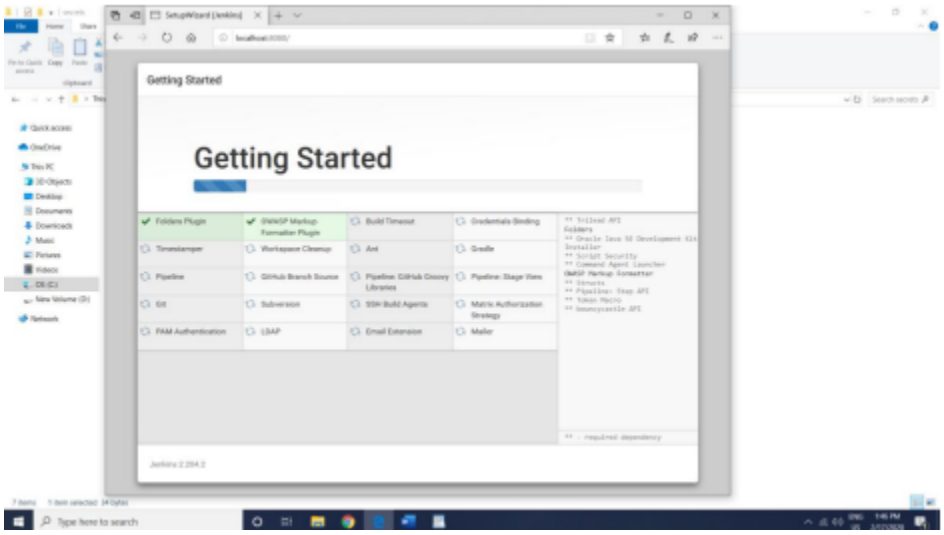
4)Then click on finish once installation is complete



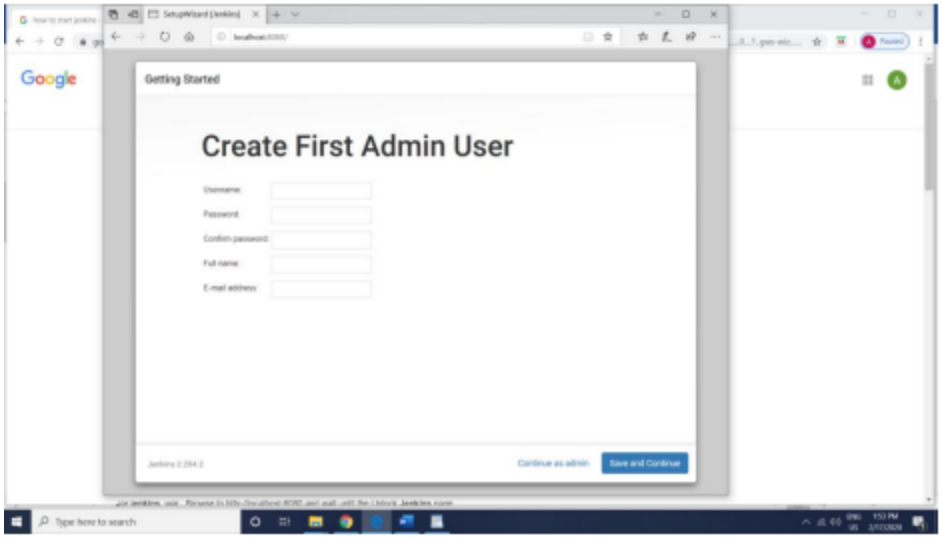
5)Then in secret folder there is initial login password , copy and paste it



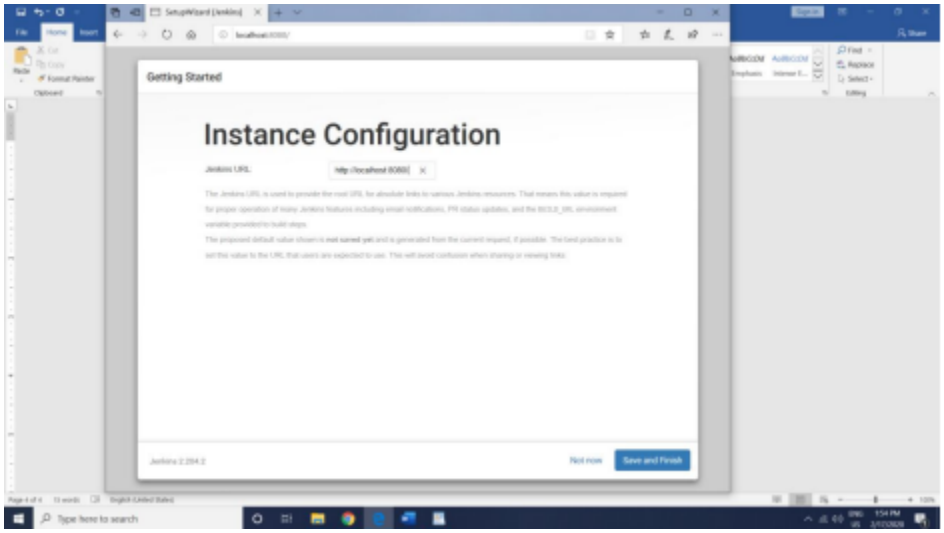
6)Then jenkins will further configure



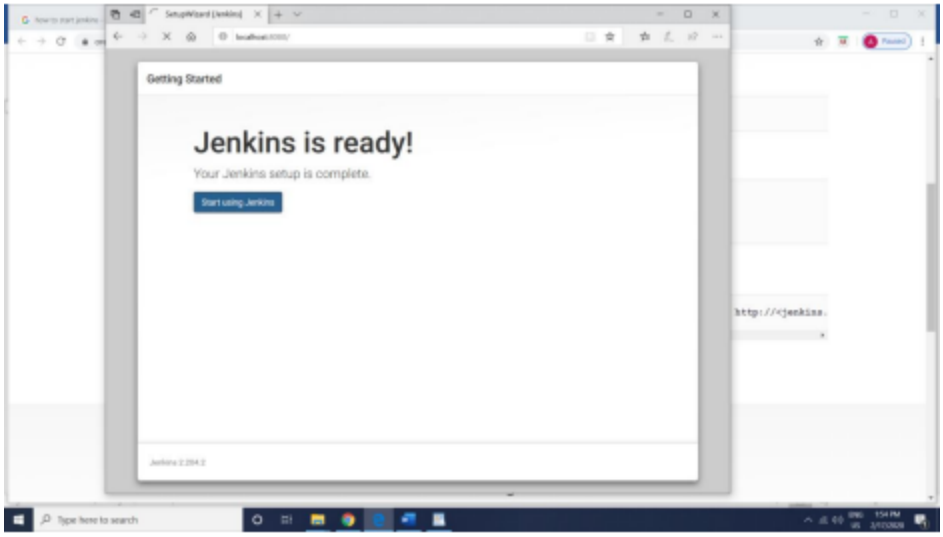
7)create the first admin by entering username , password etc



8)Then configure the instance of Jenkins



9)Then installation and configuration of Jenkins is done and Jenkins is ready



Conclusion:-

We have successfully installed Jenkins in our local system and also have configured Jenkins by creating first user , password and username was required here .

Experiment 5

Aim :- to build java program using Jenkins

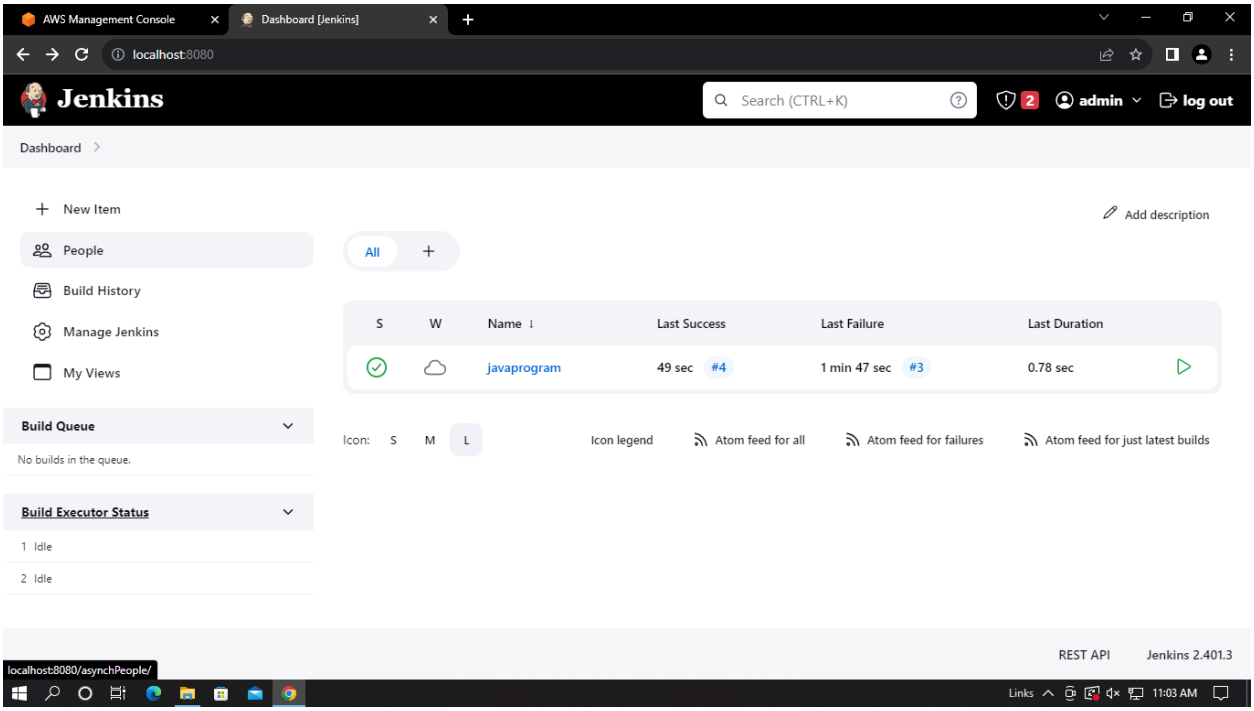
LO Mapped : LO1 , LO3

Theory (Steps) :-

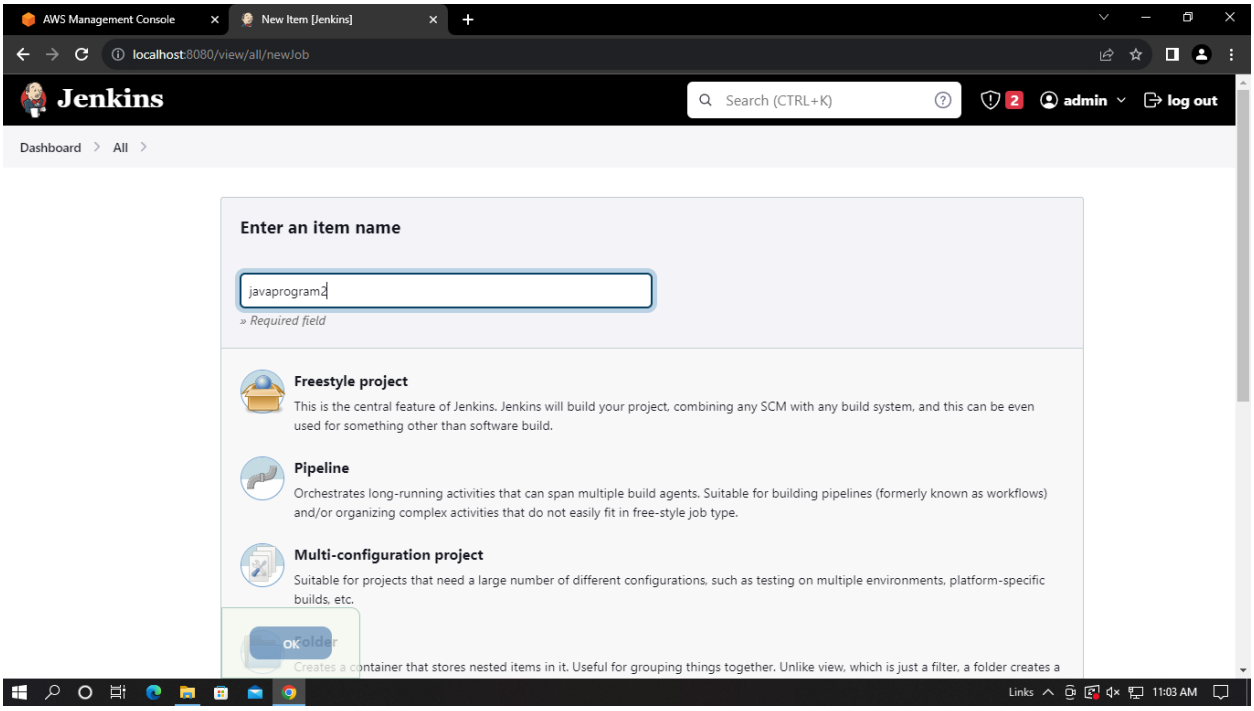
1)Java program (hello.java)

```
class hello{
 public static void main(String args[]){
  int i,fact=1;
  int number=5;//It is the number to calculate factorial
  for(i=1;i<=number;i++){
      fact=fact*i;
  }
  System.out.println("Factorial of "+number+" is: "+fact);
 }
}
```

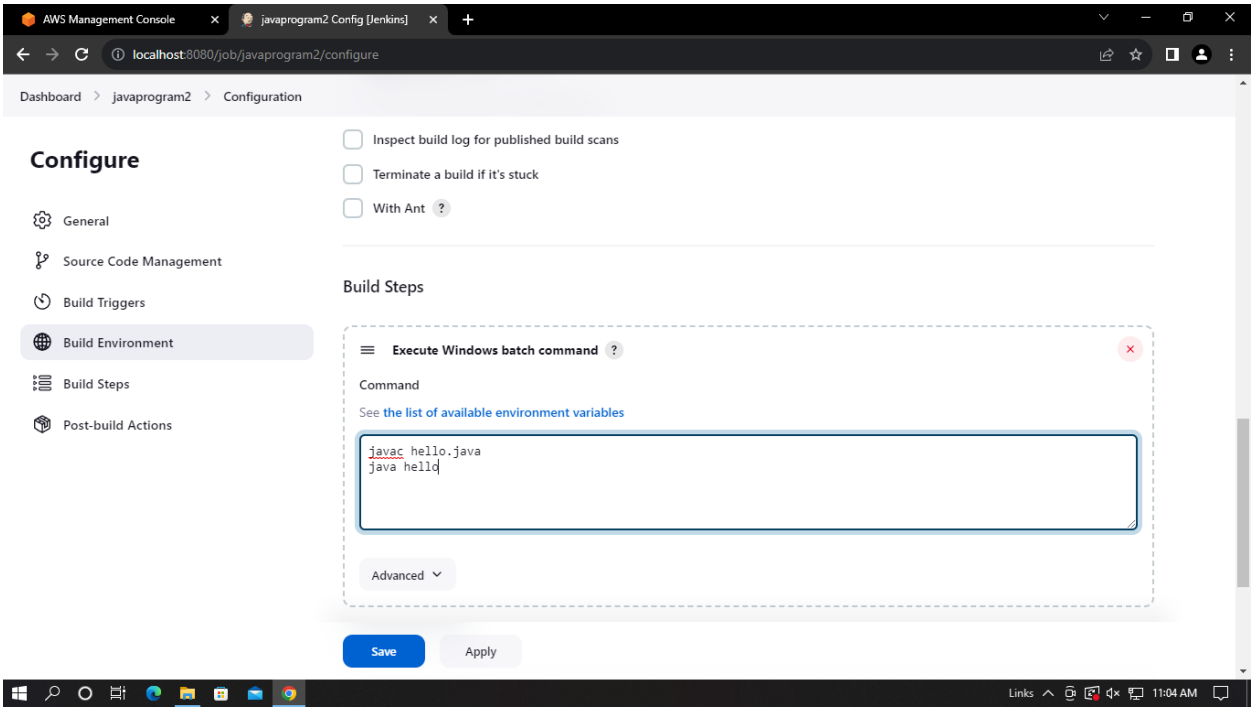
2)Create a job in jenkins to run the program by clicking on new item



3)enter the item name and select freestyle project

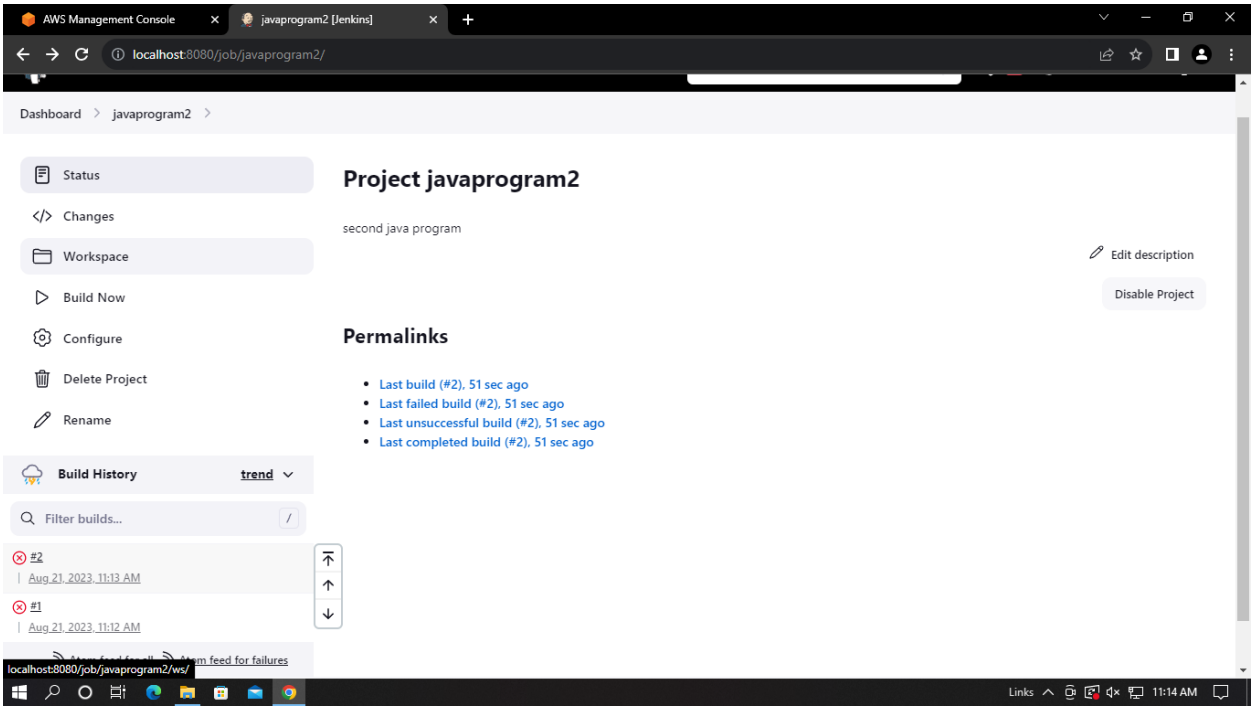


4)Then scroll down and in build steps click on execute windows batch command and enter two commands as shown below

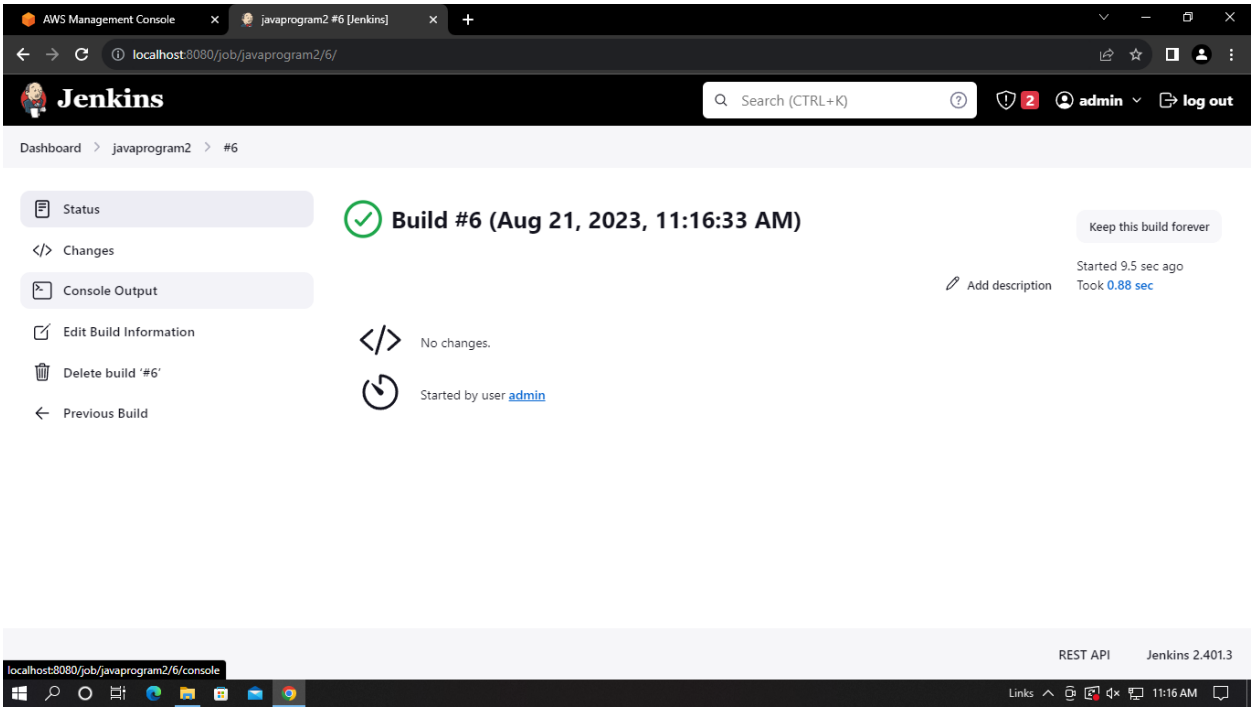


In above image hello.java is program name and it can be any name

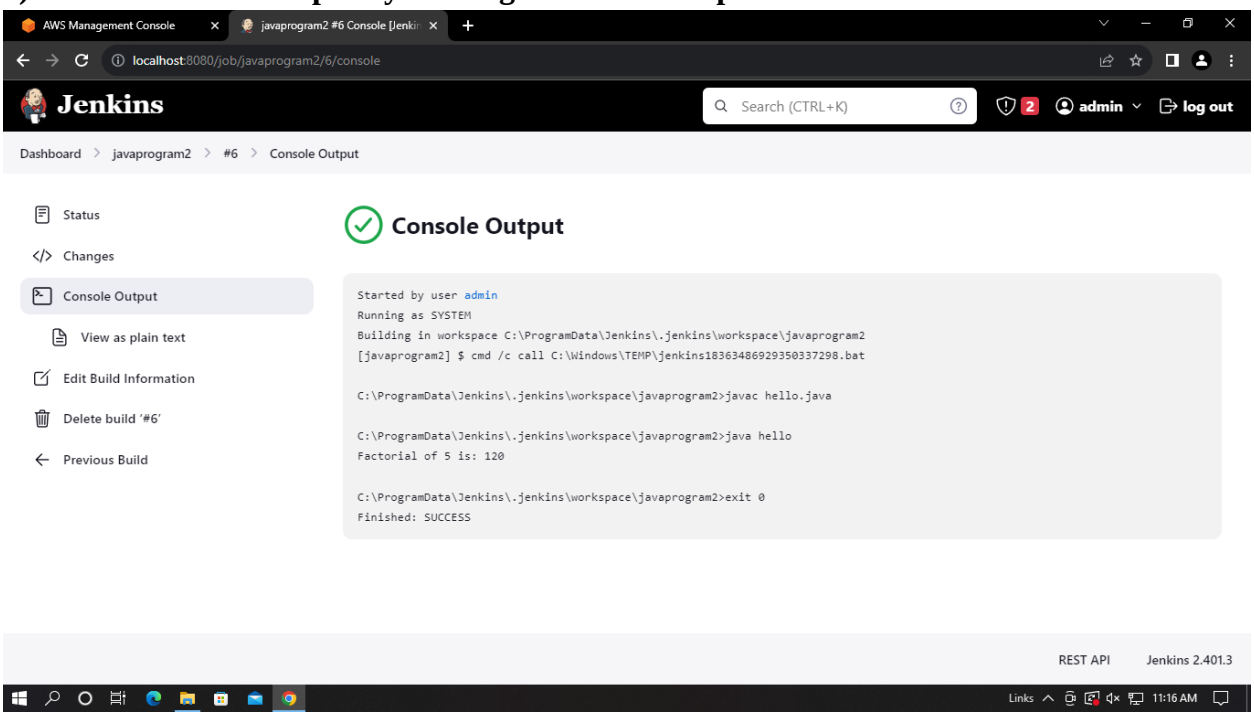
5)Then write the code and save it into workspace for Jenkins in its folder



6)Then click on build now and if build is success then green arrow is displayed



7)After this see the output by clicking on console output



Conclusion :-

We have created a Java program that prints factorial for a given number and have used Jenkins to build the program and checked console output in Jenkins itself , hence java code is executed

Assignment 6

Aim :- To build pipeline using Jenkin

LO Mapped :- LO1 , LO3

Theory :-

A Jenkins Pipeline is a **suite of plugins** that supports implementing and integrating continuous delivery pipelines into Jenkins. It provides a way to define a sequence of build, test, and deployment actions as code, allowing you to automate and manage the entire software delivery process.

Jenkins Pipelines come in **two flavors: Declarative and Scripted.**

Declarative Pipeline: This is a more structured and opinionated way of defining your pipeline. It uses a simplified syntax and enforces some best practices. It's often easier to read and maintain.

Scripted Pipeline: This is more flexible and powerful, as it allows you to write your pipeline using Groovy script directly. It's suitable for more complex scenarios but can be harder to maintain.

A Jenkins Pipeline consists of a series of stages, which represent distinct steps in your software delivery process, such as building, testing, and deploying. Each stage contains one or more steps, which are individual actions performed within that stage. These steps can include running commands, triggering other jobs, sending notifications, and more.

Continuous Delivery pipelines are automated workflows that manage the process of turning code changes into working software. They include steps like building, testing, and deploying code. When developers make changes, the pipeline automatically compiles, tests, and even deploys the changes to different environments. This ensures that the code is high quality and ready for release. The ultimate goal is to deliver reliable software quickly and consistently.

A Jenkinsfile is a text file that defines the pipeline configuration and workflow as code. It's used in Jenkins Pipelines to automate the entire software delivery process, from building and testing to deploying and monitoring. There are two main syntax options for writing Jenkinsfiles: Declarative and Scripted.

Declarative Syntax:
Here's an example of a simple Declarative Jenkinsfile:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'echo "Building..."'
            }
        }
        stage('Test') {
            steps {
                sh 'echo "Testing..."'
            }
        }
        stage('Deploy') {
            steps {
                sh 'echo "Deploying..."'
            }
        }
    }
}
```

Scripted Syntax:

```
node {
    stage('Build') {
        echo 'Building...'
    }
    stage('Test') {
        echo 'Testing...'
    }
    stage('Deploy') {
        echo 'Deploying...'
    }
}
```

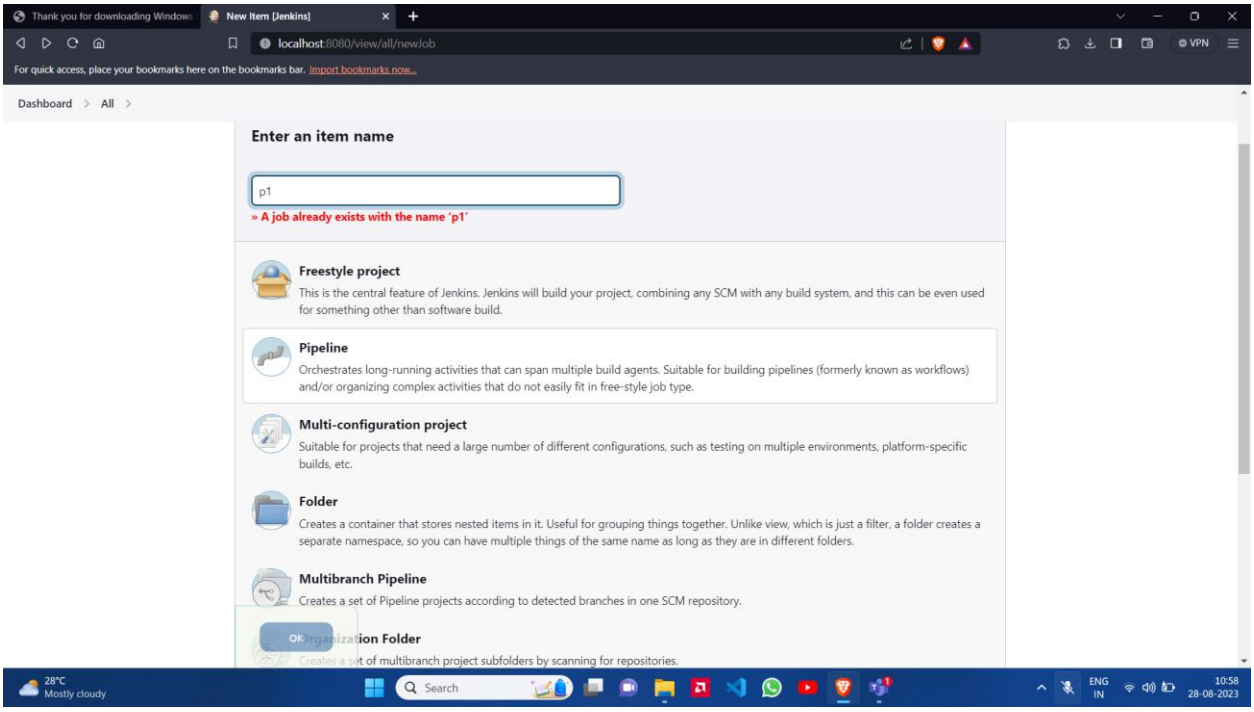
With Jenkins Pipelines, you can define your entire CI/CD workflow as code, which brings several benefits:

Reproducibility: The entire pipeline configuration is versioned alongside your code, ensuring consistent builds across different environments.
Traceability: The pipeline code documents the entire delivery process, making it easy to track changes and understand the workflow.
Automated Testing: You can test your pipeline itself, treating it as code, ensuring its correctness before deploying it.
Flexibility: Pipelines can be adapted to complex workflows and customized to your project's needs.
Integration: Pipelines can integrate with version control systems, issue trackers, and other tools in your software development ecosystem.

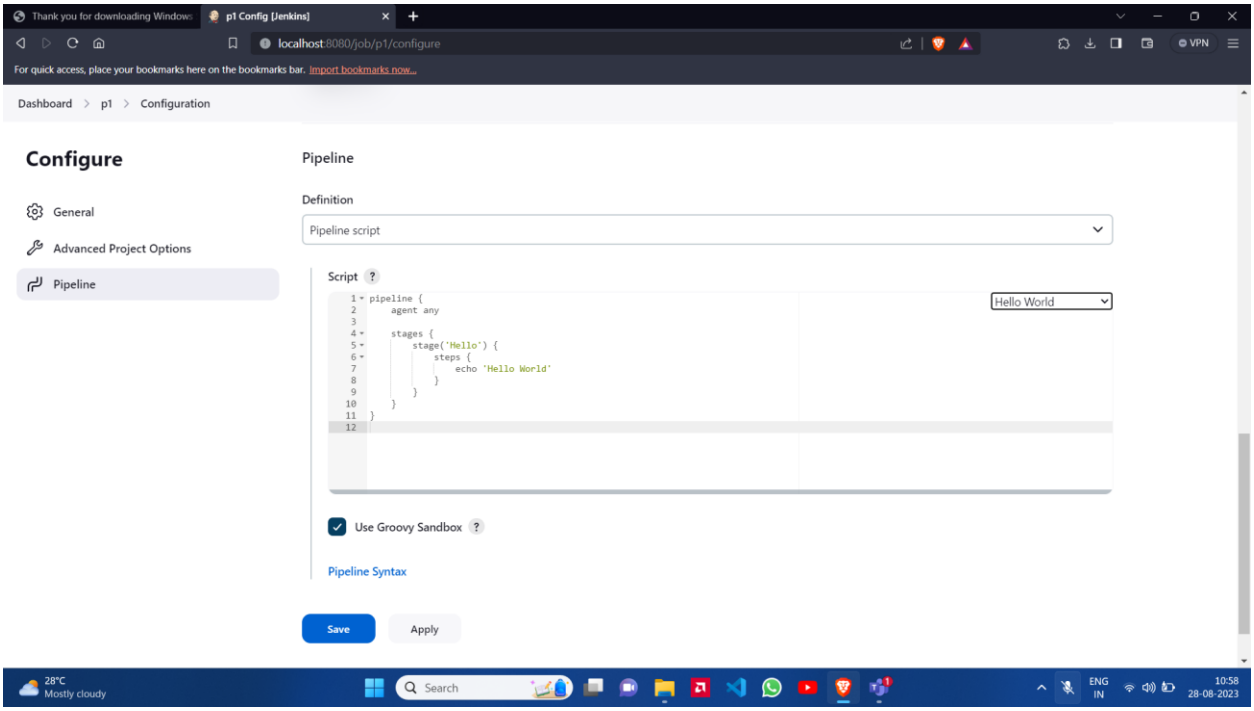
Output steps and screenshots :-

a) For hello world script

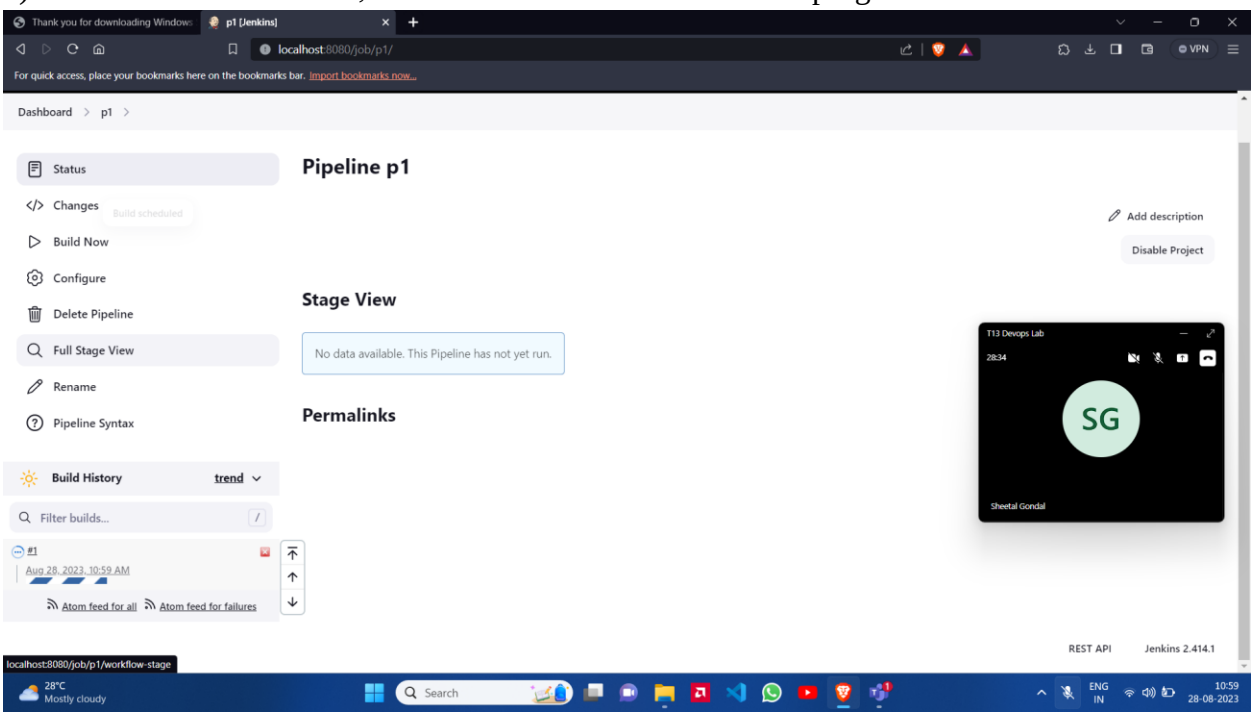
1)click on new item on user dashboard and then select pipeline as item and enter any name



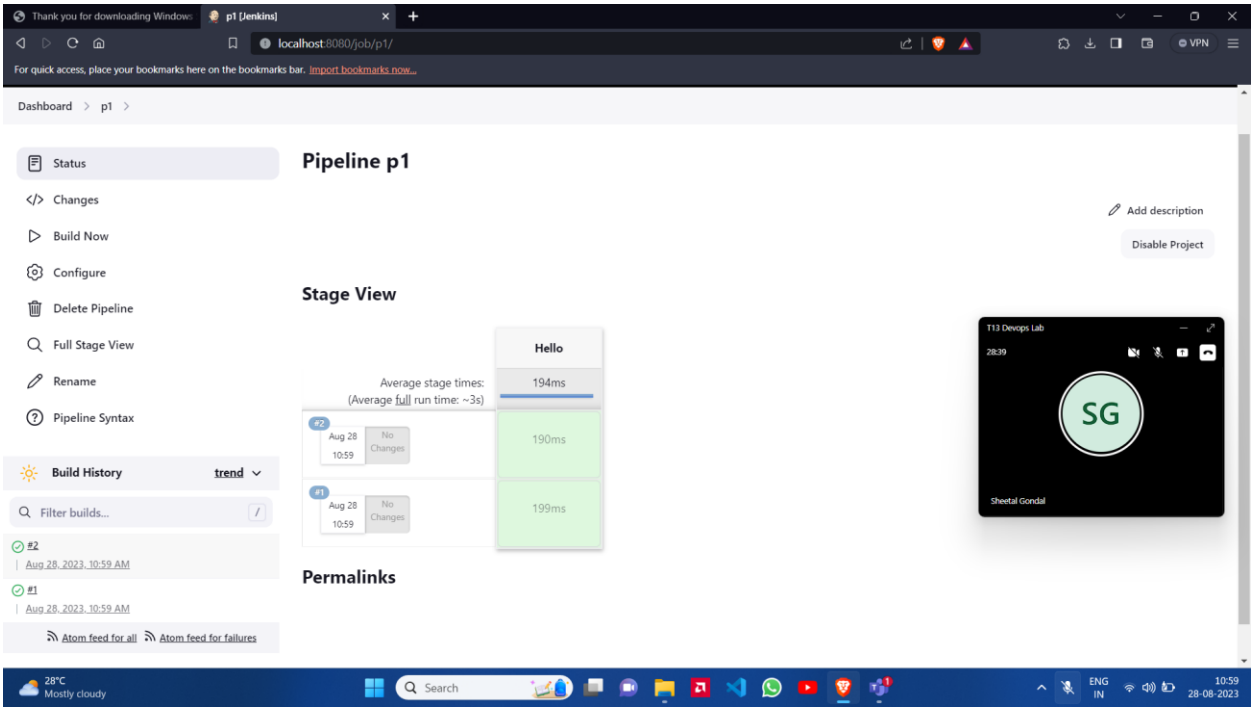
2) select predefined script as hello world in pipeline script section



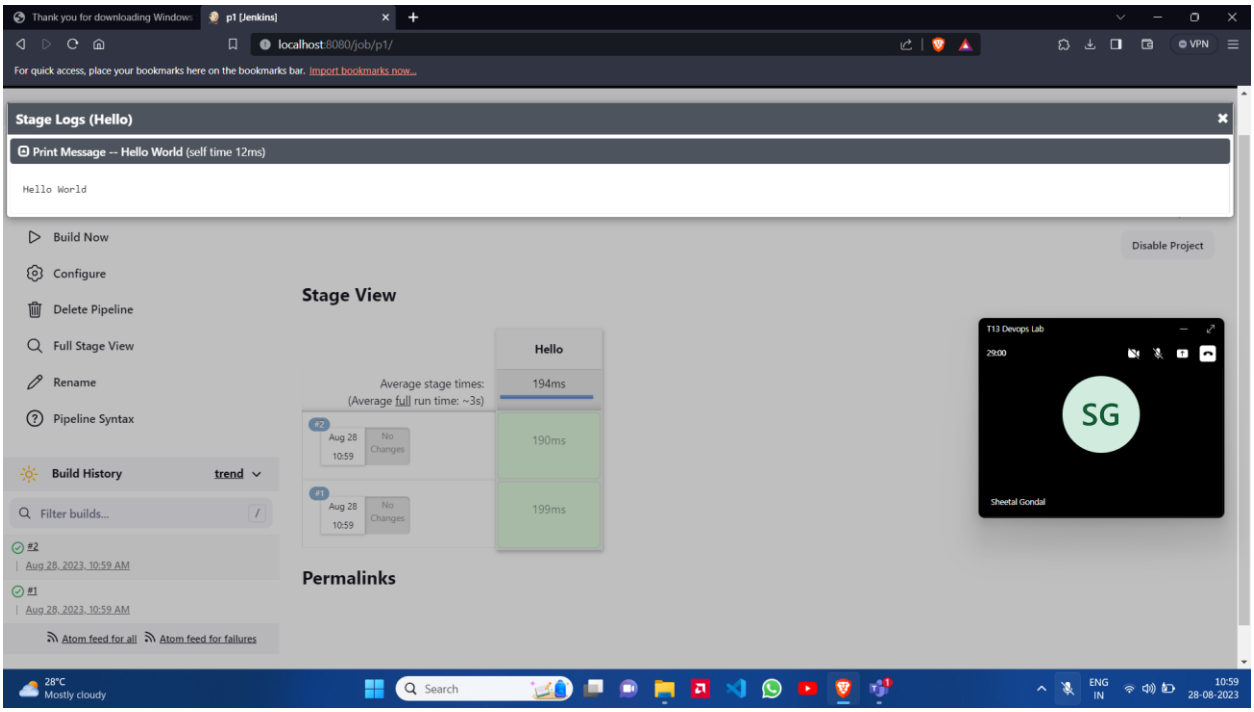
3) Then click on build now , at bottom left corner there will be progress of build shown



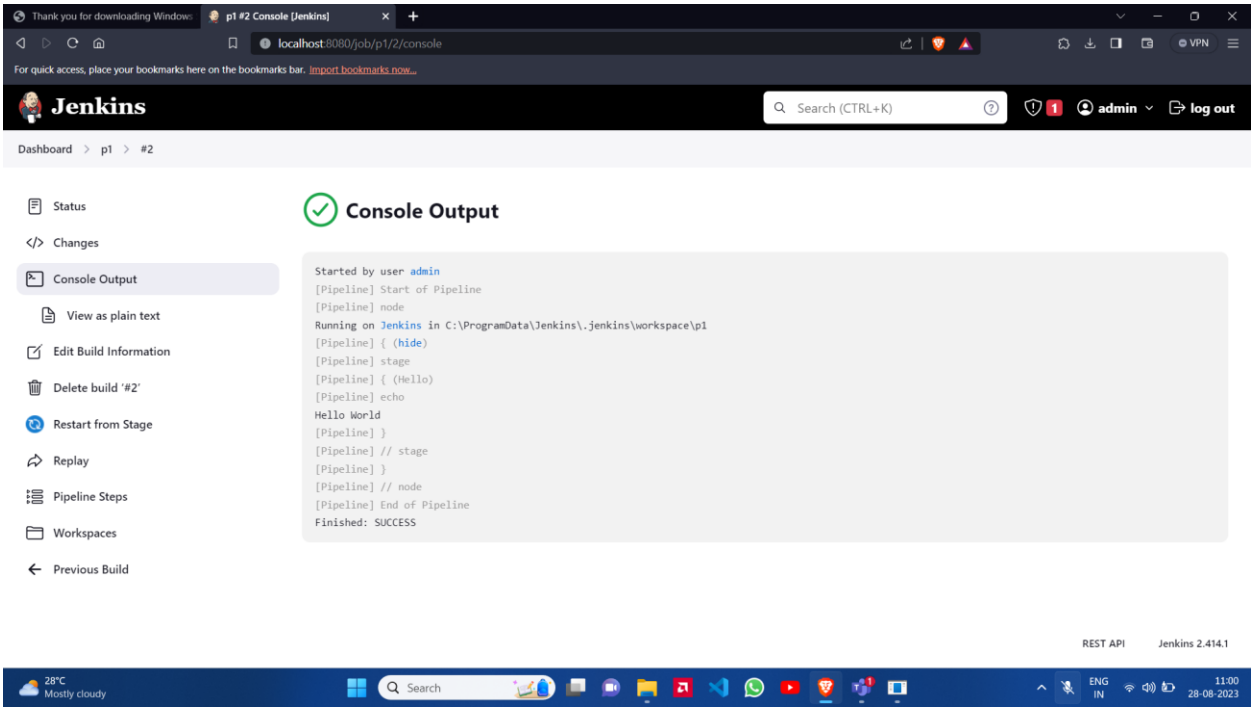
4)Then pipeline block will be seen at center of screen and shows completion of each step in pipeline



5)Then if we hover over green section , logs option is present and if we click on it , we can get output as hello world

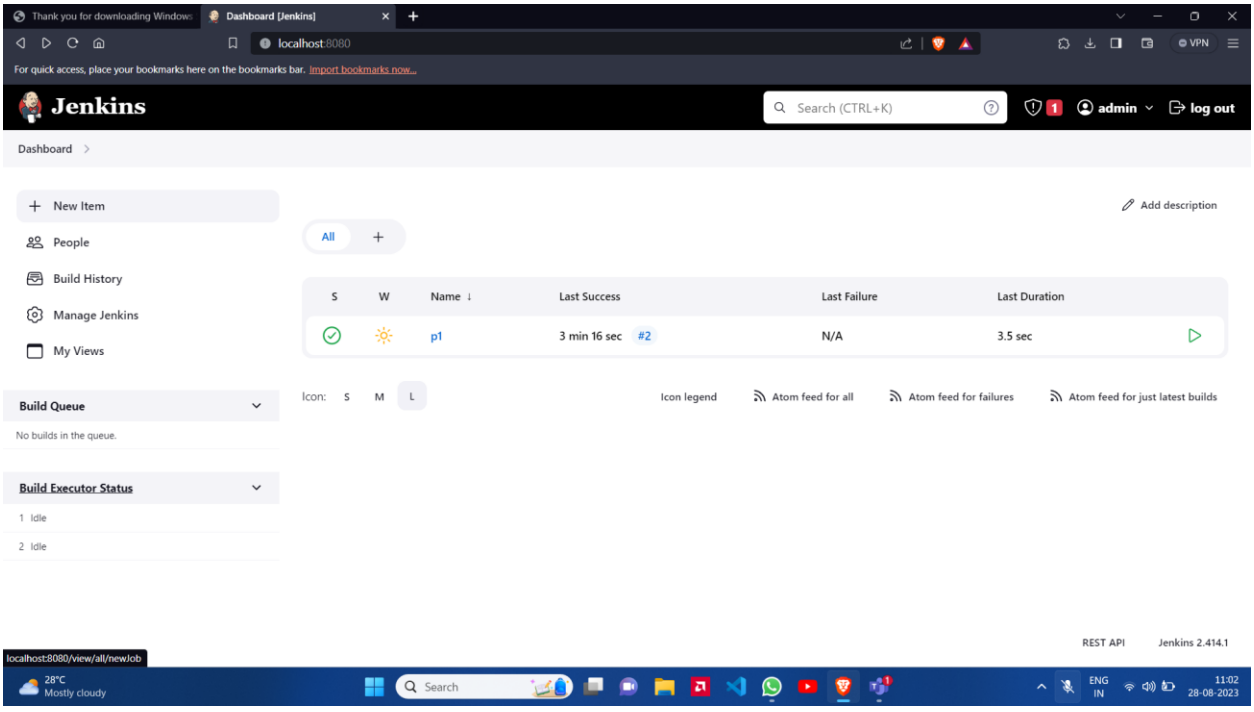


6) Output can also be seen on clicking onto process at bottomleft corner and then clicking on console log

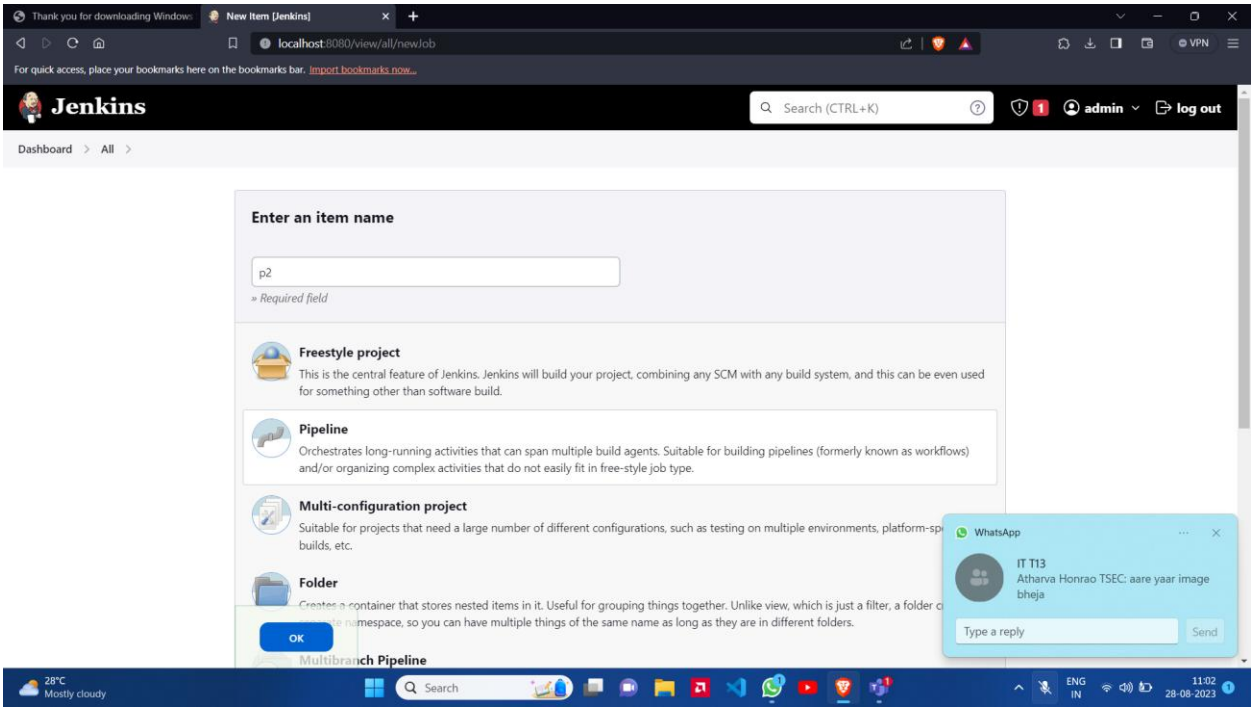


b) using custom pipeline script

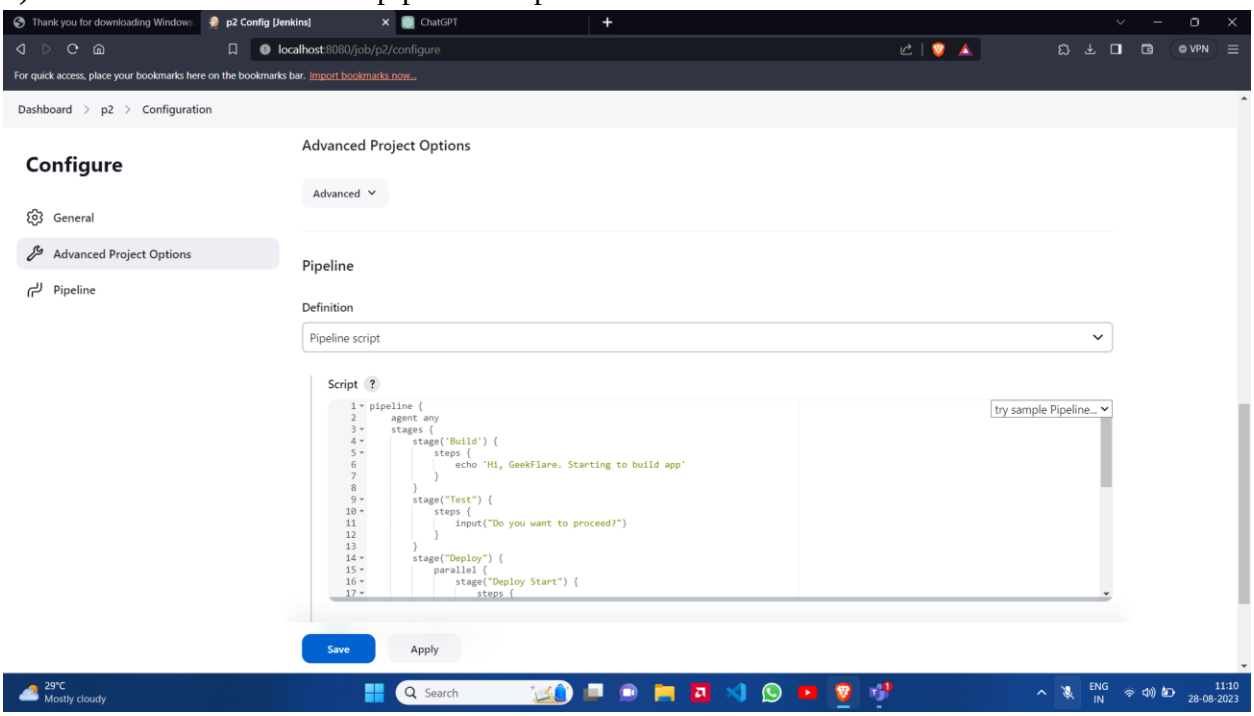
1)create a new item



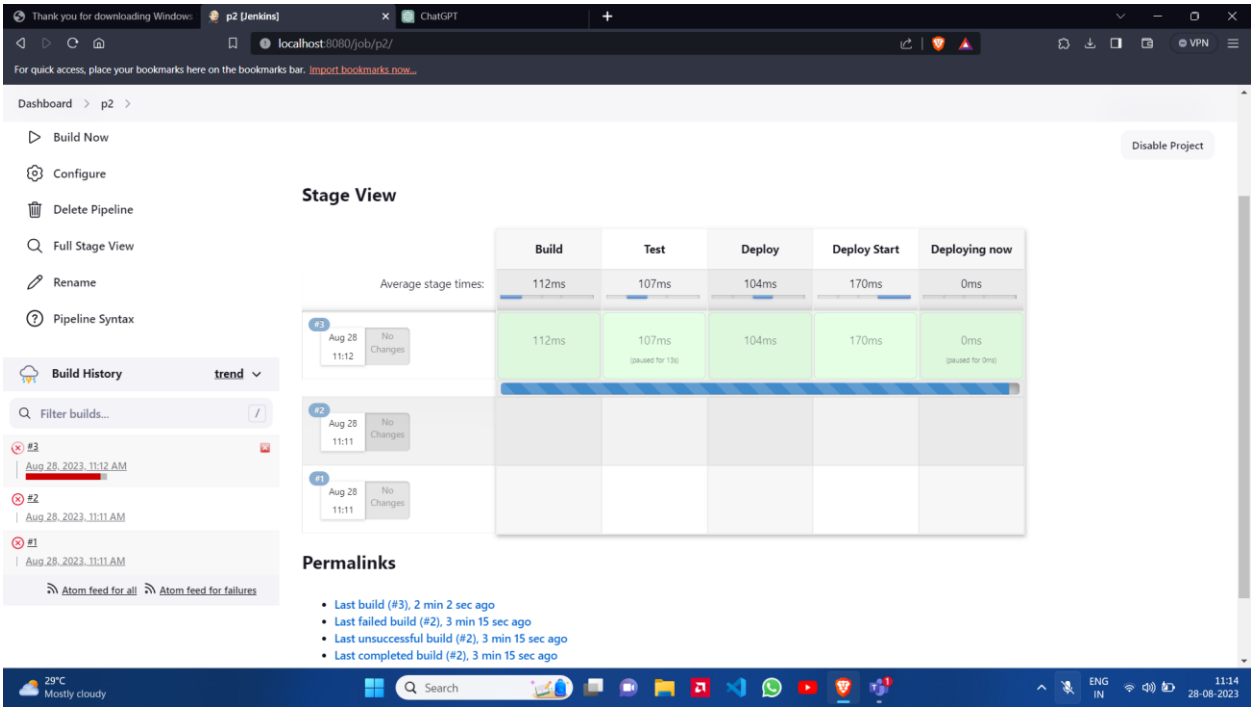
2)then give pipeline a name and select pipeline option



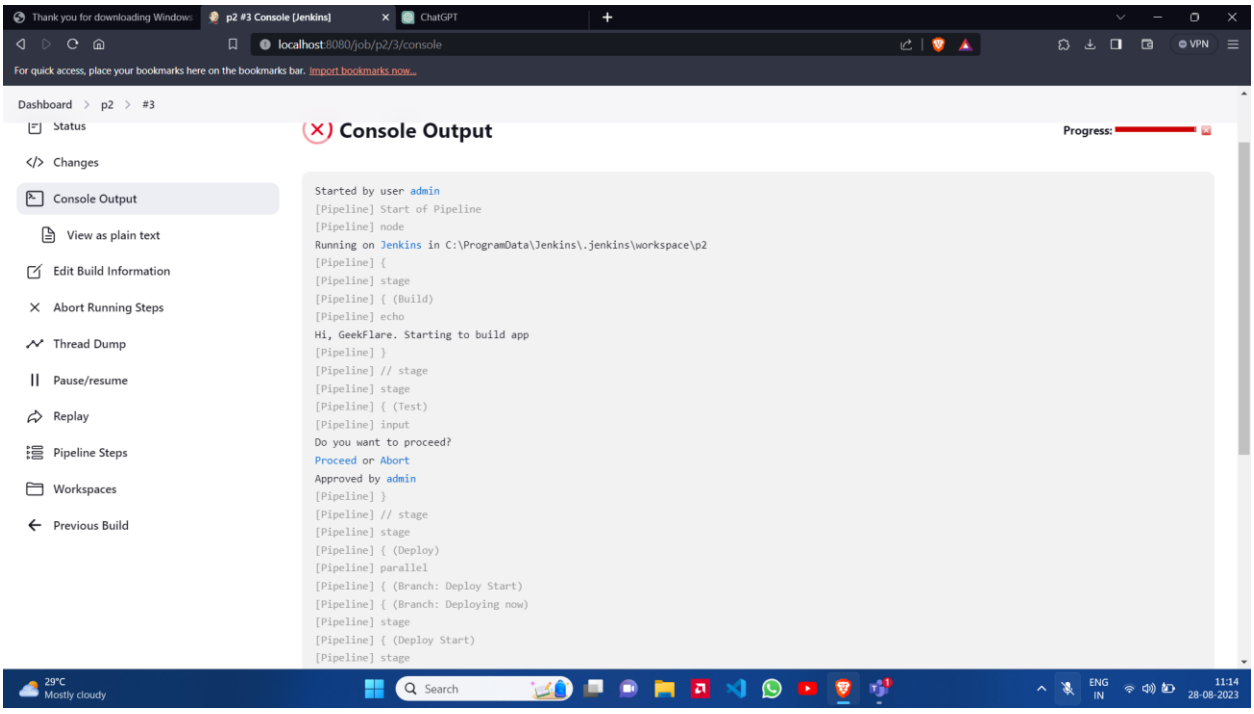
3)Then write the code inside pipeline script block



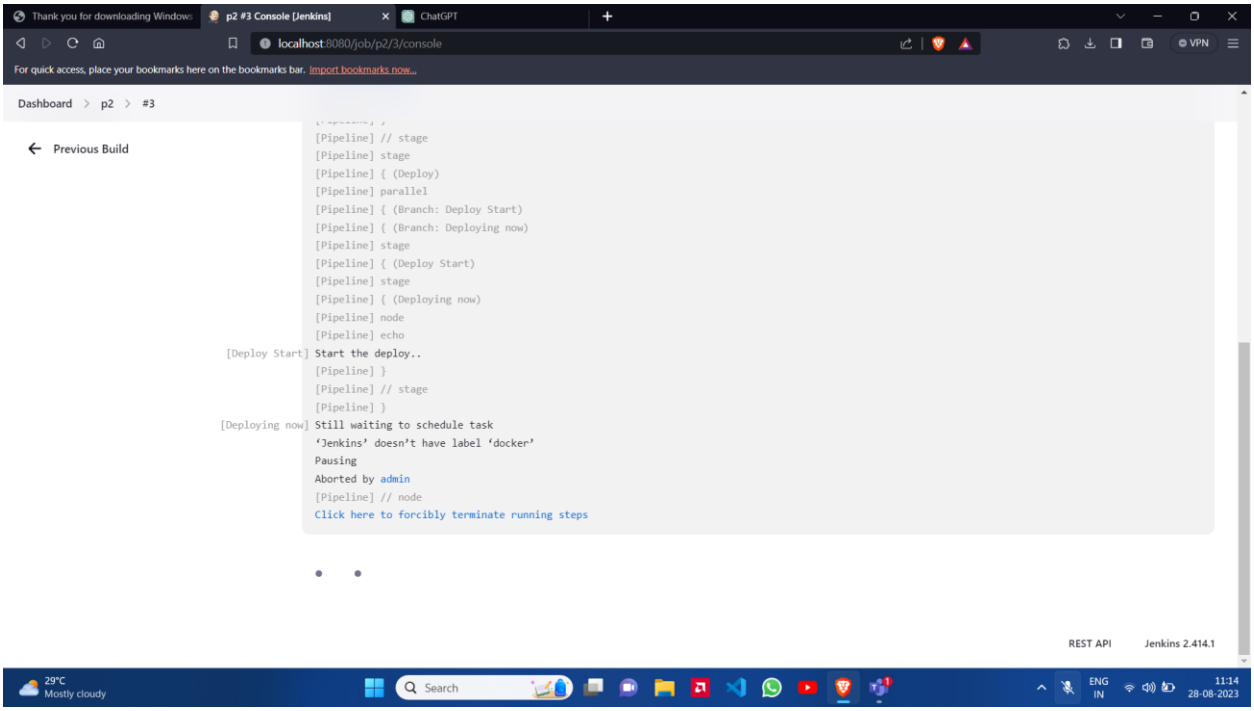
4)Then click on build now and progress of every step is shown , completed step is present into green box



5)Then we can see the console output , as docker is not installed execution of all steps cannot be completed hence red cross is shown above



6)Further scroll down to see complete output



Conclusion :-

We have created a pipeline in Jenkins that automates the work flow for a given pipeline script and there were two pipelines made one was predefined and one was custom , hence learnt more about pipeline and its advantages

Assignment 7

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

LO Mapped :- LO1 , LO5

Theory :-

Docker is a popular platform for developing, shipping, and running applications in containers. Containers are lightweight, portable, and self-sufficient environments that contain all the necessary components to run an application, including the code, runtime, libraries, and dependencies. Docker provides a way to package applications and their dependencies into a standardized format, making it easier to deploy and manage software across different environments, such as development, testing, and production.

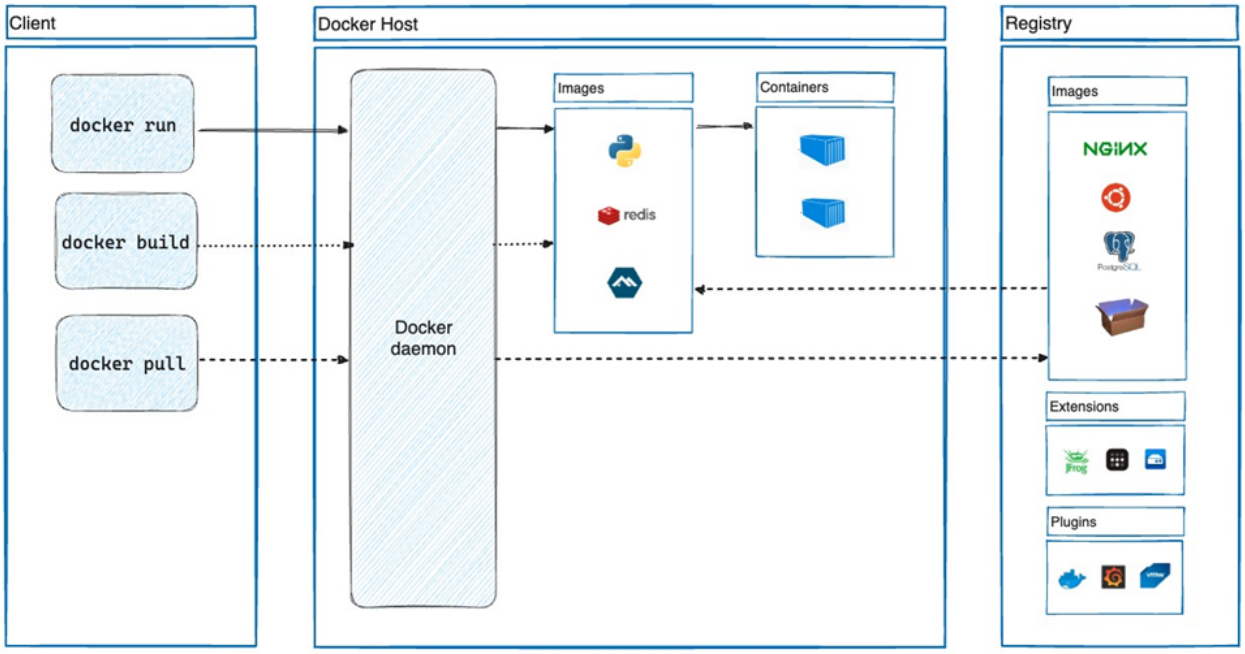
Here are some key components and concepts related to Docker:
Docker Containers: Containers are instances of Docker images. An image is a lightweight, standalone, and executable package that includes everything needed to run an application, including the code, runtime, system tools, libraries, and settings. Containers are isolated from one another and the host system, ensuring consistency and reproducibility of applications across different environments.
Docker Engine: The Docker Engine is the core component of Docker that manages containers. It includes a server, REST API, and command-line interface (CLI) for interacting with Docker. It can run on various operating systems, including Linux and Windows.
Dockerfile: A Dockerfile is a text file that contains a set of instructions for building a Docker image. It specifies the base image, adds application code, sets environment variables, and configures the container. Dockerfiles are used to create customized images for specific applications.
Docker Compose: Docker Compose is a tool for defining and running multi-container
Docker applications. It allows you to define the services, networks, and volumes in a single YAML file, making it easier to manage complex applications with multiple interconnected containers.
Docker Hub: Docker Hub is a cloud-based registry service provided by Docker, Inc. It allows users to share and discover Docker images. You can find a wide range of pre-built Docker images on Docker Hub, which can be used as a base for your own containers.
Orchestration: Docker Swarm and Kubernetes are popular tools for orchestrating containers in production environments. They provide features like automatic scaling, load balancing, service discovery, and high availability for containerized applications.

Portability: Docker containers are highly portable, meaning you can run the same containerized application on different platforms and cloud providers without modification. This portability simplifies development, testing, and deployment workflows.

DOCKER ARCHITECTURE

Docker uses a client-server architecture that consists of several key components. Understanding the Docker architecture is essential for grasping how containers work and interact with each other and the host system. Here's an overview of the main components:

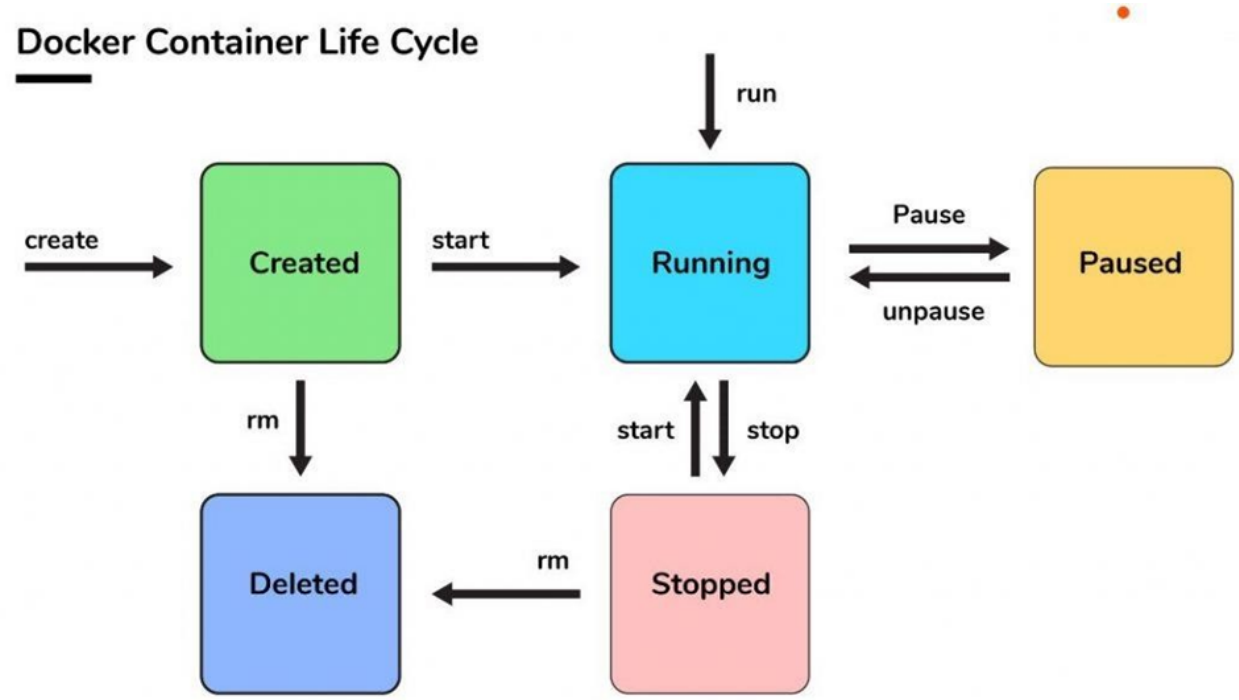
1. **Docker Daemon (dockerd):** The Docker daemon is a background service that runs on the host system. It is responsible for managing Docker containers, images, networks, and volumes. Docker client tools communicate with the daemon to execute commands and manage containers.
2. **Docker Client:** The Docker client is a command-line tool or graphical user interface (GUI) that allows users to interact with the Docker daemon. Users issue commands to the client, which then sends requests to the daemon for container management.
3. **Docker Registry:** Docker images are stored in registries, which are repositories for container images. Docker Hub is a popular public registry, but you can also set up private registries for your organization's custom images. When you run a Docker container, it typically pulls the required image from a registry if it's not already available on the host system.
4. **Docker Images:** Docker images are read-only templates that contain an application's code, runtime, libraries, and dependencies. Images are used as a basis for creating containers. Images can be built from Dockerfiles, which are plain text files containing instructions for building an image layer by layer.
5. **Docker Containers:** Containers are instances of Docker images that run in isolated environments on the host system. Each container has its own filesystem, processes, networking, and resource constraints. Containers are lightweight, fast to start, and can be easily moved between different Docker hosts.



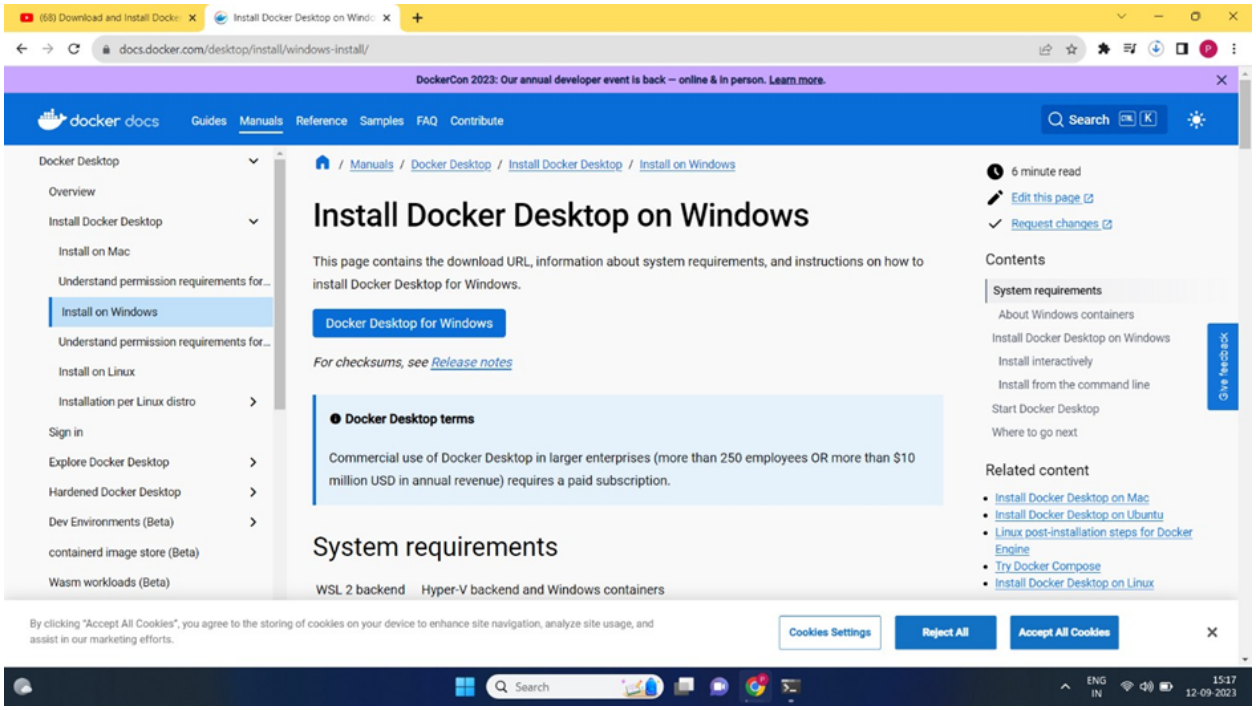
LIFECYCLE OF A CONTAINER

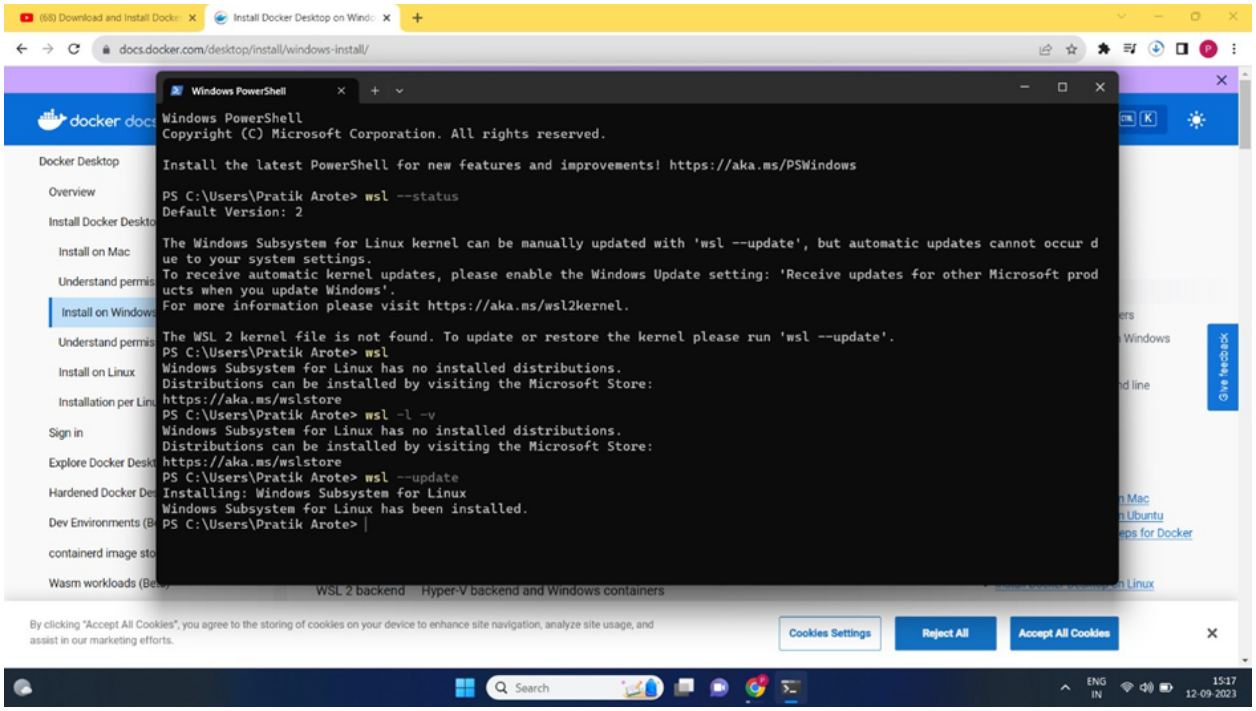
Creating a Container: To create a Docker container, you typically start with a Docker image. You can use a pre-built image from a registry or build your own using a Dockerfile. The Docker image is essentially a snapshot of a container's filesystem and configuration.
Running a Container: To run a container, you use the docker run command, specifying the image and any additional options or configurations.
When you run a container, Docker: Allocates resources (CPU, memory, etc.) as per your specifications. Creates a unique instance of the container from the image. Starts the container's processes.
Container Execution: The container executes the processes defined in its image. These processes can be your application, services, or any command specified in the Dockerfile or provided when starting the container.
Interacting with a Container: You can interact with a running container by attaching to its console using the docker exec or docker attach command. This allows you to run commands inside the container or access its logs. Containers can also expose network ports, allowing communication with other containers or external systems.
Stopping and Restarting a Container: You can stop a running container using the docker stop command. This sends a termination signal to the container, allowing it to perform cleanup tasks before stopping. You can restart a stopped container using the docker start command.
Container Cleanup: When a container is stopped and removed using the docker rm command, it is deleted along with its filesystem and any changes made during its execution. However, the Docker image remains intact and can be used to create new containers.

Monitoring and Logs: Docker provides tools for monitoring container resource usage and collecting container logs, allowing you to troubleshoot and manage running containers effectively.
Updating and Versioning: If your application code or dependencies change, you can update the Docker image and recreate containers with the latest changes. This ensures consistency in different environments.



INSTALLATION:

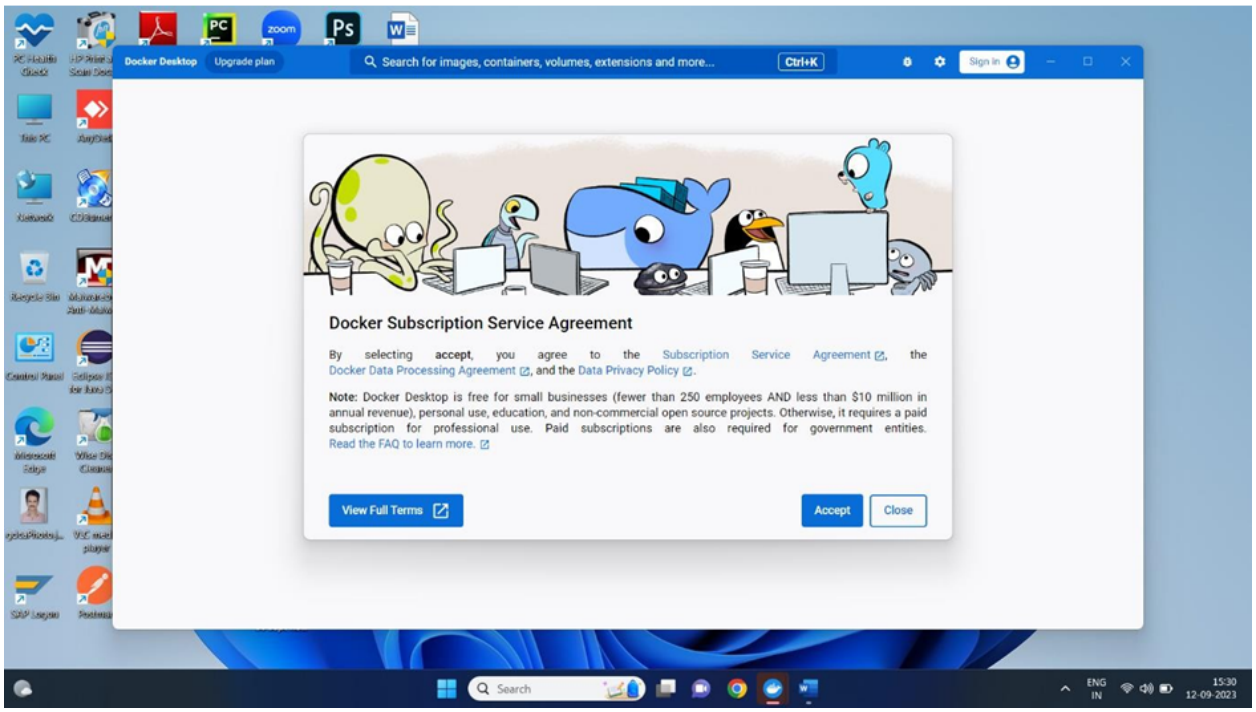

(68) Download and Install Docker
Install Docker Desktop on Windo
docs.docker.com/desktop/install/windows-install/
DockerCon 2023: Our annual developer event is back — online & in person. Learn more.
docker docs
Guides
Manuals
Reference
Samples
FAQ
Contribute
Search
Docker Desktop
Overview
Install Docker Desktop
Install on Mac
Understand permission requirements for...
Install on Windows
Understand permission requirements for...
Install on Linux
Installation per Linux distro
Sign in
Explore Docker Desktop
Hardened Docker Desktop
Dev Environments (Beta)
containerd image store (Beta)
Wasm workloads (Beta)
Manuals / Docker Desktop / Install Docker Desktop / Install on Windows
Install Docker Desktop on Windows
This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.
Docker Desktop for Windows
For checksums, see Release notes
Docker Desktop terms
Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.
System requirements
WSL 2 backend
Hyper-V backend and Windows containers
6 minute read
Edit this page
Request changes
Contents
System requirements
About Windows containers
Install Docker Desktop on Windows
Install interactively
Install from the command line
Start Docker Desktop
Where to go next
Related content
Install Docker Desktop on Mac
Install Docker Desktop on Ubuntu
Linux post-installation steps for Docker Engine
Try Docker Compose
Install Docker Desktop on Linux
Give feedback
By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.
Cookies Settings
Reject All
Accept All Cookies
Search
ENG
IN
15:17
12-09-2023

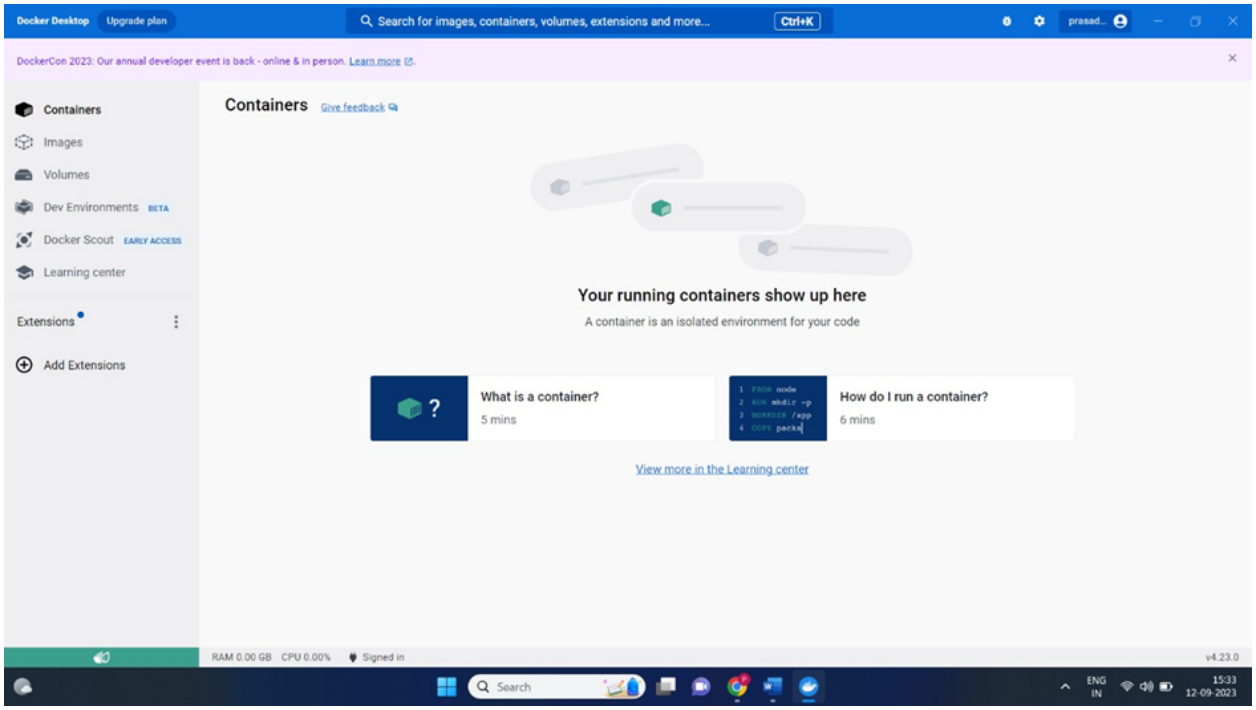

Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Pratik Arote> wsl --status
Default Version: 2
The Windows Subsystem for Linux kernel can be manually updated with 'wsl --update', but automatic updates cannot occur due to your system settings.
To receive automatic kernel updates, please enable the Windows Update setting: 'Receive updates for other Microsoft products when you update Windows'.
For more information please visit https://aka.ms/wsl2kernel.
The WSL 2 kernel file is not found. To update or restore the kernel please run 'wsl --update'.
PS C:\Users\Pratik Arote> wsl
Windows Subsystem for Linux has no installed distributions.
Distributions can be installed by visiting the Microsoft Store:
https://aka.ms/wslstore
PS C:\Users\Pratik Arote> wsl -l -v
Windows Subsystem for Linux has no installed distributions.
Distributions can be installed by visiting the Microsoft Store:
https://aka.ms/wslstore
PS C:\Users\Pratik Arote> wsl --update
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
PS C:\Users\Pratik Arote>


Open File - Security Warning
Do you want to run this file?
Name: ...tik Arote\Downloads\Docker Desktop Installer.exe
Publisher: Docker Inc
Type: Application
From: C:\Users\Pratik Arote\Downloads\Docker Desktop ...
Run
Cancel
Always ask before opening this file
While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?
31°C
Partly sunny
Search
ENG
IN
15:19
12-09-2023

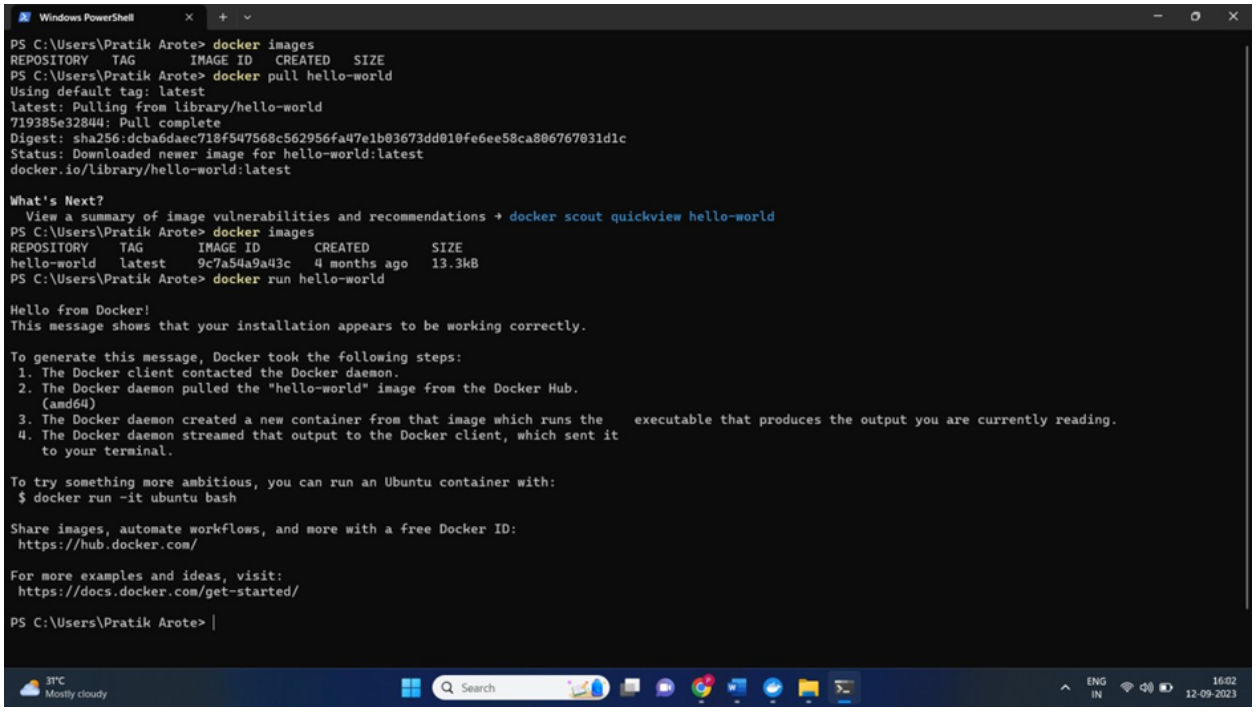

Installing Docker Desktop 4.23.0 (120376)
Docker Desktop 4.23.0
Unpacking files...
Unpacking file: resources/wsl/docker-for-wsl.iso
Unpacking file: resources/windows-daemon-options.json
Unpacking file: resources/tile-icon.png
Unpacking file: resources/tile-error.png
Unpacking file: resources/services.iso.sha256
Unpacking file: resources/services.iso
Unpacking file: resources/linux-daemon-options.json
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
31°C
Partly sunny
Search
ENG
IN
15:21
12-09-2023

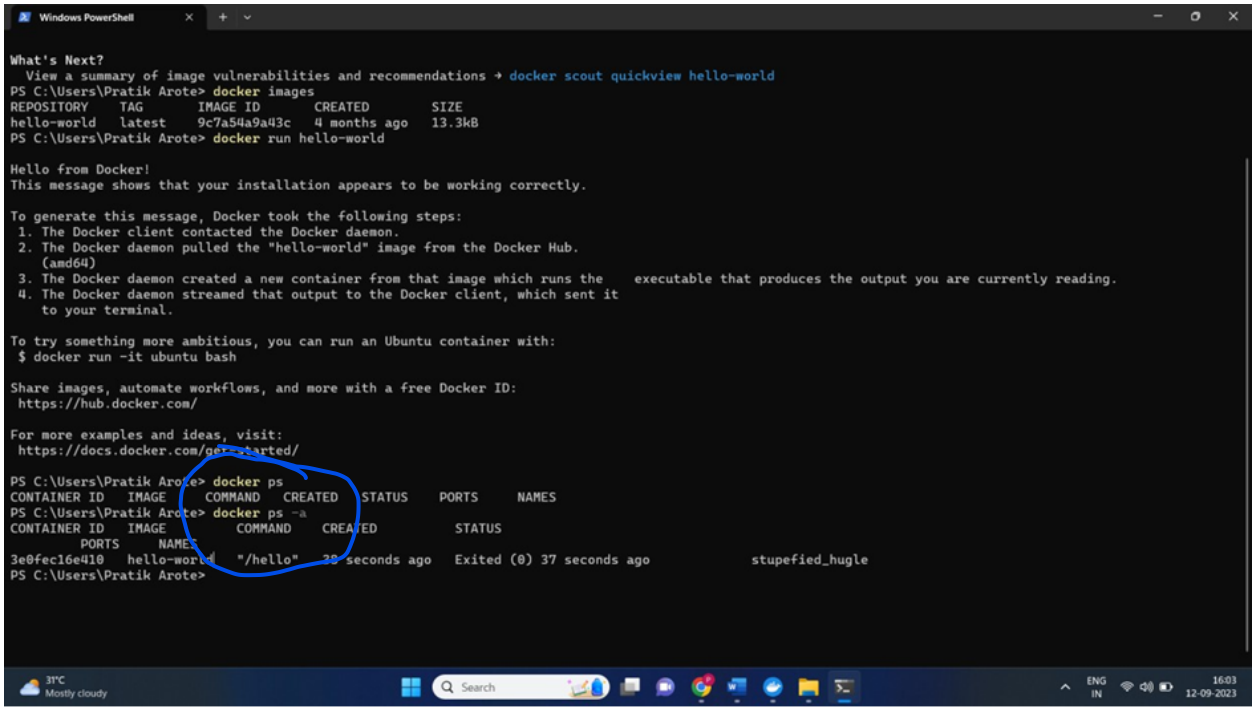

Installing Docker Desktop 4.23.0 (120376)
Docker Desktop 4.23.0
Installation succeeded
You must log out of Windows to complete installation.
Close and log out
31°C
Partly sunny
Search
ENG
IN
15:28
12-09-2023

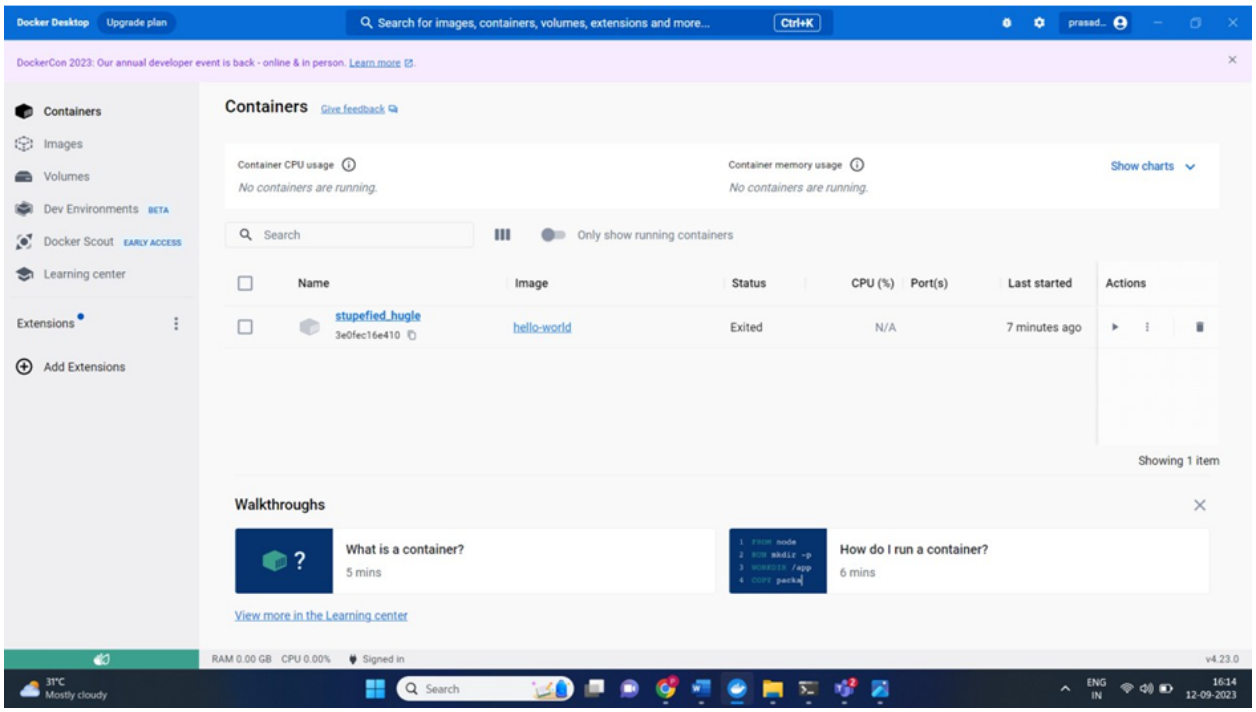

Docker Desktop
Upgrade plan
Search for images, containers, volumes, extensions and more...
Ctrl+K
Sign in
Docker Subscription Service Agreement
By selecting accept, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, and the Data Privacy Policy.
Note: Docker Desktop is free for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects. Otherwise, it requires a paid subscription for professional use. Paid subscriptions are also required for government entities. Read the FAQ to learn more.
View Full Terms
Accept
Close
Search
ENG
IN
15:30
12-09-2023



DOCKER CONTAINER







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.

Assignment 8

Aim :- To build an image for sample web application using Dockerfile.

LO Mapped :- LO1 , LO5

Theory and Steps:-

Building a Docker image for a sample web application using a Dockerfile: Building a Docker Image for a Sample Web Application: A Dockerfile is a script used to create a custom Docker image. In the context of a sample web application, you would typically follow these steps:

1. Choose a Base Image: Start by selecting a base image that provides the foundational operating system and environment for your web application. Common choices include official Linux distributions like Ubuntu or Alpine.

2. Set the Working Directory: Use the `WORKDIR` command in your Dockerfile to specify the directory where your application's files will be placed.

3. Copy Application Files: Use the `COPY` command to copy the necessary files for your web application into the Docker image. This includes HTML, CSS, JavaScript, and any other assets.

4. Install Dependencies: If your web application has any dependencies, such as a web server or runtime environment, use appropriate package managers (e.g., `apt-get`, `apk`, `npm`, or `pip`) to install them in the Docker image.

5. Expose Ports: If your web application listens on a specific port, use the `EXPOSE` command to specify which port should be exposed for incoming connections.

6. Define Startup Command: Use the `CMD` or `ENTRYPOINT` command to define the command that will be executed when the Docker container is run. This should start your web application or web server.

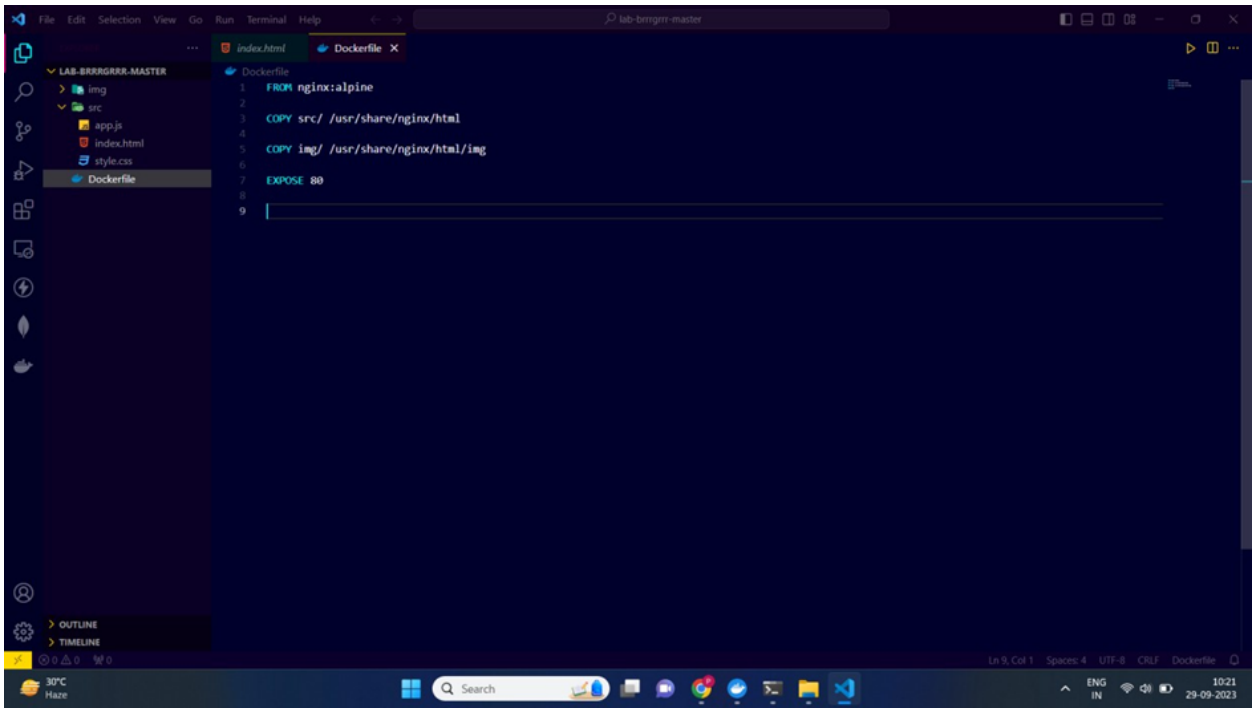
7. Build the Image: Use the `docker build` command to build the Docker image based on your Dockerfile. This command should be run in the same directory as your Dockerfile.

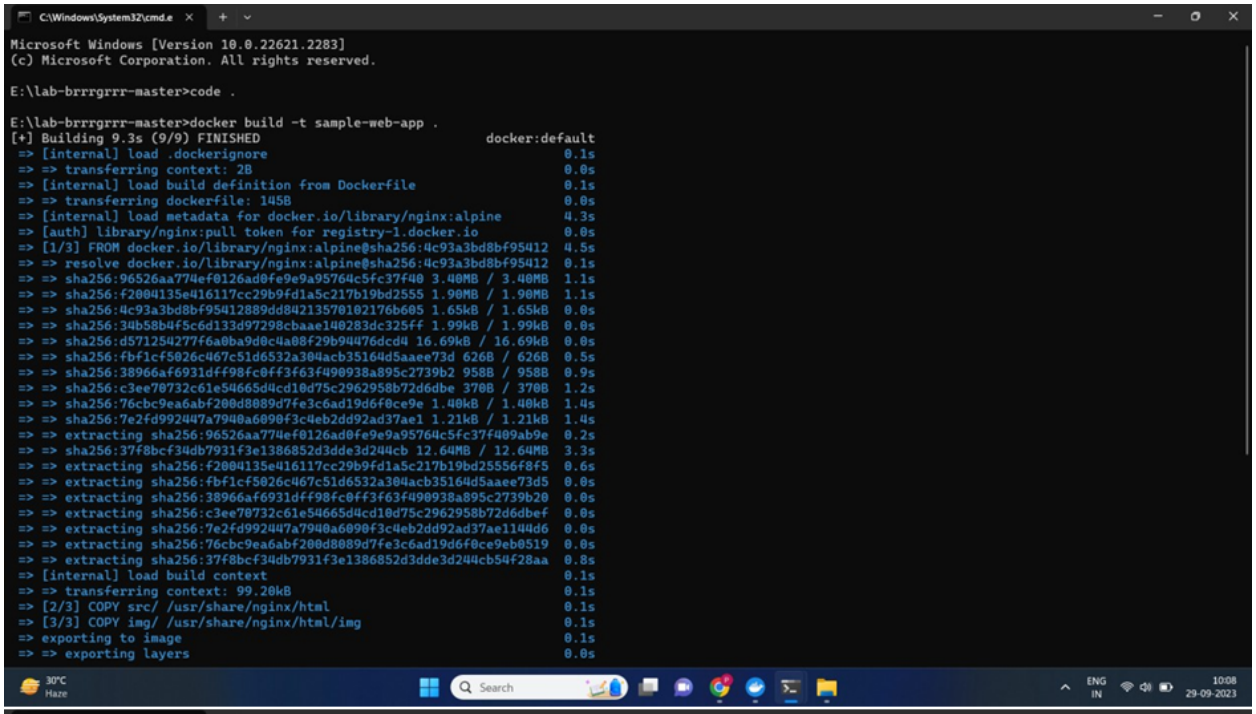
8. Run a Container: Once the image is built, you can create and run a Docker container from the image using the `docker run` command. You can specify options like port mapping and environment variables during container creation.

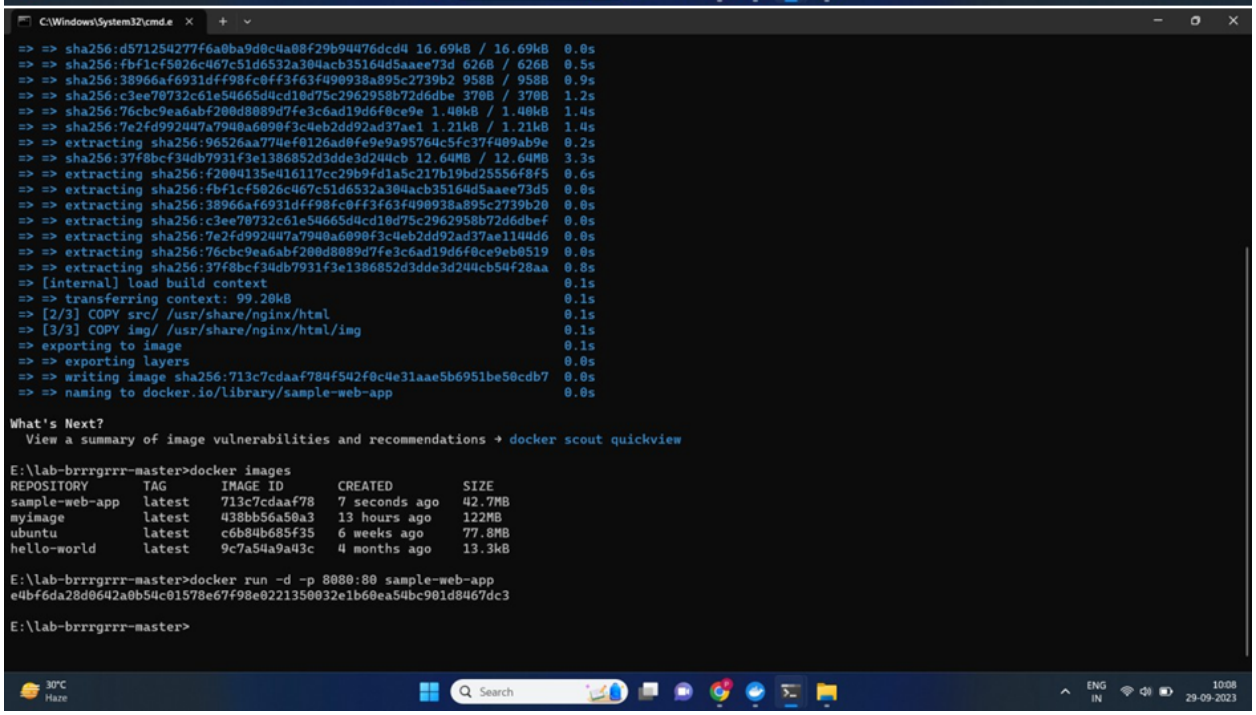
9. Access the Web Application: With the container running, you can access your web application by opening a web browser and navigating to the exposed port on your host system.

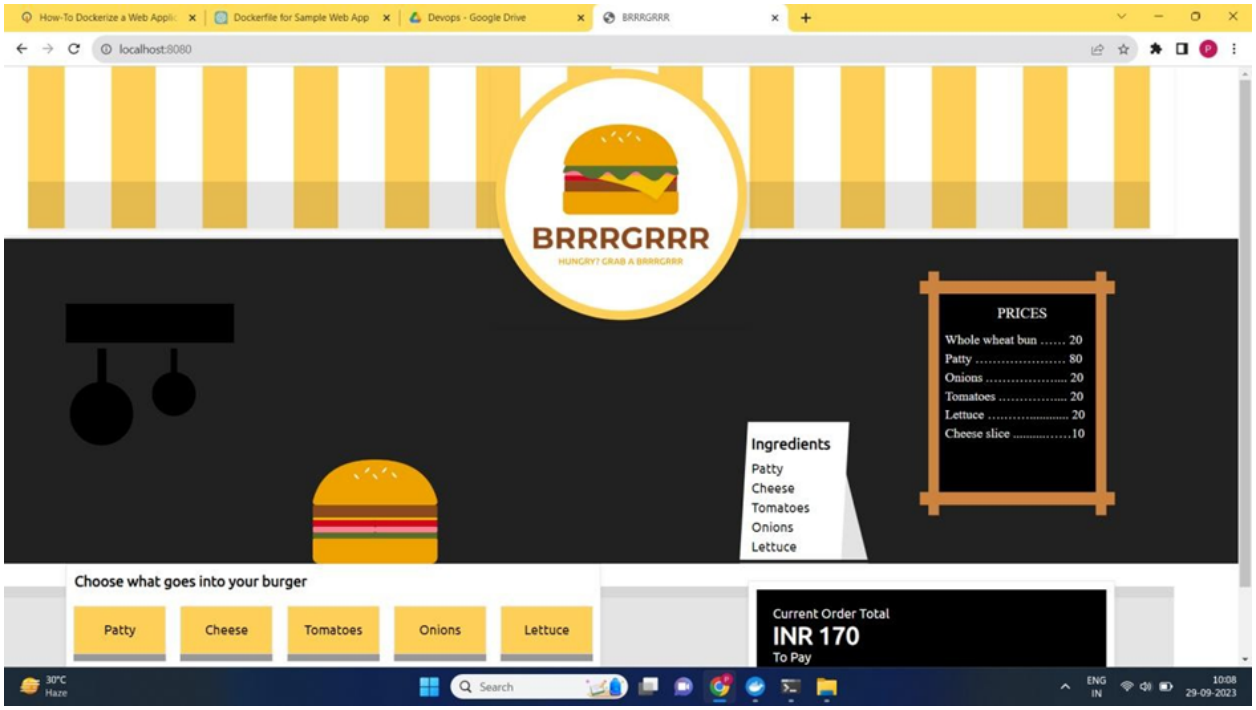
By following these steps and customizing your Dockerfile, you can create a Docker image for your sample web application. This image can be easily shared and deployed in various environments, ensuring consistency and portability for your application

Output screenshots :-




C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.
E:\lab-brrrgrrr-master>code .
E:\lab-brrrgrrr-master>docker build -t sample-web-app .
[+] Building 9.3s (9/9) FINISHED docker:default
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 145B 0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine 4.3s
=> [auth] library/nginx:pull token for registry-1.docker.io 0.0s
=> [1/3] FROM docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412 4.5s
=> => resolve docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412 0.1s
=> => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f40 3.40MB / 3.40MB 1.1s
=> => sha256:f2004135e416117cc29b9fd1a5c217b19bd2555 1.90MB / 1.90MB 1.1s
=> => sha256:4c93a3bd8bf95412889dd84213570102176b605 1.65kB / 1.65kB 0.0s
=> => sha256:34b58b4f5c6d133d97298cbaae140283dc325ff 1.99kB / 1.99kB 0.0s
=> => sha256:d571254277f6a0ba9d0c4a08f29b94476dcd4 16.69kB / 16.69kB 0.0s
=> => sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d 626B / 626B 0.5s
=> => sha256:38966af6931dff98fc0ff3f63f490938a895c2739b2 958B / 958B 0.9s
=> => sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbe 370B / 370B 1.2s
=> => sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9e 1.40kB / 1.40kB 1.4s
=> => sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1 1.21kB / 1.21kB 1.4s
=> => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e 0.2s
=> => sha256:37f8bcf34db7931f3e1386852d3dde3d244cb 12.64MB / 12.64MB 3.3s
=> => extracting sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f5 0.6s
=> => extracting sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d5 0.0s
=> => extracting sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20 0.0s
=> => extracting sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbef 0.0s
=> => extracting sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1144d6 0.0s
=> => extracting sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb0519 0.0s
=> => extracting sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aa 0.8s
=> [internal] load build context 0.1s
=> => transferring context: 99.20kB 0.1s
=> [2/3] COPY src/ /usr/share/nginx/html 0.1s
=> [3/3] COPY img/ /usr/share/nginx/html/img 0.1s
=> exporting to image 0.1s
=> => exporting layers 0.0s
30°C Haze
Search
ENG IN
10:08 29-09-2023


C:\Windows\System32\cmd.e
=> => sha256:d571254277f6a0ba9d0c4a08f29b94476dcd4 16.69kB / 16.69kB 0.0s
=> => sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d 626B / 626B 0.5s
=> => sha256:38966af6931dff98fc0ff3f63f490938a895c2739b2 958B / 958B 0.9s
=> => sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbe 370B / 370B 1.2s
=> => sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9e 1.40kB / 1.40kB 1.4s
=> => sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1 1.21kB / 1.21kB 1.4s
=> => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e 0.2s
=> => sha256:37f8bcf34db7931f3e1386852d3dde3d244cb 12.64MB / 12.64MB 3.3s
=> => extracting sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f5 0.6s
=> => extracting sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d5 0.0s
=> => extracting sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20 0.0s
=> => extracting sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbef 0.0s
=> => extracting sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1144d6 0.0s
=> => extracting sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb0519 0.0s
=> => extracting sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aa 0.8s
=> [internal] load build context 0.1s
=> => transferring context: 99.20kB 0.1s
=> [2/3] COPY src/ /usr/share/nginx/html 0.1s
=> [3/3] COPY img/ /usr/share/nginx/html/img 0.1s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:713c7cdaaf784f542f0c4e31aae5b6951be50cdb7 0.0s
=> => naming to docker.io/library/sample-web-app 0.0s
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
E:\lab-brrrgrrr-master>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
sample-web-app latest 713c7cdaaf78 7 seconds ago 42.7MB
myimage latest 438bb56a50a3 13 hours ago 122MB
ubuntu latest c6b84b685f35 6 weeks ago 77.8MB
hello-world latest 9c7a54a9a43c 4 months ago 13.3kB
E:\lab-brrrgrrr-master>docker run -d -p 8080:80 sample-web-app
e4bf6da28d0642a0b54c01578e67f98e0221350032e1b60ea54bc901d8467dc3
E:\lab-brrrgrrr-master>
30°C Haze
Search
ENG IN
10:08 29-09-2023



Conclusion :-

Here we understood the steps to build an image of a web application and also we successfully created an image of sample web application.

Assignment 9

Aim :- Installation of Nagios on Ubuntu System

LO Mapped :- LO1 , LO5

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

- 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.

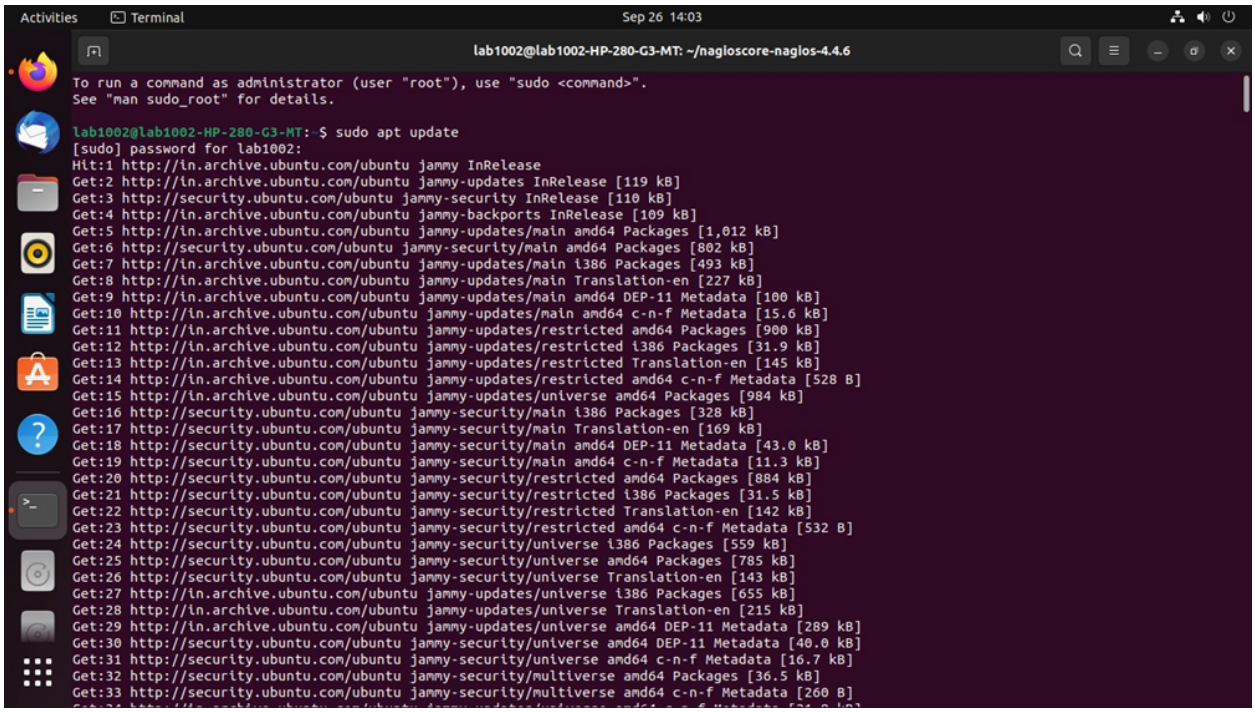
Nagios tools

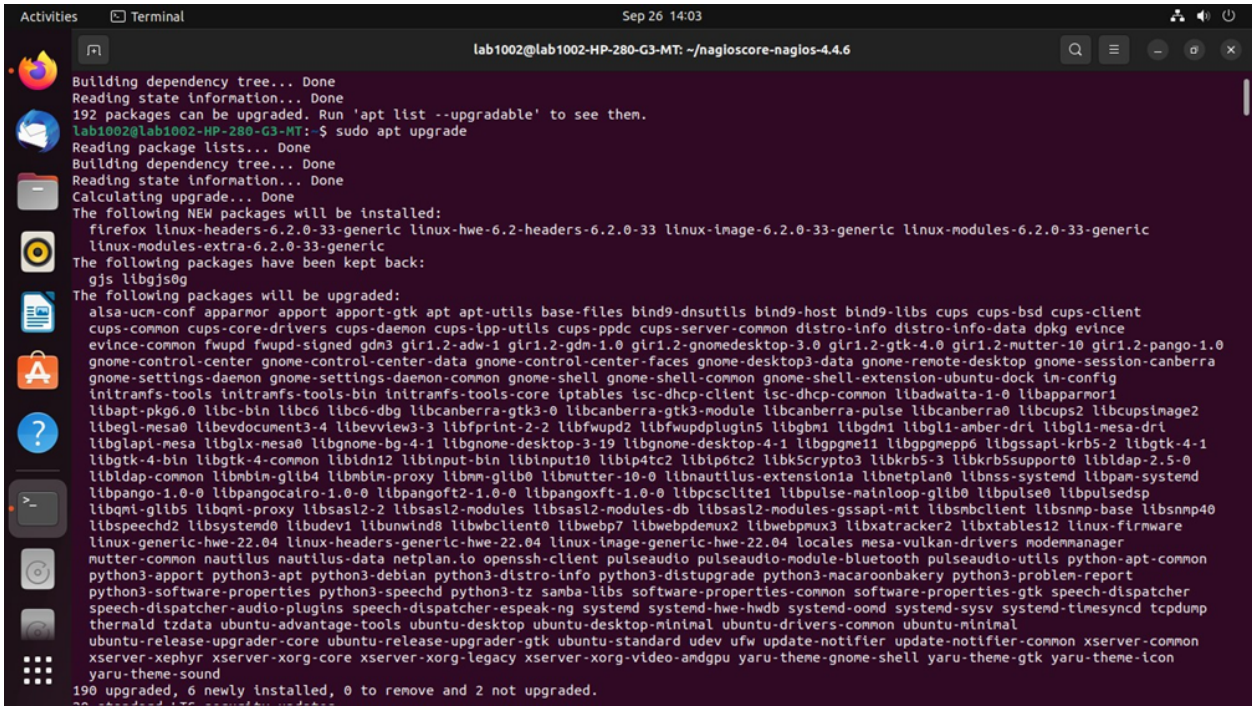
- Nagios has proven popular among small and large businesses, as well as internet service providers, educational institutions, government agencies, healthcare institutions, manufacturing companies and financial institutions.
- Users can choose among free and paid options, depending on the needed services and support.

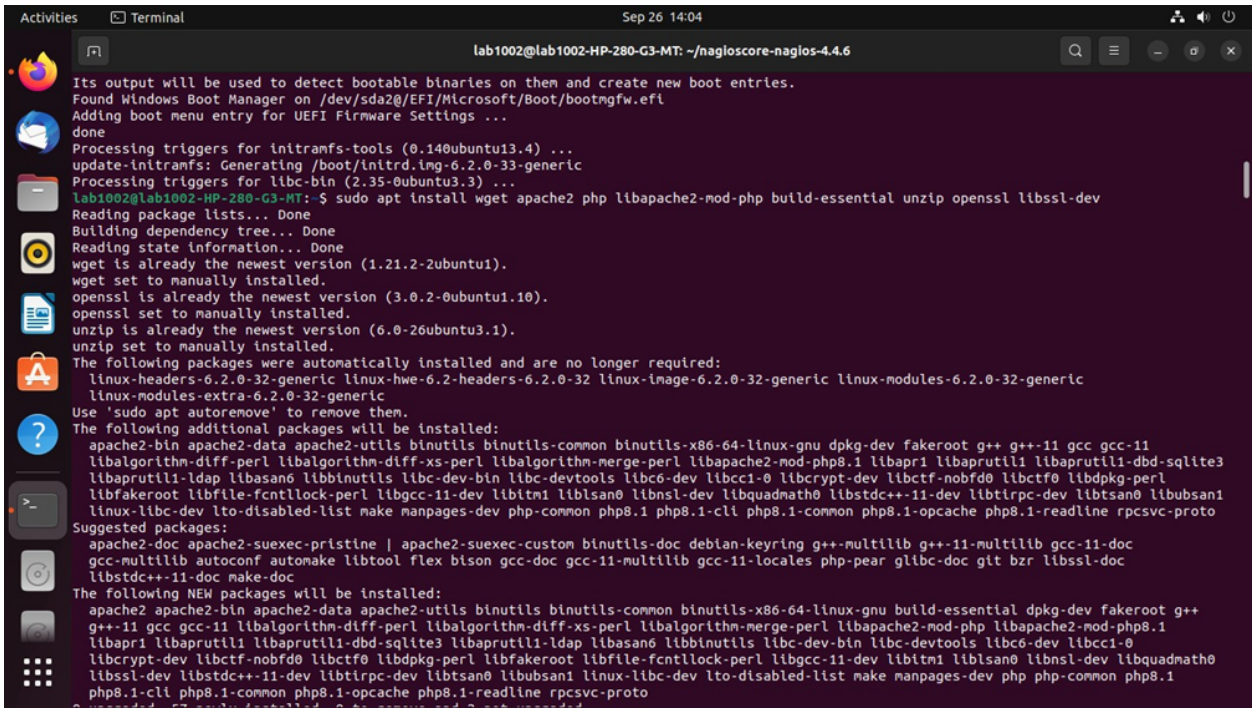
Nagios Core

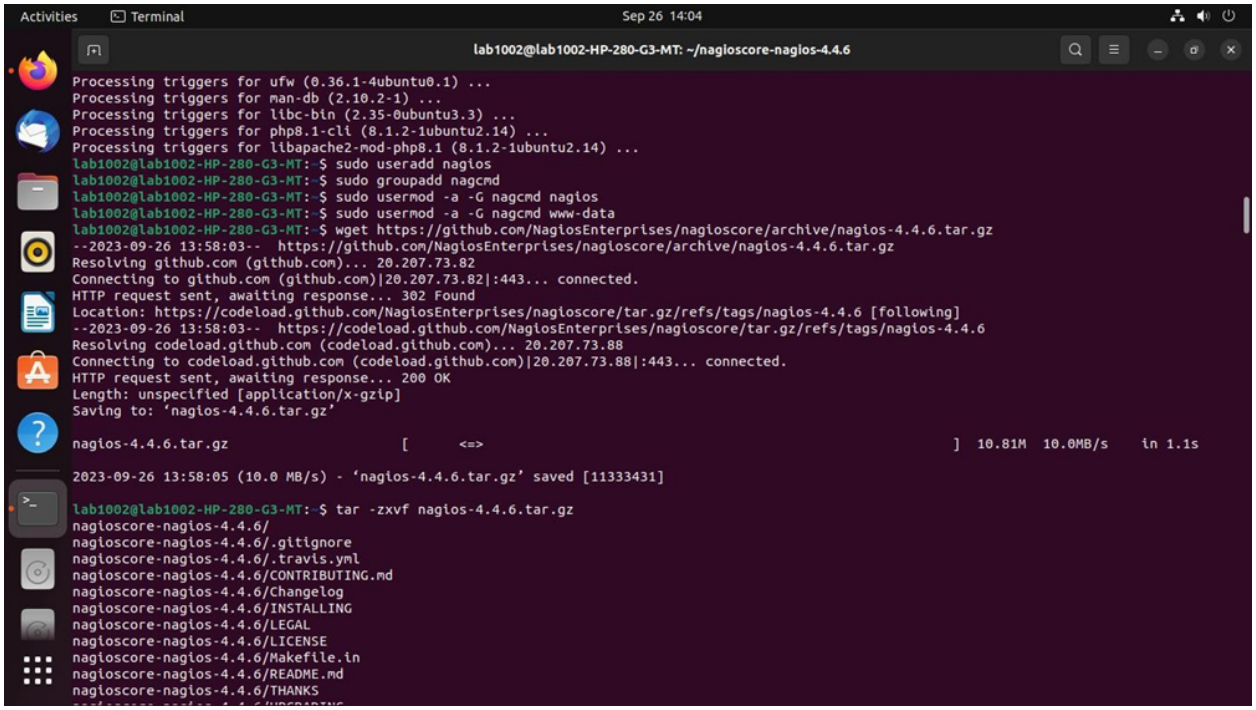
- The service that was originally known as Nagios is now referred to as Nagios Core. Core is freely available as an open source monitoring software for IT systems, networks and infrastructure. Core contains a wide array of infrastructure monitoring through allowing plugins to extend its monitoring capabilities.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

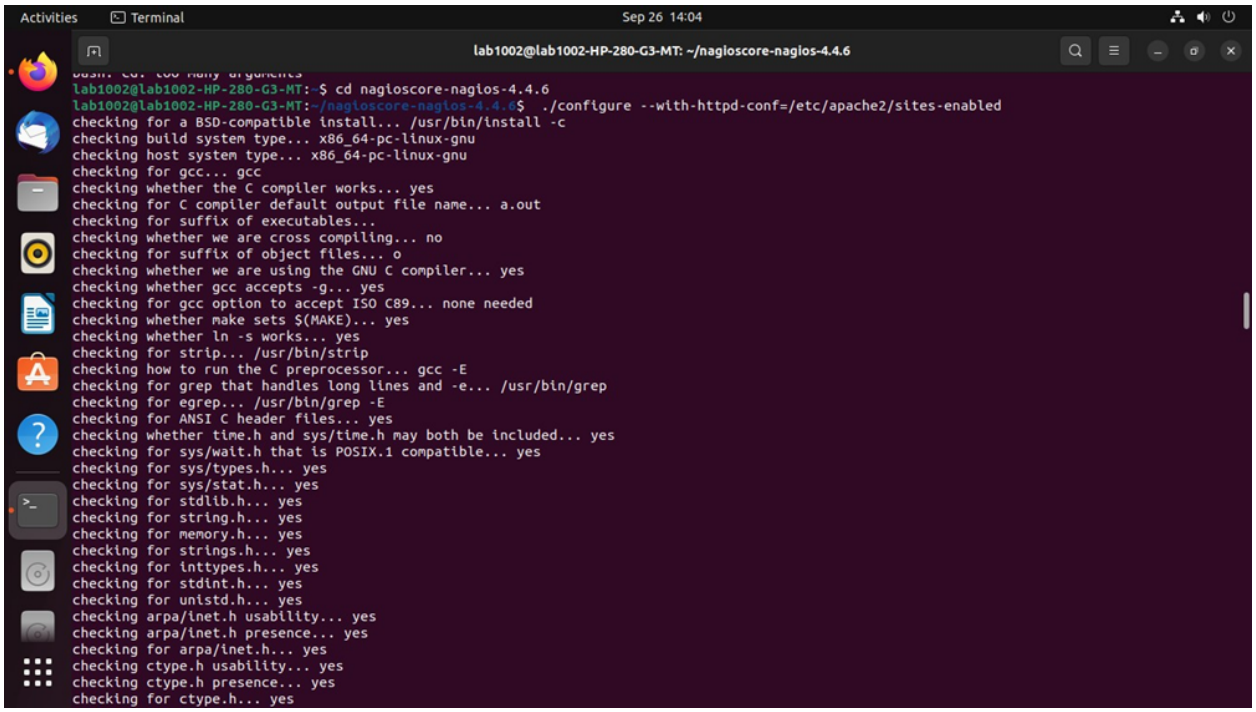
Output screenshots :- (Nagios installation and startup)

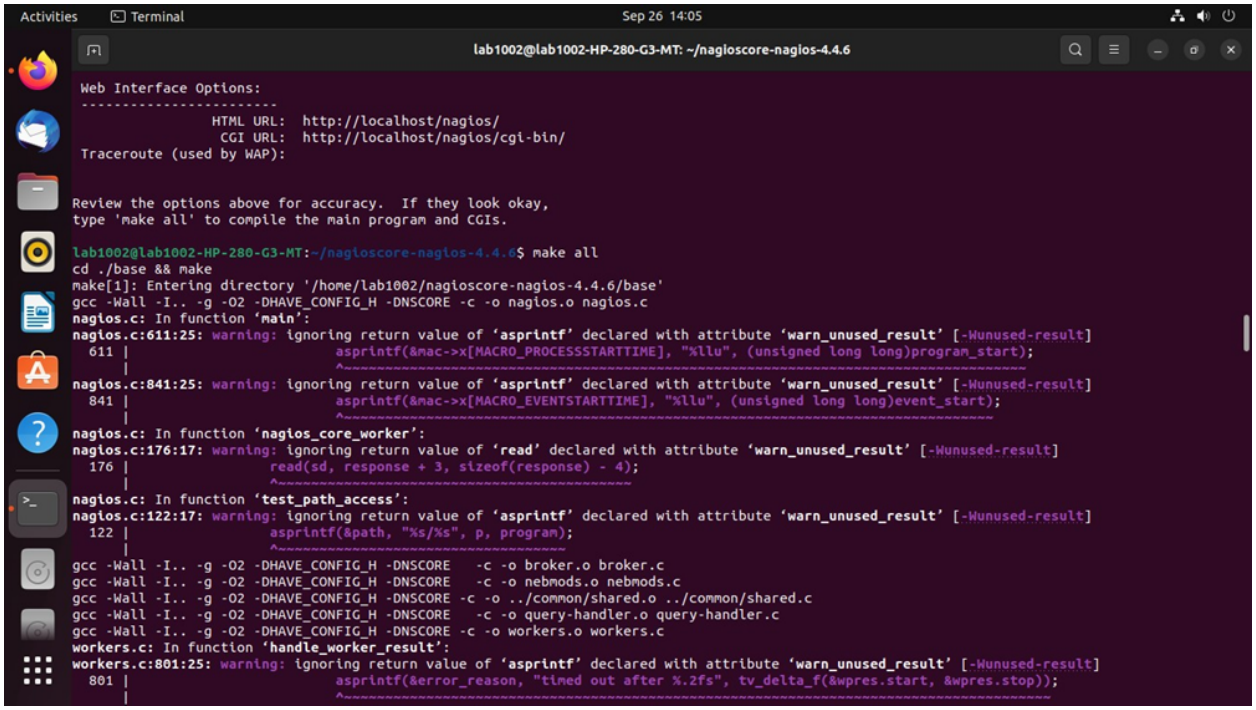

Activities
Terminal
Sep 26 14:03
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
lab1002@lab1002-HP-280-G3-MT:~$ sudo apt update
[sudo] password for lab1002:
Hit:1 http://in.archive.ubuntu.com/ubuntu 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://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,012 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [802 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [493 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://in.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [31.9 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [145 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [528 B]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [984 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [328 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [169 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.0 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [11.3 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [884 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [31.5 kB]
Get:22 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [142 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [532 B]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [559 kB]
Get:25 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [785 kB]
Get:26 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [143 kB]
Get:27 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [655 kB]
Get:28 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [215 kB]
Get:29 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [289 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [40.0 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.7 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [36.5 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]

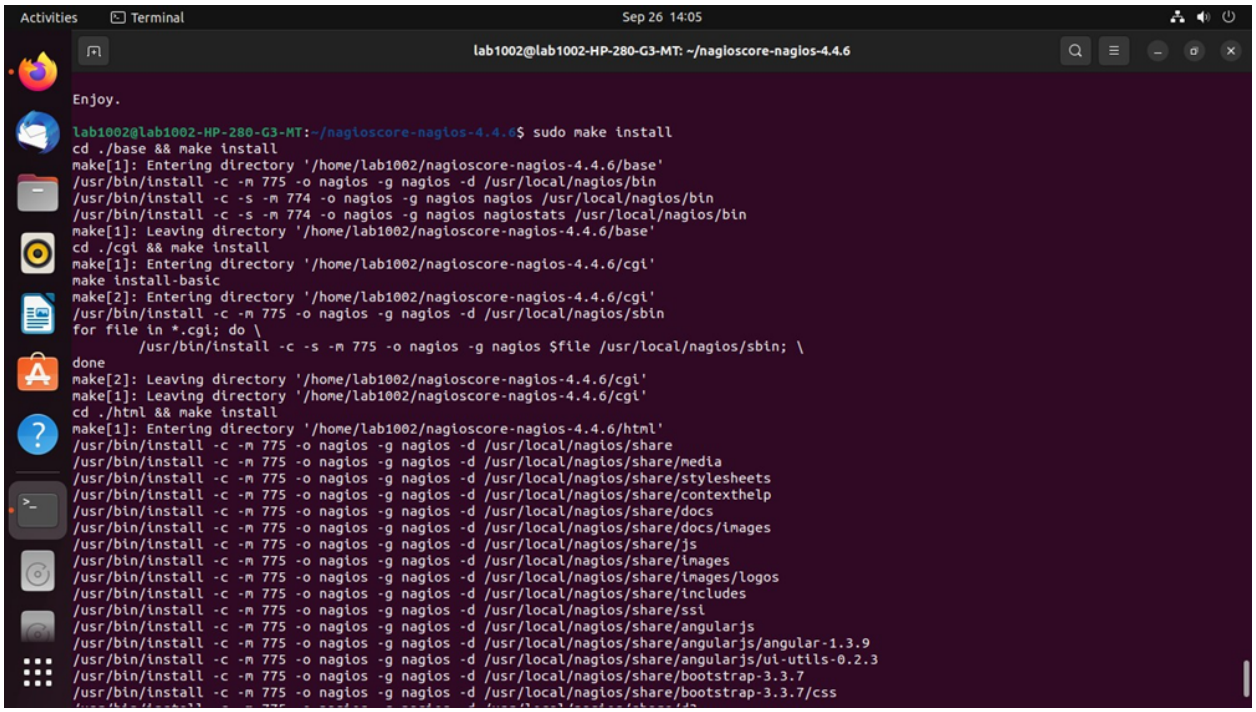

Activities
Terminal
Sep 26 14:03
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
Building dependency tree... Done
Reading state information... Done
192 packages can be upgraded. Run 'apt list --upgradable' to see them.
lab1002@lab1002-HP-280-G3-MT:~$ sudo apt upgrade
Reading package lists... 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:
alsa-ucm-conf apparmor apport apport-gtk 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 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 im-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-gtk3-module libcanberra-pulse libcanberra0 libcups2 libcupsimage2
libegl-mesa0 libevdocument3-4 libevview3-3 libfprint-2-2 libfwupd2 libfwupdplugin5 libgbm1 libgdm1 libgl1-amber-dri libgl1-mesa-dri
libglapi-mesa libglx-mesa0 libgnome-bg-4-1 libgnome-desktop-3-19 libgnome-desktop-4-1 libgpgme11 libgpgmepp6 libgssapi-krb5-2 libgtk-4-1
libgtk-4-bin libgtk-4-common libidn12 libinput-bin libinput10 libip4tc2 libip6tc2 libk5crypto3 libkrb5-3 libkrb5support0 libldap-2.5-0
libldap-common libmbim-glib4 libmbim-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 libpangoxft-1.0-0 libpcsclite1 libpulse-mainloop-glib0 libpulse0 libpulsedsp
libqmi-glib5 libqmi-proxy libsasl2-2 libsasl2-modules libsasl2-modules-db libsasl2-modules-gssapi-mit libsmbclient libsnmp-base libsnmp40
libspeechd2 libsystemd0 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 modemmanager
mutter-common nautilus nautilus-data netplan.io openssh-client pulseaudio pulseaudio-module-bluetooth pulseaudio-utils python-apt-common
python3-apport python3-apt python3-debian python3-distro-info python3-distupgrade 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 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 ufw update-notifier update-notifier-common xserver-common
xserver-xephyr xserver-xorg-core xserver-xorg-legacy xserver-xorg-video-amdgpu yaru-theme-gnome-shell yaru-theme-gtk yaru-theme-icon
yaru-theme-sound
190 upgraded, 6 newly installed, 0 to remove and 2 not upgraded.

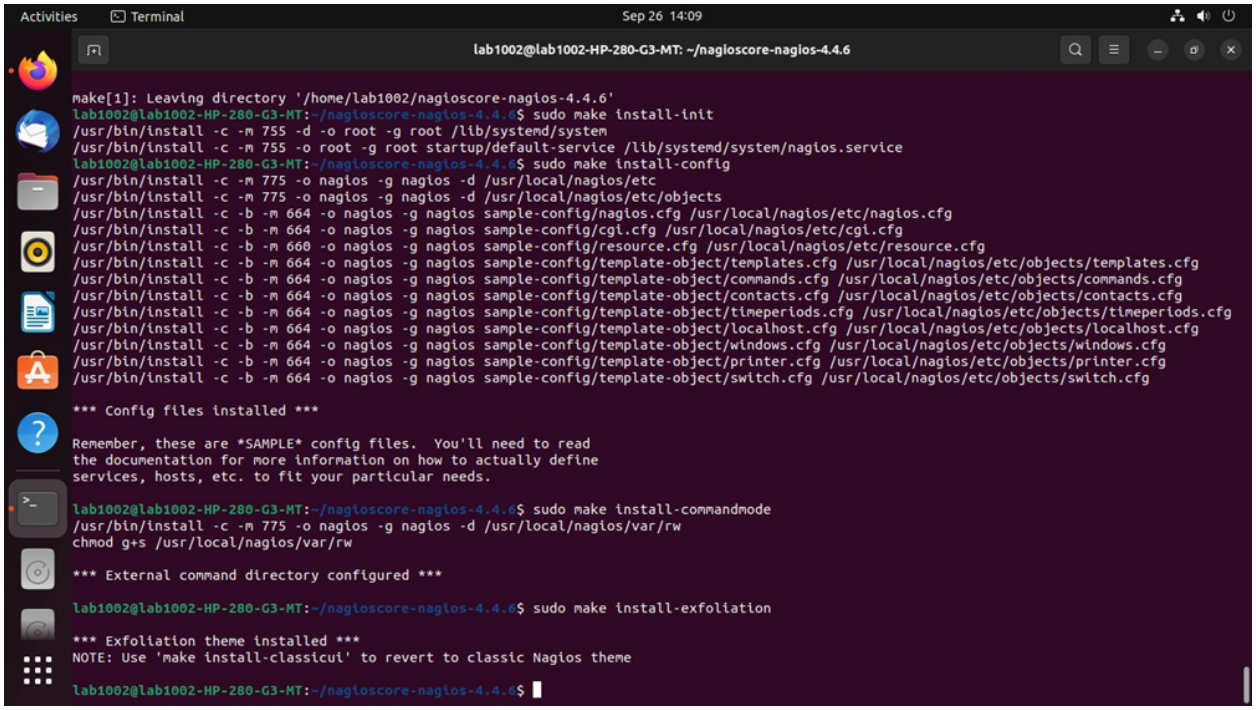

Activities
Terminal
Sep 26 14:04
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/sda2@/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
Processing triggers for initramfs-tools (0.140ubuntu13.4) ...
update-initramfs: Generating /boot/initrd.img-6.2.0-33-generic
Processing triggers for libc-bin (2.35-0ubuntu3.3) ...
lab1002@lab1002-HP-280-G3-MT:~$ sudo apt install wget apache2 php libapache2-mod-php build-essential unzip openssl libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wget is already the newest version (1.21.2-2ubuntu1).
wget set to manually installed.
openssl is already the newest version (3.0.2-0ubuntu1.10).
openssl set to manually installed.
unzip is already the newest version (6.0-26ubuntu3.1).
unzip set to manually installed.
The following packages were automatically installed and are no longer required:
linux-headers-6.2.0-32-generic linux-hwe-6.2-headers-6.2.0-32 linux-image-6.2.0-32-generic linux-modules-6.2.0-32-generic
linux-modules-extra-6.2.0-32-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
apache2-bin apache2-data apache2-utils binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-11 gcc gcc-11
libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libapache2-mod-php8.1 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
libaprutil1-ldap libasan6 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl
libfakeroot libfile-fcntllock-perl libgcc-11-dev libitm1 liblsan0 libnsl-dev libquadmath0 libstdc++-11-dev libtirpc-dev libtsan0 libubsan1
linux-libc-dev lto-disabled-list make manpages-dev php-common php8.1 php8.1-cli php8.1-common php8.1-opcache php8.1-readline rpcsvc-proto
Suggested packages:
apache2-doc apache2-suexec-pristine | apache2-suexec-custom binutils-doc debian-keyring g++-multilib g++-11-multilib gcc-11-doc
gcc-multilib autoconf automake libtool flex bison gcc-doc gcc-11-multilib gcc-11-locales php-pear glibc-doc git bzr libssl-doc
libstdc++-11-doc make-doc
The following NEW packages will be installed:
apache2 apache2-bin apache2-data apache2-utils binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev fakeroot g++
g++-11 gcc gcc-11 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libapache2-mod-php libapache2-mod-php8.1
libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libasan6 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0
libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-11-dev libitm1 liblsan0 libnsl-dev libquadmath0
libssl-dev libstdc++-11-dev libtirpc-dev libtsan0 libubsan1 linux-libc-dev lto-disabled-list make manpages-dev php php-common php8.1
php8.1-cli php8.1-common php8.1-opcache php8.1-readline rpcsvc-proto

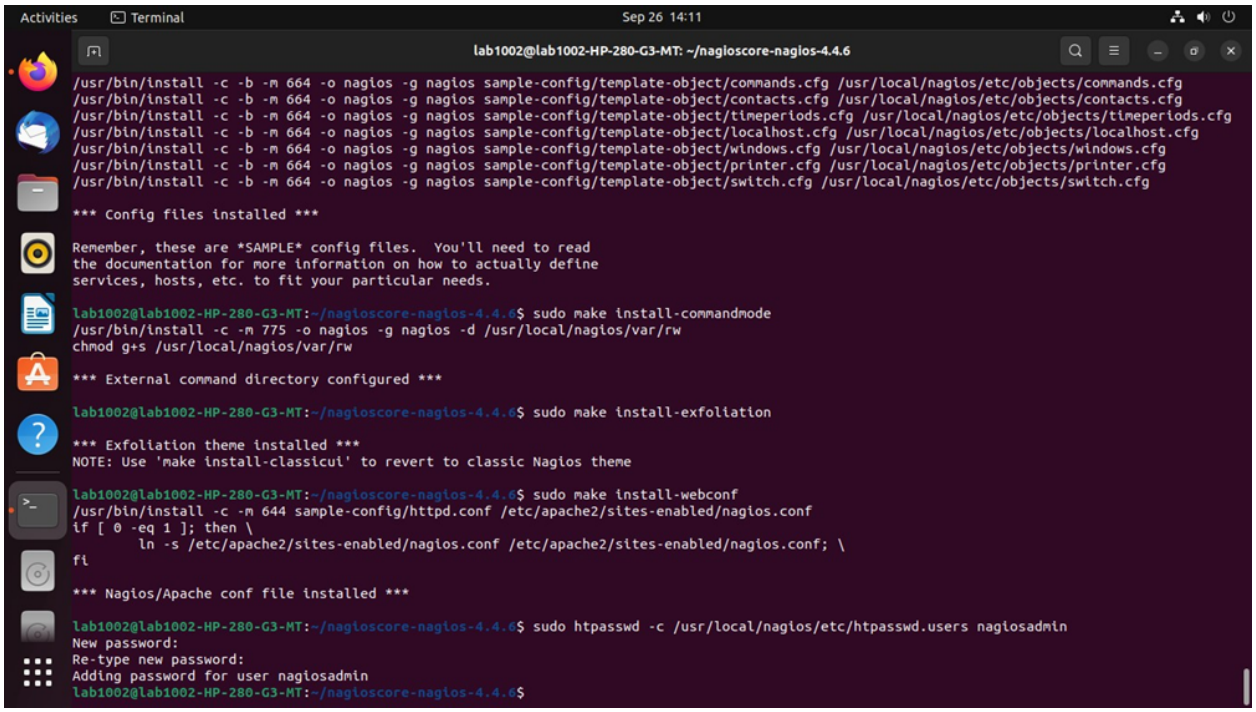

Activities
Terminal
Sep 26 14:04
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.3) ...
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.14) ...
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 13:58:03-- 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://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6 [following]
--2023-09-26 13:58:03-- https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6
Resolving codeload.github.com (codeload.github.com)... 20.207.73.88
Connecting to codeload.github.com (codeload.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 10.0MB/s in 1.1s
2023-09-26 13:58:05 (10.0 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333431]
lab1002@lab1002-HP-280-G3-MT:~$ tar -zxvf 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/LEGAL
nagioscore-nagios-4.4.6/LICENSE
nagioscore-nagios-4.4.6/Makefile.in
nagioscore-nagios-4.4.6/README.md
nagioscore-nagios-4.4.6/THANKS

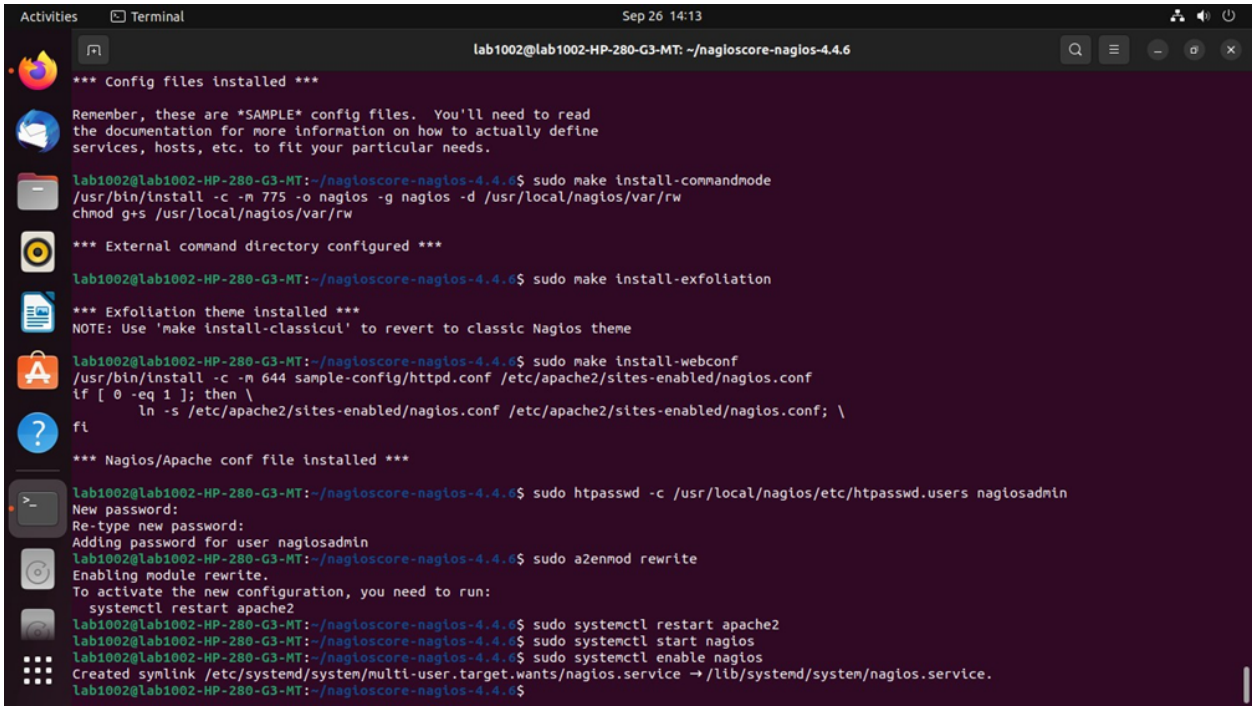

Activities
Terminal
Sep 26 14:04
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
lab1002@lab1002-HP-280-G3-MT:~$ cd nagioscore-nagios-4.4.6
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ ./configure --with-httpd-conf=/etc/apache2/sites-enabled
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking arpa/inet.h usability... yes
checking arpa/inet.h presence... yes
checking for arpa/inet.h... yes
checking ctype.h usability... yes
checking ctype.h presence... yes
checking for ctype.h... yes

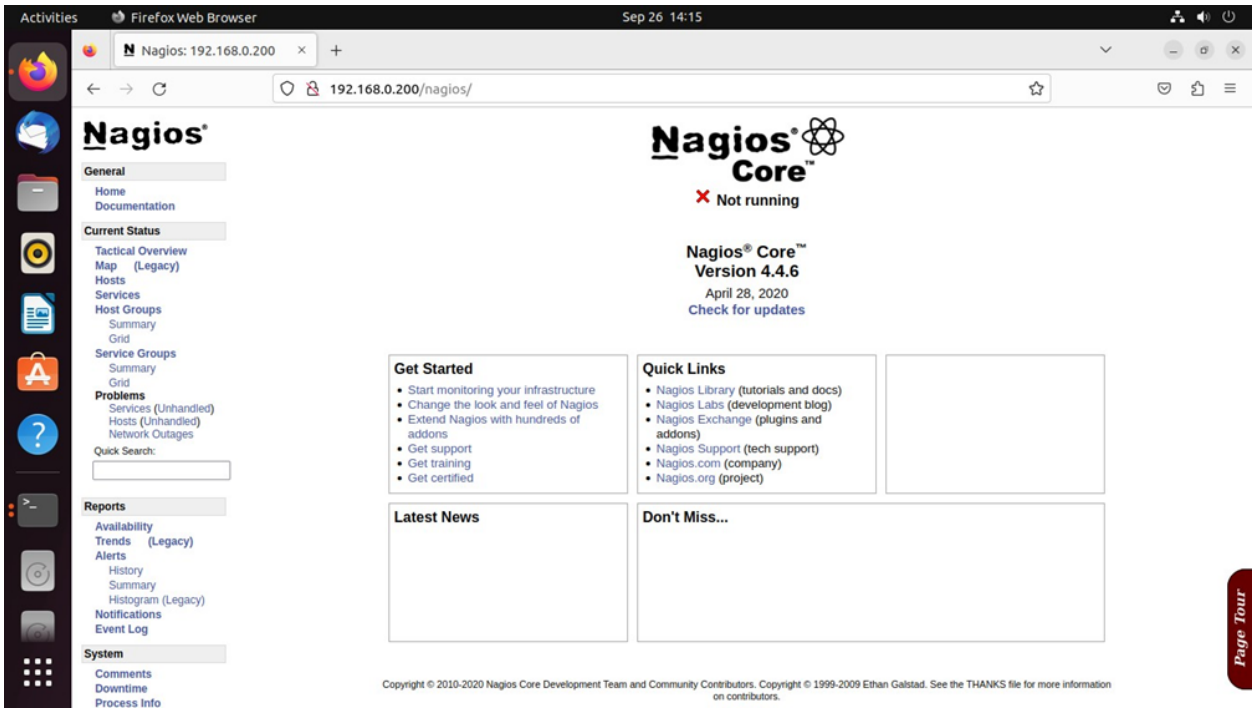

Activities
Terminal
Sep 26 14:05
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
Web Interface Options:
HTML URL: http://localhost/nagios/
CGI URL: http://localhost/nagios/cgi-bin/
Traceroute (used by WAP):
Review the options above for accuracy. If they look okay,
type 'make all' to compile the 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]
nagios.c:841:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
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]
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]
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]


Activities
Terminal
Sep 26 14:05
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
Enjoy.
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/lab1002/nagioscore-nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
/usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/lab1002/nagioscore-nagios-4.4.6/cgi'
make[1]: Leaving directory '/home/lab1002/nagioscore-nagios-4.4.6/cgi'
cd ./html && make install
make[1]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs/angular-1.3.9
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/angularjs/ui-utils-0.2.3
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/bootstrap-3.3.7
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/bootstrap-3.3.7/css


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 -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/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.cfg
/usr/bin/install -c -b -m 660 -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/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.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/switch.cfg /usr/local/nagios/etc/objects/switch.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-classicui' 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/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/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.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/switch.cfg /usr/local/nagios/etc/objects/switch.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-classicui' 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-enabled/nagios.conf
if [0 -eq 1]; then \
ln -s /etc/apache2/sites-enabled/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$





Conclusion :-

Here , we studied about Nagios and successfully installed nagios on ubuntu system.

Assignment 10

Aim :- To study puppet tools

LO Mapped :- LO1 , LO6

Theory :-

Q) What is Puppet ?

Puppet is an efficient system management tool for centralizing and automating the configuration management process.

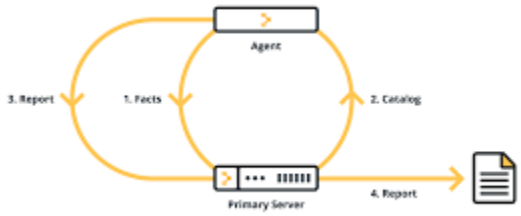
It can be used as a software deployment tool , It can also be utilized as open-source configuration management for server configuration, management, deployment, and orchestration.
Puppet is specially designed to manage the configuration of Linux and Windows systems.
It is written in Ruby and uses its unique Domain Specific Language (DSL) to describe system configuration.
Configuration management is maintaining software and computer systems (servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of the system state for project management and audit purposes.

There are two main versions of Puppet available in the market:

Open Source Puppet: This is a basic version of the Puppet configuration management tool. It is licensed under the Apache 2.0 system and can be downloaded from Puppet's website.

Puppet Enterprise: is a paid version that allows enterprises to manage nodes effectively with features like compliance reporting, orchestration, role-based access control, GUI, API, and command-line tools.

The diagram below shows how the server-agent architecture of a Puppet run works.



Q) What puppet can do?

Some of the key things that Puppet can do:

Configuration Management: Puppet allows you to define and manage the desired state of your infrastructure components (e.g., servers, applications, services) in a declarative manner. You specify how you want your systems to be configured, and Puppet ensures that they stay in that desired state.

Automation: Puppet automates the process of configuring and maintaining systems, reducing manual intervention and human error. It can handle routine tasks like software installation, configuration file management, and service provisioning.

Infrastructure as Code (IaC): Puppet enables the practice of treating infrastructure as code, meaning you define and manage your infrastructure using code files. This approach makes it easier to version, test, and collaborate on infrastructure changes.

Multi-Platform Support: Puppet supports various operating systems and can be used to manage both Unix/Linux and Windows environments. This makes it versatile for heterogeneous infrastructure.

Node Classification: Puppet allows you to classify nodes (servers or devices) based on their roles and responsibilities. You can define different configurations for different types of nodes, making it easy to scale and maintain large and diverse infrastructure.

Version Control: Puppet code can be stored in version control systems like Git, enabling collaboration, tracking changes, and ensuring a history of configuration changes.

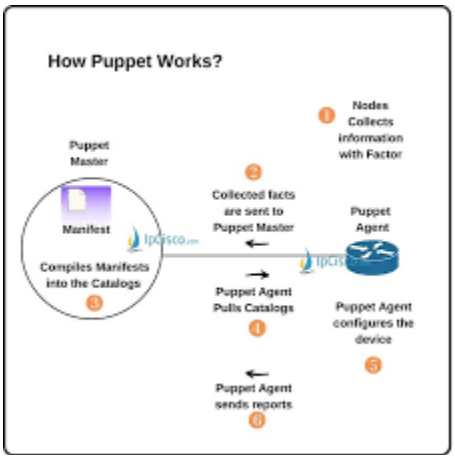
Reporting and Monitoring: Puppet provides reporting and monitoring capabilities to help you track the status of your infrastructure and configuration changes over time. You can identify issues, compliance violations, and performance bottlenecks.

Integration: Puppet can integrate with other DevOps tools and systems, such as continuous integration/continuous deployment (CI/CD) pipelines, monitoring tools, and orchestration platforms, to create a comprehensive automation and management ecosystem.

Q) How puppet works?

Puppet has a primary-secondary node architecture.

Puppet Architecture



The clients are distributed across the network and communicate with the primary-secondary environment where Puppet modules are present. The client agent sends a certificate with its ID to the server; the server then signs that certificate and sends it back to the client. This authentication allows for secure and verifiable communication between the client and the master.

The factor then collects the state of the clients and sends it to the master. Based on the fact sent, the master compiles the manifests into the catalogs, which are sent to the clients, and an agent executes the manifests on its machine. A report is generated by the client that describes any changes made and is sent to the master.

This process is repeated at regular intervals, ensuring all client systems are up to date. In the next section, let us find out about the various companies adopting Puppet as a part of our learning about what is Puppet.

Q) Puppet Blocks- Puppet Resources, Puppet Classes, Puppet Manifest, Puppet Modules.

In Puppet, various concepts and constructs are used to manage and configure systems and infrastructure. The four terms mentioned—**Puppet Resources, Puppet Classes, Puppet Manifests, and Puppet Modules**—are fundamental components of Puppet's configuration management system.

Puppet Resources:

Puppet resources represent individual components or objects that you want to manage on your systems, such as files, packages, services, users, and groups.

Resources are defined in Puppet code using a specific syntax. For example, you can define a file resource to ensure the existence of a file or a package resource to ensure that a particular package is installed.

Puppet resources are the building blocks used to declare the desired state of your infrastructure.

Puppet Classes:
Puppet classes are a way to organize and group related resources and configurations together.
Classes are defined in Puppet code and provide a modular and reusable way to organize your infrastructure configurations.
For example, you might create a "webserver" class that includes resources like a web server package, configuration files, and service management.

Puppet Manifests:
Puppet manifests are files that contain Puppet code. They are used to define the configuration and desired state of your systems.
Manifests typically have a .pp file extension and contain a collection of resource declarations, class includes, and other Puppet constructs.
Puppet manifests are the entry point for Puppet's configuration management, and they specify how resources and classes should be applied to nodes.

Puppet Modules:
Puppet modules are a way to package and distribute Puppet code and configurations in a structured and reusable manner.
A module is a directory structure that contains Puppet manifests, templates, files, and other assets needed to manage a specific piece of infrastructure or application.
Modules can be shared and reused across different Puppet environments and projects, making it easier to maintain and collaborate on Puppet code.
Common modules are often available on the Puppet Forge, a public repository of Puppet modules.

Q) What are benefits of Puppet tool?

Using a configuration management tool like Puppet in DevOps has many benefits.

Puppet's leveraging of Infrastructure as Code (IAC) features allow for efficient and flexible resolution of issues with version control, continuous delivery, peer review, and automated testing and deployment.

The other great benefit of using Puppet is that its **Master-Agent communication** using catalogs helps to identify the root causes of downtime and allows for a quick automated fix. This leads to overall downtime and faster Time To Recovery (TTR).

Puppet can also rely on the support of a **large open-source developer community** that assists developers in seeking solutions for various issues.

Puppet also **removes developers' need to adapt to a separate scripting language** to fulfill their tasks. It works seamlessly on platforms like Microsoft Windows, BSD, and Debian for Puppet.

Q) What are Puppet Manifest Files?

In Puppet, manifest files are used to define the desired configuration and state of your systems and infrastructure. Manifests are written in Puppet's domain-specific language (DSL), and they contain declarations of resources, classes, and other Puppet constructs. Manifests serve as the foundation for Puppet's configuration management process. Here are some key points about Puppet manifest files:

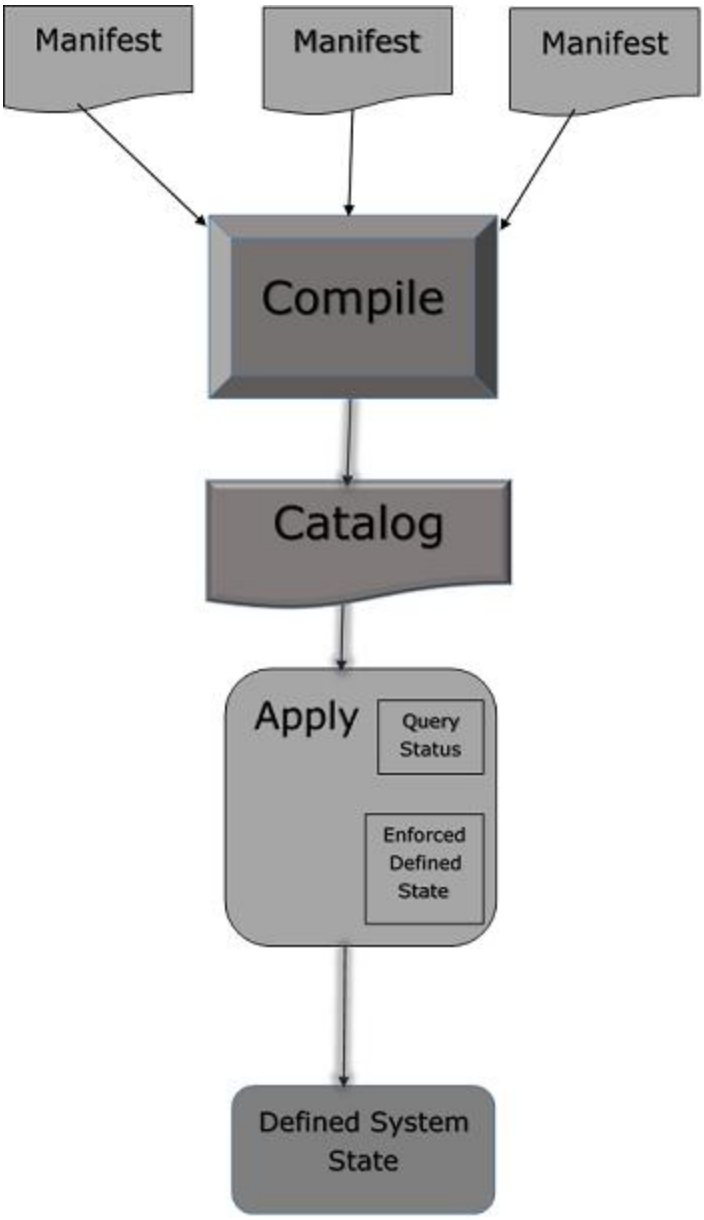
File Extension: Puppet manifest files typically have a .pp extension, which stands for Puppet Programming.

Resource Declarations: The primary purpose of a Puppet manifest is to declare the desired state of resources on a target node. Resources can represent various system components, such as files, packages, services, users, and groups. Each resource declaration specifies the resource type, a title (name or identifier for the resource), and attributes that define the desired state. For example, a file resource declaration might ensure that a specific file exists, has certain permissions, and is owned by a particular user.

Class Includes: Manifests can include or "include" Puppet classes. Classes are defined separately in modules (typically within a .pp file inside a module directory), and you can include them in your manifests to apply a predefined set of configurations to a node.
Variables and Facts: You can use variables and facts in Puppet manifests to make your configurations more dynamic and flexible. Facts provide information about the target node, and you can use them in your manifest logic.

Comments: Like any programming language, Puppet manifests can include comments to document your code and make it more understandable for others.

Modular Structure: Puppet manifests are often organized into modules for better code management and reuse. Modules encapsulate related Puppet code, including manifests, templates, and files, into a directory structure.



Q) Syntax of a Manifest File? why do we need Puppet Manifest Files? Writing a basic Manifest File with Examples.

A Puppet manifest file is written in Puppet's **domain-specific language (DSL)** and is used to declare the desired configuration and state of resources on target nodes. These manifest files are the heart of Puppet's configuration management system.

Syntax of a Puppet Manifest File:

A Puppet manifest file **consists of resource declarations, class includes, and other Puppet constructs.**

Resource Declaration: A resource declaration defines a specific resource (e.g., file, package, service) and its desired state. It follows this general syntax:

```
resource_type { 'resource_title':
  attribute1 => value1,
  attribute2 => value2,
  # Additional attributes as needed
}
```

resource_type: Specifies the type of resource (e.g., file, package, service).

'**resource_title':** Provides a name or identifier for the resource.

attribute1, attribute2: Represents attributes specific to the resource type (e.g., ensure, owner, content).

Class Include: You can include Puppet classes within your manifest using the include keyword:

include class_name

class_name: The name of the Puppet class to include.

Conditional Statements: Puppet manifests can include conditional statements to apply configurations based on conditions:

```
if condition {
  # Configuration for when the condition is true
} else {
  # Configuration for when the condition is false
}
```

Variables and Facts: You can use variables and facts to make your configurations dynamic:

$variable_name = value

$variable_name: Represents a variable that stores a value.

value: The value assigned to the variable.

Why Do We Need Puppet Manifest Files:

Declarative Configuration: Manifests allow you to declare the desired state of resources, making it clear what your infrastructure should look like.

Automation: Manifests automate the configuration of systems, reducing manual intervention and the potential for human error.

Consistency: Puppet ensures that configurations remain consistent across all nodes, even as they evolve or change.

Scalability: Puppet allows you to manage configurations for a large number of nodes simultaneously.
Version Control: Manifests can be version-controlled, enabling collaboration, change tracking, and rollback capabilities.

Example of a Basic Puppet Manifest File:

Simple example of a Puppet manifest that ensures the installation of the Nginx web server and starts the Nginx service:

```
# This manifest ensures that Nginx is installed and running.
class nginx {
  package { 'nginx':
    ensure => 'installed',
  }

  service { 'nginx':
    ensure  => 'running',
    enable  => true,
    require => Package['nginx'],
  }
}
```

In this example:
We define a class named "nginx" using the class keyword.
Inside the class, we declare a package resource to ensure that the Nginx package is installed.
We declare a service resource to ensure that the Nginx service is running and enabled.
The require attribute in the service resource specifies that it depends on the Nginx package being installed first.
When you apply this manifest to a node, Puppet will ensure that Nginx is installed and running according to the defined specifications.

Conclusion :-

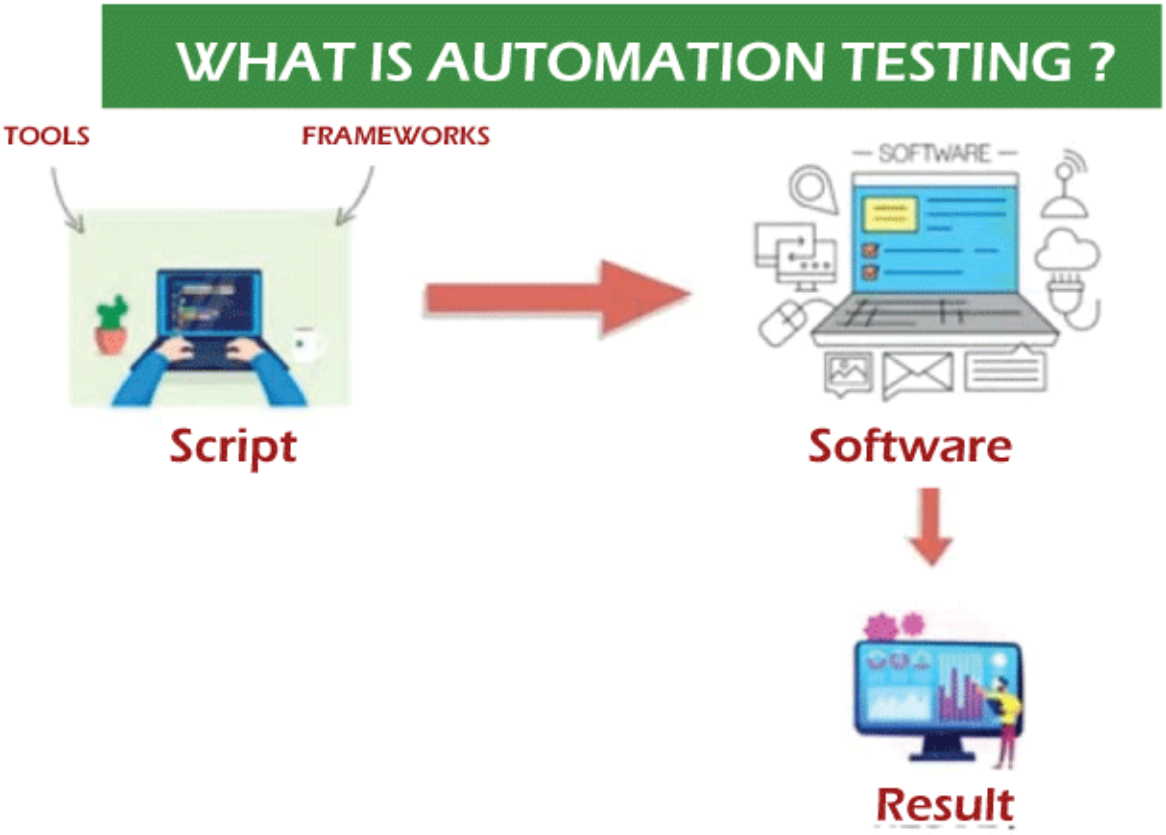
Learnt about basics of puppet tools in devops , their usage , benefits , basic syntax related to it and also implemented several examples to demonstrate above discussed things and explored about manifest files as well.

Written Assignment 1

Q1) What is test automation or automation testing? What are the advantages of automation testing?

Software testing is not a one-time process, as the developer test the software until it is declared error-free. It checks whether the software is showing the expected output per input. Software testing is done by the testers who test cases repeatedly to gain output according to the requirement in two ways: a) manually and b) automatically.

Every project or software goes through a testing procedure, and the type of testing method depends on various factors like the budget, expertise, suitability, requirement and timeline of the project. When the tester writes the testing script manually and tests the software until it functions properly known as manual testing & when this manual testing process becomes automatic, it can be defined as **automatic testing.**



"Automation testing refers to the automatic testing of the software in which developer or tester write the test script once with the help of testing tools and framework and run it on

the software. The test script automatically test the software without human intervention and shows the result (either error, bugs are present or software is free from them)."

There are certain steps followed in the automation testing process:

Step1: To convince the management
→ In the business, testing is not only in the hand of the tester because automation testing tools are expensive, and large licence fees are associated with them. A tester alone can't do it; they need to convince the management about the benefits test automation offers.
As we know, automation is a little expensive, so after doing a cost-benefit analysis, we need to convince management by preparing a detailed report about the benefit of test automation.
They have to convenience that automation test results can't be seen immediately. It takes 2-3 months to show results. You have to convenience yourself that it's a matter of patience to achieve an error-free application.

Step 2: Recruitment of Automation tool expert
→ Automation testing is done by automation tool experts who know how to use them for better results. There are two types: Automation architectures and automation engineers.
Automation architecture builds the automation framework, creates rules for scripting & designs the naming convention. They help the management analyze applications and tools used in the application so that they can select the right tool for automation. They are experienced in using different kinds of tools as they understand them well with their pros and cons, so they can select the test case that needs to be automated.
Automation engineers are experts in converting manual test cases into automated test scripts and work under the automation architect. Engineers must be good in programming (object-oriented language), which helps them understand, create & execute the scripts. Companies can hire them from outside or prepare them inside by training their existing manual testers.

Step 3: Selection of the right automation tool
→ This is the most important and challenging step in the testing process because the wrong tool gives you results that can harm the business. Before selecting an automation tool, one must know its pros and cons.
After that, business needs and requirements are considered and analyzed before making the final decision. Points considered in selecting the tool: The tool must lie in your budget, support the technologies used in the application, skilled resources are necessary, and tools have a good reporting mechanism.

Step 4: Choosing the application
→ For automation testing, it is important to select the right tests for automation with the right application, and it depends on certain factors like:
The selected application must be bug-free after manual testing
Application UI must not change frequently and must be stable
Selected application for automation must be in the early stage of its development with stable modules and manually tested results

Step 5: Automation teams are trained
→ With the technological change, manual testers need to be upgraded to become automation engineers. They must be trained in automation concepts and terminologies. The automation testing team must be trained to use the tool for testing and giving the expected results.

Step 6: Automation testing framework is developed
→ After selecting tools, applications and training for the automation team, it's time to develop the framework for the automation testing. The framework combines planning strategies and rules to write the test script. This could lead to the least amount of maintenance in the application, and if there is a need for any change that it's easy to deal with. It includes modular, data-driven, hybrid, keyword driven and linear frameworks.

Step 7: Preparing an execution plan
→ The automation testing is executed with an execution plan involving selecting an environment on which the script will execute, including browser, operating system and other hardware configurations. When writing the script, which configuration will be executed? The automation team executes the plan and states who will execute the scripts. The execution plan varies from company to company as some ask its developers to execute the plan before release, and some hire a direct resource.

Step 8: Writing of scripts
→ When the ground-level work is complete (including framework designing, execution of the plan, and resources trained on the new tool), the time comes to write the scripts. Points should be remembered while writing the script:
Code loss should be prevented by preserving it in source control
In source code history and version control should be preserved
Every script should be maintained in a formulated manner (proper naming conventions)
The scriptwriter must know programming languages because automation testing is a part of software development

Step 9: Reporting
→ Reporting includes the result of the testing created at the end of the execution. It consists of all the detailing of the testing and what is yet remaining. The result is written in tables and charts according to the management requirement and mailed to management.

Step 10: Script maintenance
→ Script maintenance step is required when a change is requested in the application. To cope with the changes, scripts are immediately updated to ensure flawless execution. Regular script maintenance is important for the smooth running of the application and bug-free operation.

Automation testing tools :
1) Smartbear
2) TestComplete

Advantages of Automation testing :

1) It saves time and cost in testing and provides an increment in the efficiency of testing.
2) Automation testing improves the accuracy of testing
3) With automation, more cycles can be achieved
4) It also ensures consistency in testing
5) It's test scripts can be reusable
6) Ability to cover the test application features widely
7) Automation testing results are reliable
8) In this testing, human intervention is not required
9) Speedily executes the testing process frequently and thoroughly

Q2) What is XPath? Explain the difference between single slash and double slash in XPath.

XPath stands for XML Path Language. It uses a non-XML syntax to provide a flexible way of addressing (pointing to) different parts of an XML document. It can also be used to test addressed nodes within a document to determine whether they match a pattern or not.

XPath is mainly used in XSLT, but can also be used as a much more powerful way of navigating through the DOM of any XML-like language document using XPathExpression, such as HTML and SVG, instead of relying on the Document.getElementById() or Document.querySelectorAll() methods, the Node.childNodes properties, and other DOM Core features.

XPath uses a path notation (as in URLs) for navigating through the hierarchical structure of an XML document. It uses a non-XML syntax so that it can be used in URIs and XML attribute values.

Single Slash (/):

1) It is used to create an absolute path.
1) Selects nodes that are direct children of the current node.
2) It specifies the direct path to the desired node.

Example: If you have the XML structure:

```
<bookstore>
 <book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
 </book>
 <book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J.K. Rowling</author>
 </book>
</bookstore>
```

Using the XPath expression /bookstore/book will select all the book elements that are direct children of the bookstore element.

Double Slash (//):

1) It is used to create a relative path.
2) Selects nodes from the current node regardless of their location in the document.
3) It enables the selection of nodes at any level beneath the current node.

Example: If you have the same XML structure, using the XPath expression //title will select all the title elements in the document, regardless of their position.

Difference between single slash and double slash

Differences	Single Slash (/)	Double Slash (//)
Type of Path	Specifies an absolute path.	Specifies a relative path.
Node Selection	Selects only the immediate children of the current node.	Selects nodes at any level beneath the current node.
Scope	Limited to the immediate child nodes of the current node.	Includes all descendant nodes, regardless of their level.
Traversal Depth	It moves only one level down the hierarchy.	It traverses through all levels of the XML hierarchy.
Performance	Generally faster than double slash as it narrows down the search scope.	Relatively slower compared to single slash, especially for large XML documents.

Specificity	**Provides specific and targeted node selection.**	**Provides a broader and less specific node selection.**
Usage	**Ideal when you want to access a specific child node directly.**	**Useful when you want to access nodes at any level regardless of their position in the XML structure.**
Syntax Example	`/bookstore/book`	`//title`

Written Assignment 2

Q1) What is Selenium? What are the Selenium suite components?

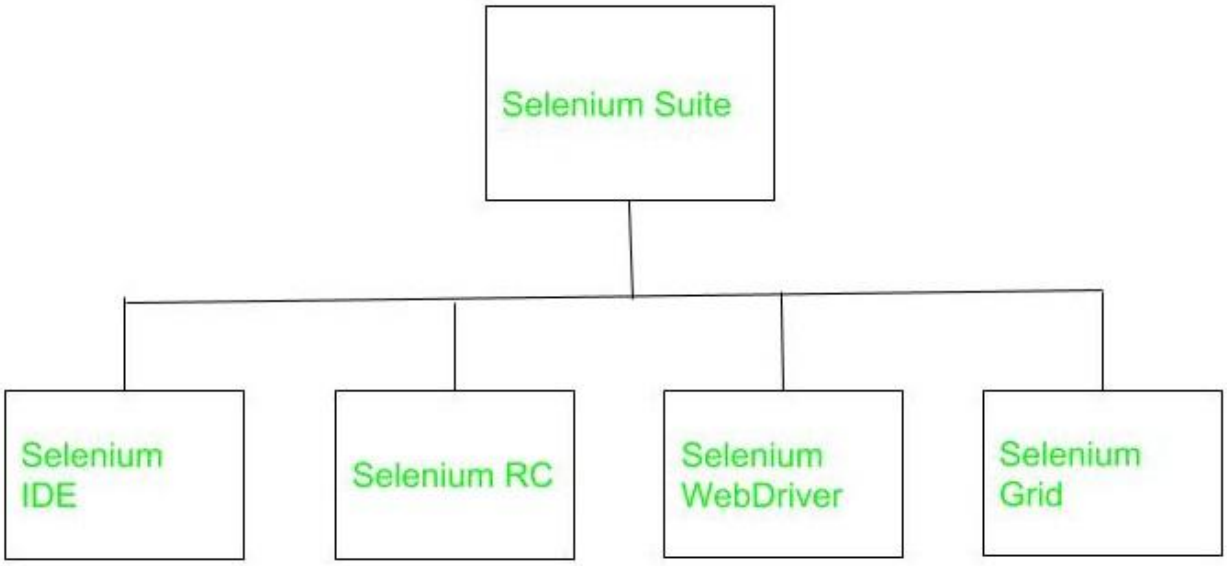
Selenium is a popular open-source software testing framework used for automating web applications. It is widely used for functional testing, regression testing, and performance testing. Selenium supports multiple programming languages, including Java, C#, Python, and Ruby, making it accessible to a wide range of developers.

Selenium provides several tools that allow developers to automate their tests, including:
Selenium WebDriver: A tool for automating browser interactions and for controlling web browsers programmatically.
Selenium IDE: A browser-based tool for recording and playing back user interactions with a web application.
Selenium Grid: A tool for running tests in parallel on multiple machines, allowing for faster test execution and improved resource utilization.

Selenium is an important tool in the software engineering process as it enables developers to automate repetitive and time-consuming testing tasks, freeing up their time to focus on other aspects of the software development process. The ability to automate testing also helps to ensure that software is thoroughly tested and that issues are identified and resolved more quickly, improving the overall quality of the software.

Selenium is an automation tool and portable software testing tool for web applications. A test domain-specific language is also provided, to write test cases one can use programming languages, including C#, Java, Perl, PHP, Python, Ruby, Scala, and Groovy. It does not support RIA(Rich Internet Application) Technology such as Silverlight JavaFX and Flex\Flash. Selenium can be easily used on the platforms like Windows, Linux, Solaris, and Macintosh. Selenium also supports Operating Systems for mobile applications like Android and iOS and windows.

Selenium uses many programming languages like Ruby, python C#, Java, Perl, and PHP. Safari, Internet Explorer, Mozilla Firefox, and Google Chrome are some of the browsers which are supported by Selenium.Selenium is not a single tool. It is a product suite of software consisting of the following components:



Selenium IDE: At the beginning Selenium IDE(Integrated Development Environment) was implemented as a Firefox add-on/plugin and now it can be used Selenium IDE on every web browser. It provides record and playback functionality. The figure shows Selenium IDE.

selenium ide

- It is an open-source tool.
- Provide base, for extensions.
- It provides multi-browser support.
- No programming language experience is required while using Selenium IDE.
- The user can set breakpoints and debug.
- It provides record and playback functions.
- Easy to Use and install.
- Languages supported by Selenium IDE are Java, Python, C# etc.

Selenium RC: RC stands for Remote Control. It allows the programmers to code in different programming languages like C#, Java, Perl, PHP, Python, Ruby, Scala, Groovy. The figure shows how the Remote Control Server works.

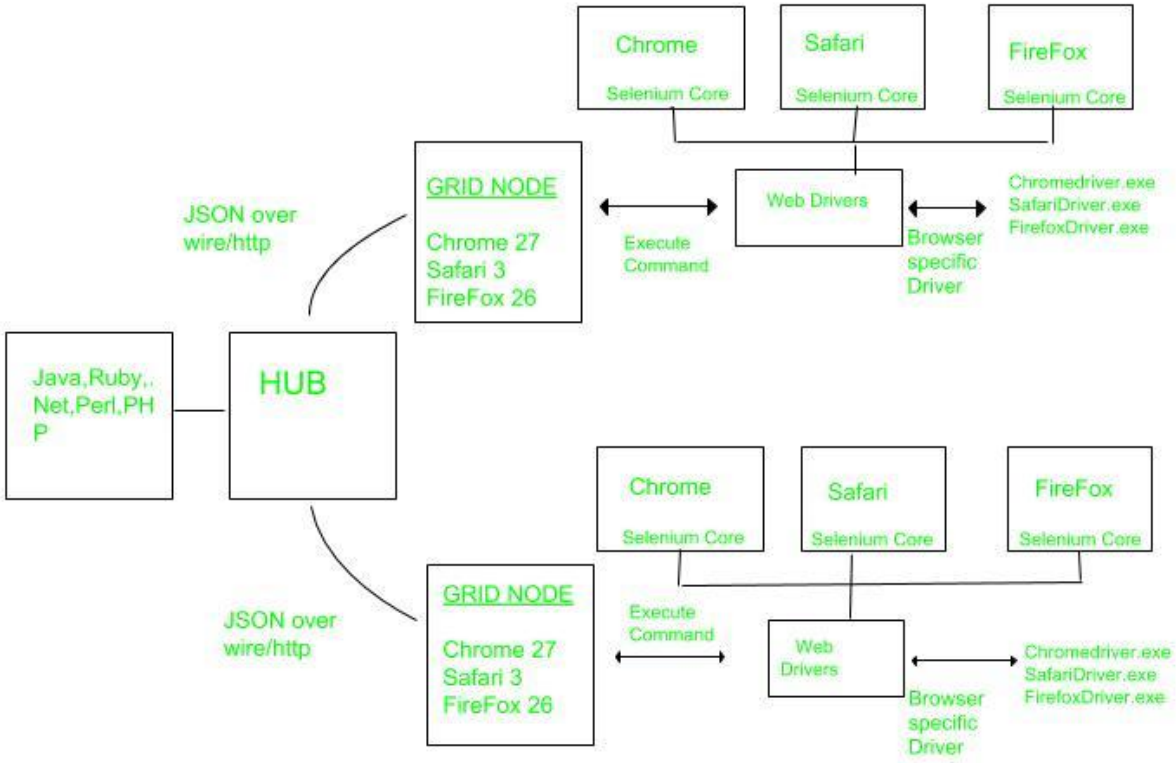
- It supports all web browsers.
- It can perform iteration and conditional operations.
- Execution is faster as compared to IDE.
- It has built-in test result generators.
- It supports data-driven testing.
- It has a matured and complete API.

- There is also support for cross browser testing.
- There is also support for user preferred languages.

Selenium Web Driver: Selenium Web Driver automates and controls initiated by the web browser. It does not rely on JavaScript for automation. It controls the browser directly by communicating with it. The figure shows how web driver works as an interface between Drivers and Bindings

- It directly communicates with the web browser.
- Execution is faster.
- It supports listeners.
- It supports IOS/Android application testing.
- Installation is simpler than Selenium RC.
- Purely object-oriented.
- No need of any separate Component.

Selenium Grid: Basically, it is a server that allows the test to use a web browser instance running on remote machines. It provides the ability to run the test on a remote web browser, which helps to divide a load of testing across multiple machines and it will save enormous time. It allows executing parallel tests across different platforms and operating systems.



Selenium Grid is a network of HUB & nodes. Each node registers to the HUB with a certain configuration and HUB is aware of the browsers available on the node. When a request comes to the HUB for a specific browser [with desired capabilities object], the HUB, if found a match for the requested web browser, redirects the call to that particular Grid Node and then a session is established bi-directionally and execution starts. It makes the easy use for multiple machines to run the tests in parallel.

Q2) 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 open-source automated testing tool for web applications. Its popularity can be attributed to several key factors, making it a preferred choice for testing web applications or systems:

Open-Source Nature: Selenium is an open-source tool, which means it is free to use. This makes it accessible to a broad user base and allows for continuous community-driven development and support.

Cross-Browser Compatibility: Selenium supports various web browsers such as Chrome, Firefox, Internet Explorer, and Safari. This cross-browser compatibility ensures that the application behaves consistently across different browsers.

Support for Multiple Programming Languages: Selenium supports multiple programming languages including Java, Python, C#, Ruby, and others. This flexibility allows testers and developers to use their preferred language, making it easier to integrate Selenium into existing development and testing environments.

Flexibility and Extensibility: Selenium offers flexibility in terms of integrating with other tools and frameworks, enabling testers to build custom test frameworks suited to their specific needs. It can also be integrated with various continuous integration tools, enabling the automation of the entire testing process.

Parallel Test Execution: Selenium supports parallel test execution, which significantly reduces the time required to execute a large suite of test cases. This feature is crucial for improving the efficiency of the testing process, especially in Agile and DevOps environments.

Support for Various Operating Systems: Selenium can be used on multiple operating systems, including Windows, macOS, and Linux, allowing for testing across different environments without limitations.

Rich Set of Features: Selenium provides a rich set of features for automated testing, including the capability to simulate user interactions such as clicking, typing, and selecting elements. It also supports the verification of web page content, enabling comprehensive testing of web applications.

Active Community Support: Selenium has a large and active community of users, developers, and testers who contribute to its development, provide support, and share knowledge. This active community ensures that Selenium stays up-to-date with the latest web technologies and best practices.

Due to these factors, Selenium is often advised as the preferred testing tool for web applications or systems, as it offers a powerful, flexible, and reliable solution for automated testing that can be seamlessly integrated into the development and testing processes of various organizations