Datta Meghe College of Engineering **Airoli, Navi Mumbai**	**DEPARTMENT OF INFORMATION TECHNOLOGY** **ACADEMIC YEAR : 2025 – 26 (TERM – I)** **List of Experiments** **Course Name: Advance DevOPs Lab** **Course Code : ITL504**

Sr. No	Name of experiment	Cos Covered	Page No.	Date of Performance	Date of Submission	Marks & Signature
1	To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.	ITL504.1				
2	To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.	ITL504.1				
3	To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.	ITL504.1 ITL504.2				
4	To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.	ITL504.1, ITL504.2				
5	To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.	ITL504.1 ITL504.3				
6	To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.	ITL504.1 ITL504.3				
7	To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.	ITL504.1, ITL504.4				
8	Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.	ITL504., ITL504.4				

9	To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.	ITL504.1, ITL504.5				
10	To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.	ITL504.1 ITL504.5				
11	To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python/ Java/ Nodejs	ITL504.1 ITL504.6				
12	ASSIGNMENT 1					
13	ASSIGNMENT 2					

This is to certify that Mr. / Miss _____

of_____Roll No._____has performed the Experiments Work mentioned above in the premises of the institution.

Practical In charge

DATTA MEGHE COLLEGE OF ENGINEERING, AIROLI, MUMBAI

Vision	To create values-based technocrats to fit in the world of work and research.
Mission	To adopt the best practices for creating competent human beings to work in the world of technology and research.

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision	To develop and foster students for successful career in the dynamic field of Information Technology.	
Mission	**M1**	To create and disseminate knowledge through research, teaching & learning and to enhance society in meaningful and sustainable ways.
	M2	To impart suitable environment for students and staff to showcase innovative ideas in the field of IT.
	M3	To bridge the curriculum gap by facilitating effective interaction among industry and Staff/Students.
Program Educational Objectives (PEOs)	**PEO1**	Develop proficiency as an IT technocrat with an ability to solve a wide range of computational problems in industry, government, or other work environments.
	PEO2	Attain the ability to adapt quickly to new environments and technologies, assimilate new information, and work in multi-disciplinary areas with a strong focus on innovation and entrepreneurship.
	PEO3	Prepare graduates with the ability of life-long learning to innovate in ever-changing global economic and technological environments of the current era.
	PEO4	Possess the ability to function ethically and responsibly with good cultural values and integrity to apply the best principles and practices of Information Technology towards the society.
Program Specific Outcomes (PSOs)	**PSO 1**	Apply Core Information Technology knowledge to develop stable and secure IT system.
	PSO 2	Design, IT infrastructures for an enterprise using concepts of best practices in information Technology and security domain.
	PSO 3	Ability to work in multidisciplinary projects and make it IT-enabled by adapting latest trends and technologies like Analytics, Blockchain, Cloud, Data science.

Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Datta Meghe College of Engineering, Airoli

Department of Information Technology

Course Name: Advance DevOPs Lab **Course Code: ITL504**

Class: T.E. Semester: V **A. Y.: 2023-24**

Course Outcomes

ITL504.1	To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements
ITL504.2	To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
ITL504.3	To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure
ITL504.4	To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Technique
ITL504.5	To use Continuous Monitoring Tools to resolve any system errors (low memory, unreachable server etc.) before they have any negative impact on the business productivity
ITL504.6	To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework

Datta Meghe College of Engineering, Airoli

Department of Information Technology

Course Name: Advance DevOPs Lab **Course Code: ITL504**

Class: B.E. Semester: V **A. Y.: 2023-24**

RUBRICS FOR GRADING EXPERIMENTS

Rubric Number	Rubric Title	Criteria	Marks (out of 15)
R1	Tool Understanding	Shows adequate knowledge of the experiment/ problem statement given.	4,5
		Shows some understanding of the experiment/ problem statement given.	3
		Shows hardly any understanding of the experiment/ problem statement given.	1,2
R2	Use of Tool and Result Validation	Output was perfectly desired and expected one. It was satisfying all the requirement of problem statement.	4,5
		Output was partially satisfying the requirement of Problem statement. It was not completely expected/desired one.	3
		Output was not problem specific. It was quite random and vague.	1,2
R3	Lab Ethics and Presentation	-Student's presence and behavior was professional and respectful. - Implemented the experiment and submitted writeup on time.	4,5
		- Student need a bit improvement in their presence and behavior to be professional and respectful. - Implemented the experiment and submitted writeup with the delay of one week i.e. till next Session.	3
		- Student need a significant improvement in their presence and behavior to be professional and respectful. - Deadline for implementing the experiment and submitting writeup has been exceeded by more than a week.	1,2

Datta Meghe College of Engineering, Airoli

Department of Information Technology

Course Name: Advance DevOPs Lab **Course Code: ITL504**

Class: B.E. Semester: V **A. Y.: 2023-24**

RUBRICS FOR GRADING ASSIGNMENTS

Rubric Number	Rubric Title	Criteria	Marks (out of 5)
R1	**Punctuality, Neatness**	On-time	3
		Delayed by not more than a week	2
		Delayed more than a week	1
R2	**Originality of Content**	Original Content	2
		Plagiarized work	1

EXPERIMENT : 1

Date of Performance	
Date of Submission	

AIM

To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.

PROBLEM DEFINITION

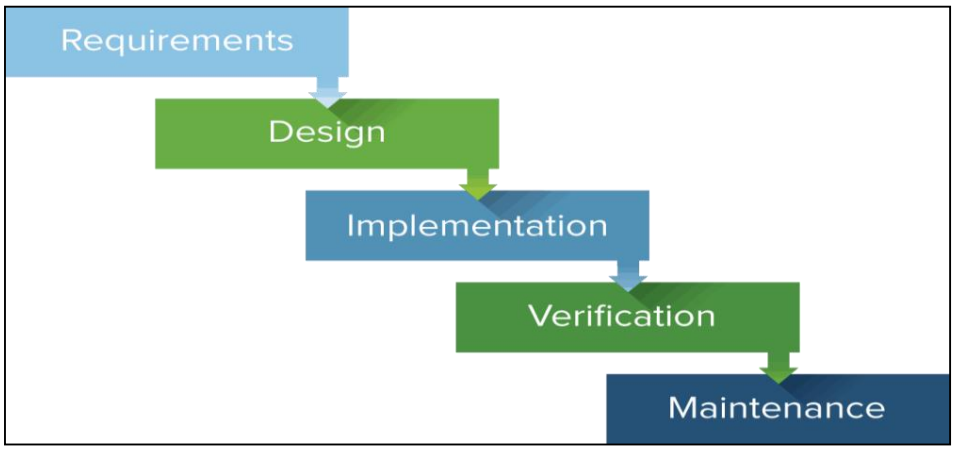
Explore DevOps principles, practices, and the responsibilities of a DevOps Engineer.

THEORY

EVOLUTION OF SOFTWARE DEVELOPMENT OVER THE YEARS :

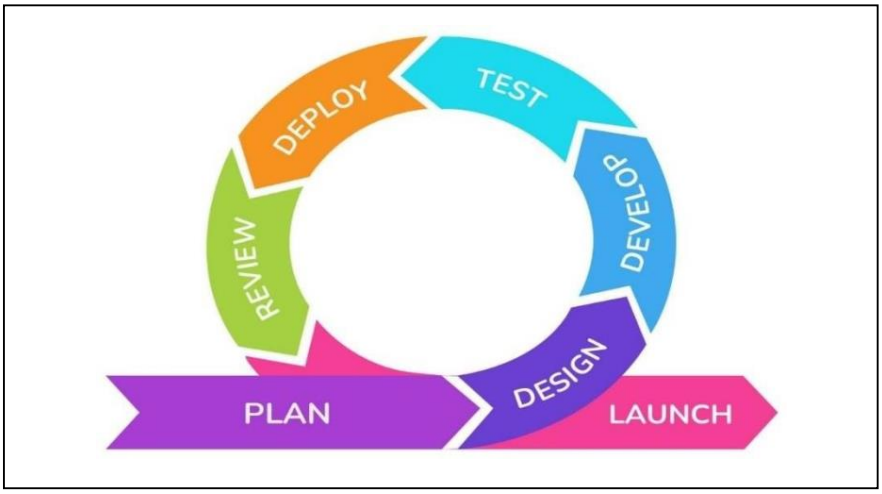
The WATERFALL METHODOLOGY is a traditional, linear approach to software development.

- Linear Process: Follows a sequential, step-by-step approach where each phase must be completed before the next begins.
- No Overlapping Stages: Each stage is strictly dependent on the previous one, and once a stage is completed, it cannot be revisited.
- Clear Documentation: Extensive documentation is produced at every stage, ensuring that the requirements and solutions are clearly defined.
- Limited Flexibility: Changing requirements during the development process is difficult, as each phase relies on the completion of the previous one.
- Easy to Manage: Its structured nature makes it easy for managers to track progress, deadlines, and costs.
- Best for Stable Projects: It is ideal for projects with well-understood requirements that are unlikely to change during development.
- Time-Consuming: The linear nature can result in longer project timelines due to the lack of iteration and flexibility.



The AGILE METHODOLOGY is a flexible, iterative approach to software development that emphasizes collaboration, adaptability, and continuous delivery.

- Iterative Process: Agile uses short, iterative cycles called sprints, producing functional software at the end of each sprint, allowing for ongoing feedback.
- Overlapping Stages: Development, testing, and design can occur simultaneously, enabling faster adjustments.
- Minimal Documentation: Focus is on working software rather than detailed documentation, prioritizing communication.
- High Flexibility: Agile adapts easily to changing requirements and feedback.
- Frequent Collaboration: Teams, stakeholders, and customers collaborate continuously to ensure alignment.
- Best for Uncertain Projects: Ideal for projects with evolving requirements, where flexibility is crucial.
- Faster Delivery: Enables frequent releases by delivering working software at the end of each sprint.
- Continuous Improvement: Regular retrospectives after each sprint help the team improve processes and performance.



WHY DEVOPS?

1. Bridging Development and Operations: DevOps was created to eliminate the gap between development and operations teams, which caused delays and errors in software releases.

2. Streamlining Delivery: By promoting collaboration, automation, and continuous feedback, DevOps accelerates software development, ensuring faster, high-quality product delivery while maintaining stability and security.

BENEFITS OF DEVOPS

- Faster Delivery: Continuous integration and delivery (CI/CD) pipelines enable rapid, frequent releases.
- Improved Collaboration: Breaks down silos between teams, leading to better communication and alignment.
- Increased Efficiency: Automation of repetitive tasks reduces manual errors and accelerates processes.
- Higher Quality: Early bug detection through continuous testing improves the reliability and stability of software.
- Scalability and Flexibility: DevOps practices like Infrastructure as Code (IaC) ensure scalable and consistent infrastructure management.
- Enhanced Security: Integrating security early in the development lifecycle (DevSecOps) ensures compliance and reduces vulnerabilities.

DEVOPS PRINCIPLES

DevOps emphasizes a culture of shared responsibility and collaboration between development and operations teams. Automation reduces manual tasks, increases speed, and minimizes errors. Continuous Integration and Delivery (CI/CD) ensures frequent testing, integration, and deployment for faster, reliable releases. DevOps is customer-focused, delivering updates based on feedback to enhance software quality.

DEVOPS PRACTICES

DevOps practices are designed to support the principles of the methodology:

- Infrastructure as Code (IaC) enables teams to manage and provision infrastructure using code, ensuring consistent and repeatable environments across development, testing, and production.
- Continuous Testing automates the testing process, allowing for early detection of bugs and issues before they reach production. This ensures higher quality code with every release.
- Monitoring and Logging help track the performance and health of systems in real-time, enabling teams to quickly respond to issues and maintain system reliability and uptime. This also feeds into continuous feedback loops for constant improvement.

DEVOPS ENGINEER ROLE AND RESPONSIBILITIES

A DevOps engineer is responsible for bridging the gap between development and operations. Their role includes:

- Automation: Automating workflows, including code deployments, system configurations, and environment setup, to streamline processes.
- Managing CI/CD Pipelines: Setting up and maintaining continuous integration and delivery pipelines to ensure quick, reliable, and frequent releases of code to production.
- Monitoring Infrastructure: Implementing monitoring tools and techniques to ensure that the systems are performing optimally and identifying potential bottlenecks or failures before they occur.
- Collaboration: Working closely with both development and IT operations teams to create seamless communication and efficient workflows.
- Security and Compliance: Ensuring that the automation processes follow security and compliance requirements throughout the software delivery lifecycle.

CONCLUSION

We conclude that DevOps unites development and operations through automation, continuous integration, and deployment, with engineers driving improved software quality and delivery speed.

R1 **(5 marks)**	**R2** **(5 marks)**	**R3** **(5 marks)**	**TOTAL** **(15 marks)**	**SIGN**

EXPERIMENT : 2

Date of Performance	
Date of Submission	

AIM

To understand Version Control System / Source Code Management, install git and create a GitHub account.

PROBLEM DEFINITION

Learn about version control systems and source code management by setting up Git and creating a GitHub account.

THEORY

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Some basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Some advanced Git operations are:

1. Branching
2. Merging
3. Rebasing

Installation of Git

1. In Windows, download Git from https://git-scm.com/ and perform the straightforward installation.

2. In Ubuntu, install Git using `$ sudo apt install git`. Confirm the version after installation with `$ git --version`.

Execution

To perform version control, create a directory named `dvcs` (Distributed Version Control System) and change the directory to `dvcs`:

$ mkdir git-dvcs

$ cd git-dvcs/

Check the user information using:

$ git config --global

Since there are no users defined, define them using the following commands:

$ git config --global user.name "abcd"

$ git config --global user.email “xyz….@gmail.com”

Check the list of users:

$ git config --global --list

user.name=abcde

user.email=xyz….@gmail.com

Create a repository for version control named "git-demo-project":

$ mkdir git-demo-project

$ cd git-demo-project/

Initialize the repository:

$ git init

If you have an existing repository, delete the `.git` file and reinitialize it:

$ rm -rf .git/

$ git init

Add files to the repository:

$ git add .

To check the status of the repository:

$ git status

Commit the changes:

$ git commit -m "First Commit"

Add `index.html` to the directory

$ git add .

Commit changes:

$ git commit -am "Express Commit"

Make changes to `index.html` and create a file `teststatus`:

$ nano index.html

$ touch teststatus

Discard changes:

$ git restore <file>

Add `index.html` and `teststatus`:

$ git add index.html

$ git add teststatus

$ git commit -am "Express commit"

View commit history:

$ git log

Create a repository on GitHub:

1. Open github.com and create an account.
2. After logging in, select "New repository" from the menu.
3. Specify a name for the repository and select the public option, then click "Create repository."

Fork the repository:

1. Log in with another account.

2. Copy and paste the URL of the repository.

3. Click "Fork" to clone it to another account.

Push changes to the web repository:

$ git remote add origin https://github.com/MSid01/TriviaQuiz.git

$ git remote show origin

$ git remote add origin https://github.com/bhushanjadhav1/Myrepository.git

$ git remote rm origin

$ git push -u origin master

Pull changes:

$ git pull

Fetch changes:

$ git fetch

Merge fetched changes:

$ git merge origin/master

OUTPUT

```
PS C:\Users\Admin> git --version
git version 2.45.2.windows.1
PS C:\Users\Admin> cd git-dvcs/
PS C:\Users\Admin\git-dvcs> git config --global user.name "ashitosh"
PS C:\Users\Admin\git-dvcs> git config --global user.email "ashitoshsabale@gmail.com"
PS C:\Users\Admin\git-dvcs> git congig --global --list
git: 'congig' is not a git command. See 'git --help'.

The most similar command is
        config
PS C:\Users\Admin\git-dvcs> git config --global --list
user.name=ashitosh
user.email=ashitoshsabale@gmail.com
```

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs
$ mkdir demo-project

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs
$ cd demo-project

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project
$ git init
Initialized empty Git repository in D:/Devops/git-dvcs/demo-project/.git/
```

MINGW64:/d/Devops/git-dvcs/demo-project

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git add .

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git commit -m "first commit"
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git add .

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git commit -am "express commit"
[master (root-commit) d015038] express commit
 2 files changed, 173 insertions(+)
 create mode 100644 exp2.html
 create mode 100644 styles2.css
```

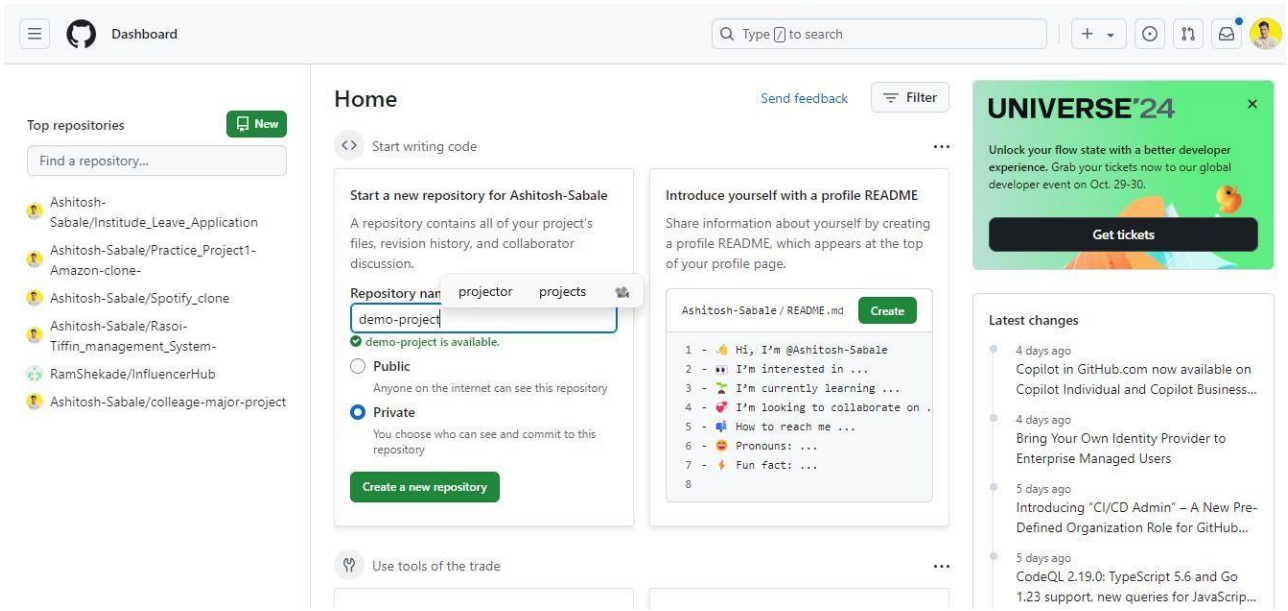
```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ nano exp2.html
```

MINGW64:/d/Devops/git-dvcs/demo-project

```
  GNU nano 8.2                                exp2.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment2</title>
    <link rel="stylesheet" href="styles2.css">
</head>
<body>
    <header>
        <h2>KRISH PATIL</h2>
        <p>Web-dev | Editor | Designer</p>
    </header>

    <nav>
        <ul>
            <li><a href="">Personal Info</a></li>
            <li><a href="">Education</a></li>
            <li><a href="">Skills</a></li>
            <li><a href="">Projects</a></li>
            <li><a href="">Personal Interest</a></li>

        </ul>
    </nav>

    <section id="personal-info">
        <h2>Pesonal Info</h2>
        <b>Name: </b>Krish Namdev Patil <br>
        <b>Age: </b>20 <br>
        <b>Address: </b>Bhiwandi , Thane , Maharashtra 421302
    </section>

    <section>
        <h2>Education</h2>
        <table>
            <thead>
                <tr>
                    <th>Qualification</th>
                    <th>From</th>
                    <th>Year</th>
                </tr>
            </thead>

            <tbody>
                <tr>
                [ Read 99 lines (converted from DOS format) ]
^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git restore exp2.html

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git add exp2.html

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git add teststatus

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git commit -am "express commit"
[master 6993d75] express commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 teststatus

Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git log
commit 6993d75b94e02dccfe0ad2d5d4375e6beb8c91aa (HEAD -> master)
Author: Krish <krishpatil5677@gmail.com>
Date:   Sun Sep 22 20:53:49 2024 +0530

    express commit

commit d0150386b3f2996f33474594ed34243c8762790b
Author: Krish <krishpatil5677@gmail.com>
Date:   Sun Sep 22 20:47:22 2024 +0530

    express commit
```

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git remote add origin https://github.com/Krish-Patil-5677/Devops.git
git branch -M main
git push -u origin main

Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 1.89 KiB | 969.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Krish-Patil-5677/Devops.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



CONCLUSION

Thus, we understood the Version Control System and Source Code Management by installing Git and creating a GitHub account, which facilitated efficient code management and collaboration.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT : 3

Date of Performance	
Date of Submission	

AIM

To perform various GIT operations on local and remote repositories using GIT Cheat-Sheet 4.

PROBLEM DEFINITION

Execute various Git operations on local and remote repositories with the help of a Git cheat sheet.

THEORY

What is GIT?

Git is the most widely used modern version control system in the world today. It 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 significant number of software projects, both commercial and open source, rely on Git for version control. Developers experienced 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 Integrated Development Environments (IDEs).

Git employs a distributed architecture, making it a Distributed Version Control System (DVCS). Unlike older version control systems such as CVS or Subversion (SVN), which have a single central repository for the full version history of the software, Git allows each developer's working copy of the code to be a repository that contains the complete history of all changes.

In addition to being distributed, Git is designed with performance, security, and flexibility in mind. Local repositories are physical, locally-managed repositories into which you can deploy artifacts. Using local repositories, Artifactory provides a central location to store your internal binaries. Repository replication enables sharing binaries with teams located in remote locations.A remote repository in Git, also known as a remote, is a Git repository hosted on the Internet or another network.

Here are some common Git operations:

1. `git init` - Initialize a new Git repository.
2. `git clone <repository>` - Clone an existing repository from a remote source.
3. `git add <file>` - Stage changes for the next commit.
4. `git commit -m "<message>"` - Commit staged changes with a descriptive message.
5. `git status` - Check the status of changes in the working directory and staging area.

6. `git log` - View the commit history.

7. `git branch` - List, create, or delete branches.

8. `git checkout <branch>` - Switch to a different branch.

9. `git merge <branch>` - Merge changes from one branch into the current branch.

10. `git pull` - Fetch and merge changes from a remote repository into the current branch.

11. `git push` - Upload local commits to a remote repository.

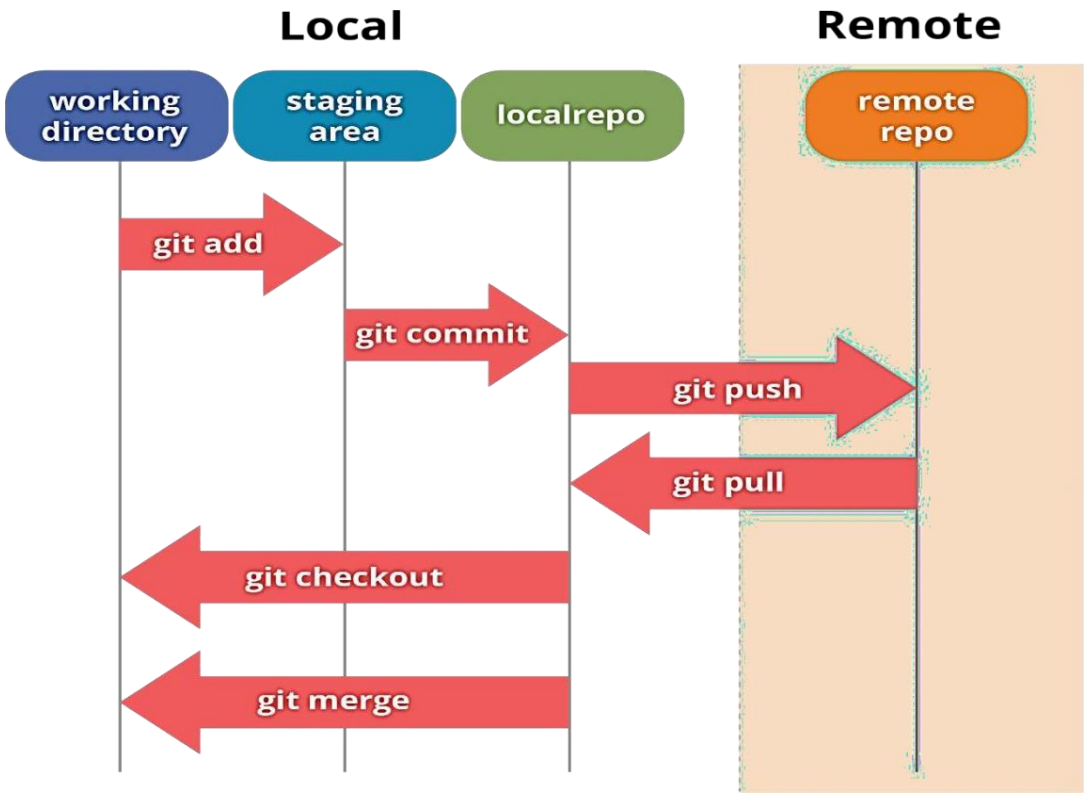
12. `git fetch` - Download changes from a remote repository without merging.

13. `git revert <commit>` - Create a new commit that undoes changes from a previous commit.

14. `git reset <file>` - Unstage a file from the staging area.

15. `git rm <file>` - Remove a file from the working directory and staging area.

These operations cover most common tasks when working with Git repositories.



OUTPUT

Git init

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project
$ git init
Initialized empty Git repository in D:/Devops/git-dvcs/demo-project/.git/
```

Git clone

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (main)
$ git clone https://github.com/kubowania/mario.git
Cloning into 'mario'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 28 (delta 7), reused 7 (delta 7), pack-reused 17 (from 1)
Receiving objects: 100% (28/28), 9.10 KiB | 4.55 MiB/s, done.
Resolving deltas: 100% (7/7), done.
```

Git log

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (master)
$ git log
commit 6993d75b94e02dccfe0ad2d5d4375e6beb8c91aa (HEAD -> master)
Author: Krish <krishpatil5677@gmail.com>
Date:   Sun Sep 22 20:53:49 2024 +0530

    express commit

commit d0150386b3f2996f33474594ed34243c8762790b
Author: Krish <krishpatil5677@gmail.com>
Date:   Sun Sep 22 20:47:22 2024 +0530

    express commit
```

Git branch

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (main)
$ git branch
* main
```

Git pull

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (main)
$ git pull
Already up to date.
```

Git push

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (main)
$ git push
Everything up-to-date
```

Git rm

```
Admin@DESKTOP-GML2GLO MINGW64 /d/Devops/git-dvcs/demo-project (main)
$ git rm teststatus
rm 'teststatus'
```

CONCLUSION

Therefore, performing various Git operations on local and remote repositories using Git Cheat Sheets enabled effective code management and version control.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT : 4

Date of Performance	
Date of Submission	

AIM

To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job

PROBLEM DEFINITION

Implement continuous integration by installing and configuring Jenkins with Maven, Ant, or Gradle to establish a build job.

THEORY

Jenkins is a popular open-source tool for continuous integration and build automation. It allows executing a predefined list of steps, such as compiling Java source code and building a JAR from the resulting classes. The execution can be triggered by time or events, for instance, compiling your Java-based application every 20 minutes or after a new commit in the related Git repository. Possible steps executed by Jenkins include:

- Performing a software build using a build system like Apache Maven or Gradle

- Executing a shell script

- Archiving a build result

- Running software tests

Jenkins monitors the execution of these steps and can halt the process if any step fails. It can also send notifications in case of a build success or failure. Jenkins is extendable with additional plugins, such as those for building and testing Android applications.

Continuous integration is the process of integrating all development work as early as possible. The resulting artifacts are automatically created and tested, allowing errors to be identified early in the project. The Jenkins build server provides this functionality.How to Download Jenkins?

Steps to Install Jenkins:

1. Go to https://www.jenkins.io/download/ and select the platform (Windows in this case).

2. Go to the download location on your local computer and unzip the downloaded package. Double-click on the unzipped `jenkins.msi`. Note that using a WAR (Web Application Archive) is possible but not recommended.

3. In the Jenkins setup screen, click "Next."

4. Choose the installation location (default is `C:\Program Files (x86)\Jenkins`) and click "Next."

5. Once the installation is complete, click "Finish."

6. During installation, an info panel may appear indicating that a system reboot might be needed for a complete setup. Click "OK" when prompted.

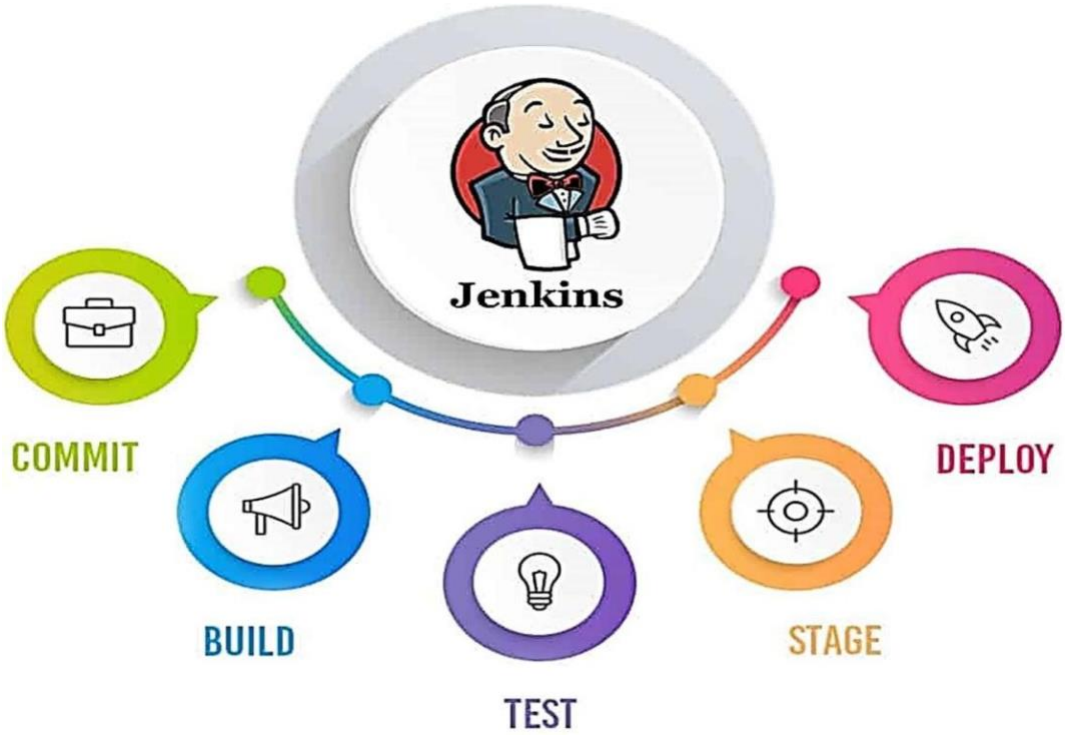
How to Unblock Jenkins:

1. After completing the Jenkins installation, a browser tab will pop up asking for the initial Administrator password. Access Jenkins by navigating to http://localhost:8080. If you can access this URL, Jenkins is successfully installed.

2. Find the initial Administrator password under the Jenkins installation path (set in Step 4). For the default location (`C:\Program Files (x86)\Jenkins`), the file `initialAdminPassword` is located at `C:\Program Files (x86)\Jenkins\secrets`. If a custom path was used, check that location for the `initialAdminPassword` file.

3. Open the `initialAdminPassword` file and copy its contents.

4. Paste the password into the browser's pop-up tab at http://localhost:8080/login?form=%2F and click "Continue."



Customize Jenkins:

1. Click on the "Install suggested plugins" button to allow Jenkins to retrieve and install essential plugins. Jenkins will start downloading and installing all necessary plugins.

Note: You can choose the "Select Plugins to Install" option to manually select the plugins you want to install.

2. After installing the suggested plugins, the "Create First Admin User" panel will appear. Fill in the fields with desired account details and click "Save and Finish."

3. You will then be asked for URL information to configure the default instance path for Jenkins. Leave it as is to avoid confusion. If port 8080 is already in use, choose a different port for Jenkins, save the settings, and click "Save and Continue."

Congratulations! Jenkins is now successfully installed. Click the "Start using Jenkins" button to access your Jenkins instance, ready to create your first Jenkins jobs.

Adding Tools and Plugins:

1. Add Git.

2. Add Gradle.

3. Add Ant.

4. Add Maven.

Create a New Build Job in Jenkins:

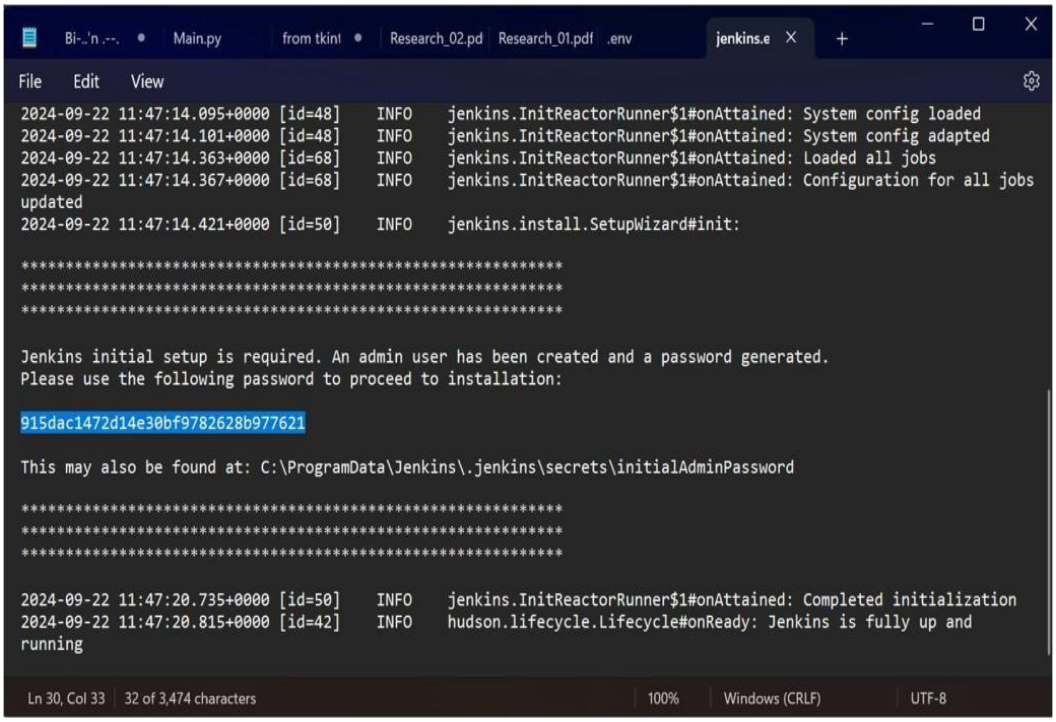
1. Connect to Jenkins for initial configuration by opening a browser and navigating to http://localhost:8080.

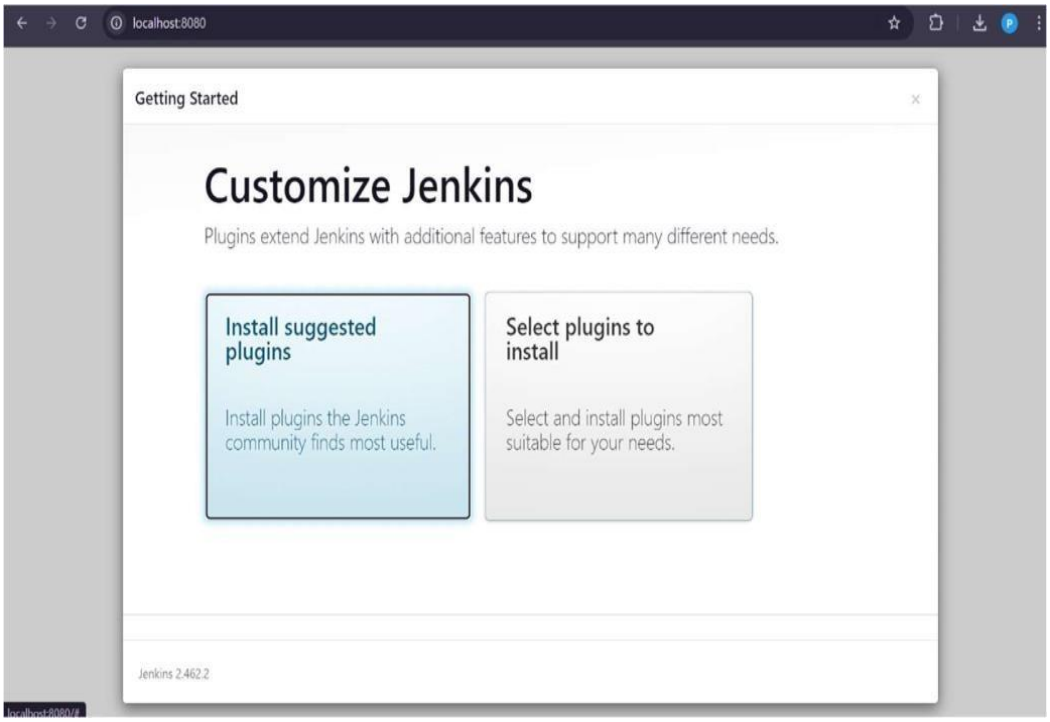
2. Copy the initial password from the file system of the server.

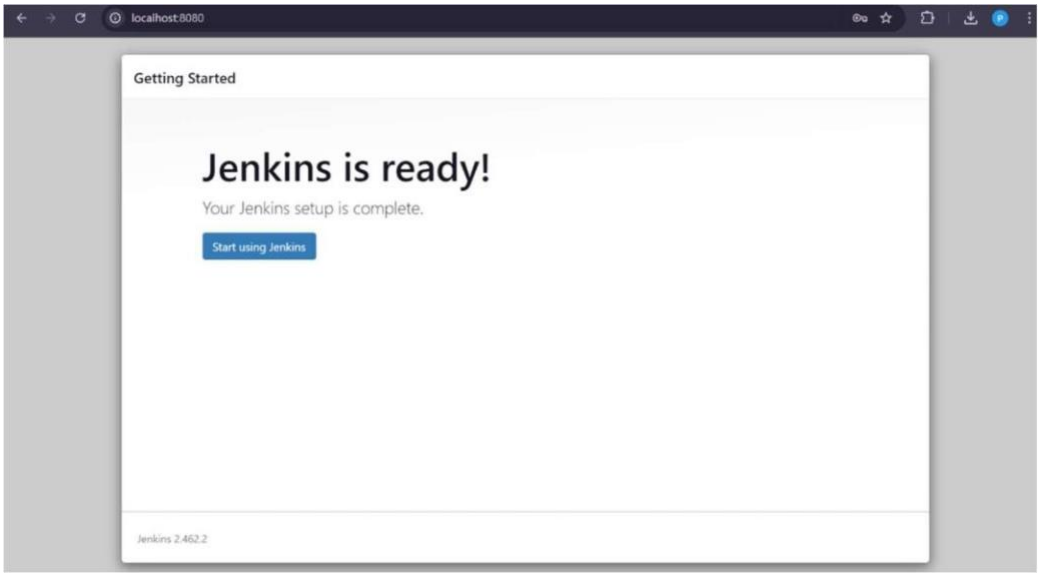
3. Select to install plugins, choosing "Install suggested Plugins" for a typical configuration.

4. Create an admin user and click "Save and Finish."

OUTPUT


Bi-..'n .--. • Main.py from tkint • Research_02.pd Research_01.pdf .env jenkins.e
File Edit View
2024-09-22 11:47:14.095+0000 [id=48] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
2024-09-22 11:47:14.101+0000 [id=48] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2024-09-22 11:47:14.363+0000 [id=68] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2024-09-22 11:47:14.367+0000 [id=68] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2024-09-22 11:47:14.421+0000 [id=50] INFO jenkins.install.SetupWizard#init:
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
915dac1472d14e30bf9782628b977621
This may also be found at: C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
2024-09-22 11:47:20.735+0000 [id=50] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2024-09-22 11:47:20.815+0000 [id=42] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Ln 30, Col 33 32 of 3,474 characters 100% Windows (CRLF) UTF-8


localhost:8080
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.462.2



Git installations

Git

Name

Default

Path to Git executable

C:\Program Files\git.exe

There's no such file: C:\Program Files\git.exe

Install automatically

Add Git

Save Apply

Gradle installations

Add Gradle

Gradle

name

Gradle

Install automatically

Install from Gradle.org

Version

Gradle 8.10.2-milestone-1

Add Installer

Save Apply

Ant installations

Add Ant

Ant

Name

Ant

Install automatically

Install from Apache

Version

1.10.15

Add Installer

Save Apply

Maven installations

Add Maven

Maven

Name

Maven

Install automatically

Install from Apache

Version

3.9.9

Add Installer

Save Apply

Dashboard > All > New Item

New Item

Enter an item name

Exp4

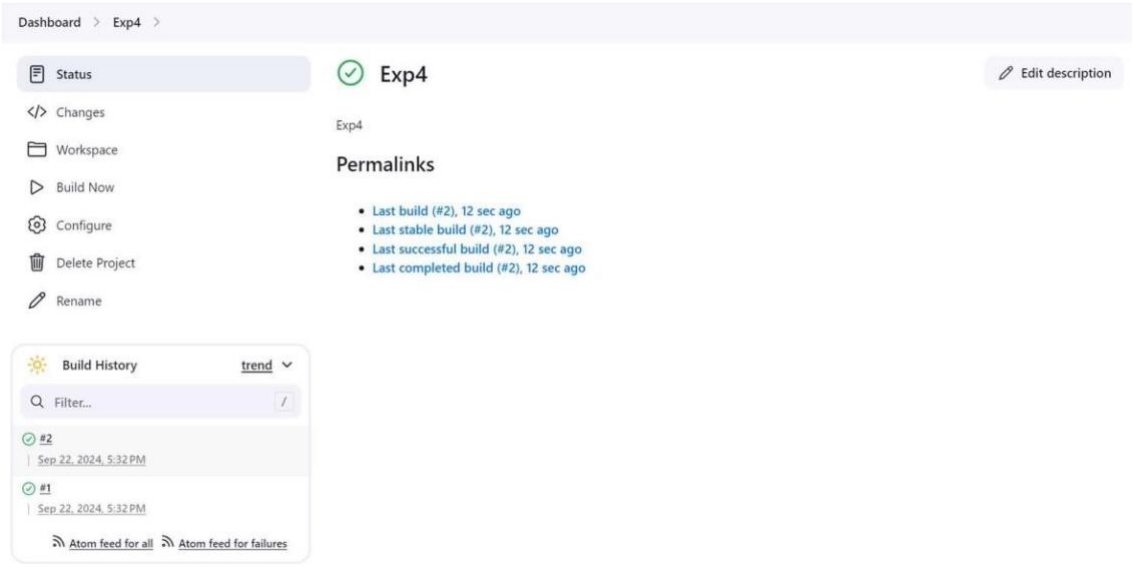
Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK



CONCLUSION

As a result, we understood Continuous Integration by installing and configuring Jenkins with Maven/Ant/Gradle, setting up build jobs to streamline the integration process.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT : 5

Date of Performance	
Date of Submission	

AIM

To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server

PROBLEM DEFINITION

Develop a pipeline of jobs using Maven, Gradle, or Ant in Jenkins and create a script to test and deploy an application on a Tomcat server.

THEORY

Maven

Maven is a powerful project management tool based on the POM (Project Object Model). It is used for building projects, managing dependencies, and documentation. Maven simplifies the build process similar to ANT but is more advanced. In short, Maven is a tool used for building and managing Java-based projects, making the day-to-day work of Java developers easier and improving comprehension of Java-based projects.

What Maven Does:

1. Build projects easily using Maven.
2. Add JARs and other dependencies to the project with ease.
3. Provide project information (log documents, dependency list, unit test reports, etc.).
4. Help update the central repository of JARs and other dependencies.
5. Build various projects into output types like JAR, WAR, etc., without scripting.
6. Integrate projects with source control systems (such as Subversion or Git).

Pipeline

Jenkins Pipeline, or simply 'Pipeline,' is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins. A continuous delivery pipeline is an automated process for delivering software from version control to users and customers. Jenkins Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines as code. The definition of a Jenkins pipeline is typically written into a text file called a Jenkinsfile, which is checked into a project's source control repository.

Both Declarative and Scripted Pipelines are DSLs used to describe parts of your software delivery pipeline. While standard Jenkins 'Freestyle' jobs support simple Continuous Integration by defining sequential tasks in an application lifecycle, they do not create a persistent record of execution, enable one script to address all steps in a complex workflow, or offer the advantages of a pipeline.

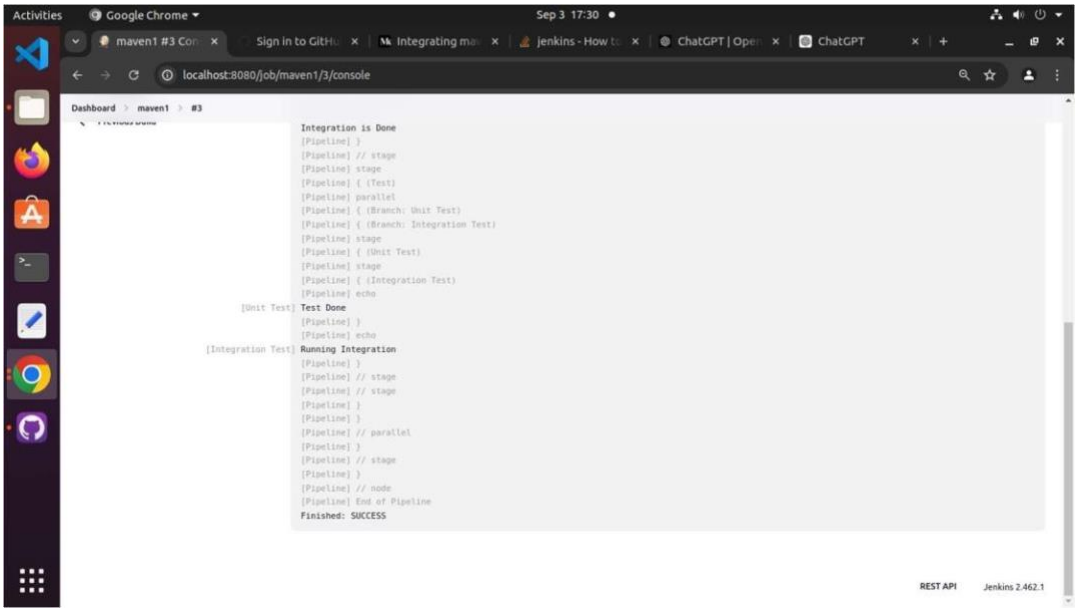
Pipeline Functionality:

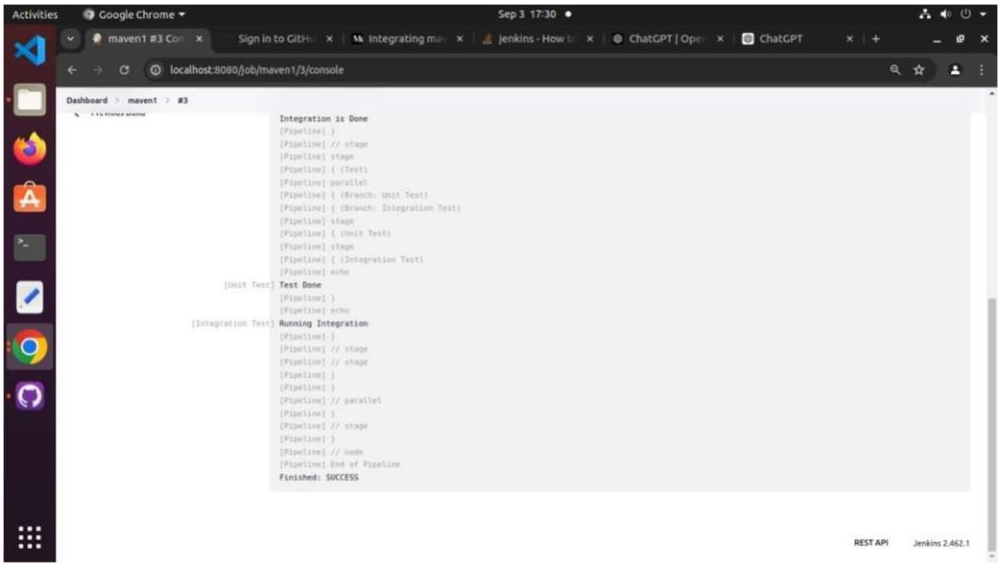
1. **Durable:** Pipelines can survive both planned and unplanned restarts of Jenkins.
2. **Pausable:** Pipelines can optionally pause and wait for human input or approval before completing jobs.
3. **Efficient:** Pipelines support and can restart from various saved checkpoints.

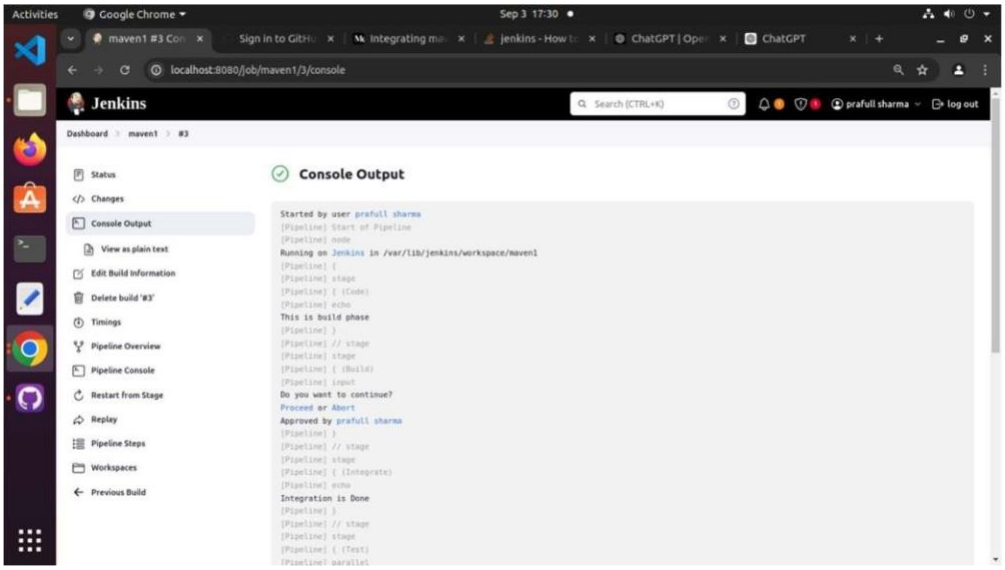
OUTPUT

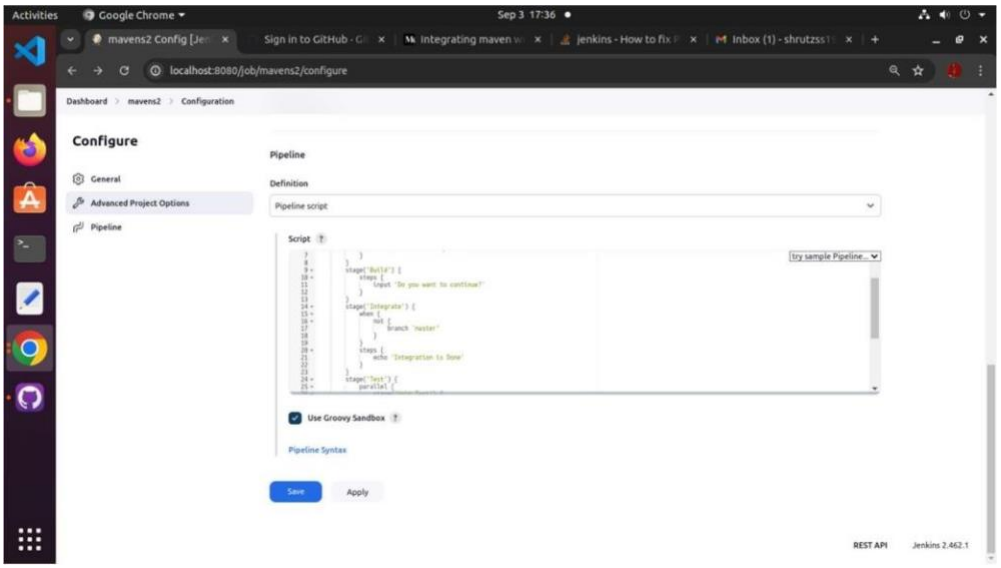
```
pipeline {
    agent any
    stages {
        stage('Code') {
            steps {
                echo 'This is build phase'
            }
        }
        stage('Build') {
            steps {
                input 'Do you want to continue?'
            }
        }
        stage('Integrate') {
            when {
                not {
                    branch 'master'
                }
            }
            steps {
                echo 'Integration is Done'
```

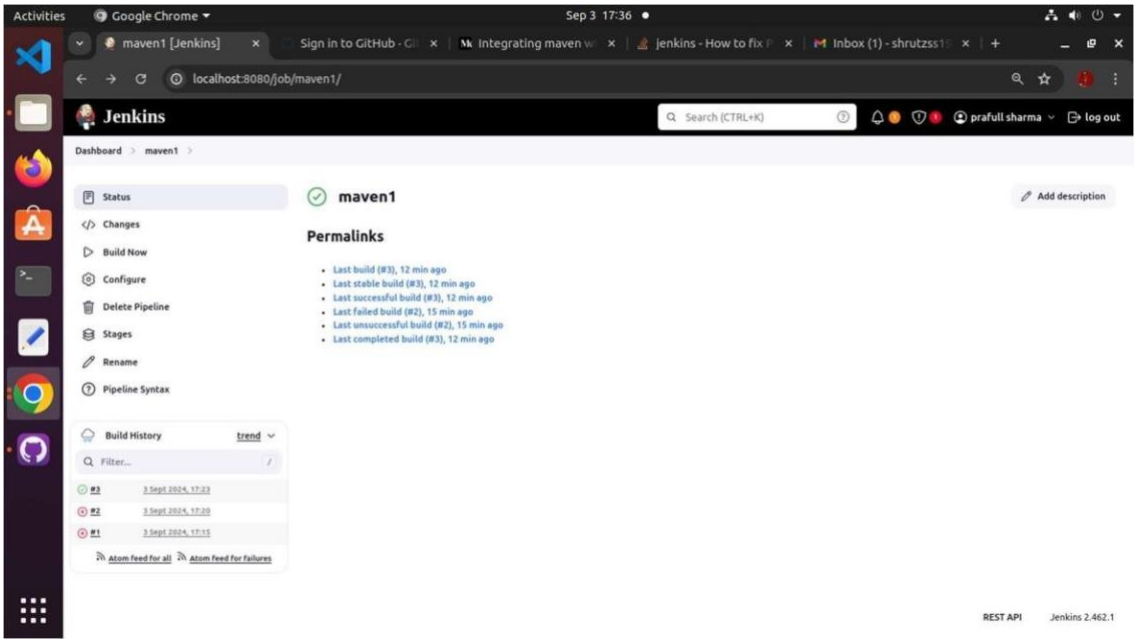
```
        }
      }
      stage('Test') {
        parallel {
          stage('Unit Test') {
            steps {
              echo 'Test Done'
            }
          }
          stage('Integration Test') {
            steps {
              echo 'Running Integration'
            }
          }
        }
      }
    }
}
```




Activities
Google Chrome
Sep 3 17:30
localhost:8080/job/maven1/3/console
Dashboard > maven1 > #3
Integration is Done
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] parallel
[Pipeline] { (Branch: Unit Test)
[Pipeline] { (Branch: Integration Test)
[Pipeline] stage
[Pipeline] { (Unit Test)
[Pipeline] stage
[Pipeline] { (Integration Test)
[Pipeline] echo
[Unit Test] Test Done
[Pipeline] }
[Pipeline] echo
[Integration Test] Running Integration
[Pipeline] }
[Pipeline] // stage
[Pipeline] // stage
[Pipeline] }
[Pipeline] }
[Pipeline] // parallel
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
REST API
Jenkins 2.462.1


Activities
Google Chrome
Sep 3 17:30
localhost:8080/job/maven1/3/console
Jenkins
Search (CTRL+K)
prafull sharma
log out
Dashboard > maven1 > #3
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#3'
Timings
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
Console Output
Started by user prafull sharma
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/maven1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
This is Build phase
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] input
Do you want to continue?
Proceed or Abort
Approved by prafull sharma
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integrate)
[Pipeline] echo
Integration is Done
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)


Activities
Google Chrome
Sep 3 17:36
localhost:8080/job/mavens2/configure
Dashboard > mavens2 > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
Use Groovy Sandbox
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.462.1



CONCLUSION

Consequently, we built pipelines of jobs using Maven/Gradle/Ant in Jenkins and created pipeline scripts for testing and deploying applications over Tomcat, enhancing automated deployment processes.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT No: 6

Date of Performance	
Date of Submission	

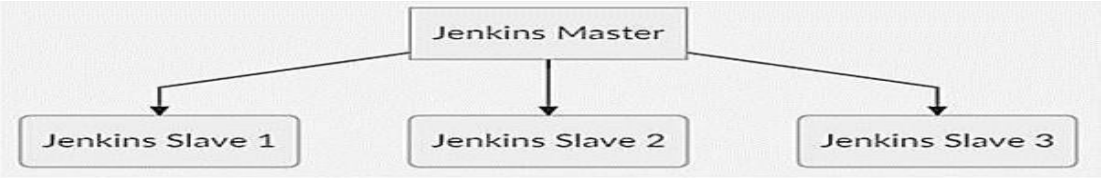
AIM

To understand Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

PROBLEM DEFINITION

Understand Jenkins Master-Slave architecture and expand a standalone Jenkins setup by adding slave nodes.

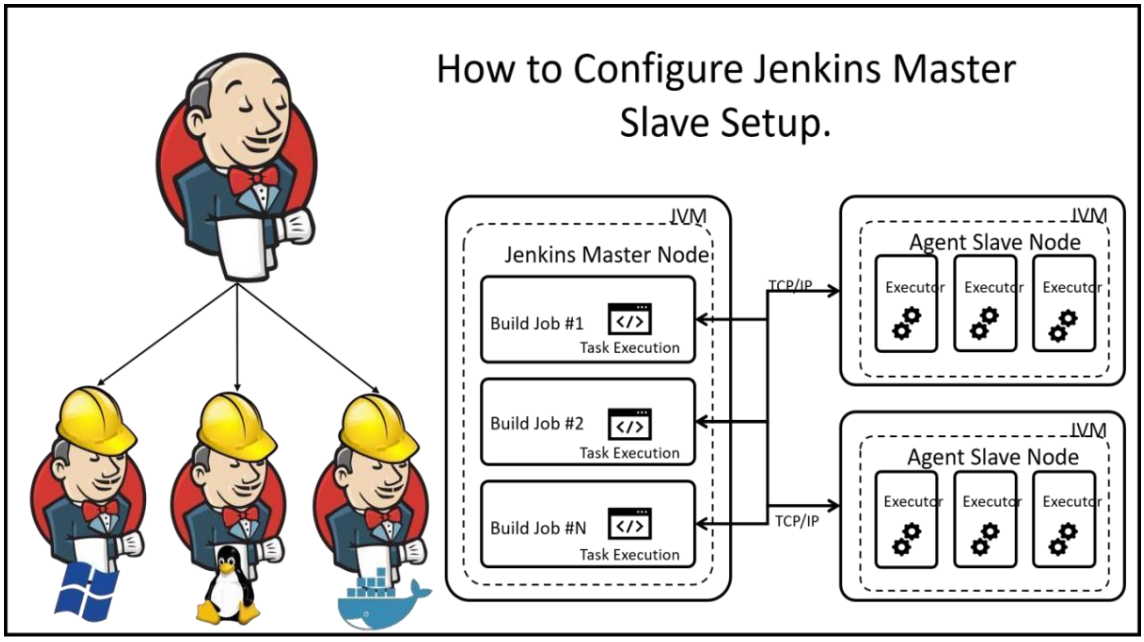
THEORY



Understanding the Master and Slave Architecture

A standalone Jenkins instance can quickly grow into a resource-intensive application. To manage this, Jenkins can be scaled using a slave node architecture, which offloads some responsibilities from the master Jenkins instance. A Jenkins slave node is a deviceconfigured to act as an automation executor on behalf of the master. The Jenkins master represents the base Jenkins installation, performing basic operations and serving the user interface, while the slaves handle the heavy lifting.

This distributed computing model allows the Jenkins master to remain responsive to users while offloading automation execution to connected slave(s). For example, a Jenkins master schedules jobs, assigns them to slaves, sends builds to slaves for execution, monitors slave states (online or offline), and displays build results.



Steps to Configure Jenkins Master and Slave Nodes:

1. Click on "Manage Jenkins" in the Jenkins dashboard.
2. Click on "Manage Nodes."
3. Select "New Node" and enter the node name in the "Node Name" field.
4. Select "Permanent Agent" and click "OK." Initially, only "Permanent Agent" will be available. After adding one or more slaves, the "Copy Existing Node" option will appear.
5. Enter the required information.
6. Enter the Hostname in the "Host" field.
7. Click the "Add" button to add credentials and click "Jenkins."
8. Enter Username, Password, ID, and Description.
9. Select the dropdown menu to add credentials in the "Credentials" field.
10. Select the dropdown to add the Host Key Verification Strategy under "Non-verifying Verification Strategy."
11. Select "Keep this agent online as much as possible" in the "Availability" field.

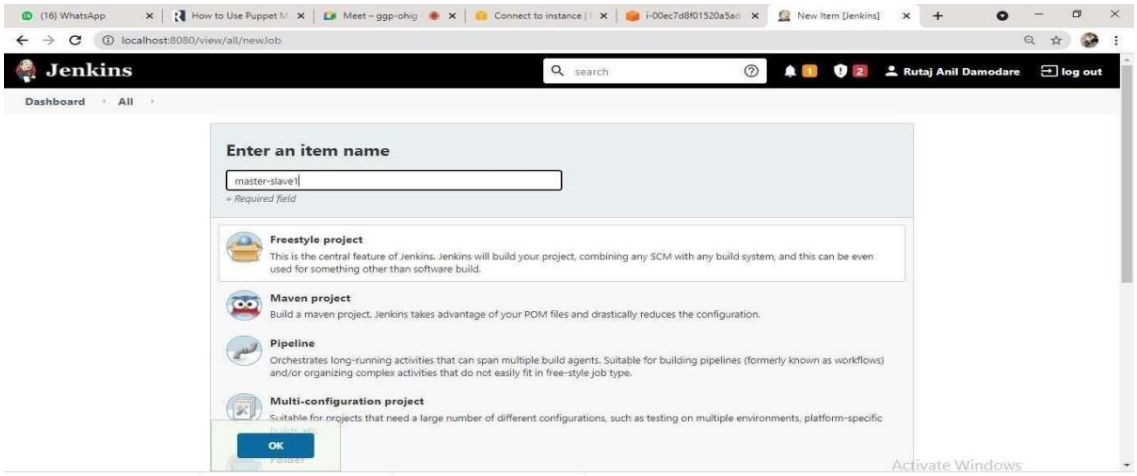
Creating a Freestyle Project and Running on the Slave Machine:

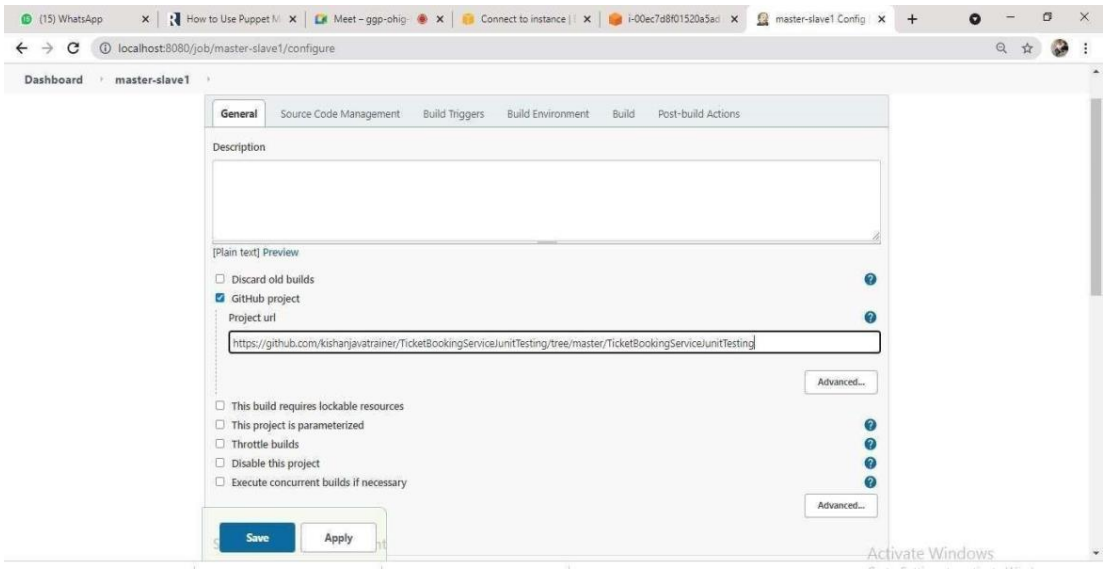
1. Click "Save," which will redirect to the job's view page.
2. On the left pane, click the "Build Now" button to execute your pipeline.
3. Verify the history of the executed build under "Build History" by clicking the build number.
4. Click on the build number and select "Console Output" to view the executed job and output on the remote host.

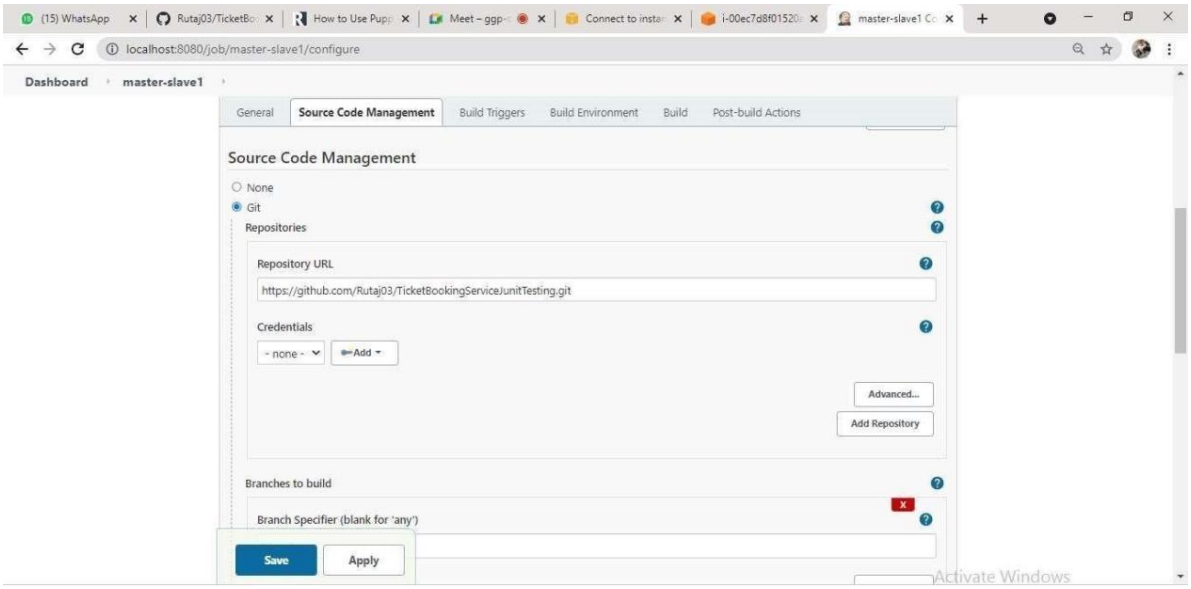
Creating a Pipeline and Running on the Slave Machine:

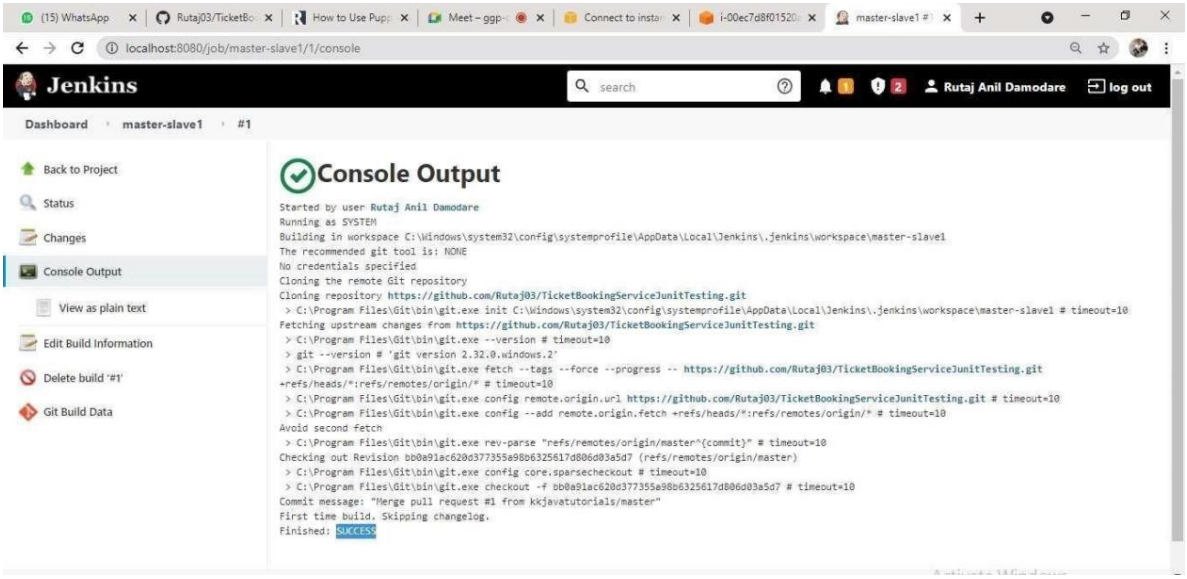
1. Click "New Item" in the top left corner on the dashboard.
2. Enter the name of your project in the "Enter an item name" field, select the "Pipeline" project, and click "OK."
3. Enter a description (optional).
4. Go to the "Pipeline" section, ensure the "Definition" field is set to "Pipeline script."
5. Copy and paste the declarative Pipeline script into the script field.
6. Click "Save," which will redirect to the Pipeline view page.
7. On the left pane, click the "Build Now" button to execute your pipeline.
8. After execution, the Pipeline view will display the results.
9. Verify the history of executed builds under "Build History" by clicking the build number.
10. Click on the build number and select "Console Output" to see that the pipeline ran on a slave machine.

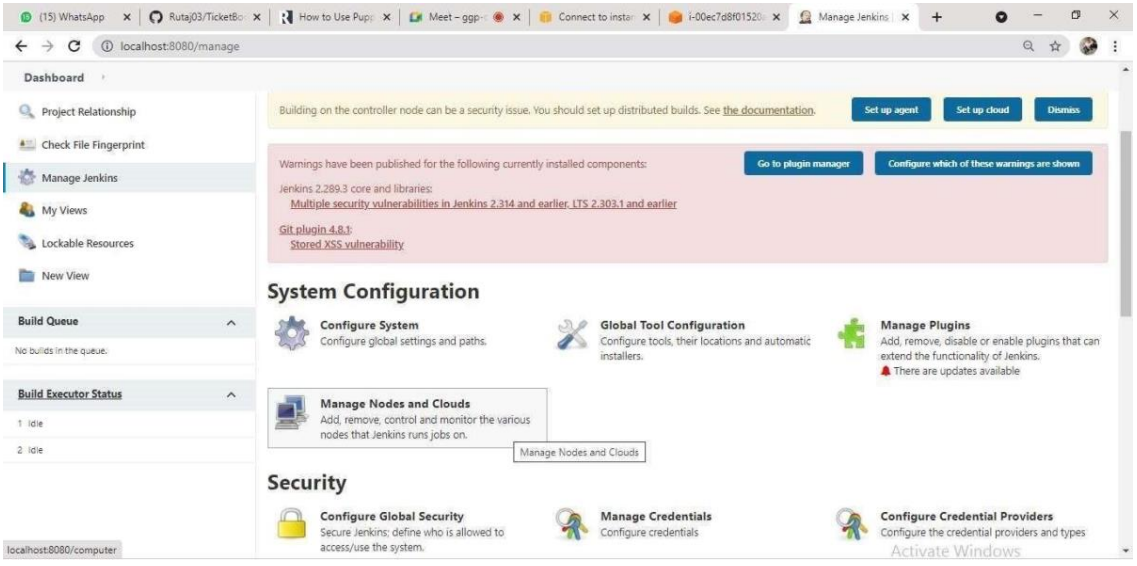
OUTPUT

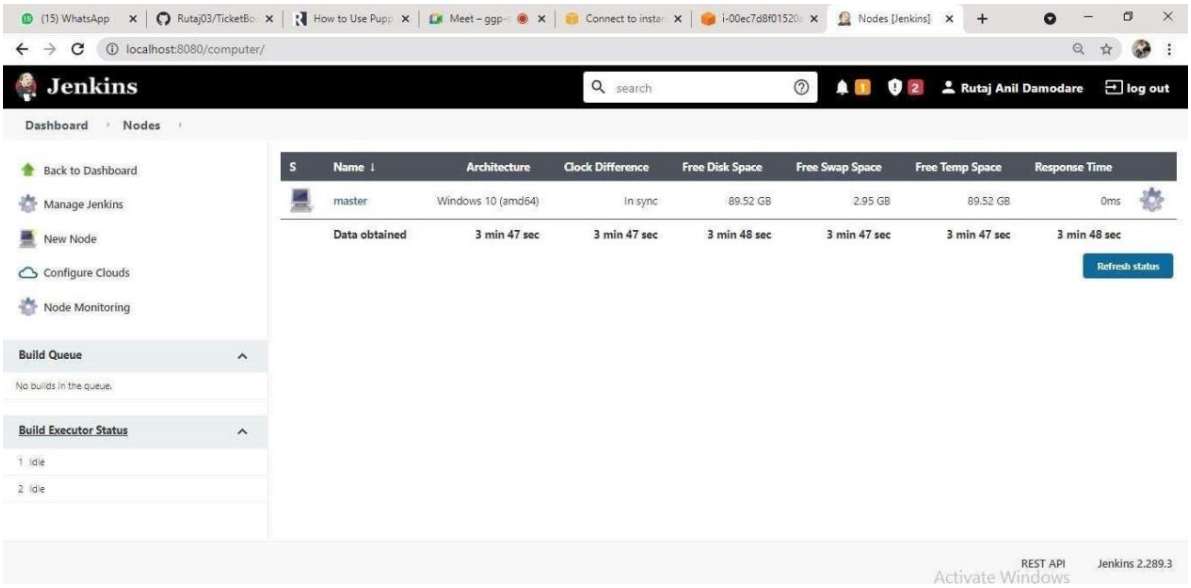


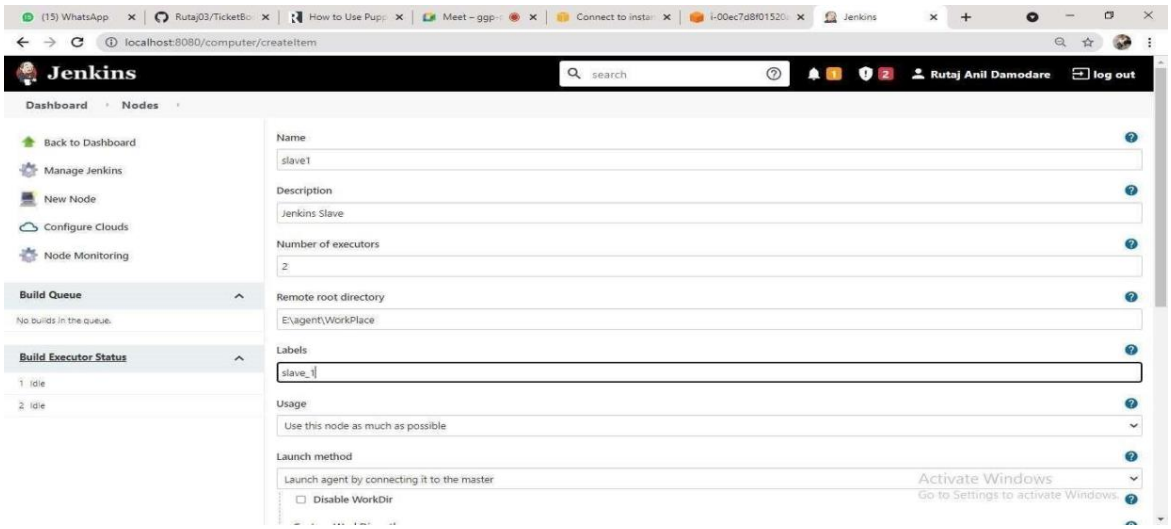


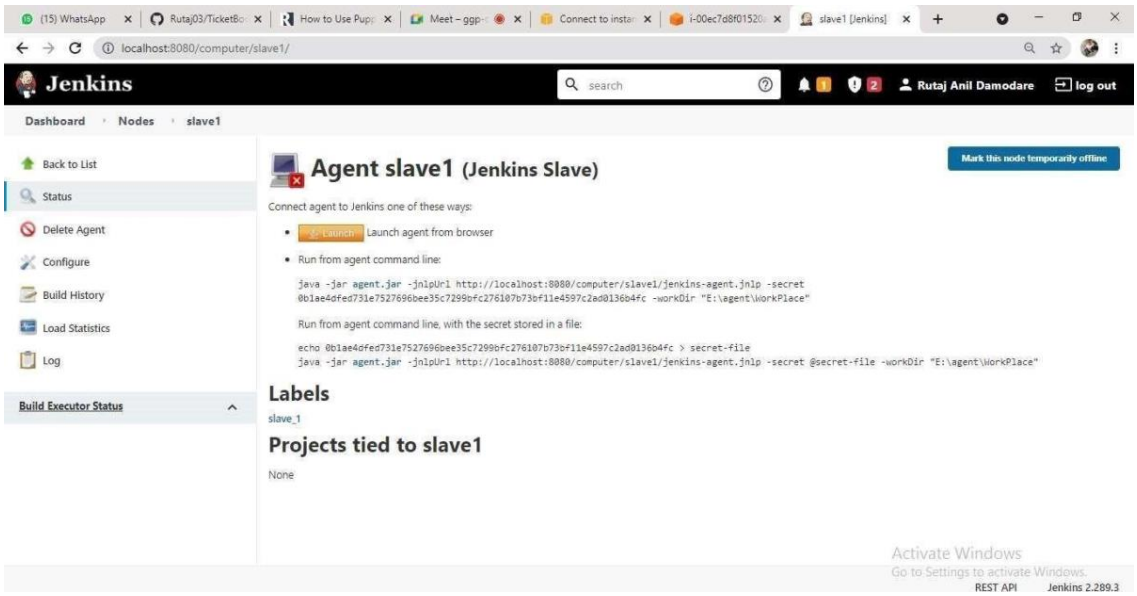

localhost:8080/job/master-slave1/configure
Dashboard
master-slave1
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Rutaj03/TicketBookingServiceJunitTesting.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
Save
Apply


localhost:8080/job/master-slave1/1/console
Jenkins
search
Rutaj Anil Damodare
log out
Dashboard
master-slave1
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Git Build Data
Console Output
Started by user Rutaj Anil Damodare
Running as SYSTEM
Building in workspace C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\master-slave1
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Rutaj03/TicketBookingServiceJunitTesting.git
> C:\Program Files\Git\bin\git.exe init C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\master-slave1 # timeout=10
Fetching upstream changes from https://github.com/Rutaj03/TicketBookingServiceJunitTesting.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.32.0.windows.2'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/Rutaj03/TicketBookingServiceJunitTesting.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/Rutaj03/TicketBookingServiceJunitTesting.git # timeout=10
> C:\Program Files\Git\bin\git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision bb0a91ac620d377355a98b6325617d806d03a5d7 (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f bb0a91ac620d377355a98b6325617d806d03a5d7 # timeout=10
Commit message: "Merge pull request #1 from kkjavatutorials/master"
First time build. Skipping changelog.
Finished: SUCCESS


localhost:8080/manage
Dashboard
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Building on the controller node can be a security issue. You should set up distributed builds. See the documentation.
Set up agent
Set up cloud
Dismiss
Warnings have been published for the following currently installed components:
Go to plugin manager
Configure which of these warnings are shown
Jenkins 2.289.3 core and libraries:
Multiple security vulnerabilities in Jenkins 2.314 and earlier, LTS 2.303.1 and earlier
Git plugin 4.8.1:
Stored XSS vulnerability
System Configuration
Configure System
Configure global settings and paths.
Global Tool Configuration
Configure tools, their locations and automatic installers.
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
There are updates available
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Manage Nodes and Clouds
Security
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.
Manage Credentials
Configure credentials
Configure Credential Providers
Configure the credential providers and types
localhost:8080/computer


Jenkins
search
Rutaj Anil Damodare
log out
Dashboard
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
S
Name ↓
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
master
Windows 10 (amd64)
In sync
89.52 GB
2.95 GB
89.52 GB
0ms
Data obtained
3 min 47 sec
3 min 47 sec
3 min 48 sec
3 min 47 sec
3 min 47 sec
3 min 48 sec
Refresh status
REST API
Jenkins 2.289.3
Activate Windows

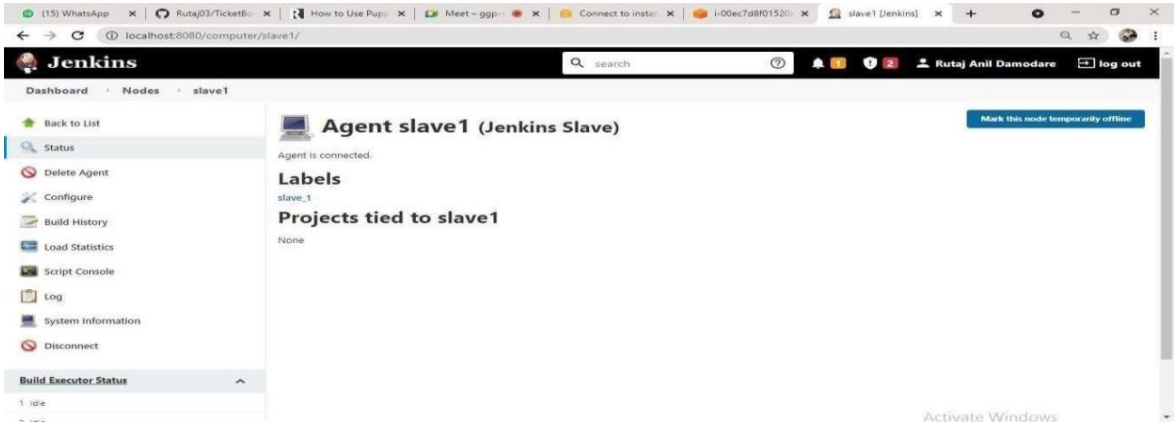

Jenkins
search
Rutaj Anil Damodare
log out
Dashboard
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Name
slave1
Description
Jenkins Slave
Number of executors
2
Remote root directory
E:\agent\WorkPlace
Labels
slave_1
Usage
Use this node as much as possible
Launch method
Launch agent by connecting it to the master
Disable WorkDir
Activate Windows
Go to Settings to activate Windows.

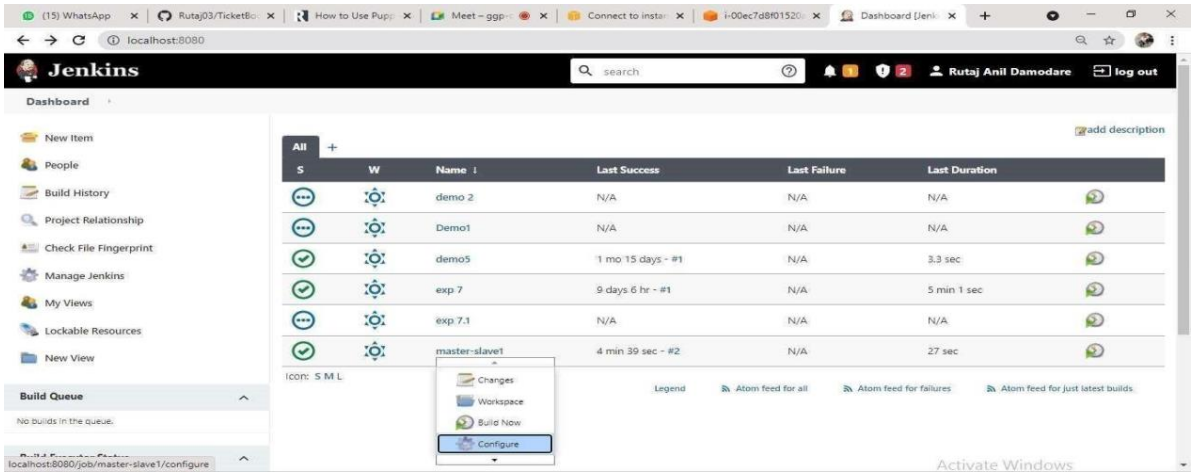

Jenkins
search
Rutaj Anil Damodare
log out
Dashboard
Nodes
slave1
Back to List
Status
Delete Agent
Configure
Build History
Load Statistics
Log
Build Executor Status
Agent slave1 (Jenkins Slave)
Mark this node temporarily offline
Connect agent to Jenkins one of these ways:
Launch
Launch agent from browser
Run from agent command line:
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/slave1/jenkins-agent.jnlp -secret
0b1ae4dfed731e7527696bee35c7299bfc276107b73bf11e4597c2ad0136b4fc -workDir "E:\agent\WorkPlace"
Run from agent command line, with the secret stored in a file:
echo 0b1ae4dfed731e7527696bee35c7299bfc276107b73bf11e4597c2ad0136b4fc > secret-file
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/slave1/jenkins-agent.jnlp -secret @secret-file -workDir "E:\agent\WorkPlace"
Labels
slave_1
Projects tied to slave1
None
Activate Windows
Go to Settings to activate Windows.
REST API
Jenkins 2.289.3

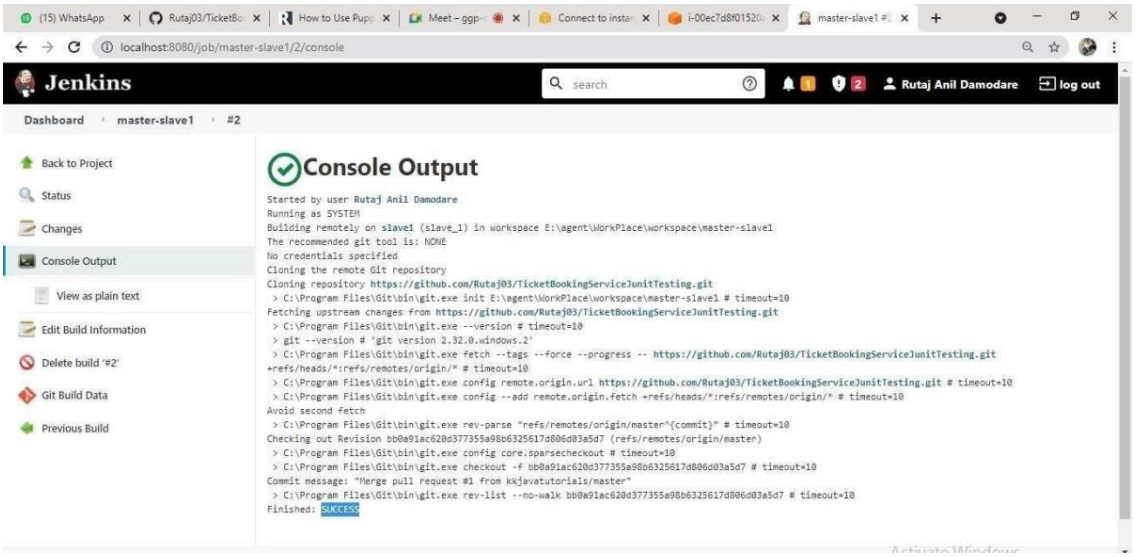
Untitled - Notepad

File Edit Format View Help

```
java -jar E:\agent\agent.jar -jnlpUrl http://localhost:8080/computer/slave1/jenkins-agent.jnlp -secret
0b1ae4dfed731e7527696bee35c7299bfc276107b73bf11e4597c2ad0136b4fc -workDir "E:\agent\WorkPlace"
```







CONCLUSION

Thus, understanding Jenkins Master-Slave architecture and implementing slave nodesallowed us to scale Jenkins installations and improve continuous integration efficiency.

R1 (5 marks)	**R2 (5 marks)**	**R3 (5 marks)**	**TOTAL (5 marks)**	**SIGN (5 marks)**

EXPERIMENT No: 7

Date of Performance	
Date of Submission	

AIM

To Setup and Run Selenium Tests in Jenkins Using Maven.

PROBLEM DEFINITION

Configure and execute Selenium tests in Jenkins using Maven.

THEORY

Jenkins is an open-source automation tool created with Java. It is extensively used as a Continuous Integration (CI) and Continuous Deployment (CD) tool.

Maven:

- Maven primarily provides developers with:
    1. A comprehensive, reusable, and easily maintainable model for projects.
    2. Plugins or tools to interact with and operate within this model.
- Maven is a POM (Project Object Model)-based build automation and project management tool written in Java. However, it is compatible with projects written in C#, Python, Ruby, etc.

A few Mavens features worth mentioning are:

1. Maven can be used to build projects into predefined output types like .jar, .war, metadata, etc.
2. Maven can automatically download necessary files from the repository when building a project.

Selenium Using Maven in Jenkins

Selenium is a widely used test automation framework for validating web applications across different combinations of browsers, platforms, and devices (or emulators). It is extensively used for testing areas such as functional testing, end-to-end testing, and more.

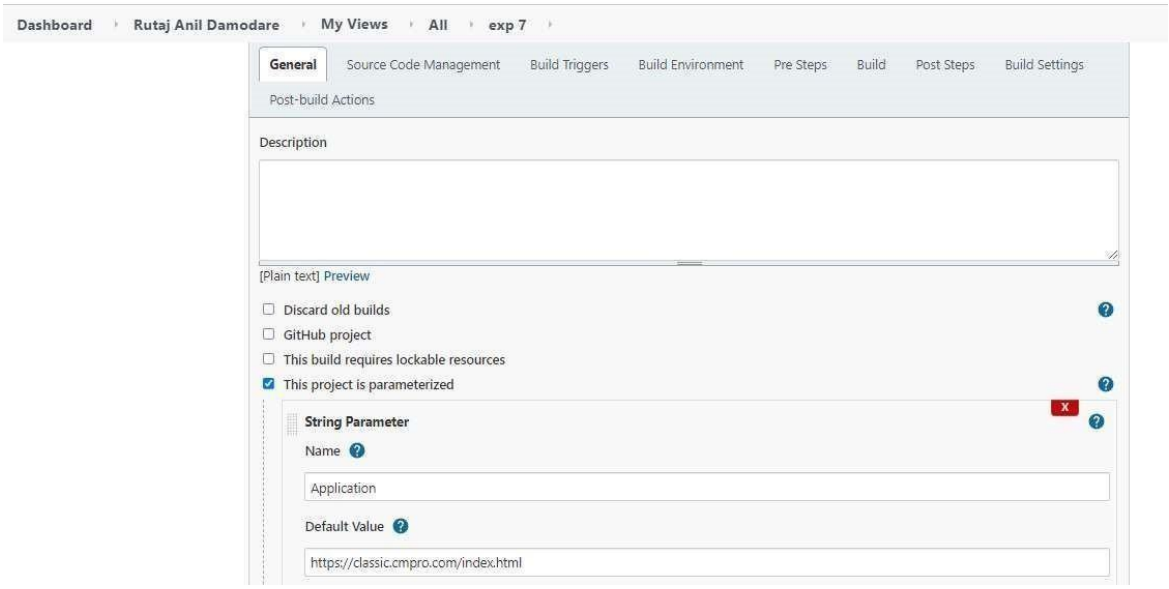
Why Jenkins and Selenium?

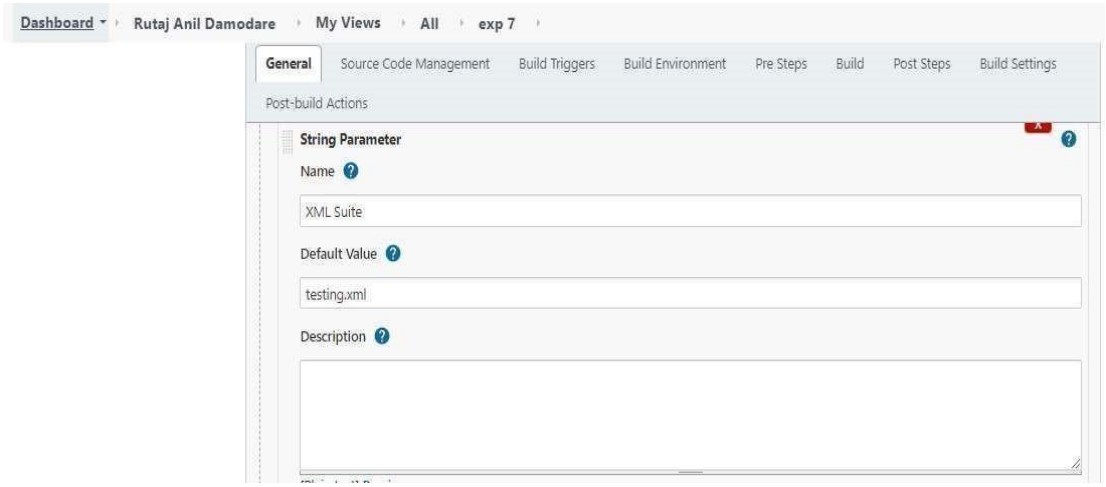
- Running Selenium tests in Jenkins allows you to execute your tests every time your software changes and deploy the software to new environments when the tests pass.

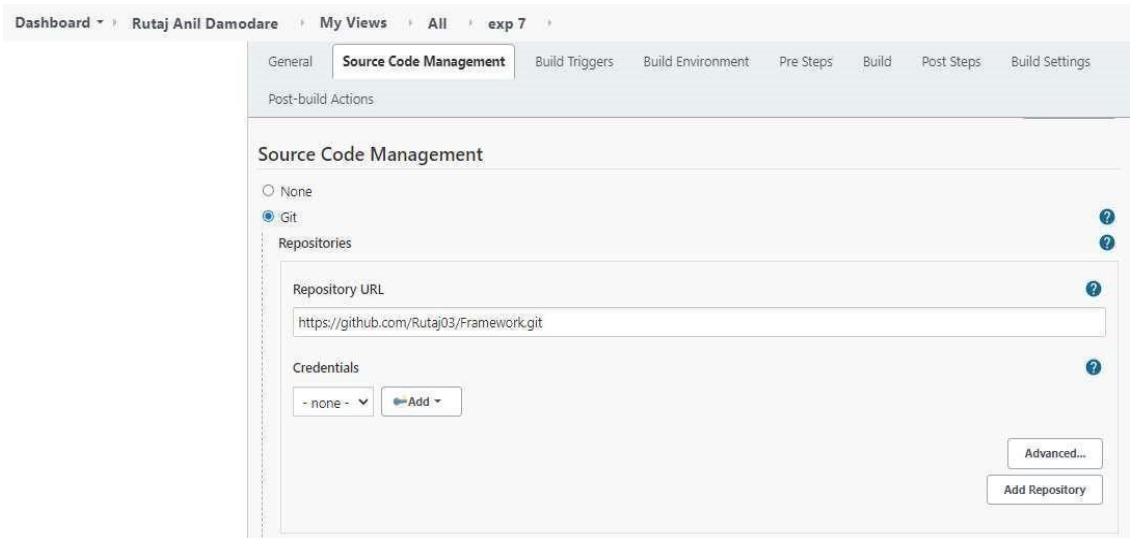
Advantages of Using Maven and Jenkins with Selenium:

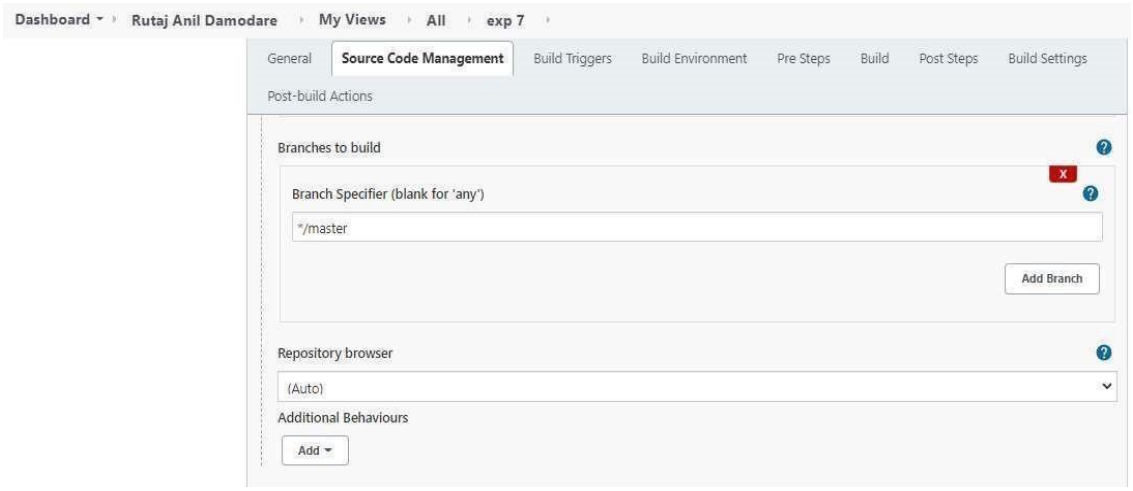
- Whenever a change is made in the implementation, the changes are deployed inthe test environment. Automation testing is performed continuously, and developers are kept informed about the build and test stage results.
- Test suites comprising many test scenarios might take a longer duration for testing. In such cases, a nightly build run can be scheduled for build and execution on the Jenkins server.

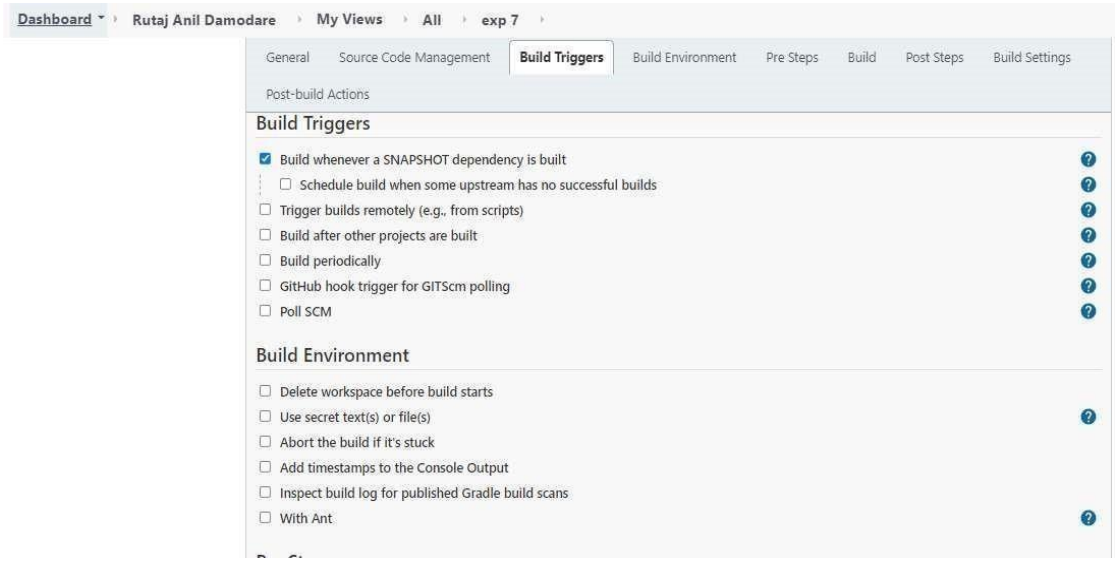
OUTPUT

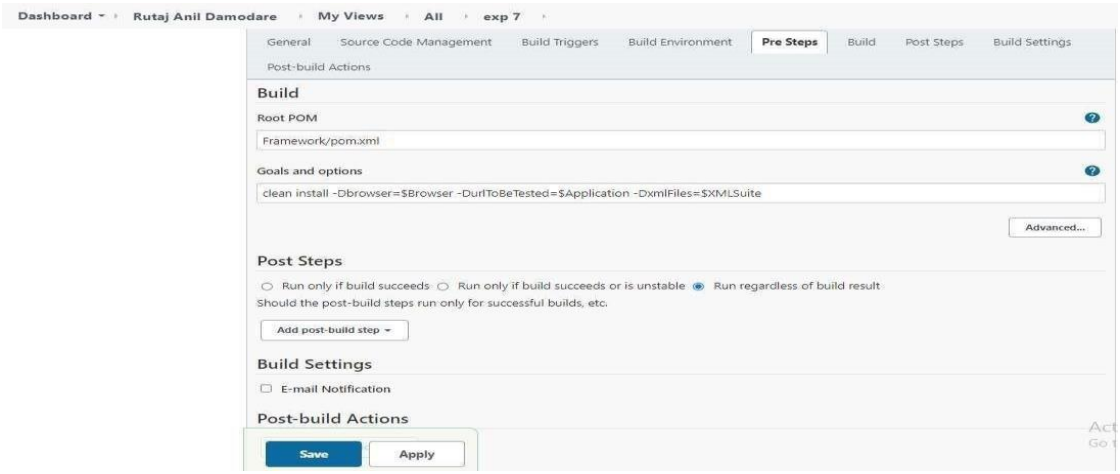


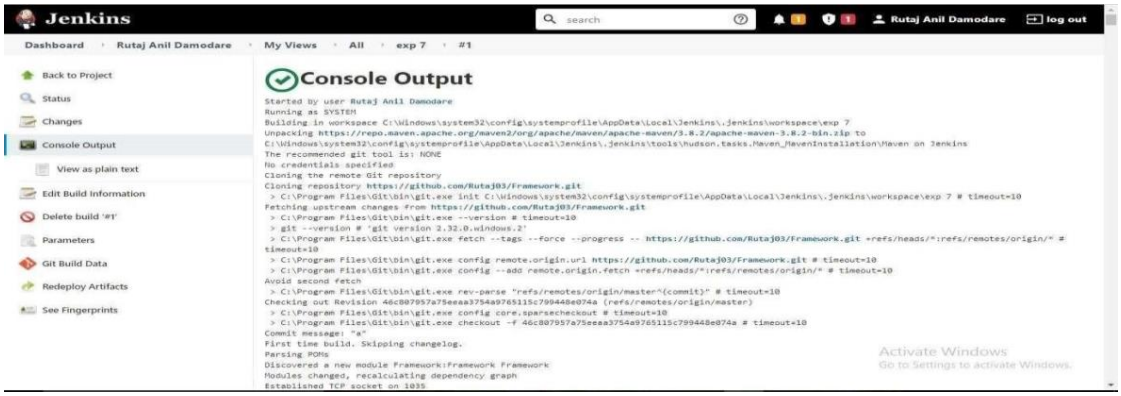


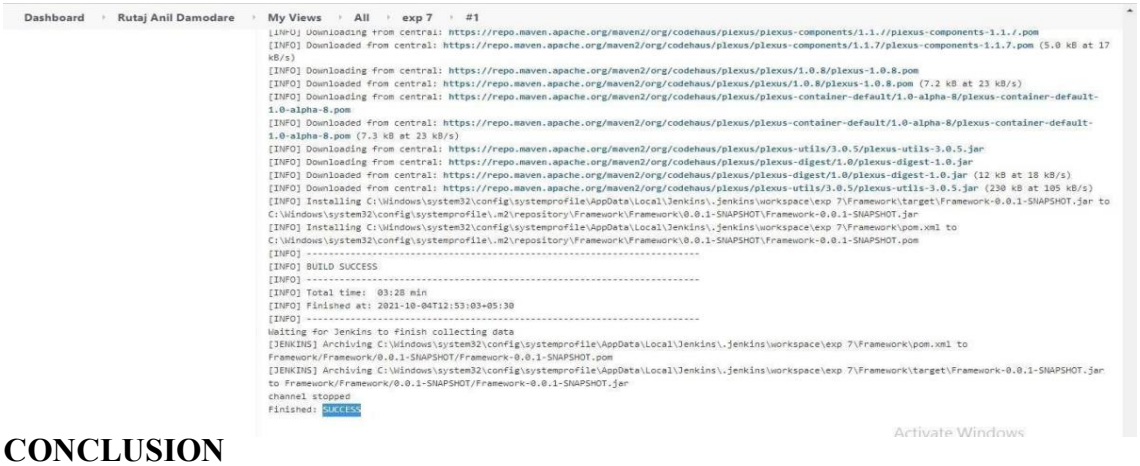

Dashboard
Rutaj Anil Damodare
My Views
All
exp 7
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Rutaj03/Framework.git
Credentials
- none -
Add
Advanced...
Add Repository


Dashboard
Rutaj Anil Damodare
My Views
All
exp 7
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add


Dashboard
Rutaj Anil Damodare
My Views
All
exp 7
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build Triggers
Build whenever a SNAPSHOT dependency is built
Schedule build when some upstream has no successful builds
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant







CONCLUSION

Therefore, setting up and running Selenium tests in Jenkins using Maven ensured thatweb applications were thoroughly tested as part of the continuous integration process.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT : 8

Date of Performance	
Date of Submission	

AIM

To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

PROBLEM DEFINITION

Gain insight into Docker architecture and container lifecycle, including installing Docker and managing images and containers.

THEORY

Jenkins is an open-source automation tool created with Java. It is extensively used as a Continuous Integration (CI) and Continuous Deployment (CD) tool.

Maven:

- Maven primarily provides developers with:
    1. A comprehensive, reusable, and easily maintainable model for projects.
    2. Plugins or tools to interact with and operate within this model.
- Maven is a POM (Project Object Model)-based build automation and project management tool written in Java. However, it is compatible with projects written in C#, Python, Ruby, etc.

A few Maven features worth mentioning are:

1. Maven can be used to build projects into predefined output types like .jar, .war, metadata, etc.
2. Maven can automatically download necessary files from the repository when building a project.

Selenium Using Maven in Jenkins

Selenium is a widely used test automation framework for validating web applications across different combinations of browsers, platforms, and devices (or emulators). It is extensively used for testing areas such as functional testing, end-to-end testing, and more.

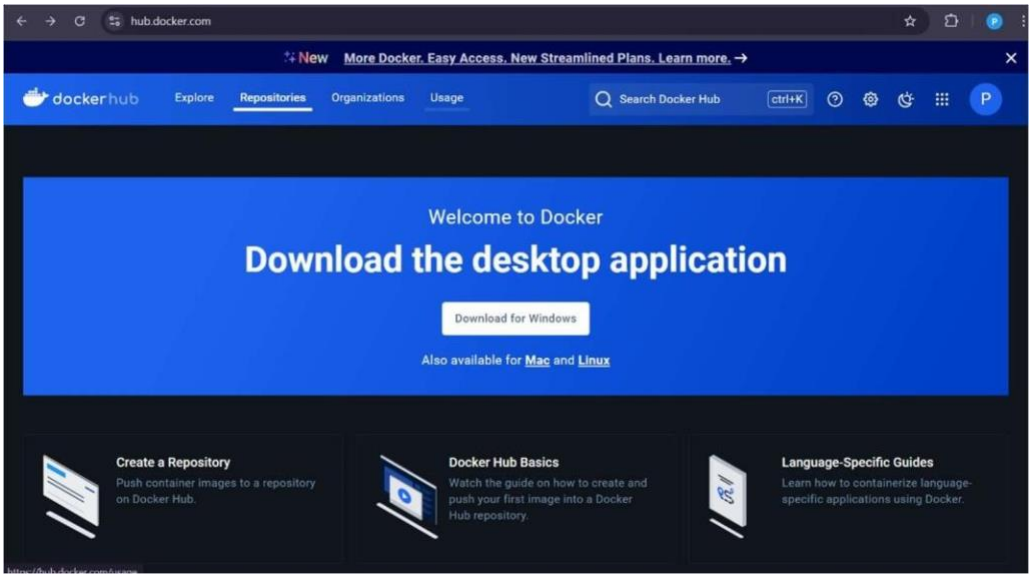
Why Jenkins and Selenium?

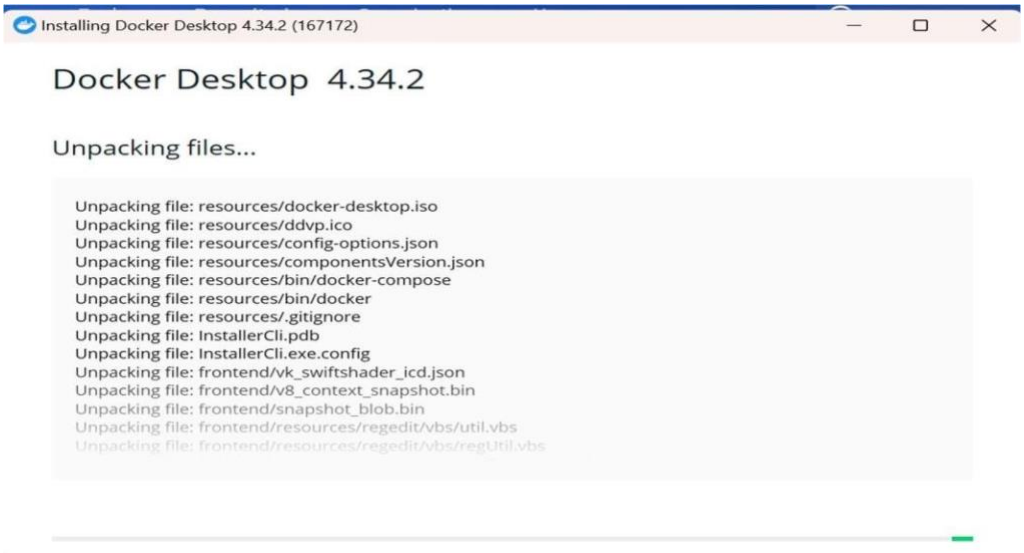
- Running Selenium tests in Jenkins allows you to execute your tests every time your software changes and deploy the software to new environments when the tests pass.

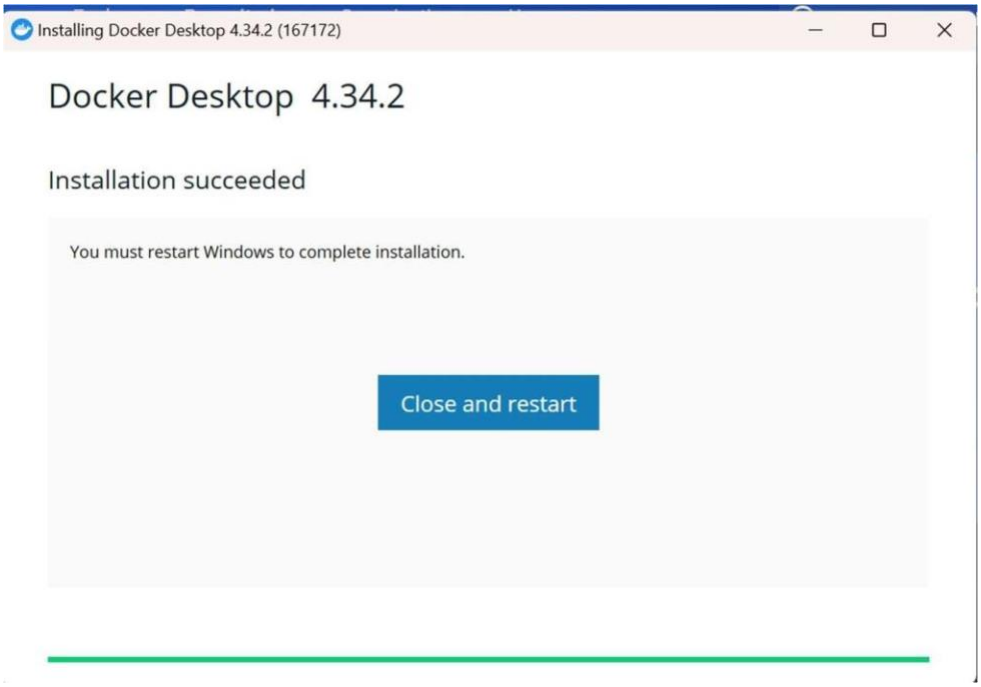
Advantages of Using Maven and Jenkins with Selenium:

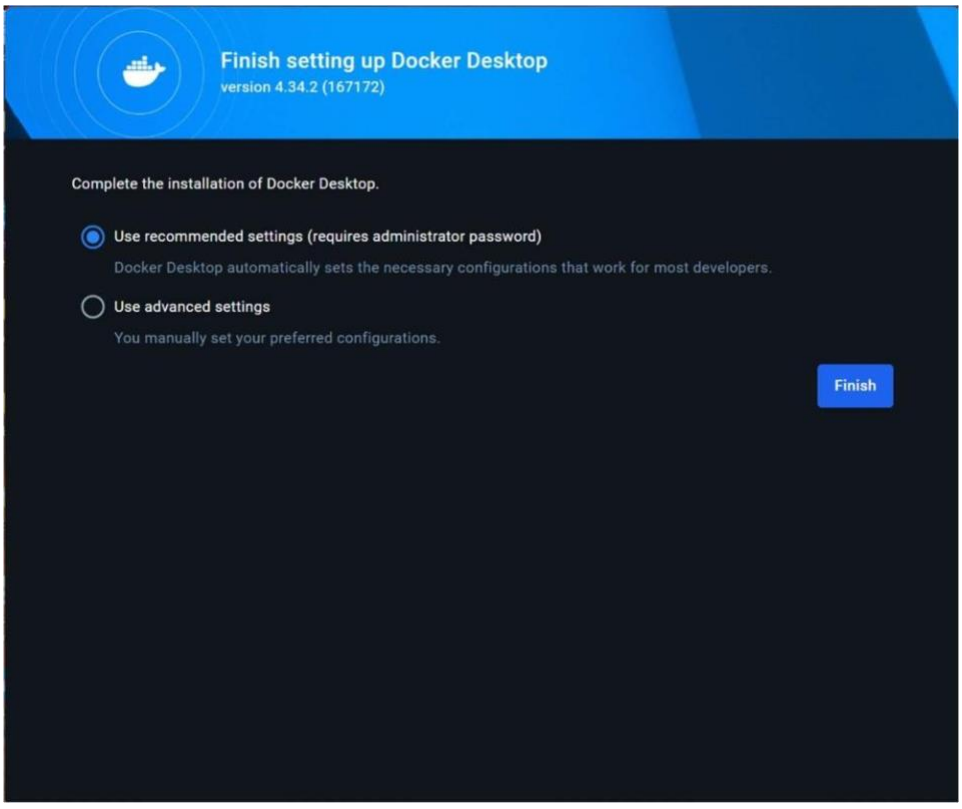
- Whenever a change is made in the implementation, the changes are deployed in the test environment. Automation testing is performed continuously, and developers are kept informed about the build and test stage results.
- Test suites comprising many test scenarios might take a longer duration for testing. In such cases, a nightly build run can be scheduled for build and execution on the Jenkins server.

OUTPUT

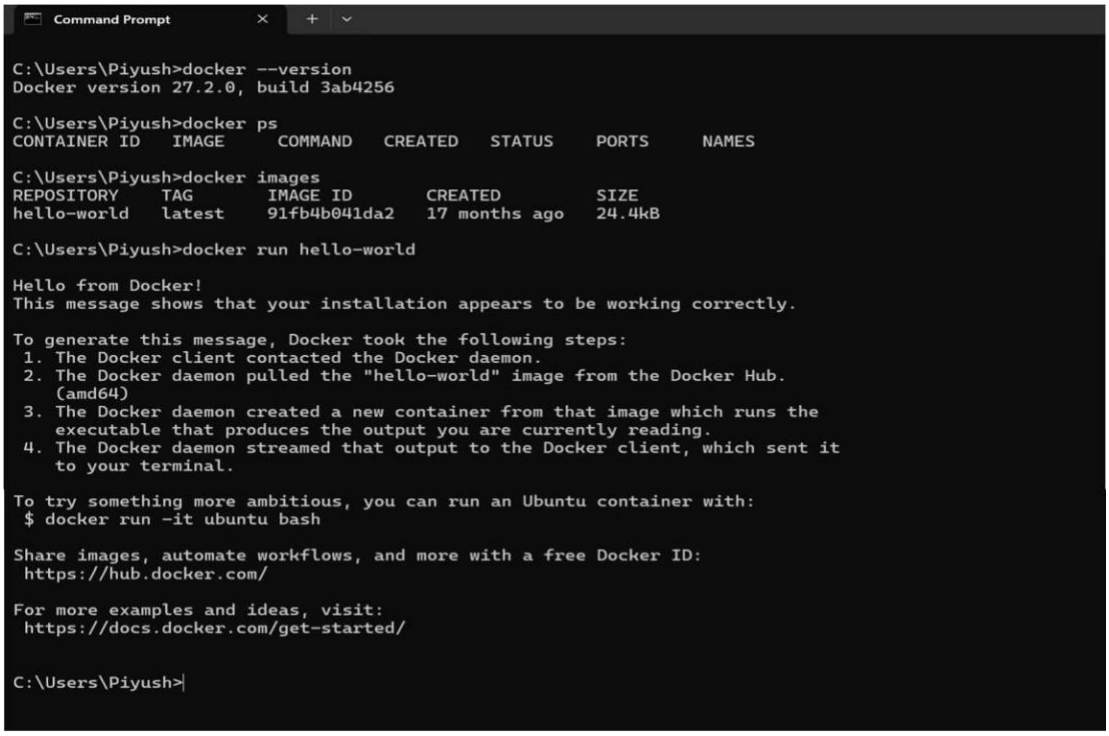




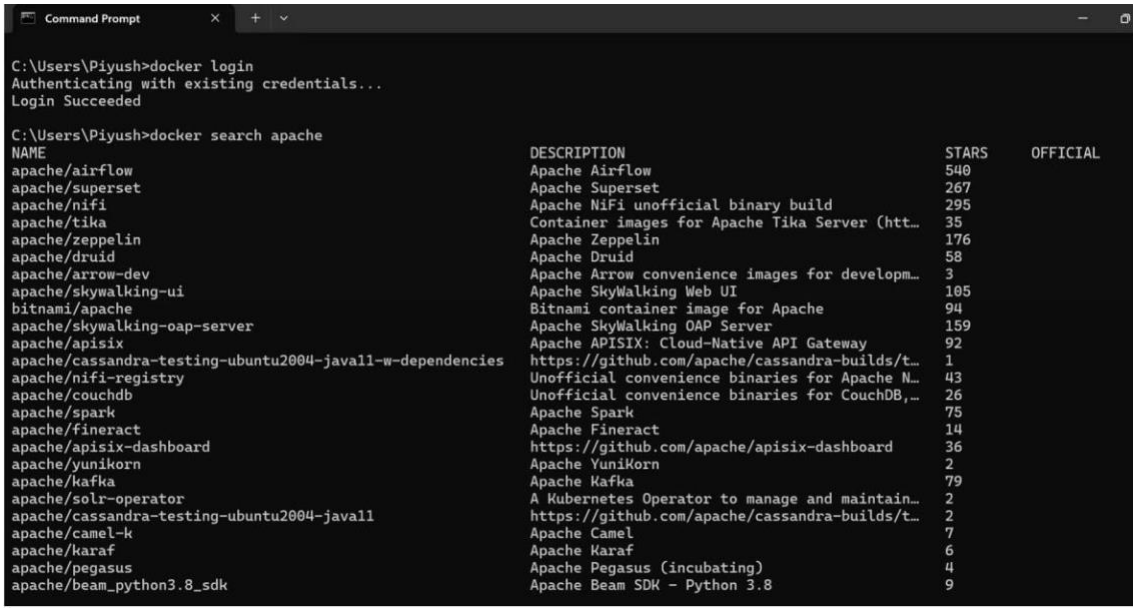

Installing Docker Desktop 4.34.2 (167172)
Docker Desktop 4.34.2
Installation succeeded
You must restart Windows to complete installation.
Close and restart


Finish setting up Docker Desktop
version 4.34.2 (167172)
Complete the installation of Docker Desktop.
Use recommended settings (requires administrator password)
Docker Desktop automatically sets the necessary configurations that work for most developers.
Use advanced settings
You manually set your preferred configurations.
Finish

Docker –version, docker ps, docker run



Docker login, docker search apache



Docker search redis

```
C:\Users\Piyush>docker search redis
NAME                             DESCRIPTION                                     STARS     OFFICIAL
redis                            Redis is the world's fastest data platform f…   13029     [OK]
redis/redis-stack-server         redis-stack-server installs a Redis server w…   83
redis/redis-stack                redis-stack installs a Redis server with add…   118
redis/redisinsight               Redis Insight - our best official GUI for Re…   17
bitnami/redis                    Bitnami container image for Redis               303
circleci/redis                   CircleCI images for Redis                       17
redislabs/redis                  Clustered in-memory database engine compatib…   42
cimg/redis                                                                       2
bitnamicharts/redis                                                              2
redis/rdi-monitor                                                                0
redis/rdi-api                                                                    0
ubuntu/redis                     Redis, an open source key-value store. Long-…   22
rapidfort/redis                  RapidFort optimized, hardened image for Redi…   21
webhippie/redis                  Docker image for redis                          11
redis/rdi-operator                                                               0
redis/rdi-processor                                                              0
elestio/redis                    Redis, verified and packaged by Elestio         1
jumpserver/redis                 Redis is an open source key-value store that…   1
redis/rdi-cli                                                                    0
redis/rdi-collector-initializer   Init container for RDI Collector               0
drud/redis                       redis                                           0
jelastic/redis                   An image of the Redis database server mainta…   0
datasense/redis                                                                  0
sameersbn/redis                                                                  84
wodby/redis                      Redis container image with orchestration        2
```

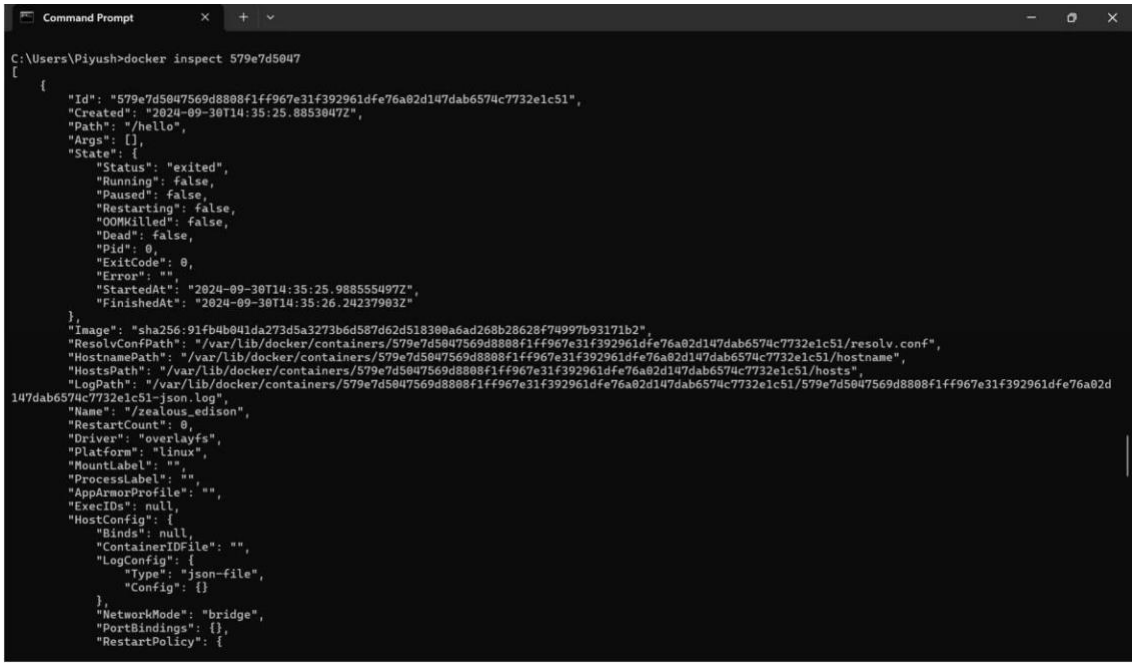
Docker stats

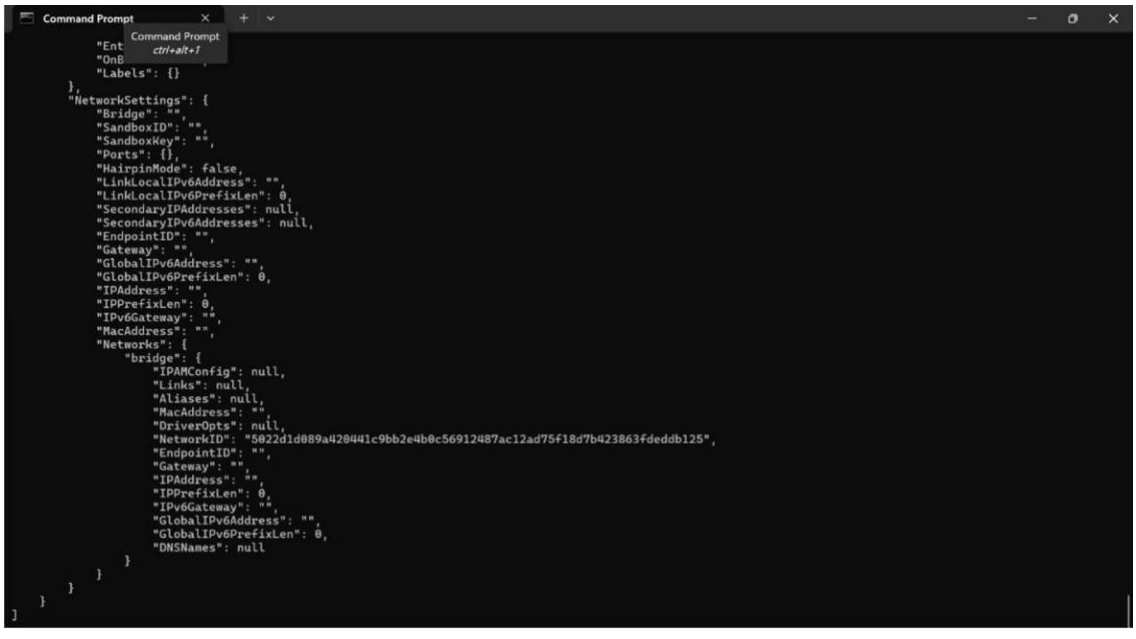
```
C:\Users\Piyush>docker stats
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O   BLOCK I/O   PIDS
```

Docker network ls

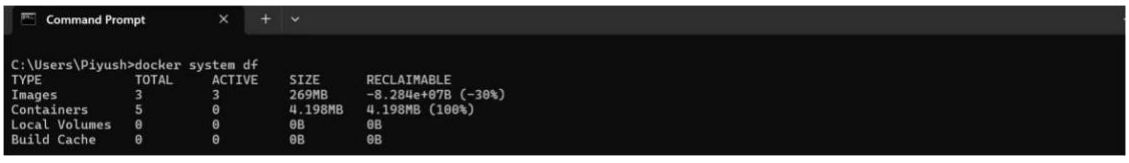
```
C:\Users\Piyush>docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5022d1d089a4   bridge    bridge    local
be13d7356d63   host      host      local
d5e0275a9668   none      null      local

C:\Users\Piyush>docker volume ls
DRIVER    VOLUME NAME
```

Docker inspect

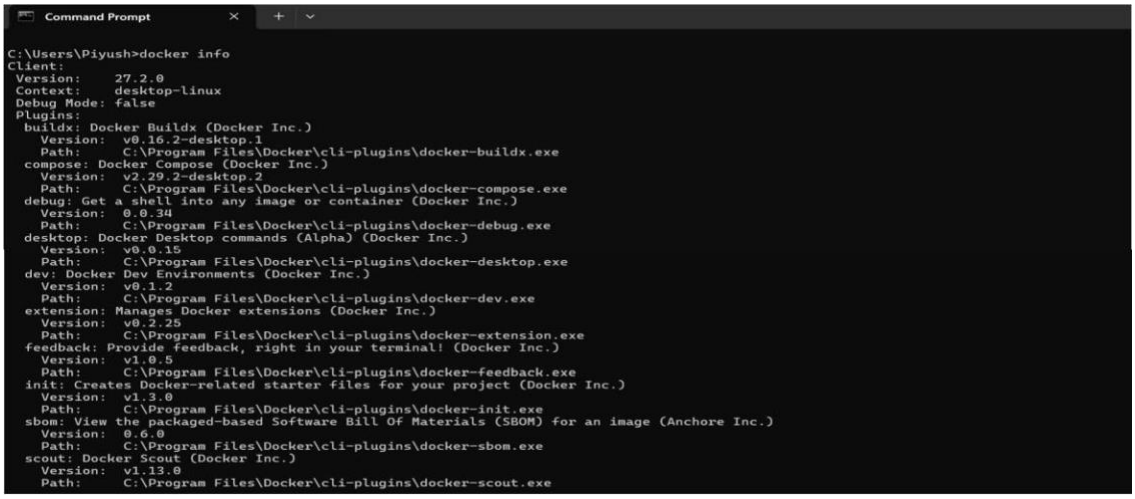


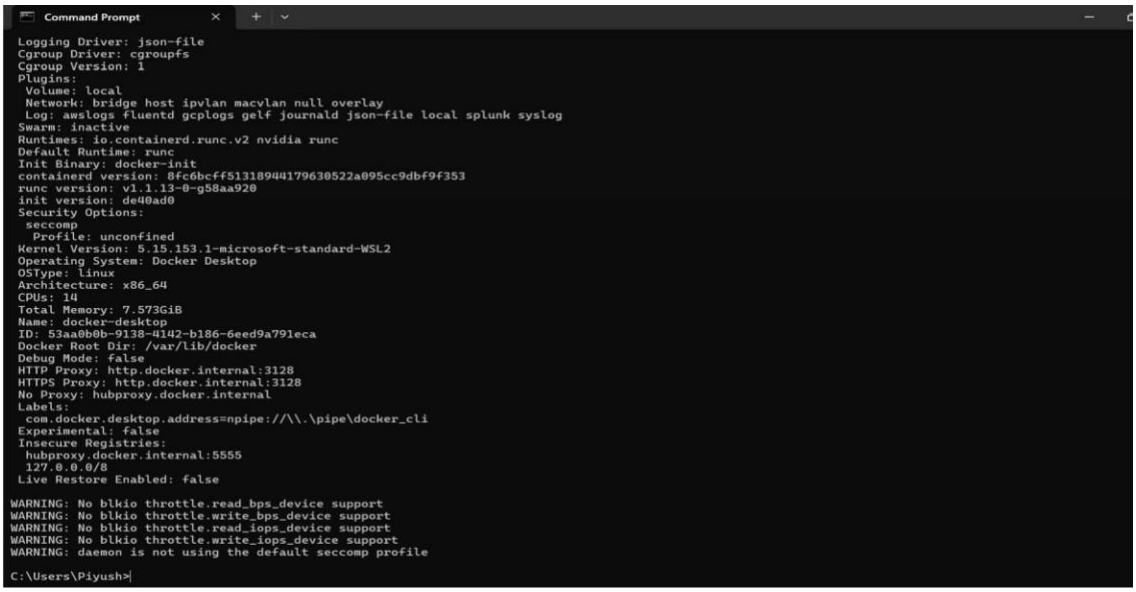


Docker system df



Docker info





CONCLUSION

As a result, we understood Docker architecture and container life cycle by installing Docker and executing commands to manage images and interact with containers effectively.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT : 9

Date of Performance	
Date of Submission	

AIM

To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

PROBLEM DEFINITION

Master Dockerfile instructions to build an image for a sample web application.

THEORY

Jenkins is an open-source automation tool created with Java. It is extensively used as a Continuous Integration (CI) and Continuous Deployment (CD) tool.

Maven:

- Maven primarily intends to provide developers with:
    1. A comprehensive, reusable, easily maintainable model for projects.
    2. Plugins or tools to interact with and operate within this model.
- Maven is a POM (Project Object Model)-based build automation and project management tool written in Java. However, it is compatible with projects written in C#, Python, Ruby, etc.

A few Maven features worth mentioning are:

1. Maven can be used to build any number of projects into predefined output types like .jar, .war, metadata, etc.
2. Maven can automatically download necessary files from the repository when building a project.

Selenium using Maven in Jenkins: Selenium is a widely used test automation framework for validating web applications across different combinations of browsers, platforms, and devices (or emulators). It is widely used for testing areas such as functional testing, end-to-end testing, and more.

Why Jenkins and Selenium?

- Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to new environments when the tests pass.

Advantages of using Maven and Jenkins with Selenium:

- Whenever a change is made in the implementation, the changes are deployed on the test environment. Automation testing is performed continuously, and developers are kept informed about the build and test stage results.
- Test suites that comprise many test scenarios might take longer duration testing. A nightly build run can be scheduled for build and execution on the Jenkins server in such cases.

Traditionally, the Dockerfile is called Dockerfile and located in the root of the context. You use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

$ docker build -f /path/to/a/Dockerfile .

You can specify a repository and tag at which to save the new image if the build succeeds:

$ docker build -t shykes/myapp .

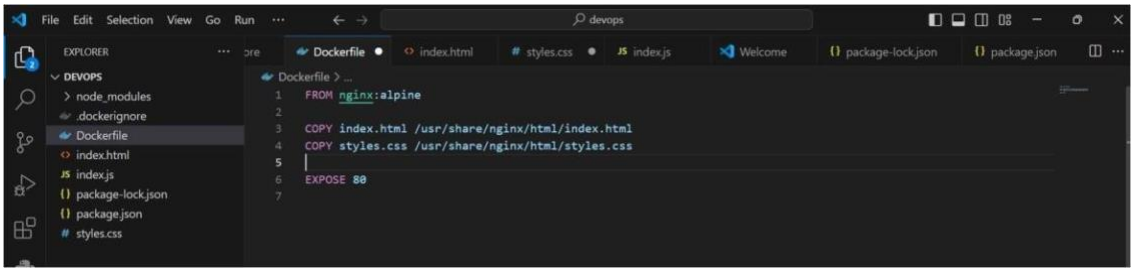
To tag the image into multiple repositories after the build, add multiple -t parameters when you run the build command:

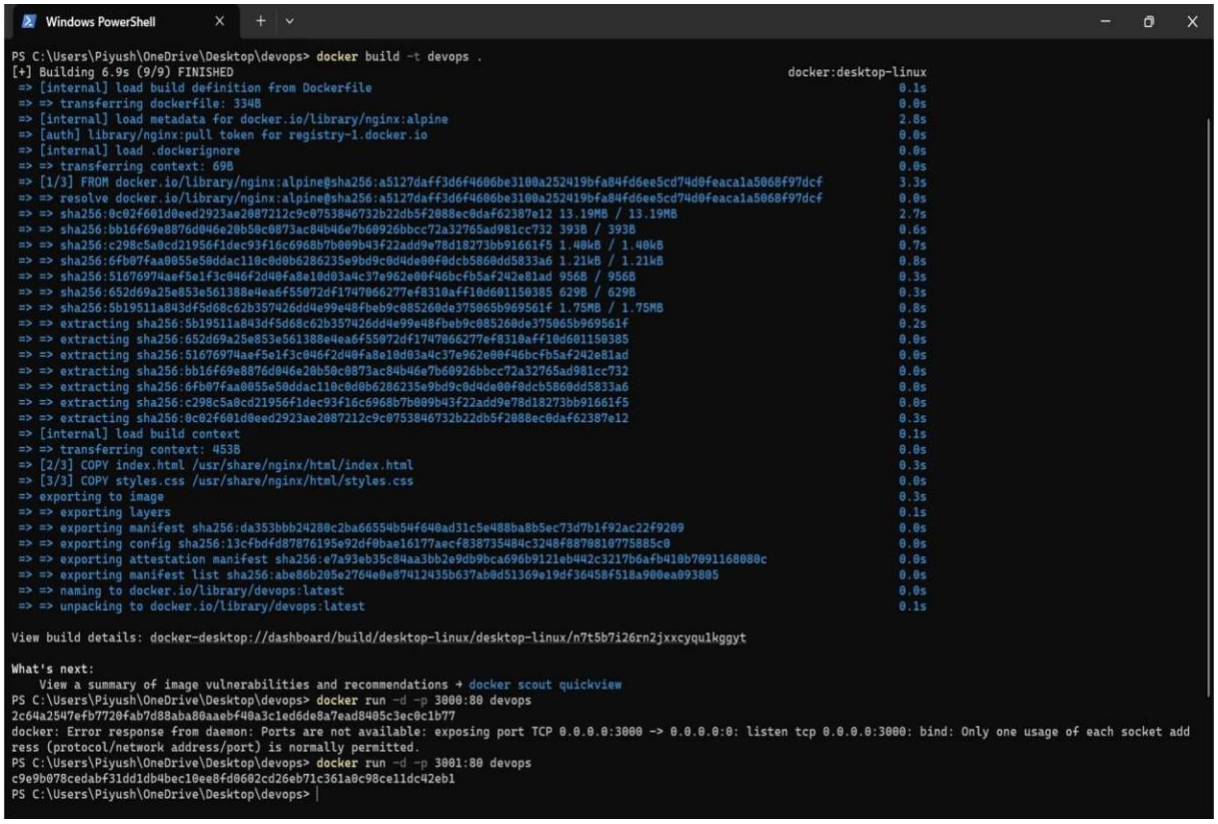
$ docker build -t shykes/myapp:1.0.2 -t shykes/myapp:latest .

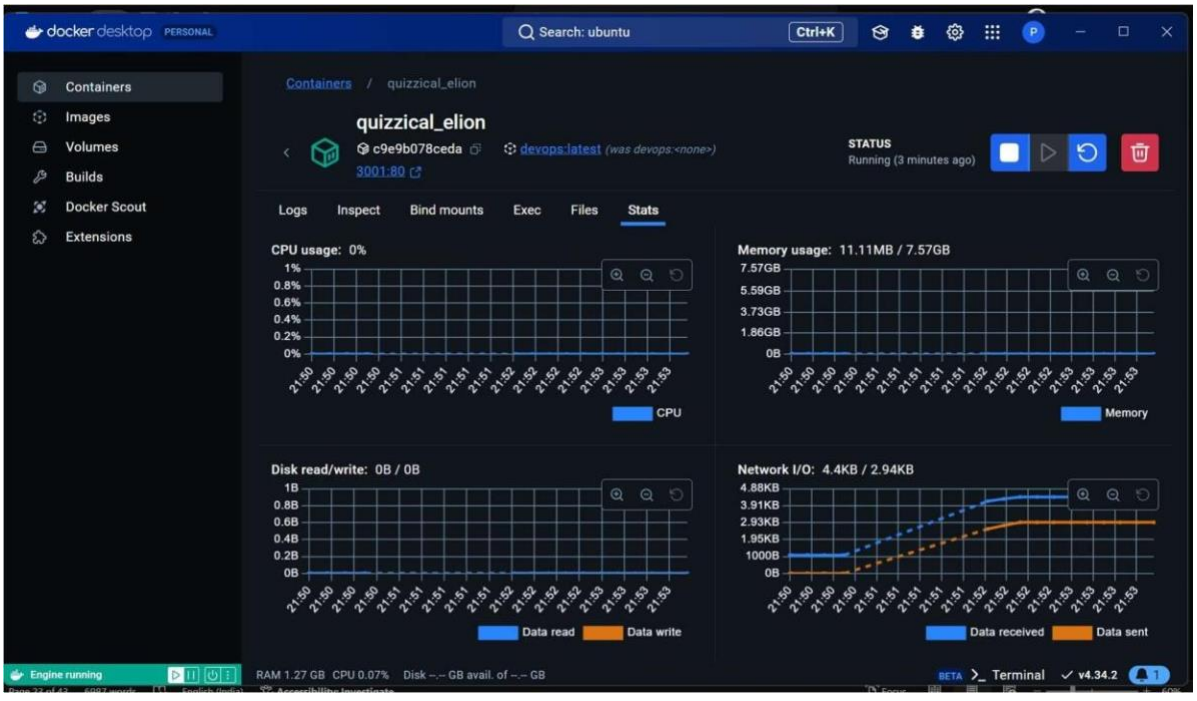
Before the Docker daemon runs the instructions in the Dockerfile, it performs a preliminary validation of the Dockerfile and returns an error if the syntax is incorrect:

$ docker build -t test/myapp . Syntax: syntax=[remote image reference]

OUTPUT

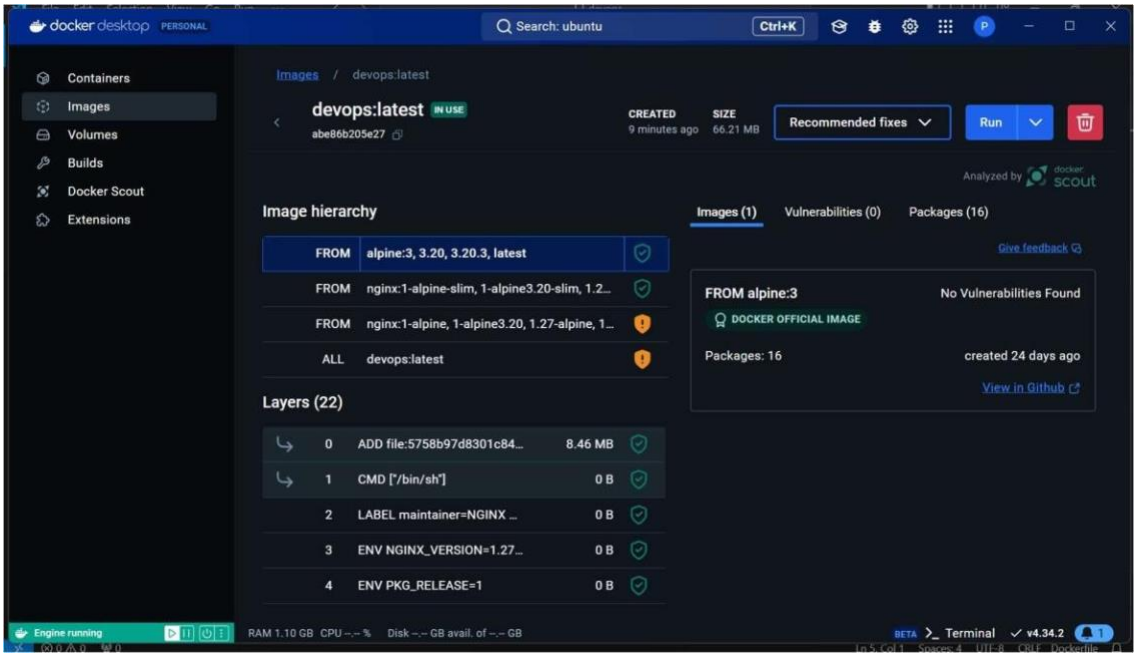






Welcome to My Sample Web Application

This is a simple web app running in a Docker container.



CONCLUSION

Consequently, learning Dockerfile instructions and building an image for a sample web application provided practical experience in containerization and application deployment.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT NO: 10

Date of Performance	
Date of Submission	

AIM

To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet

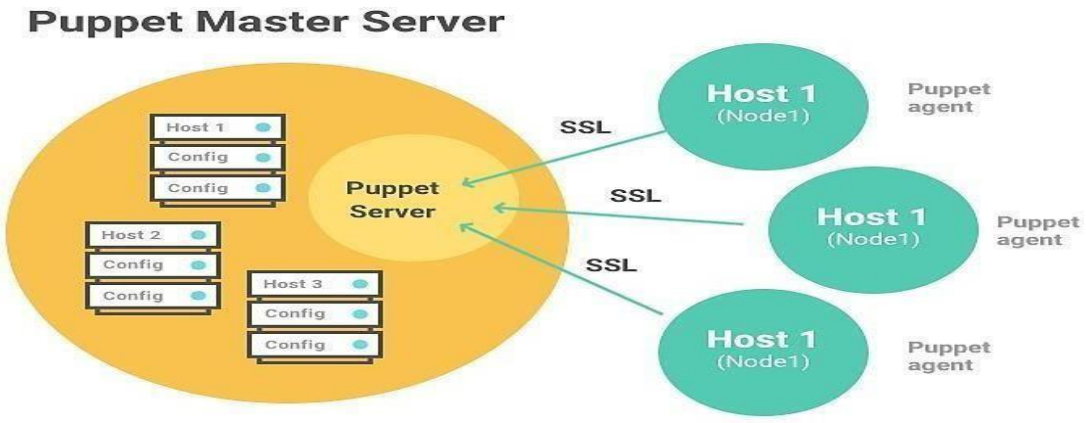
PROBLEM DEFINITION

Set up and configure pull-based software configuration management and provisioning tools with Puppet.

THEORY

Configuration Management: Configuration management is the process of maintaining software and computer systems (e.g., servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes. System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate these tasks by writing scripts. However, it is challenging when working on a massive infrastructure. Configuration management tools like Puppet were introduced to resolve such issues.

Puppet: Puppet is a system management tool for centralizing and automating the configuration management process. Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across an organization's infrastructure. 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.



Puppet performs the following functions:

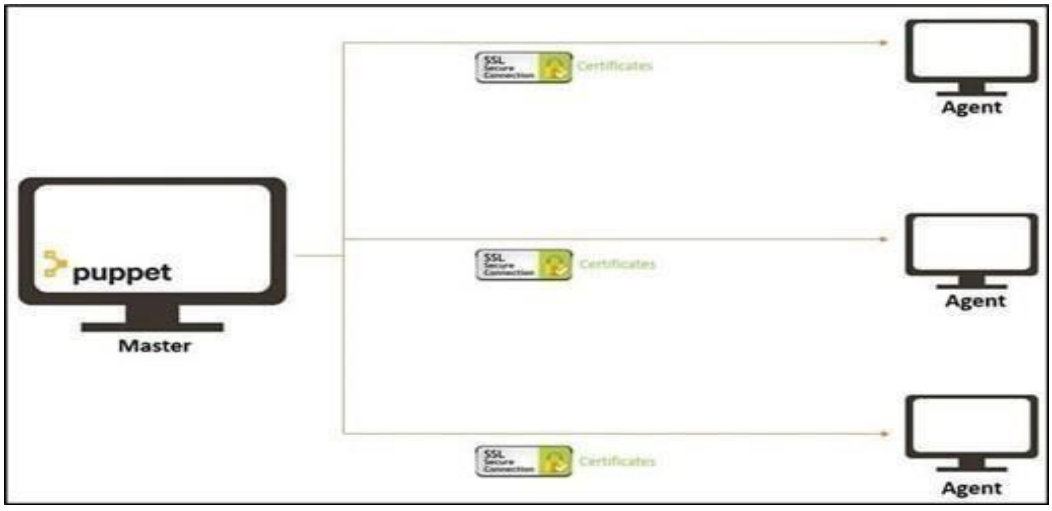
- Puppet allows you to define distinct configurations for every host.
- The tool continuously monitors servers to confirm whether the required configuration exists or not and if it is not altered. If the configuration is changed, the Puppet tool will revert to the pre-defined configuration on the host.
- It also provides control over all the configured systems, so a centralized change gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements infrastructure as code because policies and configurations are written as code.

Pull-based deployment model: In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software, and then configure themselves accordingly—for example, Puppet and Chef.

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

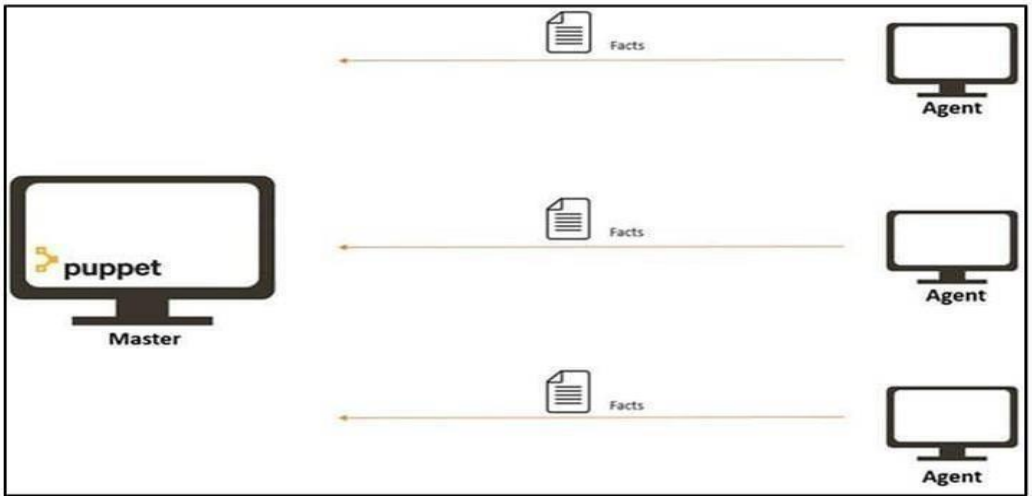
Example: Master – Agent Setup:

- **The Master:** A Linux-based machine with Puppet master software installed on it. It is responsible for maintaining configurations in the form of Puppet codes. The master node can only be Linux.
- **The Agents:** The target machines managed by Puppet with the Puppet agent software installed on them. The agent can be configured on any supported operating system such as Linux, Windows, Solaris, or Mac OS. The communication between master and agent is established through secure certificates.

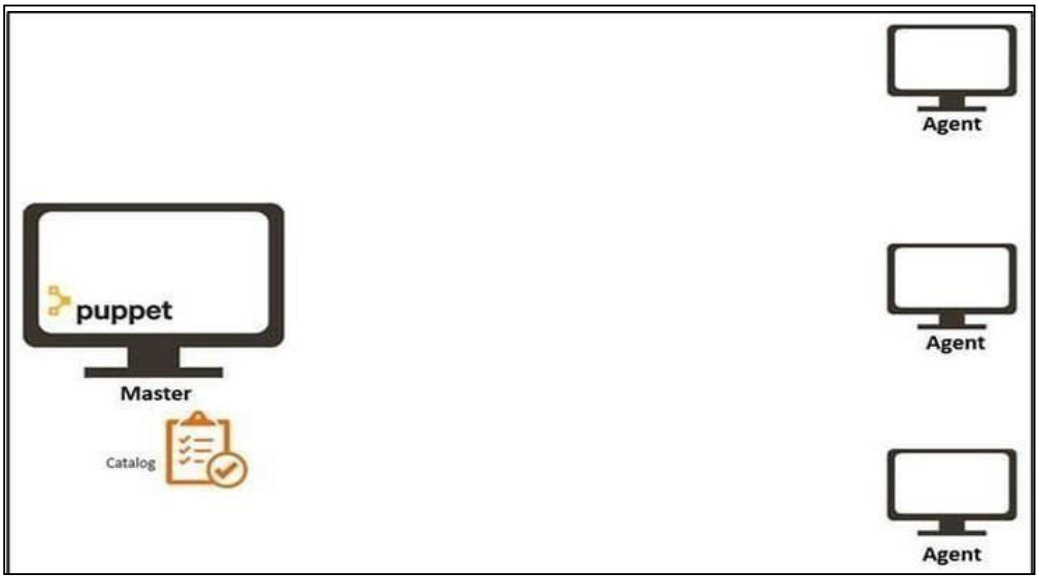


Communication between the Master and the Agent:

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

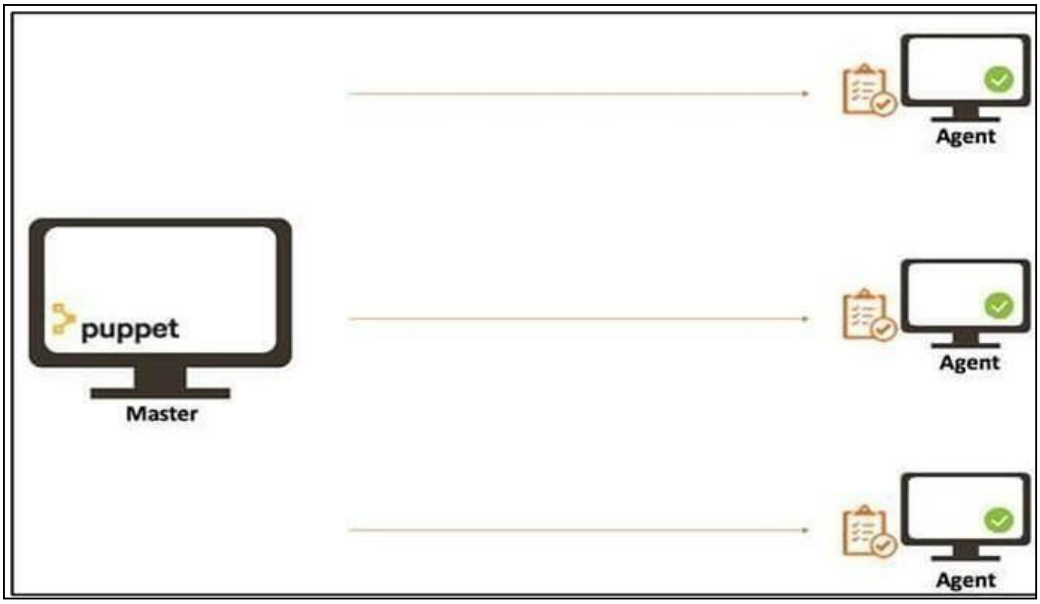


2. Puppet Master uses this data and compiles a list with the configuration to be applied to the agent. This list of configurations to be performed on an agent is known as a catalog. This could involve changes such as package installation, upgrades or removals, File System creation, user creation or deletion,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



3. The agent uses this list of configurations to apply any required configuration changes on the node. If there are no drifts in the configuration, the agent does

not perform any configuration changes and leaves the node to run with the same configuration.

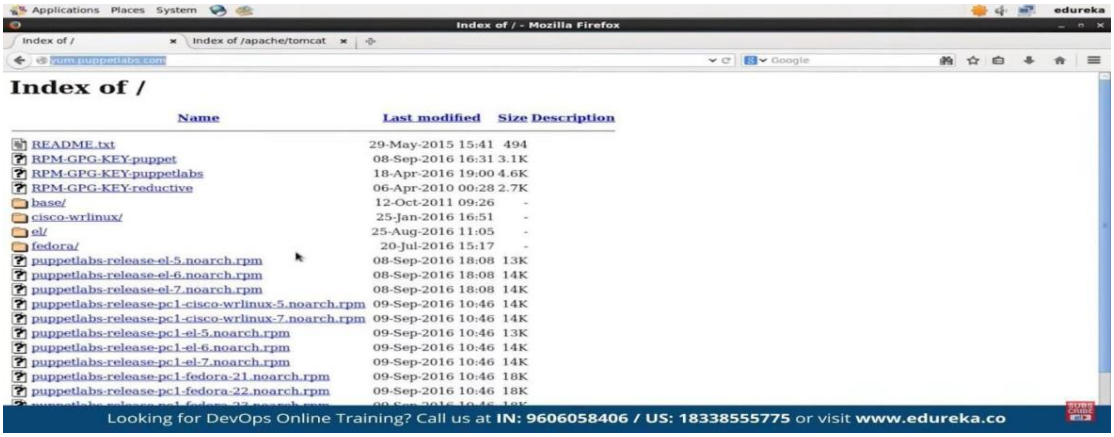


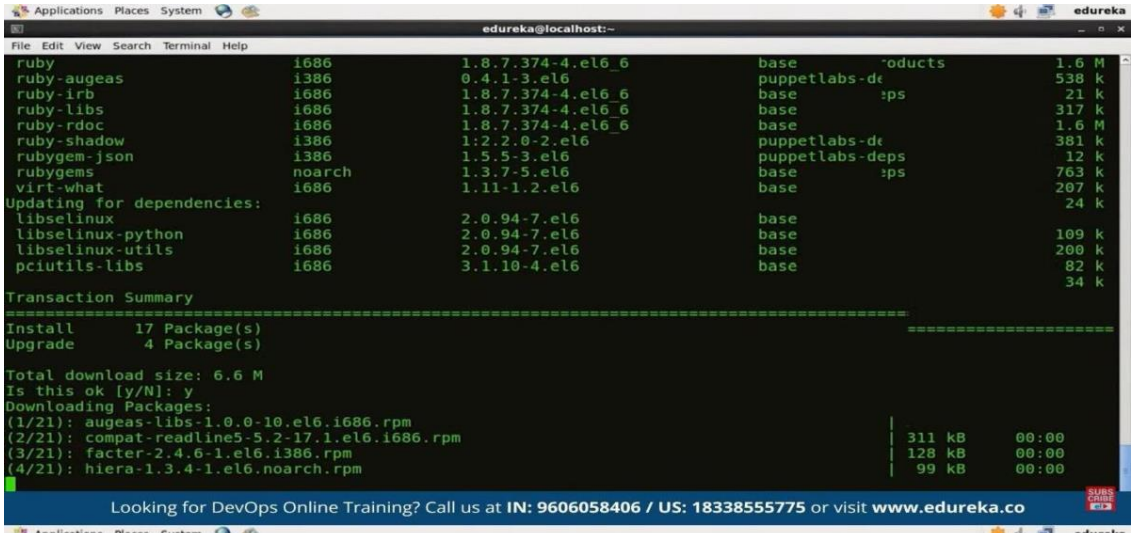
4. Once it is done, the node reports back to Puppet master indicating that the configuration has been applied and completed.

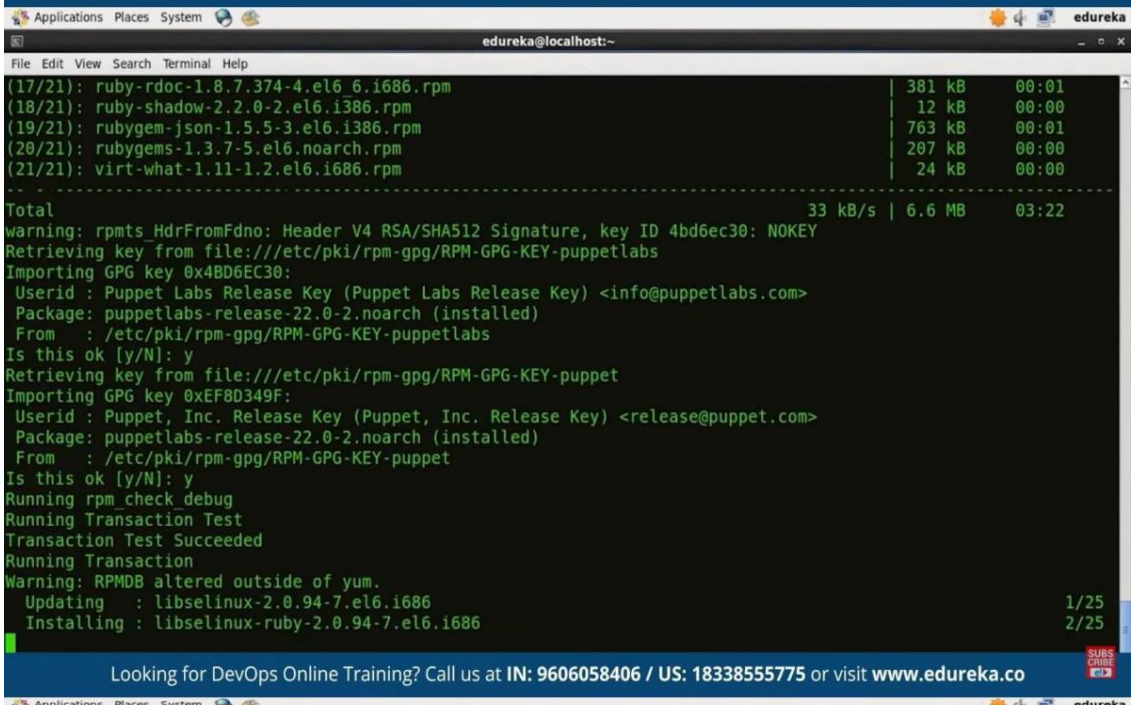
Installation of Puppet on master and agent and initial configuration:

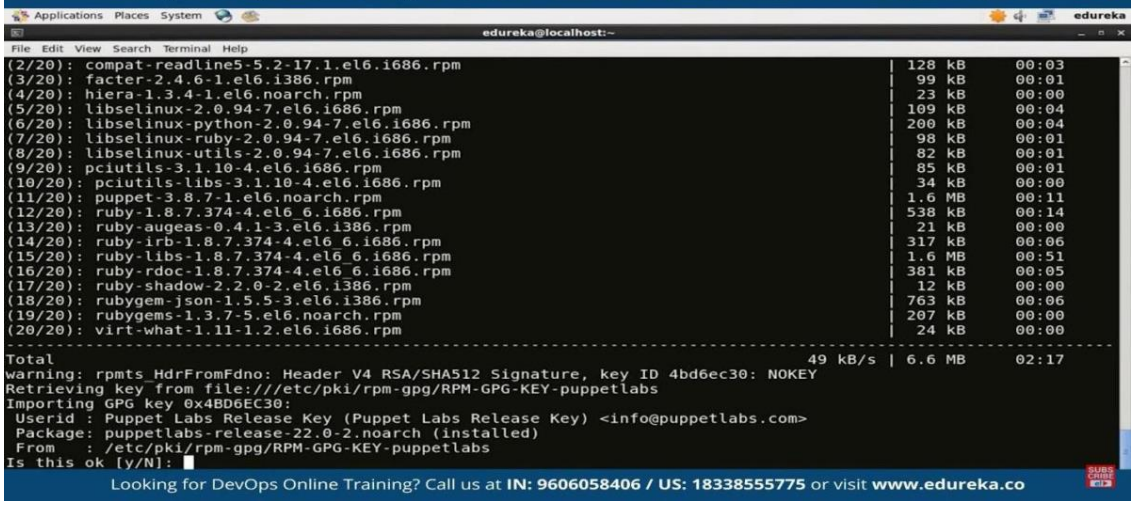
- **Initial configuration on Puppet Master:** Install Puppet software with apt install puppet master -y command on the master.
- **On Agent:** Install with apt install puppet -y command.
- **Exchanging certificates:** Signing certificate from master.

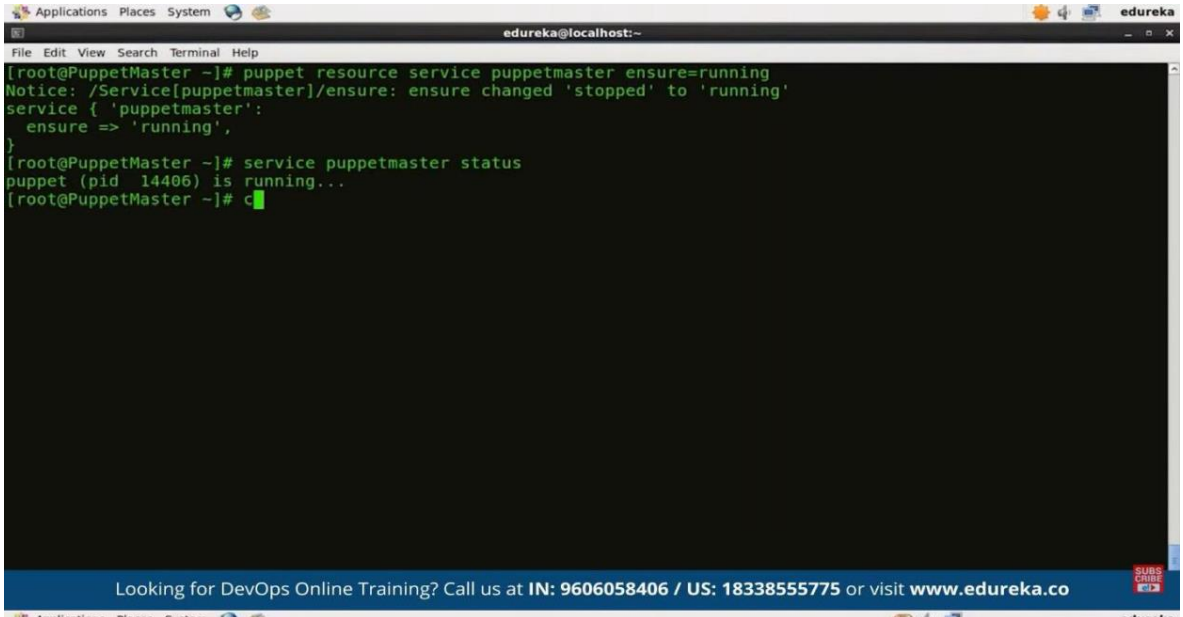
OUTPUT

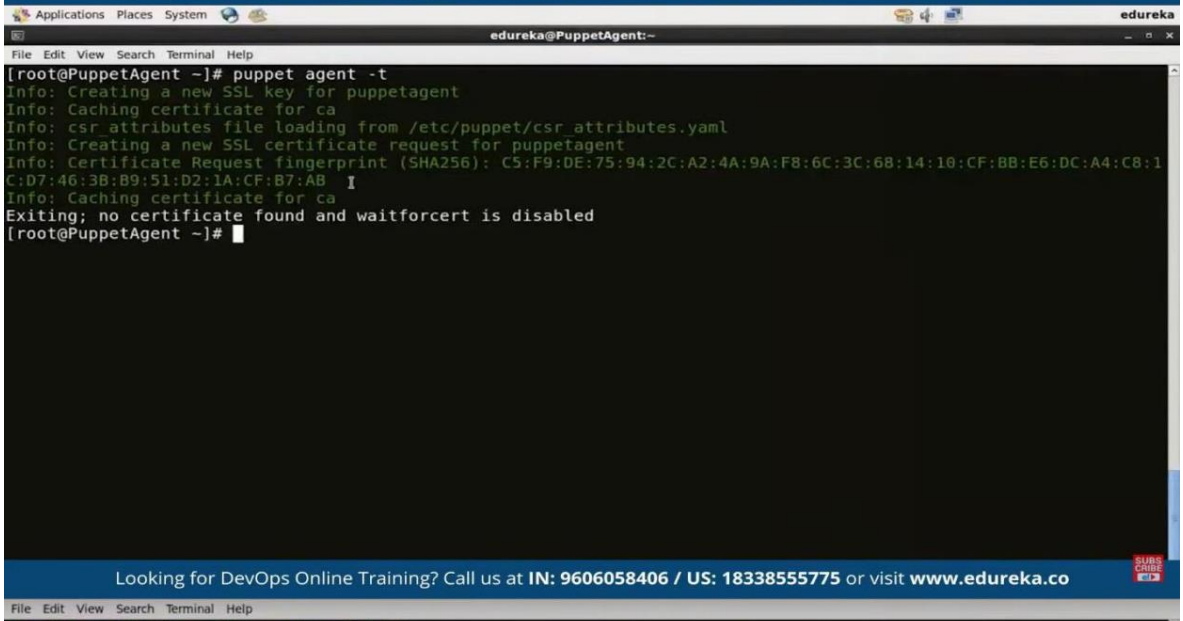


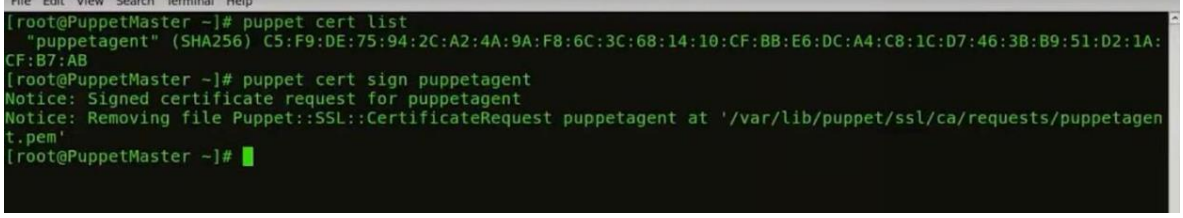

Applications Places System
edureka
edureka@localhost:~
File Edit View Search Terminal Help
ruby i686 1.8.7.374-4.el6_6 base oducts 1.6 M
ruby-augeas i386 0.4.1-3.el6 puppetlabs-d 538 k
ruby-irb i686 1.8.7.374-4.el6_6 base ps 21 k
ruby-libs i686 1.8.7.374-4.el6_6 base 317 k
ruby-rdoc i686 1.8.7.374-4.el6_6 base 1.6 M
ruby-shadow i386 1:2.2.0-2.el6 puppetlabs-d 381 k
rubygem-json i386 1.5.5-3.el6 puppetlabs-deps 12 k
rubygems noarch 1.3.7-5.el6 base ps 763 k
virt-what i686 1.11-1.2.el6 base 207 k
Updating for dependencies:
libselinux i686 2.0.94-7.el6 base 24 k
libselinux-python i686 2.0.94-7.el6 base 109 k
libselinux-utils i686 2.0.94-7.el6 base 200 k
pciutils-libs i686 3.1.10-4.el6 base 82 k
34 k
Transaction Summary
Install 17 Package(s)
Upgrade 4 Package(s)
Total download size: 6.6 M
Is this ok [y/N]: y
Downloading Packages:
(1/21): augeas-libs-1.0.0-10.el6.i686.rpm
(2/21): compat-readline5-5.2-17.1.el6.i686.rpm | 311 kB 00:00
(3/21): facter-2.4.6-1.el6.i386.rpm | 128 kB 00:00
(4/21): hiera-1.3.4-1.el6.noarch.rpm | 99 kB 00:00
Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

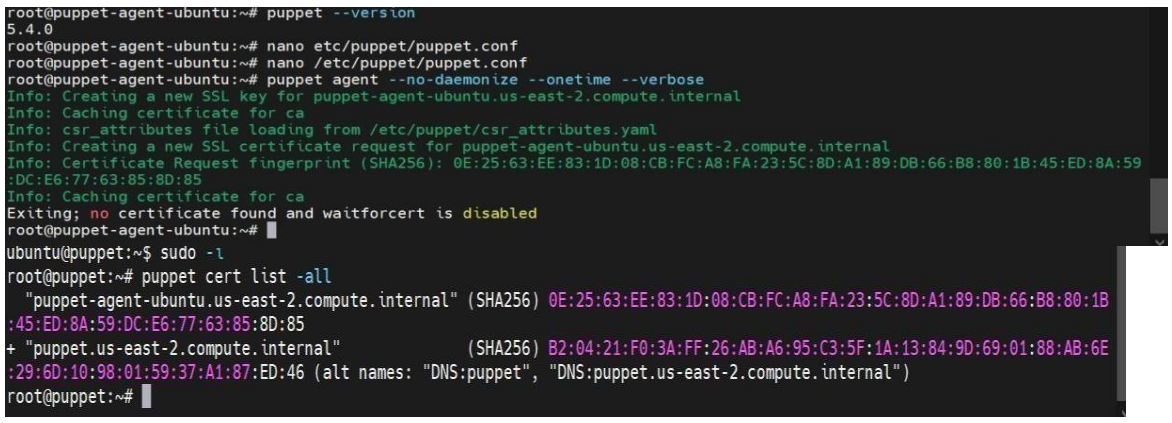

Applications Places System
edureka
edureka@localhost:~
File Edit View Search Terminal Help
(17/21): ruby-rdoc-1.8.7.374-4.el6_6.i686.rpm | 381 kB 00:01
(18/21): ruby-shadow-2.2.0-2.el6.i386.rpm | 12 kB 00:00
(19/21): rubygem-json-1.5.5-3.el6.i386.rpm | 763 kB 00:01
(20/21): rubygems-1.3.7-5.el6.noarch.rpm | 207 kB 00:00
(21/21): virt-what-1.11-1.2.el6.i686.rpm | 24 kB 00:00
Total 33 kB/s | 6.6 MB 03:22
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA512 Signature, key ID 4bd6ec30: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Importing GPG key 0x4BD6EC30:
Userid : Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
Package: puppetlabs-release-22.0-2.noarch (installed)
From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Is this ok [y/N]: y
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet
Importing GPG key 0xEF8D349F:
Userid : Puppet, Inc. Release Key (Puppet, Inc. Release Key) <release@puppet.com>
Package: puppetlabs-release-22.0-2.noarch (installed)
From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppet
Is this ok [y/N]: y
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
Updating : libselinux-2.0.94-7.el6.i686 1/25
Installing : libselinux-ruby-2.0.94-7.el6.i686 2/25
Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co


Applications Places System
edureka
edureka@localhost:~
File Edit View Search Terminal Help
(2/20): compat-readline5-5.2-17.1.el6.i686.rpm | 128 kB 00:03
(3/20): facter-2.4.6-1.el6.i386.rpm | 99 kB 00:01
(4/20): hiera-1.3.4-1.el6.noarch.rpm | 23 kB 00:00
(5/20): libselinux-2.0.94-7.el6.i686.rpm | 109 kB 00:04
(6/20): libselinux-python-2.0.94-7.el6.i686.rpm | 200 kB 00:04
(7/20): libselinux-ruby-2.0.94-7.el6.i686.rpm | 98 kB 00:01
(8/20): libselinux-utils-2.0.94-7.el6.i686.rpm | 82 kB 00:01
(9/20): pciutils-3.1.10-4.el6.i686.rpm | 85 kB 00:01
(10/20): pciutils-libs-3.1.10-4.el6.i686.rpm | 34 kB 00:00
(11/20): puppet-3.8.7-1.el6.noarch.rpm | 1.6 MB 00:11
(12/20): ruby-1.8.7.374-4.el6_6.i686.rpm | 538 kB 00:14
(13/20): ruby-augeas-0.4.1-3.el6.i386.rpm | 21 kB 00:00
(14/20): ruby-irb-1.8.7.374-4.el6_6.i686.rpm | 317 kB 00:06
(15/20): ruby-libs-1.8.7.374-4.el6_6.i686.rpm | 1.6 MB 00:51
(16/20): ruby-rdoc-1.8.7.374-4.el6_6.i686.rpm | 381 kB 00:05
(17/20): ruby-shadow-2.2.0-2.el6.i386.rpm | 12 kB 00:00
(18/20): rubygem-json-1.5.5-3.el6.i386.rpm | 763 kB 00:06
(19/20): rubygems-1.3.7-5.el6.noarch.rpm | 207 kB 00:00
(20/20): virt-what-1.11-1.2.el6.i686.rpm | 24 kB 00:00
Total 49 kB/s | 6.6 MB 02:17
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA512 Signature, key ID 4bd6ec30: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Importing GPG key 0x4BD6EC30:
Userid : Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
Package: puppetlabs-release-22.0-2.noarch (installed)
From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Is this ok [y/N]:
Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

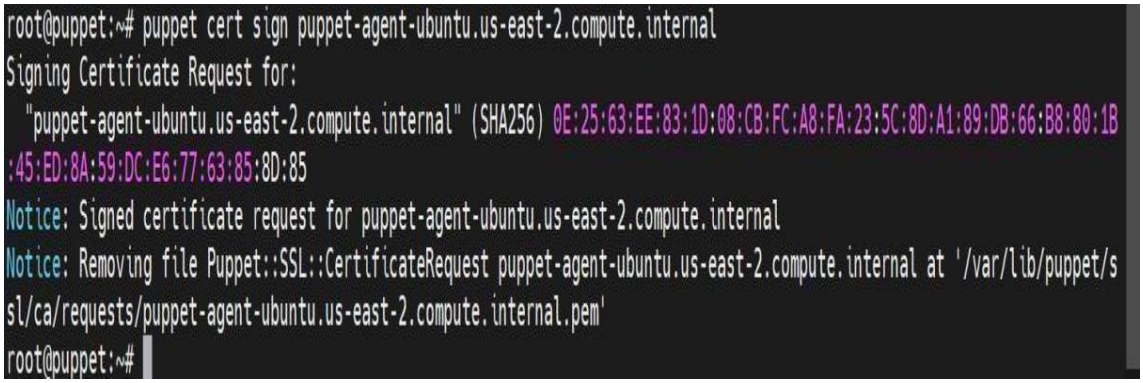

Applications Places System
edureka
edureka@localhost:~
File Edit View Search Terminal Help
[root@PuppetMaster ~]# puppet resource service puppetmaster ensure=running
Notice: /Service[puppetmaster]/ensure: ensure changed 'stopped' to 'running'
service { 'puppetmaster':
ensure => 'running',
}
[root@PuppetMaster ~]# service puppetmaster status
puppet (pid 14406) is running...
[root@PuppetMaster ~]# c
Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co


Applications Places System
edureka
edureka@PuppetAgent:~
File Edit View Search Terminal Help
[root@PuppetAgent ~]# puppet agent -t
Info: Creating a new SSL key for puppetagent
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for puppetagent
Info: Certificate Request fingerprint (SHA256): C5:F9:DE:75:94:2C:A2:4A:9A:F8:6C:3C:68:14:10:CF:BB:E6:DC:A4:C8:1C:D7:46:3B:B9:51:D2:1A:CF:B7:AB
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
[root@PuppetAgent ~]#
Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co


File Edit View Search Terminal Help
[root@PuppetMaster ~]# puppet cert list
"puppetagent" (SHA256) C5:F9:DE:75:94:2C:A2:4A:9A:F8:6C:3C:68:14:10:CF:BB:E6:DC:A4:C8:1C:D7:46:3B:B9:51:D2:1A:CF:B7:AB
[root@PuppetMaster ~]# puppet cert sign puppetagent
Notice: Signed certificate request for puppetagent
Notice: Removing file Puppet::SSL::CertificateRequest puppetagent at '/var/lib/puppet/ssl/ca/requests/puppetagent.pem'
[root@PuppetMaster ~]#


root@puppet-agent-ubuntu:~# puppet --version
5.4.0
root@puppet-agent-ubuntu:~# nano etc/puppet/puppet.conf
root@puppet-agent-ubuntu:~# nano /etc/puppet/puppet.conf
root@puppet-agent-ubuntu:~# puppet agent --no-daemonize --onetime --verbose
Info: Creating a new SSL key for puppet-agent-ubuntu.us-east-2.compute.internal
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for puppet-agent-ubuntu.us-east-2.compute.internal
Info: Certificate Request fingerprint (SHA256): 0E:25:63:EE:83:1D:08:CB:FC:A8:FA:23:5C:8D:A1:89:DB:66:B8:80:1B:45:ED:8A:59:DC:E6:77:63:85:8D:85
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
root@puppet-agent-ubuntu:~#
ubuntu@puppet:~$ sudo -i
root@puppet:~# puppet cert list -all
"puppet-agent-ubuntu.us-east-2.compute.internal" (SHA256) 0E:25:63:EE:83:1D:08:CB:FC:A8:FA:23:5C:8D:A1:89:DB:66:B8:80:1B:45:ED:8A:59:DC:E6:77:63:85:8D:85
+ "puppet.us-east-2.compute.internal" (SHA256) B2:04:21:F0:3A:FF:26:AB:A6:95:C3:5F:1A:13:84:9D:69:01:88:AB:6E:29:6D:10:98:01:59:37:A1:87:ED:46 (alt names: "DNS:puppet", "DNS:puppet.us-east-2.compute.internal")
root@puppet:~#

Signing certificate from master



CONCLUSION

Hence, installing and configuring pull-based software configuration management and provisioning tools using Puppet facilitated efficient configuration management and automation.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN

EXPERIMENT NO: 11

Date of Performance	
Date of Submission	

AIM

To learn Software Configuration Management and provisioning using Puppet Blocks (Manifest, Modules, Classes, Function).

PROBLEM DEFINITION

Acquire knowledge in software configuration management and provisioning using Puppet components such as manifests, modules, classes, and functions.

THEORY

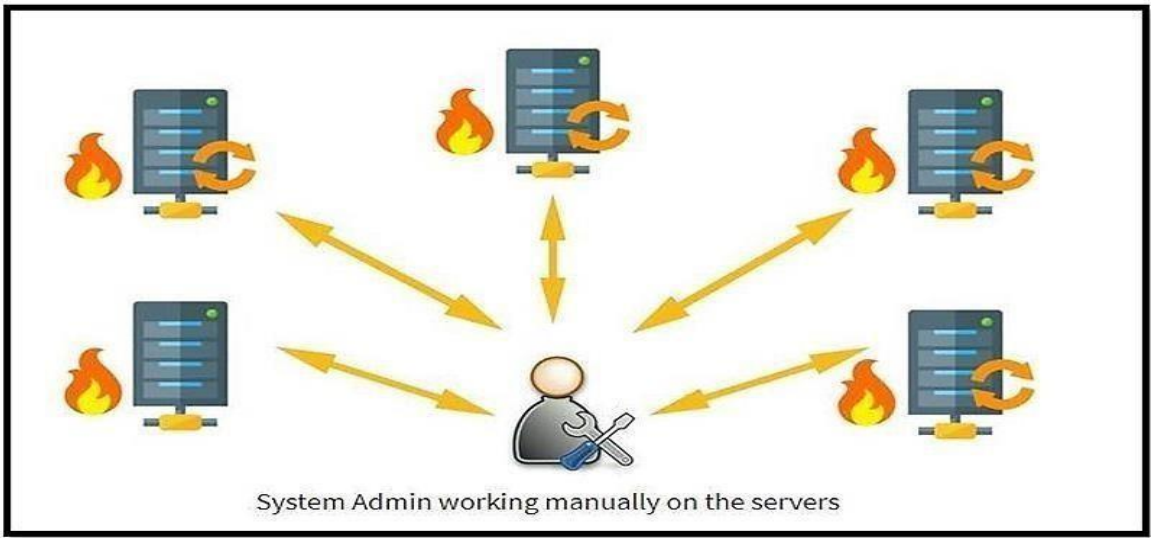
What is Configuration Management? Configuration management is the process of maintaining software and computer systems (e.g., servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes. System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate these tasks by writing scripts. However, it is challenging when working on a massive infrastructure. Configuration management tools like Puppet were introduced to resolve such issues.

What is Puppet? Puppet is a system management tool for centralizing and automating the configuration management process. Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across an organization's infrastructure. 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.

What are the Puppet versions? Puppet comes in two versions:

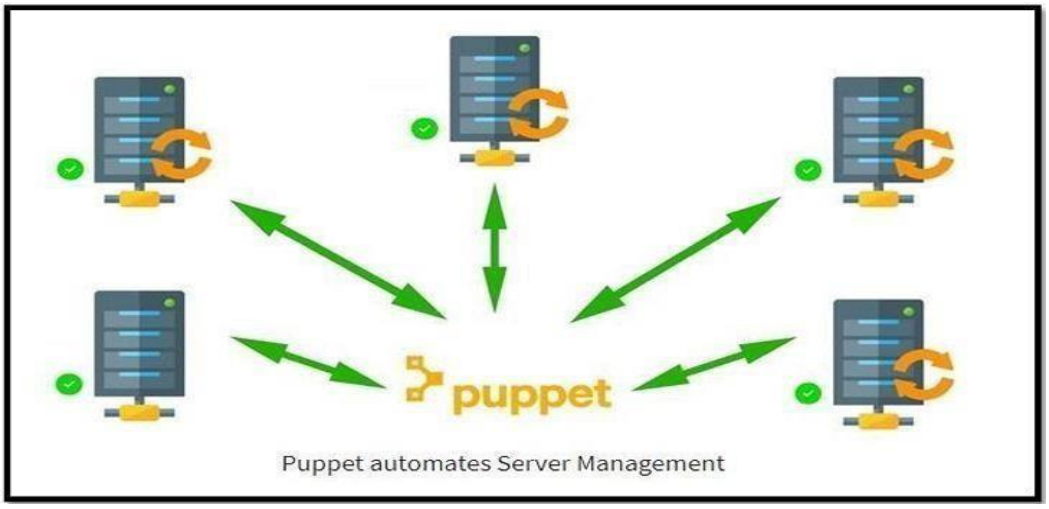
- **Open Source Puppet:** It is a basic version of the Puppet configuration management tool, also known as Open Source Puppet. It is available directly from Puppet's website and is licensed under the Apache 2.0 system.
- **Puppet Enterprise:** A commercial version that offers features like compliance reporting, orchestration, role-based access control, GUI, API, and command-line tools for effective management of nodes.

What Puppet can do: For example, if you have an infrastructure with about 100 servers, Puppet allows you to write simple code that can be deployed automatically on these servers. This reduces human effort and makes the development process faster and more effective.



Puppet performs the following functions:

- Puppet allows you to define distinct configurations for every host.
- The tool continuously monitors servers to confirm whether the required configuration exists or not and if it is not altered. If the configuration is changed, the Puppet tool will revert to the pre-defined configuration on the host.
- It also provides control over all the configured systems, so a centralized change gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements infrastructure as code because policies and configurations are written as code.



Puppet DSL and Programming Paradigms: Before learning Puppet DSL, let's understand programming paradigms. A programming paradigm is a style used in computer programming. Four types of paradigms are:

1. Imperative
2. Declarative
3. Functional (a subset of the declarative paradigm)
4. Object-oriented

We will focus on Imperative and Declarative.

Imperative Paradigms: This programming paradigm expresses the logic of a computation (what to do) and describes its control flow (how to do it). Example: If you are going to your office, you book a cab and give step-by-step directions to the driver until you reach the office. Specifying what to do and how to do it is an imperative style.

Declarative Paradigms: This programming paradigm expresses the logic of a computation (what to do) without describing its control flow (how to do it). Example: If you are going to your office, you book an Uber cab and specify the final destination (office). Specifying what to do, not how to do it, is a declarative style. Puppet uses a declarative programming approach.

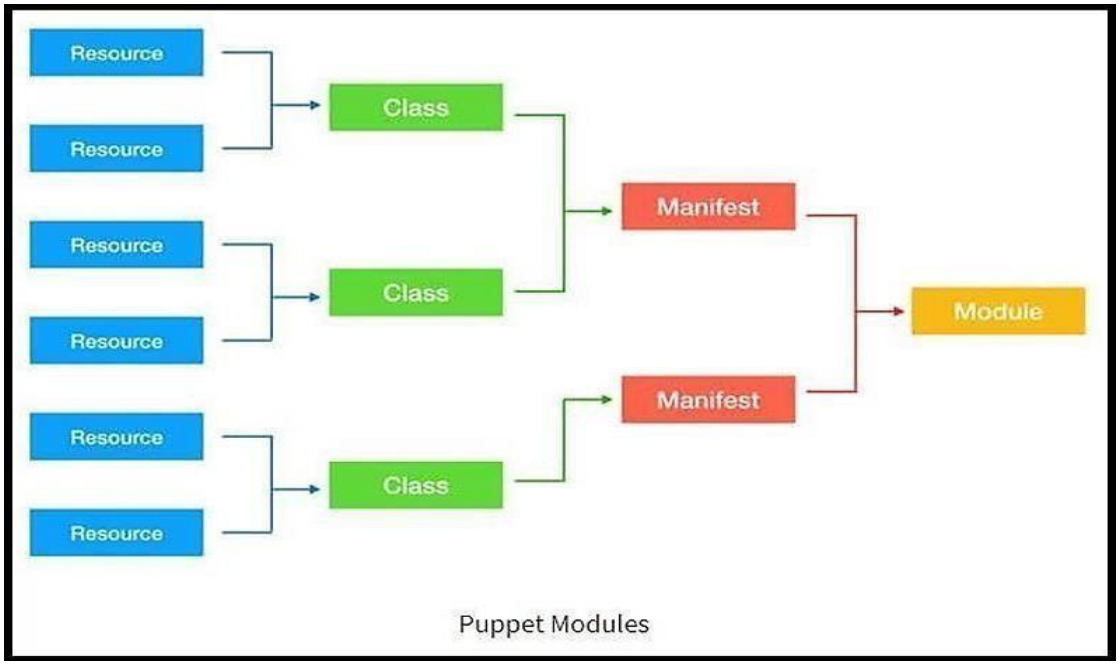
Example: Create a user on the system. It can be done using the imperative programming pattern by a shell script: Here, we specify how to create the user and what commands to use on the operating system. However, it can be done using the declarative programming pattern with only a few lines of Puppet code, Puppet DSL, and still achieve the same result.

Deployment models of configuration management tools: There are two deployment models for configuration management tools:

- **Push-based deployment model:** Initiated by a master node.
- **Pull-based deployment model:** Initiated by agents.

Push-based deployment model: In this deployment model, the master server pushes the configurations and software to the individual agents. After verifying a secure connection, the master runs commands remotely on the agents. For example, Ansible and Salt Stack.

Pull-based deployment model: In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software, and then configure themselves accordingly—for example, Puppet and Chef.



Types of Puppet resources

In general, a system consists of files, users, services, processes, packages, etc. In Puppet, these are called resources. Resources are the fundamental building blocks in Puppet. All the operations on puppet agents are performed with the help of puppet resources. Puppet resources are the readymade tools that are used to perform various tasks and operations on any supported platform. We can use a single puppet resource to performa specific task, or we can use multiple puppet resources together to perform some complex application configurations deployments.

Resources can have different types. Puppet uses resources and resource types in order to describe a system's configuration. There are three kinds of resource types:

1. Puppet core or built-in resource types.

2. Puppet defined resource types.

3. Puppet custom resource types.

Puppet core or built-in resource types:

Core or built-in resource types are the pre-built puppet resource types shipped with puppet software. All of the core or built-in Puppet resource types are written and maintained by Puppet team.

Puppet defined resource types: Defined resource types are lightweight resource types written in Puppet declarative language using a combination of existing resource types.

Puppet custom resource types:

Custom resource types are completely customized resource types written in Ruby. command to display a list of Puppet relevant subcommands:

In our case, we are interested in the subcommand “resource” which we will use to find the information about inbuilt puppet resource types. In the terminal, type any of the following commands to display a list of actions associated with the puppet subcommand “resource“:

In this case, we have the resource as subcommand and –types as action. Puppet has 49 inbuilt core resource types. In the terminal, type the following command to display a list of available inbuilt puppet resource types:

Each type supports a list of attributes. These attributes provide a detailed description that Puppet uses to manage the resource. To find out all the attributes associated with the puppet resource type, use the following command:

puppet describe <resource type name>

Parameters will list all the available attributes for that resource type. puppet describe package. It’s hard for a new person to understand and relate many unmanaged puppet code files. This is where we need some grouping to tie together operations. The aim is to solve a single problem, such as all operations required to configure ssh on a server or ntp service or a complete web server or database server from scratch.

What Are Puppet Classes?
Puppet classes are the collection of Puppet resources bundled together as a single unit. Puppet introduced classes to make the structure reusable and organized. First, we need to define a class using class definition syntax; classes must be unique and can be declared only once with the same name:

```
class <class-name> {
<Resource declarations>
}
```

Example:

```
class ntpconfig {

  file {
    "/etc/ntp.conf":
    ensure => "present",
    content => "server 0.centos.pool.ntp.org iburst\n",
  }
}
```

So far, we have only defined the class, but we have not used it anywhere. Meaning this code that we have written will never get executed unless we declare this class elsewhere.

Class Declaration
To use a defined class in code, use the include keyword.

```
class ntpconfig {
  file {
  "/etc/ntp.conf":
    ensure => "present",
    content => "server 0.centos.pool.ntp.org iburst\n",
  }
}

include ntpconfig
```

Demo Install NTP First, make sure the NTP package is not already present on the server; the following command will return nothing if the telnet is not present on the server:

As we can see, the NTP package is already present on the server. Let's remove the existing NTP package:

After removing the package, ensure that the ntp.conf file is not existing:

Verify the NTP service does not exist by running the following command:

Create a new .pp file to save the code. From the command line:

Change to insert mode by pressing i from the keyboard. Code to create a new file:

After done with Editing:Press Esc. To save the file, press :wq!.

Next step is to check whether the code has any syntax errors. Execute the following command:
puppet parser validate demontp.pp

Make sure that you switch to root to be able to complete the test without any error, by executing the command:
su root

Testing is the next step in the code creation process. Execute the following command to perform a smoke test:
puppet apply demontp.pp --noop

Last step is to run Puppet in real mode and verify the output. puppet apply demontp.pp

Puppet didn't perform anything because the demo class was just defined but not declared.

So, until you declare the Puppet class, the code will not get applied. Let's declare the demo class inside the same code using include class name at the end of the code:

Again check whether the code has any syntax errors. Execute the following command: puppet parser validate demontp.pp

Make sure that you switch to root to be able to complete the test without any error, by executing the command:
su root

Testing is the next step in the code creation process. Execute the following command to perform a smoke test:
puppet apply demontp.pp --noop

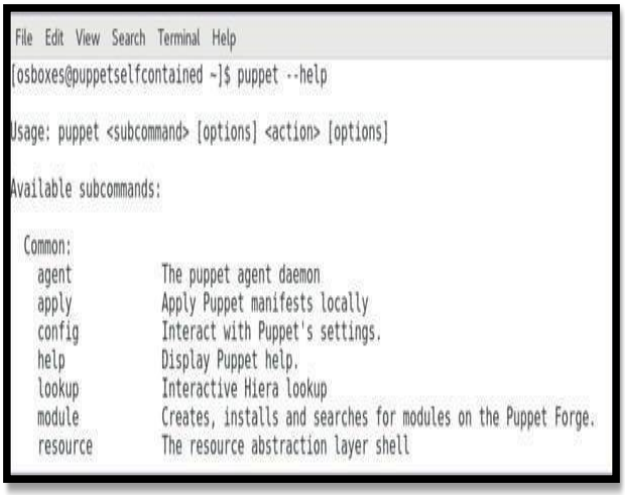
Last step is to run Puppet in real mode and verify the output. puppet apply demontp.pp

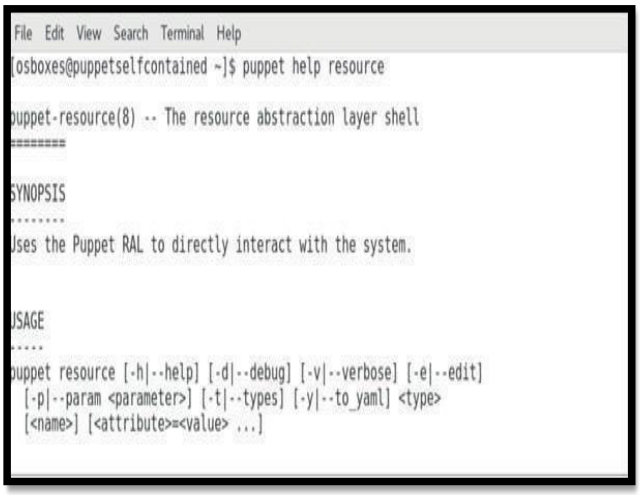
This time the code gets applied because the class was defined and then declared.

Ensure that ntp.conf is now existing:
ls -lrt /etc/ntp.conf

Verify the NTP service has been started by running the following command: systemctl status ntpd

OUTPUT





```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet resource --types
augeas
cron
exec
file
filebucket
group
host
mount
notify
package
resources
schedule
scheduled_task
selboolean
selmodule
service
ssh_authorized_key
sshkey
stage
tidy
user
whit
yumrepo
zfs
```

```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet describe package

package
=======
Manage packages.  There is a basic dichotomy in package
support right now:  Some package types (such as yum and apt) can
retrieve their own package files, while others (such as rpm and sun)
cannot.  For those package formats that cannot retrieve their own files,
you can use the `source` parameter to point to the correct file.
Puppet will automatically guess the packaging format that you are
using based on the platform you are on, but you can override it
using the `provider` parameter; each provider defines what it
requires in order to function, and you must meet those requirements
to use a given provider.
You can declare multiple package resources with the same `name`, as long
as they specify different providers and have unique titles.
Note that you must use the _title_ to make a reference to a package
resource; `Package[<NAME>]` is not a synonym for `Package[<TITLE>]` like
it is for many other resource types.
**Autorequires:** If Puppet is managing the files specified as a
package's `adminfile`, `responsefile`, or `source`, the package
resource will autorequire those files.


Parameters
----------
```

```
File  Edit  View  Search  Terminal  Help


Parameters
----------

- **adminfile**
  A file containing package defaults for installing packages.
  This attribute is only used on Solaris. Its value should be a path to a
  local file stored on the target system. Solaris's package tools expect
  either an absolute file path or a relative path to a file in
  `/var/sadm/install/admin`.
  The value of `adminfile` will be passed directly to the `pkgadd` or
  `pkgrm` command with the `-a <ADMINFILE>` option.

- **allow_virtual**
  Specifies if virtual package names are allowed for install and
  uninstall.
  Valid values are `true`, `false`, `yes`, `no`.
  Requires features virtual_packages.

- **allowcdrom**
  Tells apt to allow cdrom sources in the sources.list file.
  Normally apt will bail if you try this.
  Valid values are `true`, `false`.

- **category**
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained etc]# rpm -qa | grep -i ntp
ntp-4.2.6p5-28.el7.centos.x86_64
python-ntplib-0.3.2-1.el7.noarch
fontpackages-filesystem-1.44-8.el7.noarch
ntpdate-4.2.6p5-28.el7.centos.x86_64
[root@puppetselfcontained etc]#
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained etc]# ls -lrt /etc/ntp.conf
ls: cannot access /etc/ntp.conf: No such file or directory
[root@puppetselfcontained etc]#
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# systemctl status ntpd
Unit ntpd.service could not be found.
[root@puppetselfcontained demo]#
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# puppet apply demontp.pp
Notice: Compiled catalog for puppetselfcontained.example.com in environment production
in 0.69 seconds
Notice: /Stage[main]/Ntpconfig/Package[ntp]/ensure: created
Notice: /Stage[main]/Ntpconfig/File[/etc/ntp.conf]/content: content changed '{md5}dc9e5
754ad2bb6f6c32b954c04431d0a' to '{md5}83fd3914bd6bb10f2b717c277d995b75'
Notice: /Stage[main]/Ntpconfig/Service[ntpd]/ensure: ensure changed 'stopped' to 'runni
ng'
Notice: Applied catalog in 1.88 seconds
[root@puppetselfcontained demo]#
```

Testing

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; disabled; vendor preset: disab
led)
   Active: active (running) since Sat 2019-03-23 04:14:35 EDT; 15s ago
  Process: 21487 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SU
CCESS)
```

CONCLUSION

Thus, learning software configuration management and provisioning with Puppet Blocks (Manifest, Modules, Classes, Function) provided in-depth knowledge for effective configuration and automation.

R1 (5 marks)	R2 (5 marks)	R3 (5 marks)	TOTAL (15 marks)	SIGN