AIM : understanding the different phases of software life cycle

1. **WATERFALL MODEL**

- **Sequential Approach:** Waterfall follows a linear and sequential approach to software development. Each phase (requirements, design, implementation, testing, deployment, maintenance) is completed before moving to the next one.
- **Rigidity:** It's relatively rigid, with limited scope for changes once a phase is completed.
- **Suitability:** Best suited for projects with wellunderstood and stable requirements where changes are unlikely.
- **Feedback:** Feedback from one phase to another is limited, making it challenging to accommodate changes later in the development process.
- **Risk Management:** Limited risk management as it doesn't involve iterative cycles to identify and mitigate risks.

2. **INCREMENTAL MODEL**

- **Iterative Development:** The Incremental model divides the project into small, manageable parts or increments, each delivering a portion of the functionality.
- **Phased Delivery:** Each increment goes through the phases of requirements, design, implementation, and testing. Additional functionality is added in subsequent increments.
- **Flexibility:** Allows for accommodating changes and enhancements in subsequent increments based on feedback obtained from previous increments.
- **Suitability:** Suited for projects where requirements are clear but can evolve over time, allowing for early delivery of partial functionality.
- **Risk Management:** Provides some level of risk management as it involves iterative cycles, though not as comprehensive as the Spiral model.

3. **SPIRAL MODEL**

- **RiskDriven Approach:** Spiral is a riskdriven model that combines iterative development with systematic aspects of the Waterfall model.
- **Risk Analysis:** It involves cycles of risk identification, analysis, and mitigation, along with development and testing.
- **Flexibility:** Offers flexibility and adaptability by accommodating changes at each spiral based on risk analysis.
- **Suitability:** Suited for projects with high risks and evolving requirements where flexibility and early risk management are critical.
- **Continuous Improvement:** Emphasizes continuous improvement through iterative cycles, allowing for refinement of requirements and design over time.

4. **RAD (Rapid Application Development) Model**

- **Approach:** RAD emphasizes rapid prototyping and iterative development to accelerate the software development process.
- **Phased Development:** It involves iterative cycles of prototyping, feedback, and adjustments, with the goal of quickly delivering a fully functional system.
- **User Involvement:** RAD encourages extensive user involvement and feedback throughout the development process to ensure the final product meets user expectations.
- **Suitability:** Suited for projects with rapidly changing requirements and where quick delivery of functional software is essential.
- **Risk Management:** While it prioritizes speed and user feedback, RAD may involve higher risks associated with potential scope creep and insufficient planning.

5. **V Model (Verification and Validation Model)**

- **Testing Integration:** The V model integrates testing phases in parallel with development phases, emphasizing early verification and validation of each phase's deliverables.
- **Phased Approach:** It follows a structured approach where each development phase has a corresponding testing phase, forming a Vshaped structure.
- **Rigidity:** The V model is relatively rigid compared to Agile methodologies, as it requires adhering to predefined processes and testing at each stage.
- **Suitability:** Suited for projects with welldefined requirements and where rigorous testing and validation are critical, such as safetycritical systems.
- **Risk Management:** Provides comprehensive risk management by ensuring early detection and resolution of defects through systematic testing.

6. **Agile Model**

- **Flexibility:** Agile methodologies, including Scrum, Kanban, and Extreme Programming (XP), prioritize flexibility and adaptability to changing requirements.
- **Iterative and Incremental Development:** Agile promotes iterative and incremental development, with small, manageable increments delivered frequently.
- **Customer Collaboration:** Agile encourages continuous customer collaboration and feedback throughout the development process, ensuring the delivered product aligns with customer needs.
- **Suitability:** Suited for projects with evolving requirements and where rapid delivery of value to customers is essential.
- **Risk Management:** Agile mitigates risks by delivering small increments of functionality, allowing for early validation and course correction based on feedback.

7. **Prototype Model**

- **Purpose:** The Prototype Model focuses on creating a preliminary version of the software to gather feedback and refine requirements.
- **Development Approach:** It involves building a simplified version of the software, known as a prototype, to demonstrate key features and functionalities.
- **Feedback Gathering:** The primary goal is to gather feedback from stakeholders, including users, to validate requirements and refine the design.
- **Iterations:** While multiple iterations of the prototype may be developed, the focus is on refining the design and gathering feedback rather than delivering functional increments of the software.
- **Final Product:** The prototype is not intended to be the final product but rather a tool for requirements elicitation and validation.
- **Suitability:** Suitable for projects with unclear or evolving requirements, where stakeholders' feedback is essential for requirement refinement, and where there's a need to mitigate risks associated with unclear requirements.

8. Iterative Model

- **Purpose:** The Iterative Model focuses on delivering the software incrementally through repetitive cycles of development.
- **Development Approach:** It involves breaking down the project into smaller iterations, with each iteration delivering a functional increment of the software.
- **Feedback Gathering:** While feedback from stakeholders is important, the primary goal of each iteration is to deliver a working increment of the software.
- **Iterations:** Each iteration goes through all phases of the software development life cycle, including requirements analysis, design, implementation, testing, and deployment.
- **Final Product:** The final product evolves over time as new features and enhancements are added with each iteration.
- **Suitability:** Suitable for projects where requirements are expected to evolve over time, where early delivery of functional increments is desired, and where there's a need for continuous stakeholder feedback throughout the development process.

AIM : Git installation and create a repository and perform fetch, pull, branching operations.

DESCRIPTION :

- **Understanding Version Control:**
  Version control systems like Git are indispensable tools in modern software development. They allow teams to collaborate effectively, track changes, revert to previous versions if necessary, and maintain a structured development workflow.

- **Importance of Git Installation:**
  Git installation is the foundational step in establishing a versioncontrolled environment. It provides a local instance of Git on the developer's machine, enabling them to manage code versions efficiently.

- **Repository Creation:**
  Creating a Git repository serves as the cornerstone of version control. It centralizes project files, facilitating collaborative development and ensuring a centralized location for code history and changes.

- **Fetch Operation:**
  The fetch operation in Git is crucial for retrieving changes from a remote repository without automatically merging them into the local repository. It allows developers to review changes before integrating them into their working copy, ensuring code stability and quality.

- **Pull Operation:**
  Pulling changes from a remote repository is vital for synchronizing the local repository with the latest updates from the central codebase. It enables seamless collaboration among team members by ensuring that everyone is working with the most recent codebase.

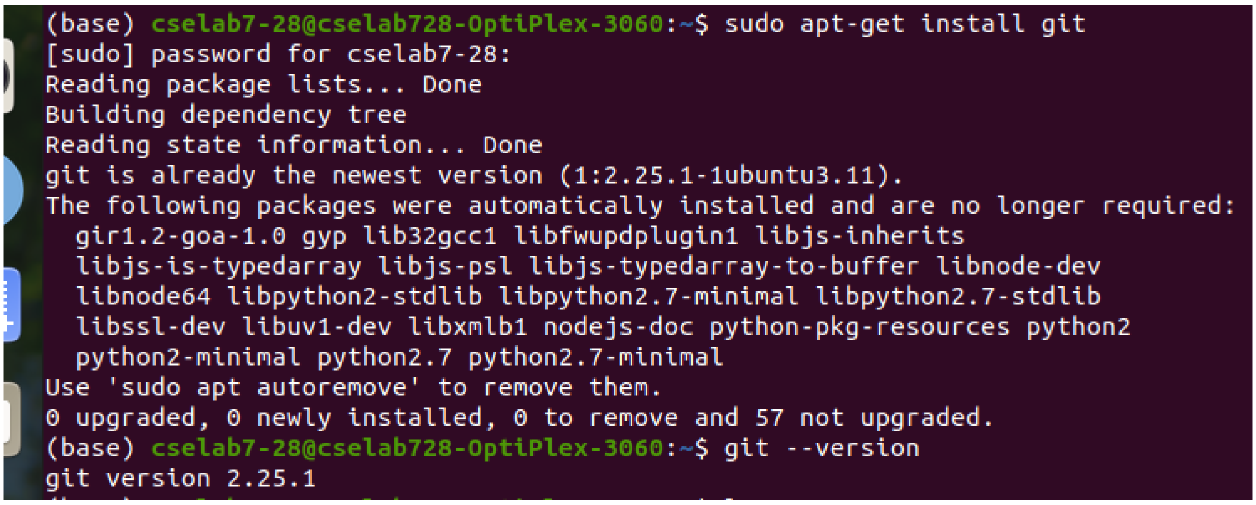
- **Branching Operations:**
  Branching is a fundamental concept in Git that facilitates parallel development efforts within a project. Creating branches enables developers to work on new features, bug fixes, or experiments without affecting the main codebase. Branches promote code isolation, experimentation, and seamless integration through merging.

PROCECURE :

Step 1: Installing Git

Use your package manager to install Git. For example, on Ubuntu, you can run
sudo apt-get install git.
Verify Installation:

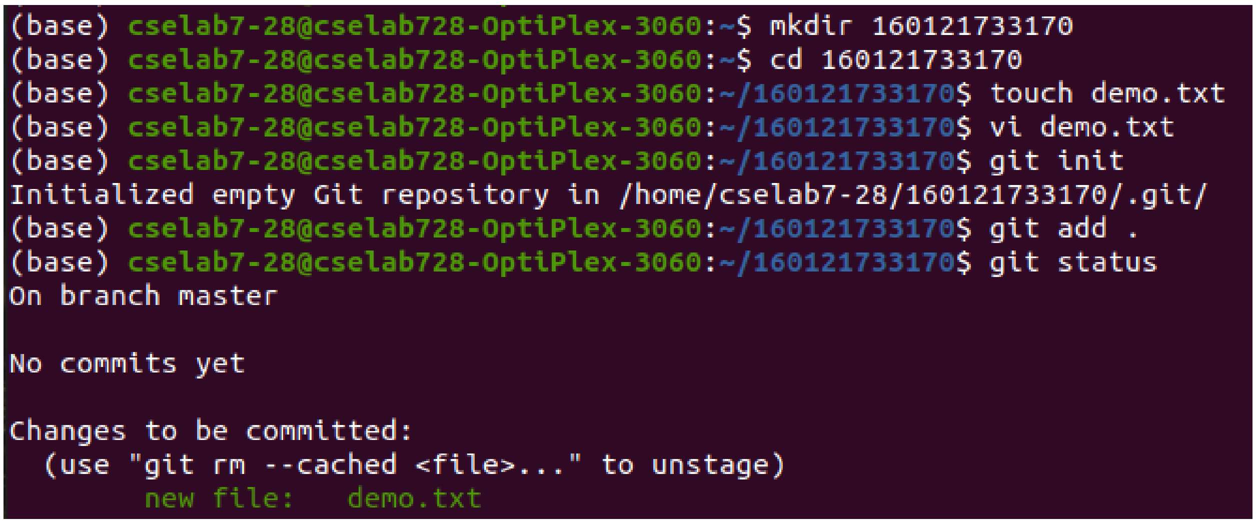
- Open a terminal (or command prompt on Windows).
- Type ***git --version*** and press Enter. You should see the installed Git version.



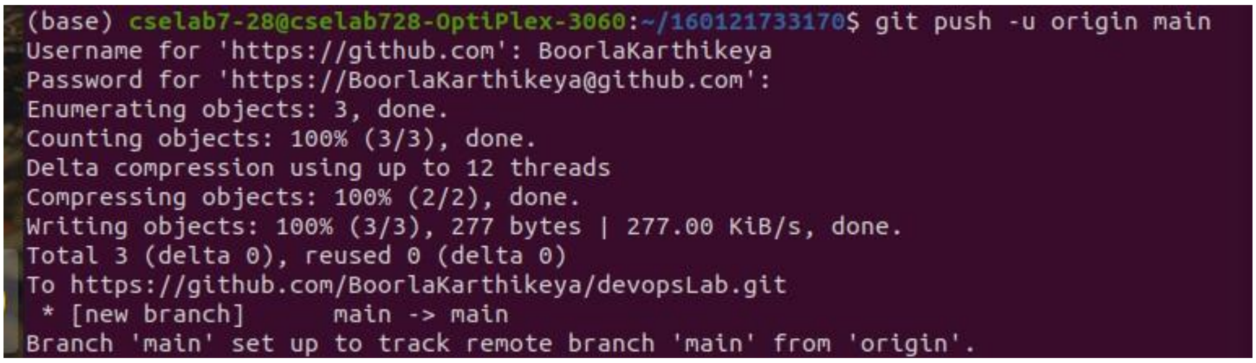
```
(base) cselab7-28@cselab728-OptiPlex-3060:~$ sudo apt-get install git
[sudo] password for cselab7-28:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.11).
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 gyp lib32gcc1 libfwupdplugin1 libjs-inherits
  libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libnode-dev
  libnode64 libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib
  libssl-dev libuv1-dev libxmlb1 nodejs-doc python-pkg-resources python2
  python2-minimal python2.7 python2.7-minimal
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
(base) cselab7-28@cselab728-OptiPlex-3060:~$ git --version
git version 2.25.1
```

Step 2: Creating a Repository

- Create a New Directory:
  - Open a terminal or command prompt.
  - Navigate to the directory where you want to create your Git repository. You can use **cd** to change directories.
- Initialize Git Repository:
  - Run the command **git init**. This initializes a new Git repository in the current directory.
- Add Files:
  - Place the files you want to track in the repository into this directory.
  - Use the command **git add <file>** to add files to the staging area.
- Commit Changes:
  - After adding files, commit them to the repository with **git commit -a -m "Your commit message"**.



```
(base) cselab7-28@cselab728-OptiPlex-3060:~$ mkdir 160121733170
(base) cselab7-28@cselab728-OptiPlex-3060:~$ cd 160121733170
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ touch demo.txt
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ vi demo.txt
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git init
Initialized empty Git repository in /home/cselab7-28/160121733170/.git/
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git add .
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git status
On branch master

No commits yet

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

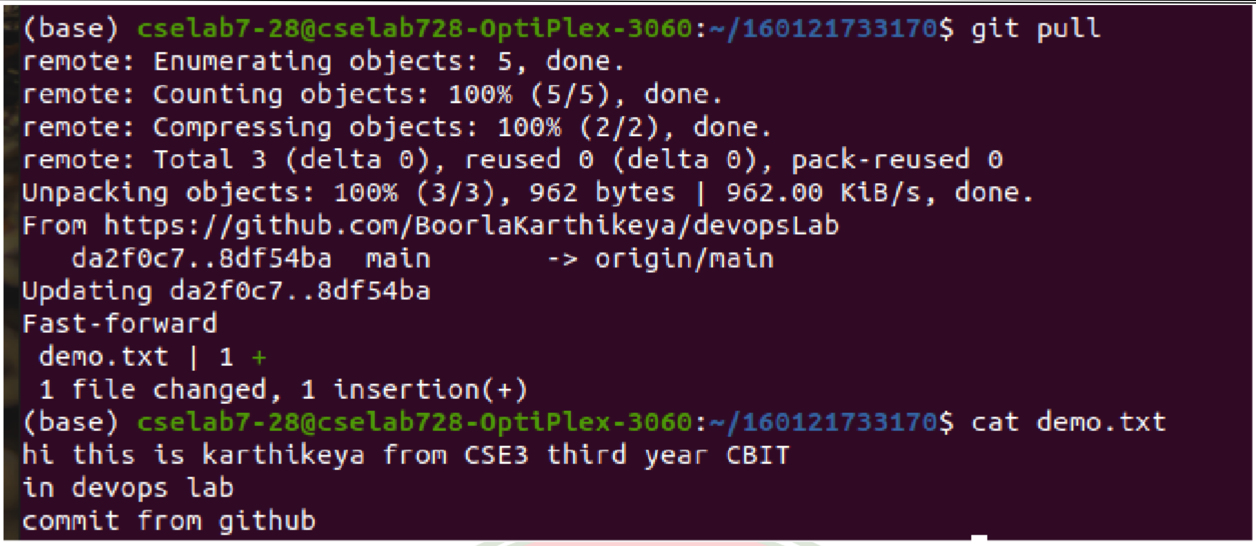


```
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git push -u origin main
Username for 'https://github.com': BoorlaKarthikeya
Password for 'https://BoorlaKarthikeya@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 277 bytes | 277.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/BoorlaKarthikeya/devopsLab.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Step 3: Basic Git Operations

Pull:

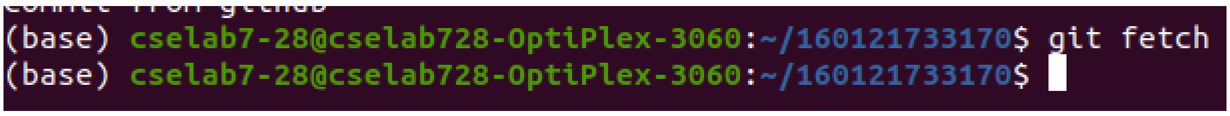
- To fetch changes from a remote repository and merge them into your current branch, use **git pull**



```
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 962 bytes | 962.00 KiB/s, done.
From https://github.com/BoorlaKarthikeya/devopsLab
   da2f0c7..8df54ba  main       -> origin/main
Updating da2f0c7..8df54ba
Fast-forward
 demo.txt | 1 +
 1 file changed, 1 insertion(+)
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ cat demo.txt
hi this is karthikeya from CSE3 third year CBIT
in devops lab
commit from github
```

Fetch:

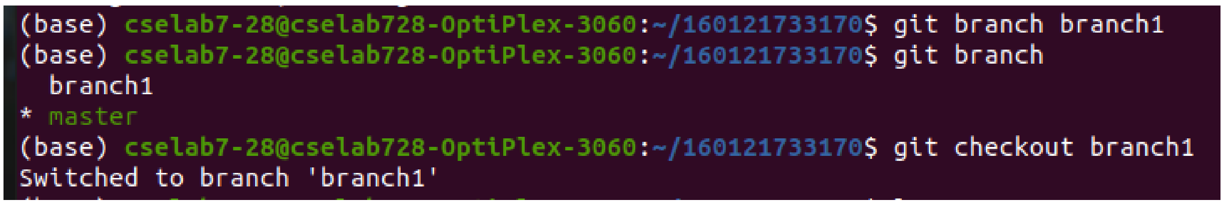
- To fetch changes from a remote repository, use **git fetch**



```
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git fetch
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ 
```

Branching:

- To create a new branch, use **git branch <branch_name>**.
- To switch to a different branch, use **git checkout <branch_name>**.
- To create and switch to a new branch simultaneously, use **git checkout -b <branch_name>**.
-



```
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git branch branch1
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git branch
  branch1
* master
(base) cselab7-28@cselab728-OptiPlex-3060:~/160121733170$ git checkout branch1
Switched to branch 'branch1'
```

AIM : Jenkins Installation and implement continues Integration and Continues deployment, build a job using Jenkins.

DESCRIPTION :

- Jenkins is an open-source automation server used for continuous integration and continuous delivery (CI/CD) pipelines.
- It allows developers to automate various stages of the software development process, including building, testing, and deploying applications.
- Jenkins supports integration with various version control systems such as Git, SVN, and Mercurial.
- It offers a web-based interface for easy configuration and management of jobs and pipelines.
- Jenkins provides a wide range of plugins to extend its functionality, allowing integration with different tools and technologies.
- With Jenkins, developers can schedule and trigger builds based on code commits, time intervals, or external events.
- It supports distributed builds, allowing users to distribute work across multiple machines or nodes.
- Jenkins offers robust security features, including role-based access control and support for various authentication mechanisms.
- It provides extensive logging and monitoring capabilities to track build statuses, logs, and performance metrics.
- Jenkins is highly customizable and scalable, making it suitable for projects of any size and complexity.

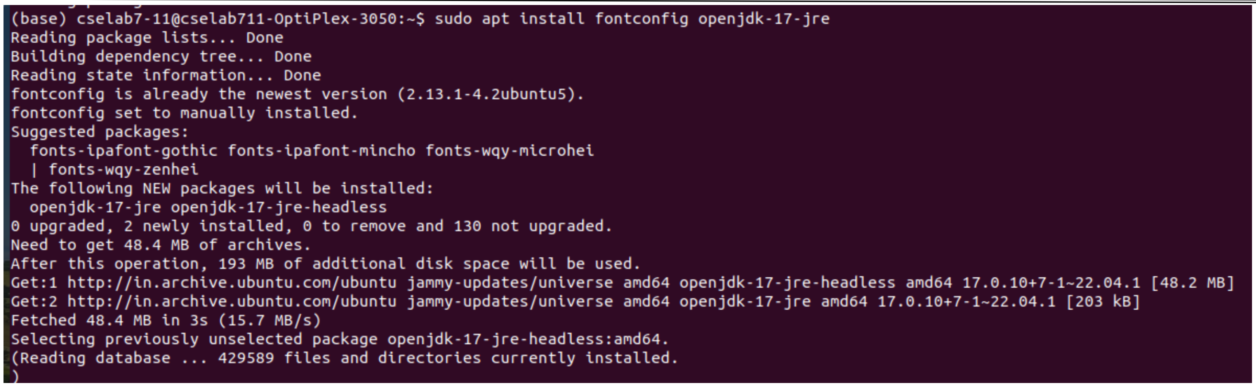
PROCEDURE :

Step 1: Update Package Repository
Before installing Jenkins, it's a good practice to update the package repository on your Linux system:

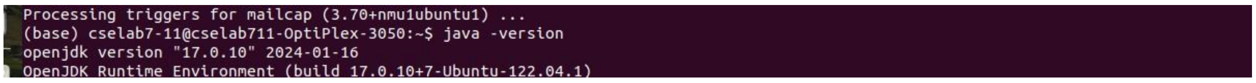
```
Sudo apt update
```

Step 2: Install Java
Jenkins requires Java to run. You can install OpenJDK, which is an open-source implementation of the Java Platform:

```
sudo apt install fontconfig openjdk-17-jre
java -version
```



```
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo apt install fontconfig openjdk-17-jre
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
fontconfig is already the newest version (2.13.1-4.2ubuntu5).
fontconfig set to manually installed.
Suggested packages:
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  | fonts-wqy-zenhei
The following NEW packages will be installed:
  openjdk-17-jre openjdk-17-jre-headless
0 upgraded, 2 newly installed, 0 to remove and 130 not upgraded.
Need to get 48.4 MB of archives.
After this operation, 193 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre-headless amd64 17.0.10+7-1~22.04.1 [48.2 MB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre amd64 17.0.10+7-1~22.04.1 [203 kB]
Fetched 48.4 MB in 3s (15.7 MB/s)
Selecting previously unselected package openjdk-17-jre-headless:amd64.
(Reading database ... 429589 files and directories currently installed.
)
```

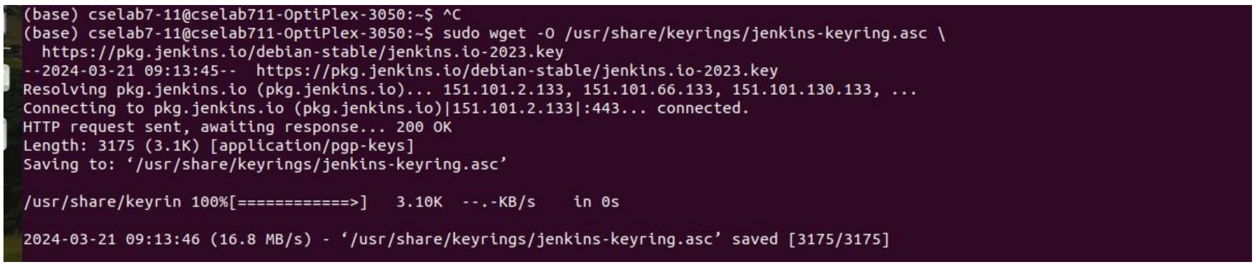


```
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
(base) cselab7-11@cselab711-OptiPlex-3050:~$ java -version
openjdk version "17.0.10" 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
```

Step 3: Add Jenkins Repository Key

To ensure the authenticity of the Jenkins packages, add the Jenkins repository key to your system:

**sudo wget -O /usr/share/keyrings/jenkins-keyring.asc **
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

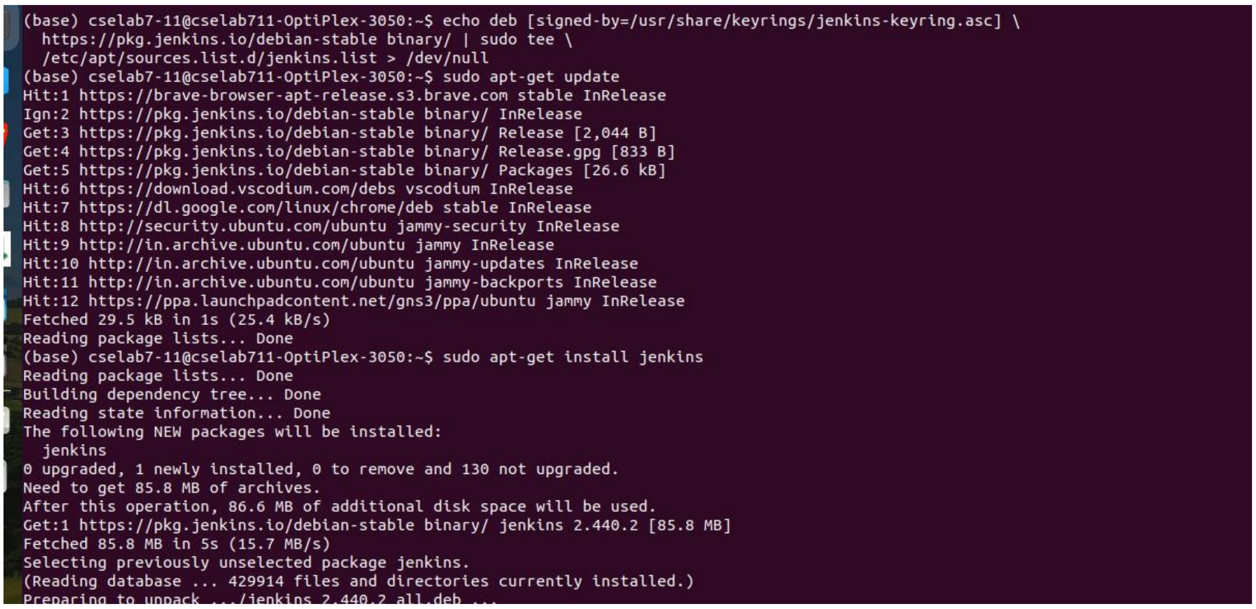


```
(base) cselab7-11@cselab711-OptiPlex-3050:~$ ^C
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2024-03-21 09:13:45--  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.2.133, 151.101.66.133, 151.101.130.133, ...
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.2.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: ‘/usr/share/keyrings/jenkins-keyring.asc’

/usr/share/keyrin 100%[============>]   3.10K  --.-KB/s    in 0s

2024-03-21 09:13:46 (16.8 MB/s) - ‘/usr/share/keyrings/jenkins-keyring.asc’ saved [3175/3175]
```

Step 4: Add Jenkins Repository

Add the Jenkins repository to your system's list of package sources:

'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'



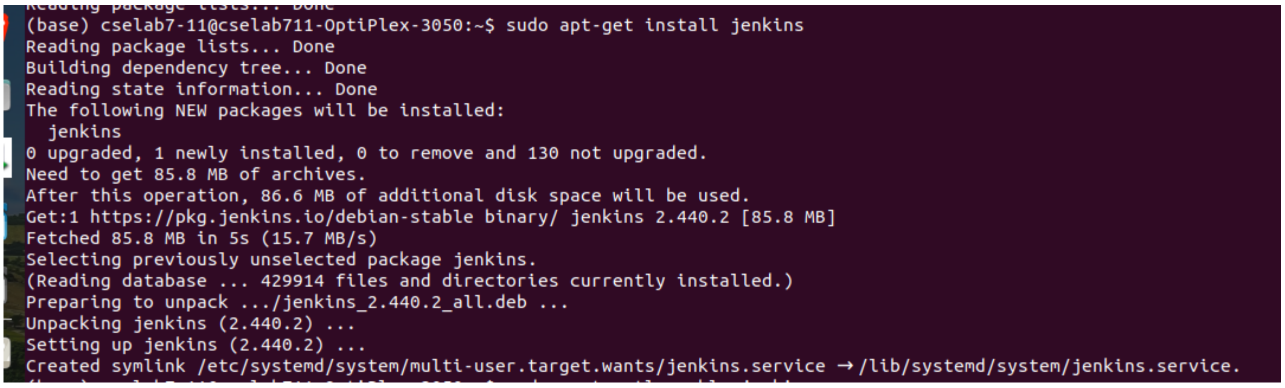
```
(base) cselab7-11@cselab711-OptiPlex-3050:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo apt-get update
Hit:1 https://brave-browser-apt-release.s3.brave.com stable InRelease
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]
Get:4 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:5 https://pkg.jenkins.io/debian-stable binary/ Packages [26.6 kB]
Hit:6 https://download.vscodium.com/debs vscodium InRelease
Hit:7 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:8 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:9 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:10 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:11 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:12 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Fetched 29.5 kB in 1s (25.4 kB/s)
Reading package lists... Done
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 130 not upgraded.
Need to get 85.8 MB of archives.
After this operation, 86.6 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.440.2 [85.8 MB]
Fetched 85.8 MB in 5s (15.7 MB/s)
Selecting previously unselected package jenkins.
(Reading database ... 429914 files and directories currently installed.)
Preparing to unpack .../jenkins_2.440.2_all.deb ...
```

Step 5: Install Jenkins

Now, update the package repository again and install Jenkins:

sudo apt-get update

sudo apt-get install jenkins



```
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 130 not upgraded.
Need to get 85.8 MB of archives.
After this operation, 86.6 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.440.2 [85.8 MB]
Fetched 85.8 MB in 5s (15.7 MB/s)
Selecting previously unselected package jenkins.
(Reading database ... 429914 files and directories currently installed.)
Preparing to unpack .../jenkins_2.440.2_all.deb ...
Unpacking jenkins (2.440.2) ...
Setting up jenkins (2.440.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
```

Step 6 : Start Jenkins

You can enable the Jenkins service to start at boot with the command:

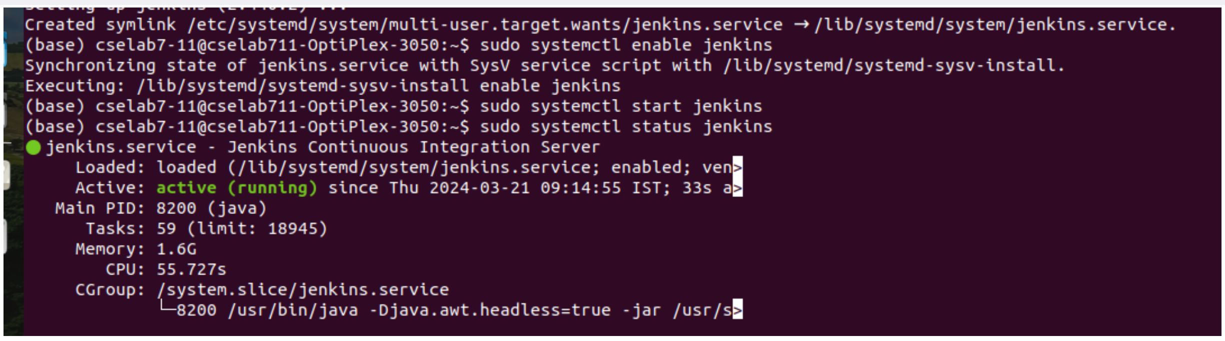
```
sudo systemctl enable jenkins
```

You can start the Jenkins service with the command:

```
sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

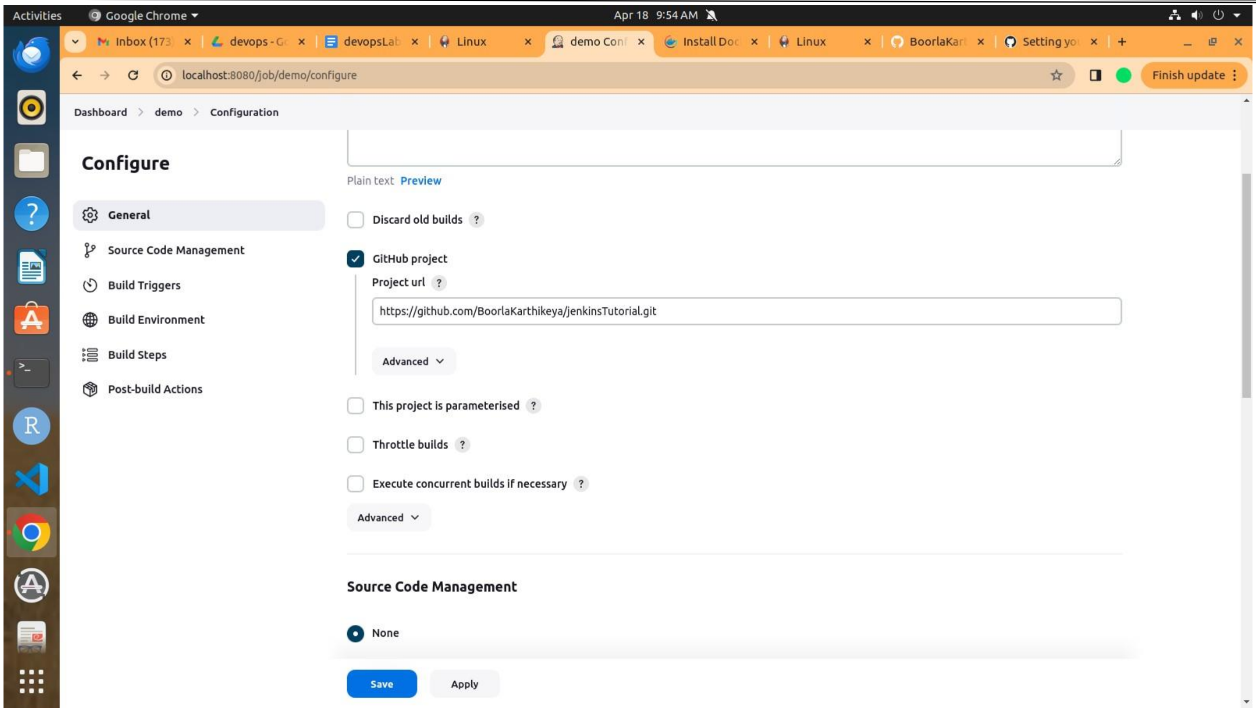
```
sudo systemctl status Jenkins
```



```
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable jenkins
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo systemctl start jenkins
(base) cselab7-11@cselab711-OptiPlex-3050:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; ven>
     Active: active (running) since Thu 2024-03-21 09:14:55 IST; 33s a>
   Main PID: 8200 (java)
      Tasks: 59 (limit: 18945)
     Memory: 1.6G
        CPU: 55.727s
     CGroup: /system.slice/jenkins.service
             └─8200 /usr/bin/java -Djava.awt.headless=true -jar /usr/s>
```

Step7 : configure jenkins

Dashboard -> new Item -> github url -> save -> build -> console output

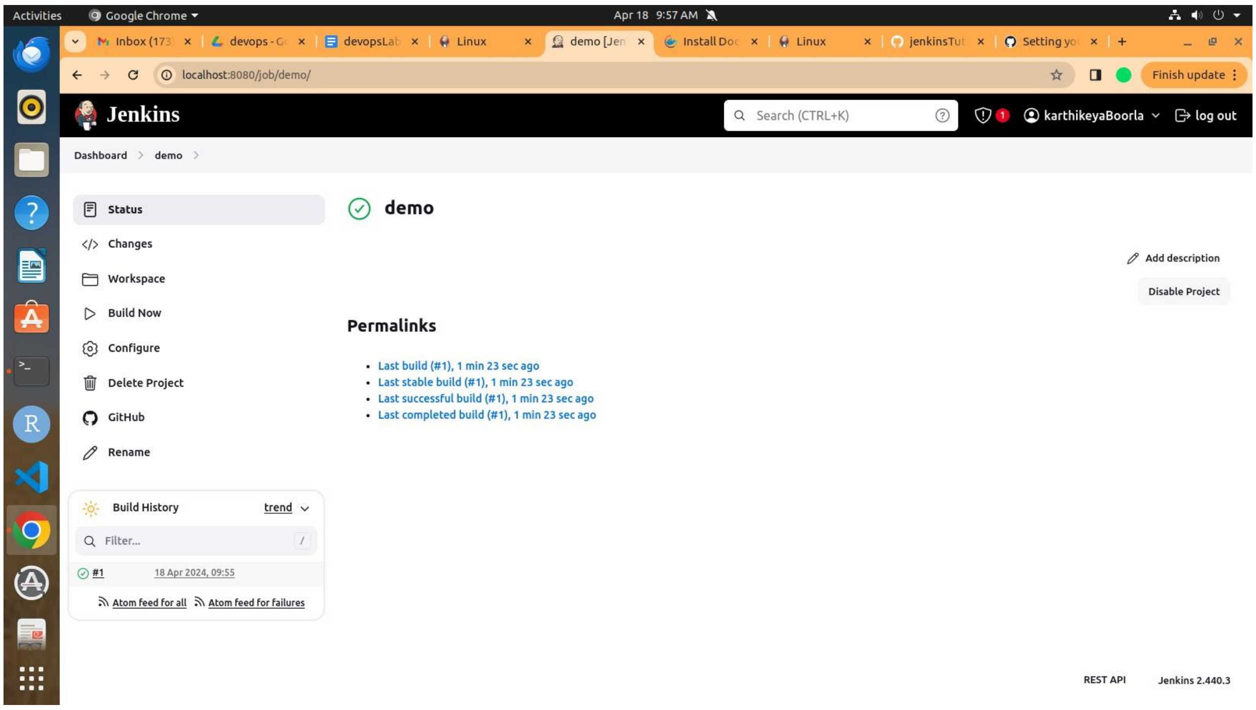


- [ ] Discard old builds
- [x] GitHub project

Project url: https://github.com/BoorlaKarthikeya/jenkinsTutorial.git

- [ ] This project is parameterised
- [ ] Throttle builds
- [ ] Execute concurrent builds if necessary

Source Code Management: [✓ None]



demo

Permalinks

- Last build (#1), 1 min 23 sec ago
- Last stable build (#1), 1 min 23 sec ago
- Last successful build (#1), 1 min 23 sec ago
- Last completed build (#1), 1 min 23 sec ago

Build History: #1 18 Apr 2024, 09:55

Console Output

```
Started by user karthikeyaBoorla
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/demo
Finished: SUCCESS
```

AIM : To install and configure Docker for creating containers of different Operating System (Virtualization Concept)

DESCRIPTION :

- Docker is a platform for developing, shipping, and running applications in containers.
- Containers are lightweight, portable, and self-sufficient units that encapsulate all the dependencies and libraries required for an application to run.
- Docker simplifies the process of packaging applications and their dependencies into containers, ensuring consistency across different environments.
- It uses containerization technology to isolate applications from the underlying infrastructure, enabling easy deployment across various environments, including development, testing, and production.
- Docker provides a command-line interface (CLI) and a set of tools for building, managing, and orchestrating containers.
- It utilizes a client-server architecture, where the Docker daemon manages container lifecycle operations and interacts with the Docker client.
- Docker images serve as blueprints for containers, containing everything needed to run an application, including the code, runtime, libraries, and dependencies.
- Docker Hub is a cloudbased registry service provided by Docker, offering a vast collection of public and private Docker images that developers can use as base images for their containers.
- Docker Compose is a tool for defining and running multicontainer Docker applications, simplifying the management of complex applications composed of multiple interconnected containers.
- Docker Swarm and Kubernetes are popular container orchestration platforms that can be used with Docker to automate deployment, scaling, and management of containerized applications across clusters of machines.

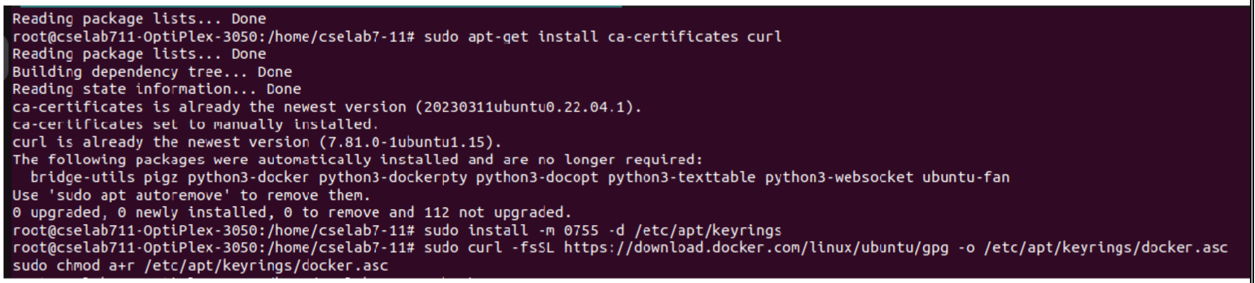
PROCECURE :

Step1 :

Add Docker's official GPG key:

- This command retrieves Docker's GPG key and adds it to your system's keyring for package verification.
- ***sudo apt-get update***: Updates the package index to ensure the availability of the latest packages.

- ***sudo apt-get install ca-certificates curl***: Installs the necessary dependencies, including ca-certificates and curl, required to fetch the GPG key securely.
- ***sudo install -m 0755 -d /etc/apt/keyrings***: Creates a directory /etc/apt/keyrings with the appropriate permissions (0755).
- s***udo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc***: Downloads Docker's GPG key from the specified URL and saves it as docker.asc in the /etc/apt/keyrings directory.
- ***sudo chmod a+r /etc/apt/keyrings/docker.asc***: Changes the permissions of the downloaded GPG key to make it readable (a+r) by all users.

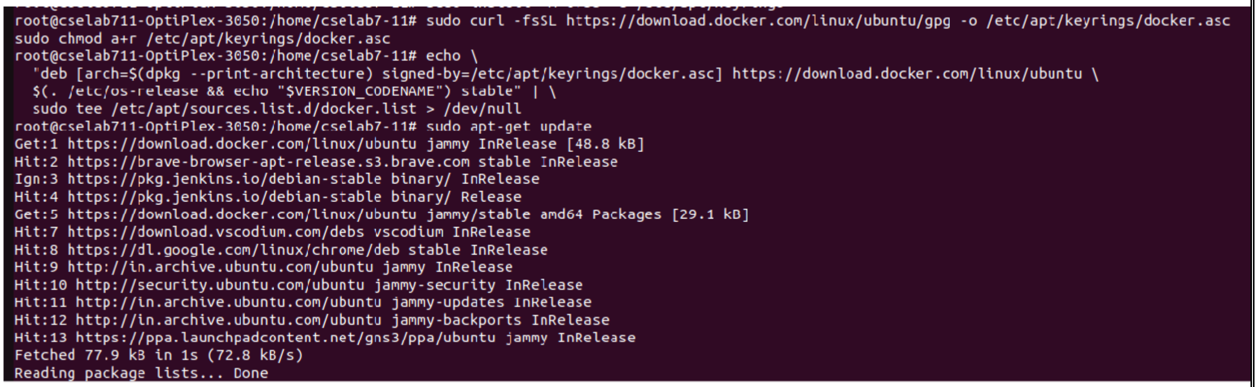


```
Reading package lists... Done
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.15).
The following packages were automatically installed and are no longer required:
  bridge-utils pigz python3-docker python3-dockerpty python3-docopt python3-texttable python3-websocket ubuntu-fan
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 112 not upgraded.
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo install -m 0755 -d /etc/apt/keyrings
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Step 2 : Add the repository to APT sources:

- This command adds Docker's repository to the APT sources list, allowing you to install Docker packages using apt-get.
- echo ...: Echoes the Docker repository information into a new file named ***docker.list under /etc/apt/sources.list.d/.***
- ***"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] ...***: Specifies the Docker repository URL and its configuration.
- ***$(. /etc/os-release && echo "$VERSION_CODENAME")***: Dynamically retrieves the Ubuntu version codename ($VERSION_CODENAME) from the /etc/os-release file.
- ***sudo tee /etc/apt/sources.list.d/docker.list > /dev/null***: Writes the echoed Docker repository information to the docker.list file and redirects the standard output to /dev/null to suppress any output.

- sudo apt-get update: Updates the package index again to include the newly added Docker repository.

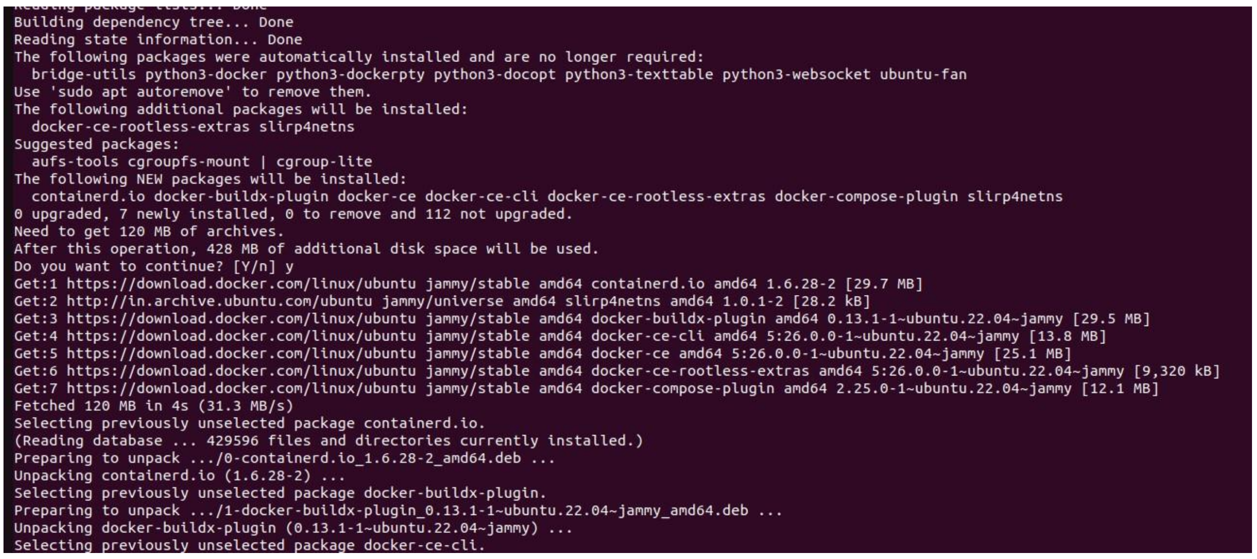


```
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
root@cselab711-OptiPlex-3050:/home/cselab7-11# echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:2 https://brave-browser-apt-release.s3.brave.com stable InRelease
Ign:3 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 https://pkg.jenkins.io/debian-stable binary/ Release
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [29.1 kB]
Hit:7 https://download.vscodium.com/debs vscodium InRelease
Hit:8 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:9 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:10 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:11 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:12 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:13 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Fetched 77.9 kB in 1s (72.8 kB/s)
Reading package lists... Done
```

Step 3 : Install docker :

- To install the latest version, run:

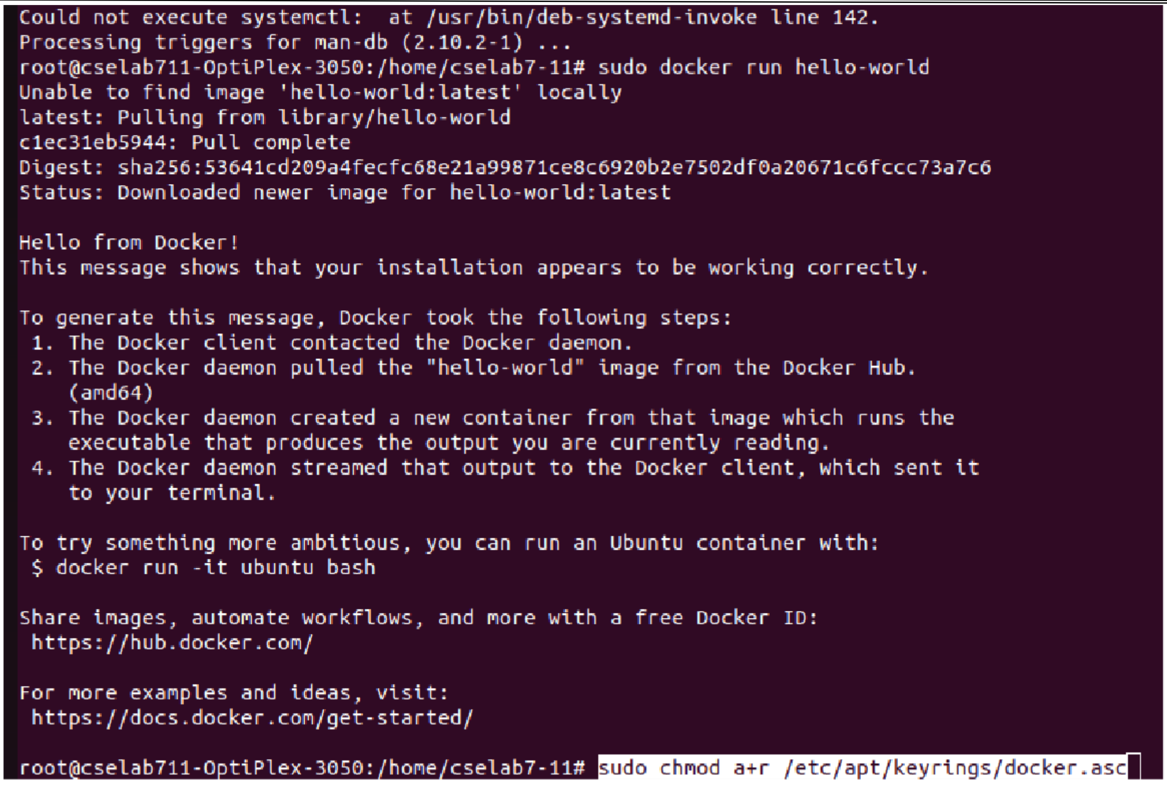
  sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin



```
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bridge-utils python3-docker python3-dockerpty python3-docopt python3-texttable python3-websocket ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  docker-ce-rootless-extras slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin slirp4netns
0 upgraded, 7 newly installed, 0 to remove and 112 not upgraded.
Need to get 120 MB of archives.
After this operation, 428 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.6.28-2 [29.7 MB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64 1.0.1-2 [28.2 kB]
Get:3 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-buildx-plugin amd64 0.13.1-1~ubuntu.22.04~jammy [29.5 MB]
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli amd64 5:26.0.0-1~ubuntu.22.04~jammy [13.8 MB]
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:26.0.0-1~ubuntu.22.04~jammy [25.1 MB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:26.0.0-1~ubuntu.22.04~jammy [9,320 kB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 2.25.0-1~ubuntu.22.04~jammy [12.1 MB]
Fetched 120 MB in 4s (31.3 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 429596 files and directories currently installed.)
Preparing to unpack .../0-containerd.io_1.6.28-2_amd64.deb ...
Unpacking containerd.io (1.6.28-2) ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../1-docker-buildx-plugin_0.13.1-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-buildx-plugin (0.13.1-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package docker-ce-cli.
```

- Verify that the Docker Engine installation is successful by running the hello-world image.

  sudo docker run hello-world



```
Could not execute systemctl:  at /usr/bin/deb-systemd-invoke line 142.
Processing triggers for man-db (2.10.2-1) ...
root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:53641cd209a4fecfc68e21a99871ce8c6920b2e7502df0a20671c6fccc73a7c6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@cselab711-OptiPlex-3050:/home/cselab7-11# sudo chmod a+r /etc/apt/keyrings/docker.asc
```

AIM : Deployment Tool (Team City /Ansible) Install Docker and execute commands in a Docker and deploy the application in to Docker file

DESCRPTION :

Containerization is a technology that allows you to package an application and its dependencies into a single unit called a container. Docker is one of the most popular containerization platforms. Here's how containerization and deployment using Docker typically work:

1. Containerization:

- Developers define a Dockerfile, which specifies the environment and dependencies required for their application.
- Docker builds an image based on the Dockerfile, encapsulating the application code, runtime, libraries, and dependencies.
- This image serves as a portable and self-contained unit that can run consistently across different environments.

2. Deployment:

- Once the Docker image is built, it can be deployed to any environment that has Docker installed, whether it's a developer's laptop, a testing server, or a production cluster.
- Docker images can be pushed to a registry such as Docker Hub, where they can be shared with others or pulled onto different machines for deployment.
- Using tools like Docker Compose, developers can define multi-container applications and their interdependencies in a single configuration file, making it easy to manage complex applications.
- Orchestration platforms like Docker Swarm or Kubernetes can be used to automate deployment, scaling, and management of containerized applications across clusters of machines.

Benefits of containerization and deployment using Docker:

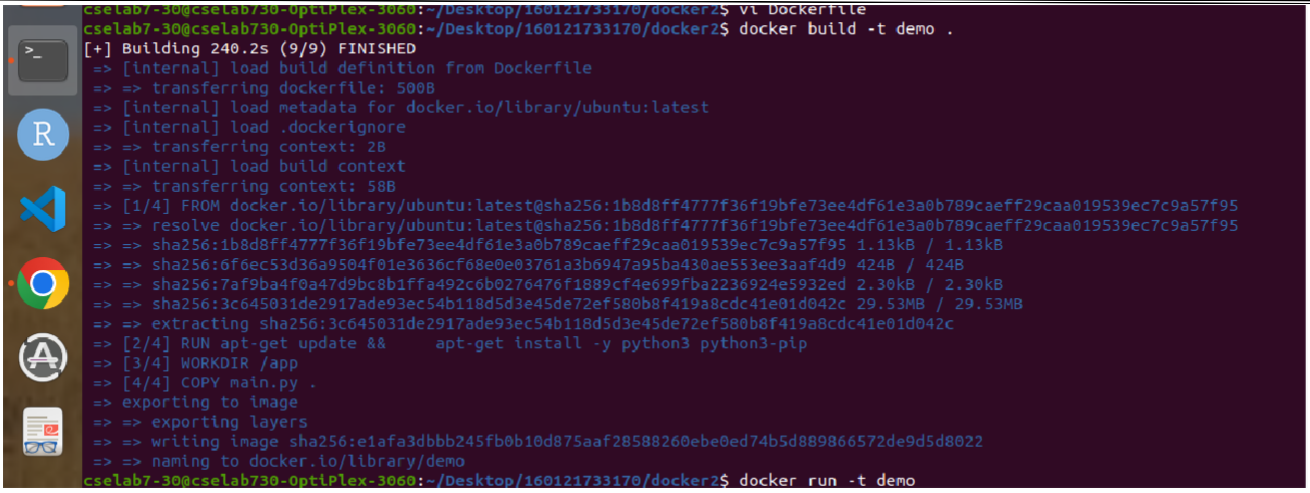
- **Consistency**: Containers ensure that applications run the same way in different environments, reducing the risk of "it works on my machine" issues.
- **Isolation**: Containers isolate applications from the underlying infrastructure, providing security and ensuring that changes to one application don't affect others.
- **Portability:** Docker images are portable and can be easily moved between environments, making it simple to deploy applications consistently across development, testing, and production environments.
- **Efficiency:** Containers are lightweight and start quickly, enabling fast deployment and scaling of applications.
- **Scalability**: Docker enables horizontal scaling by deploying multiple instances of containers, allowing applications to handle increased load easily.
- **Resource Utilization**: Containers share the host OS kernel, resulting in efficient resource utilization and allowing for higher density of applications on the same hardware compared to traditional virtualization.

PROCEDURE :

Step 4 : create a Dockerfile for container

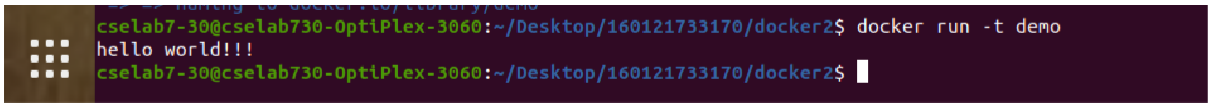
```
FROM ubuntu:latest

# Update package lists and install Python

RUN apt-get update && \

    apt-get install -y python3 python3-pip


# Set the working directory inside the container

WORKDIR /app


# Copy the Python script into the container

COPY main.py .


# Install any Python dependencies if needed (uncomment and adjust as necessary)

# COPY requirements.txt .

# RUN pip3 install -r requirements.txt


# Command to run the Python script

CMD ["python3", "main.py"]
```

Step 5 : build docker container



```
cselab7-30@cselab730-OptiPlex-3060:~/Desktop/160121733170/docker2$ vi Dockerfile
cselab7-30@cselab730-OptiPlex-3060:~/Desktop/160121733170/docker2$ docker build -t demo .
[+] Building 240.2s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 500B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 58B
 => [1/4] FROM docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95
 => => resolve docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95
 => => sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95 1.13kB / 1.13kB
 => => sha256:6f6ec53d36a9504f01e3636cf68e0e03761a3b6947a95ba430ae553ee3aaf4d9 424B / 424B
 => => sha256:7af9ba4f0a47d9bc8b1ffa492c6b0276476f1889cf4e699fba2236924e5932ed 2.30kB / 2.30kB
 => => sha256:3c645031de2917ade93ec54b118d5d3e45de72ef580b8f419a8cdc41e01d042c 29.53MB / 29.53MB
 => => extracting sha256:3c645031de2917ade93ec54b118d5d3e45de72ef580b8f419a8cdc41e01d042c
 => [2/4] RUN apt-get update &&     apt-get install -y python3 python3-pip
 => [3/4] WORKDIR /app
 => [4/4] COPY main.py .
 => exporting to image
 => => exporting layers
 => => writing image sha256:e1afa3dbbb245fb0b10d875aaf28588260ebe0ed74b5d889866572de9d5d8022
 => => naming to docker.io/library/demo
cselab7-30@cselab730-OptiPlex-3060:~/Desktop/160121733170/docker2$ docker run -t demo
```

Step 6 : run docker container



```
cselab7-30@cselab730-OptiPlex-3060:~/Desktop/160121733170/docker2$ docker run -t demo
hello world!!!
cselab7-30@cselab730-OptiPlex-3060:~/Desktop/160121733170/docker2$ 
```