DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

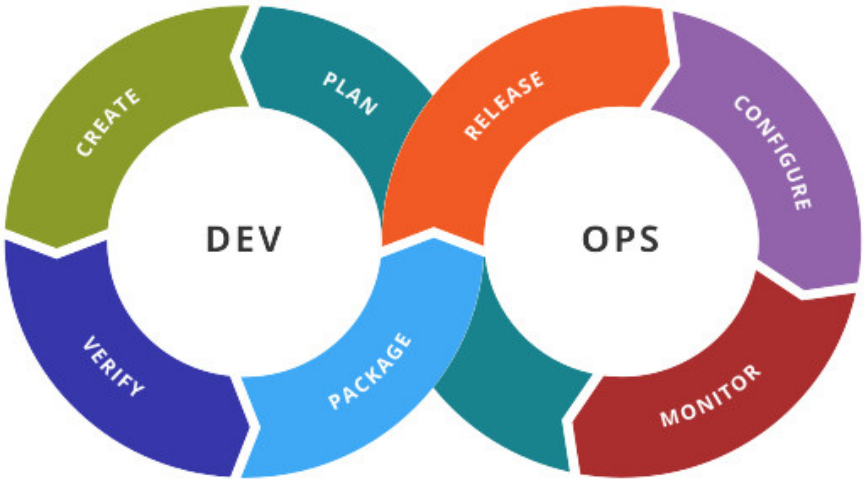
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**01**
DOP	**20/01/2023**
DOS	

Core DevOps principles

The DevOps methodology comprises four key principles that guide the effectiveness and efficiency of application development and deployment. These principles, listed below, center on the best aspects of modern software development.

Automation of the software development lifecycle
Collaboration and communication
Continuous improvement and minimization of waste
Hyperfocus on user needs with short feedback loops
By adopting these principles, organizations can improve code quality, achieve a faster time to market, and engage in better application planning.

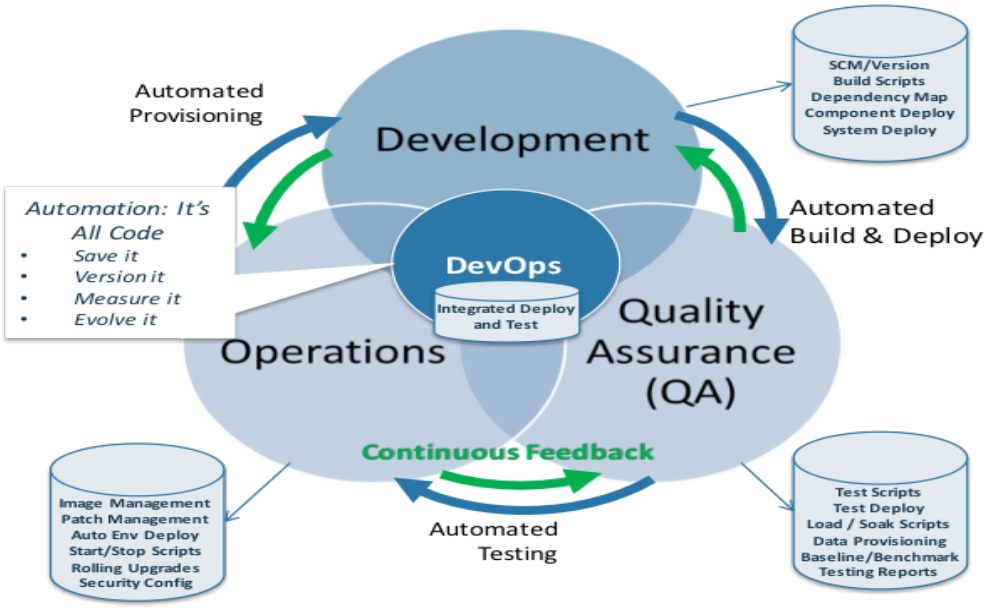


DevOps practices

DevOps practices reflect the idea of continuous improvement and automation. Many practices focus on one or more development cycle phases. These practices include:

- **Continuous development.** This practice spans the planning and coding phases of the DevOps lifecycle. Version-control mechanisms might be involved.
- **Continuous integration (CI).** This practice brings configuration management (CM) tools together with other test and development tools to track how much of the code being developed is ready for production.
- **Continuous delivery.** This practice automates the delivery of code changes, after testing, to a pre production or staging environment.
- **Continuous deployment (CD).** Similar to continuous delivery, this practice automates the release of new or changed code into production.
- **Continuous monitoring.** This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it.

- **Infrastructure as code.** This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release.



DevOps practices

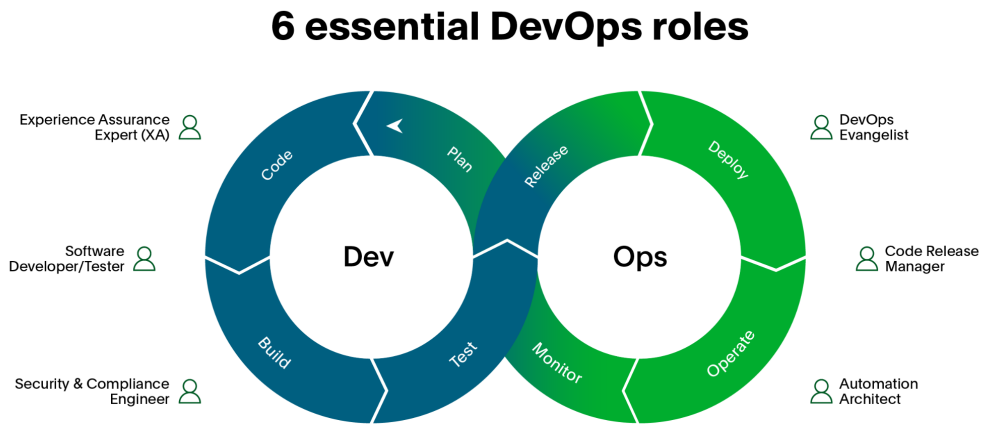
What Is a DevOps Engineer?

A DevOps engineer is responsible for the smooth operation of a company's IT infrastructure. They work with developers to deploy and manage code changes, and with operations staff to ensure that systems are up and running smoothly. To be successful in this role, a DevOps engineer must have a deep understanding of both development and operations processes, as well as a strong technical background.

DevOps Engineer Job Description: Roles and Responsibilities

A DevOps engineer's roles and responsibilities are a combination of technical and management roles. It is essential to have excellent communication and coordination skills to successfully integrate various functions in a coordinated manner and deliver the responsibilities to the customer's satisfaction.

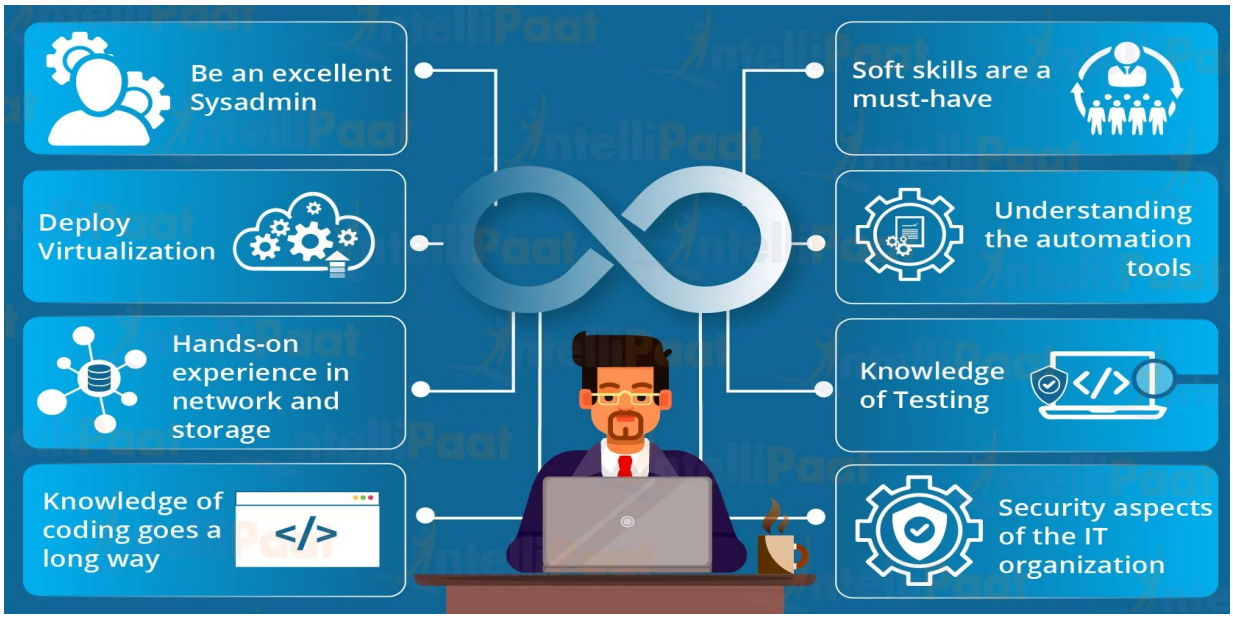
The DevOps engineer's responsibilities are multi-prong - they need to be agile enough to wear a technical hat and manage operations simultaneously.



Roles

Some of the core **responsibilities** of DevOps Engineer include –

- Understanding customer requirements and project KPIs
- Implementing various development, testing, automation tools, and IT infrastructure
- Encouraging and building automated processes wherever possible
- Incidence management and root cause analysis
- Coordination and communication within the team and with customers
- Selecting and deploying appropriate CI/CD tools
- Mentoring and guiding the team members
- Monitoring and measuring customer experience and KPIs
- Managing periodic reporting on the progress to the management and the customer



Responsibilities

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**02**
DOP	**27/01/2023**
DOS	

What is a “version control system”?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Why is the Version Control system so Important?

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes to some specific kind of functionality/features. So in order to contribute to the product, they made modifications to the source code(either by adding or removing).

Types of Version Control Systems:

- **Local Version Control Systems**
- **Centralized Version Control Systems**
- **Distributed Version Control Systems**

Local Version Control Systems: It is one of the simplest forms and has a database that keeps all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.

Centralized Version Control Systems: Centralized version control systems contain just one repository globally and every user needs to commit for reflecting one’s changes in the repository. It is possible for others to see your changes by updating.

Two things are required to make your changes visible to others which are:

You commit

They update

Steps to install Git on Linux :

- **Step 1 - $ sudo apt-get install git**
- **Step 2 - $ sudo apt-get install libcurl4-gnutls-dev libexpat1-dev gettext libz-dev libssl-dev asciidoc xmlto docbook2x**
- **Step 3 - Initial Configuration**

Git is by default installed under the /usr/bin/git directory on recent Linux systems.

Once the installation is done, verify it by using the following command –

$ whereis git

The output should be like this –

git: /usr/bin/git /usr/share/man/man1/git.1.gz

To get the version number of Git, you can use the following command –

$ git --version

The output will be like this –

```
(base) computer@computer-ThinkCentre:~$ git --version
git version 2.34.1
```

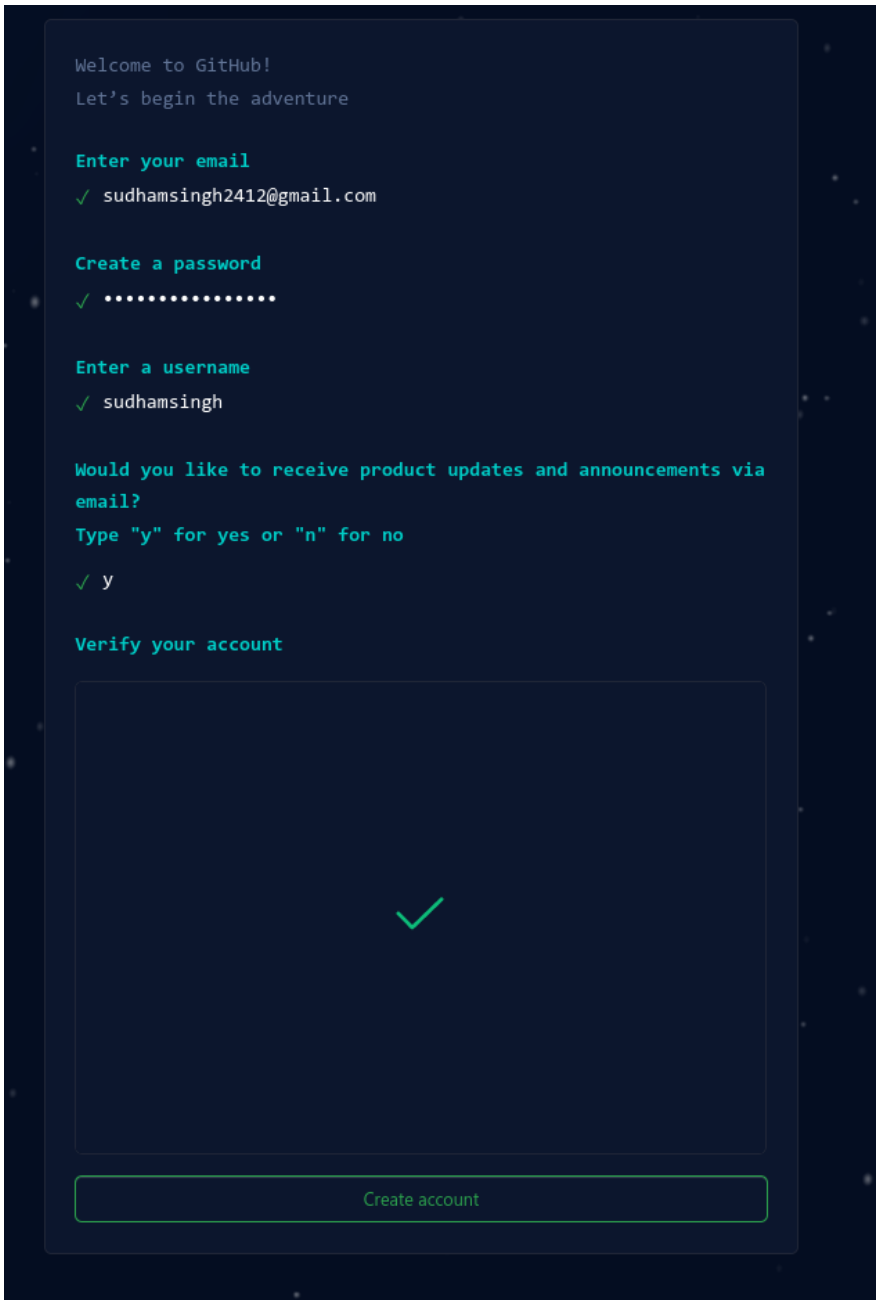
Steps to open an account on Github :

Step 1 - Go to “https://github.com”

Step 2 - Choose Sign up for GitHub

Step 3 - Enter you email

Step 4 - Create a password



Step 5 - Click on Create account

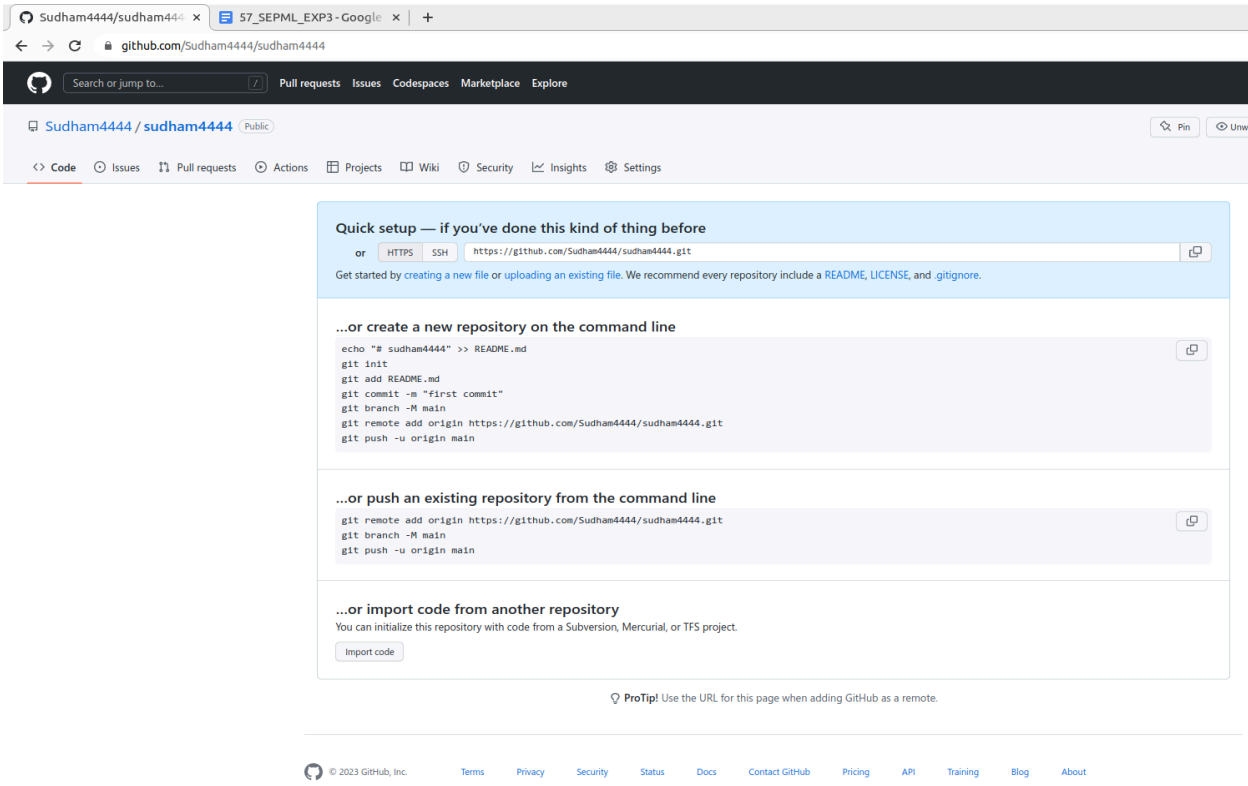
Step 6 - Then enter the OTP, that you got on your email account

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	
DOP	
DOS	

Create new repository on github :



Steps(git) :

Step 0 : Check version (git installed or not) : **git --version**
Step 1 : touch exp3.txt
Step 2 : touch exp3a.txt
Step 3 : git init
Step 4 : git init
Step 5 : git remote add origin https://github.com/Sudham4444/sudham4444
Step 6 : git remote add origin https://github.com/Sudham4444/sudham4444
Step 7 : git status
Step 8 : git add .
Step 9 : git status
Step 10 : git commit -m "hello_sudham"
Step 11 : git config --global user.email "singhsudham4444@gmail.com"
Step 12 : git config --global user.name "Sudham44444"
Step 13 : git push -u origin master
Enter username and password
Step 14 : if authentication failed, then get a new token from your github account

Procedure :

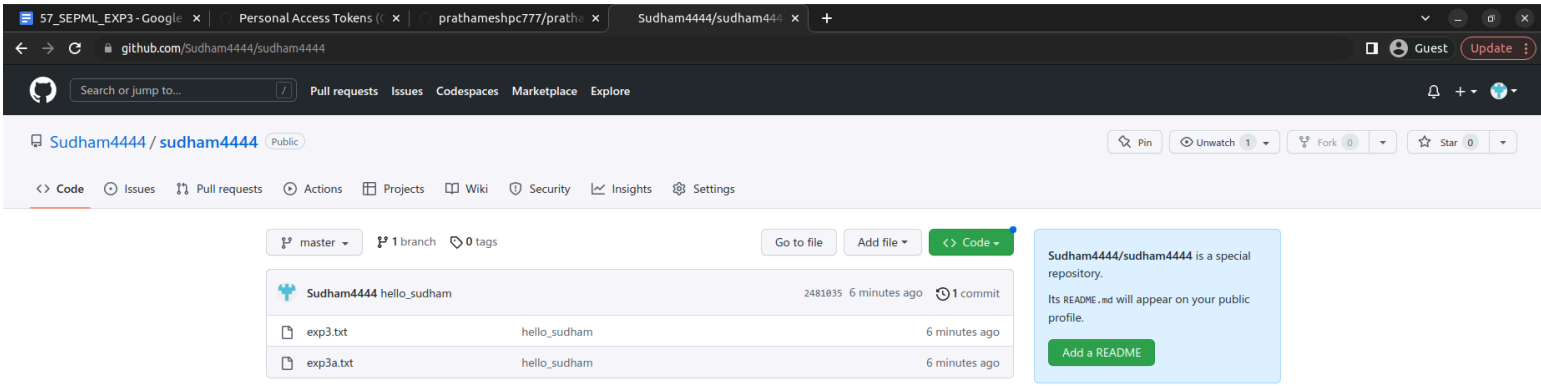
1. Go to your github account
2. Settings
3. <> Developer settings
4. Get a new token

Step 15 : git push -u origin master
Enter username and password

```
computer@computer-ThinkCentre: ~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git --version
git version 2.34.1
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ touch exp3.txt
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ touch exp3a.txt
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/computer/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML/.git/
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git init
Reinitialized existing Git repository in /home/computer/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML/.git/
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git remote add origin https://github.com/Sudham4444/sudham4444.git
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git remote add origin https://github.com/Sudham4444/sudham4444.git
error: remote origin already exists.
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        exp3.txt
        exp3a.txt

nothing added to commit but untracked files present (use "git add" to track)
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git add .
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   exp3.txt
        new file:   exp3a.txt
```

```
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git config --global user.email "singhsudham4444@gmail.com"
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git config --global user.name "Sudham4444"
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git commit -m "hello_sudham"
[master (root-commit) 2481035] hello_sudham
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 exp3.txt
 create mode 100644 exp3a.txt
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git config --global user.email "singhsudham4444@gmail.com"
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git config --global user.name "Sudham4444"
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git push -u origin master
Username for 'https://github.com': Sudham4444
Password for 'https://Sudham4444@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of
authentication.
fatal: Authentication failed for 'https://github.com/Sudham4444/sudham4444.git/'
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$ git push -u origin master
Username for 'https://github.com': Sudham4444
Password for 'https://Sudham4444@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), 217 bytes | 217.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sudham4444/sudham4444.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
(base) computer@computer-ThinkCentre:~/Documents/CSE-AIML/TE/SEM6/AIML57_SUDHAM/SEPML$
```

Output :


57_SEPML_EXP3 - Google
Personal Access Tokens (
prathameshpc777/pratha
Sudham4444/sudham444
github.com/Sudham4444/sudham4444
Guest
Update
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Sudham4444 / sudham4444
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
Sudham4444 hello_sudham
2481035 6 minutes ago
1 commit
exp3.txt
hello_sudham
6 minutes ago
exp3a.txt
hello_sudham
6 minutes ago
Sudham4444/sudham4444 is a special repository.
Its README.md will appear on your public profile.
Add a README

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**04**
DOP	**24/03/2023**
DOS	

Continuous integration

Introduction : Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

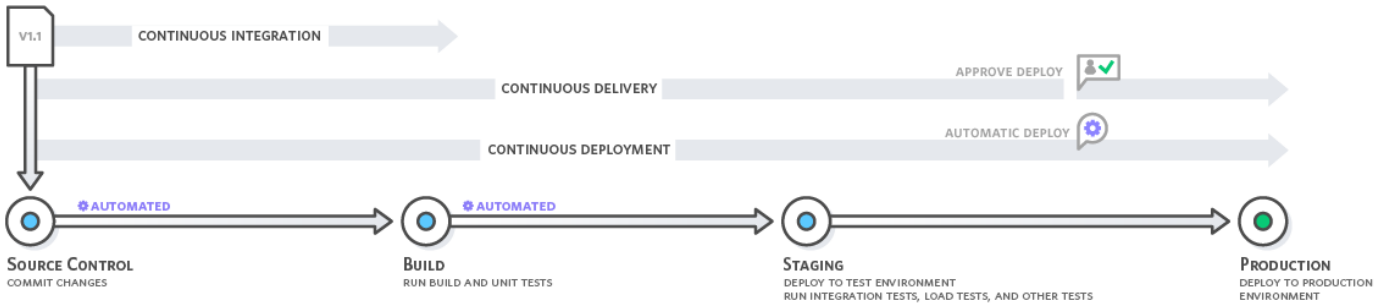
Uses :

1. Write tests for the most critical parts of the codebase
2. Run the tests automatically with a CI service on every push to the main repository
3. Make everyone in the team integrate their changes every day
4. As soon as the build is broken, fix it
5. For every new story that is implemented, write a test

Tools :

1. Bitbucket Pipelines :
    a. Easy setup and configuration
    b. Unified Bitbucket experience
    c. Cloud by 3rd party
2. Jenkins :
    a. On-premise
    b. Open source
    c. Robust addon / plugin ecosystem
3. CircleCI :
    a. Notification triggers from CI events
    b. Performance optimized for quick builds
    c. Easy debugging through SSH and local builds
    d. Analytics to measure build performance
4. Azure Pipelines :
    a. Azure platform integration
    b. Windows platform support
    c. Container support
    d. Github integration
5. GitLab
    a. On-prem or cloud hosting
    b. Continuous security testing
    c. Easy to learn UX
6. Atlassian Bamboo
    a. Best integration with Atlassian product suite
    b. An extensive market place of add-ons and plugins
    c. Container support with Docker agents
    d. Trigger API for IFTTT functionality

Working :



Importance :

To understand the importance of CI, here are some of its benefits:

1. Reduces Risk :

The frequent testing and deployment of code reduces the project's risk level, as now the code defects and bugs can be detected earlier. This states that these bugs and errors can be easily fixed and take less time, making the overall process cheaper. The general working speeds up the feedback mechanism that makes the communication smoother and effective.

2. Better Communication

The Continuous Integration process collaborates with the Continuous Delivery workflow that makes code sharing easy and regularized. This makes the process more transparent and collaborative among team members. In the long term, this makes the communication speed more efficient and makes sure that everyone in the organization is on the same page.

3. Higher Product Quality

Continuous Integration provides features like Code review and Code quality detection, making the identification of errors easy. If the code does not match the standard level or a mistake, it will be alerted with emails or SMS messages. Code review helps the developers to improve their programming skills continually.

4. Reduced Waiting Time

The time between the application development, integration, testing, and deployment is considerably reduced. When this time is reduced, it, in turn, reduces the waiting time that may occur in the middle. CI makes sure that all these processes continue to happen no matter what.

We came across three different terms, Continuous Integration, Continuous Deployment, and Continuous Delivery.

Install and Configure Jenkins with Maven

Step 1 : Install Java

Check if you already have java installed on your Ubuntu system: **java --version**

```
(base) computer@computer-ThinkCentre:~$ java -version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu222.04, mixed mode, sharing)
```

Step 2 : Install Jenkins

1. Updating refreshes the cache and makes the system aware of the new Jenkins repository : **sudo apt update**
2. Install Jenkins by running : **sudo apt install jenkins -y**

Wait for the download and installation to complete.

```
(base) computer@computer-ThinkCentre:~$ sudo apt-get update
[sudo] password for computer:
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:8 http://archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:9 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Hit:10 https://ppa.launchpadcontent.net/swi-prolog/stable/ubuntu jammy InRelease
Reading package lists... Done
(base) computer@computer-ThinkCentre:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  jenkins
1 upgraded, 0 newly installed, 0 to remove and 300 not upgraded.
Need to get 93.7 MB of archives.
After this operation, 920 kB disk space will be freed.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.375.3 [93.7 MB]
Fetched 93.7 MB in 15s (6,177 kB/s)
(Reading database ... 306739 files and directories currently installed.)
Preparing to unpack .../jenkins_2.375.3_all.deb ...
Unpacking jenkins (2.375.3) over (2.375.2) ...
Setting up jenkins (2.375.3) ...
```

```
(base) computer@computer-ThinkCentre:~$ sudo apt update
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:6 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:7 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Get:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [102 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [267 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:13 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [7,996 B]
Get:14 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [12.5 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [41.6 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [15.2 kB]
Hit:17 https://ppa.launchpadcontent.net/swi-prolog/stable/ubuntu jammy InRelease
Fetched 782 kB in 2s (473 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
300 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

3. To check if Jenkins is installed and running, run the following command : **sudo systemctl status jenkins**
A bright green entry labeled active (running) should appear in the output, indicating that the service is running.
4. Exit the status screen by pressing **Ctrl+Z**

```
(base) computer@computer-ThinkCentre:~$ 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
```


```
Synchronizing state of jenkins.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable jenkins
```


Step 3 : Modify Firewall to Allow Jenkins

Allow Jenkins to communicate by setting up the default UFW firewall.
1. Open port 8080 by running the following commands : **sudo ufw allow 8080**
sudo ufw status

```
(base) computer@computer-ThinkCentre:~$ jenkins --version
2.375.3
(base) computer@computer-ThinkCentre:~$ 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) computer@computer-ThinkCentre:~$ sudo systemctl start jenkins
(base) computer@computer-ThinkCentre:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-02-22 09:50:23 IST; 24min ago
   Main PID: 6071 (java)
      Tasks: 52 (limit: 9050)
     Memory: 1.2G
        CPU: 1min 12.550s
     CGroup: /system.slice/jenkins.service
             └─6071 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Feb 22 09:50:08 computer-ThinkCentre jenkins[6071]: cf3dd4aee0594109ab0f67608c281299
Feb 22 09:50:08 computer-ThinkCentre jenkins[6071]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Feb 22 09:50:08 computer-ThinkCentre jenkins[6071]: ************************************************************
Feb 22 09:50:08 computer-ThinkCentre jenkins[6071]: ************************************************************
Feb 22 09:50:08 computer-ThinkCentre jenkins[6071]: ************************************************************
Feb 22 09:50:23 computer-ThinkCentre jenkins[6071]: 2023-02-22 04:20:23.765+0000 [id=46]       INFO       jenkins.InitReactorRunner$1#onAttained: Completed initialization
Feb 22 09:50:23 computer-ThinkCentre jenkins[6071]: 2023-02-22 04:20:23.776+0000 [id=24]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Feb 22 09:50:23 computer-ThinkCentre systemd[1]: Started Jenkins Continuous Integration Server.
Feb 22 09:50:24 computer-ThinkCentre jenkins[6071]: 2023-02-22 04:20:24.951+0000 [id=68]       INFO       h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.M>
Feb 22 09:50:24 computer-ThinkCentre jenkins[6071]: 2023-02-22 04:20:24.951+0000 [id=68]       INFO       hudson.util.Retrier#start: Performed the action check updates server successfully at the atter>
lines 1-20/20 (END)
[1]+  Stopped                 sudo systemctl status jenkins
(base) computer@computer-ThinkCentre:~$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
(base) computer@computer-ThinkCentre:~$ sudo ufw status
Status: inactive
```

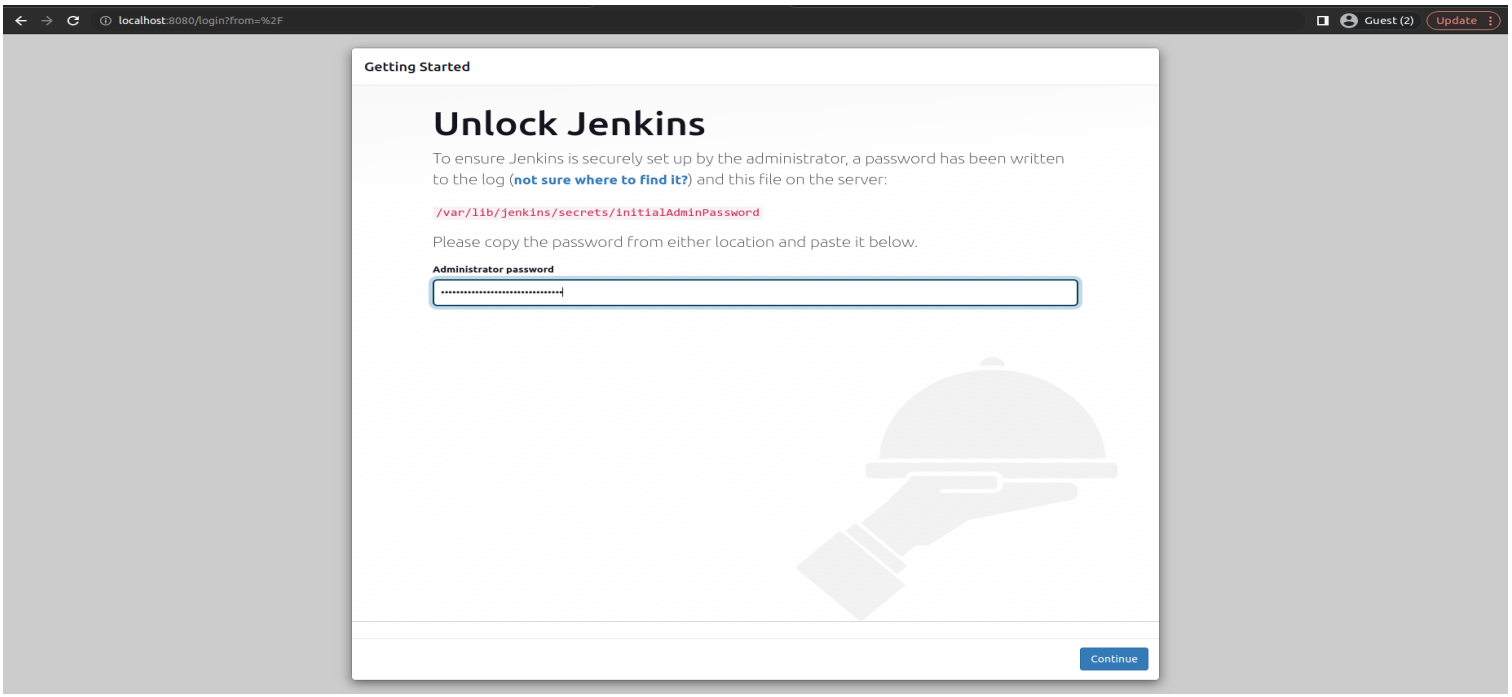
If you're using a different firewall application, follow its specific instructions to allow traffic on port 8080.
2. If you haven't configured the UFW firewall yet, it displays as inactive. Enable UFW by running:
sudo ufw enable

```
(base) computer@computer-ThinkCentre:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

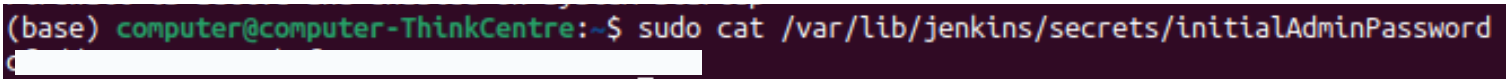
Step 4 : Set up Jenkins

Follow the steps below to set up Jenkins and start using it:

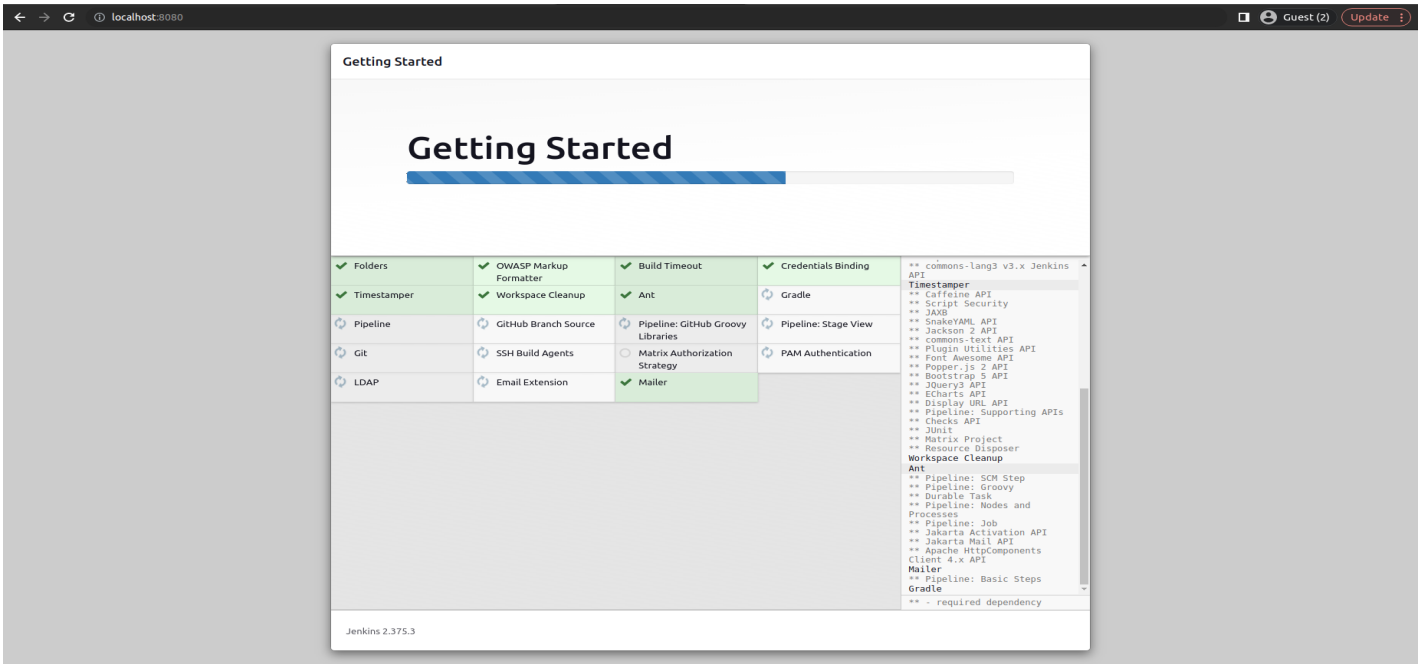
1. Open a web browser, and navigate to your server' IP address. Use the following syntax:
   http://ip_address_or_domain:8080
   Use the actual IP address or domain name for the server you're using Jenkins on. For example, if you're running Jenkins locally, use localhost (127.0.0.1) : **http://localhost:8080**
   A page opens prompting you to Unlock Jenkins. Obtain the required administrator password in the next step.



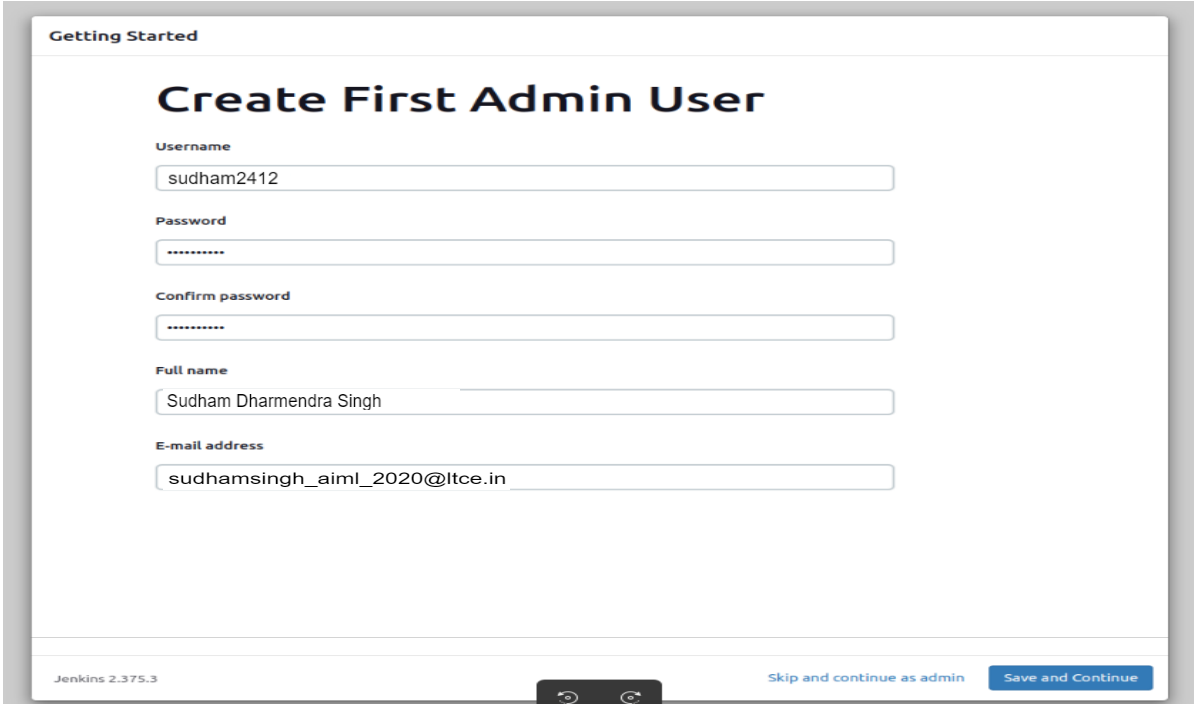
2. Obtain the default Jenkins unlock password by opening the terminal and running the following command :
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword



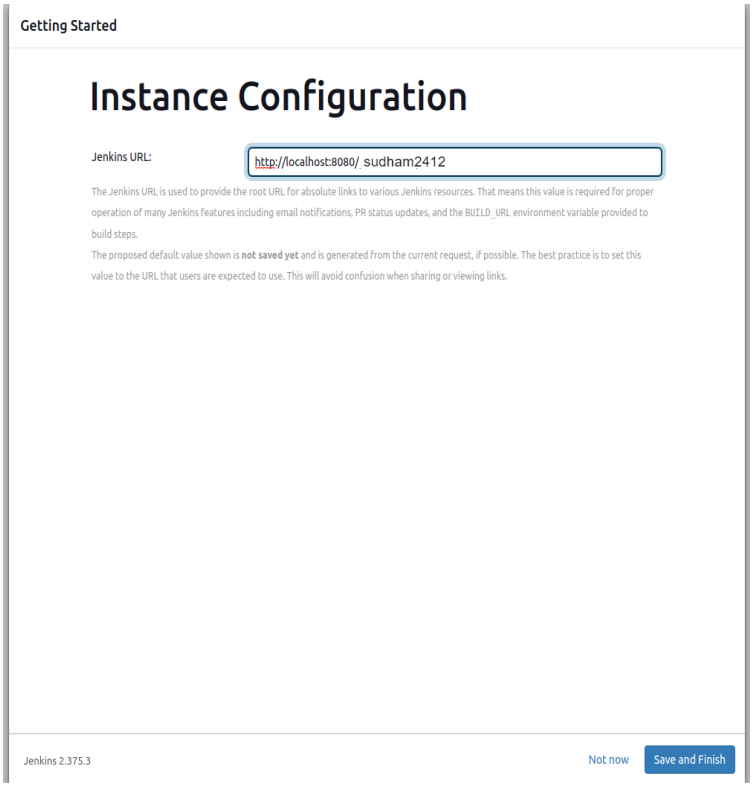
3. The system returns an alphanumeric code. Enter that code in the Administrator password field and click Continue.

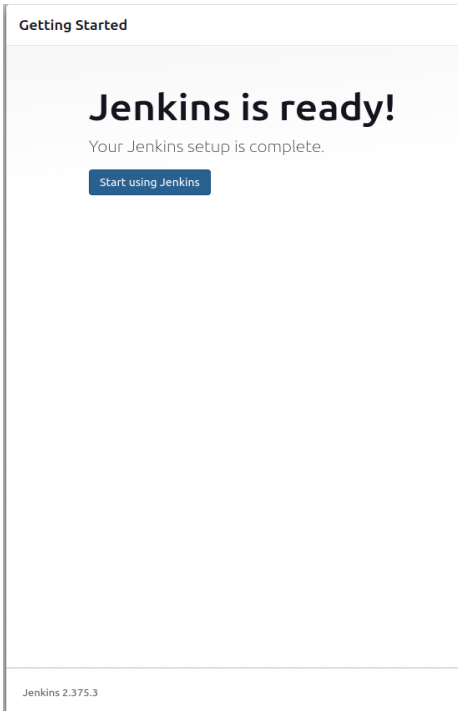


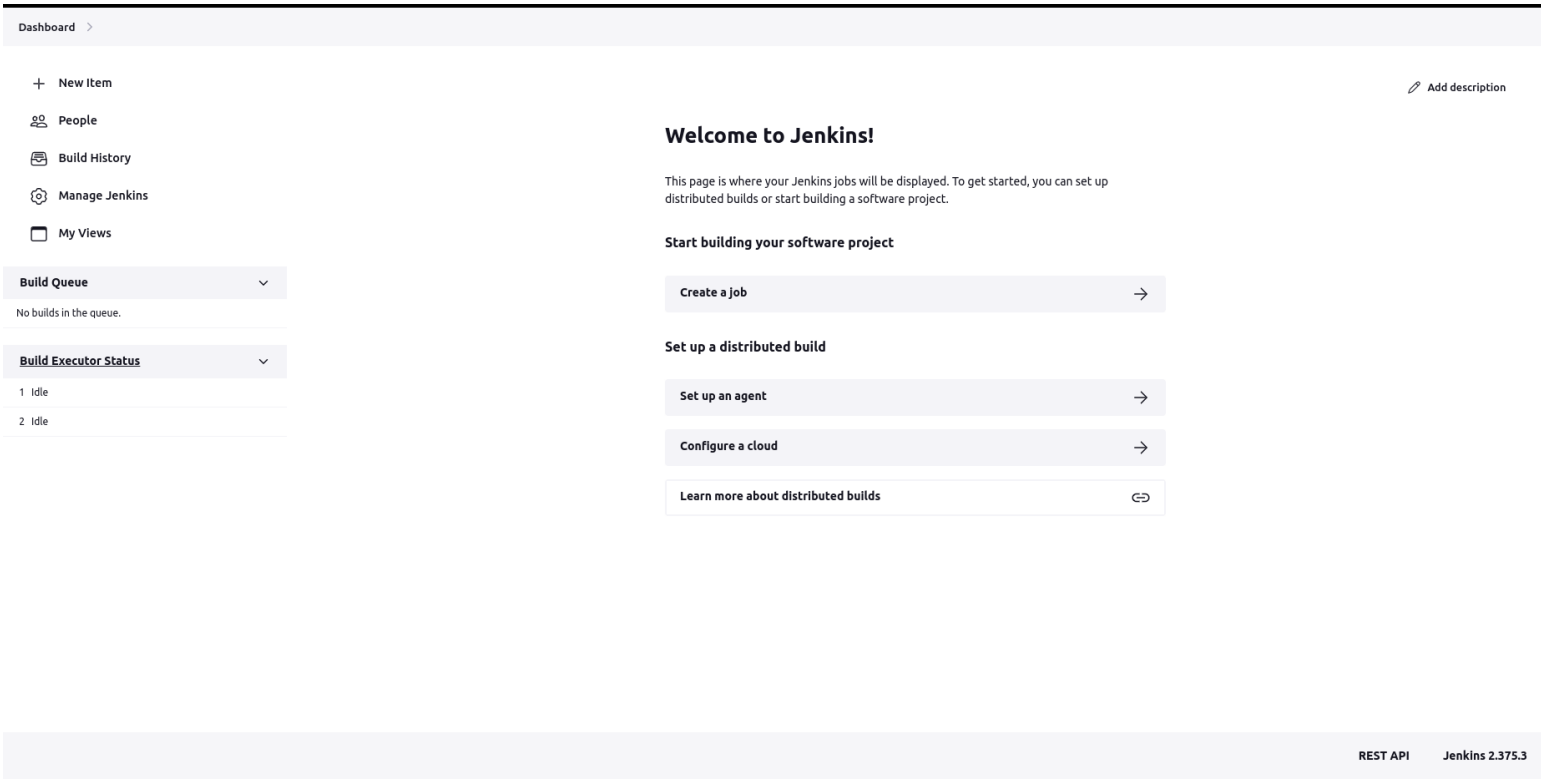
4. The setup prompts to either Install suggested plugins or Select plugins to install. It's fine to simply install the suggested plugins. You can always install more plugins later. The system continues the initial Jenkins setup.
5. The next step is the Create First Admin User. Enter the credentials you want to use for your Jenkins administrator, then click Save and Continue.



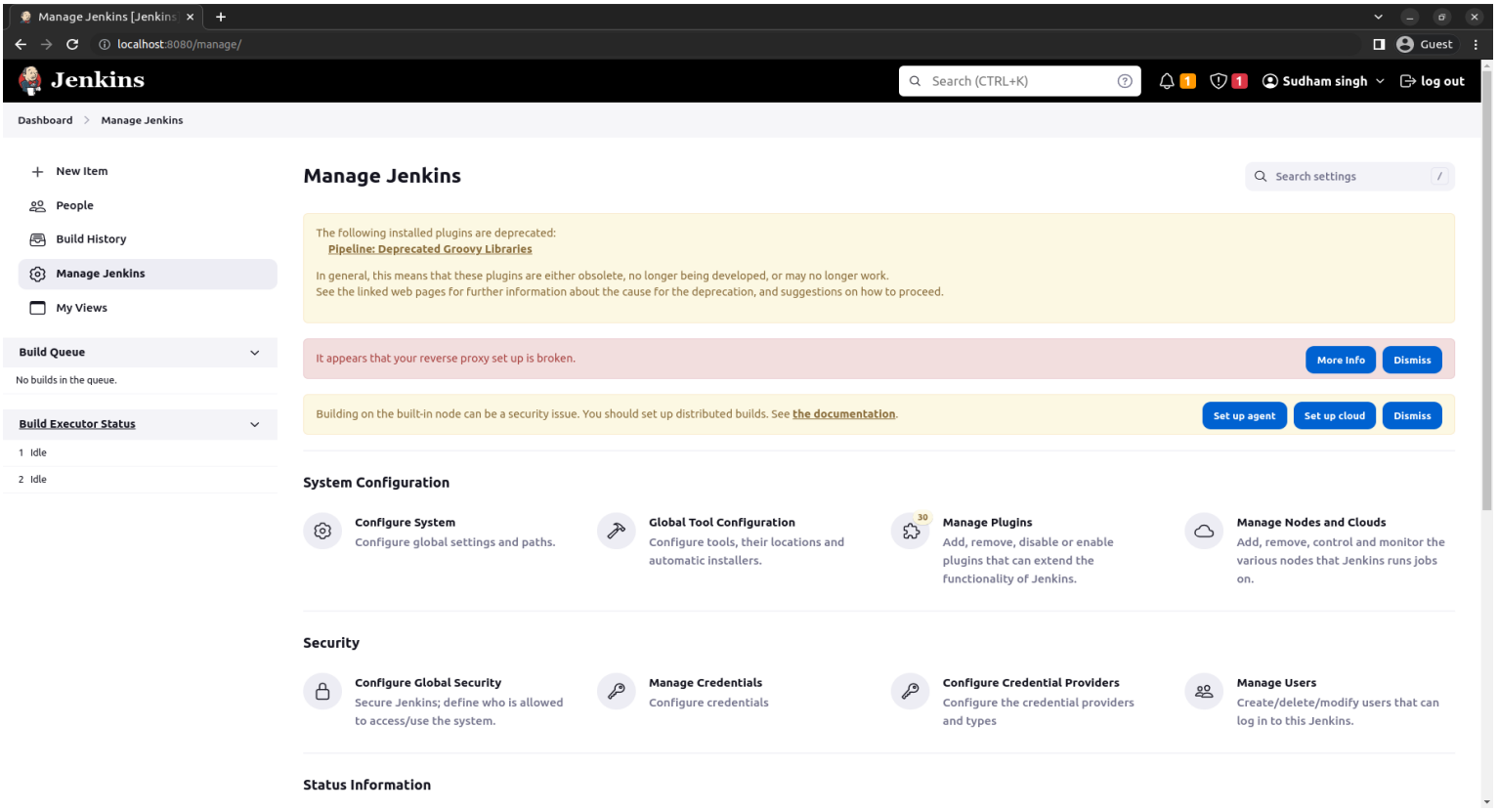
6. After this, you should set up the Instance Configuration. This is the preferred network address for this Jenkins installation. Confirm the address you want to use for your server. This is most likely the same address you used to get to this configuration page. Once you specify the Jenkins URL, click Save and Finish.
7. You should see a page that says Jenkins is ready! Click Start using Jenkins to open the Jenkins dashboard.

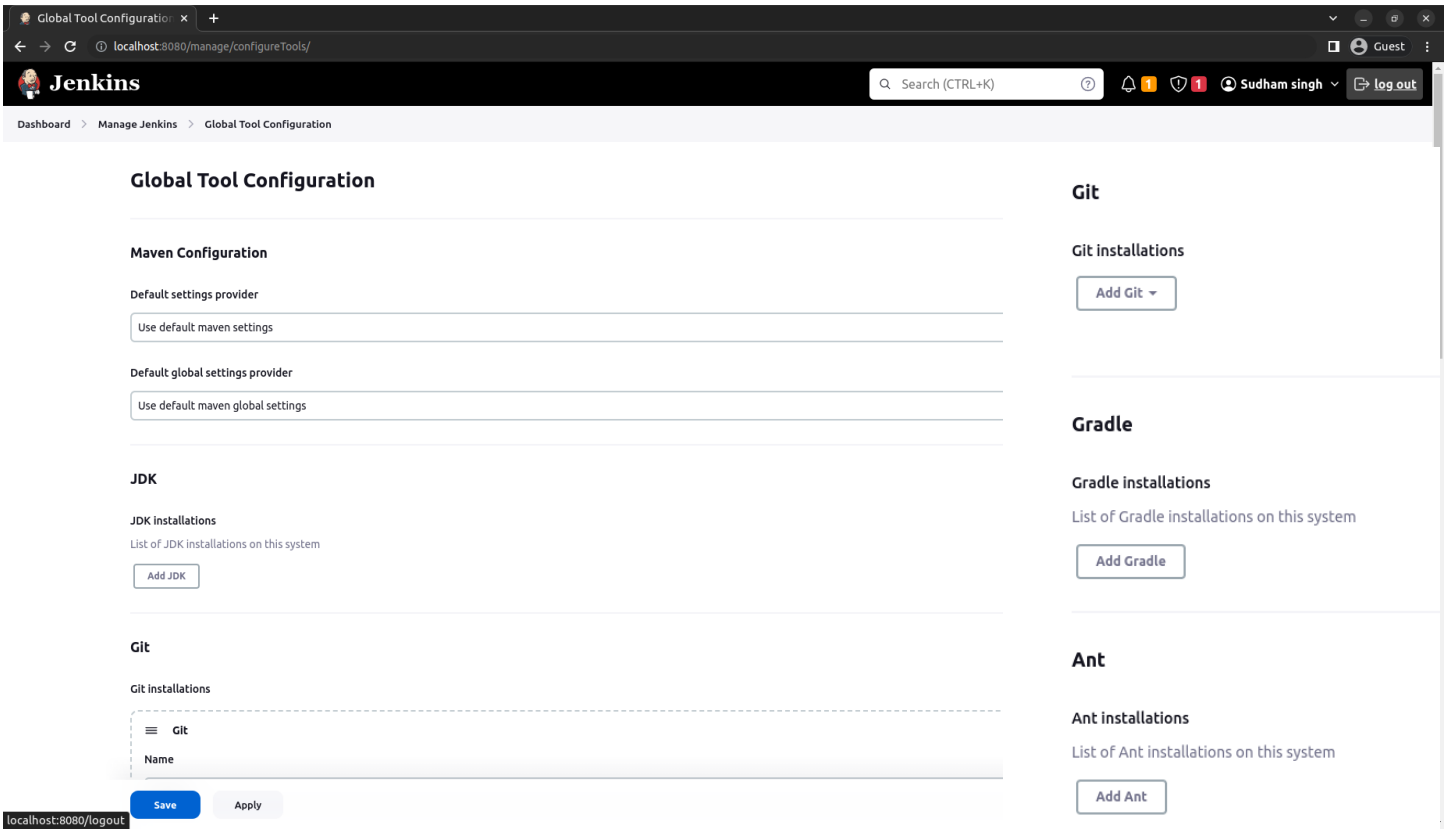


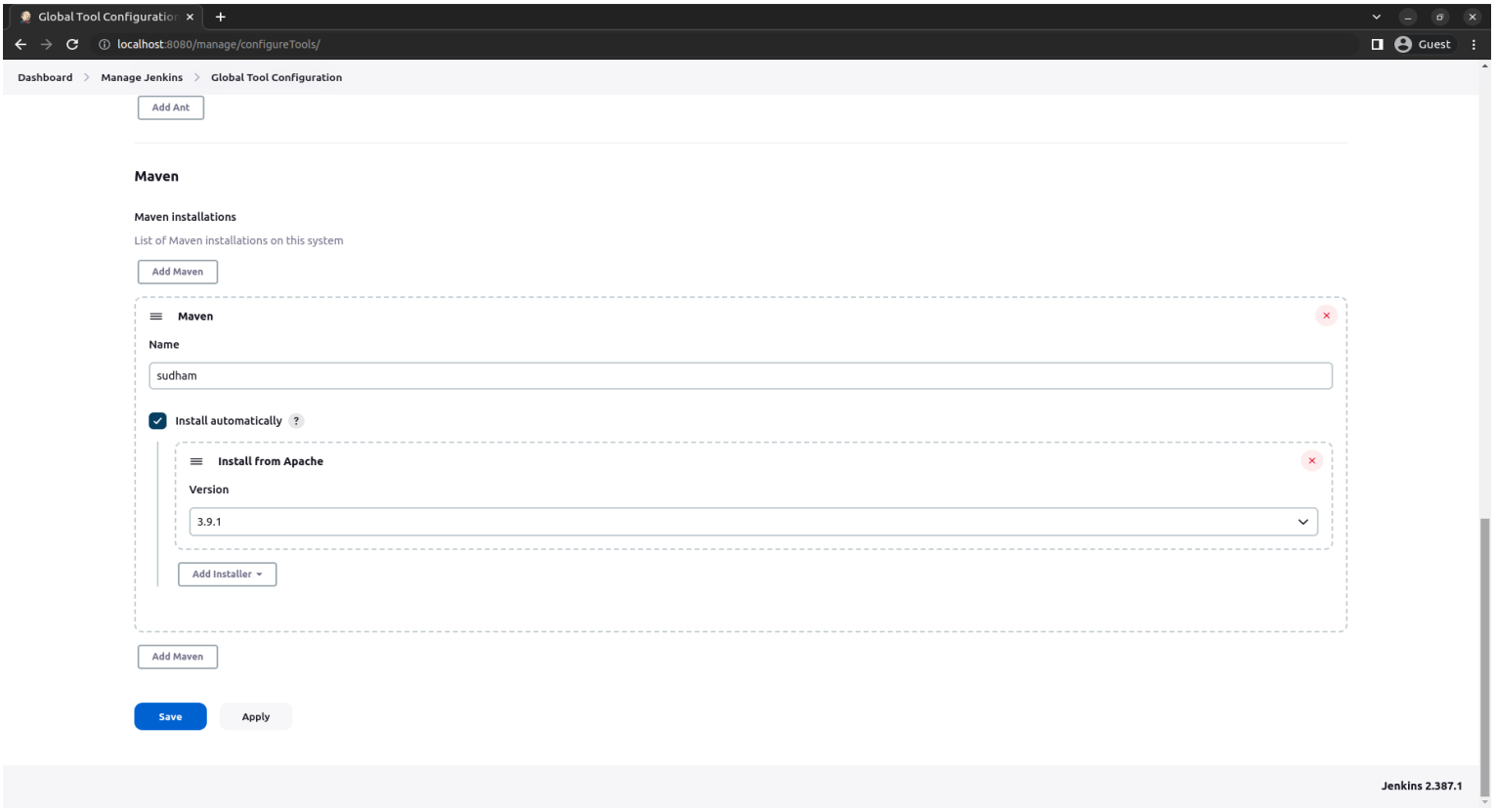




Configuring Jenkins with maven




Global Tool Configuration
localhost:8080/manage/configureTools/
Guest
Jenkins
Search (CTRL+K)
Sudham singh
log out
Dashboard
Manage Jenkins
Global Tool Configuration
Global Tool Configuration
Maven Configuration
Default settings provider
Use default maven settings
Default global settings provider
Use default maven global settings
JDK
JDK installations
List of JDK installations on this system
Add JDK
Git
Git installations
Git
Name
Save
Apply
Git
Git installations
Add Git
Gradle
Gradle installations
List of Gradle installations on this system
Add Gradle
Ant
Ant installations
List of Ant installations on this system
Add Ant
localhost:8080/logout


Global Tool Configuration
localhost:8080/manage/configureTools/
Guest
Dashboard
Manage Jenkins
Global Tool Configuration
Add Ant
Maven
Maven installations
List of Maven installations on this system
Add Maven
Maven
Name
sudham
Install automatically
Install from Apache
Version
3.9.1
Add Installer
Add Maven
Save
Apply
Jenkins 2.387.1

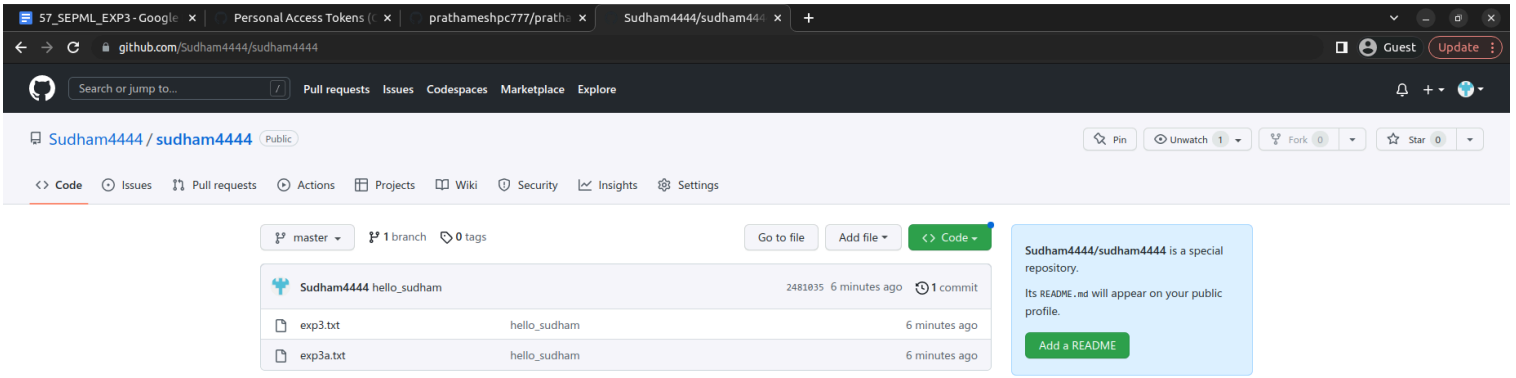
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

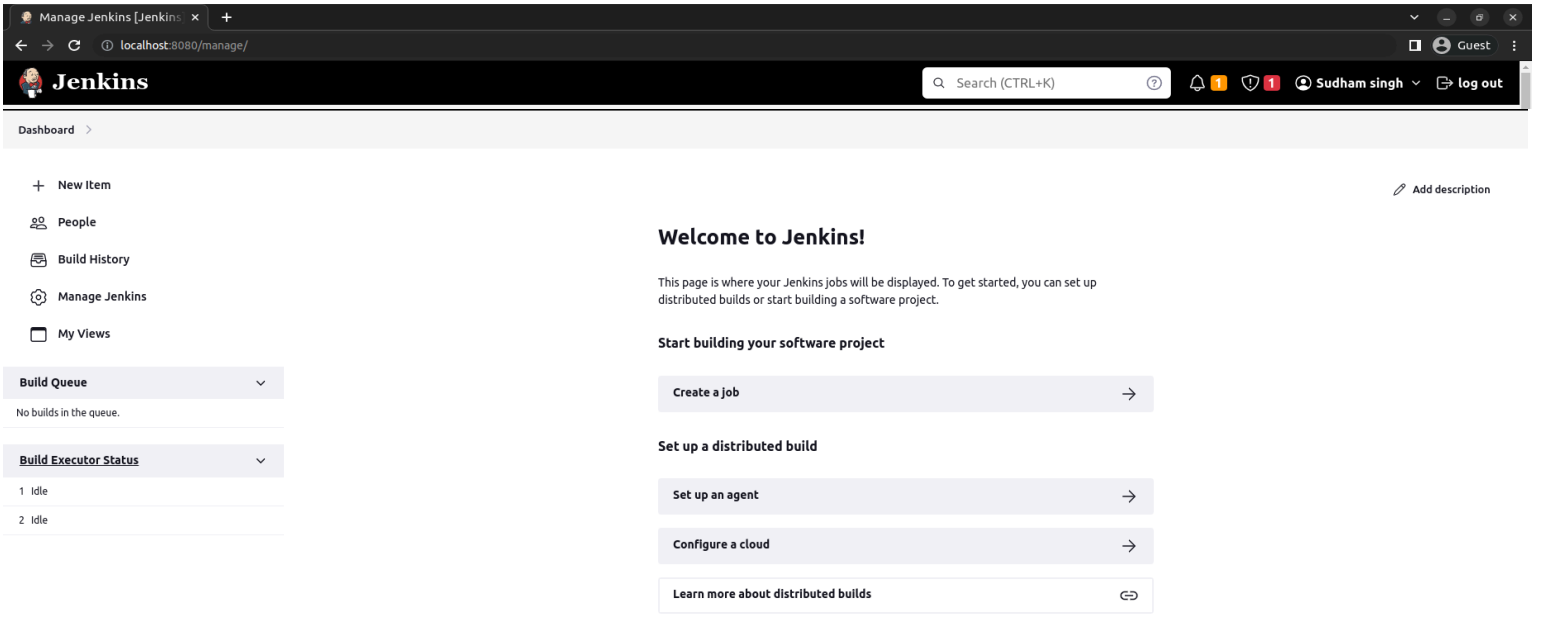
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	
DOP	
DOS	

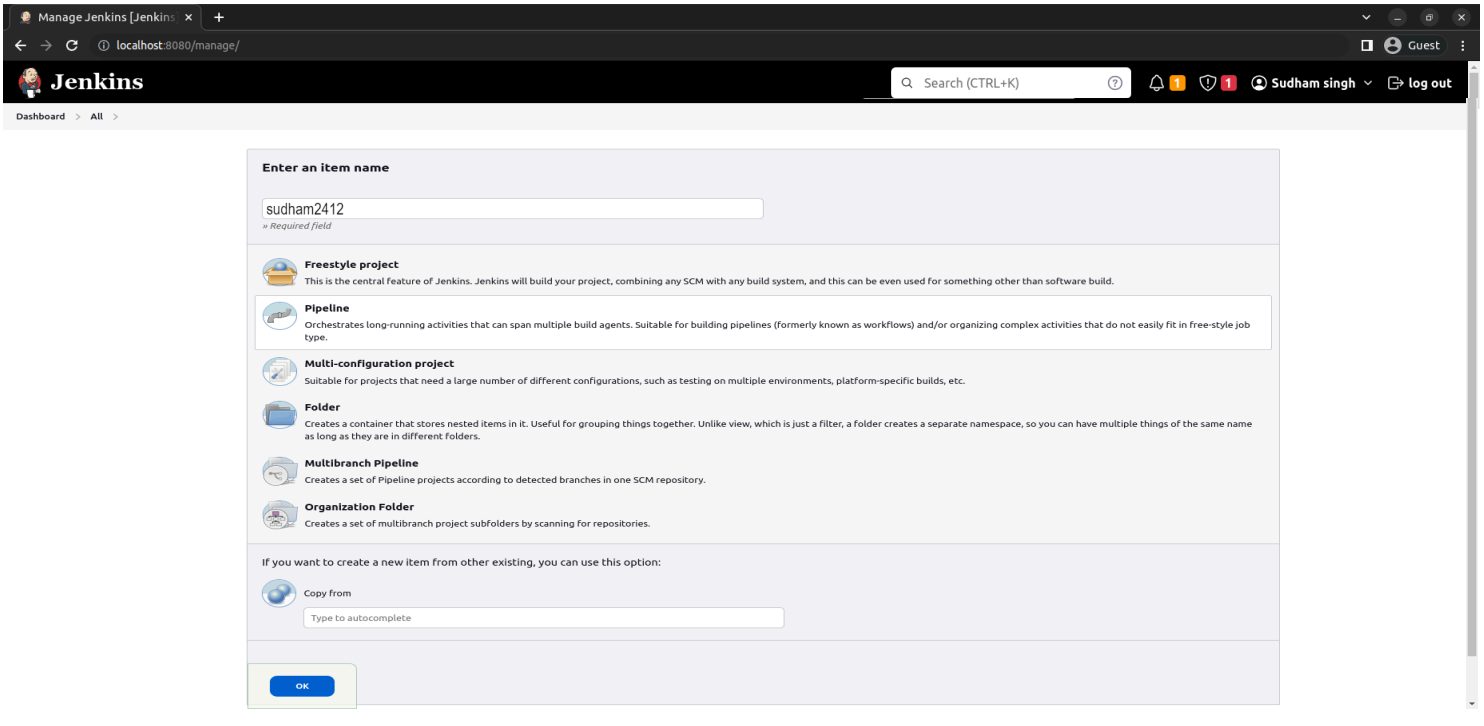
Building pipeline of jobs and creating pipelining script in jenkins

Step 1 - Log in to your Github account and Jenkins account

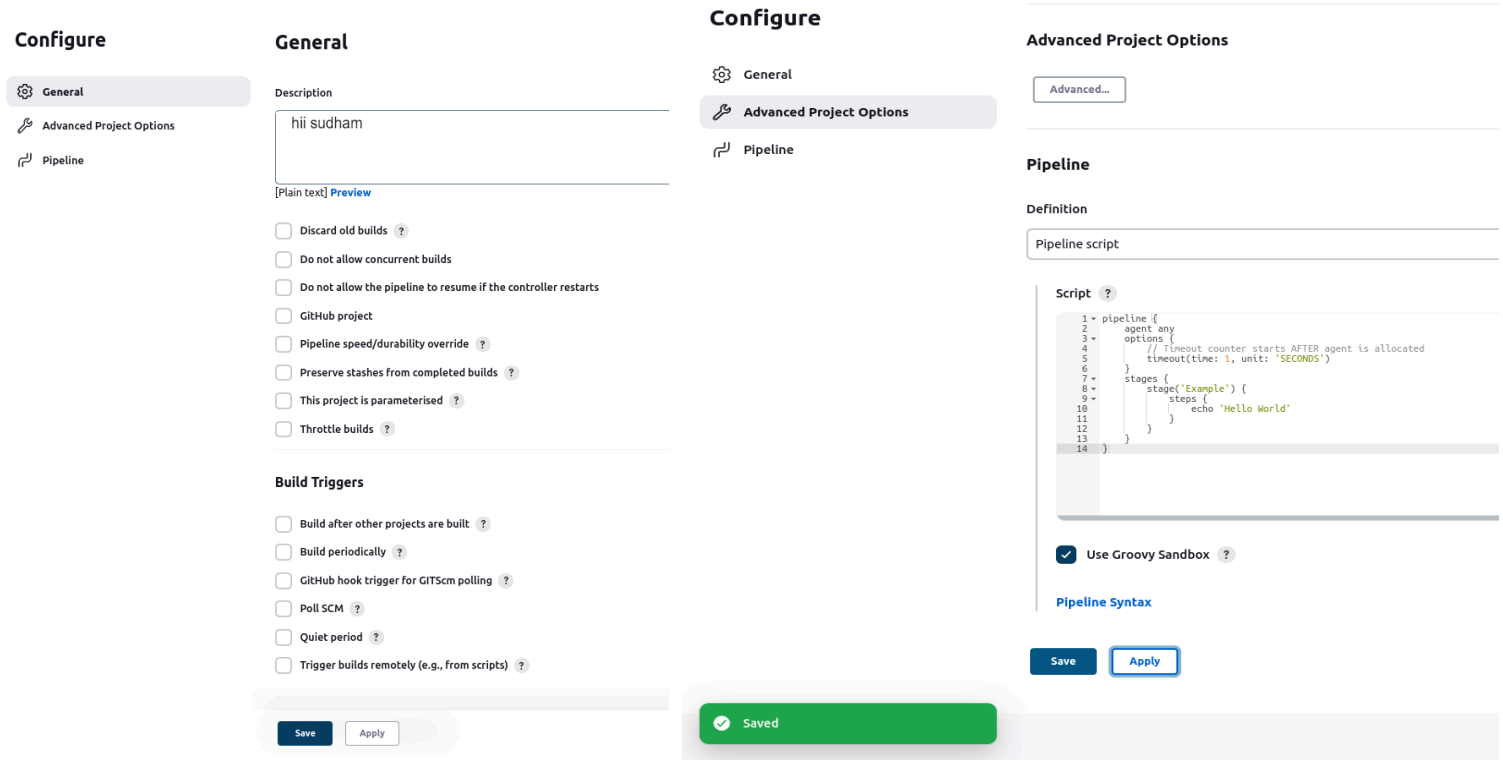




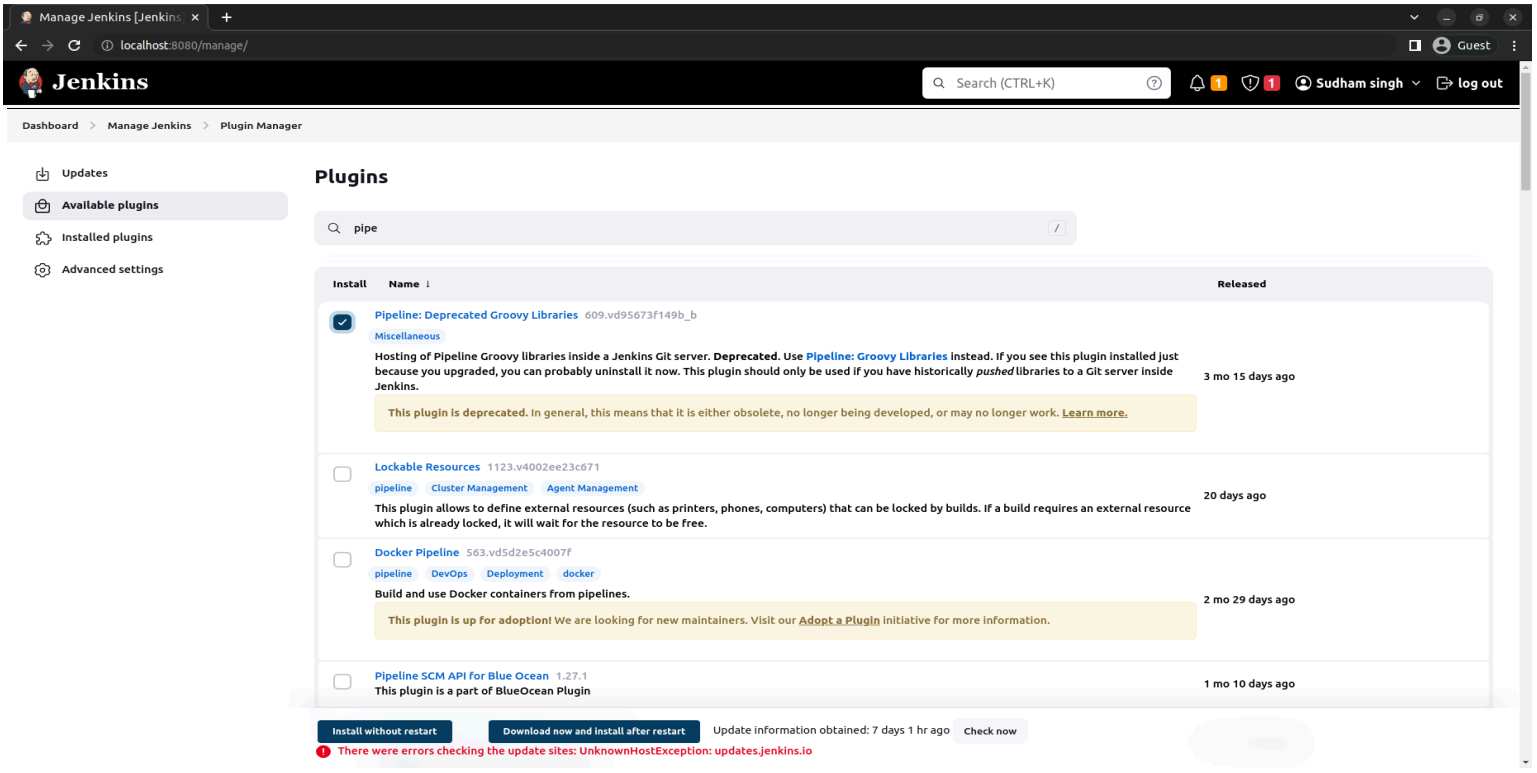
Step 2 - Click on **New item** and enter **item name** you want, select **pipeline** and then click on **OK**



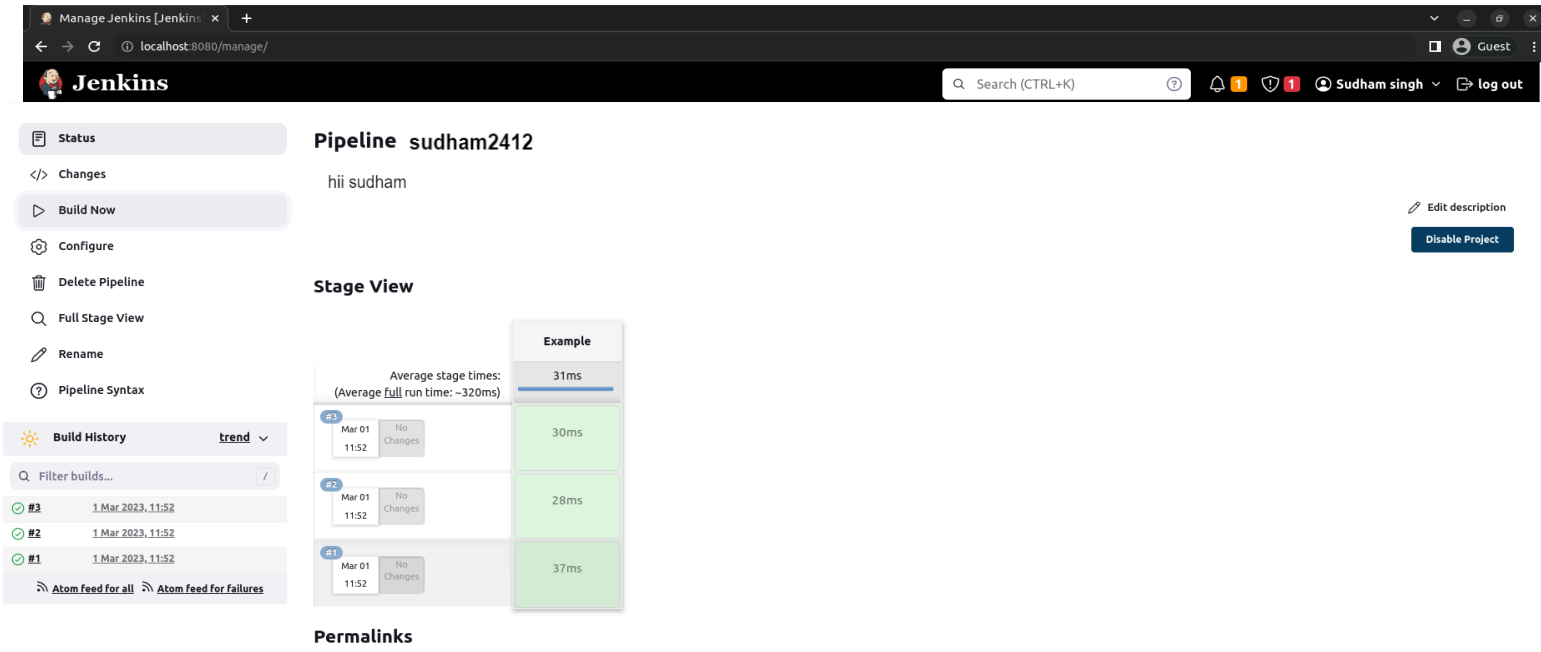
Step 3 - Configure your general, add Description then add **Pipeline script** and **Apply** and **Save (CREATING SCRIPT)**



Step 4 - Now go to dashboard and click on **manage jenkins**, then click on **manage plugins**, check available plugins

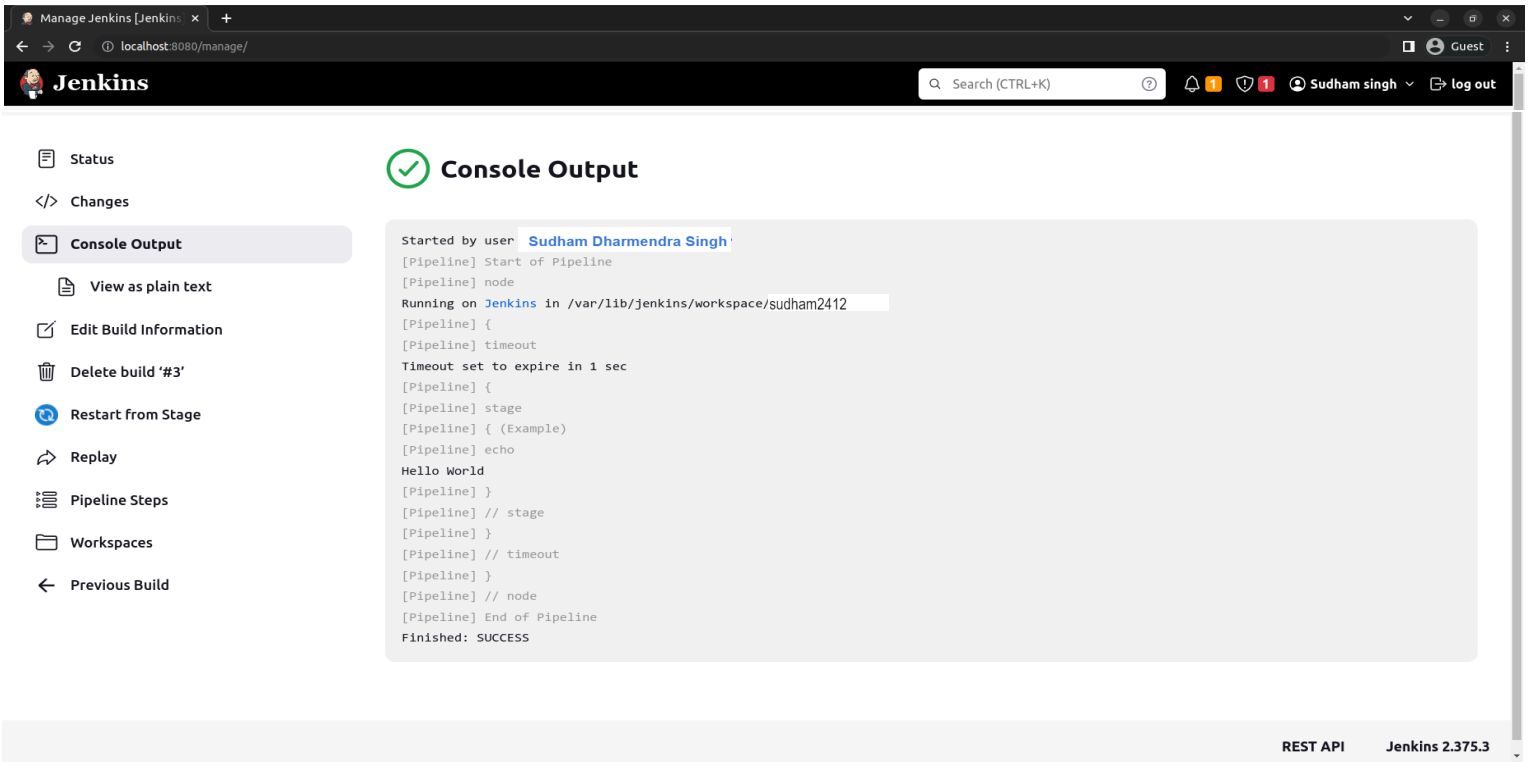


Step 5 - Select which pipeline you want to and then click on **Install without restart (BUILDING PIPELINING)**



You can check your status as well

Step 6 - After successfully **build** and **creating pipeline script,** the following **console output** you can see/check



Installing tomcat server

Step 1 - Update system repositories

Press “CTRL+ALT+T” to open the terminal of your Ubuntu 22.04 and run the below given command to update system repositories: **$ sudo apt update**

Step 2- Check java installed or not : **$ java –version**

Step 3 - Check the availability of Apache Tomcat package

After fulfilling the requirements, check the availability of the Apache Tomcat package in the repository:

$ sudo apt-cache search tomcat

The given output signifies that the “tomcat9” package for download:

Step 4 - Install Apache Tomcat Server on Ubuntu 22.04

After finding the required Apache Tomcat package, we will install it on Ubuntu 22.04 with the help of the below-given command: **$ sudo apt install tomcat9 tomcat9-admin**

```
computer@computer-ThinkCentre: ~
computer@computer-ThinkCentre:~$ sudo apt update
[sudo] password for computer:
Ign:1 https://pkg.jenkins.io/debian binary/ InRelease
Get:2 https://pkg.jenkins.io/debian binary/ Release [2,044 B]
Get:3 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Ign:3 https://pkg.jenkins.io/debian binary/ Release.gpg
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Hit:8 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Get:9 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:10 https://ppa.launchpadcontent.net/swi-prolog/stable/ubuntu jammy InRelease
Reading package lists... Done
W: GPG error: https://pkg.jenkins.io/debian binary/ Release: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY FCEF32E745F2C3D5
E: The repository 'https://pkg.jenkins.io/debian binary/ Release' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
computer@computer-ThinkCentre:~$ java --version
openjdk 11.0.18 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
computer@computer-ThinkCentre:~$ sudo apt-cache search tomcat
centreon-plugins - Collection of Nagios plugins to monitor OS, services and network devices
libapache-mod-jk-doc - Documentation of libapache2-mod-jk package
libapache2-mod-jk - Apache 2 connector for the Tomcat Java servlet engine
libjnlp-servlet-java - simple and convenient packaging format for JNLP applications
liblogback-java - flexible logging library for Java
liblogback-java-doc - flexible logging library for Java - documentation
libnetty-tcnative-java - Tomcat native fork for Netty
libnetty-tcnative-jni - Tomcat native fork for Netty (JNI library)
libspring-instrument-java - modular Java/J2EE application framework - Instrumentation
libtcnative-1 - Tomcat native library using the Apache Portable Runtime
libtomcat9-embed-java - Apache Tomcat 9 - Servlet and JSP engine -- embed libraries
libtomcat9-java - Apache Tomcat 9 - Servlet and JSP engine -- core libraries
libtomcatjss-java - JSSE implementation using JSS for Tomcat
monitoring-plugins-contrib - Plugins for nagios compatible monitoring systems
python3-ajpy - Python module to craft AJP requests
resource-agents-extra - Cluster Resource Agents
tomcat-jakartaee-migration - Apache Tomcat migration tool for Jakarta EE
tomcat9 - Apache Tomcat 9 - Servlet and JSP engine
tomcat9-admin - Apache Tomcat 9 - Servlet and JSP engine -- admin web applications
tomcat9-common - Apache Tomcat 9 - Servlet and JSP engine -- common files
tomcat9-docs - Apache Tomcat 9 - Servlet and JSP engine -- documentation
tomcat9-examples - Apache Tomcat 9 - Servlet and JSP engine -- example web applications
tomcat9-user - Apache Tomcat 9 - Servlet and JSP engine -- tools to create user instances
yasat - simple stupid audit tool
computer@computer-ThinkCentre:~$ sudo apt install tomcat9 tomcat9-admin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapr1 libeclipse-jdt-core-java libtcnative-1 libtomcat9-java tomcat9-common
Suggested packages:
  tomcat9-docs tomcat9-examples tomcat9-user
The following NEW packages will be installed:
  libapr1 libeclipse-jdt-core-java libtcnative-1 libtomcat9-java tomcat9 tomcat9-admin tomcat9-common
0 upgraded, 7 newly installed, 0 to remove and 299 not upgraded.
Need to get 12.7 MB of archives.
After this operation, 16.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libeclipse-jdt-core-java all 3.27.0+eclipse4.21-1 [6,240 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libtomcat9-java all 9.0.58-1ubuntu0.1 [6,047 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9-common all 9.0.58-1ubuntu0.1 [60.9 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9 all 9.0.58-1ubuntu0.1 [37.0 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9-admin all 9.0.58-1ubuntu0.1 [68.8 kB]
```

Press “y” to permit the installation for a few minutes :

Step 5 - Check ports for Apache Tomcat Server

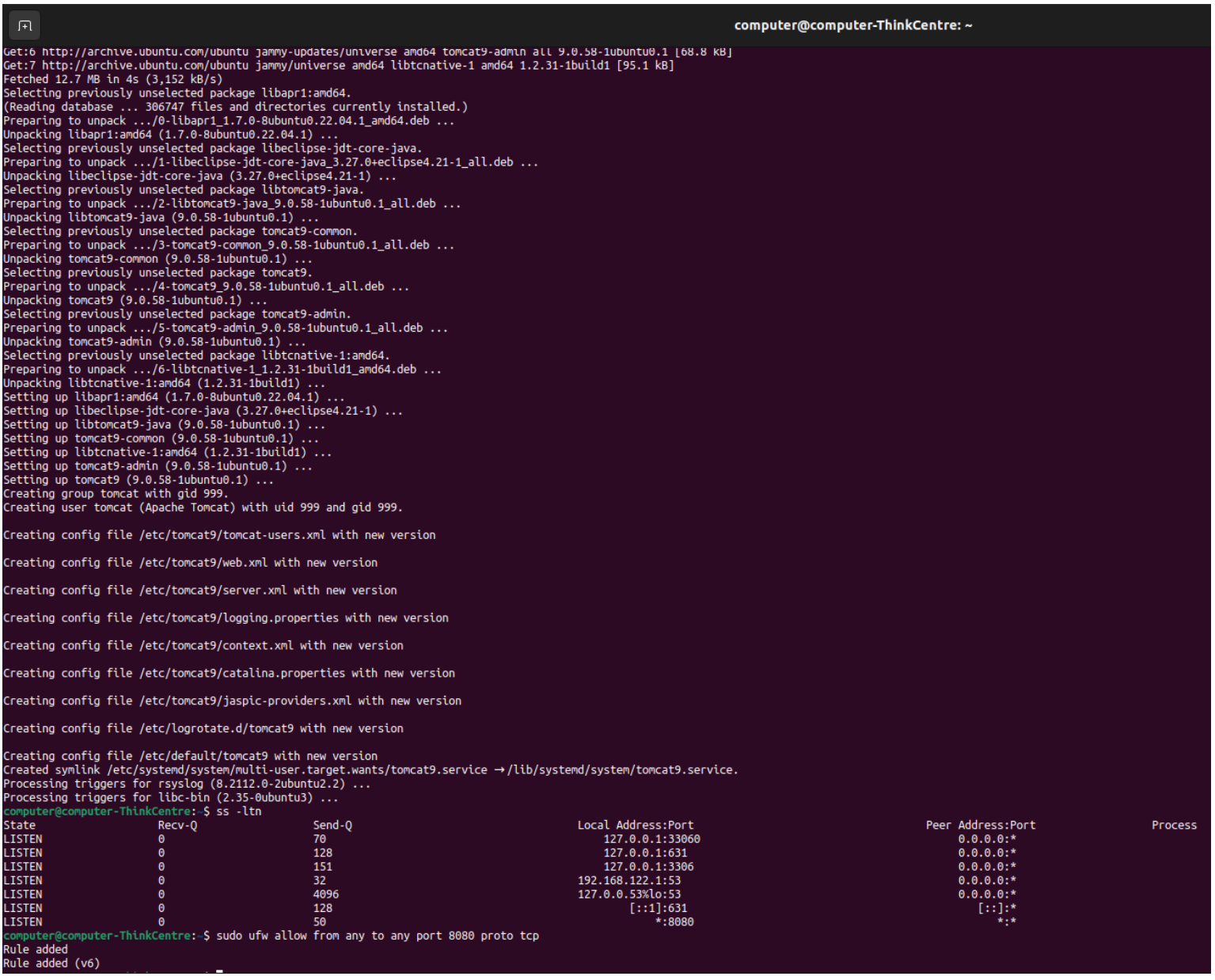
On Ubuntu 22.04, the Apache Tomcat Server automatically starts working after completing the installation. To validate this operation, you can utilize the “ss” command for displaying the network socket related information:

$ ss -ltn

The default port for the Apache Tomcat server is “8080” and it can be seen in the following output that port “8080” is listening for all incoming connections:

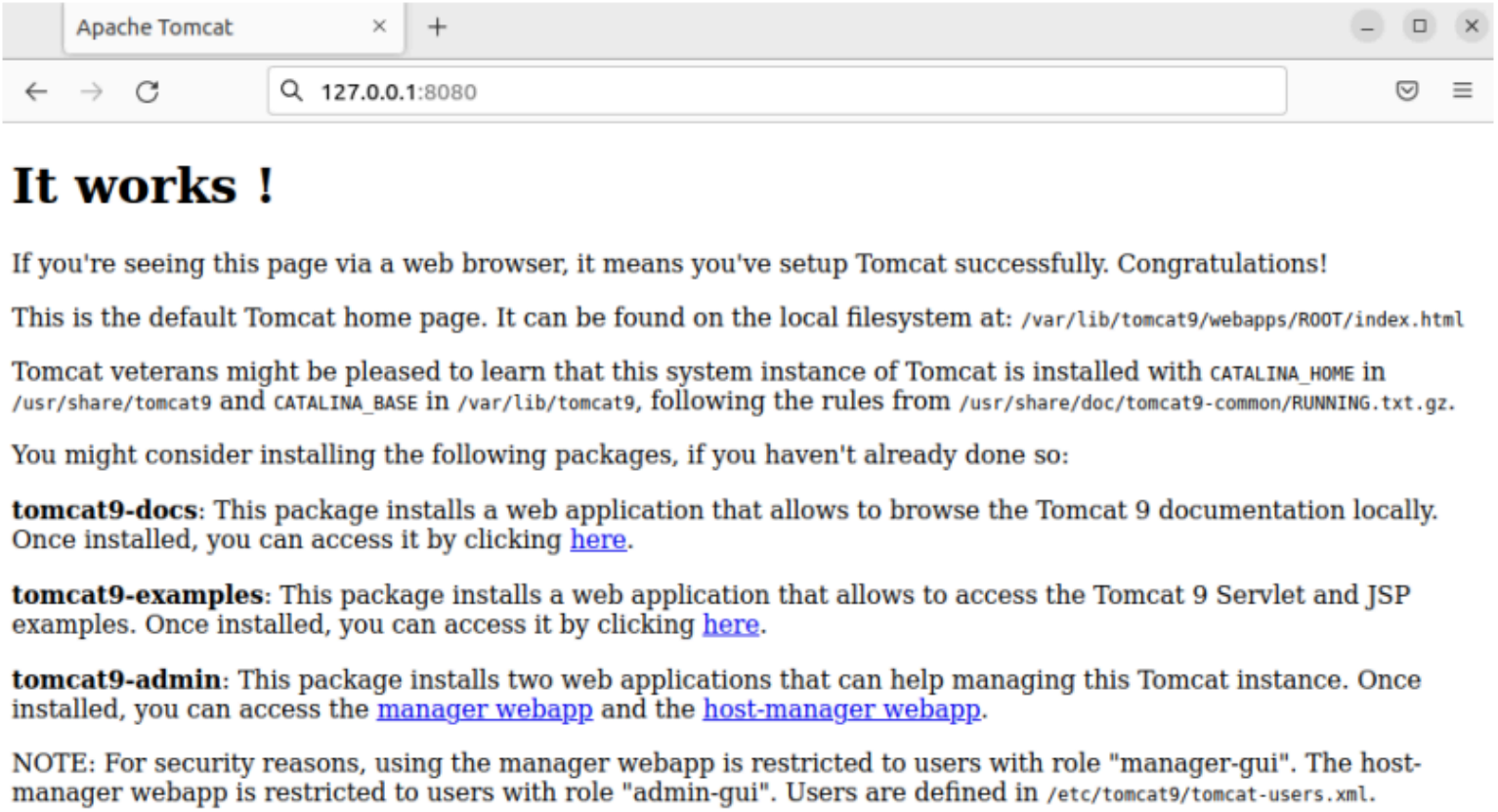
Step 6 - Open ports for Apache Tomcat Server

In case if the UFW firewall is activated on your system, then it may cause trouble while connecting external devices. So, to permit the incoming from any type of source to port “8080”, write out the following “ufw” command: **sudo ufw allow from any to any port 8080 proto tcp**



Step 7 - Test working of Apache Tomcat Server

If you have carefully followed all of the previous given, then at this point, the Apache Tomcat Server should be running on Ubuntu 22.04. To test its working specify your system loopback address with the number of the opened port for Apache Tomcat Server : **http://127.0.0.1:8080**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**06**
DOP	
DOS	

Experiment No. : 06

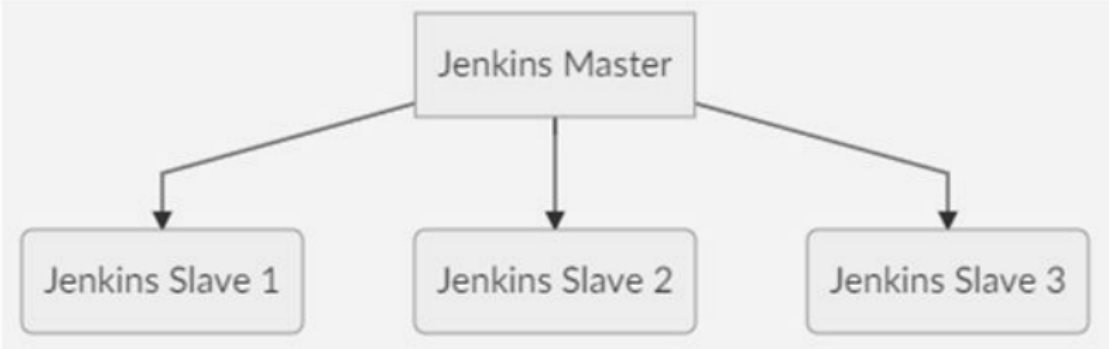
Aim : To understand Jenkins Master-Slave Architecture and scale your Jenkins Standalone implementation by implementing slave nodes.

Theory :

Jenkins Master and Slave Concept -

A Jenkins master comes with the basic installation of Jenkins, and in this configuration, the master handles all the tasks for your build system. If you are working on multiple projects, you may run multiple jobs on each project. Some projects need to run on some nodes, and in this process, we need to configure slaves. Jenkins slaves connect to the Jenkins master using the Java Network Launch Protocol.

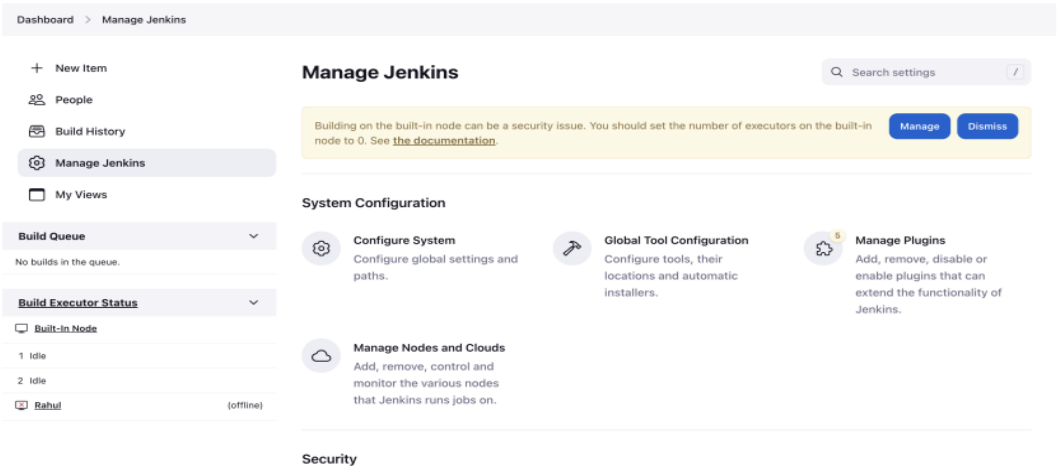
Jenkins Master and Slave Architecture



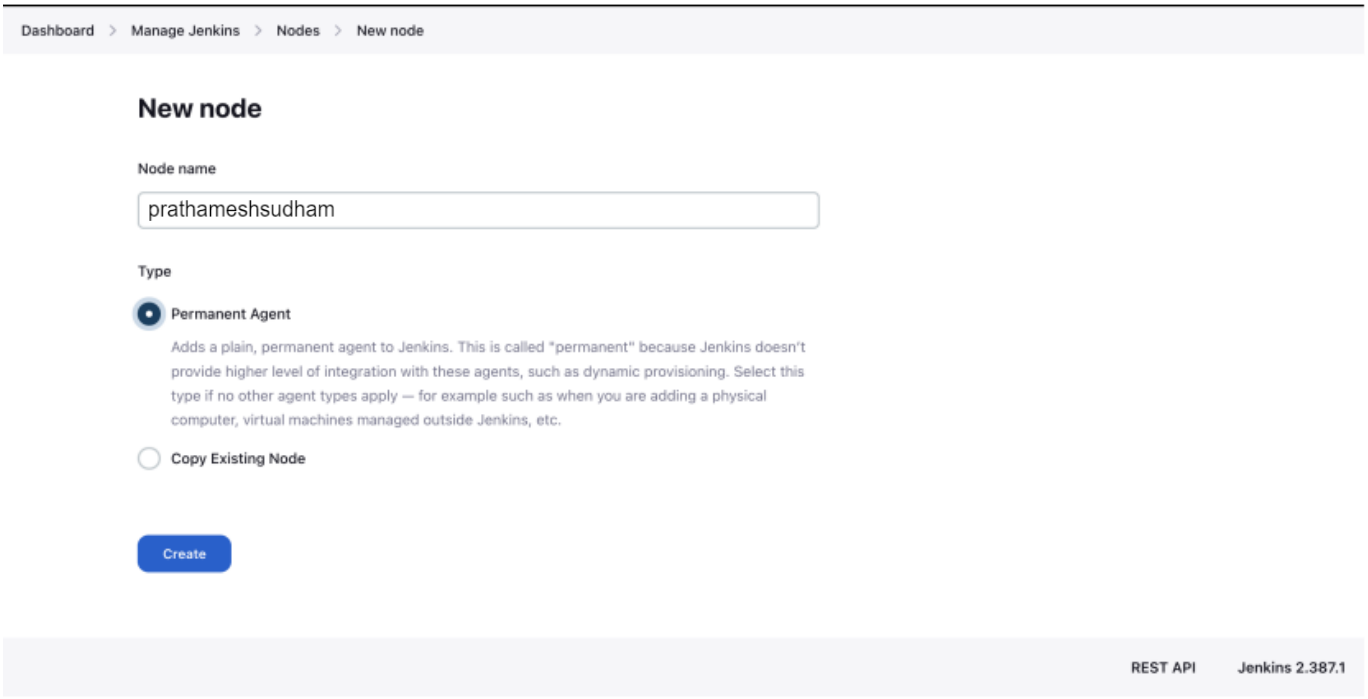
- The Jenkins master acts to schedule the jobs, assign slaves, and send builds to slaves to execute the jobs.
- It will also monitor the slave state (offline or online) and get back the build result responses from slaves and the display build results on the console output.
- The workload of building jobs is delegated to multiple slaves.

Steps to Configure Jenkins Master and Slave Nodes :

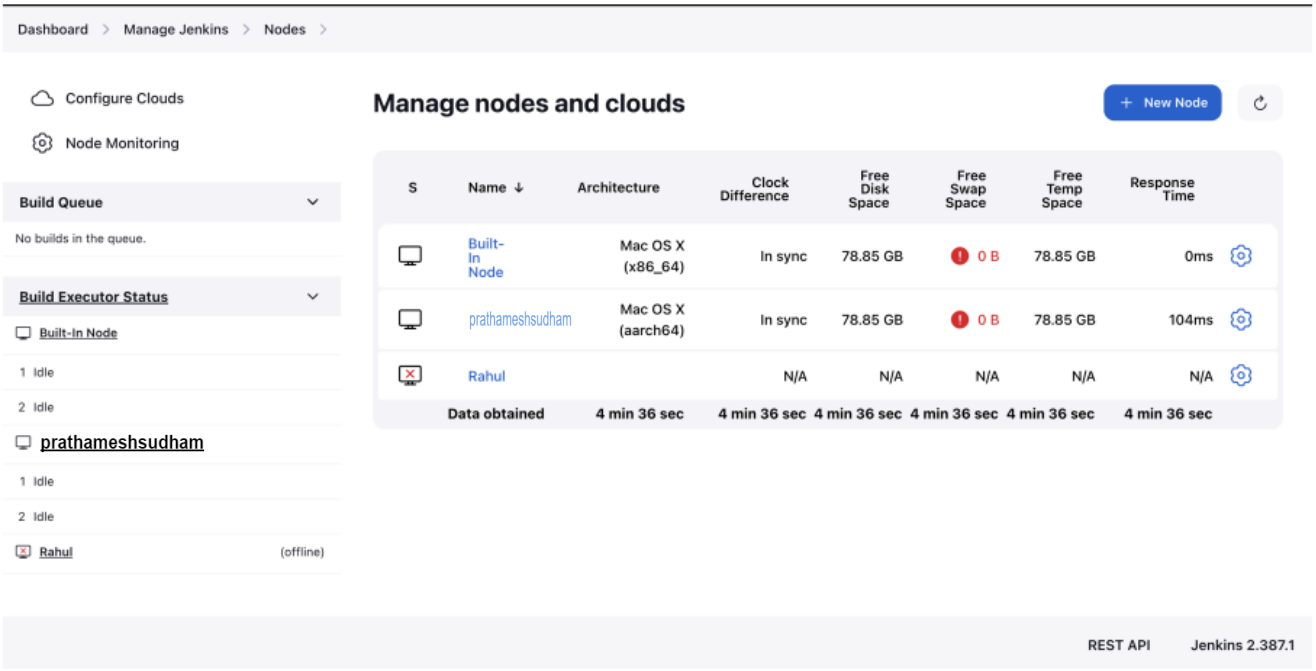
1. Click on Manage Jenkins in the left corner on the Jenkins dashboard.
2. Scroll down, Click on Manage Nodes and clouds



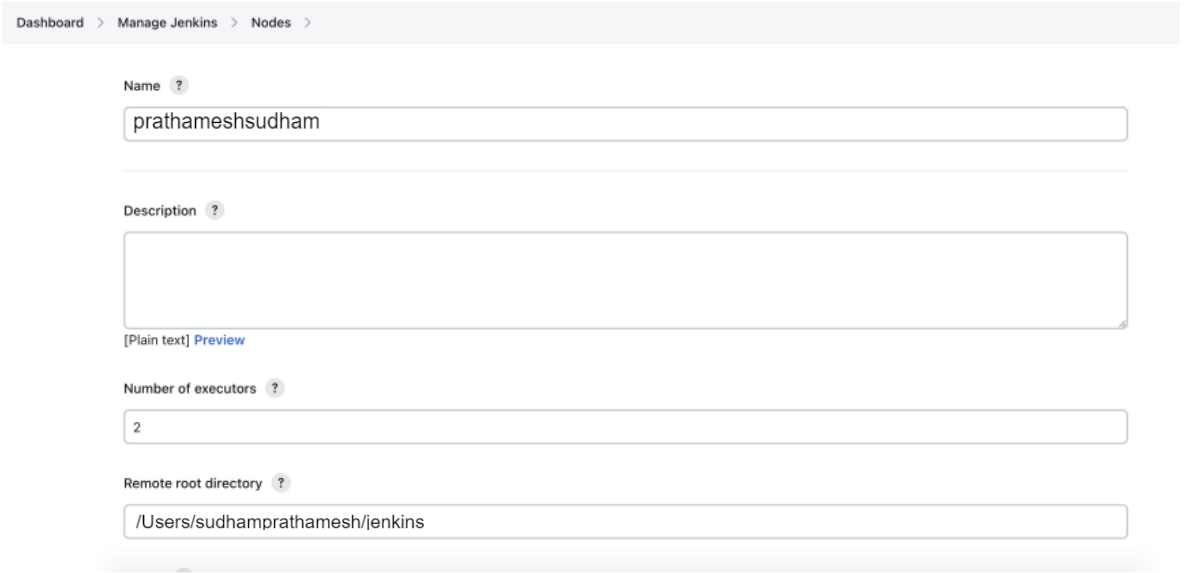
3. Select New Node and enter the name of the node in the Node Name field

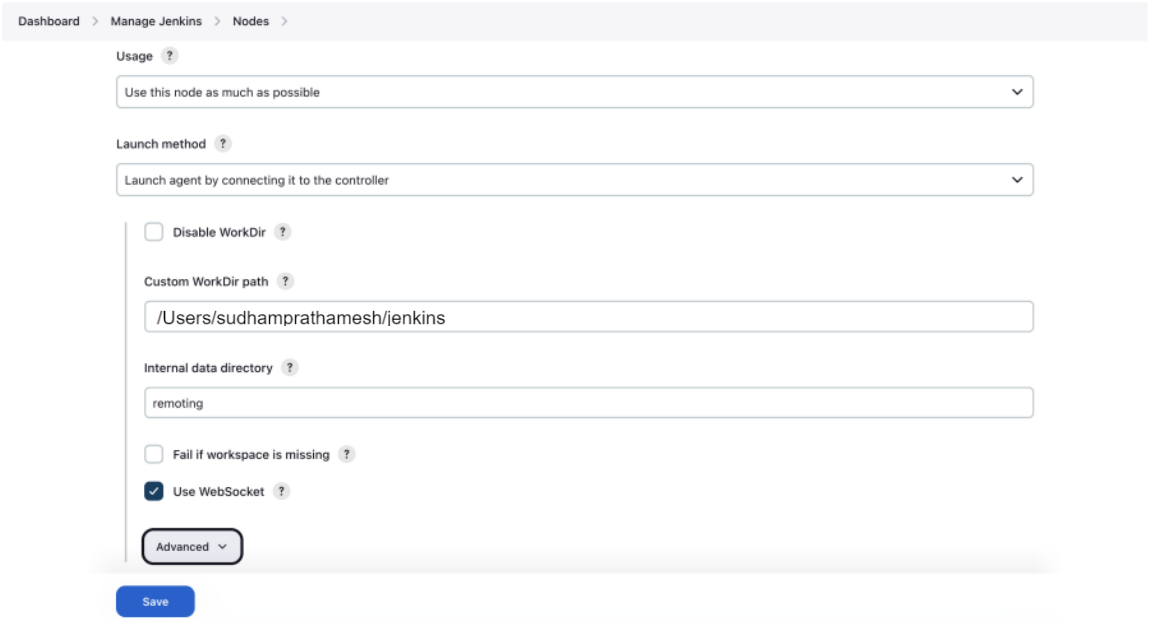


4. Select Permanent Agent and click the OK button. Initially, you will get only one option, “Permanent Agent.” Once you have one or more slaves you will get the “Copy Existing Node” option.

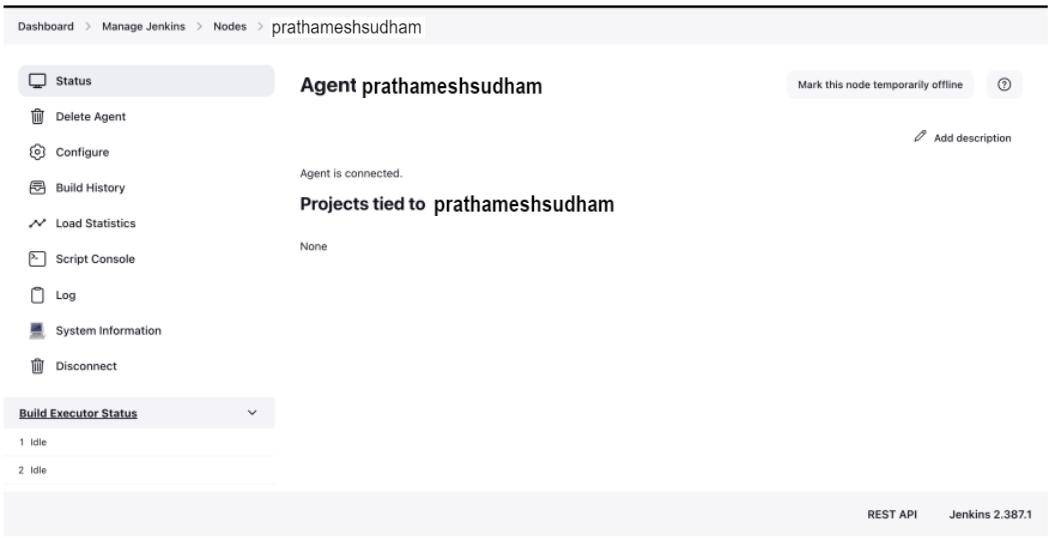


5. In the above screen shot, “prathameshsudham” named Slave is agent is created. Click on configure, Provide the details.

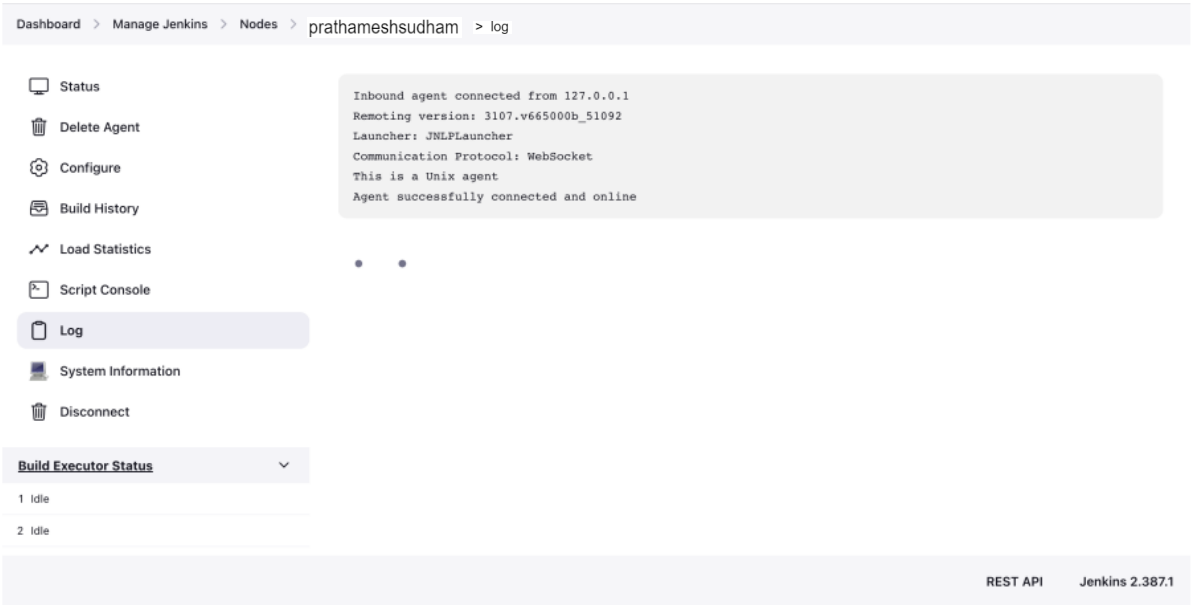




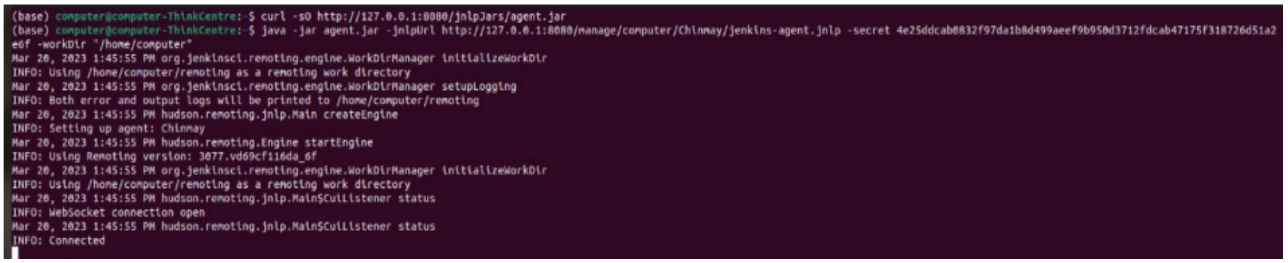
6. Now the agent is connected i.e. the project is tied.



7. Click on log for checking whether the agent is online or not.



8. The agent is connected online through terminal using some commands.



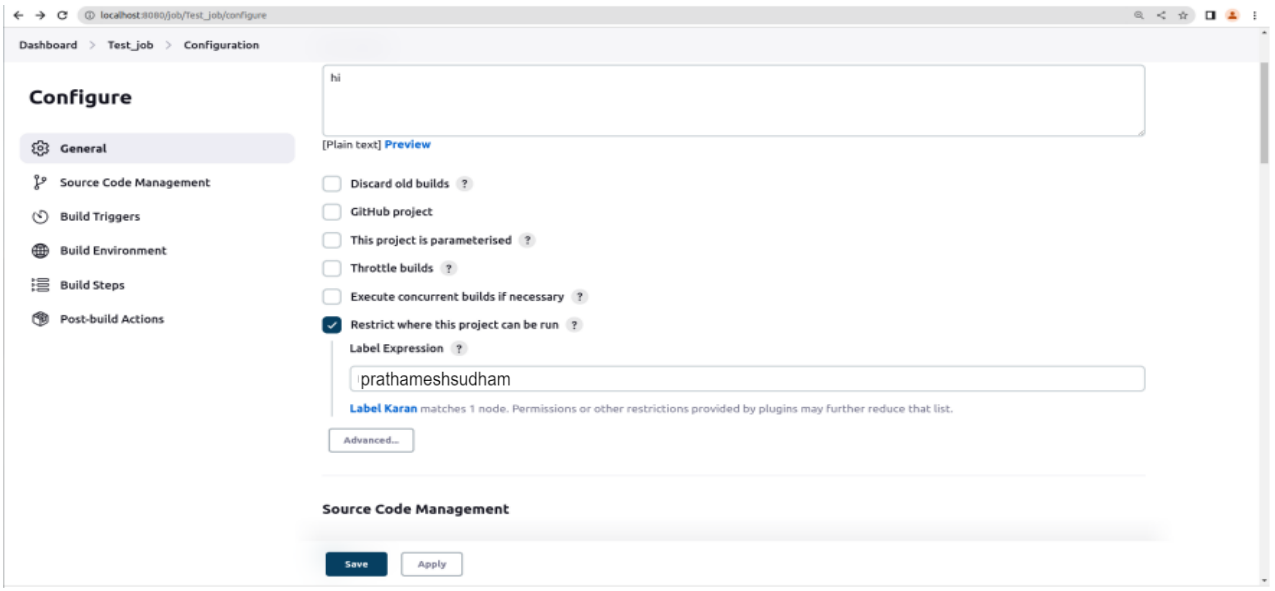
Jenkins standalone implementation -

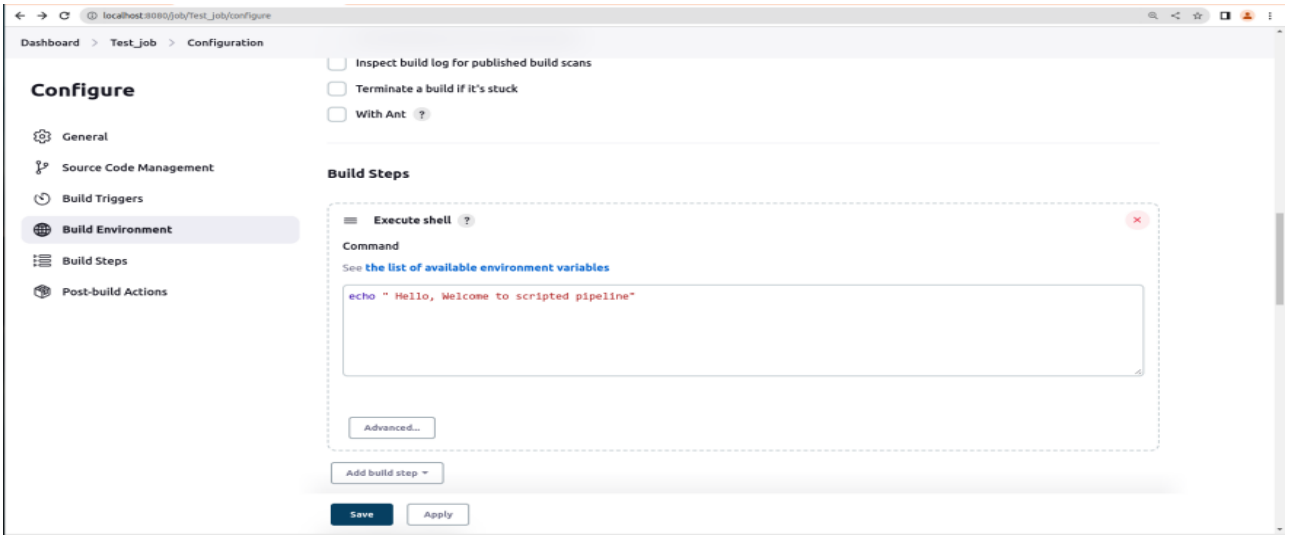
Jenkins is a popular open-source automation server that allows you to automate various aspects of your software development pipeline. It is used for building, testing, and deploying software applications. Jenkins can be implemented in a standalone mode, which means that it is installed and configured on a single machine or server.

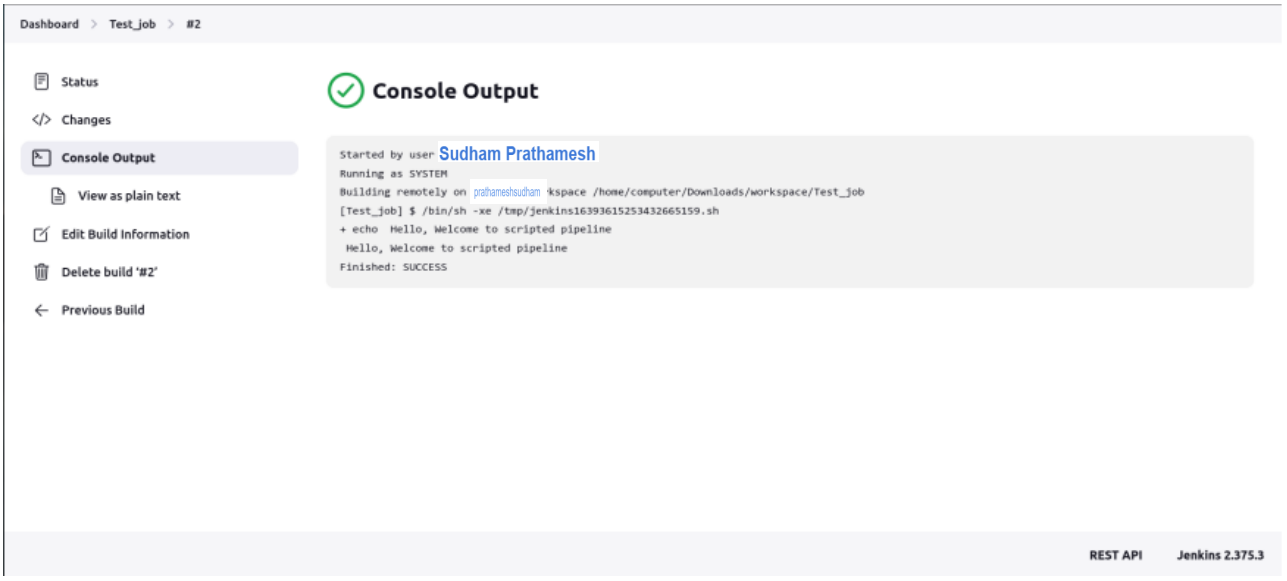
Here are the steps to implement Jenkins in standalone mode:

- Download the Jenkins installer for your operating system from the Jenkins website.
- Install Jenkins on your machine or server by following the instructions provided in the installer.
- Once Jenkins is installed, open a web browser and go to http://localhost:8080 to access the Jenkins web interface.
- Follow the on-screen instructions to set up the initial admin user account.
- Install any necessary plugins that you need for your development pipeline by going to Manage Jenkins -> Manage Plugins -> Available.
- Configure Jenkins to work with your version control system (e.g. Git, SVN) and build tools (e.g. Maven, Gradle) by going to Manage Jenkins -> Global Tool Configuration.
- Create a new Jenkins job by going to the Jenkins home page and clicking on "New Item". Choose the type of job that you want to create (e.g. Freestyle project, Pipeline) and configure it to meet your requirements.
- Run the job to test your pipeline.
- That's it! You now have a fully functional Jenkins server in standalone mode. You can use it to automate your software development pipeline and improve your team's productivity.

Assigning a job to Slave node :







DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

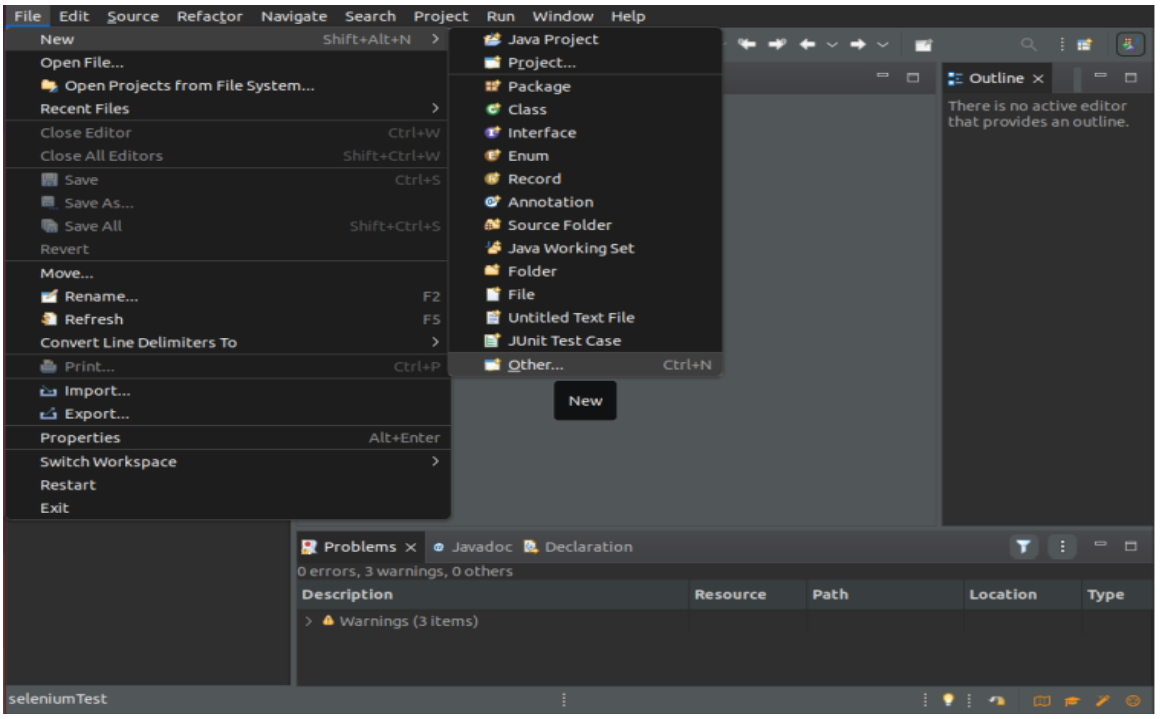
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

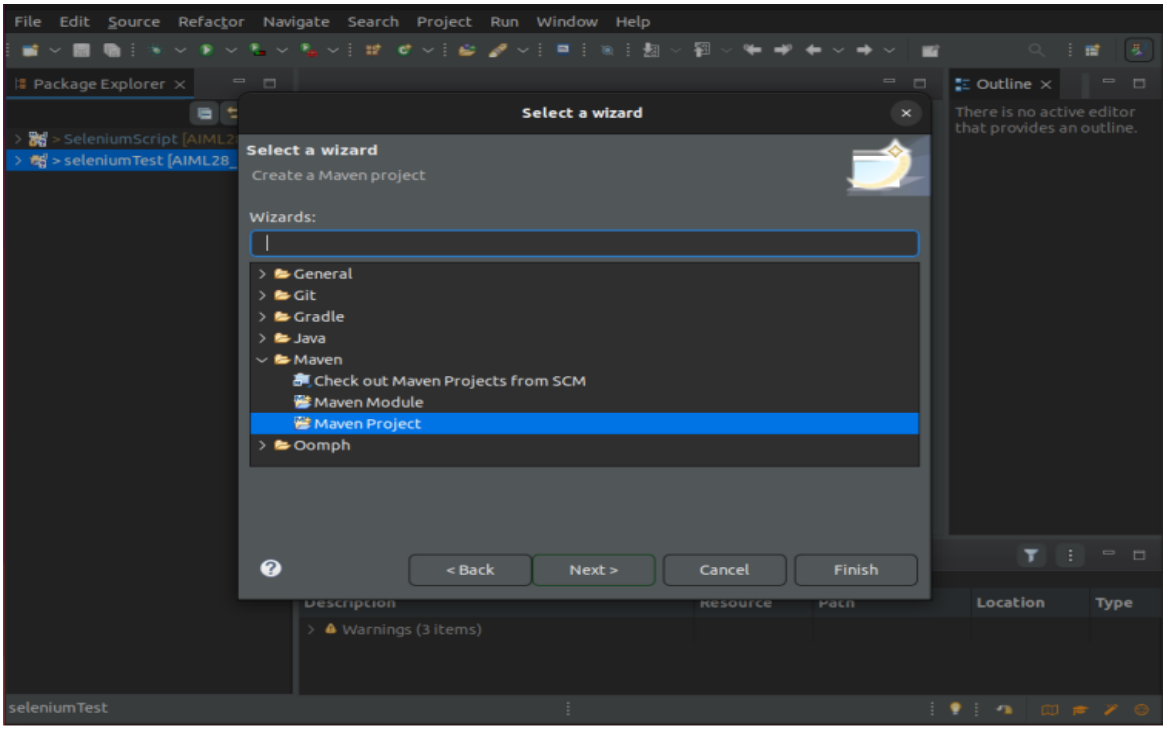
NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**07**
DOP	
DOS	

Selenium Tests in Jenkins Using Maven

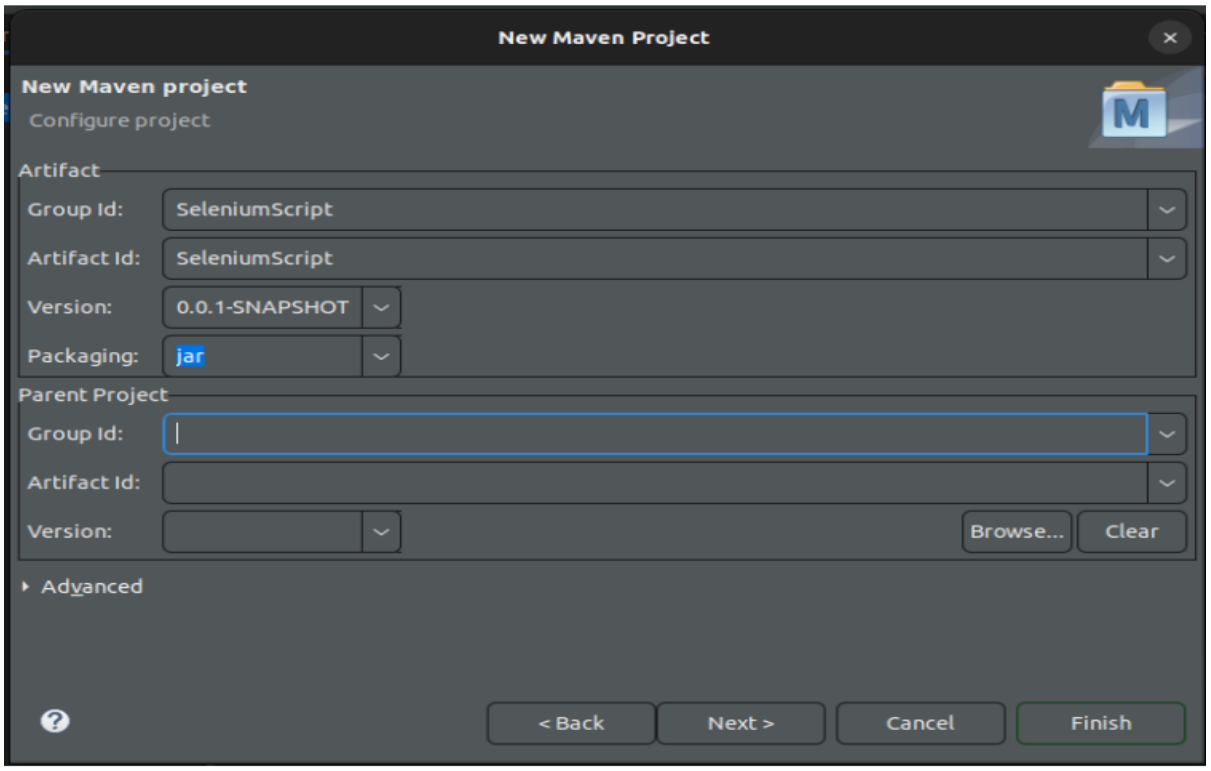
Steps :

1. In Eclipse IDE, create a new project by selecting File | New | Other | Maven Project from Eclipse menu

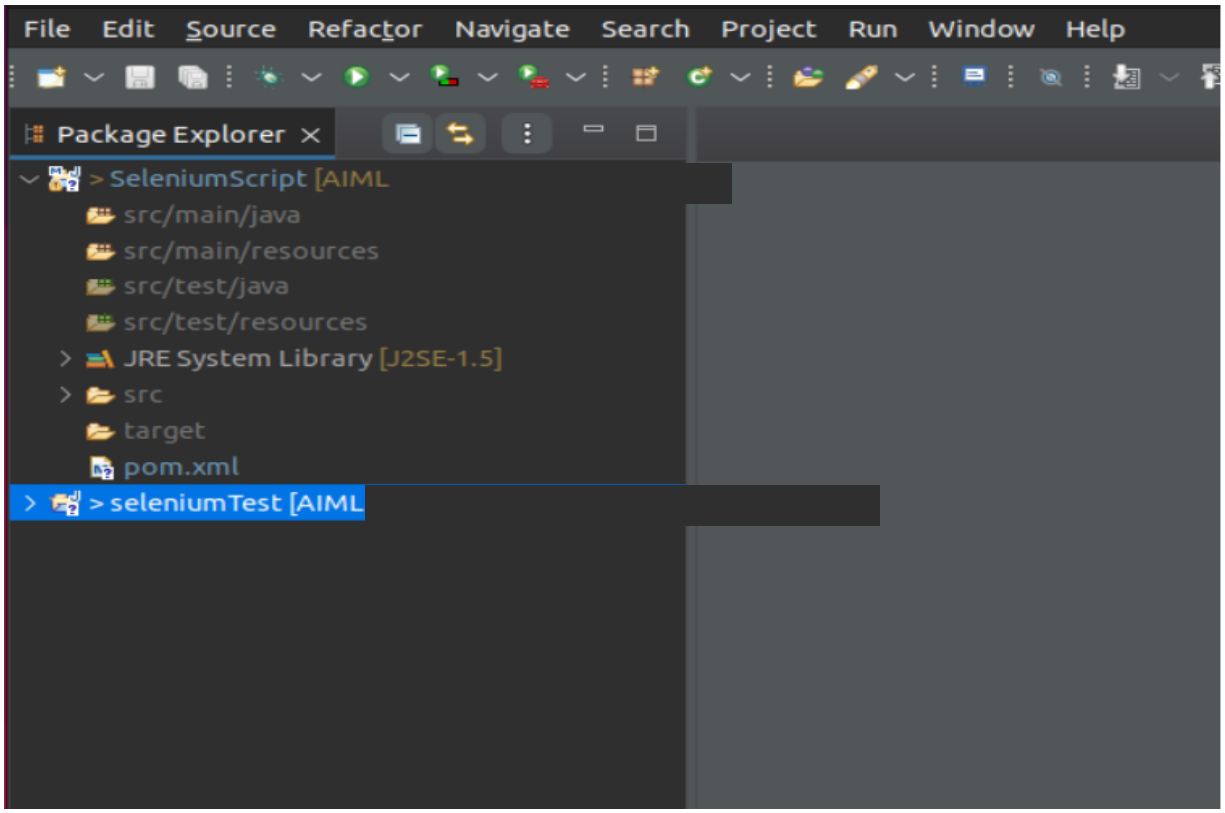




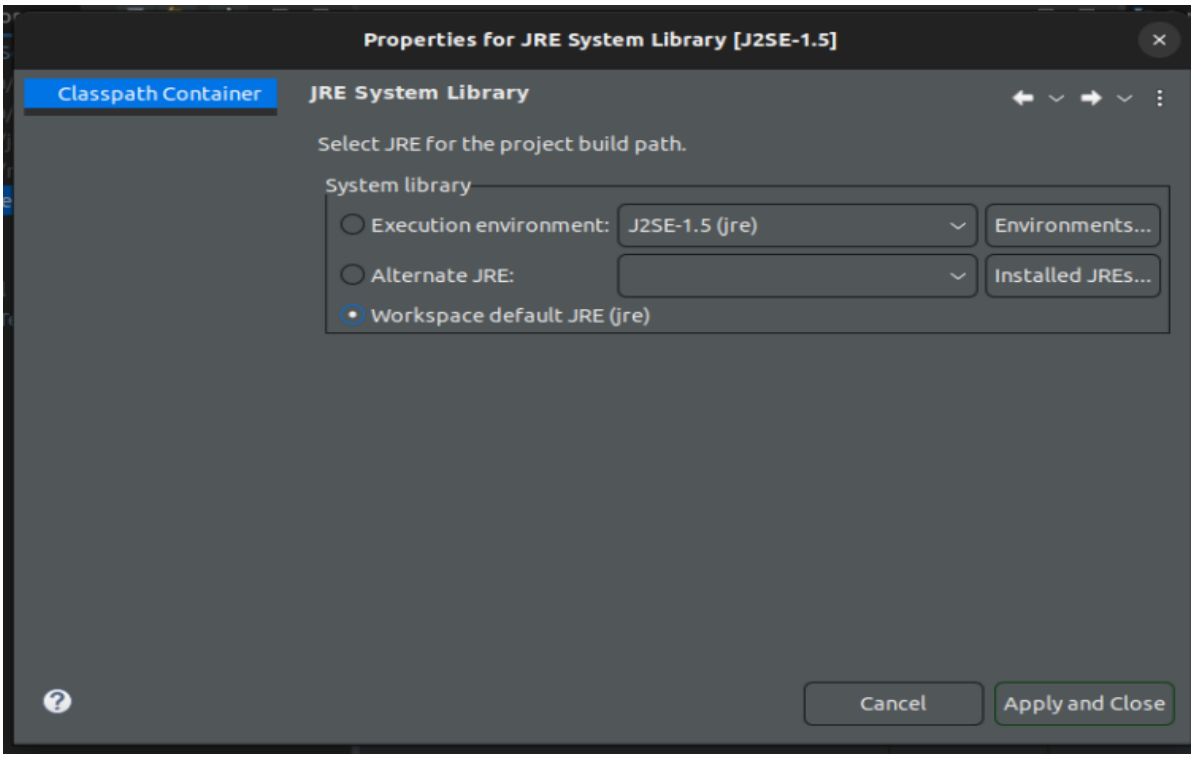
2. On the New Maven Project dialog select the Create a simple project and click Next.
3. Enter SeleniumScript in Group Id: and Artifact Id: and click finish.



4. Eclipse will create WebdriverTest with following structure:

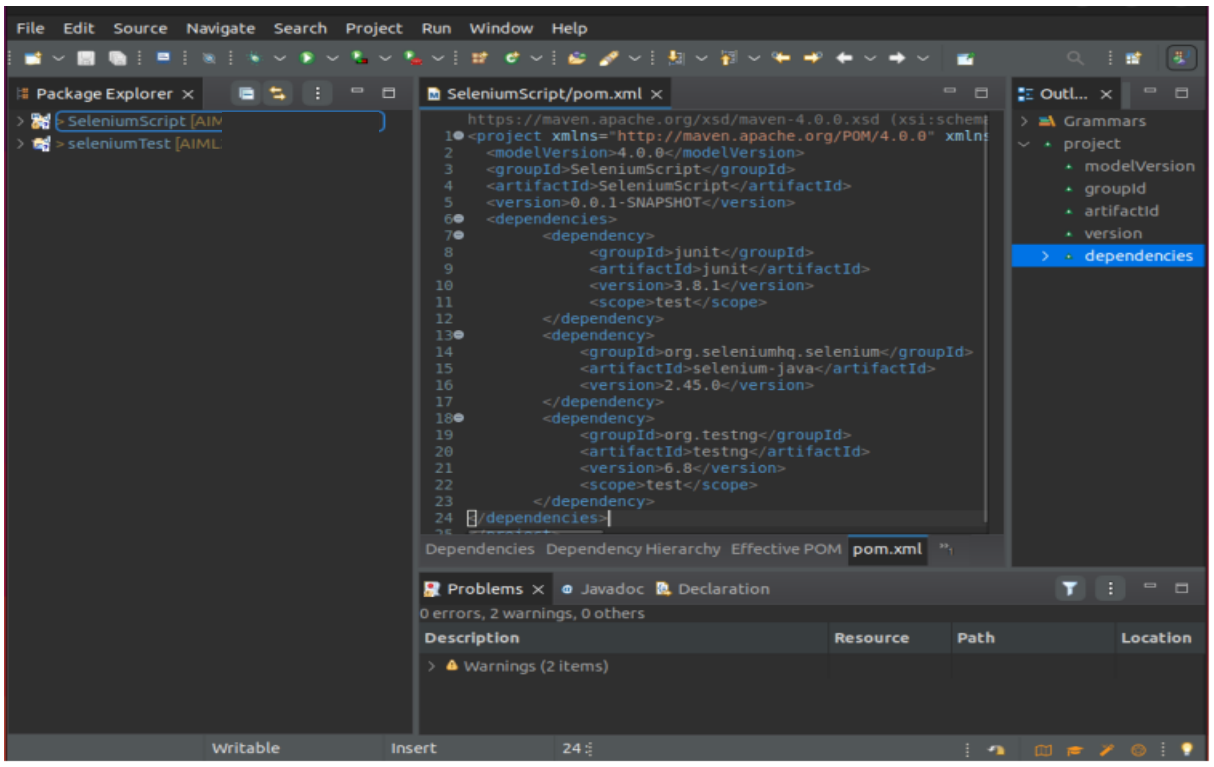


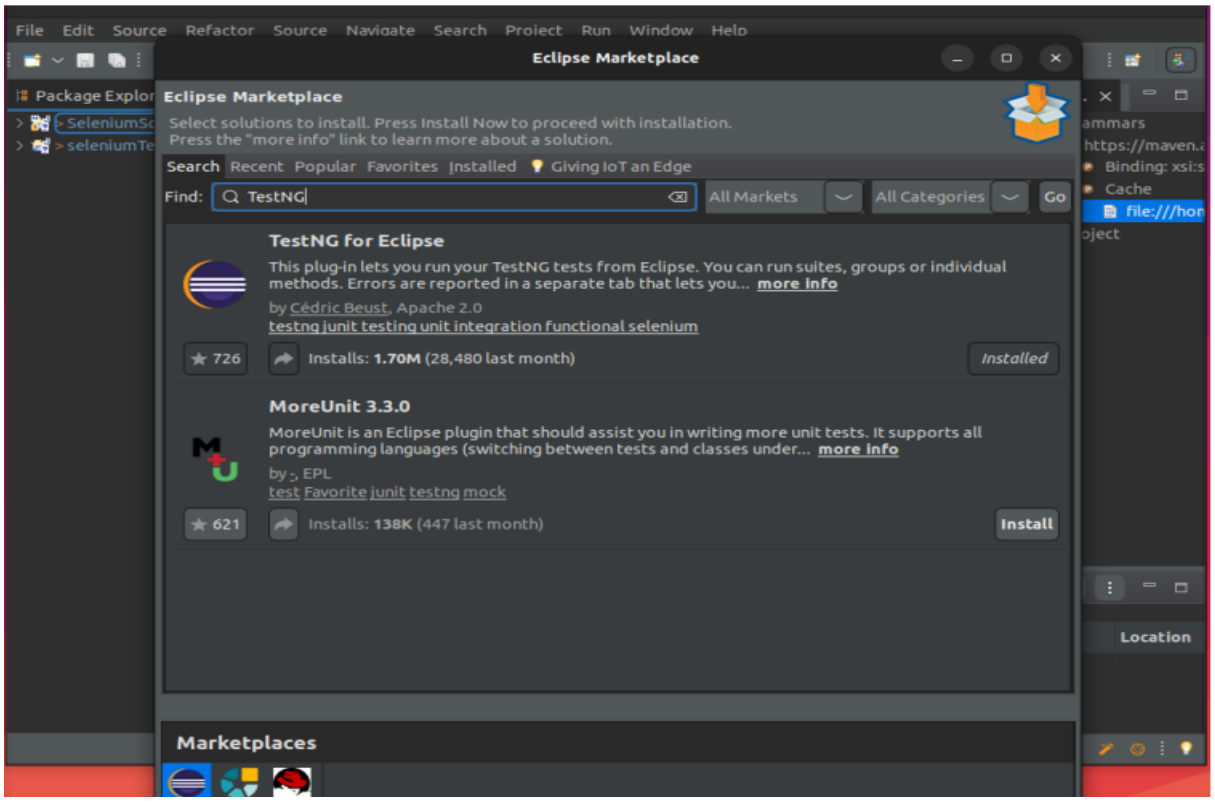
5. Right-click on JRE System Library and select the Properties option from the menu.
6. On the Properties for JRE System Library dialog box, make sure Workspace default JRE is selected and click OK.



7. Select pom.xml from Project Explorer.
8. Add the Selenium, Maven, TestNG, Junit dependencies to pom.xml in the node:

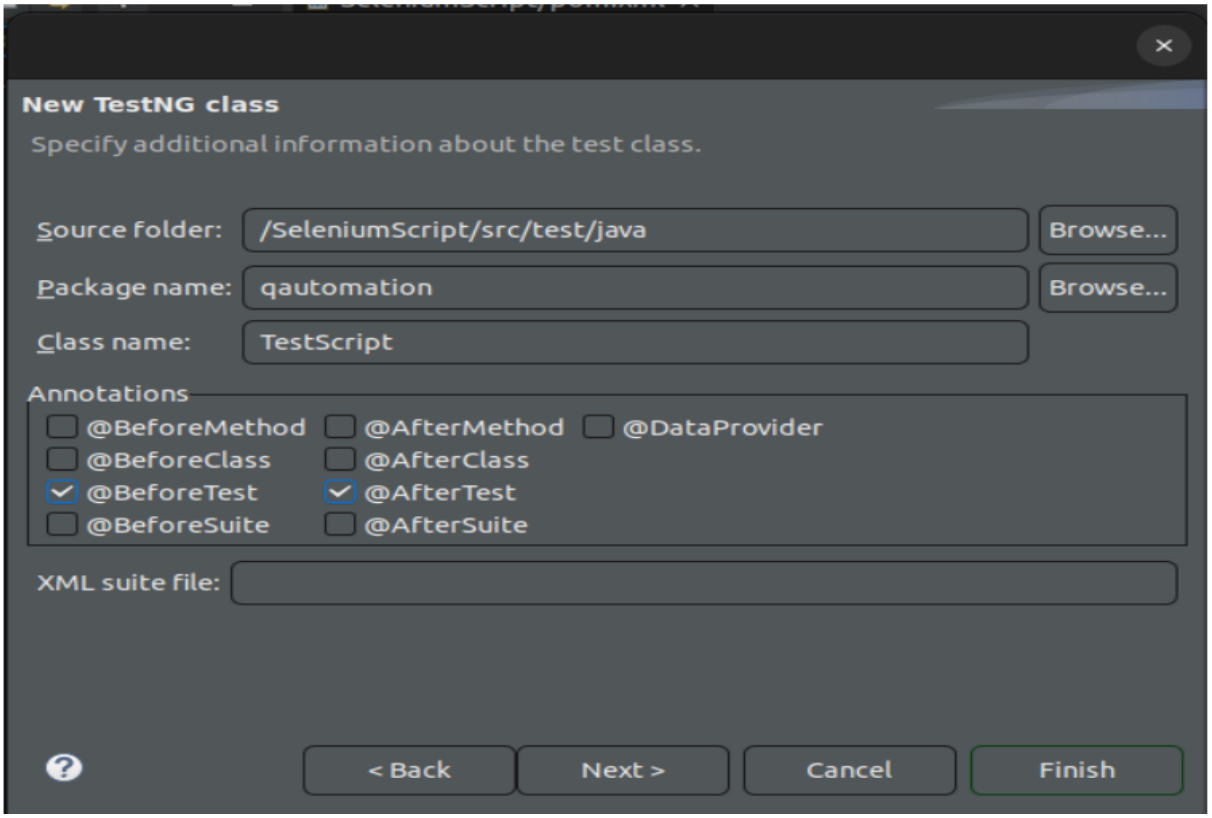
```
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>2.45.0</version>
</dependency>
<dependency>
<groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>6.8</version>
<scope>test</scope>
</dependency>
</dependencies>
```

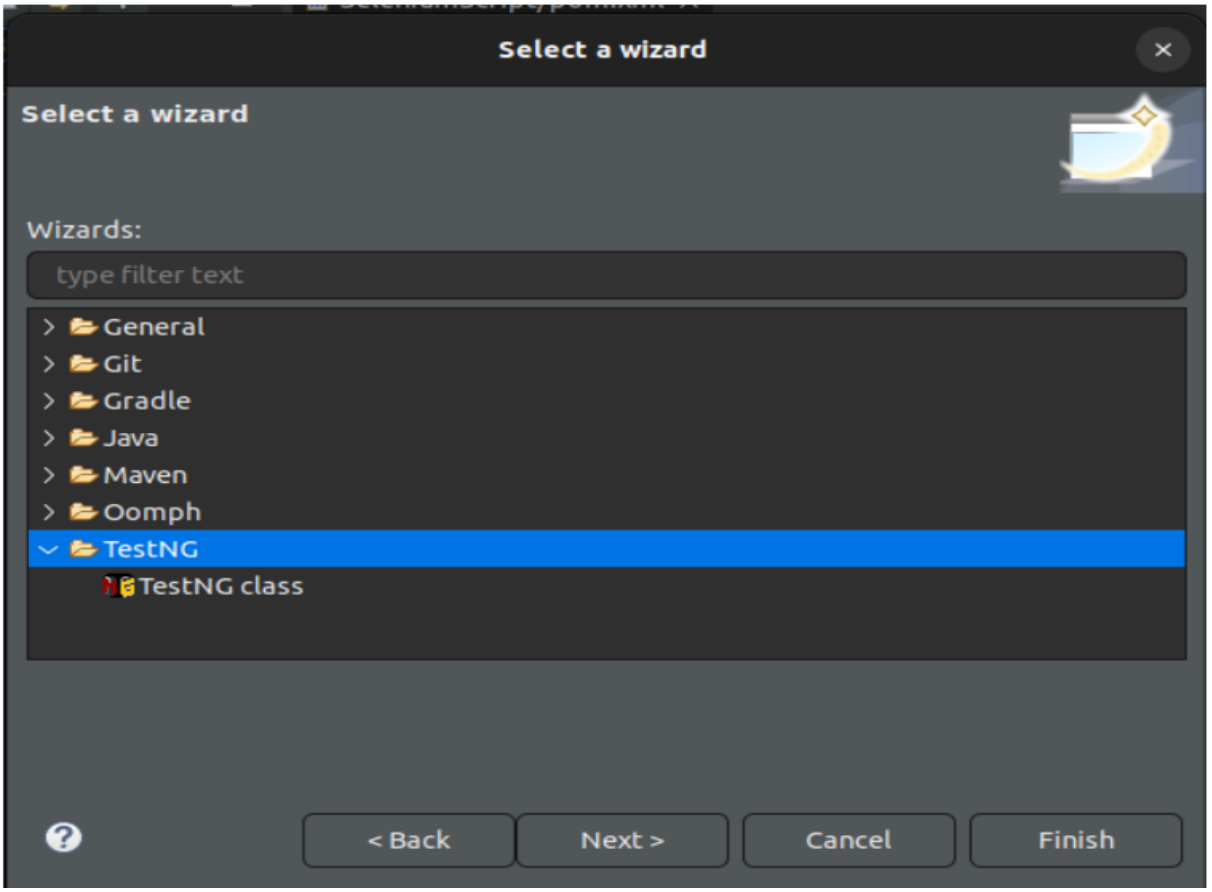




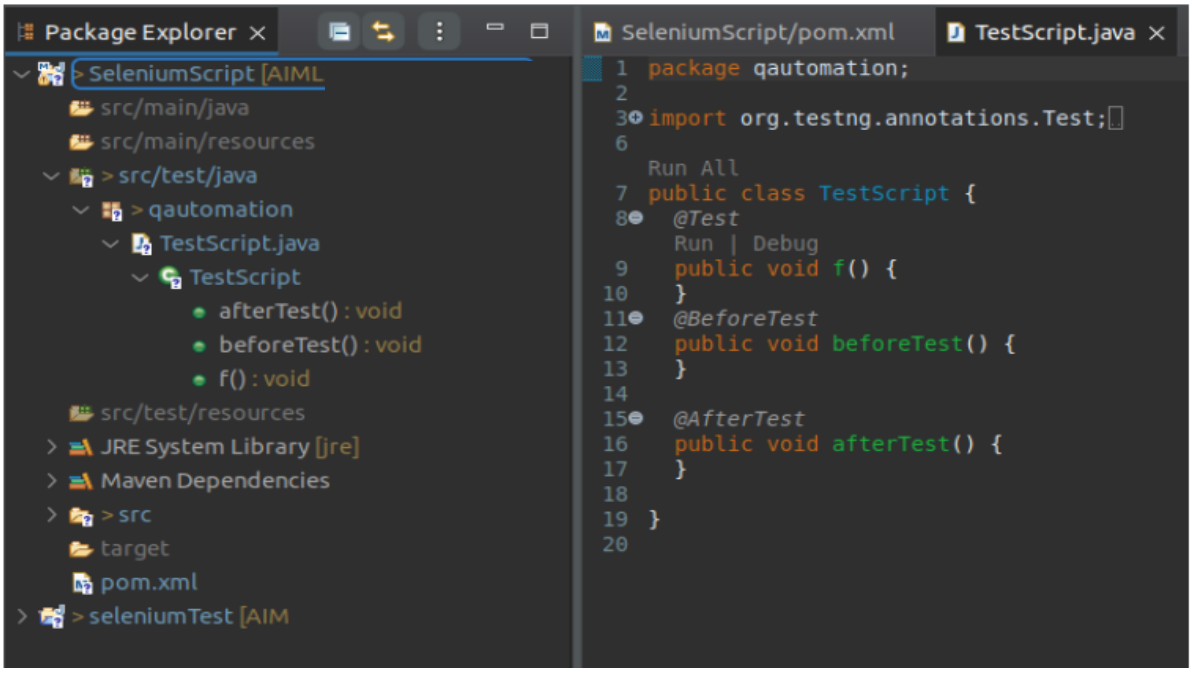
Go to Eclipse Marketplace and download the TestNG if not already installed.
Then restart the IDE to see the below options.

9. Create a New TestNG Class File | New | Others | TestNG | TestNG class. Enter Package name as “Qautomation” and “TestScript” in the Name: textbox and click on the Finish button as shown in the following screenshot:





10. Eclipse will create the TestScript class
11. Add the following code to the TestScript class and respective browser drivers for Chrome , Firefox and IDE :




```
package qautomation;
import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
public class TestScript {
	public static WebDriver driver=null;
	public String browser = System.getProperty("browser");
	public String url = System.getProperty("URL");
	@BeforeTest
	public void beforeTest() {
		if(browser.equalsIgnoreCase("Chrome"))
		{
			System.setProperty("webdriver.chrome.driver",
			System.getProperty("user.dir")+"\\chromedriver.exe");
			Map<String, Object> prefs = new HashMap<String, Object>();
			ChromeOptions options = new ChromeOptions();
			options.setExperimentalOption("prefs", prefs);
```

```
                options.addArguments("--disable-arguments");
                options.addArguments("--test-type");
                options.addArguments("test");
                options.addArguments("disable-infobars");
                driver = new ChromeDriver(options);
        }
        else if(browser.equalsIgnoreCase("FireFox"))
        {
                System.setProperty(FirefoxDriver.SystemProperty.DRIVER_USE_MARIONETTE, "true");
                System.setProperty(FirefoxDriver.SystemProperty.BROWSER_LOGFILE,System.getPro
                perty("user.dir")+"\\FireFoxLogs.txt");
                System.setProperty("webdriver.gecko.driver",
        System.getProperty("user.dir")+"\\geckodriver_v23.exe");
                FirefoxProfile profile = new FirefoxProfile();
                profile.setAcceptUntrustedCertificates(false);
                FirefoxOptions options = new FirefoxOptions().setProfile(profile);
                driver = new FirefoxDriver(options);
                driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
                driver.manage().window().maximize();
        }
        else if (browser.equalsIgnoreCase("IE"))
        {
                System.setProperty("webdriver.ie.driver",
                System.getProperty("user.dir")+"\\IEDriverServer351.exe");
                DesiredCapabilities caps = DesiredCapabilities.internetExplorer();
                caps.setCapability(InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNO
        RING_SECURITY_DOMAINS,true);
                caps.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING,true);
                caps.setCapability(InternetExplorerDriver.UNEXPECTED_ALERT_BEHAVIOR,"accept");
                caps.setCapability(InternetExplorerDriver.REQUIRE_WINDOW_FOCUS,true);
                caps.setCapability(InternetExplorerDriver.INITIAL_BROWSER_URL,"http://www.google.c
        om/");
                driver = new InternetExplorerDriver(caps);
                driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
                driver.manage().window().maximize();
        }
        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
        driver.manage().window().maximize();
    }
@Test
public void TestApplication() {
        driver.get(url);
        String title = driver.getTitle();
        System.out.println("Title="+title);
        Assert.assertTrue(title.contains("QAutomation"));
    }
    @AfterTest
    public void afterTest() {
        driver.quit();
    }
}
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**08**
DOP	
DOS	

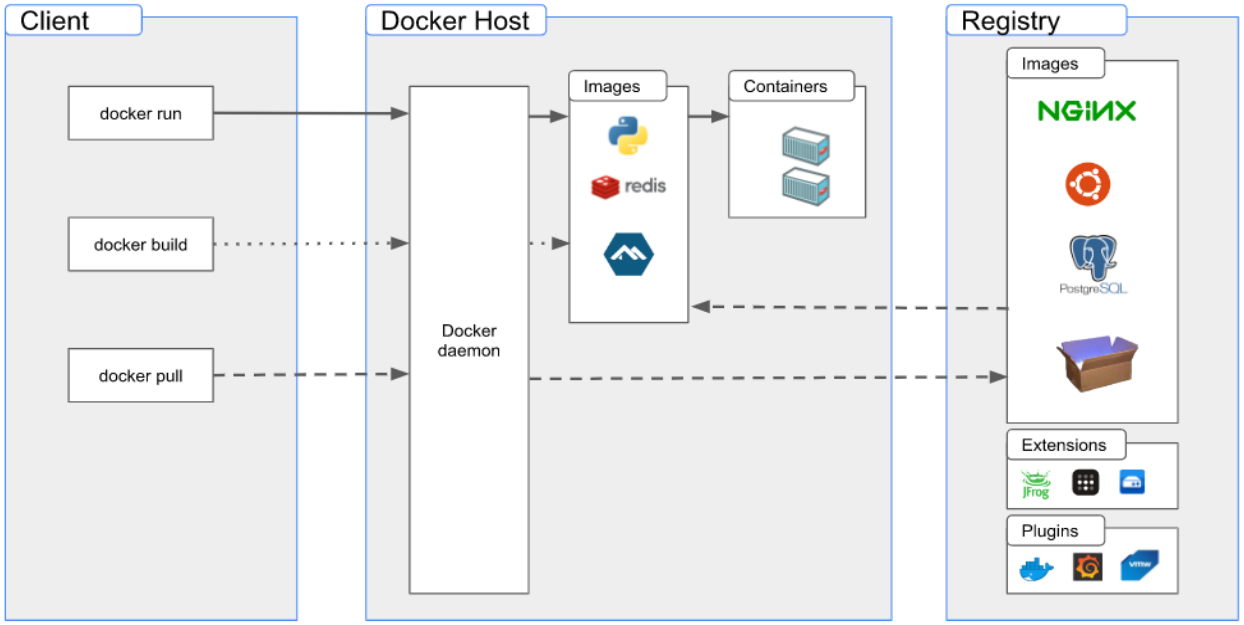
Experiment No. : 08

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

Theory : Docker Architecture and Container Life Cycle

Docker is a containerization platform that allows developers to package applications and their dependencies into portable, self-contained units called containers.

Docker Architecture -

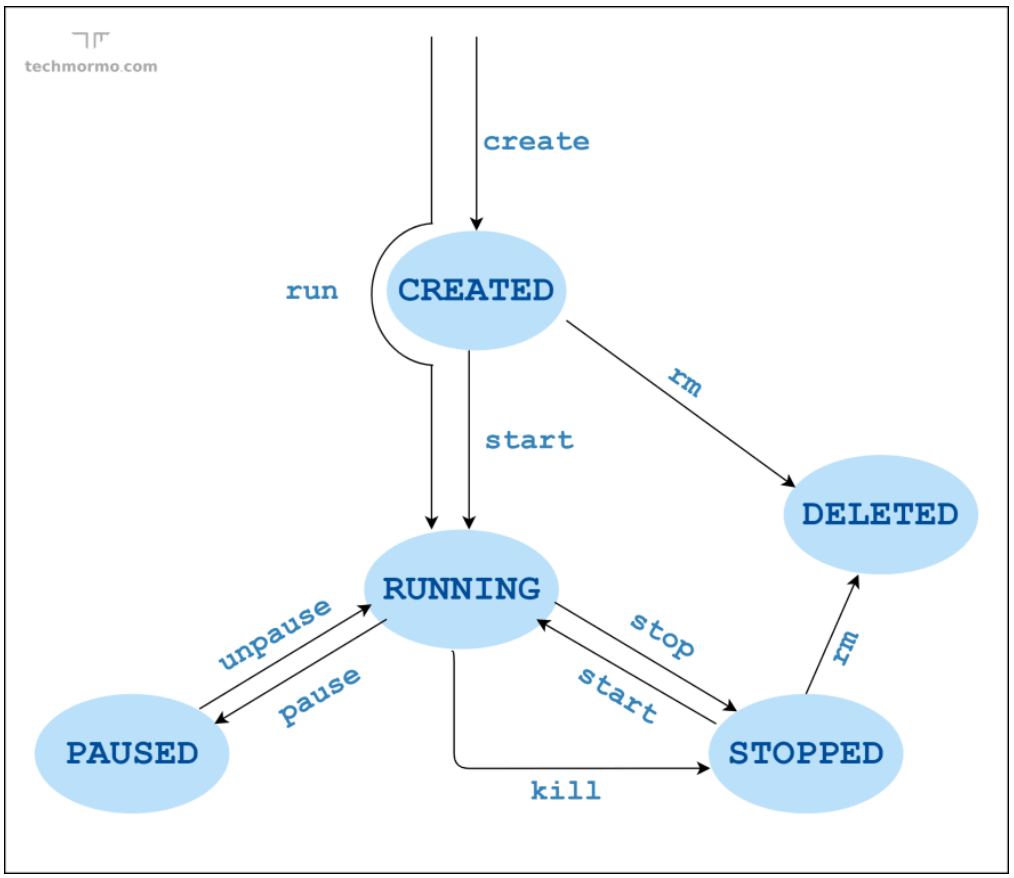


Docker architecture consists of the following components:

- Docker daemon: The Docker daemon is a background process that runs on the host machine and manages Docker objects such as images, containers, networks, and volumes.
- Docker client: The Docker client is a command-line interface (CLI) tool that communicates with the Docker daemon to execute commands and manage Docker objects.
- Docker registry: The Docker registry is a repository that stores Docker images. It can be a public or private registry and can be used to share images between developers or deploy them to production environments.
- Docker image: A Docker image is a read-only template that contains the application code, runtime environment, libraries, and dependencies required to run the application.
- Docker container: A Docker container is a lightweight, standalone, executable package that includes everything needed to run the application. It is created from a Docker image and can be started, stopped, and deleted as needed.
- Docker network: A Docker network is a virtual network that allows containers to communicate with each other and with other services on the host machine.

The life cycle of a Docker container can be broken down into four stages: create, start, stop, and delete.

Container Life Cycle -

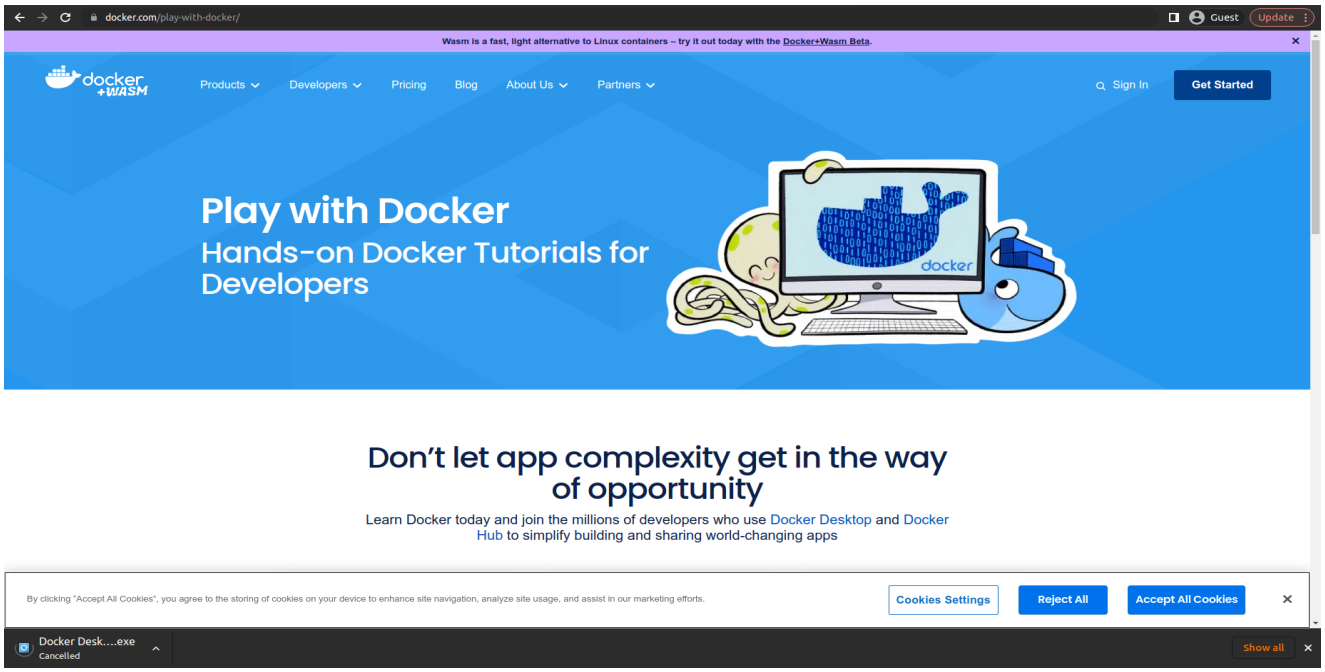


- Create: A Docker container is created by running a command to create a new container from an existing Docker image. The container is created in a stopped state and is not running until it is started.
- Start: A Docker container is started by running a command to start the container. When a container is started, it is assigned a unique identifier and a network interface. The container is now running and can accept requests.
- Stop: A Docker container can be stopped by running a command to stop the container. When a container is stopped, its network interface is removed and it is no longer accessible.
- Delete: A Docker container can be deleted by running a command to remove the container. When a container is deleted, its resources are freed and it is no longer accessible.

During the life cycle of a Docker container, it can be configured with various settings such as environment variables, port mappings, and volumes. These settings can be changed at any point in the container's life cycle, and they will take effect the next time the container is started.

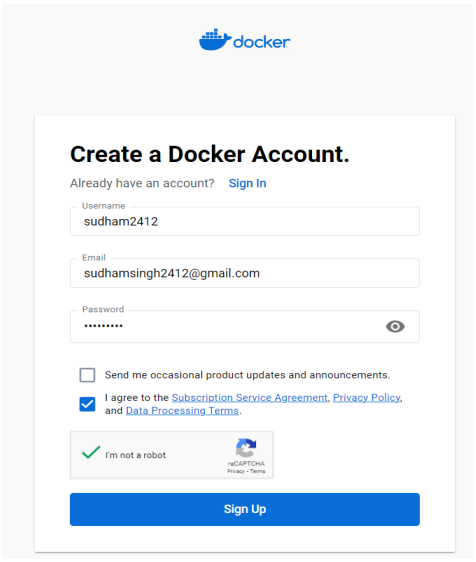
Installing Docker

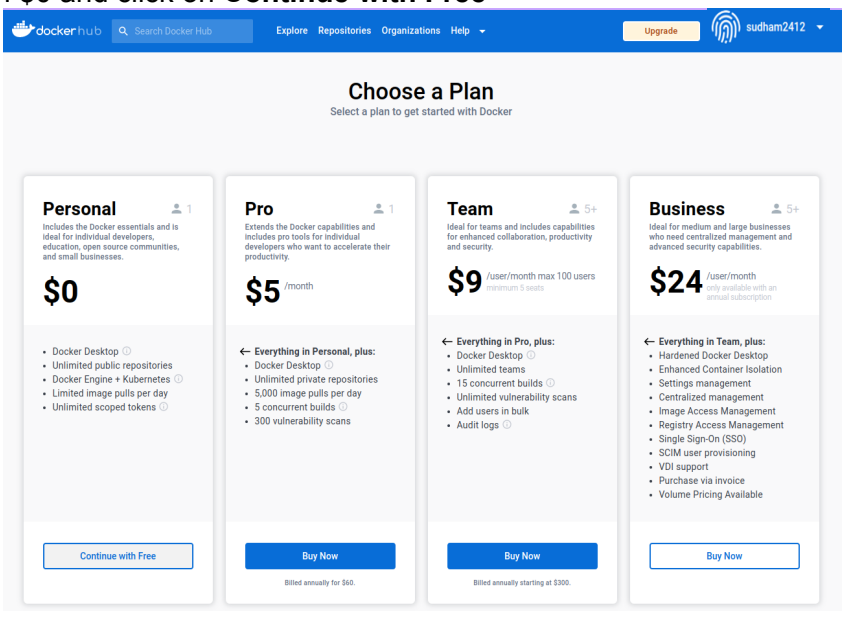
Step 1 - Go to official website of docker : https://hub.docker.com/



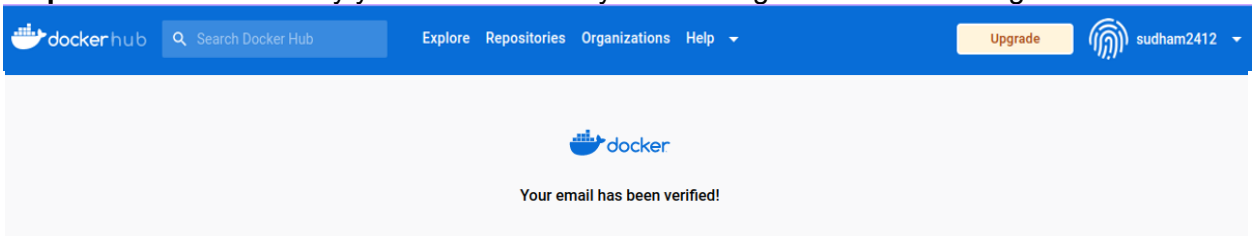
Step 2 - Create your docker account

Step 3 - Choose for personal use with $0 and click on **Continue with Free**



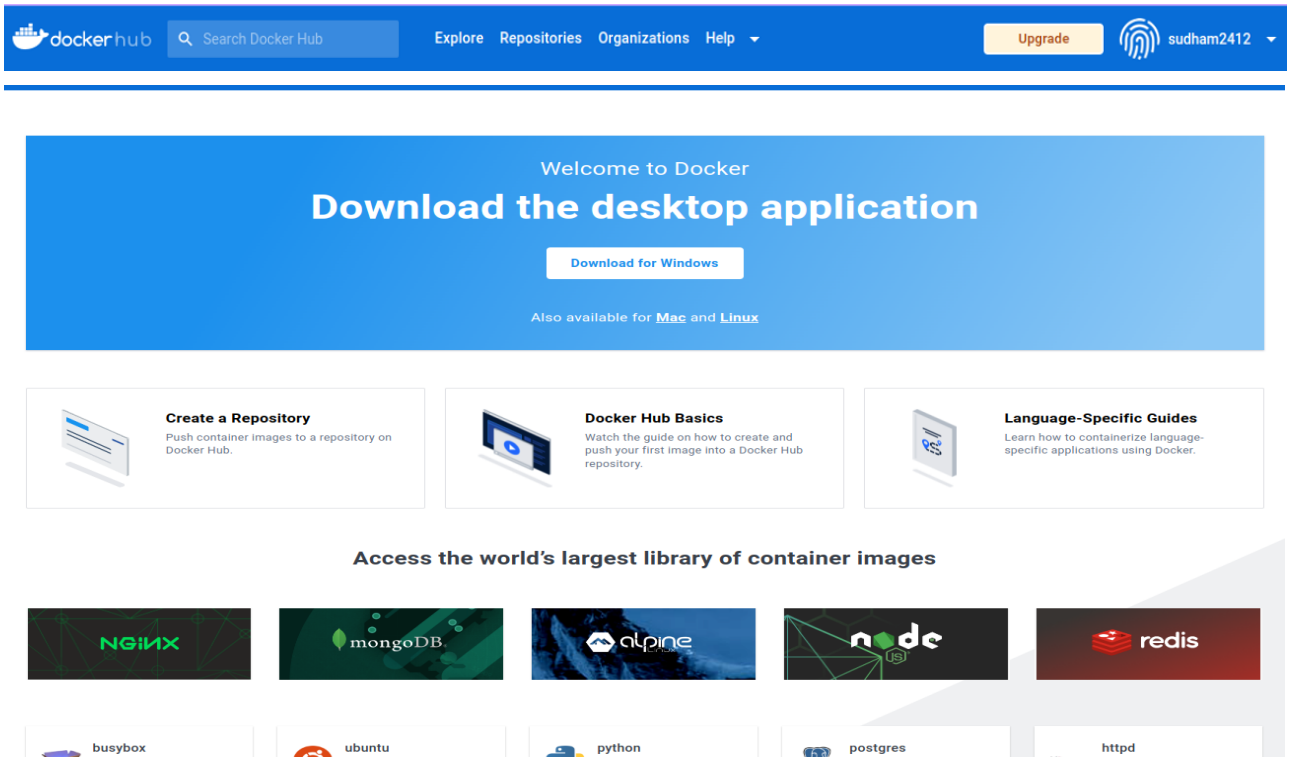


Step 4 - You have to verify your account from your added gmail account through mail



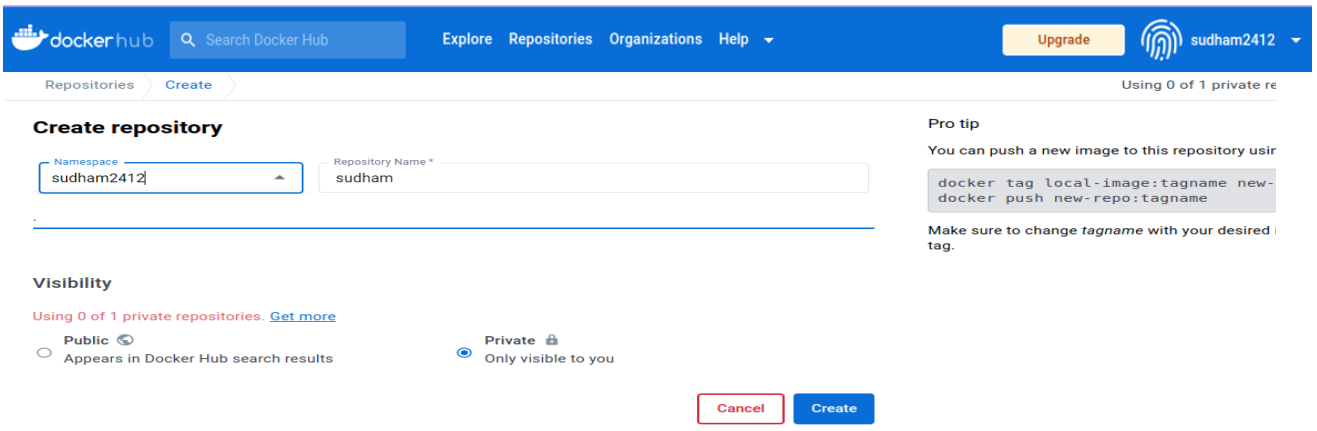
Step 5 - Creating first repository

Click on **Create a Repository**

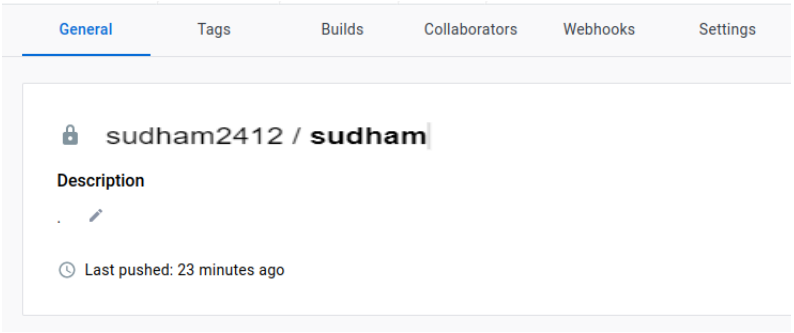


Name it <your-username>/sudham

Set the visibility to private

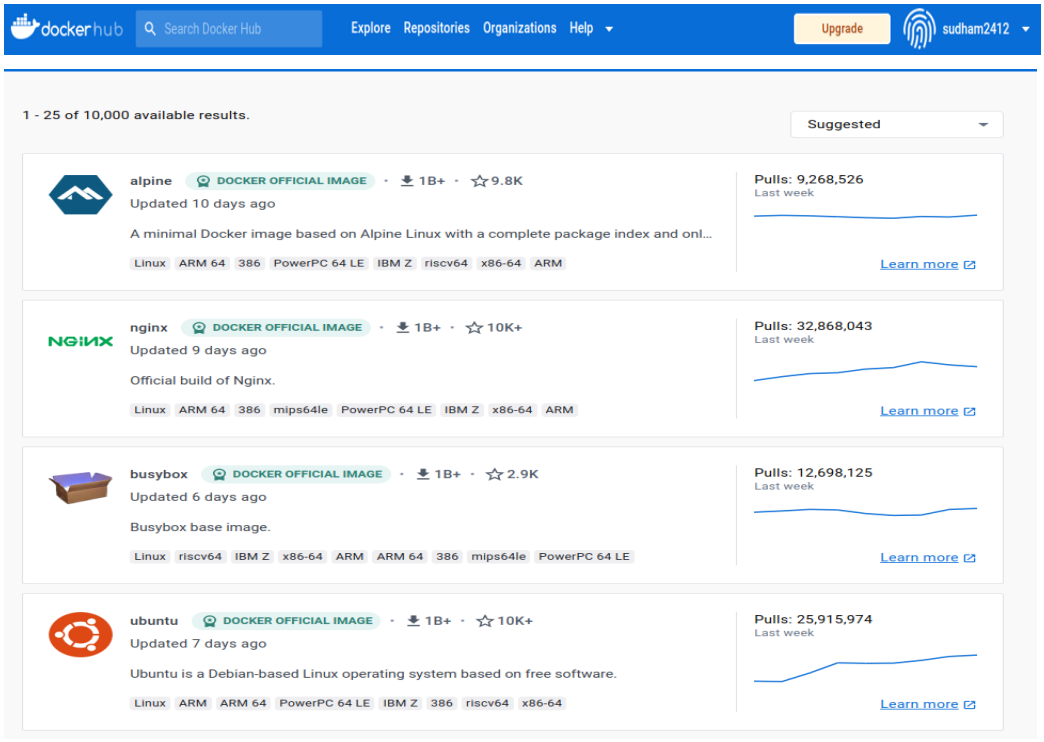


Click on **Create**

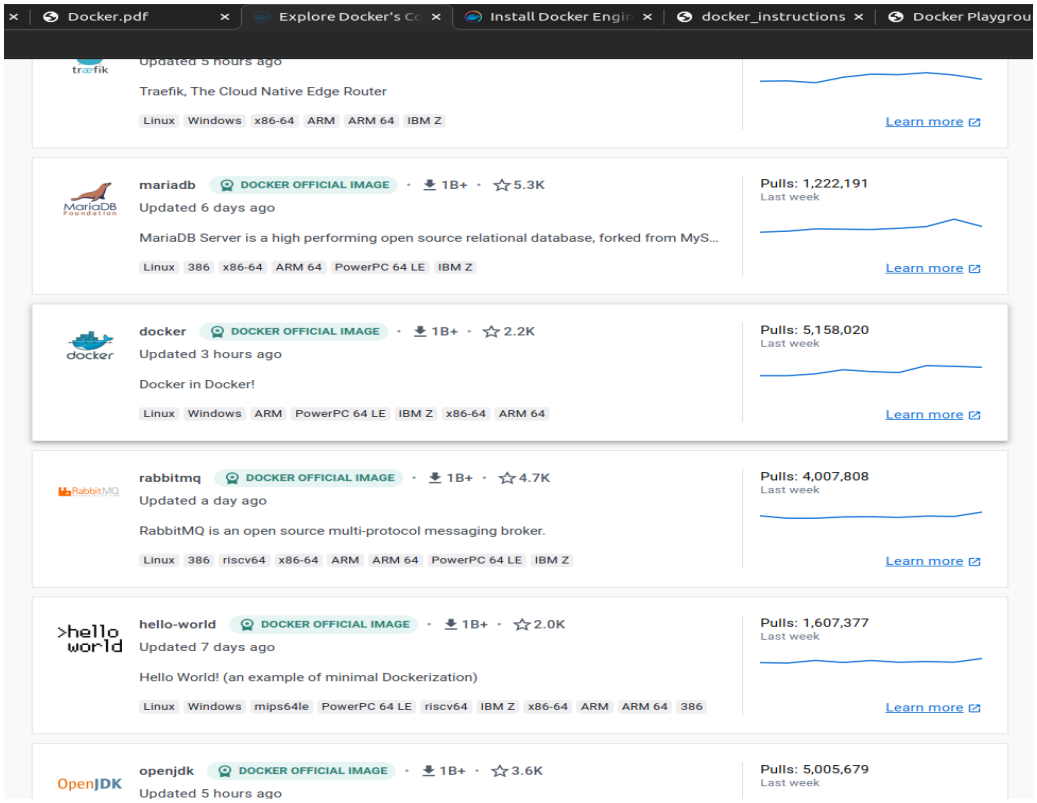


Now next,

Step 6 - Click on Explore tab to see official and publisher images

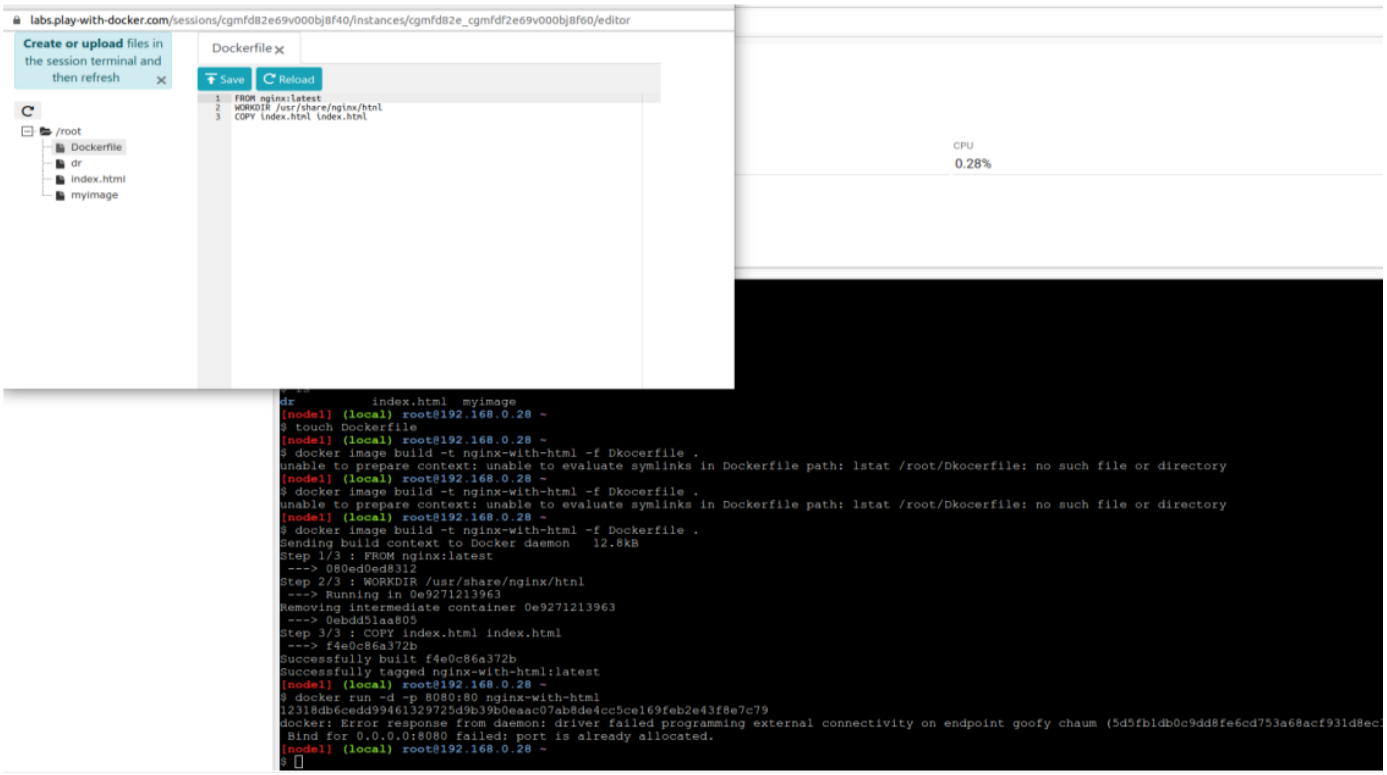


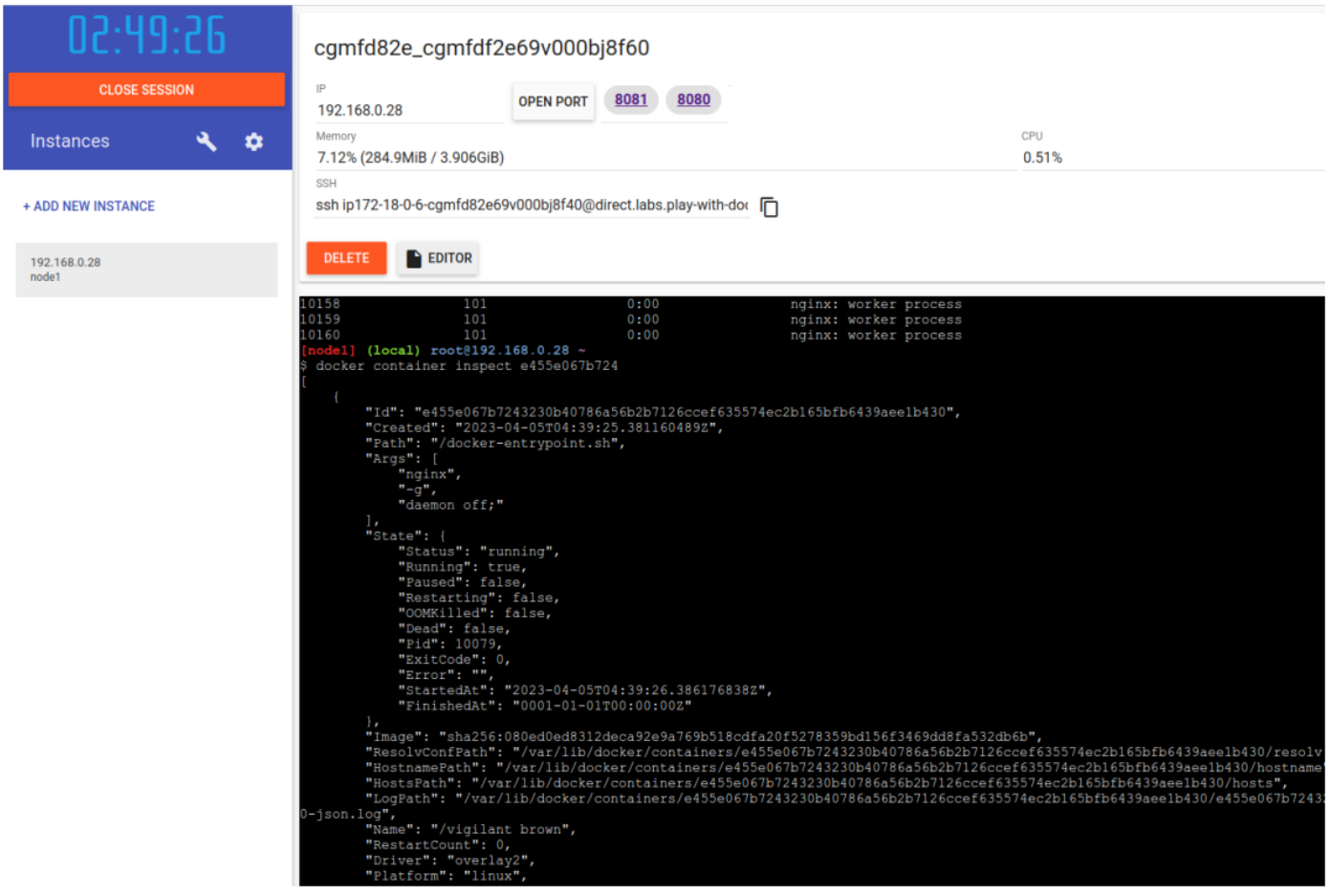
Here you can click on the docker tab

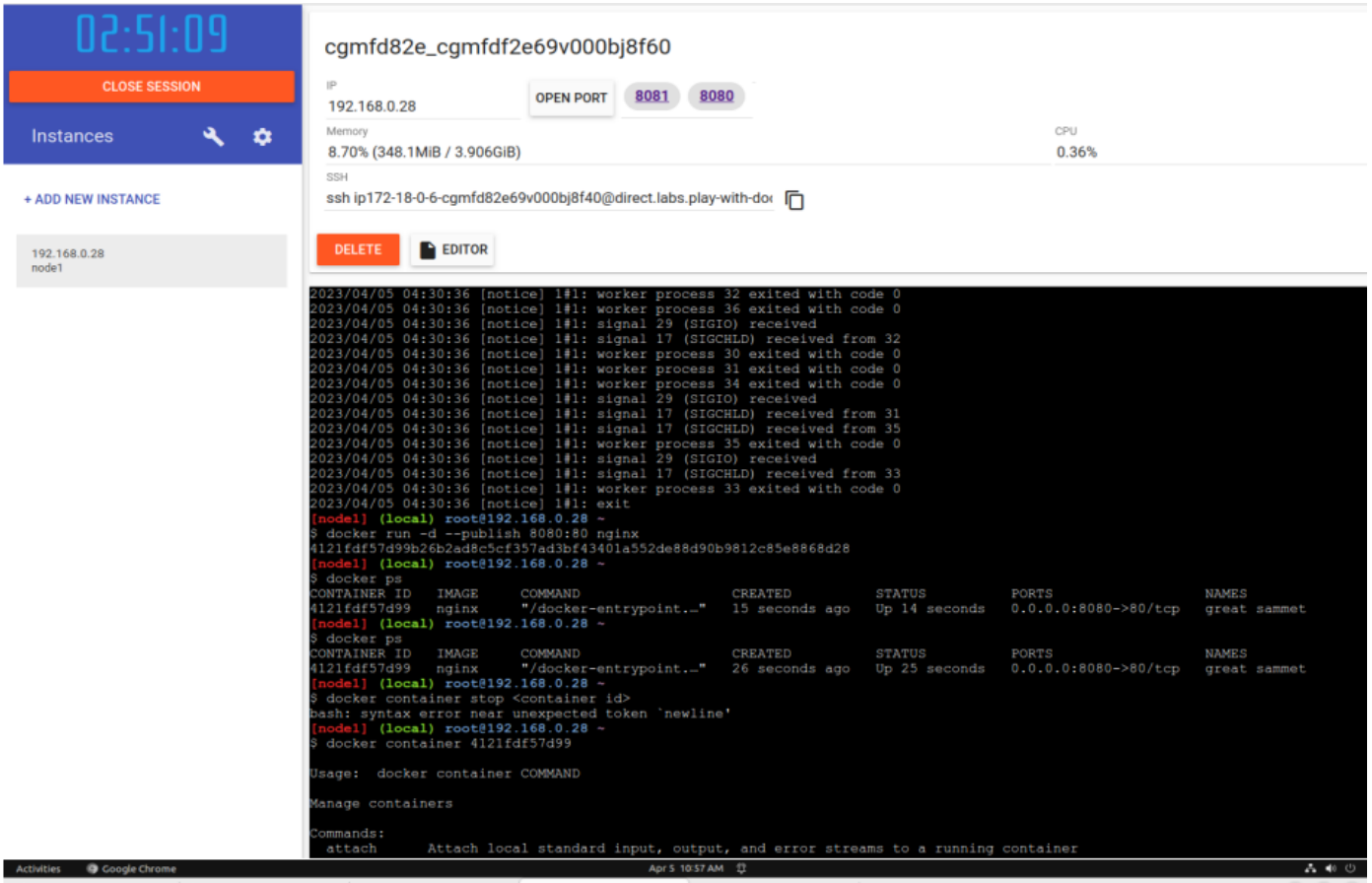


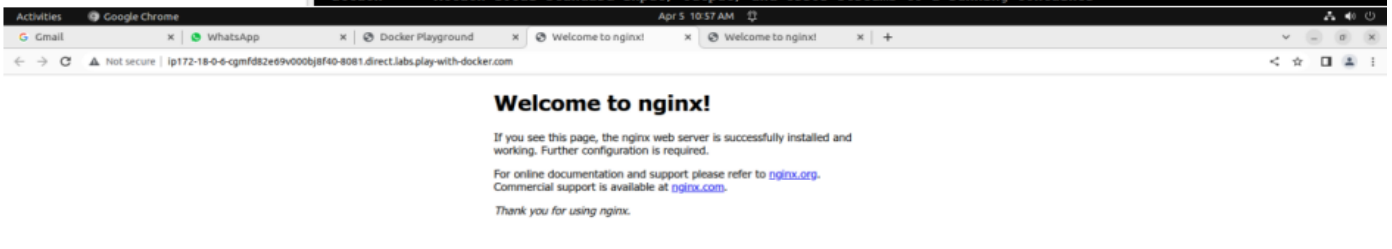
You can explore that tab in which various versions of docker's downloads are given.
You can select various platforms, architecture and operating systems

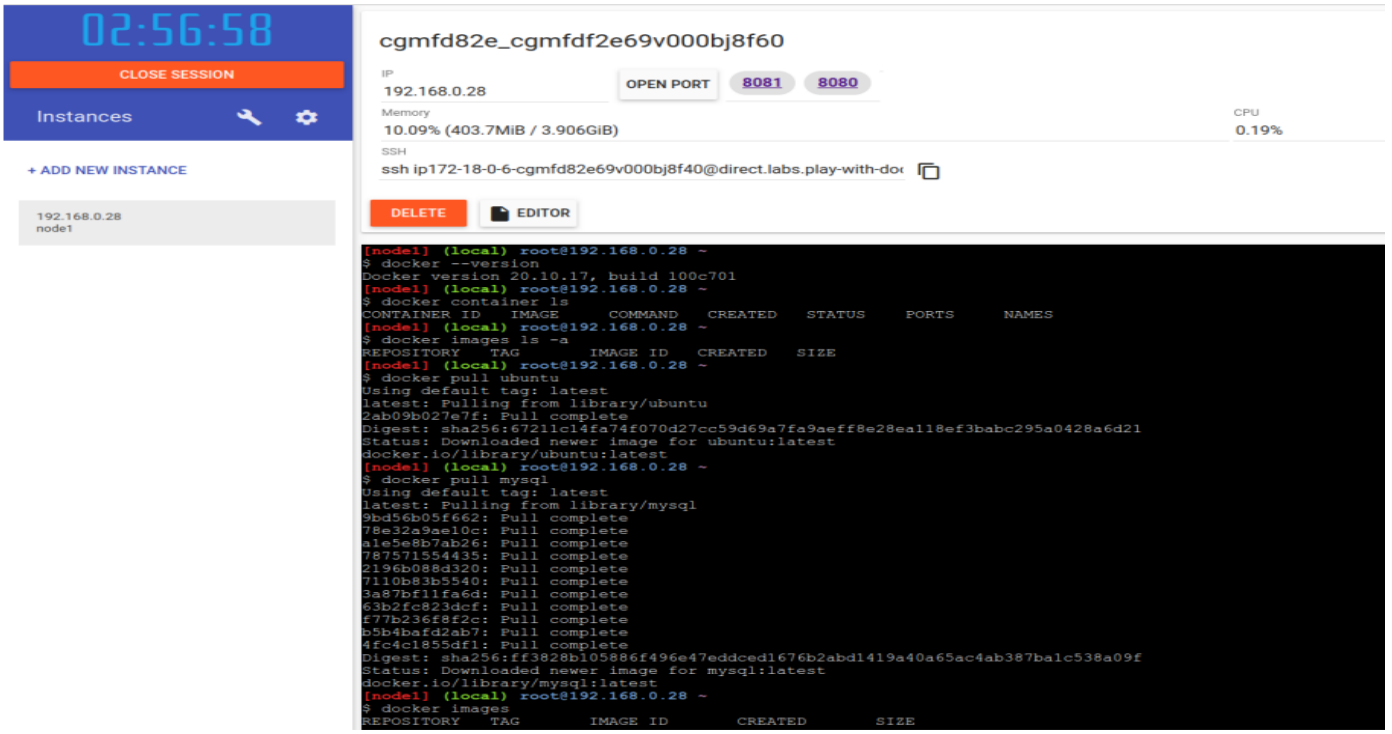
Executing docker commands






02:51:09
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
192.168.0.28
node1
cgmfd82e_cgmfdf2e69v000bj8f60
IP
192.168.0.28
OPEN PORT
8081
8080
Memory
8.70% (348.1MiB / 3.906GiB)
CPU
0.36%
SSH
ssh ip172-18-0-6-cgmfd82e69v000bj8f40@direct.labs.play-with-do
DELETE
EDITOR
2023/04/05 04:30:36 [notice] 1#1: worker process 32 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: worker process 36 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: signal 29 (SIGIO) received
2023/04/05 04:30:36 [notice] 1#1: signal 17 (SIGCHLD) received from 32
2023/04/05 04:30:36 [notice] 1#1: worker process 30 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: worker process 31 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: worker process 34 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: signal 29 (SIGIO) received
2023/04/05 04:30:36 [notice] 1#1: signal 17 (SIGCHLD) received from 31
2023/04/05 04:30:36 [notice] 1#1: signal 17 (SIGCHLD) received from 35
2023/04/05 04:30:36 [notice] 1#1: worker process 35 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: signal 29 (SIGIO) received
2023/04/05 04:30:36 [notice] 1#1: signal 17 (SIGCHLD) received from 33
2023/04/05 04:30:36 [notice] 1#1: worker process 33 exited with code 0
2023/04/05 04:30:36 [notice] 1#1: exit
[node1] (local) root@192.168.0.28 ~
$ docker run -d --publish 8080:80 nginx
4121fdf57d99b26b2ad8c5cf357ad3bf43401a552de88d90b9812c85e8868d28
[node1] (local) root@192.168.0.28 ~
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4121fdf57d99 nginx "/docker-entrypoint.…" 15 seconds ago Up 14 seconds 0.0.0.0:8080->80/tcp great sammet
[node1] (local) root@192.168.0.28 ~
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4121fdf57d99 nginx "/docker-entrypoint.…" 26 seconds ago Up 25 seconds 0.0.0.0:8080->80/tcp great sammet
[node1] (local) root@192.168.0.28 ~
$ docker container stop <container id>
bash: syntax error near unexpected token `newline'
[node1] (local) root@192.168.0.28 ~
$ docker container 4121fdf57d99
Usage: docker container COMMAND
Manage containers
Commands:
attach Attach local standard input, output, and error streams to a running container


Activities
Google Chrome
Apr 5 10:57 AM
Gmail
WhatsApp
Docker Playground
Welcome to nginx!
Welcome to nginx!
Not secure
ip172-18-0-6-cgmfd82e69v000bj8f40-8081.direct.labs.play-with-docker.com
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.


02:56:58
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
192.168.0.28
node1
cgmfd82e_cgmfdf2e69v000bj8f60
IP
192.168.0.28
OPEN PORT
8081
8080
Memory
10.09% (403.7MiB / 3.906GiB)
CPU
0.19%
SSH
ssh ip172-18-0-6-cgmfd82e69v000bj8f40@direct.labs.play-with-do
DELETE
EDITOR
[node1] (local) root@192.168.0.28 ~
$ docker --version
Docker version 20.10.17, build 100c701
[node1] (local) root@192.168.0.28 ~
$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[node1] (local) root@192.168.0.28 ~
$ docker images ls -a
REPOSITORY TAG IMAGE ID CREATED SIZE
[node1] (local) root@192.168.0.28 ~
$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
2ab09b027e7f: Pull complete
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[node1] (local) root@192.168.0.28 ~
$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
9bd56b05f662: Pull complete
78e32a9ae10c: Pull complete
a1e5e8b7ab26: Pull complete
78757155443 5: Pull complete
2196b088d320: Pull complete
7110b83b5540: Pull complete
3a87bf11fa6d: Pull complete
63b2fc823dcf: Pull complete
f77b236f8f2c: Pull complete
b5b4bafd2ab7: Pull complete
4fc4c1855df1: Pull complete
Digest: sha256:ff3828b105886f496e47eddced1676b2abd1419a40a65ac4ab387ba1c538a09f
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
[node1] (local) root@192.168.0.28 ~
$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**09**
DOP	
DOS	

Dockerfile instructions

- **Installing using apt repository on ubuntu -**

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

Step 1 - Update the apt package index and install packages to allow apt to use a repository over HTTPS:

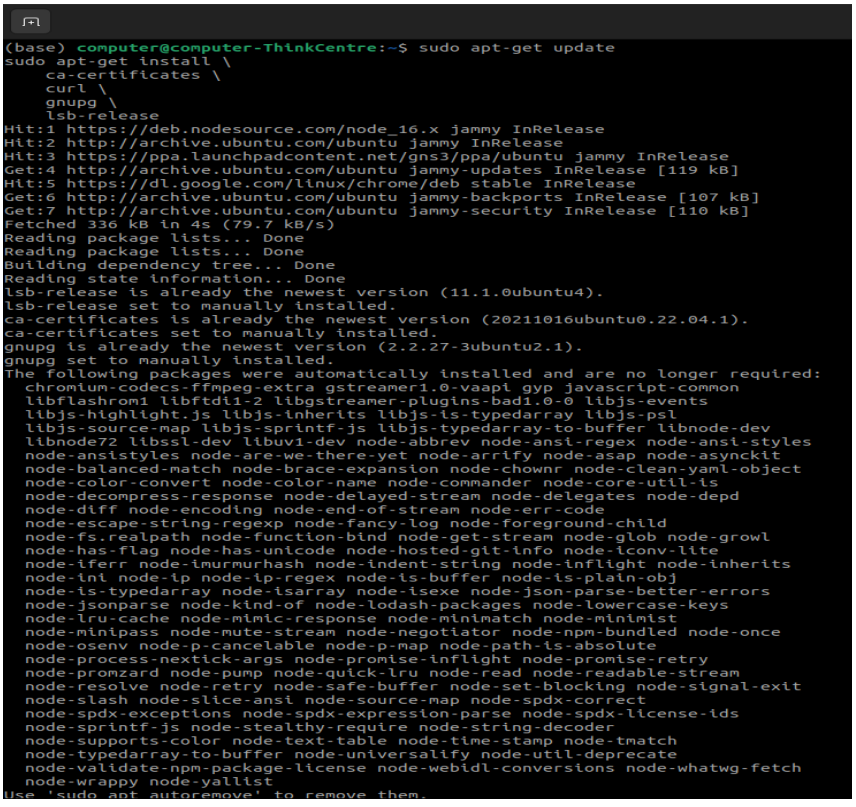
```
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg
```

Step 2 - Add Docker's official GPG key:

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Step 3 - Use the following command to set up the repository:

```
echo \
 "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
 "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```



- **Installing Docker Engine -**

Step 1 - Update the apt package index : **sudo apt-get update**

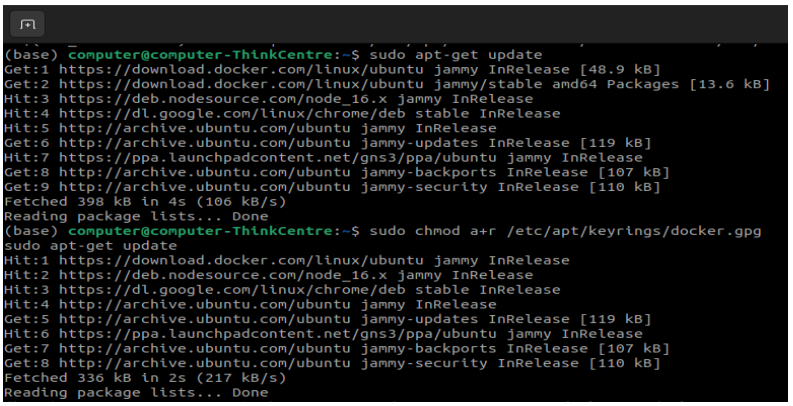
Receiving a GPG error when running apt-get update?

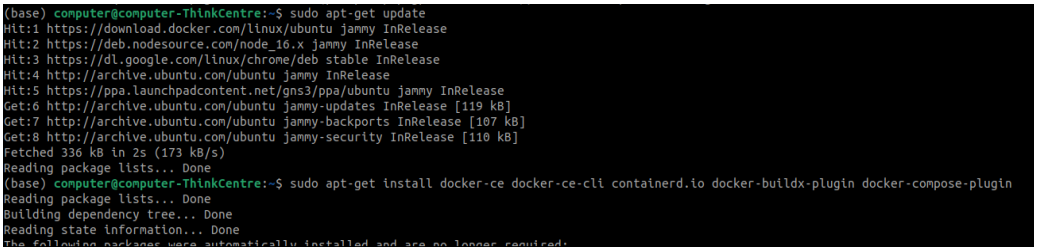
Your default umask may be incorrectly configured, preventing detection of the repository public key file. Try granting read permission for the Docker public key file before updating the package index : **sudo chmod a+r /etc/apt/keyrings/docker.gpg**

sudo apt-get update

Step 2 - Install Docker Engine, containerd, and Docker Compose.

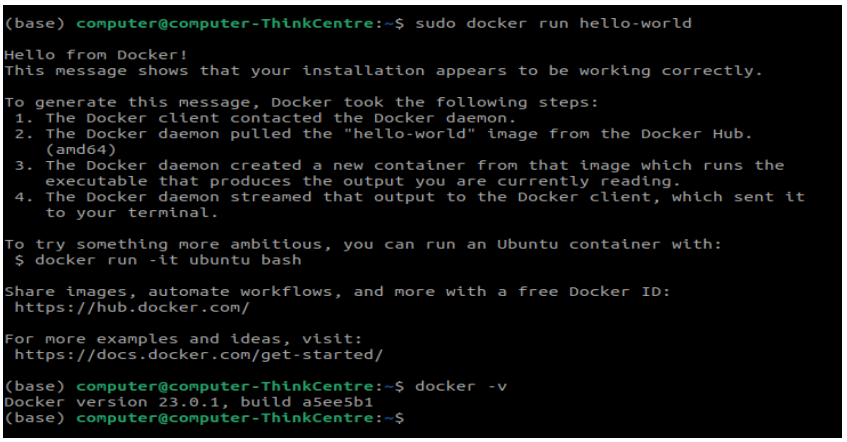
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin





Step 3 - Verify that the Docker Engine installation is successful by running the hello-world image : **sudo docker run hello-world**

Step 4 - Check docker version to ensure that its installed successfully : **docker -v**



Building an image for a sample web application using Dockerfile

(Creating containerised docker)

Step 1 - Cloning : **git clone https://github.com/docker/Cloning into 'getting-started'...**

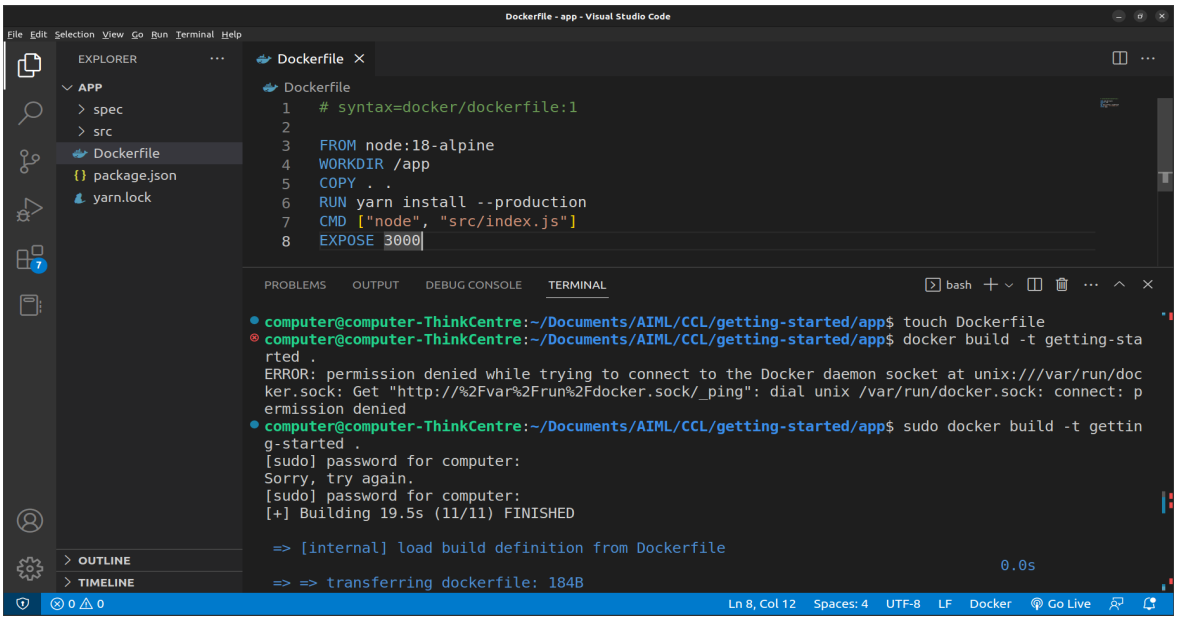
Step 2 - Open code : **code ./getting-started/app/**

```
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$ git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 957, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 957 (delta 0), reused 1 (delta 0), pack-reused 952
Receiving objects: 100% (957/957), 5.24 MiB | 21.05 MiB/s, done.
Resolving deltas: 100% (541/541), done.
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$ code ./getting-started/app/
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$
```

Step 3 - Create file : **Dockerfile** and locate localhost, then enter :

sudo docker build -t getting-started .

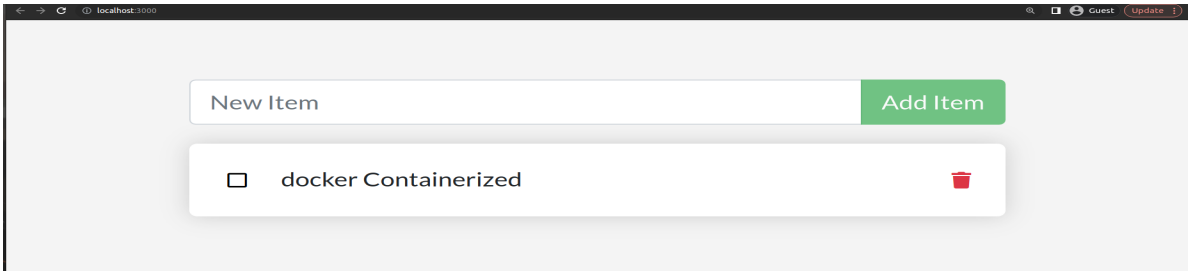
Type required password of your system



Step 4 - Getting-started : **sudo docker run -dp 3000:3000 getting-started**

```
computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$ sudo docker run -dp 3000:3000 getting-started
33762
computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$
```

You can see that containerized docker created successfully on your localhost



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**
COURSE CODE	**CSL603**
PRACTICAL NO.	**10**
DOP	
DOS	

Experiment No. : 10

Aim : To learn pull based Software Management and Provisioning Tools Using puppet.

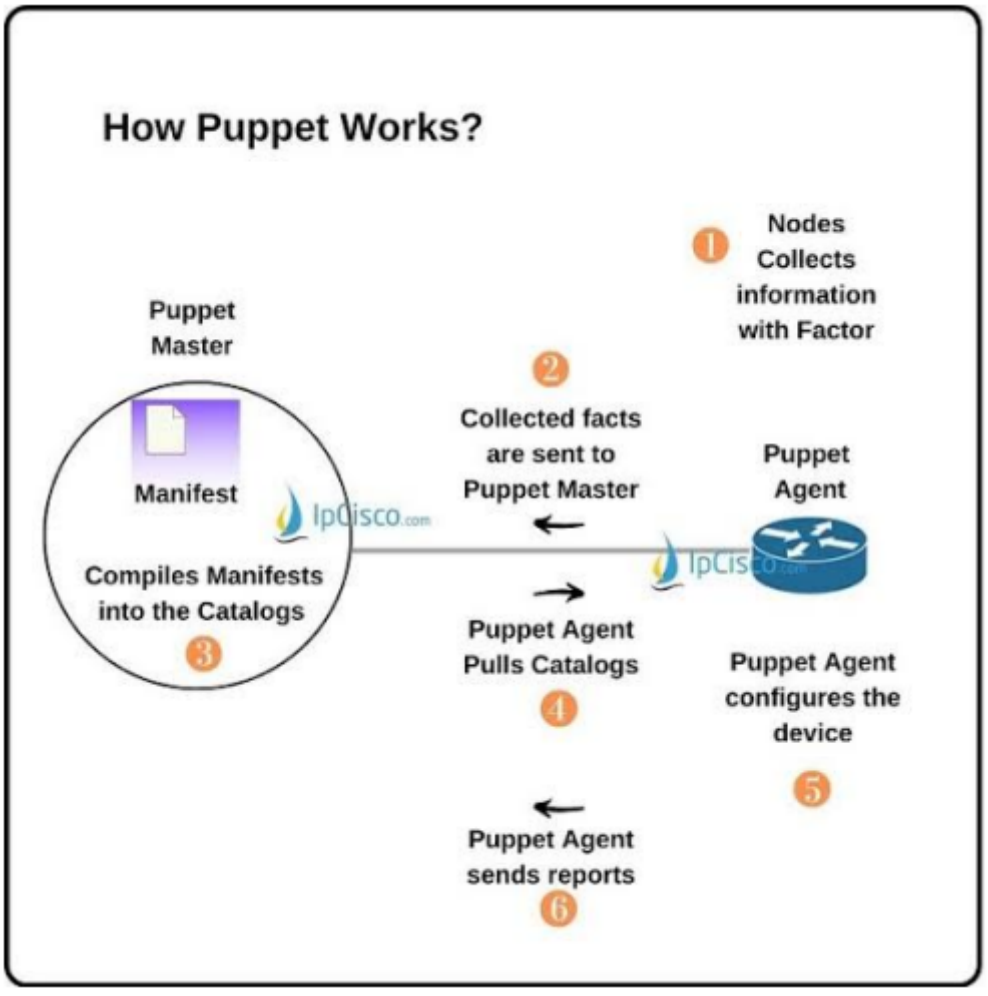
Theory : Puppet

In software engineering, a puppet is a tool used for configuration management. Puppet allows you to automate the management of your infrastructure and the configuration of your software. With Puppet, you can define the desired state of your infrastructure and software and then use Puppet to ensure that your systems remain in that state.

☐ **How does a puppet work?**

Puppet is a popular open-source configuration management tool that allows IT administrators and DevOps engineers to automate the management of infrastructure configurations. Here's an overview of how Puppet works:

- Agent and Master architecture: Puppet uses an agent and master architecture. The Puppet agent runs on each node in your infrastructure, while the Puppet master is a central server that stores the configuration data and manages the agents.



- Declarative language: Puppet uses a declarative language to define the desired state of configurations. This means that you define the end state you want the system to be in, and Puppet figures out how to get there.
- Resources: Puppet manages configurations using resources, which represent the various components of your infrastructure. Resources can include files, users, services, and packages.
- Modules: Puppet organizes resources into modules, which are collections of related configurations. Modules can be shared and reused across different systems and environments.
- Manifests: Puppet uses manifests to define the configurations for each node. Manifests are written in Puppet's declarative language and specify the desired state of each resource.
- Catalogs: Once the manifest is defined, the Puppet master compiles a catalog for each node. The catalog is a list of resources and their desired states, and it is sent to the Puppet agent on the node.
- Agent execution: The Puppet agent on the node receives the catalog from the Puppet master and compares it to the current state of the system. If there are any differences, the agent applies the necessary changes to bring the system into the desired state.
- Reporting: Puppet provides reporting and auditing capabilities, allowing you to track changes and monitor the state of your infrastructure configurations.

☐ **What is configuration management?** A comprehensive guide

Configuration management is the process of managing the configuration of a system, software, or hardware throughout its lifecycle. It involves identifying, documenting, and controlling changes to the configuration items of a system or product, including hardware, software, documentation, and processes.Here is a more detailed explanation of each of these processes:

- Configuration identification: This process involves identifying the configuration items (CIs) of a system or product, such as hardware components, software modules, documentation, and processes. Each CI is assigned a unique identifier and is tracked throughout the development and maintenance lifecycle.
- Configuration control: This process involves controlling changes to the CIs of a system or product. It includes the establishment of a change control board (CCB) or change advisory board (CAB) to evaluate proposed changes, prioritise them, and approve or reject them based on their impact on the system or product.
- Configuration status accounting: This process involves tracking the current status of each CI throughout its lifecycle. It includes recording changes made to each CI, their current status, and the status of related CIs.
- Configuration auditing: This process involves conducting periodic reviews of the configuration management processes and activities to ensure they are being followed correctly and are effective in achieving the desired outcomes.

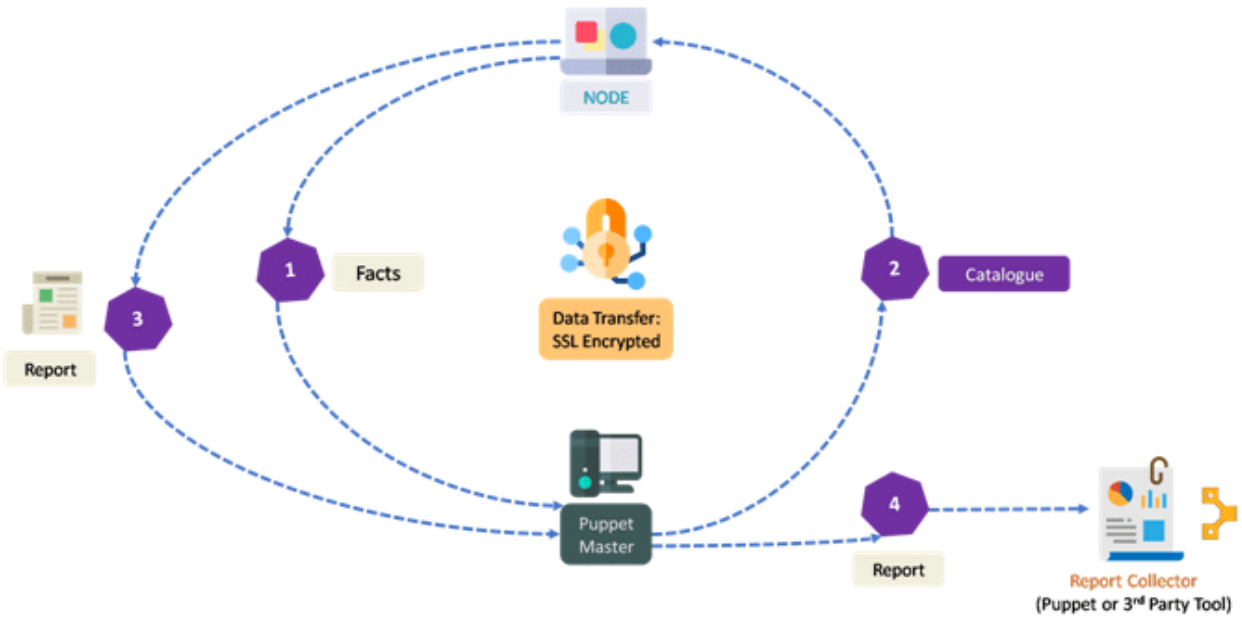
☐ **Learning pull-based software management and provisioning tools using Puppet involves understanding the following concepts and steps:**

- Puppet Architecture: You need to understand the architecture of Puppet, which is based on a master-slave architecture. The Puppet master is responsible for storing the configuration data, and the Puppet slaves (or agents) are responsible for applying the configurations.

- Modules: You need to understand Puppet modules, which are self-contained units of code that can be used to manage specific aspects of your infrastructure, such as installing software, configuring users, or managing files.
- Resources: You need to understand Puppet resources, which are the basic building blocks of a Puppet configuration. Resources can be used to manage files, packages, users, services, and other aspects of your infrastructure.
- Manifests: You need to understand Puppet manifests, which are files that contain Puppet code. Manifests are used to define the desired state of your infrastructure and software.
- Catalogs: You need to understand Puppet catalogs, which are the compiled configurations that Puppet applies to the Puppet agents. Catalogs are created by compiling the manifests, modules, and other configuration data.
- Pull-based Configuration Management: Puppet is a pull-based configuration management tool, which means that the Puppet agents periodically pull the latest configuration data from the Puppet master server. You need to understand how this process works and how to configure it.
- Provisioning Tools: Puppet can be used as a provisioning tool to automate the process of provisioning new infrastructure components, such as servers or virtual machines. You need to understand how to use Puppet to provision new infrastructure and how to integrate Puppet with other provisioning tools.

To learn pull-based software management and provisioning tools using Puppet, you can start by reading the official Puppet documentation and tutorials. You can also join Puppet user groups and forums to learn from other Puppet users and ask for help when you need it. Additionally, you can attend Puppet training courses or workshops to get hands-on experience with Puppet and learn best practices for using it in real-world scenarios.

☐ Here is a **simplified diagram** of the pull-based software management and provisioning process using Puppet:



☐ **Advantages of pull-based software management and provisioning tools using Puppet:**

- Consistency: Pull-based configuration management tools like Puppet provide a consistent way of managing infrastructure and software configurations across all systems, which reduces errors and increases efficiency.
- Flexibility: Puppet provides a flexible and modular approach to configuration management, which allows you to manage a wide range of infrastructure components and software applications.
- Automation: Puppet automates the process of configuration management and provisioning, which saves time and reduces manual errors.
- Version Control: Puppet uses version control for configurations, which allows you to easily roll back to previous versions and track changes over time.
- Centralized Management: Puppet provides centralized management of infrastructure and software configurations, which makes it easier to manage large and complex environments.

☐ **Disadvantages of pull-based software management and provisioning tools using Puppet:**

- Learning Curve: Puppet has a steep learning curve, which can make it challenging for new users to get started with the tool.
- Complexity: Puppets can be complex to configure and manage, especially for large and complex environments.
- Resource Intensive: Puppet can be resource-intensive, especially when managing large numbers of systems or when running complex configurations.
- Dependency on Master Server: Puppet requires a master server to manage the configuration data, which can create a single point of failure and impact availability.
- Limited Support for Windows: Puppet has limited support for Windows environments, which can make it challenging to manage Windows-based systems using Puppet.

Conclusion : Overall, Puppet is a powerful tool for managing infrastructure and software configurations in a consistent and repeatable way, which helps to reduce errors and increase efficiency in software engineering.
So we learned pull based Software Management and Provisioning Tools Using puppet.