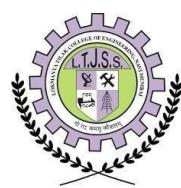




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)**

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

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>01</b>
<b>DOP</b>	<b>18/01/2023</b>
<b>DOS</b>	



## 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.

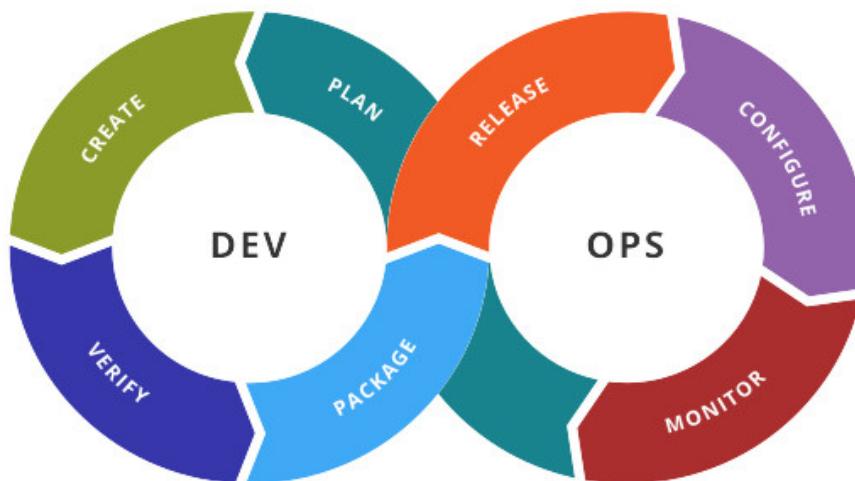
Automation of the software development lifecycle

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.

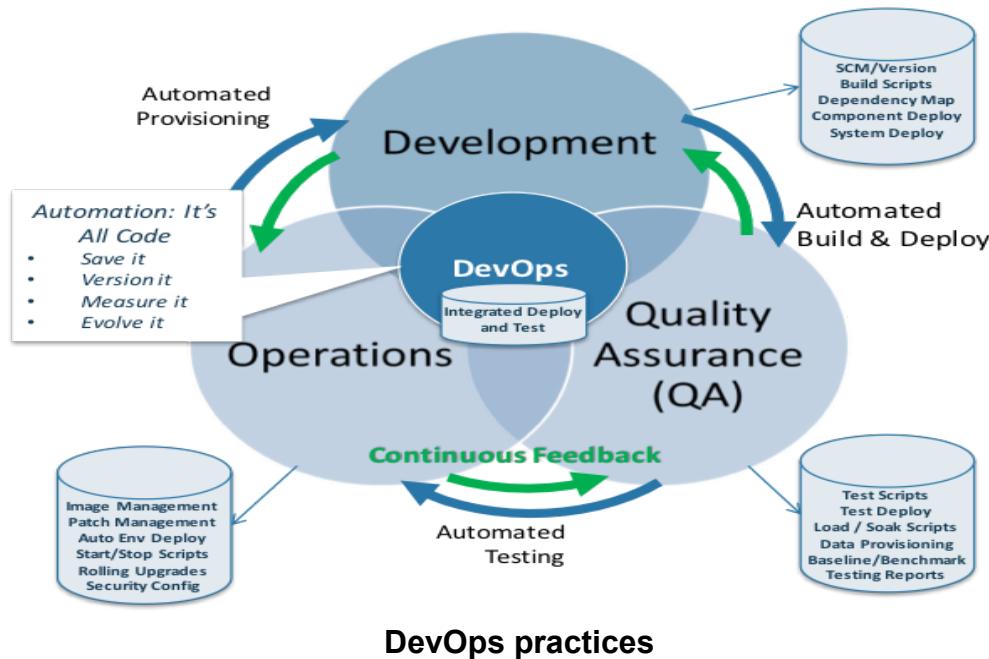


## DevOps practices

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

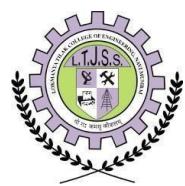
#### What Is a DevOps Engineer?

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.

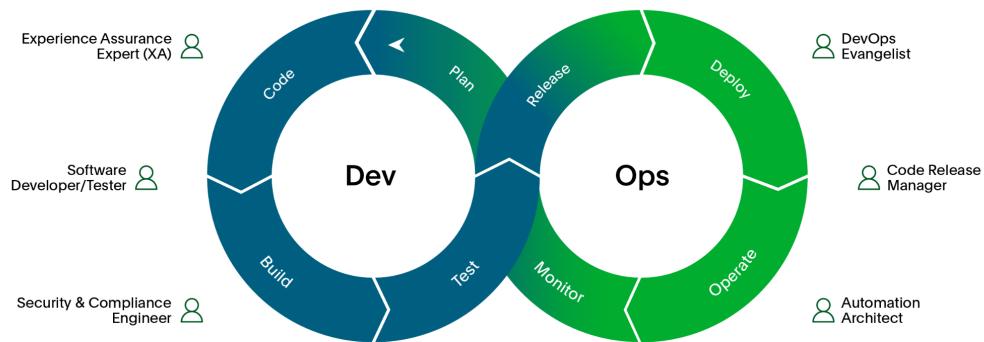
#### DevOps Engineer Job Description: Roles and Responsibilities

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.



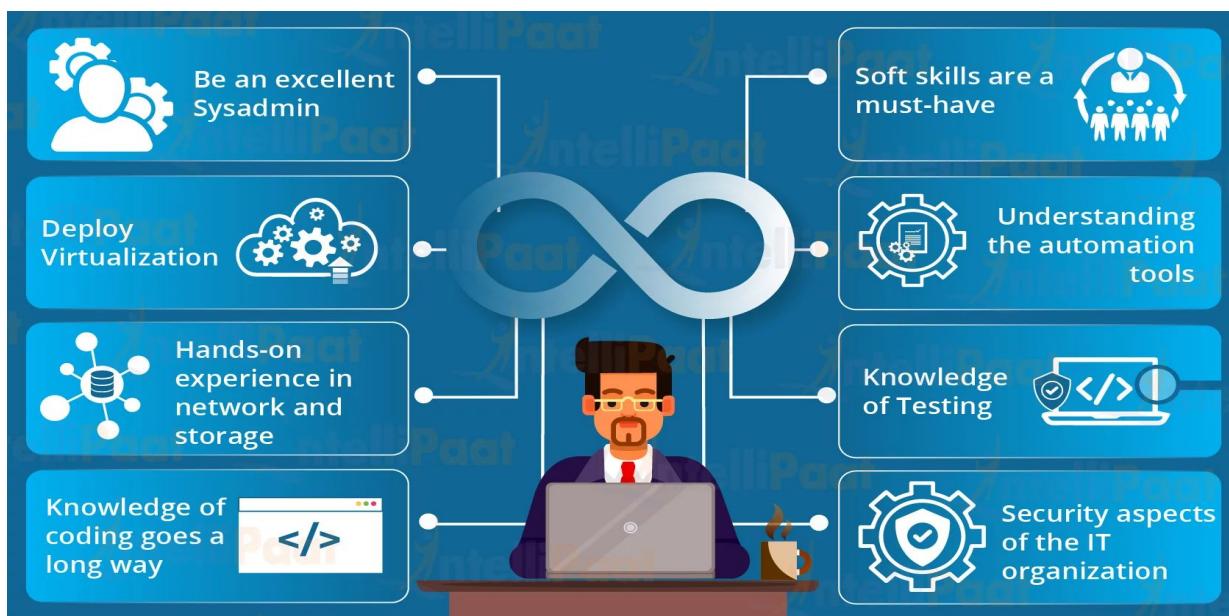
## 6 essential DevOps roles



### 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**

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>02</b>
<b>DOP</b>	<b>25/01/2023</b>
<b>DOS</b>	



## 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 libbz-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

Welcome to GitHub!  
Let's begin the adventure

**Enter your email**  
✓ prathu3322@gmail.com

**Create a password**  
✓ .....\*

**Enter a username**  
✓ prathameshpc777

**Would you like to receive product updates and announcements via email?**  
Type "y" for yes or "n" for no  
✓ y

**Verify your account**

✓

[Create account](#)

**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**

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>03</b>
<b>DOP</b>	<b>01/02/2023</b>
<b>DOS</b>	



## Create new repository on github :

The screenshot shows the GitHub interface for creating a new repository. At the top, it says "prathameshpc777 / prathameshpc777 Public". Below that, there are tabs for "Code", "Issues", "Pull requests", "Actions", "Projects", "Wiki", "Security", "Insights", and "Settings". The "Code" tab is selected. A large central box contains instructions for "Quick setup — if you've done this kind of thing before" and "...or create a new repository on the command line". It includes a code block:

```
echo "# prathameshpc777" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/prathameshpc777/prathameshpc777.git
git push -u origin main
```

Below that, another section says "...or push an existing repository from the command line" with a code block:

```
git remote add origin https://github.com/prathameshpc777/prathameshpc777.git
git branch -M main
git push -u origin main
```

Finally, "...or import code from another repository" with a "Import code" button.

## 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/prathameshpc777/prathameshpc777.git>

**Step 6** : git remote add origin <https://github.com/prathameshpc777/prathameshpc777.git>

**Step 7** : git status

**Step 8** : git add .

**Step 9** : git status

**Step 10** : git commit -m "hello\_prathamesh"

**Step 11** : git config --global user.email "[prathu3322@gmail.com](mailto:prathu3322@gmail.com)"

**Step 12** : git config --global user.name "prathameshpc777"

**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/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git --version
git version 2.34.1
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ touch exp3.txt
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ touch exp3a.txt
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ 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>
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git init
Reinitialized existing Git repository in /home/computer/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777/.git/
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git remote add origin https://github.com/prathameshpc777/prathameshpc777.git
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git remote add origin https://github.com/prathameshpc777/prathameshpc777.git
error: remote origin already exists.
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ 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/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git add .
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git status
On branch master

No commits yet
```

```
computer@computer-ThinkCentre: ~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ 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/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git commit -m "hello_prathamesh"
[master (root-commit) d97ad7f] hello_prathamesh
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 exp3.txt
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git config --global user.email "prathu332@gmail.com"
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git config --global user.name "prathameshpc777"
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git push -u origin master
Username for 'https://github.com': prathameshpc777,
Password for 'https://prathameshpc777.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/prathameshpc777/prathameshpc777.git'
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$ git push -u origin master
Username for 'https://github.com': prathameshpc777
Password for 'https://prathameshpc777.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), 223 bytes | 223.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/prathameshpc777/prathameshpc777.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
(base) computer@computer-ThinkCentre:~/Documents/CSEAIML/TE/SEM6/AIML11_PRATHAMESH/SEPML/EXP3/prathameshpc777$
```



## Output :

Screenshot of a GitHub repository page for "prathameshpc777/prathameshpc777".

The repository has the following details:

- Owner: PrathameshPC777
- Name: hello\_prathamesh
- Branch: master
- Commits: 1
- Tags: 0
- Last commit: d97ad7f, 1 hour ago
- File list:
  - exp3.txt (hello\_prathamesh, 1 hour ago)
  - exp3a.txt (hello\_prathamesh, 1 hour ago)

Repository description:

prathameshpc777/prathameshpc777 is a special repository.  
Its README.md will appear on your public profile.

Buttons:

- Add a README

Statistics:

- 0 stars
- 1 watching
- 0 forks

Sections:

- About
- Releases
- Packages

Footer:

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About



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

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

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>04</b>
<b>DOP</b>	<b>08/02/2023   22/02/2023</b>
<b>DOS</b>	



## 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.

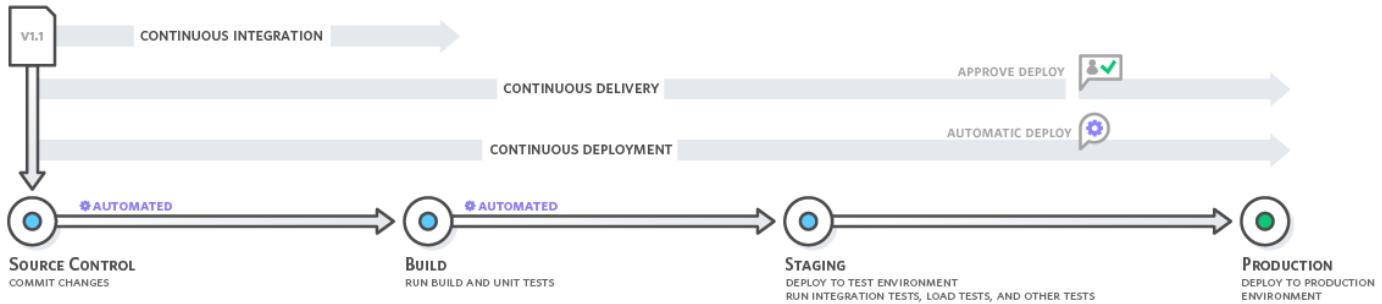
### Uses :

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-1ubuntu22.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu22.04, 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]: cf3dd4aae0594109ab6f67608c281299
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$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.M...
Feb 22 09:50:24 computer-ThinkCentre jenkins[6071]: 2023-02-22 04:20:24.951+0000 [id=68]           INFO    hudson.util.Retriger#start: Performed the action check updates server successfully at the attempt
lines 1-20/20 (END)
[1]- Stopped                  sudo systemctl status jenkins
(base) computer@computer-ThinkCentre:~$ 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.

The screenshot shows a 'Getting Started' page titled 'Unlock Jenkins'. It instructs the user to ensure Jenkins is securely set up by an administrator, who has written a password to a log file and a file on the server. The path is given as /var/lib/jenkins/secrets/initialAdminPassword. A text input field is provided for pasting the password, followed by a 'Continue' button at the bottom right.

2. Obtain the default Jenkins unlock password by opening the terminal and running the following command :

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

```
(base) computer@computer-ThinkCentre:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
cf3dd4aee0594109ab0f67608c281299
```

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

The screenshot shows the 'Getting Started' page with several Jenkins features listed in a grid. The 'Mailer' feature is highlighted with a green checkmark. A tooltip or dropdown menu is open over the 'Mailer' row, displaying a detailed list of Jenkins API endpoints related to Mailer, such as 'commons-lang3 v3.x Jenkins API', 'Caffeine API', 'Script Security API', 'Jackson 2 API', 'Jackson 2 API', 'JSON API', 'Font Awesome API', 'jQuery3 API', 'Display URL API', 'Pipeline: Supporting APIs', 'Config API', 'JUnit', 'Matrix Project', 'Report Disposer', 'Workspace Cleanup', 'Ant', 'Pipeline: SCM Step', 'Pipeline: Groovy', 'Durable Task', 'Pipeline: Nodes and Processes', 'Pipeline: Job', 'Job Activation API', 'Jakarta Mail API', 'Apache HttpComponents Client 4.x API', 'Mailer', 'Pipeline: Basic Steps', 'Grade', and 'required dependency'.



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.

Screenshot from 2023-02-22 10-38-47.png

localhost:8080

Guest (2)

Update

Getting Started

## Create First Admin User

Username  
prathameshpc77

Password  
.....

Confirm password  
.....

Full name  
Prathamesh Shivaji Chikankar

E-mail address  
prathameshchikankar\_aiml\_2020@ltce.in

Jenkins 2.375.3

Skip and continue as admin

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.

Getting Started

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.375.3

Not now

Save and Finish

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Jenkins 2.375.3

AIML11\_PRATHAMESH



11\_SEPML\_EXP4-jenkins x | (1) WhatsApp x | Linux x | Dashboard [Jenkins] x | New Tab x | +

localhost:8080 Guest (2) Update

## Jenkins

Dashboard >

+ New Item

People

Build History

Manage Jenkins

My Views

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

REST API Jenkins 2.375.3

## Configuring Jenkins with maven

11\_SEPML\_EXP4-jenkins x | (2) WhatsApp x | Linux x | Manage Jenkins [Jenkins] x | github - Google Search x | +

localhost:8080/manage Guest (2) Update

## Jenkins

Dashboard > Manage Jenkins

+ New Item

People

Build History

**Manage Jenkins**

My Views

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

It appears that your reverse proxy set up is broken.

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

More Info Dismiss

**System Configuration**

**Configure System** Configure global settings and paths.

**Global Tool Configuration** Configure tools, their locations and automatic installers.

**Manage Plugins** Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

**Manage Nodes and Clouds** Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Security**

**Configure Global Security** Secure Jenkins; define who is allowed to access/use the system.

**Manage Credentials** Configure credentials.

**Configure Credential Providers** Configure the credential providers and types.

**Manage Users** Create/delete/modify users that can log in to this Jenkins.

**Status Information**

**System Information** Displays various environmental information to assist troubleshooting.

**System Log** System log captures output from java.util.logging output related to Jenkins.

**Load Statistics** Check your resource utilization and see if you need more computers for your builds.

**About Jenkins** See the version and license information.



11\_SEPML\_EXP4-jenkins x | (2) WhatsApp x | Linux x | Global Tool Configuration x | github - Google Search x | get java path in ubuntu t x | +

localhost:8080/manage/configureTools/ Guest (2) Update

## Jenkins

Search (CTRL+K) 90% Reset hivaji Chikankar 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

≡ JDK  
Name prathamesh

JAVA\_HOME  
/usr/lib/jvm/java-11-openjdk-amd64

Install automatically ?

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

11\_SEPML\_EXP4-jenkins x | (2) WhatsApp x | Linux x | Global Tool Configuration x | github - Google Search x | get java path in ubuntu t x | +

localhost:8080/manage/configureTools/ Guest (2) Update

## Jenkins

Search (CTRL+K) 90% Reset hivaji Chikankar log out

Dashboard > Manage Jenkins > Global Tool Configuration

Add Ant

#### Maven

Maven installations  
List of Maven installations on this system  
Add Maven

≡ Maven  
Name prathamesh

Install automatically ?

≡ Install from Apache  
Version 3.9.0  
Add Installer ▾

Add Maven

Save Apply

Jenkins 2.375.3

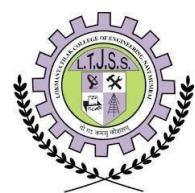
AIML11\_PRATHAMESH



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

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

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>05</b>
<b>DOP</b>	<b>01/03/2023</b>
<b>DOS</b>	



## Building pipeline of jobs and creating pipelining script in jenkins

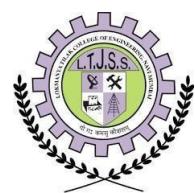
### Step 1 - Log in to your Github account and Jenkins account

The screenshot shows a GitHub repository page for 'prathameshpc777/prathameshpc777'. The repository has 1 branch ('master') and 0 tags. There are 2 commits, both by 'hello\_prathamesh' from 'PrathameshPC77 hello\_prathamesh'. The most recent commit is 'd97ad7f' from last month. A green button labeled 'Add a README' is located on the right side of the repository details.

The screenshot shows the Jenkins dashboard at 'localhost:8080'. On the left, there are sections for 'Build Queue' (empty), 'Build Executor Status' (1 Idle), 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with links to 'Create a job', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds', and a 'Set up a distributed build' section.

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

The screenshot shows the 'New Item' dialog in Jenkins. The 'Project Type' dropdown is set to 'Pipeline'. Other options shown include 'Freestyle project', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. Below the project type, there's a note about creating a new item from an existing one using 'Copy from'. At the bottom, a blue 'OK' button is highlighted.



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

The screenshot shows the Jenkins Pipeline configuration page for a job named 'prathameshpc777'. On the left, under 'General' settings, there is a 'Description' field containing 'hi prathamesh'. Below it are several build options like 'Discard old builds', 'Do not allow concurrent builds', etc. Under 'Build Triggers', none of the options are selected. At the bottom, there are 'Save' and 'Apply' buttons, with a green 'Saved' message displayed.

The right side of the screen shows the 'Pipeline' configuration. It includes an 'Advanced Project Options' section with an 'Advanced...' button, and a 'Pipeline' section with a 'Definition' tab. The 'Script' tab displays a Groovy pipeline script:

```
1 > pipeline {
2   agent any
3   options {
4     //Timeout counter starts AFTER agent is allocated
5     timeout(time: 1, unit: 'SECONDS')
6   }
7   stages {
8     stage('Example') {
9       steps {
10         echo 'Hello World'
11     }
12   }
13 }
14 }
```

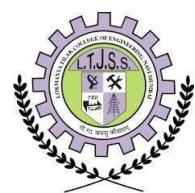
Below the script, there is a checked checkbox for 'Use Groovy Sandbox'. At the bottom right are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins Plugin Manager. The left sidebar has tabs for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', and 'Advanced settings'. A search bar at the top contains the text 'pipe'. The main area shows a list of available plugins:

- Pipeline: Deprecated Groovy Libraries** (609.vd95673f149b\_b)  
Miscellaneous  
Hosting of Pipeline Groovy libraries inside a Jenkins Git server. Deprecated. Use [Pipeline: Groovy Libraries](#) instead. If you see this plugin installed just because you upgraded, you can probably uninstall it now. This plugin should only be used if you have historically pushed libraries to a Git server inside Jenkins.  
This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more.](#)  
Released 3 mo 15 days ago
- Lockable Resources** (1123.v4002ee23c671)  
pipeline Cluster Management Agent Management  
This plugin allows to define external resources (such as printers, phones, computers) that can be locked by builds. If a build requires an external resource which is already locked, it will wait for the resource to be free.  
Released 20 days ago
- Docker Pipeline** (563.vd5d2e5c4007f)  
pipeline DevOps Deployment docker  
Build and use Docker containers from pipelines.  
This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.  
Released 2 mo 29 days ago
- Pipeline SCM API for Blue Ocean** (1.27.1)  
This plugin is a part of BlueOcean Plugin  
Released 1 mo 10 days ago

At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. A red error message says: 'There were errors checking the update sites: UnknownHostException: updates.jenkins.io'



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

Screenshot of the Jenkins Pipeline page for 'prathameshpc777'. The pipeline name is 'Pipeline prathameshpc777'. The stage view shows three stages: #3 (Mar 01 11:52), #2 (Mar 01 11:52), and #1 (Mar 01 11:52). The average stage time is 31ms. A sidebar on the left includes options like Status, Changes, Build Now, Configure, Delete Pipeline, Rename, Pipeline Syntax, Build History, and Full Stage View.

You can check your status as well

Screenshot of the Jenkins Stage Logs page for 'prathameshpc777'. It shows the log entry: 'Print Message – Hello World (self time 5ms)' followed by the output 'Hello World'.

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

Screenshot of the Jenkins Console Output page for build #3. The output shows the pipeline script execution:

```
Started by user Prathamesh Shivaji Chikankar
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/prathameshpc77
[Pipeline]
[Pipeline] timeout
Timeout set to expire in 1 sec
[Pipeline]
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Example)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



## 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-jar - 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-jar 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-jar 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-jar 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:

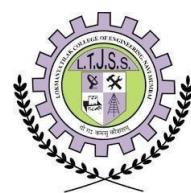
## Step 6 - Open ports for Apache Tomcat Server

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**

```
computer@computer-ThinkCentre: ~
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9-admin all 9.0.58-1ubuntu0.1 [68.8 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libtcnative-1 amd64 1.2.31-1build1 [95.1 kB]
Fetched 12.7 MB in 4s (3,152 kB/s)
Selecting previously unselected package libapr1:amd64.
(Reading database ... 306747 files and directories currently installed.)
Preparing to unpack .../0-libapr1_1.7.0-8ubuntu0.22.04.1_amd64.deb ...
Unpacking libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...
Selecting previously unselected package libeclipse-jdt-core-java.
Preparing to unpack .../1-libeclipse-jdt-core-java_3.27.0+eclipse4.21-1_all.deb ...
Unpacking libeclipse-jdt-core-java (3.27.0+eclipse4.21-1) ...
Selecting previously unselected package libtomcat9-java.
Preparing to unpack .../2-libtomcat9-java_9.0.58-1ubuntu0.1_all.deb ...
Unpacking libtomcat9-java (9.0.58-1ubuntu0.1) ...
Selecting previously unselected package tomcat9-common.
Preparing to unpack .../3-tomcat9-common_9.0.58-1ubuntu0.1_all.deb ...
Unpacking tomcat9-common (9.0.58-1ubuntu0.1) ...
Selecting previously unselected package tomcat9.
Preparing to unpack .../4-tomcat9_9.0.58-1ubuntu0.1_all.deb ...
Unpacking tomcat9 (9.0.58-1ubuntu0.1) ...
Selecting previously unselected package tomcat9-admin.
Preparing to unpack .../5-tomcat9-admin_9.0.58-1ubuntu0.1_all.deb ...
Unpacking tomcat9-admin (9.0.58-1ubuntu0.1) ...
Selecting previously unselected package libtcnative-1:amd64.
Preparing to unpack .../6-libtcnative-1_1.2.31-1build1_amd64.deb ...
Unpacking libtcnative-1:amd64 (1.2.31-1build1) ...
Setting up libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...
Setting up libeclipse-jdt-core-java (3.27.0+eclipse4.21-1) ...
Setting up libtomcat9-java (9.0.58-1ubuntu0.1) ...
Setting up tomcat9-common (9.0.58-1ubuntu0.1) ...
Setting up libtcnative-1:amd64 (1.2.31-1build1) ...
Setting up tomcat9-admin (9.0.58-1ubuntu0.1) ...
Setting up tomcat9 (9.0.58-1ubuntu0.1) ...
Creating group tomcat with gid 999.
Creating user tomcat (Apache Tomcat) with uid 999 and gid 999.

Creating config file /etc/tomcat9/tomcat-users.xml with new version
Creating config file /etc/tomcat9/web.xml with new version
Creating config file /etc/tomcat9/server.xml with new version
Creating config file /etc/tomcat9/logging.properties with new version
Creating config file /etc/tomcat9/context.xml with new version
Creating config file /etc/tomcat9/catalina.properties with new version
Creating config file /etc/tomcat9/jaspic-providers.xml with new version
Creating config file /etc/logrotate.d/tomcat9 with new version

Creating config file /etc/default/tomcat9 with new version
Created symlink /etc/systemd/system/multi-user.target.wants/tomcat9.service → /lib/systemd/system/tomcat9.service.
Processing triggers for rsyslog (8.2112.0-2ubuntu2.2) ...
Processing triggers for libc-bin (2.35-0ubuntu3) ...
computer@computer-ThinkCentre:~$ ss -ltn
State      Recv-Q      Send-Q      Local Address:Port          Peer Address:Port      Process
LISTEN      0            70      127.0.0.1:33060      0.0.0.0:*
LISTEN      0            128      127.0.0.1:631      0.0.0.0:*
LISTEN      0            151      127.0.0.1:3306      0.0.0.0:*
LISTEN      0            32       192.168.122.1:53      0.0.0.0:*
LISTEN      0            4096     127.0.0.53%lo:53      0.0.0.0:*
LISTEN      0            128      [:1]:631          *:8080      [:]:*
LISTEN      0            50       *:8080          *:*
computer@computer-ThinkCentre:~$ sudo ufw allow from any to any port 8080 proto tcp
Rule added
Rule added (v6)
```



## 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>



## It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat9/webapps/ROOT/index.html`

Tomcat veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat9` and `CATALINA_BASE` in `/var/lib/tomcat9`, following the rules from `/usr/share/doc/tomcat9-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

**tomcat9-docs**: This package installs a web application that allows to browse the Tomcat 9 documentation locally. Once installed, you can access it by clicking [here](#).

**tomcat9-examples**: This package installs a web application that allows to access the Tomcat 9 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).

**tomcat9-admin**: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the [manager webapp](#) and the [host-manager webapp](#).

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat9/tomcat-users.xml`.



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

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

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>06</b>
<b>DOP</b>	
<b>DOS</b>	

## Experiment No. : 06

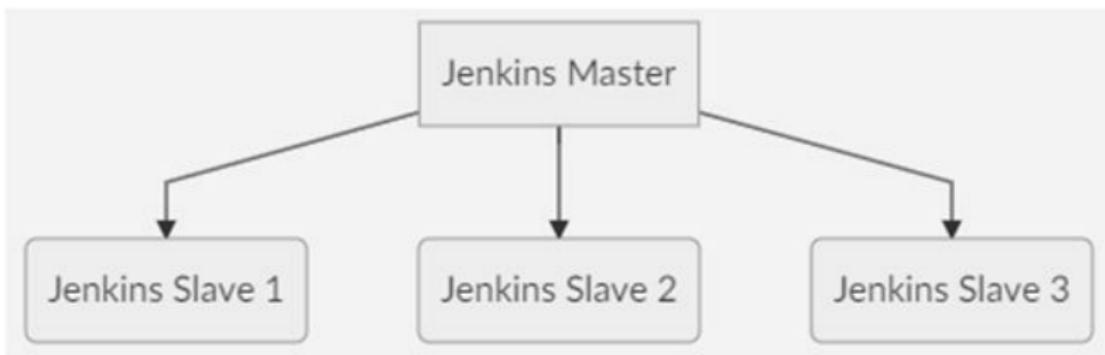
**Aim :** To understand Jenkins Master-Slave Architecture and scale your Jenkins Standalone implementation by implementing slave nodes.

### 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

Dashboard > Manage Jenkins > Nodes > New node

#### New node

Node name

prathameshsudham

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Copy Existing Node

Create

REST API Jenkins 2.387.1

### 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.

Dashboard > Manage Jenkins > Nodes >

Configure Clouds

Node Monitoring

#### Manage nodes and clouds

+ New Node



Build Queue

No builds in the queue.

Build Executor Status

Built-In Node

1 Idle

2 Idle

prathameshsudham

1 Idle

2 Idle

Rahul

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Mac OS X (x86_64)	In sync	78.85 GB	0 B	78.85 GB	0ms
	prathameshsudham	Mac OS X (aarch64)	In sync	78.85 GB	0 B	78.85 GB	104ms
	Rahul		N/A	N/A	N/A	N/A	N/A

Data obtained

4 min 36 sec

REST API Jenkins 2.387.1



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

Dashboard > Manage Jenkins > Nodes >

Name ?  
prathameshsudham

Description ?  
[Plain text] Preview

Number of executors ?  
2

Remote root directory ?  
/Users/sudhamprathamesh/jenkins

Usage ?  
Use this node as much as possible

Launch method ?  
Launch agent by connecting it to the controller

Disable WorkDir ?  
Custom WorkDir path ?  
/Users/sudhamprathamesh/jenkins

Internal data directory ?  
remoting

Fail if workspace is missing ?  
 Use WebSocket ?

Advanced ▾

Save

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

Dashboard > Manage Jenkins > Nodes > prathameshsudham

Status Agent prathameshsudham Mark this node temporarily offline ⓘ

Delete Agent Configure Add description

Build History Projects tied to prathameshsudham

Load Statistics None

Script Console

Log

System information

Disconnect

Build Executor Status ▾

1 Idle  
2 Idle

REST API Jenkins 2.387.1



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

The screenshot shows the Jenkins interface for managing nodes. A specific node named 'prathameshsudham' is selected. In the left sidebar, there are several tabs: Status, Delete Agent, Configure, Build History, Load Statistics, Script Console, Log (which is currently selected), System Information, and Disconnect. Below the sidebar, a section titled 'Build Executor Status' shows two idle executors. At the bottom right of the main content area, it says 'REST API Jenkins 2.387.1'.

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

```
(base) computer@computer-ThinkCentre: ~ curl -s0 http://127.0.0.1:8080/jnlpJars/agent.jar
(base) computer@computer-ThinkCentre: ~ java -jar agent.jar -jnlpUrl http://127.0.0.1:8080/manage/computer/Chinmay/jenkins-agent.jnlp -secret 4e25ddcab0832f97da1b8d499aeef9b950d3712fdcab47175f318726d51a
edf -workDir "/home/computer"
Mar 28, 2023 1:45:55 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/computer/remoting as a remoting work directory
Mar 28, 2023 1:45:55 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/computer/remoting
Mar 28, 2023 1:45:55 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: Chinmay
Mar 28, 2023 1:45:55 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3077.vd69cf116da_0f
Mar 28, 2023 1:45:55 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/computer/remoting as a remoting work directory
Mar 28, 2023 1:45:55 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: WebSocket connection open
Mar 28, 2023 1:45:55 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

### 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 :

Screenshot of Jenkins Configuration page for 'Test\_job' under 'Configuration' tab.

**Configure**

**General**

- Discard old builds
- GitHub project
- This project is parameterised
- Throttle builds
- Execute concurrent builds if necessary
- Restrict where this project can be run

Label Expression: prathameshsudham

Label Karan matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

**Source Code Management**

**Save** **Apply**

Screenshot of Jenkins Configuration page for 'Test\_job' under 'Configuration' tab.

**Configure**

**General**

**Source Code Management**

**Build Triggers**

**Build Environment**

**Build Steps**

**Post-build Actions**

**Build Steps**

Inspect build log for published build scans

Terminate a build if it's stuck

With Ant

**Execute shell**

Command:

```
echo "Hello, Welcome to scripted pipeline"
```

**Advanced...**

**Add build step ▾**

**Save** **Apply**

Screenshot of Jenkins Build History for 'Test\_job' showing build #2.

**Status**

**Changes**

**Console Output**

**View as plain text**

**Edit Build Information**

**Delete build '#2'**

**Previous Build**

### Console Output

Started by user Sudham Prathamesh  
Running as SYSTEM  
Building remotely on prathameshsudham kspace /home/computer/Downloads/workspace/Test\_job  
[Test\_job] \$ /bin/sh -xe /tmp/jenkins16393615253432665159.sh  
+ echo Hello, Welcome to scripted pipeline  
Hello, Welcome to scripted pipeline  
Finished: SUCCESS

**REST API** **Jenkins 2.375.3**



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

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

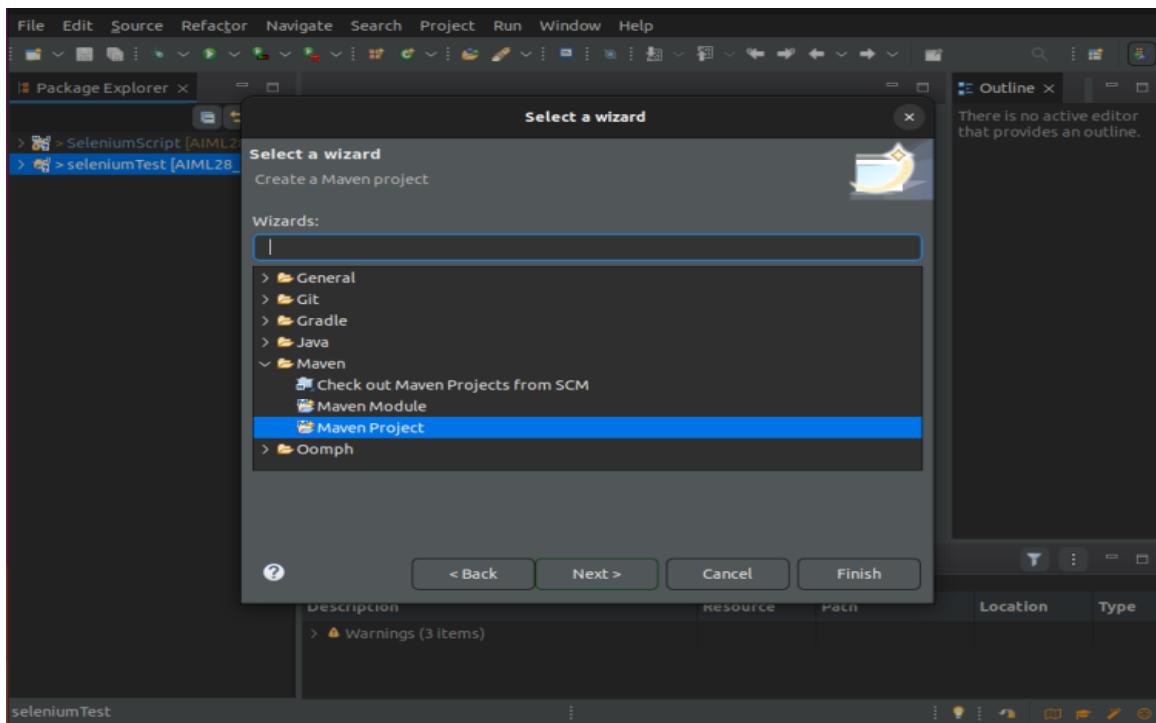
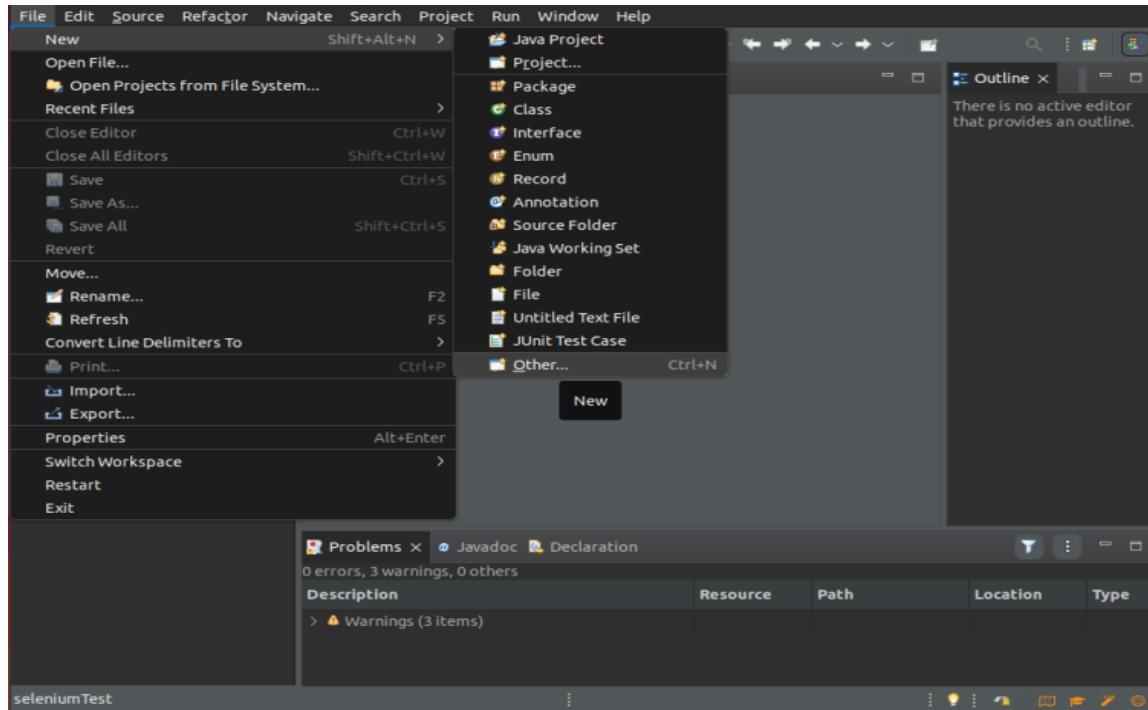
<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>07</b>
<b>DOP</b>	
<b>DOS</b>	



## Selenium Tests in Jenkins Using Maven

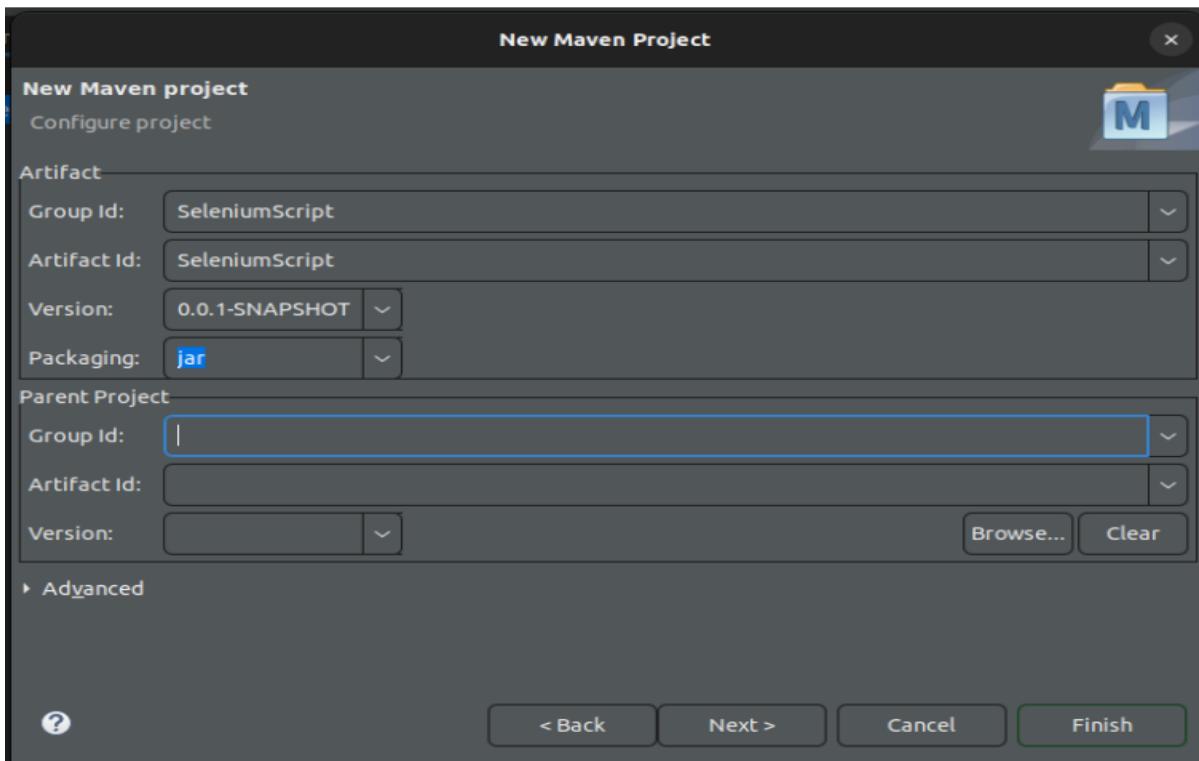
### Steps :

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

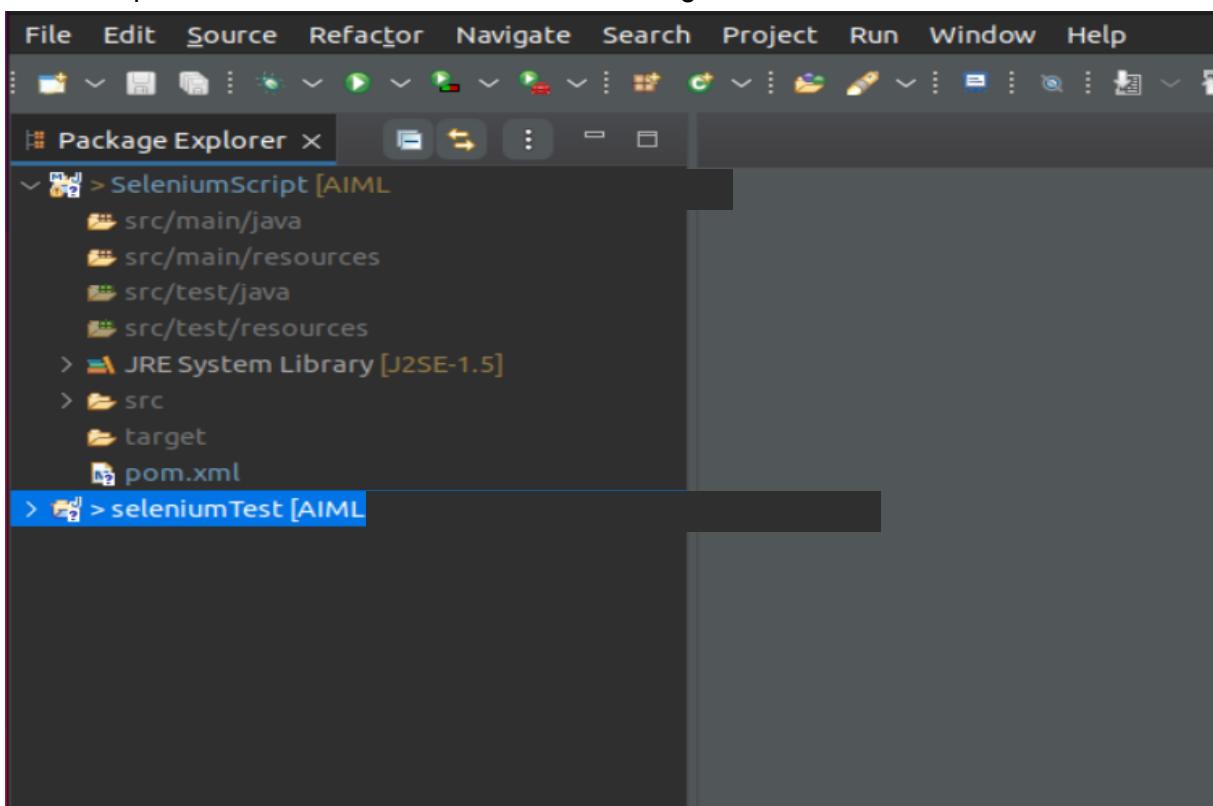




2. On the New Maven Project dialog select the Create a simple project and click Next.
3. Enter SeleniumScript in Group Id: and Artifact Id: and click finish.

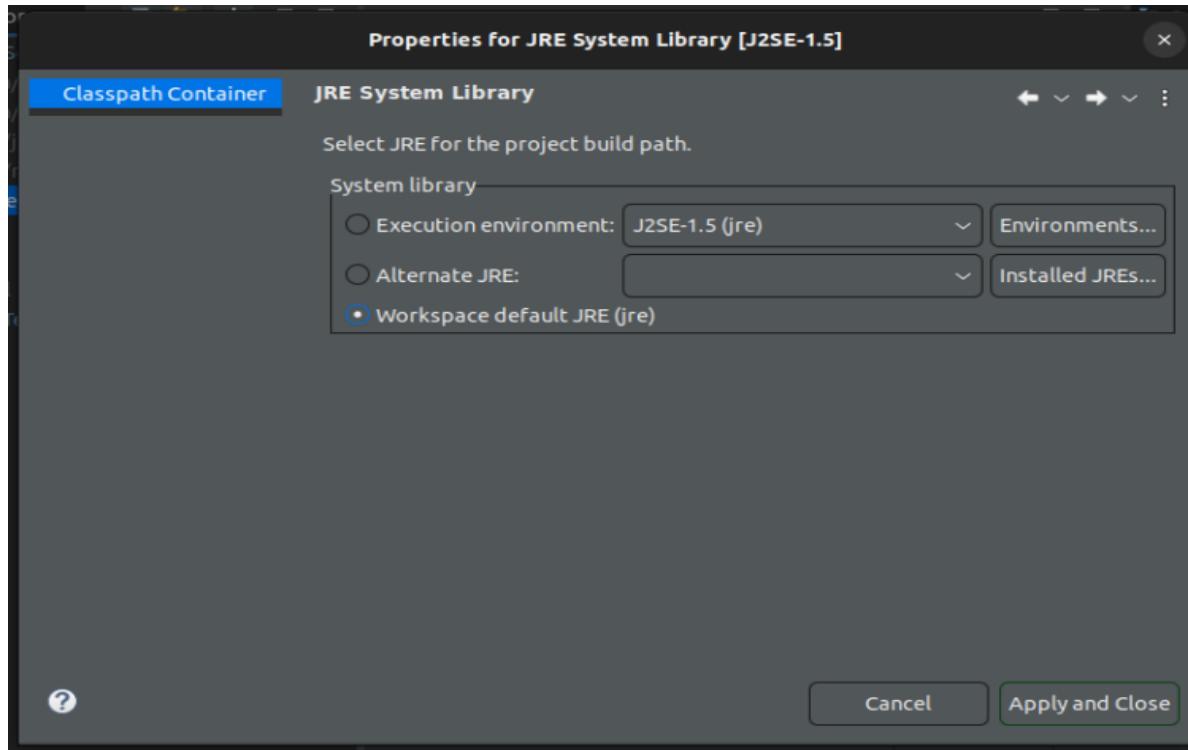


4. Eclipse will create WebDriverTest with following structure:





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>
```



The screenshot shows the Eclipse IDE interface. In the center, the 'SeleniumScript/pom.xml' file is open, displaying XML code for a Maven project. The 'dependencies' section is highlighted. To the right, the 'Out...' view shows a tree structure of the project's dependencies, with 'dependencies' selected. The bottom status bar indicates 'Writable' and 'Insert' modes.

The screenshot shows the Eclipse Marketplace interface. A search bar at the top contains 'TestNG'. Below it, two plugins are listed: 'TestNG for Eclipse' and 'MoreUnit 3.3.0'. Both have 'Installed' status indicators. The 'Marketplaces' tab is visible at the bottom.

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:

New TestNG class

Specify additional information about the test class.

Source folder: /SeleniumScript/src/test/java

Package name: qautomation

Class name: TestScript

Annotations

@BeforeMethod  @AfterMethod  @DataProvider  
 @BeforeClass  @AfterClass  
 @BeforeTest  @AfterTest  
 @BeforeSuite  @AfterSuite

XML suite file:

< Back  Cancel

Select a wizard

Select a wizard

Wizards:

type filter text

> General  
> Git  
> Gradle  
> Java  
> Maven  
> Oomph  
TestNG  
  TestNG class

< Back  Cancel



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 :

The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' view displays the project structure for 'SeleniumScript [AIML]'. It includes a 'src/main/java' folder containing 'src/test/java/qautomation/TestScript.java', which contains three methods: 'afterTest()', 'beforeTest()', and 'f()'. Other visible files include 'pom.xml' and 'seleniumTest [AIM]'. On the right, the 'SeleniumScript/pom.xml' tab is active, showing the Maven configuration. Below it, the 'TestScript.java' tab is active, displaying the Java code for the TestScript class.

```
1 package qautomation;
2
3 import org.testng.annotations.Test;
4
5
6     Run All
7 public class TestScript {
8     @Test
9         Run | Debug
10    public void f() {
11    }
12    @BeforeTest
13    public void beforeTest() {
14    }
15    @AfterTest
16    public void afterTest() {
17    }
18
19 }
```

```
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.getProperty("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_IGNORING_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.com/");
    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**

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>08</b>
<b>DOP</b>	
<b>DOS</b>	

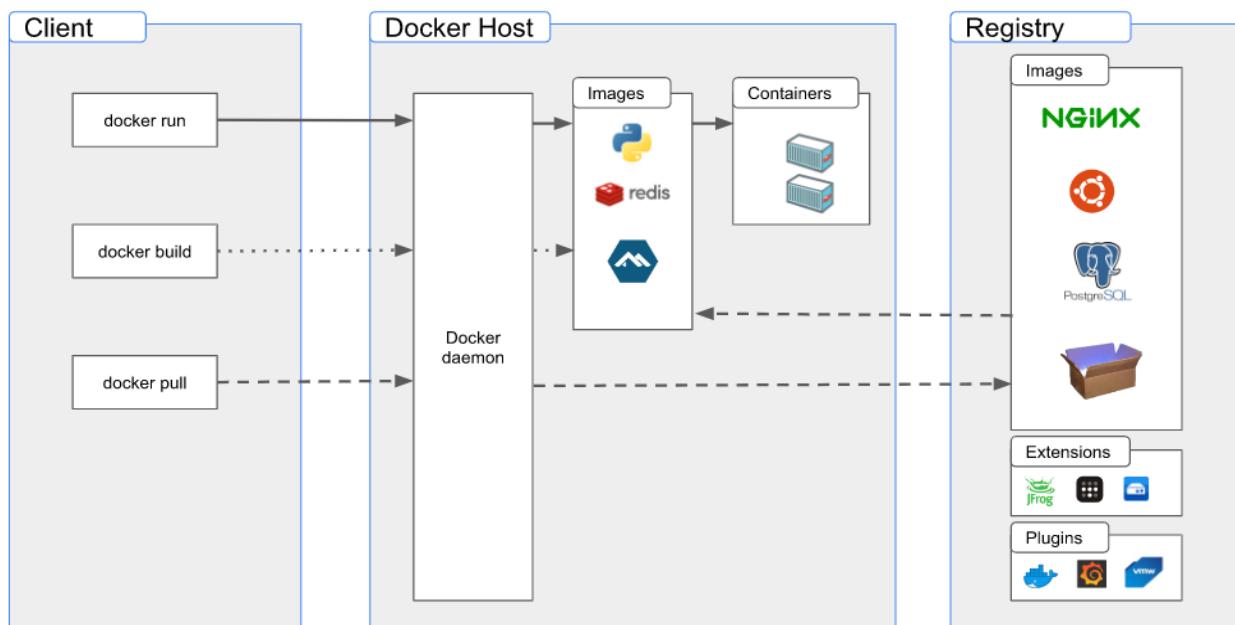
## Experiment No. : 08

**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 -**

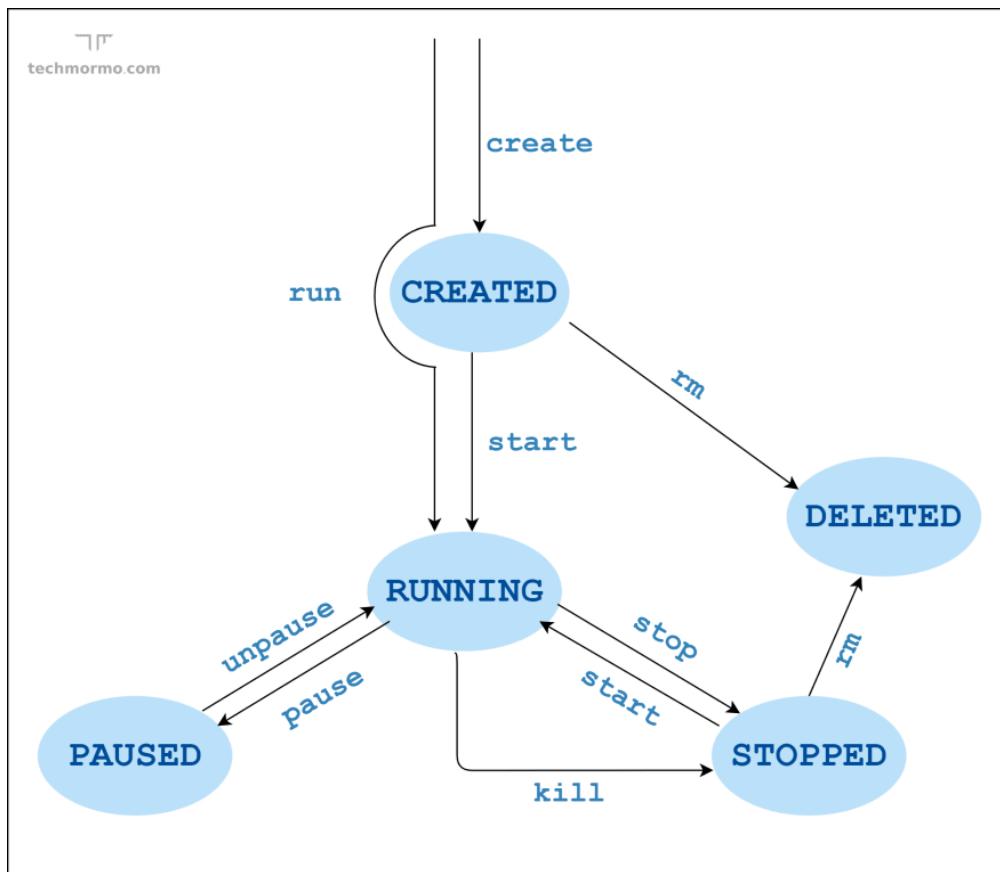


Docker architecture consists of the following components:

- **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/>**

The screenshot shows the Docker Hub homepage. At the top, there's a navigation bar with links for Products, Developers, Pricing, Blog, About Us, and Partners. On the right, there are 'Sign In' and 'Get Started' buttons. Below the navigation, a large banner features the text 'Play with Docker' and 'Hands-on Docker Tutorials for Developers'. To the right of the text is a cartoon illustration of a blue whale-like character interacting with a computer monitor displaying a Docker logo. Below the banner, a call-to-action reads 'Don't let app complexity get in the way of opportunity' followed by the subtext 'Learn Docker today and join the millions of developers who use Docker Desktop and Docker Hub to simplify building and sharing world-changing apps'. At the bottom of the page, there's a cookie consent banner with options for 'Cookies Settings', 'Reject All', and 'Accept All Cookies'.

**Step 2 - Create your docker account**

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

This screenshot shows two side-by-side browser windows. The left window displays the 'Create a Docker Account' form, where the username 'prathameshpc77' and email 'prathahu332@gmail.com' are entered. A checkbox for receiving updates is unchecked, while the terms and privacy policies are checked. The 'I'm not a robot' reCAPTCHA box is checked. The 'Sign Up' button is visible at the bottom. The right window shows the 'Choose a Plan' section, listing four plans: Personal (\$0), Pro (\$5/month), Team (\$9/month), and Business (\$24/month). The Personal plan is selected, and the 'Continue with Free' button is highlighted.

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

This screenshot shows a browser with multiple tabs open. The active tab is 'hub.docker.com/verify-email'. The Docker Hub header is visible, along with the 'Explore', 'Repositories', 'Organizations', 'Help', and 'Upgrade' buttons. The main content area displays a message stating 'Your email has been verified!'.



## Step 5 - Creating first repository

Click on **Create a Repository**

Welcome to Docker  
Download the desktop application  
Download for Windows  
Also available for Mac and Linux

**Create a Repository**  
Push container images to a repository on Docker Hub.

**Docker Hub Basics**  
Watch the guide on how to create and push your first image into a Docker Hub repository.

**Language-Specific Guides**  
Learn how to containerize language-specific applications using Docker.

Access the world's largest library of container images

nginx, mongoDB, alpine, node, redis, busybox, ubuntu, python, postgres, httpd

Name it <your-username>/prathamesh  
Set the visibility to private

Wasm is a fast, light alternative to Linux containers – try it out today with the Docker+Wasm Beta.

**Create repository**

prathameshpc77 prathamesh

**Visibility**

Using 0 of 1 private repositories. [Get more](#)

Public Appears in Docker Hub search results

Private Only visible to you

**Pro tip**  
You can push a new image to this repository using the CLI  
`docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname`

Make sure to change `tagname` with your desired image repository tag.

[Cancel](#) [Create](#)

Click on **Create**

prathameshpc77 / prathamesh

Description

Last pushed: 23 minutes ago



Now next,

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

The screenshot shows the Docker Explore tab with a list of 10,000 available results. The results are displayed in a grid format, each row containing an image icon, the image name, its status (e.g., DOCKER OFFICIAL IMAGE), size, rating, and last update. To the right of each row is a pull count chart for the previous week and a 'Learn more' link.

Image Name	Status	Last Update	Pulls (Last Week)
alpine	DOCKER OFFICIAL IMAGE	Updated 10 days ago	9,268,526
nginx	DOCKER OFFICIAL IMAGE	Updated 9 days ago	32,868,043
busybox	DOCKER OFFICIAL IMAGE	Updated 6 days ago	12,698,125
ubuntu	DOCKER OFFICIAL IMAGE	Updated 7 days ago	25,915,974

Here you can click on the docker tab

The screenshot shows the Docker tab with a list of various Docker images. Each row contains an image icon, the image name, its status (e.g., DOCKER OFFICIAL IMAGE), size, rating, and last update. To the right of each row is a pull count chart for the previous week and a 'Learn more' link.

Image Name	Status	Last Update	Pulls (Last Week)
traefik	DOCKER OFFICIAL IMAGE	Updated 5 hours ago	1,222,191
mariadb	DOCKER OFFICIAL IMAGE	Updated 6 days ago	5,158,020
rabbitmq	DOCKER OFFICIAL IMAGE	Updated a day ago	4,007,808
hello-world	DOCKER OFFICIAL IMAGE	Updated 7 days ago	1,607,377
openjdk	DOCKER OFFICIAL IMAGE	Updated 5 hours ago	5,005,679

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

The screenshot shows a Docker session editor interface. On the left, there's a sidebar with a tree view of files in the current directory: /root. Inside /root, there are Dockerfile, dr, index.html, and myimage. The main area is titled "Dockerfile" and contains the following Dockerfile content:

```
1 FROM nginx:latest
2 WORKDIR /usr/share/nginx/html
3 COPY index.html /index.html
```

Below the code editor are two buttons: "Save" and "Reload". To the right of the editor, there's a performance monitoring section showing "CPU" usage at "0.28%".

```
drwxr-xr-x 2 root root 64 Jan 10 10:30 index.html myimage
[incodel] [local] root@192.168.0.28 ~
9 touch Dockerfile
[incodel] [local] root@192.168.0.28 ~
9 docker build -t nginx-with-html -f Dockerfile .
unable to prepare context: unable to evaluate symlinks in Dockerfile path: lstat /root/Dockerfile: no such file or directory
[incodel] [local] root@192.168.0.28 ~
9 docker image build -t nginx-with-html -f Dockerfile .
unable to prepare context: unable to evaluate symlinks in Dockerfile path: lstat /root/Dockerfile: no such file or directory
[incodel] [local] root@192.168.0.28 ~
9 docker image build -t nginx-with-html -f Dockerfile .
9 docker image build context to Docker daemon 12.8kB
Step 1/3 : FROM nginx:latest
--> 080ed0ed8312
Step 2/3 : WORKDIR /usr/share/nginx/html
--> Running in 0e9271213963
Removing intermediate container 0e9271213963
--> 1e0d0db18a605
Step 3/3 : COPY index.html index.html
--> f4ae0c86a372b
Successfully built f4ae0c86a372b
Successfully tagged nginx-with-html:latest
[incodel] [local] root@192.168.0.28 ~
9 docker run -d -p 8080:80 nginx-with-html
12318db6ecd9946132972d9b39b0eac077a8de4cc0a19feb2e4f3f8e7c79
Error: Error response from daemon: driver failed programming external connectivity on endpoint goofy_chaum (5d5fb1db0c9dd8fe6cd753a68acf931d8ec1)
For more details about this error, refer to the logs of the last command.
[incodel] [local] root@192.168.0.28 ~
9 [ ]
```



02:51:09

cgmfd82e\_cgmfdf2e69v000bj8f60

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.28 node1

IP: 192.168.0.28 OPEN PORT: 8081 8080

Memory: 8.70% (348.1MB / 3.906GB)

SSH: ssh ip172-18-0-6-cgmfd82e69v000bj8f40@direct.labs.play-with-docker

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
4121fdf57d99b26b2ad8c5cf357ad1bf43401a552de88d90b9812c85e8868d28
[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

Gmail WhatsApp Docker Playground Welcome to nginx! Welcome to nginx!

Apr 5 10:57 AM

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](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).  
Thank you for using nginx.

02:56:58

cgmfd82e\_cgmfdf2e69v000bj8f60

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.28 node1

IP: 192.168.0.28 OPEN PORT: 8081 8080

Memory: 10.09% (403.7MB / 3.906GB)

SSH: ssh ip172-18-0-6-cgmfd82e69v000bj8f40@direct.labs.play-with-docker

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: sha256:67211c14fa74f070d27cc59d69a7fa9aef2e8eall8ef3babca295a0428a6d21
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
9bd16b05f662: Pull complete
78e32a9ae10c: Pull complete
41a1a1a1a1a1: Pull complete
787571954425: Pull complete
2196b088d320: Pull complete
7110b83b5540: Pull complete
3a87bf11fa6d: Pull complete
63b2fc823dcf: Pull complete
f77b236f8f2c: Pull complete
b5b4ba9fd2ab: Pull complete
4fc4c55511: Pull complete
Digest: sha256:ff3828b105886f496e47eddced1676b2abd1419a40a65ac4ab387balc538a09f
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**

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>09</b>
<b>DOP</b>	
<b>DOS</b>	



## 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
```

```
(base) computer@computer-ThinkCentre:~$ sudo apt-get update  
sudo apt-get install \  
    ca-certificates \  
    curl \  
    gnupg \  
    lsb-release  
Hit:1 https://deb.nodesource.com/node_16.x jammy InRelease  
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease  
Hit:3 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease  
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]  
Get:5 https://deb.debian.org/debian stable InRelease [107 kB]  
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]  
Fetched 356 kB in 45 (79.7 kB/s)  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Reading state information... Done  
lsb-release is already the newest version (11.1.0ubuntu4).  
lsb-release set to manually installed.  
ca-certificates is already the newest version (20210101ubuntu0.22.04.1).  
ca-certificates set to manually installed.  
gnupg is already the newest version (2.2.27-3ubuntu2.1).  
gnupg set to manually installed.  
The following packages were automatically installed and are no longer required:  
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi gyp javascript-common  
libflashrom1 libfftdi1-2 libgstreamer-plugins-bad1.0-0 libjs-events  
libjs-highlight.js libjs-inherits libjs-is-typedarray libjs-psl  
libjs-source-map libjs-sprintf.js libjs-typedarray-to-buffer libnode-dev  
libnode72 libssl-dev libuv1-dev node-abrev node-ansi-regex node-ansi-styles  
node-ansi-styles node-are-we-there-yet node-arrify node-asap node-asynkit  
node-balanced-match node-brace-expansion node-chown node-clean-yaml-object  
node-color-convert node-color-name node-commander node-core-util-is  
node-decompress-response node-delayed-stream node-delegates node-depd  
node-diff node-encoding node-end-of-stream node-err-code  
node-escape-string-regexp node-fancy-log node-foreground-child  
node-fs.realpath node-function-bind node-get-stream node-glob node-growl  
node-has-flag node-has-unicode node-hosted-git-info node-iconv-lite  
node-iferr node-imurmurhash node-indent-string node-inflight node-inherits  
node-int node-ip node-ip-regex node-ts-buffer node-is-plain-obj  
node-ts-typedarray node-isarray node-isexe node-json-parse-better-errors  
node-jsonparse node-kind-of node-lodash-packages node-lowercase-keys  
node-lru-cache node-lru-response node-minimatch node-minimist  
node-minipass node-must-stream node-negotiator node-pipnode node-once  
node-osenv node-p-cancelable node-p-map node-path-is-absolute  
node-process-nextick-args node-promise-inflight node-promise-retry  
node-promzard node-pump node-quick-lru node-read node-readable-stream  
node-resolve node-retry node-safe-buffer node-set-blocking node-signal-exit  
node-slash node-slice-ansi node-source-map node-spxd-correct  
node-spxd-exceptions node-spxd-expression-parse node-spxd-license-ids  
node-sprintf.js node-stealthy-require node-string-decoder  
node-supports-color node-text-table node-time-stamp node-tmatch  
node-typedarray-to-buffer node-universalify node-util-deprecate  
node-validate-npm-package-license node-webidl-conversions node-whatwg-fetch  
node-wrapify node-yallist  
Use 'sudo apt autoremove' to remove them.
```



- **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`

```
(base) computer@computer-ThinkCentre:~$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [13.6 kB]
Hit:3 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
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]
Fetched 398 kB in 4s (106 kB/s)
Reading package lists... Done
(base) computer@computer-ThinkCentre:~$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:3 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:6 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 336 kB in 2s (217 kB/s)
Reading package lists... Done

(base) computer@computer-ThinkCentre:~$ sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:3 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 336 kB in 2s (173 kB/s)
Reading package lists... Done
(base) computer@computer-ThinkCentre:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer required:
```

**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`**

```
(base) computer@computer-ThinkCentre:~$ sudo docker run hello-world
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/

(base) computer@computer-ThinkCentre:~$ docker -v
Docker version 23.0.1, build a5ee5b1
(base) computer@computer-ThinkCentre:~$
```



## 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

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a project named 'APP' containing 'spec', 'src', and 'Dockerfile'. The 'Dockerfile' tab is selected, displaying the following code:

```
1 # syntax=docker/dockerfile:1
2
3 FROM node:18-alpine
4 WORKDIR /app
5 COPY .
6 RUN yarn install --production
7 CMD ["node", "src/index.js"]
8 EXPOSE 3000
```

The right side shows the terminal window with the following command history:

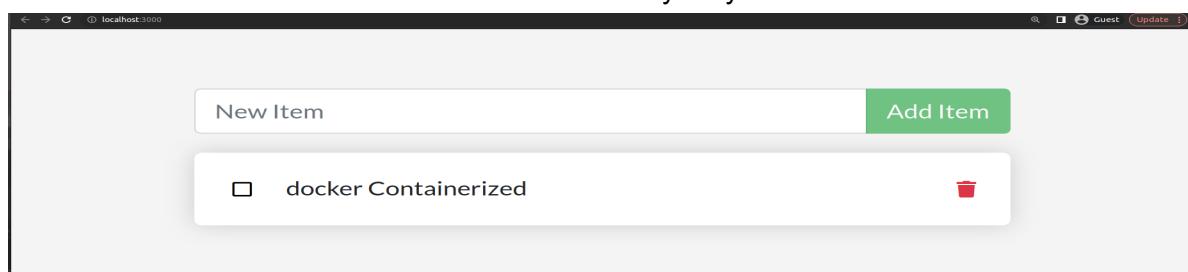
```
● computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$ touch Dockerfile
● computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$ docker build -t getting-started .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://<2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
● computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$ sudo docker build -t getting-started .
[sudo] password for computer:
Sorry, try again.
[sudo] password for computer:
[+] Building 19.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 184B
```

At the bottom, the status bar shows: Ln 8, Col 12, Spaces: 4, UTF-8, LF, Docker, Go Live.

**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
00 ge+-----+
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**

<b>NAME</b>	<b>PRATHAMESH CHIKANKAR</b>
<b>BRANCH</b>	<b>CSE-(AI&amp;ML)</b>
<b>ROLL NO.</b>	<b>11</b>
<b>SUBJECT</b>	<b>SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB</b>
<b>COURSE CODE</b>	<b>CSL603</b>
<b>PRACTICAL NO.</b>	<b>10</b>
<b>DOP</b>	
<b>DOS</b>	

## Experiment No. : 10

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

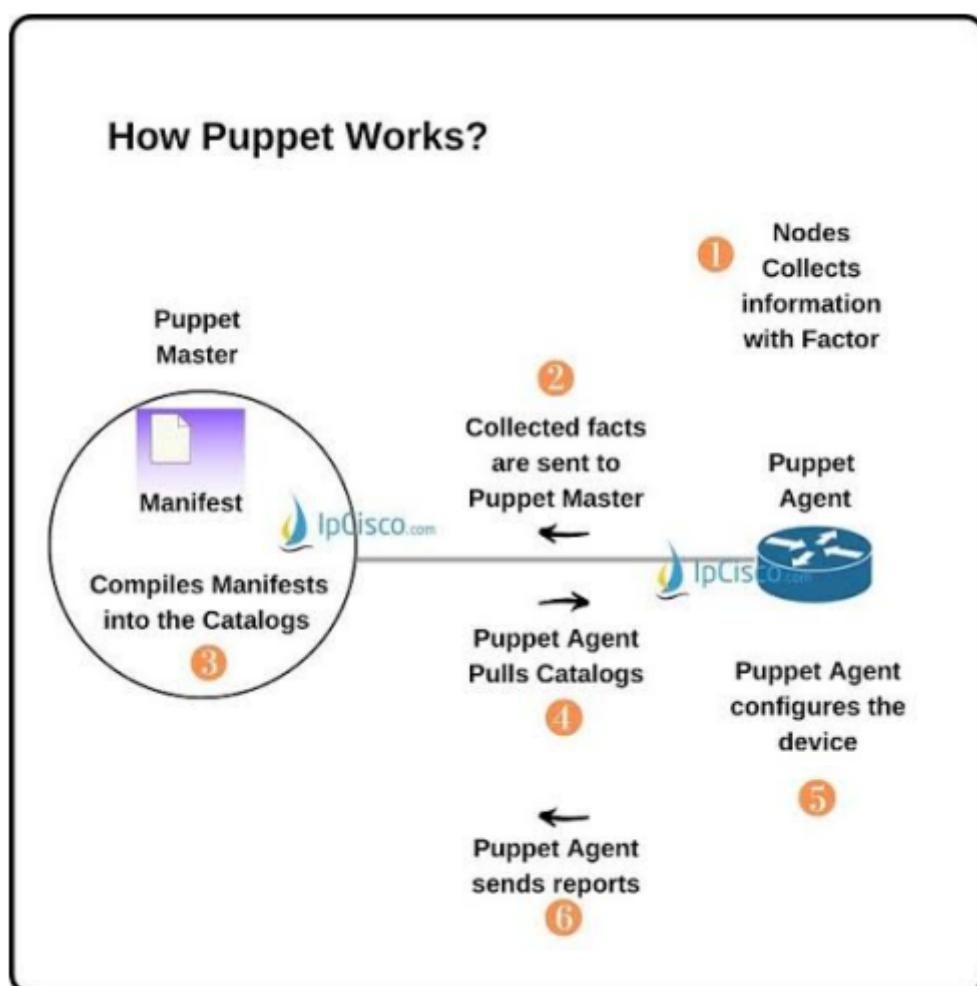
### Theory : Puppet

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.

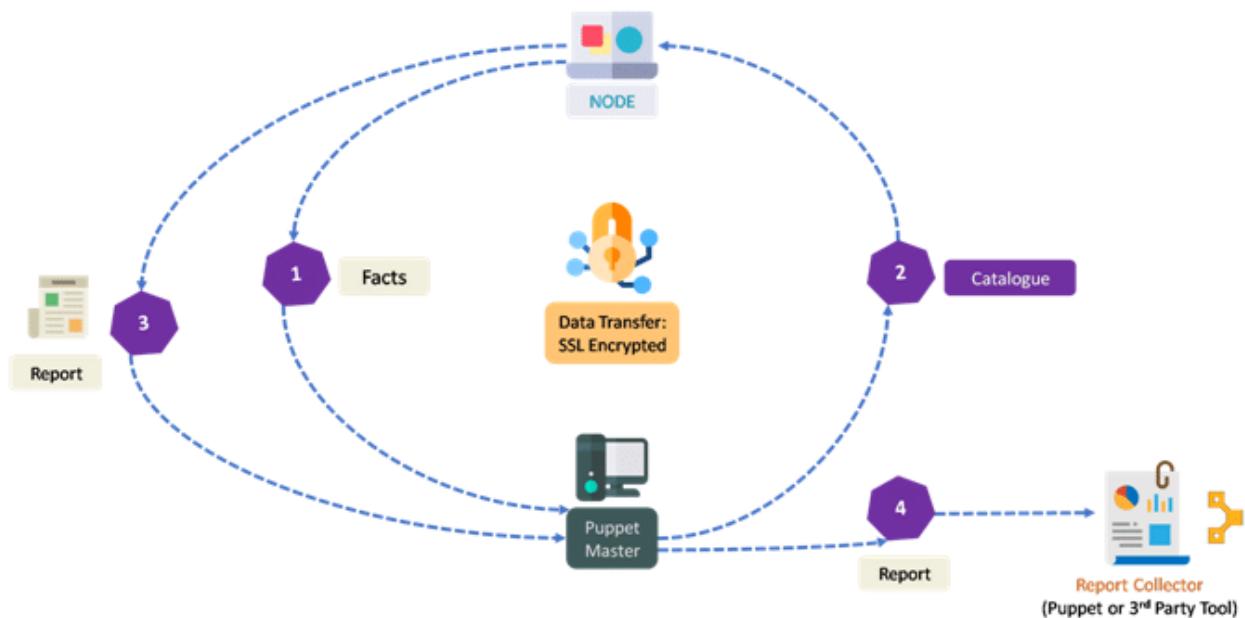
**Learning pull-based software management and provisioning tools using Puppet involves understanding the following concepts and steps:**

- 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.



Name :- Prathamesh S. Chikankar

Roll No. :- AIML11

Branch :- CSE-(AI & ML)

Year :- TE

Subject :- Software Engineering and  
Project management (SEPM)

Topic :- Assignment No. 01

Sign :- Prathy

Date :- February '23



(1) Explain in detail COCOMO model

COCOMO (constructive cost estimation model) is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of software.

The necessary steps in this model are:

1. Get an initial estimate of development effort from evaluation of thousands of delivered lines of code (KLOC)
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all multiplying factors i.e. multiply values in step 1 & step 2.

The initial estimate determines using KLOC as measure of size.

To determine initial Effort E in person-months :  $E_i = a \times (KLOC)^b$

In COCOMO, projects are categorized into three types!

1. Organic : A development project can be treated of organic type, if the project deals with a well-developed understood application program with size is reasonable & experienced team members.
2. Semi-structured : A development project can be treated with semi-structured type if the development consists of mixture of experience & unexperienced. Example includes developing a new operating system.
3. Embedded : SW being developed is strongly coupled to complex hardware, or its structure regular or open ended method exist. Eg:- ATM.

SW cost estimation should be done through three stages:

1. Basic COCOMO model : provide accurate size of project parameters.  
The following expression give basic COCOMO estimation model:



$$\text{Effort} = a_1 * (\text{KLOC})^{a_2 \cdot PM} \quad \dots \quad \text{KLOC} = \text{kilo bytes of code}$$

$$T_{Dev} = b_1 * (\text{Effort})^{b_2 \cdot PM} \quad \dots \quad a_1, b_1 = \text{constants}$$

$\text{Effort} \rightarrow$  estimated time ;  $\rightarrow T_{Dev} \Rightarrow$  To develop

(total effort required)  $\rightarrow$  in person months (PMs)

2. Intermediate model : Recognize the above basic model facts & signifies the initial estimates obtained through COCOMO model through following 4 drivers & 15 set.

i) product attributes : required software reliability extent, size of application database & complexity of product

ii) Hardware attributes : Run time performance constraints, memory constraints, the volatility of virtual machine environment & required turnabout time.

iii) personal attribute : Analyst capability, SW engg. capability, application experience, virtual machine experience and programming language experience.

iv) project attributes : Use of SW tools, application, object oriented engg. methods & required development schedules.

Equation of COCOMO at intermediate:

$$E = a_i (\text{KLOC})^{b_i + EAF}$$

$$D = c_i (E) d_i$$

3. Detailed COCOMO model : Incorporate all qualities of the standard version with an assessment of cost driver, effect on each method of SW engg. process. Six phases of detailed COCOMO :-

i) planning & requirements

ii) system structure

iii) complete structure

iv) module code & test

v) integrate and test

vi) cost constructural model



(2)

## Software Engineering & software process

Software Engineering combines two concepts, software & engineering.

Software: As we have already seen software is set of executable programs related to specific functionality. Set of executable instructions is called as a program.

Engineering: Engineering is an application of knowledge & well defined principles to develop products.

Software Engineering is the method of applying scientific & technological knowledge, procedure, and rules to design, develop, and maintain the software product.

Software process: framework approach to software process.

Divided into 4 layers:

1. A quality process: Engg. approach must rest on quality
  2. boundary for SE is process layer
  3. methods: provide technical guides for building SW.
  4. Tools: provide automated & semi-automated support for process & method
- So, SW processes in SW engg refers to the methods & techniques used to develop and maintain software. Some examples of SW process includes: waterfall, Agile, Scrum, DevOps, etc.

(3)

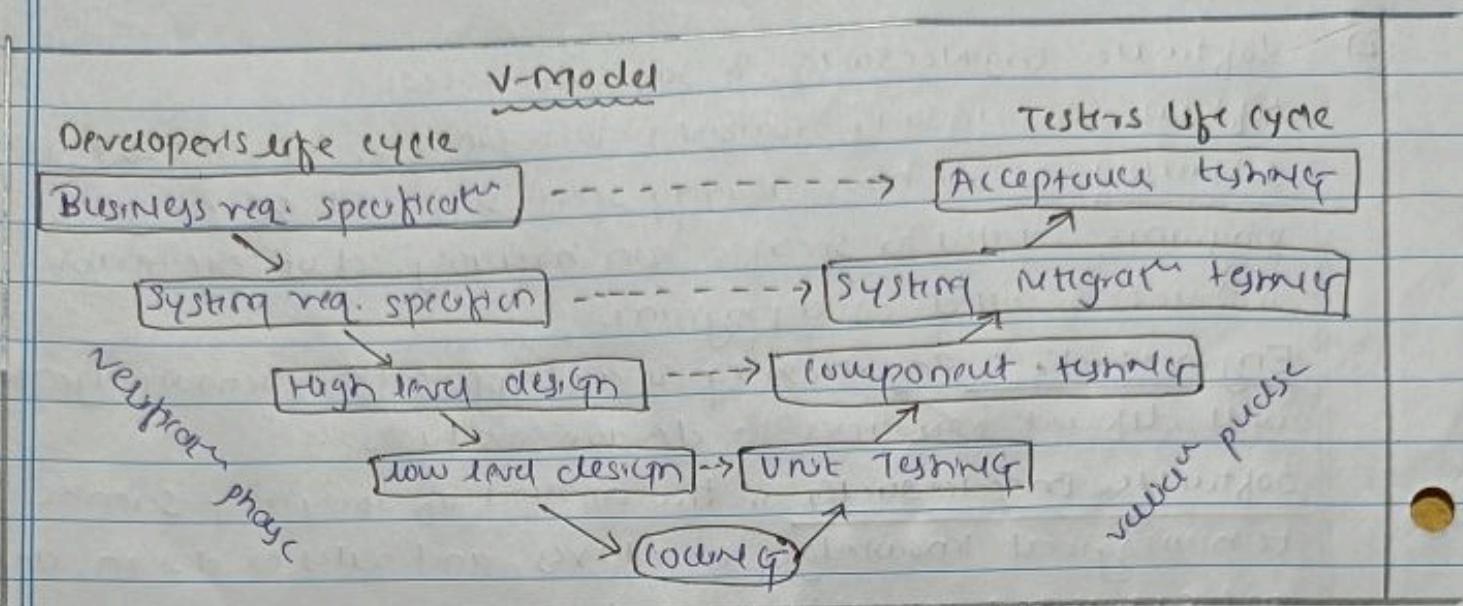
## V-model

V-model also referred to as the verification and validation model.

Verification: it involves a static analysis method (review) done without executing code.

Validation: involving dynamic analysis method (functional, non-functional) testing is done by executing code.

Verification & validation process is join by coding phase in V-shape. Thus it is known as V-model



\* Variety phases of verification phase of V-model:

- 1) Business requirement analysis: product requirements understood from the customer's side.
- 2) Design issue: proposed system by studying user requirements docs.
- 3) Architectural design: selecting the architecture so that it should understand all which typically consist of lists, dependency, map, etc..
- 4) Module design: systems break down into small modules.

\* Variety phases of validation phase of V-model:

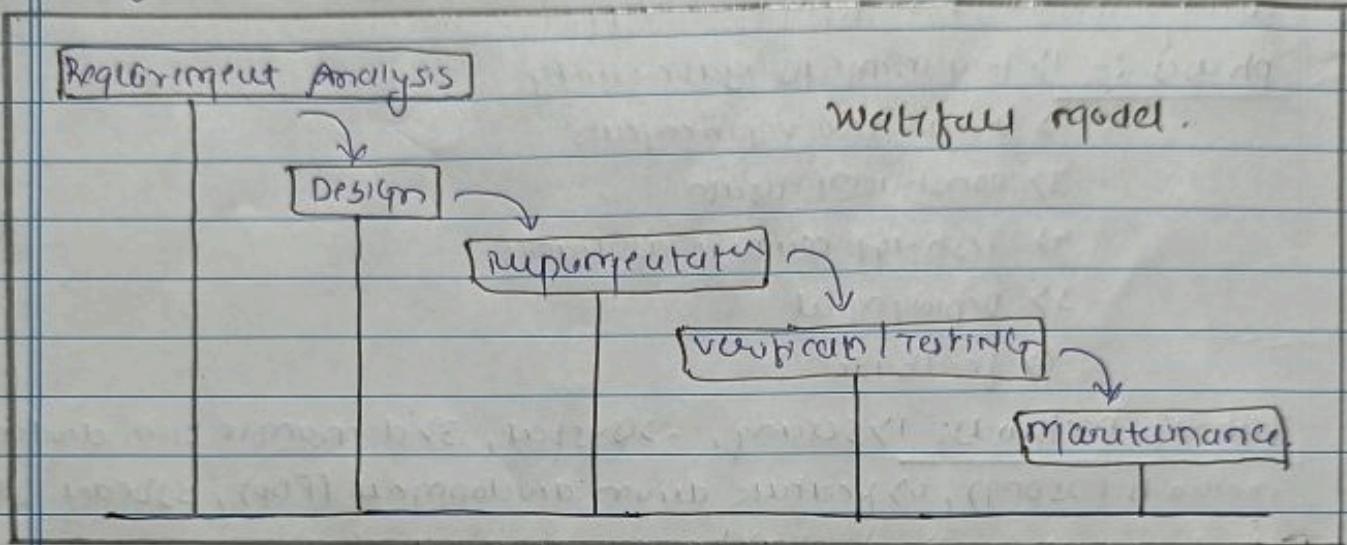
- 1) Unit testing: To eliminate the errors at code level or unit level.
- 2) Integration testing: verify that groups created and tested independently can coexist & communicate among themselves.
- 3) System Testing: composed by the above business trans.
- 4) Acceptance testing: testing the product in user atmosphere.

- Coding phase: intermediate, after designing coding phase is started. Based on the requirements, the suited programming language is decided. There are some standards & guidelines for coding; to check the performance, through many code reviews.

(4)

## Waterfall Model

- first approach used in software development process.
- Also called as classical life cycle model or linear sequential model.
- Any phase of development process begins only if previous is terminated.



1) Requirement Analysis: All business requirements are gathered & analyzed by communicating with stakeholders & customer/user.

2) Design: Based on requirement, specific document design of system is created called SW architecture.

3) Implementation: Actual coding is constructed for SW architecture using hardware & software requirements of system.

4) Verification / Testing: Coding & job done by developer is verified here.

5) Maintenance: While using SW, if user faces some problem, then those problems must be solved here by development team.

\* Advantages: very simple to understand & easy to use.  
 phases don't overlap with other.

\* Disadvantages: Not useful for large projects  
 System / product available only at the end of development process.



⑤

### What is agile process?

The meaning of agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller items or parts do not directly involve long term planning.

phases :-

- 1) Requirements gathering
- 2) Design the requirements
- 3) construction
- 4) Testing / quality assurance
- 5) Deployment
- 6) feedback

Testing methods: 1) SQA, 2) System, 3) dynamic software development methods (DSDM), 4) feature driven development (FDD), 5) lean software development (LSD), & 6) extreme programming (XP).

\* Advantage: frequent delivery

face-to-face communication with clients.

\* Disadvantage: Due to shortage of formal documents, it creates confusion and crucial decisions.

⑥

### Explain Agile principles:

The manifesto for agile software development is based on twelve principles -

- 1) customer satisfaction by early & continuous delivery of valuable software.
- 2) welcome changing requirements, even in late development.
- 3) working software is delivered frequently (weeks rather than months)
- 4) close, daily cooperation between business people & developers.
- 5) projects are built around motivated individuals, who should trust.
- 6) face-to-face conversation best form of communication (10-10 rule)
- 7) working software is the primary measure of progress.
- 8) sustainable development, able to maintain a constant pace



- 9) continuous attm to technical excellence & good discipline.
- 10) simplicity is essential.
- 11) Best architectures, requirements & design emerge from self-org. teams.
- 12) Regularly, team reflect how to become more objective, and also adjusts it accordingly.

#### (7) Explain SRS documents!

- SRS stands for System Requirements Specification.
- SRS is a special kind of document which contains user requirements for a system which states properties & constraints that must be satisfied by a SW System.
- The outcome of the requirements gathering & Analysis phase of the SDLC is software requirements specification (SRS).
- SRS is also called as requirements document.
- Generally signed at end of requirements engineering phase.

\* following are 6 requirements, which SRS document should follow:

- 1) SRS document should specify external system behaviour.
- 2) SRS document should specify implementation constraints.
- 3) It should be easily changeable if any changes occur.
- 4) It should act as reference tool for maintaining the system.
- 5) SRS document record forethought about the lifecycle of the system.
- 6) It must include acceptable response to undesired events.

\* Advantage: SRS gives base for verification of completely developed product.

Using high-quality SRS, we can develop high-quality SW product.

A high-quality SRS decreases the cost of development.



## \* characteristics of SRS:

- Characteristics of SRS:

  - ① correct
  - ② unambiguous
  - ③ complete
  - ④ Ranked for importance / stability
  - ⑤ modifiable
  - ⑥ Traceable
  - ⑦ verifiable

\* SAS format:

SAS format:  
Structure of SAS document includes following sections with the various subsections in it:

## Format of SRS

- i) purpose
  - ii) scope
  - iii) def'n, Acronym,  
& Abbreviations
  - iv) references
  - v) overview

↳ the overall description:

- i) product perspective
  - ii) product function
  - iii) user characteristics
  - iv) constraints
  - v) Assumption & dependency
  - vi) Apportioning of requirements.

## L) specific requirements

- i) Nutyfaces
  - ii) Database
  - iii) Performance
  - iv) Slow System attributes

## ↳ Change management process

### b) Document approvals

## ↳ Supporting information



Name :- Prathamesh S. CHIKANKAR

Roll No. :- AIML11

Branch :- (SE- (AI & ML))

Subject :- Software Engineering & project management  
(SE & PM)

Topic :- Assignment No. 02

Year :- TE

Date :- April '23

Sign :- Prathu



Q1. SCM is an important part of RMM because?

- In software development, SCM stands for Software Configuration Management, which is the process of identifying, organizing and controlling changes made to software artifacts throughout their lifecycle.
- On the other hand, RMM stands for Risk Management, monitoring and mitigation, which is a process used to identify, evaluate and manage potential risks that may affect a software project.
- SCM is an important part of RMM because it helps manage and mitigate risks related to changes made to software artifacts. By tracking changes and versions of software artifacts, SCM helps identify potential risks related to changes that could impact the software's functionality, performance, or security. SCM also provides a mechanism to control changes, ensuring that only authorized changes are made and that those changes are properly tested and validated before being integrated into the software.
- In addition, SCM provides visibility into the history of changes made to software artifacts, which can be used to identify the root cause of issues to roll back changes that may have introduced new risks or problems.
- Overall, SCM plays a critical role in the risk management process by ensuring that changes made to software artifacts are properly controlled and validated, and that potential risks associated with those changes are identified and managed appropriately.

Q2. What is formal technical review in software engineering?

- In software engineering, a formal technical review (FTR) is a structured and systematic process of evaluating and improving the quality of software artifacts, such as requirements specifications, design documents, code and test plans. FTRs are typically conducted by a team of peers of



a follower with the SW development process and have expertise in the relevant technologies and domains.

- The objective of a FTR cycle:

- ✓ 1) To uncovers errors in function, logic or implementation for any representation of the software.
- ✓ 2) To verify that SW under review meets its requirements.
- ✓ 3) To ensure that SW has been represented according to predefined world standards.
- ✓ 4) To achieve SW that is developed in a uniform manner; and
- ✓ 5) To make projects more manageable.

In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to SW analysis, design and the implementation.

The FTR is actually a class of reviews that include walkthroughs and inspections.

Each FTR is conducted as a meeting and will be successful only if it is properly, controlled, planned and attended.

The process usually involves the following steps:

- 1) planning
- 2) preparation
- 3) follow-up from both periods and time
- 3) meeting

CQ 3: Explain WSHH principles:

- Barry Boehm suggested an approach that addresses project objectives, milestones and schedules, responsibilities, management and technical approaches and required resources. This is called WSHH principles.  
The questions that are answered in this principle are:
  - o why is the system being developed?
  - o what will be done by whom?
  - o who is responsible for a function?



- o where they are organised & located organizationally?
  - o How will job done technically and managerially?
  - o How much of each resource is needed?
  - o why is the system being developed?
- It enables the project to assess the availability of business reasons for the S/W work. Justifies the expensiveness of people's time and money.
- what will be done? & when?
- specifies the task set required for project & it helps to determine the project schedule, helps in determining when tasks are conducted and when milestones are reached.
- who is responsible for a function?
- helps to accomplish roles & responsibilities of each member of S/W team.
- where they are organizationally located?
- S/W team chart contains all the roles & responsibilities.
- How will job be done technically and managerially?
- Management & technical standards decided when product is established.
- How much of each resource is needed?
- Helps in deriving estimates based on the ans. to the above questions.

#### Q4. Define S/W diagrams

In S/W engineering, S/W diagrams are graphical representation of a project's tasks and their dependencies. There are two primary type of S/W diagrams used in project management:

- Activity-on-node (AON)
- Activity-on-arrow (AOA)

AON diagrams, also known as precedence diagramming method (PDM) represent tasks as nodes and the dependency b/w them as arrows. The nodes represent the start and end of each task and the arrows represent the dependency b/w tasks. AON diagrams are often used in

## Critical path method (CPM) analysis

CPM diagrams is a project management technique that uses Network diagrams to determine the critical path method of a project. The critical path is the sequence of tasks that must be completed within its deadline. CPM analysis helps identify tasks that can be delayed without affecting the overall project timeline and helps project managers allocate resources and prioritize tasks.

PERT, or program Evaluation & review technique, is a project management technique that uses Network diagrams to estimate the project duration and identify critical tasks. Unlike CPM, PERT takes into account the uncertainty and variability of task duration, which is useful in projects with a high degree of uncertainty. PERT uses three estimates for task duration: optimistic, pessimistic and most likely, to calculate an expected duration for each task.

AoA diagrams, also known as Arrow Diagramming Methods (ADM), represent tasks as arrows and the nodes as events. The arrows represent the tasks & their duration, while the events represent the start and end of each task. AoA diagrams are often used in PERT analysis.

In summary, N/W diagrams, including AoA and ADM, are graphical representations of a project's tasks and dependencies. CPM and PERT are project management techniques that use N/W diagrams to analyze and optimize project schedules, estimate project durations and identify critical tasks.



Q5 what is business process Reengineering?

Business process Reengineering (BPR) is a management approach that seeks to redesign and optimize business processes within an organization to achieve significant improvements in efficiency, quality & customer satisfaction.

Business process Reengineering is a redesigning of business process to enhance product output, improve product quality, enhance speed or reduce costs.

#### Benefits of BPR:

- Reduced operational costs: Can lower company's operational costs.
- Enhanced quality of products or service: changing complex work flow.
- Increased Speed of Business operation: By Streamlining operat.
- Improved customer service: speeding up operat. waste process.
- Boosted profits: allows you to reorganize the company.
- clarified purpose: allows you to re-balance yourself.

#### Implementation of BPR:

① Analyze your employer's current business process.

- gain an understanding;
- find out how current opem. utilize resourecy & tools.

② Identify any process gaps.

- Note areas where frequent delays & errors;
- opportunity for improvement.

③ Create a future process diagram.

- solve problem that you identified

- identify how new process is more productive.

④ Implement future changes

- ask for feedback → followed additional changes...



Examples of BPR:

- 1) Upgrading technology      3) Improving O/P
- 2) Redesigning jobs      4) Cutting costs
- 6) Increase product quality      5) Streamlining processes
- 7) Creating cross-functional teams

## 6. Which are Web application testing techniques?

Web Application Testing is the process of testing web-based applications to ensure that they are functioning as expected & meeting user requirements. There are several techniques that can be used to test web applications, including:

### 1) Functionality Testing:

This technique involves testing the application's functionality to ensure that it works as expected. (Includes user interface, data operation, etc..)

### 2) Performance Testing:

This technique involves testing the application performance under different condn, such as high traffic volume or heavy data loads.

### 3) Security Testing:

This technique involves testing the application security features to ensure that they are working as expected. (vulnerabilities like SQL injection)

### 4) Compatibility Testing:

This technique involves testing the application compatibility with different browsers, OS and the devices (on different versions of browser).

### 5) Usability Testing:

using functionality & experience to ensure that, it is easy to use.

### 6) Regression Testing:

Retesting the application after changes have been made.

### 7) Acceptance Testing:

using requirement (ensure, meets user needs & expectation)



Q7.

- what are Golden Rules apply while designing a software?
- User interface Golden rules: (for GUI design) - 5W
- 1) Steve for consistency: keep words & arm consistent;  
example: use computer control,
  - 2) visibility of system status or offer informative feedback! feedback  
in a reasonable time; e.g.: OS just about status
  - 3) match b/w System & user word or design dialog to yield closure!  
System should speak user language; e.g.: payment proceeding.
  - 4) uses control & freedom or permit easy reversal of actions encourage  
exploration of unfamiliar optn; e.g.: Document History
  - 5) Error prevention & simple Error handling! Either eliminate error-prone  
conds or check for them & notify user; e.g.: password entry
  - 6) Produce short-term memory load & recognition rather than recall!  
Memory optn available & instrns visible; e.g.: Ring / silent switch.
  - 7) Enable frequent users to use shortcuts: manipulate & personalise keys,  
fincm keys, hidden Cmds, etc...; e.g.: built-in commands & shortcuts.
  - 8) Aesthetic and minimalist design: all information should be valuable and  
relevant; e.g.: simple interface
  - 9) Help users recognise, diagnose, & correct form errors: error msg may be express  
in plain language (suggest join); e.g.: plain message
  - 10) Help and documentation: easy to learn, forward on the user's task,  
list of concrete steps, and no too large

Q8.

Explain in details Design process and quality.

The design process is an important part of SE and involves creating a plan for development of a software product. A sequence of steps for implementation follows:

(1) Requirement analysis: involves identifying and understanding the user requirements for the software product.



- ② Design planning: involves creating plan for design process, including the selection of appropriate tools, techniques and resources.
- ③ High-level design: involves creating high-level design for the product including overall architecture, DFA will be used.
- ④ Detailed design: involves creating detailed design of a SW product including detailed specification for each component & module.
- ⑤ Implementation: involves writing the actual code for the SW product based on the detailed design.
- ⑥ Testing: This involves testing the SW product to ensure that it meets desired standards of functionality, performance & reliability.

Quality is an important aspect of the design process, as it ensures that the SW product meets the desired standards. Quality can be achieved by using variety of techniques, such as:

- ① Quality planning: involves identifying the quality requirements for the SW product and creating a plan.
- ② Quality control: involves monitoring SW development process to ensure that it meets the desired quality standards.
- ③ Quality assurance: This involves testing SW product to ensure that it meets the desired quality standards & implementing measures.
- ④ Quality improvement: involves using feedback from quality control & quality assurance activities to improve SW development process & ensure the SW product meets desired quality standards.

By following a rigorous design process & implementing quality control and assurance measures, SW engineers can create SW products that meet the need of user & meet the desired quality standards.