

## 1.Aim: Demonstrate working with GitHub using GitHub Portal.

### Source Code:

GitHub is a web-based platform that uses Git for version control. It is widely used by software developers for collaborative coding and project management. Here's an overview of the basic functionalities of GitHub.

**Repository (Repo):** A repository is a container for a project, where all the project files, history, and documentation are stored. Repositories can be public (visible to everyone) or private (accessible only to specified collaborators).

**Branching:** Git enables branching, allowing developers to create different lines of development. Branches can be used for features, bug fixes, or experiments without affecting the main codebase.

**Commits:** A commit is a snapshot of changes made to the code.

**Pull Requests:** Pull Requests (PRs) allow developers to propose changes from one branch to another. They facilitate code review, discussion, and collaboration before changes are merged.

**Issues:** GitHub Issues are used to track tasks, enhancements, bugs, or any other kind of work. Issues can be assigned, labeled, and commented on, providing a centralized location for project communication.

**Merging:** Merging is the process of combining changes from one branch into another.

**Forks:** Forking a repository creates a copy of the original repository in the user's account.

**Collaborators:** Collaborators are individuals with access to a repository. They can push changes, review and merge pull requests, and manage issues.

**GitHub Actions:** GitHub Actions automate workflows, allowing users to define custom CI/CD (Continuous Integration/Continuous Deployment) processes directly within the repository.

**Wiki:** Repositories can have a wiki for documentation. It's a place to store and share information about the project, providing additional context and resources.

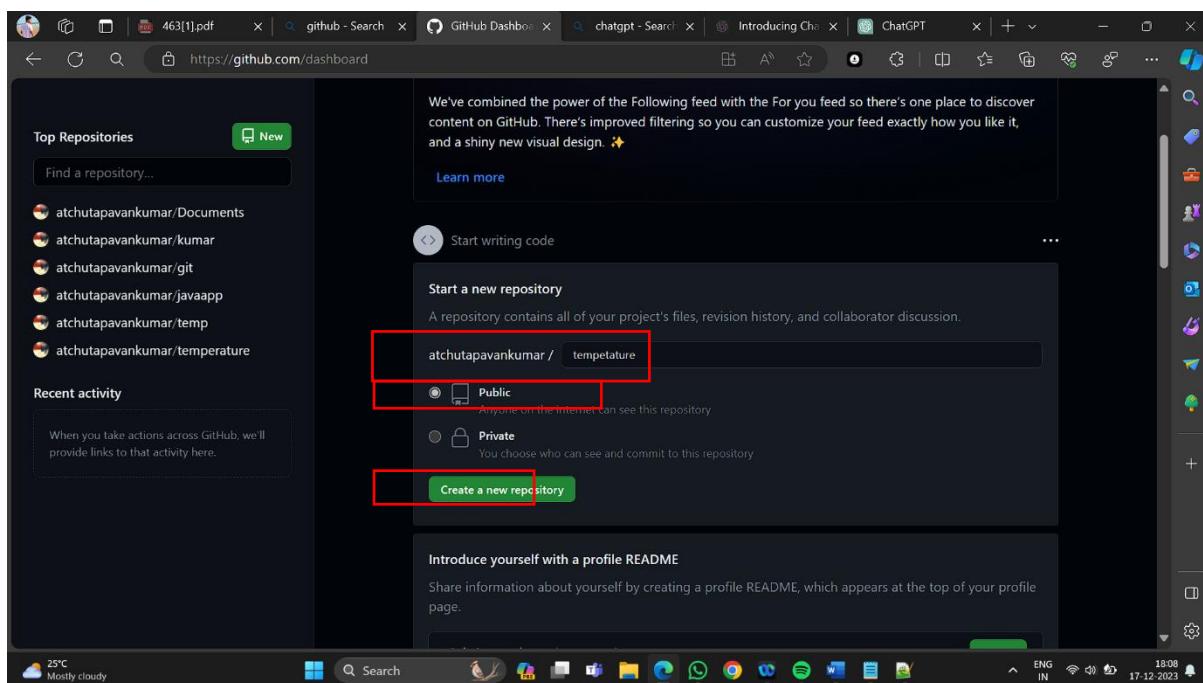
**GitHub Pages:** GitHub Pages enables the hosting of static websites directly from a GitHub repository.

**Gists:** Gists are like mini-repositories for sharing and collaborating on single files or snippets.

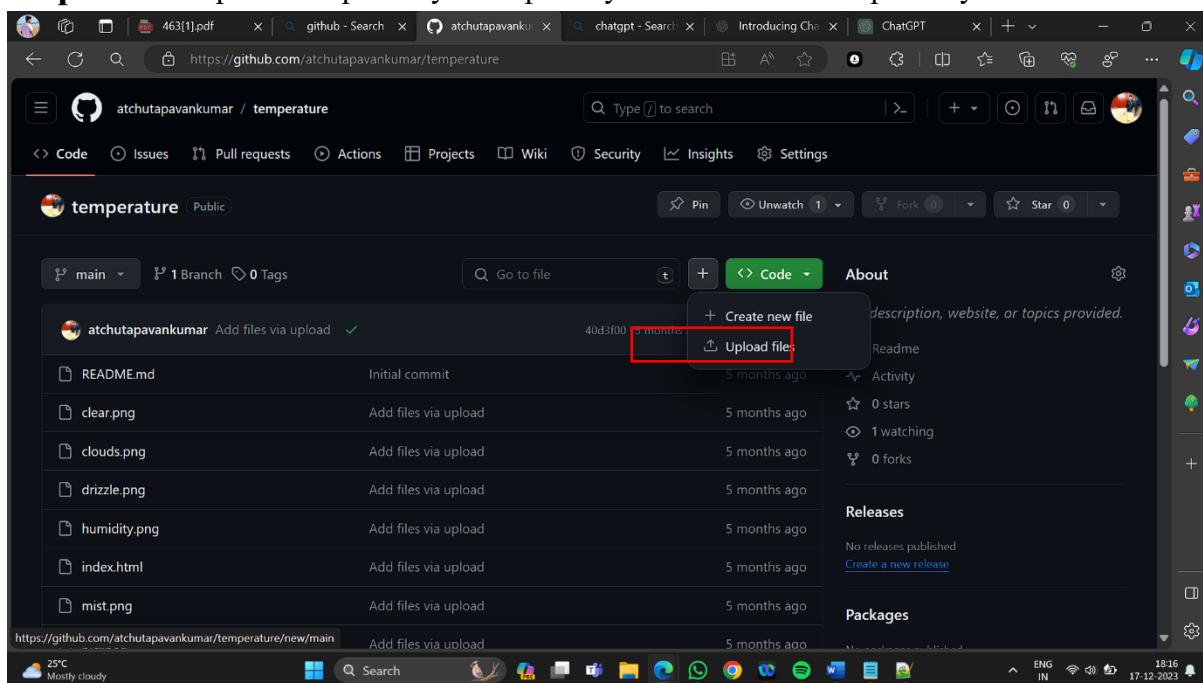
### Steps to deploy a new repository on GitHub:

**Step-1:** Logon to your GitHub account on browser. Create one if you don't have a GitHub account.

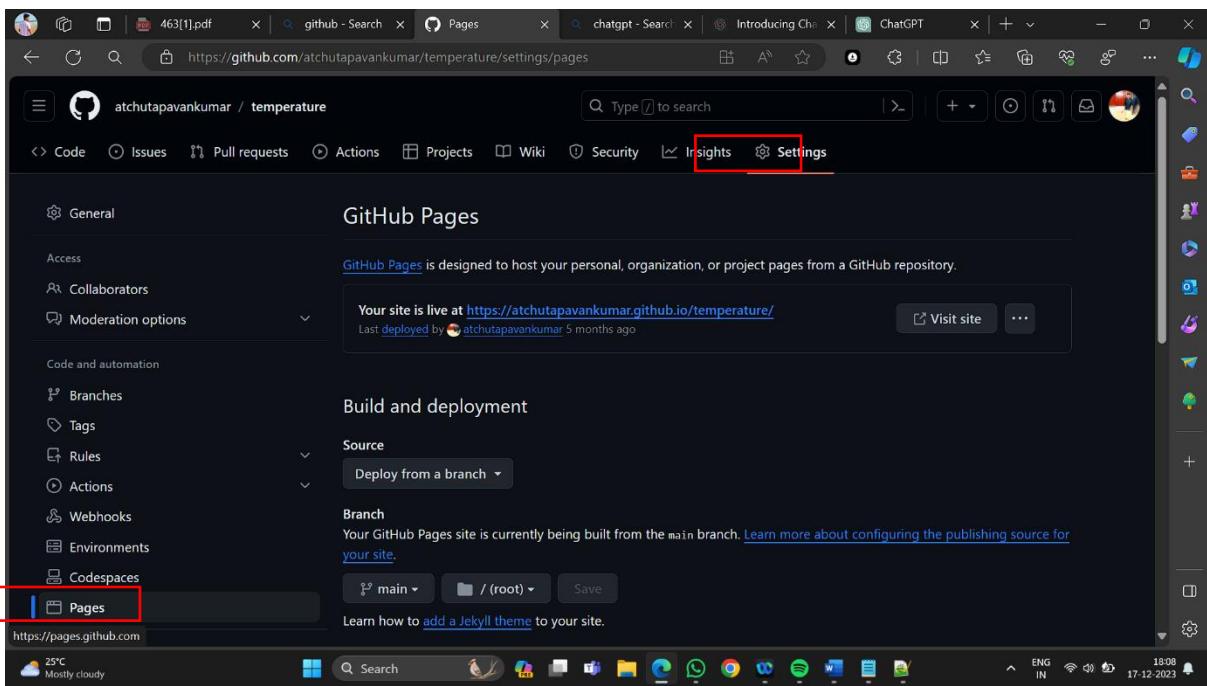
**Step-2:** To create a new repository, goto **Start a new repository** column and give the repository name. Make sure the repository is **Public** and add a **ReadMe file** to the repository.



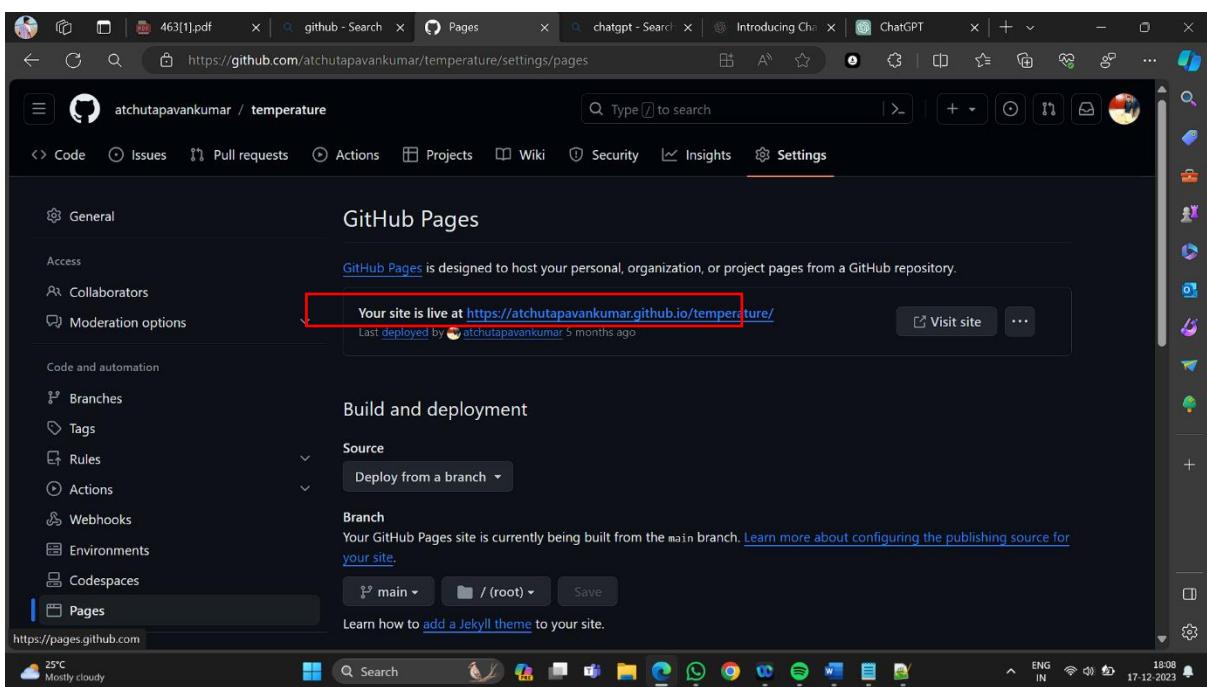
**Step-3:** Now open the repository and upload your files onto the repository.



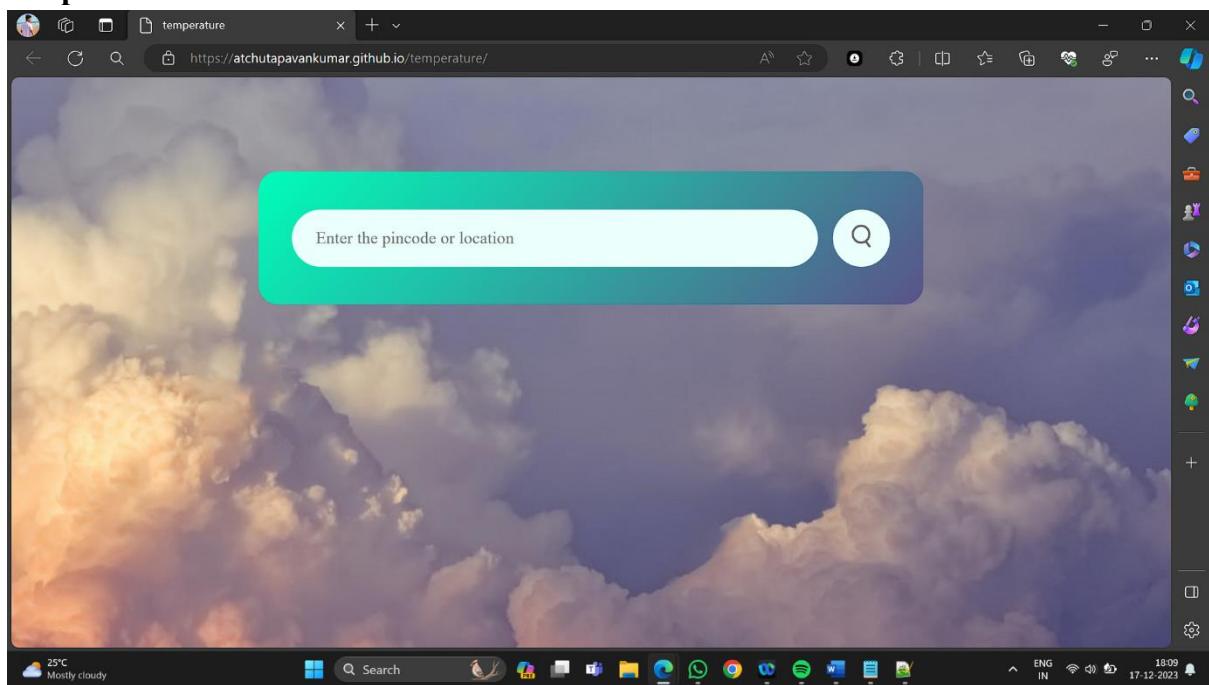
**Step-4:** Open the repository **settings** and goto **pages** under **code and automation** settings.



**Step-5:** Copy the live link of uploaded file and paste it on browser window to view your uploaded file.



**Output:**



## 2.Aim: Demonstrate working with Git Shell Commands

### Source Code:

#### 1.Git init:

To start using git,navigate to the directory where your project is located and run

```
D:\Devops>git init
Initialized empty Git repository in D:/Devops/.git/
```

#### 2.Adding files to staging area:

You can add your files to the staging area to prepare them for a commit using the ‘git add’ command for example to add a file called ‘myfile.txt’

```
D:\Devops>git init
Initialized empty Git repository in D:/Devops/.git/
D:\Devops>git add myfile.txt
```

#### 3.Commiting changes:

Once you have added files to the staging area you can commit them to repository with a commit message.using ‘git commit’ command

```
D:\Devops>git commit -m "your commit changes"
[master (root-commit) e747d12] your commit changes
 1 file changed, 1 insertion(+)
 create mode 100644 myfile.txt
```

#### 4.Checking Status:

To see the status of your repository and which files have been modified, added or deleted use ‘git status’

```
D:\Devops>git status
On branch master
nothing to commit, working tree clean
```

#### 5.Viewing commit history:

You can view the commit history of your repository using ‘git log’

```
e747d1241f9e33ae983822c7c589ab84faef061c (HEAD -> master)
```

```
Author: Narendr <narendramurukurthi@gmail.com>
```

```
Date: Sun Dec 3 19:26:28 2023 +053
```

#### 6.creating and switching Branches:

To create a new branch and switch to it use ‘git checkout -b’

```
D:\Devops>git checkout -b mynewbranch
Switched to a new branch 'mynewbranch'
```

## 7.Merging Branches:

To Merge changes from one branch to another use the ‘git merge’ command. for example to merge changes from feature-branch into another branch.

```
D:\Devops>git merge newbranch
Already up to date.
```

## 8.cloning a repository:

To clone a remote git repository to your local machine use the ‘git clone’  
git clone <https://narendra/documents>

```
Cloning into 'Documents'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 37 (delta 5), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (37/37), 1.91 MiB | 2.72 MiB/s, done.
Resolving deltas: 100% (5/5), done.
```

## 9.pushing changes to a remote repository:

To push your local changes to a remote repository use ‘git push origin main’

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up D:\Devops>git push -u origin main
to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/narendra/myfile.git
 44c9bc1..133e0f3 main -> main
branch 'main' set up to track 'origin/main'.
```

## 10.pulling changes from a remote repository:

To get the latest changes from a remote repository use the ‘git pull’ command  
git pull <https://narendra/documents>

```
remote: Enumerating objects: 18, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (16/16), done.  
Unpacking objects: 100% (18/18), 110.28 KiB | 1.38 MiB/s, done.  
remote: Total 18 (delta 0), reused 0 (delta 0), pack-reused 0  
remote: Total 18 (delta 0), reused 0 (delta 0), pack-reused 0
```

### 3. Aim: Demonstrate using Jenkins Service with Git plugin in AWS

#### Source Code:

##### Step-1: Creating the aws EC2 instances:

- 1.Log in to the AWS Management Console.
- 2.Navigate to the EC2 service.
- 3.Click on “Launch Instance” to create a new EC2 instance.
- 4.Select the desired instance type, storage, and other configurations.
- 5.Configure the security group to allow inbound traffic on port 22 for SSH access.
- 6.Launch the instance and download the private key file (.pem).

The screenshot displays two consecutive steps in the AWS EC2 instance launch wizard. The top window is titled 'Launch an instance' and shows the summary of the instance configuration: 1 instance, t2.micro instance type, a new security group, and 1 volume (8 GiB). A modal box is overlaid, stating 'Free tier: In your first year includes 750 hours of t2.micro'. The bottom window is titled 'Network settings' and shows the creation of a new security group named 'launch-wizard-1'. It includes options to allow SSH, HTTPS, and HTTP traffic from anywhere. Both windows have a prominent 'Launch instance' button at the bottom right.

##### Step-2:SSH into EC2 Instance, Install Java and Jenkins

SSH into the EC2 instance. After securely connecting to the instance, the first thing we have to do is to install the Java application onto the instance. Jenkins runs natively runs

on the Java application so it has to be properly installed in order for the Jenkins server to be operational.

1. sudo wget -O /etc/yum.repos.d/jenkins.repo \
 https://pkg.jenkins.io/redhat-stable/jenkins.repo

```
[ec2-user@ip-172-31-13-239 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2023-12-17 07:10:03-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.154.133, 2a04:4e42:24::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.154.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.r 100%[=====]     85  --.-KB/s   in 0s

2023-12-17 07:10:03 (5.74 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-13-239 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[ec2-user@ip-172-31-13-239 ~]$ sudo yum upgrade
Jenkins-stable
Dependencies resolved.
Nothing to do.
Complete!
```

2. sudo dnf install java-17-amazon-corretto -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo dnf install java-17-amazon-corretto -y
Last metadata expiration check: 0:00:06 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
 Package           Arch      Version        Repository      Size
=====
Installing:
 java-17-amazon-corretto      x86_64    1:17.0.9+8-1.amzn2023.1  amazonlinux   188 k
Installing dependencies:
 alsa-lib              x86_64    1.2.7.2-1.amzn2023.0.2  amazonlinux   504 k
 cairo                x86_64    1.17.6-2.amzn2023.0.1  amazonlinux   684 k
 dejavu-sans-fonts      noarch   2.37-16.amzn2023.0.2  amazonlinux   1.3 M
 dejavu-sans-mono-fonts    noarch   2.37-16.amzn2023.0.2  amazonlinux   467 k
 dejavu-serif-fonts      noarch   2.37-16.amzn2023.0.2  amazonlinux   1.0 M
 fontconfig            x86_64    2.13.94-2.amzn2023.0.2  amazonlinux   273 k
 fonts-filesystem       noarch   1:2.0.5-12.amzn2023.0.2  amazonlinux   9.5 k
 freetype              x86_64    2.13.0-2.amzn2023.0.1  amazonlinux   422 k
 giflib                x86_64    5.2.1-9.amzn2023.0.1  amazonlinux   49 k
 google-noto-fonts-common  noarch   20201206-2.amzn2023.0.2  amazonlinux   15 k
 google-noto-sans-vf-fonts  noarch   20201206-2.amzn2023.0.2  amazonlinux   492 k
 graphite2             x86_64    1.3.14-7.amzn2023.0.2  amazonlinux   97 k
 harfbuzz              x86_64    7.0.0-2.amzn2023.0.1  amazonlinux   868 k
 java-17-amazon-corretto-headless  x86_64    1:17.0.9+8-1.amzn2023.1  amazonlinux   91 M
 javapackages-filesystem  noarch   6.0.0-7.amzn2023.0.6  amazonlinux   12 k
 langpacks-core-font-en  noarch   3.0-21.amzn2023.0.4  amazonlinux   10 k
 libICE                 x86_64    1.0.10-6.amzn2023.0.2  amazonlinux   71 k
 libSM                  x86_64    1.2.3-8.amzn2023.0.2  amazonlinux   42 k
 libX11                 x86_64    1.7.2-3.amzn2023.0.4  amazonlinux   657 k
 libX11-common          noarch   1.7.2-3.amzn2023.0.4  amazonlinux   152 k
 libXau                 x86_64    1.0.9-6.amzn2023.0.2  amazonlinux   31 k
 libXext                 x86_64    1.3.4-6.amzn2023.0.2  amazonlinux   41 k
```

3. sudo yum install jenkins -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:00:22 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
 Package          Architecture      Version       Repository      Size
=====
Installing:
jenkins          noarch          2.426.2-1.1   jenkins        85 M
Transaction Summary
=====
Install 1 Package

Total download size: 85 M
Installed size: 85 M
Downloading Packages:
jenkins-2.426.2-1.1.noarch.rpm           7.7 MB/s | 85 MB  00:11
Total                                         7.7 MB/s | 85 MB  00:11
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :                                         1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Installing : jenkins-2.426.2-1.1.noarch        1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Verifying   : jenkins-2.426.2-1.1.noarch        1/1

Installed:
jenkins-2.426.2-1.1.noarch
```

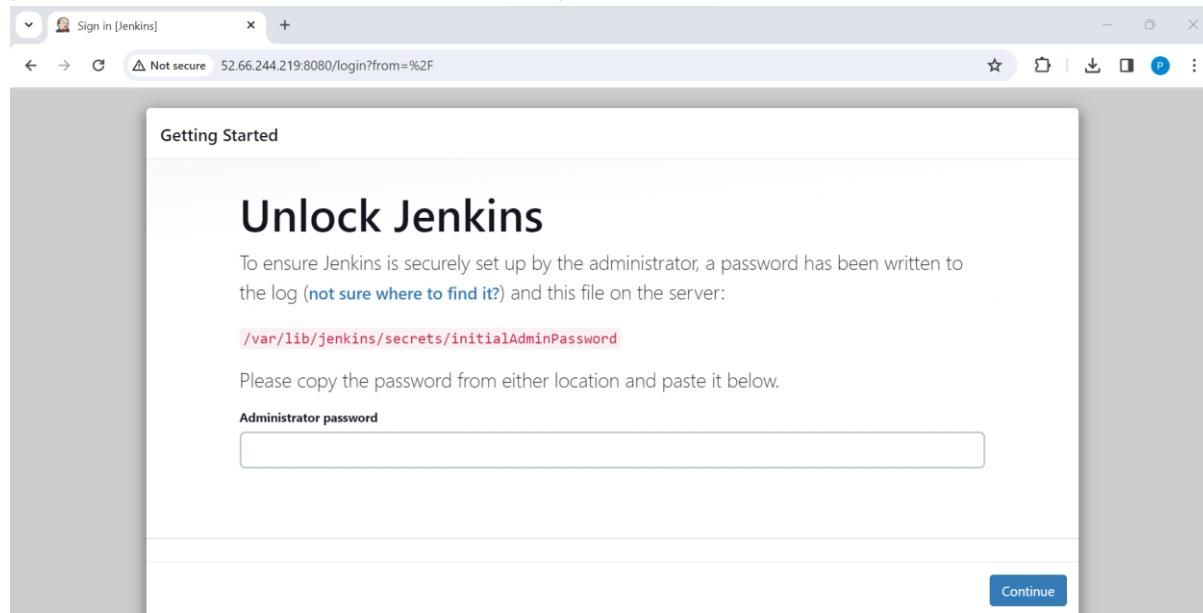
#### 4. sudo systemctl enable jenkins

```
sudo systemctl start jenkins
```

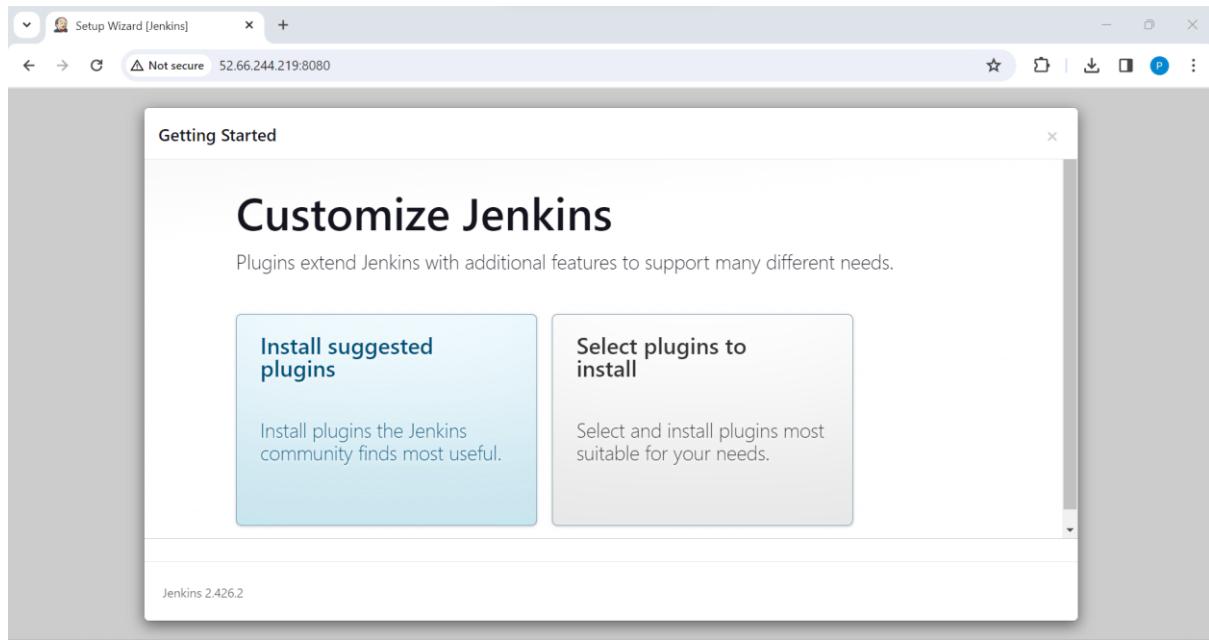
```
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl start jenkins
```

#### Step-3: Setting Up Jenkins Server

Now that the Jenkins server has been officially started on the instance, head over to the AWS Management console, grab the public IP address of the instance and paste that into your browser address bar followed by :8080



```
[ec2-user@ip-172-31-13-239 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
baaeef975c09e468a977274b59ed26a8e
[ec2-user@ip-172-31-13-239 ~]$
```



Setup an admin user. For this project we will get a dummy administrator user. In a real production you will use different credentials. Save and continue.

Username=admin

Password=Password

Full name= admin

Email=admin@admin.com

Getting Started

## Create First Admin User

Username  
admin

Password  
\*\*\*\*\*

Confirm password  
\*\*\*\*\*

Full name  
admin

E-mail address  
admin@admin.com

Jenkins 2.414.1 [Skip and continue as admin](#) [Save and Continue](#)

### Step 4: We are Live and Jenkins Ready! Time to Build

If you see the “Welcome to Jenkins” page, then you’ve successfully setup Jenkins and are ready to go to the next stage of the project, which is setting up our first Jenkins job!

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, Search bar, Notifications, User account (admin), Log out.
- Left Sidebar:**
  - New Item
  - People
  - Build History
  - Manage Jenkins
  - My Views
- Central Content:**

### 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 →
- Bottom Left:** Build Queue (No builds in the queue)
- Bottom Center:** Build Executor Status (1 idle, 2 idle)

Creating our first jenkin job

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below that is a 'Build Queue' section which says 'No builds in the queue.' To the right, the main area has a heading 'Welcome to Jenkins!' followed by a sub-section 'Start building your software project' with a 'Create a job' button. Underneath, there's another section 'Set up a distributed build' with 'Set up an agent' and 'Configure a cloud' options. At the bottom right of the dashboard, it says 'REST API Jenkins 2.414.1'. A URL '54.189.134.221:8080/newJob' is highlighted in a red box at the bottom left.

For our first job we are going to give it the name “**HelloWorldJob**” and select the option for “**Freestyle project**”. Click ok to move to the next step.

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

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

**Multi-configuration project**

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

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

**OK**



**Step-5:** Under the general configuration settings

In the description box, enter the text “HelloWorldJob”

Under the “Source Code Management” tab select “None” option

The screenshot shows the Jenkins job configuration page for 'HelloWorldJob'. At the top, there's a breadcrumb navigation: Dashboard > HelloWorldJob > Configuration. On the left, a sidebar titled 'Configure' lists several tabs: General (selected), Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'General' tab contains a 'Description' field with the value 'HelloWorldJob'. Below it are several checkboxes for build options: Discard old builds, GitHub project, This project is parameterized, Throttle builds, and Execute concurrent builds if necessary. An 'Advanced' dropdown menu is also present. The 'Source Code Management' section shows 'None' selected (radio button is filled) and 'Git' as an option. At the bottom are 'Save' and 'Apply' buttons.

- Navigate back to the Dashboard and you will see our 1st job in queue ready for the build stage.
- Click on “Build Now”
- After a few seconds you will see the the build output under the “build history”
- The console ouput should display a green checkmark indicating that the build was SUCCESSFUL.
- You can see that the echo “Hello World” and uptime commands were properly executed!

The screenshot shows the Jenkins interface for the 'HelloWorldJob' project. At the top, there's a navigation bar with the Jenkins logo, a search bar, and various icons. Below it, a breadcrumb trail shows 'Dashboard > HelloWorldJob'. The main area is titled 'Project HelloWorldJob'. On the left, a sidebar lists options: Status (highlighted), Changes, Workspace, Build Now (with a green arrow pointing to it), Configure, Delete Project, and Rename. To the right of the sidebar are buttons for 'Edit description' and 'Disable Project'. The central part of the page is titled 'Build History' with a 'trend' dropdown set to 'No builds'.

The screenshot shows the Jenkins interface for the 'Console Output' of build #1 of the 'HelloWorldJob'. At the top, there's a navigation bar with the Jenkins logo, a search bar, and various icons. Below it, a breadcrumb trail shows 'Dashboard > HelloWorldJob > #1 > Console Output'. The main area is titled 'Console Output' with a green checkmark icon. On the left, a sidebar lists options: Status, Changes, Console Output (highlighted), View as plain text, Edit Build Information, Delete build '#1', and Next Build. The central part of the page displays the console output log:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorldJob
[HelloWorldJob] $ /bin/sh -xe /tmp/jenkins13823842254370167628.sh
+ echo 'Hello World'
Hello World
+ uptime
16:45:33 up 37 min, 1 user, load average: 0.03, 0.03, 0.06
Finished: SUCCESS
```

## Step-6: Integrating git

### Source Code Management

None

Git ?

#### Repositories ?

##### Repository URL ?

`https://github.com/yankils/hello-world.git`

##### Credentials ?

- none -

Add ▾

Advanced ▾

Add Repository

#### Branches to build ?

##### Branch Specifier (blank for 'any') ?

`*/master`

Under Dashboard, you will see a new job for Git.

Click on “Build Now” if the build output shows a green checkmark indicator, that means our build was a SUCCESS!

The screenshot shows the Jenkins interface for a build named 'PullCodeFromGitHub' run #1. The title bar says 'Jenkins'. The main content area is titled 'Console Output' with a green checkmark icon. On the left, there's a sidebar with links: Status, Changes, Console Output (which is selected and highlighted in grey), View as plain text, Edit Build Information, Delete build '#1', and Git Build Data. The main pane displays the build log:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/PullCodeFromGitHub
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/yankils/hello-world.git
> git init /var/lib/jenkins/workspace/PullCodeFromGitHub # timeout=10
Fetching upstream changes from https://github.com/yankils/hello-world.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/yankils/hello-world.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/yankils/hello-world.git #
timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* #
timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision aacc9fa39ba3ca8f46cf0019f35ce4511287375
(refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f aacc9fa39ba3ca8f46cf0019f35ce4511287375 # timeout=10
Commit message: "updated pom.xml file"
First time build. Skipping changelog.
Finished: SUCCESS
```

## 4. Aim: Demonstrate using Jenkins Service with Maven plugin in AWS and build a sample project.

### Source Code:

#### Step-1: Creating the aws EC2 instances:

- 1.Log in to the AWS Management Console.
- 2.Navigate to the EC2 service.
- 3.Click on “Launch Instance” to create a new EC2 instance.
- 4.Select the desired instance type, storage, and other configurations.
- 5.Configure the security group to allow inbound traffic on port 22 for SSH access.
- 6.Launch the instance and download the private key file (.pem).

The screenshots illustrate the AWS EC2 instance creation process:

- Step 1: Launch an instance** - Shows the 'Name and tags' section with 'My Web Server' as the name, and the 'Application and OS Images (Amazon Machine Image)' section.
- Step 2: Network settings** - Shows the creation of a new security group 'Launch-wizard-1'. It includes rules for 'Allow SSH traffic from Anywhere' and 'Allow HTTPS traffic from the internet'.
- Step 3: Summary** - Displays the final launch configuration with 1 instance of t2.micro, 1 volume(s) - 8 GiB, and the newly created security group.

#### Step-2:SSH into EC2 Instance, Install Java and Jenkins

SSH into the EC2 instance. After securely connecting to the instance, the first thing we have to do is to install the Java application onto the instance. Jenkins runs natively runs on the Java application so it has to be properly installed in order for the Jenkins server to be operational.

1. sudo wget -O /etc/yum.repos.d/jenkins.repo \
 https://pkg.jenkins.io/redhat-stable/jenkins.repo

```
[ec2-user@ip-172-31-13-239 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2023-12-17 07:10:03-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.154.133, 2a04:4e42:24::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.154.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.r 100%[=====]     85  --.-KB/s   in 0s

2023-12-17 07:10:03 (5.74 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-13-239 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[ec2-user@ip-172-31-13-239 ~]$ sudo yum upgrade
Jenkins-stable
Dependencies resolved.
Nothing to do.
Complete!
```

2. sudo dnf install java-17-amazon-corretto -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo dnf install java-17-amazon-corretto -y
Last metadata expiration check: 0:00:06 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
Package           Arch      Version            Repository      Size
=====
Installing:
java-17-amazon-corretto x86_64   1:17.0.9+8-1.amzn2023.1  amazonlinux   188 k
Installing dependencies:
alsa-lib              x86_64   1.2.7.2-1.amzn2023.0.2  amazonlinux   504 k
cairo                 x86_64   1.17.6-2.amzn2023.0.1  amazonlinux   684 k
dejavu-sans-fonts    noarch   2.37-16.amzn2023.0.2   amazonlinux   1.3 M
dejavu-sans-mono-fonts noarch   2.37-16.amzn2023.0.2   amazonlinux   467 k
dejavu-serif-fonts   noarch   2.37-16.amzn2023.0.2   amazonlinux   1.0 M
fontconfig             x86_64   2.13.94-2.amzn2023.0.2  amazonlinux   273 k
fonts-filesystem      noarch   1:2.0.5-12.amzn2023.0.2  amazonlinux   9.5 k
freetype               x86_64   2.13.0-2.amzn2023.0.1  amazonlinux   422 k
glib                  x86_64   5.2.1-9.amzn2023.0.1  amazonlinux   49 k
google-noto-fonts-common noarch   20201206-2.amzn2023.0.2  amazonlinux   15 k
google-noto-sans-vf-fonts noarch   20201206-2.amzn2023.0.2  amazonlinux   492 k
graphite2              x86_64   1.3.14-7.amzn2023.0.2  amazonlinux   97 k
harfbuzz               x86_64   7.0.0-2.amzn2023.0.1  amazonlinux   868 k
java-17-amazon-corretto-headless x86_64   1:17.0.9+8-1.amzn2023.1  amazonlinux   91 M
javapackages-filesystem noarch   6.0.0-7.amzn2023.0.6  amazonlinux   12 k
langpacks-core-font-en noarch   3.0-21.amzn2023.0.4   amazonlinux   10 k
libICE                 x86_64   1.0.10-6.amzn2023.0.2  amazonlinux   71 k
libSM                  x86_64   1.2.3-8.amzn2023.0.2  amazonlinux   42 k
libX11                 x86_64   1.7.2-3.amzn2023.0.4  amazonlinux   657 k
libX11-common           noarch   1.7.2-3.amzn2023.0.4  amazonlinux   152 k
libXau                 x86_64   1.0.9-6.amzn2023.0.2  amazonlinux   31 k
libXext                 x86_64   1.3.4-6.amzn2023.0.2  amazonlinux   41 k
```

3. sudo yum install jenkins -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:00:22 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
 Package          Architecture      Version       Repository      Size
=====
Installing:
jenkins          noarch          2.426.2-1.1   jenkins        85 M
Transaction Summary
=====
Install 1 Package

Total download size: 85 M
Installed size: 85 M
Downloading Packages:
jenkins-2.426.2-1.1.noarch.rpm           7.7 MB/s | 85 MB  00:11
Total                                         7.7 MB/s | 85 MB  00:11
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Installing : jenkins-2.426.2-1.1.noarch 1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Verifying   : jenkins-2.426.2-1.1.noarch 1/1

Installed:
jenkins-2.426.2-1.1.noarch
```

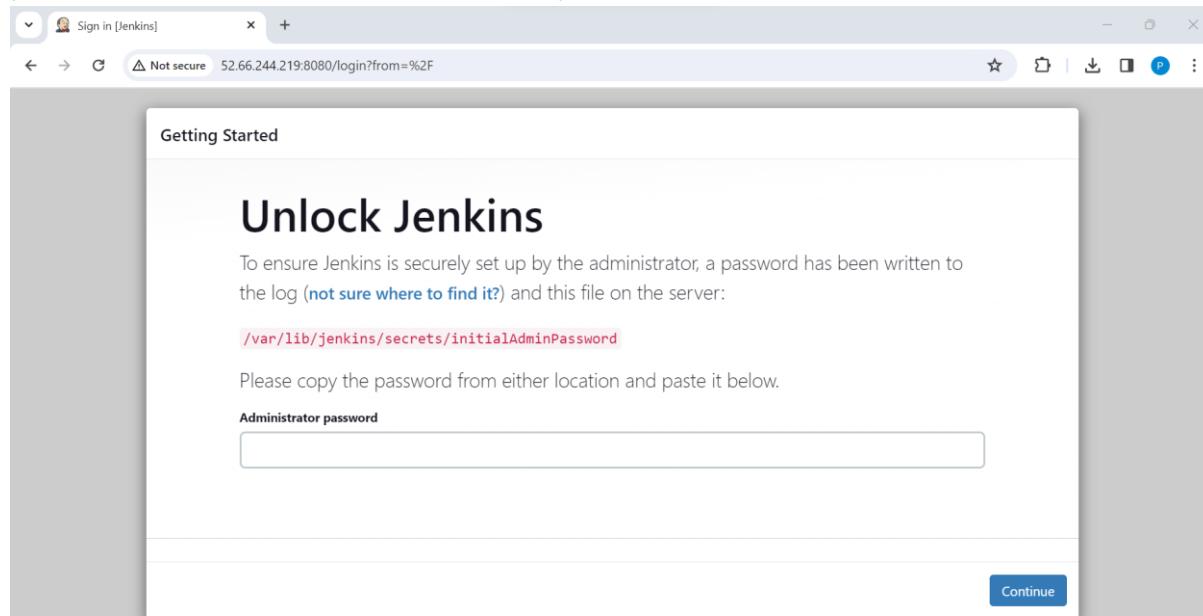
#### 4. sudo systemctl enable jenkins

```
sudo systemctl start jenkins
```

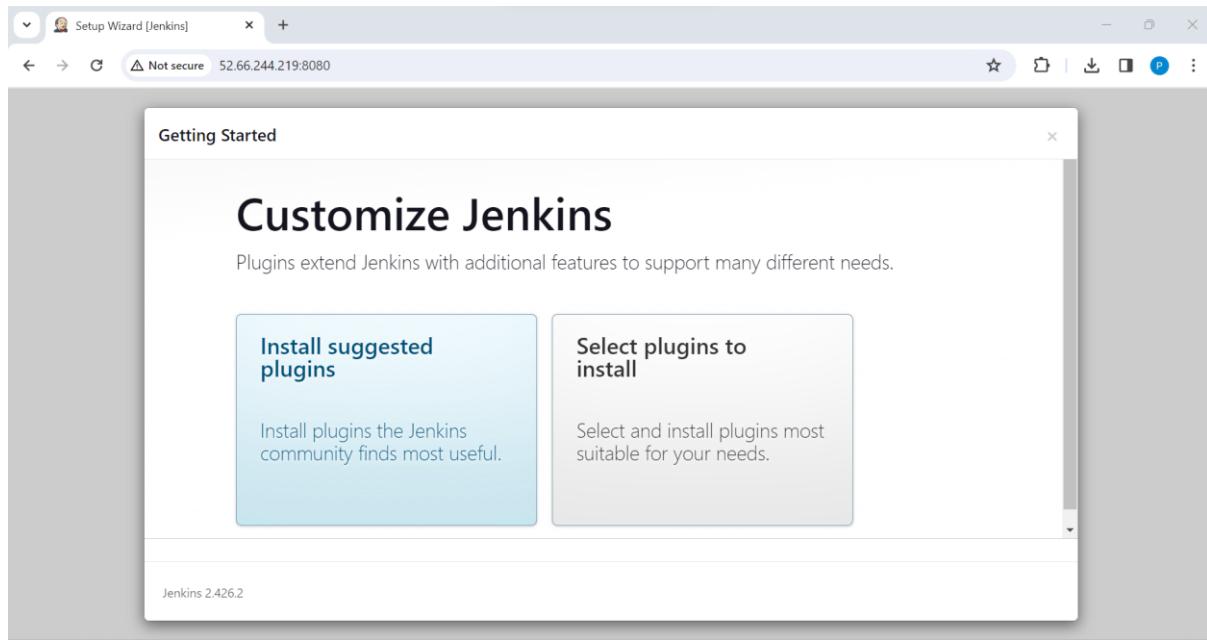
```
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl start jenkins
```

#### Step-3: Setting Up Jenkins Server

Now that the Jenkins server has been officially started on the instance, head over to the AWS Management console, grab the public IP address of the instance and paste that into your browser address bar followed by :8080



```
[ec2-user@ip-172-31-13-239 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
baaeef975c09e468a977274b59ed26a8e
[ec2-user@ip-172-31-13-239 ~]$
```



Setup an admin user. For this project we will get a dummy administrator user. In a real production you will use different credentials. Save and continue.

Username=admin

Password=Password

Full name= admin

[Email=admin@admin.com](#)

Getting Started

## Create First Admin User

Username  
admin

Password  
\*\*\*\*\*

Confirm password  
\*\*\*\*\*

Full name  
admin

E-mail address  
admin@admin.com

Jenkins 2.414.1 Skip and continue as admin Save and Continue

### Step 4: We are Live and Jenkins Ready! Time to Build

If you see the “Welcome to Jenkins” page, then you’ve successfully setup Jenkins and are ready to go to the next stage of the project, which is setting up our first Jenkins job!

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, Search bar, Notifications, User account (admin), Log out.
- Left Sidebar:**
  - Dashboard >
  - New item
  - People
  - Build History
  - Manage Jenkins
  - My Views
- Middle Content:**

### 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 →
- Bottom Left:** Build Queue (No builds in the queue)
- Bottom Center:** Build Executor Status (1 idle, 2 idle)

Creating our first jenkin job

The screenshot shows the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo, a search bar labeled "Search (%+K)", and various icons for help, security, and user management. Below the navigation bar, the left sidebar contains links for "New Item", "People", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section indicates "No builds in the queue". A "Build Executor Status" section shows 1 Idle and 2 Idle executors. The main content area features a "Welcome to Jenkins!" message, a "Start building your software project" section with a "Create a job" button (which is highlighted with a red box), and a "Set up a distributed build" section with links for "Set up an agent" and "Configure a cloud". At the bottom right, there are links for "REST API" and "Jenkins 2.414.1".

For our first job we are going to give it the name “**HelloWorldJob**” and select the option for “**Freestyle project**”. Click ok to move to the next step.

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

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

**Multi-configuration project**

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

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

**OK**



**Step-5:** Under the general configuration settings

In the description box, enter the text “HelloWorldJob”

Under the “Source Code Management” tab select “None” option

The screenshot shows the Jenkins job configuration page for 'HelloWorldJob'. The top navigation bar includes 'Dashboard', 'HelloWorldJob', and 'Configuration'. On the left, a sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions'. The 'General' tab is selected. The main configuration area has a title 'General' and a status 'Enabled' with a green toggle switch. A 'Description' field contains 'HelloWorldJob'. Below it, under 'Plain text Preview', there are several checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. An 'Advanced' dropdown menu is visible. The 'Source Code Management' section shows 'None' selected (radio button is filled) and 'Git' as an option. At the bottom are 'Save' and 'Apply' buttons.

- Navigate back to the Dashboard and you will see our 1st job in queue ready for the build stage.
- Click on “Build Now”
- After a few seconds you will see the the build output under the “build history”
- The console ouput should display a green checkmark indicating that the build was SUCCESSFUL.
- You can see that the echo “Hello World” and uptime commands were properly executed!

**Project HelloWorldJob**

- Status
- </> Changes
- Workspace
- Build Now
- Configure
- Delete Project
- Rename

Permalinks

Edit description

Disable Project

Build History trend ▾

No builds

**Console Output**

- Status
- </> Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#1'
- Next Build

```

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorldJob
[HelloWorldJob] $ /bin/sh -xe /tmp/jenkins13823842254370167628.sh
+ echo 'Hello World'
Hello World
+ uptime
16:45:33 up 37 min, 1 user, load average: 0.03, 0.03, 0.06
Finished: SUCCESS
  
```

## Step 6: Installing and Integrating Maven to Jenkins

Maven automates the process of building and managing code in both software development as well as in DevOps. Maven manages the build, testing, and packaging of code, making it easy for DevOps teams to produce consistent, reliable, and repeatable builds.

```
[root@ip-172-31-23-228 ~]# cd /opt
[root@ip-172-31-23-228 opt]# wget https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
--2023-08-28 17:18:40-- https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9336368 (8.9M) [application/x-gzip]
Saving to: 'apache-maven-3.9.4-bin.tar.gz'

100%[=====] 9,336,368 --.-K/s in 0.1s

2023-08-28 17:18:40 (81.3 MB/s) - 'apache-maven-3.9.4-bin.tar.gz' saved [9336368/9336368]
```

After installation, verify that Maven was indeed installed by list the contents of the directory: If you see the maven dependency files listed then you know it was successfully installed.

```
[root@ip-172-31-23-228 opt]# ll
total 9120
-rw-r--r-- 1 root root 9336368 Aug  3 07:56 apache-maven-3.9.4-bin.tar.gz
drwxr-xr-x  4 root root      33 Aug 22 18:26 aws
drwxr-xr-x  2 root root      6 Aug 16 2018 rh
[[root@ip-172-31-23-228 opt]# tar -xvzf apache-maven-3.9.4-bin.tar.gz
apache-maven-3.9.4/README.txt
apache-maven-3.9.4/LICENSE
apache-maven-3.9.4/NOTICE
apache-maven-3.9.4/lib/
apache-maven-3.9.4/lib/aopalliance.license
apache-maven-3.9.4/lib/commons-cli.license
apache-maven-3.9.4/lib/commons-codec.license
apache-maven-3.9.4/lib/commons-lang3.license
apache-maven-3.9.4/lib/failureaccess.license
apache-maven-3.9.4/lib/guava.license
apache-maven-3.9.4/lib/guice.license
apache-maven-3.9.4/lib/httpclient.license
apache-maven-3.9.4/lib/httpcore.license
apache-maven-3.9.4/lib/jansi.license
apache-maven-3.9.4/lib/javax.annotation-api.license
apache-maven-3.9.4/lib/javax.inject.license
apache-maven-3.9.4/lib/jcl-over-slf4j.license
apache-maven-3.9.4/lib/org.eclipse.sisu.inject.license
apache-maven-3.9.4/lib/org.eclipse.sisu.plexus.license
apache-maven-3.9.4/lib/plexus-cipher.license
apache-maven-3.9.4/lib/plexus-component-annotations.license
apache-maven-3.9.4/lib/plexus-interpolation.license
apache-maven-3.9.4/lib/plexus-sec-dispatcher.license
apache-maven-3.9.4/lib/plexus-utils.license
apache-maven-3.9.4/lib/slf4j-api.license
apache-maven-3.9.4/boot/
apache-maven-3.9.4/boot/plexus-classworlds.license
apache-maven-3.9.4/lib/jansi-native/
apache-maven-3.9.4/lib/jansi-native/Windows/
apache-maven-3.9.4/lib/jansi-native/Windows/x86/
apache-maven-3.9.4/lib/jansi-native/Windows/x86_64/
apache-maven-3.9.4/lib/jansi-native/Windows/x86/jansi.dll
apache-maven-3.9.4/lib/jansi-native/Windows/x86_64/jansi.dll
apache-maven-3.9.4/bin/m2.conf
apache-maven-3.9.4/bin/mvn.cmd
apache-maven-3.9.4/bin/mvnDebug.cmd
apache-maven-3.9.4/bin/mvn
apache-maven-3.9.4/bin/mvnDebug
```

You can choose to keep the default maven file name that is highlighted in blue or change the name of the file. I'm going to change the name to just "maven" to keep things simple. To change the name use the "mv" command.

```
[root@ip-172-31-23-228 opt]# mv apache-maven-3.9.4 maven
[root@ip-172-31-23-228 opt]# ll
total 9120
-rw-r--r-- 1 root root 9336368 Aug  3 07:56 apache-maven-3.9.4-bin.tar.gz
drwxr-xr-x 4 root root      33 Aug 22 18:26 aws
drwxr-xr-x 6 root root     99 Aug 28 17:20 maven
drwxr-xr-x 2 root root      6 Aug 16 2018 rh
[root@ip-172-31-23-228 opt]# cd maven
[root@ip-172-31-23-228 maven]# ll
total 36
drwxr-xr-x 2 root root    97 Aug 28 17:20 bin
drwxr-xr-x 2 root root    76 Aug 28 17:20 boot
drwxr-xr-x 3 root root   63 Jul 26 09:37 conf
drwxr-xr-x 4 root root  4096 Aug 28 17:20 lib
-rw-r--r-- 1 root root 18945 Jul 26 09:37 LICENSE
-rw-r--r-- 1 root root  5034 Jul 26 09:37 NOTICE
-rw-r--r-- 1 root root  2533 Jul 26 09:37 README.txt
```

## Execute Maven server

Check maven version

`mvn -v`

```
...c2-54-189-134-221.us-west-2.compute.amazonaws.com ...4-189-134-221.us-west-2.compute.amazonaws.com +
[[root@ip-172-31-23-228 ~]# mvn -v
Apache Maven 3.9.4 (dfbb324ad4a7c8fb0bf182e6d91b0ae20e3d2dd9)
Maven home: /opt/maven
Java version: 11.0.20, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.20.0.8
-1.amzn2.0.1.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.186-179.751.amzn2.x86_64", arch: "amd64", family: "unix"
[root@ip-172-31-23-228 ~]# ]
```

## Step-7: Maven Plugin for Jenkins:

- With Maven server installed, it is time to setup the Maven plugin in the Jenkins server to complete the integration
- Navigate back to the Jenkins Dashboard, select “Manage Jenkins” option, and select “Plugins”.

The screenshot shows the Jenkins 'Manage Jenkins' page. At the top, there's a navigation bar with links for 'Dashboard', 'Manage Jenkins', 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected and highlighted in grey), and 'My Views'. A search bar at the top right contains the placeholder 'Search settings'.

A yellow banner at the top right provides a security warning: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)". It includes three buttons: 'Set up agent', 'Set up cloud', and 'Dismiss'.

The main content area is divided into two main sections: 'System Configuration' and 'Security'.

**System Configuration:**

- System**: Configure global settings and paths.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. This item is highlighted with a red box.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Tools**: Configure tools, their locations and automatic installers.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Security:**

- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credential Providers**: Configure the credential providers and types.
- Credentials**: Configure credentials.
- Users**: Create/delete/modify users that can log in to this Jenkins.

- Type “Maven” and select Maven Integration. Click on install the plugin.

The screenshot shows the Jenkins Plugins management interface. On the left, there's a sidebar with links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area is titled 'Plugins' and has a search bar at the top with the query 'maven'. Below the search bar, there are buttons for 'Install' and 'Available'. A table lists the 'Maven Integration' plugin, version 3.23, under the 'Released' category. The plugin description states it provides deep integration between Jenkins and Maven, adding support for automatic triggers and automated configuration of various Jenkins publishers. The 'Install' button for this plugin is highlighted.

This screenshot is identical to the one above, showing the Jenkins Plugins management interface with the 'Maven Integration' plugin selected for installation.

Under Manage Jenkins, select the “Global Configuration” tab. Since we successfully installed JDK and Maven, you will see the options referencing JDK and Maven.

Name-java-11

JAVA\_HOME= /usr/lib/jvm/java-11-openjdk-11.0.20.0.8-1amzn2.0.1x86\_64 (Java home directory)

Deselect the “install automatically” option. We already have Java installed.

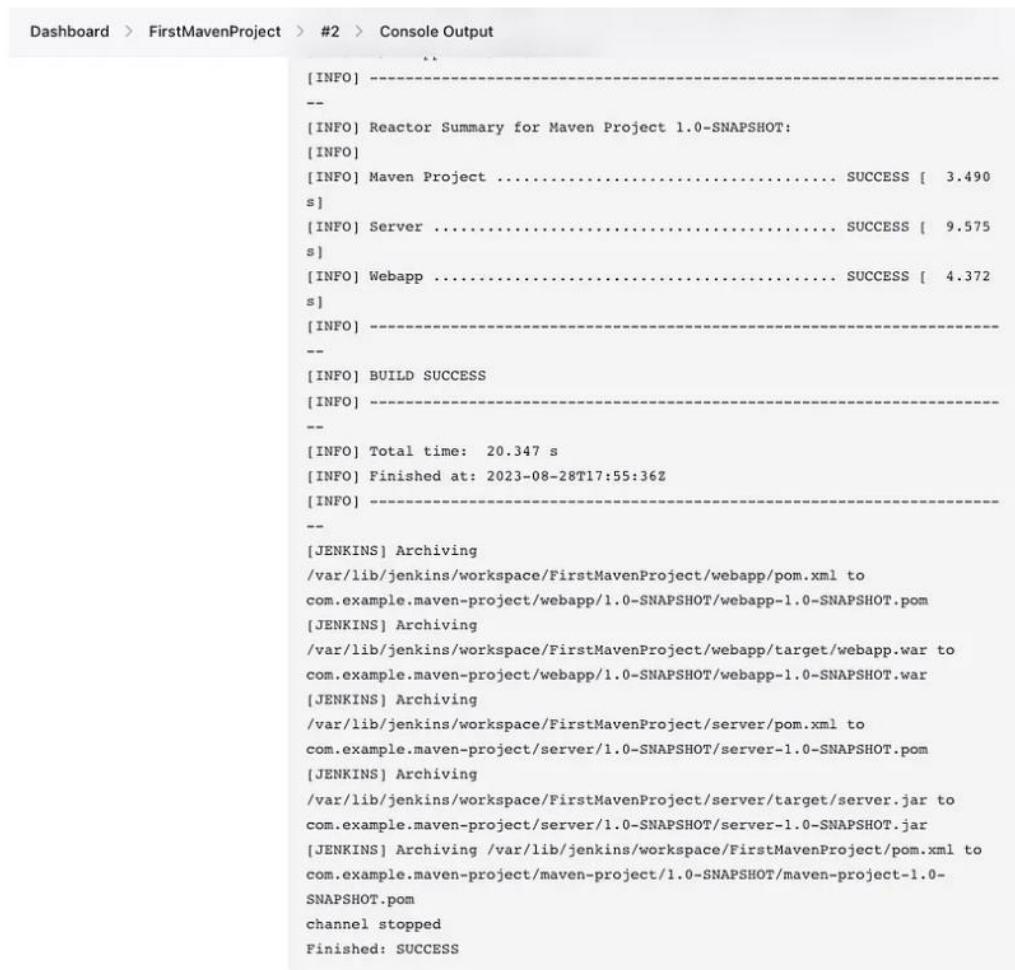
**Step-8:** Navigate to the Jenkins dashboard, select “create item”. We are going to test our CI/CD build by setting our first Maven project.

- Type “FirstMavenProject” in the name box
- Select “Maven project”. Click ok.
- Enter a description for the project build.
- Select “Git” under the Source Code Management. Copy and paste the following git repo URL into the box: <https://github.com/GeorgeBaidooJr9/hello-world>
- Under Build “Goal and Options” type “clean install”. This will result in clean copy of the git repository onto Maven.
- Apply and save

Let’s OFFICIALLY TEST THE BUILD!

- Select “Build” under configure. This build will take a few minutes since there is lots of data stored in the repo.

- After a few minutes you will see details of the build.



The screenshot shows the Jenkins console output for build #2 of the 'FirstMavenProject'. The output is as follows:

```
Dashboard > FirstMavenProject > #2 > Console Output

[INFO] -----
-- 
[INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
[INFO]
[INFO] Maven Project ..... SUCCESS [ 3.490 s]
[INFO] Server ..... SUCCESS [ 9.575 s]
[INFO] Webapp ..... SUCCESS [ 4.372 s]
[INFO] -----
-- 
[INFO] BUILD SUCCESS
[INFO] -----
-- 
[INFO] Total time: 20.347 s
[INFO] Finished at: 2023-08-28T17:55:36Z
[INFO] -----
-- 
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/webapp/pom.xml to
com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war to
com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/server/pom.xml to
com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/server/target/server.jar to
com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/pom.xml to
com.example.maven-project/maven-project/1.0-SNAPSHOT/maven-project-1.0-
SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

## 5. Deploy a Java web app with Github, Jenkins, Maven, Tomcat on AWS.

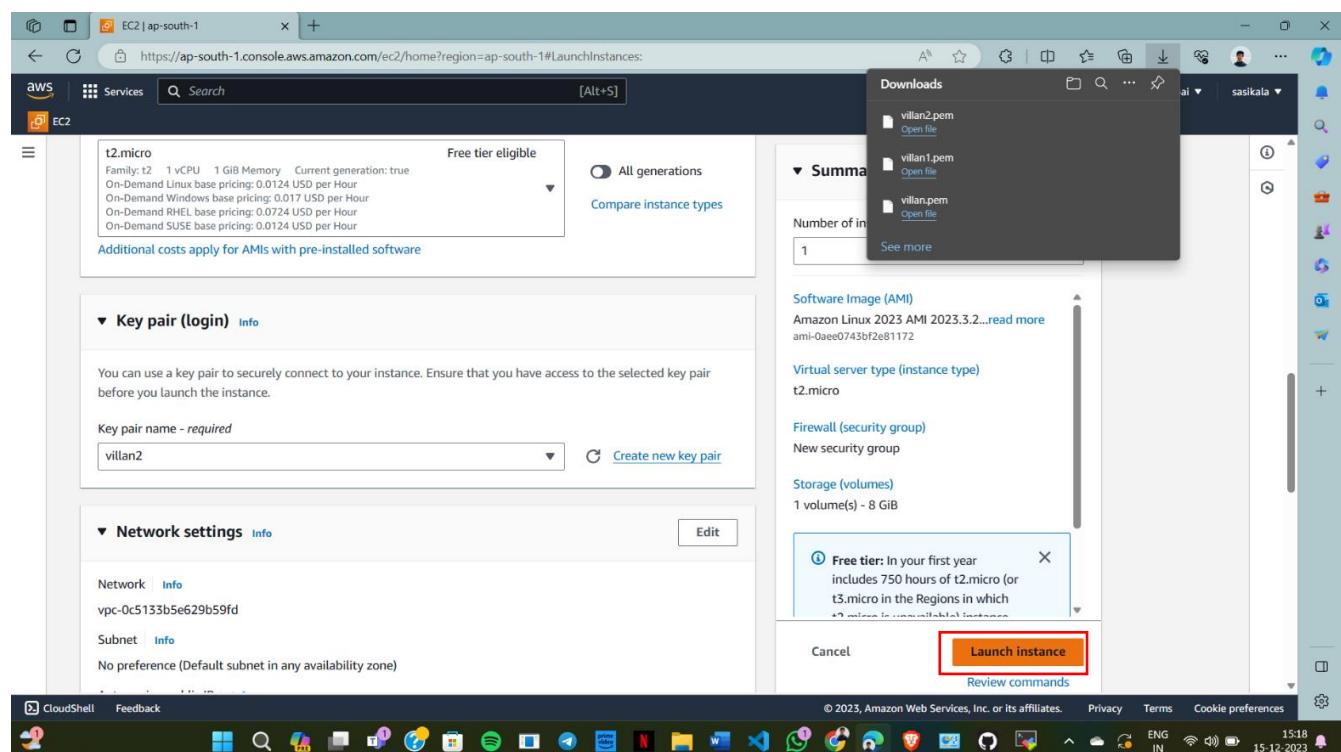
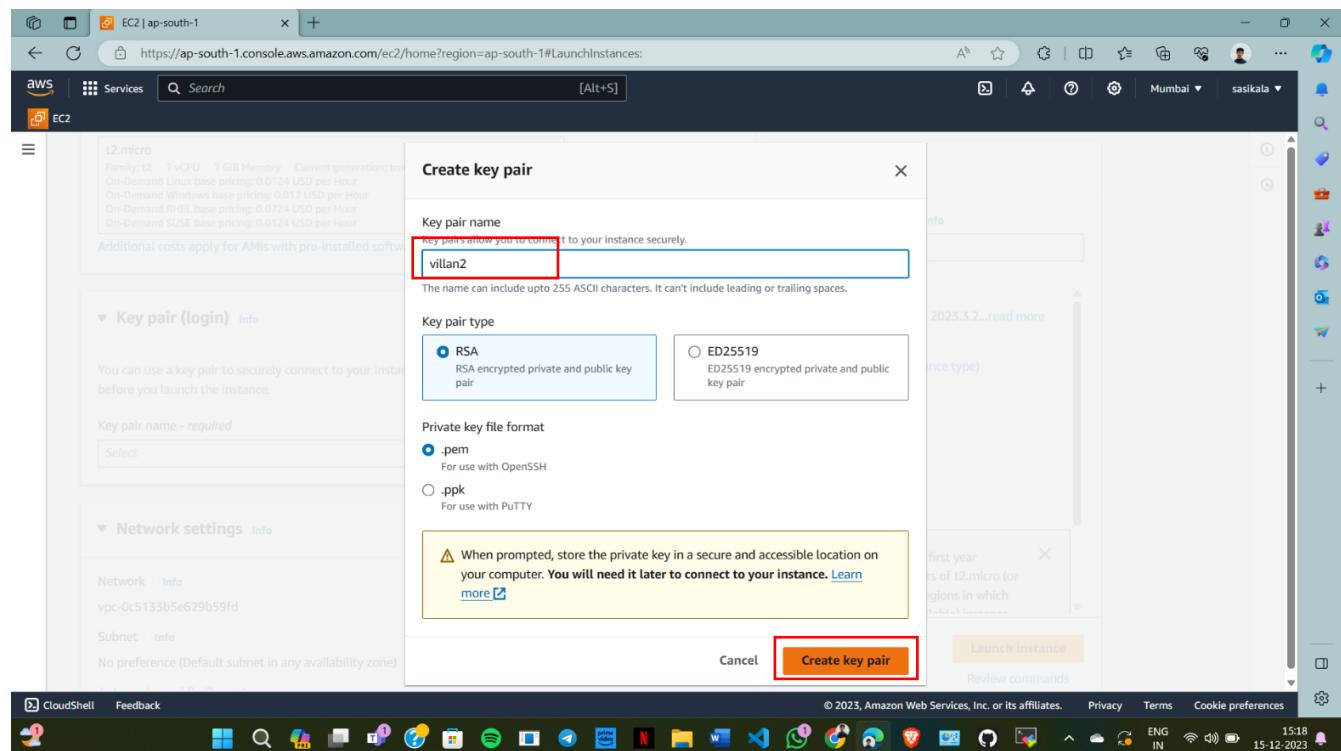
### Source Code:-

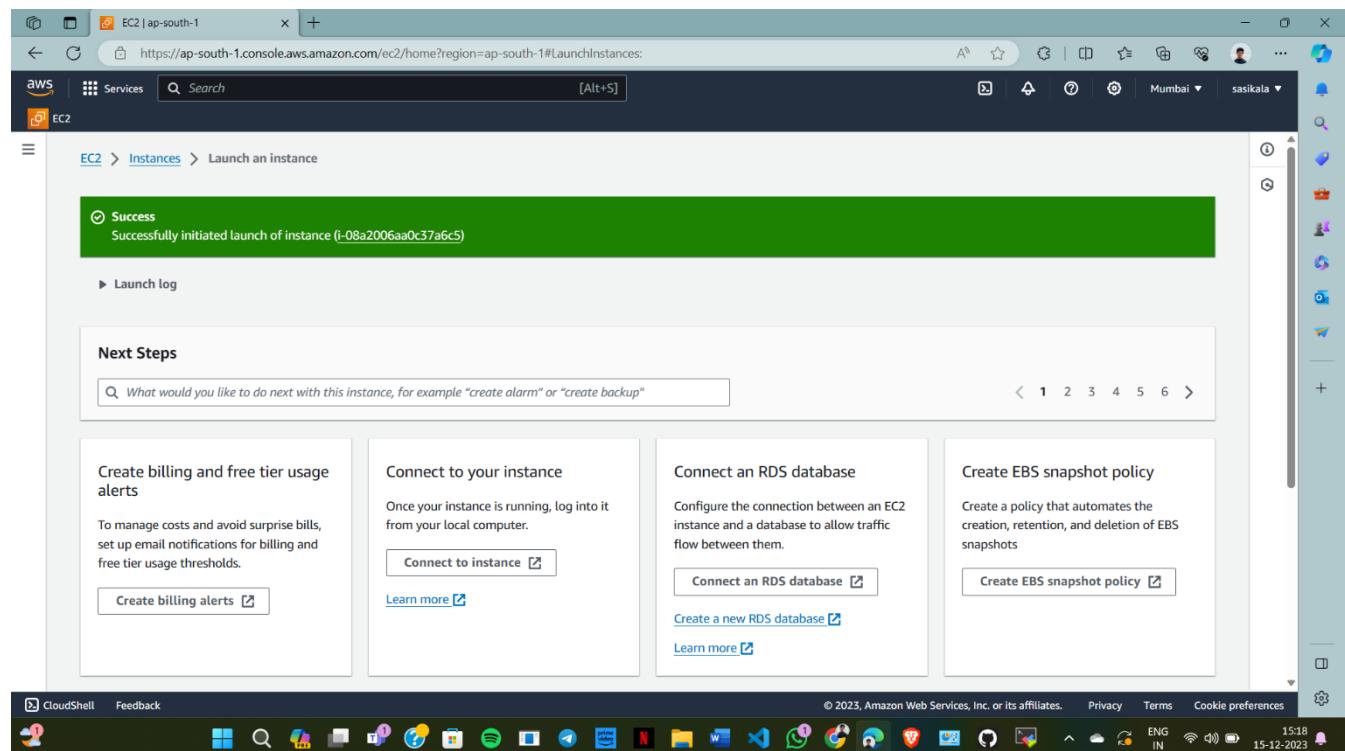
**Step1:** Logon to your AWS management console from your web browser.

The screenshot shows the AWS Management Console search results for 'ec2'. The EC2 service card is highlighted with a red box. The card displays the service name 'EC2' with a star icon, the description 'Virtual Servers in the Cloud', and links for 'Dashboard', 'Launch templates', 'Instances', 'Spot Instance requests', and 'Savings plans'. Below this, there are cards for 'EC2 Image Builder', 'Recycle Bin', and 'Amazon Inspector'.

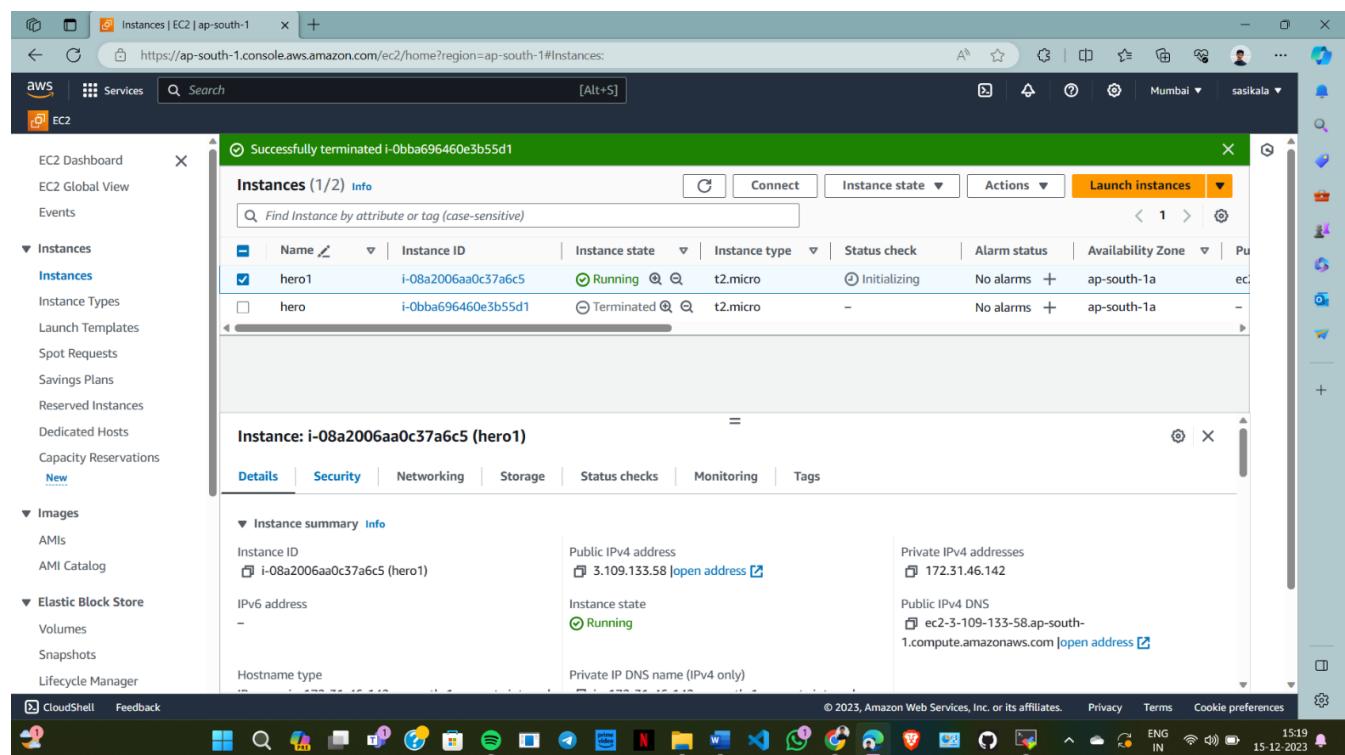
**Step-2:** Search for EC2 and create a Linux EC2 instance on your AWS management console by clicking on **launch instance**. Take a secured PEM key pair and configure the rest of additional configurations of instance as per your requirement. Later, click on '**launch instance**' to launch your EC2 instance.

The screenshot shows the AWS EC2 Dashboard. On the left, a navigation pane includes 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Instances' (selected), 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images' (selected), 'AMIs', 'AMI Catalog', and 'Elastic Block Store' (selected). The main area displays 'Resources' statistics and a 'Launch instance' button, which is highlighted with a red box. The 'Launch instance' button has a tooltip: 'To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.' Below it are buttons for 'Migrate a server' and a note: 'Note: Your instances will launch in the Asia Pacific (Mumbai) Region'. To the right, there are sections for 'EC2 Free Tier' (info), 'Service health' (AWS Health Dashboard), 'Offer usage (monthly)', and 'Storage space on EBS'.





**Step-3:** After successfully launching the instance, edit the instance's security inbound rules by clicking on **security** and then **security groups**.



Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/> hero1	i-08a2006aa0c37a6c5	Running	t2.micro	Initializing	No alarms	ap-south-1a
<input type="checkbox"/> hero	i-0bba696460e3b55d1	Terminated	t2.micro	-	No alarms	ap-south-1a

Instance: i-08a2006aa0c37a6c5 (hero1)

Security details

IAM Role: - Owner ID: 085058735202 Launch time: Fri Dec 15 2023 15:18:42 GMT+0530 (India Standard Time)

Security groups: sg-07df1b84434509139 (launch-wizard-12)

Inbound rules

Step-4: Click on **Edit inbound rules**. Add a custom TCP type and select **8080** as Port Range. Save these changes and copy the instance's public IPV4 address.

EC2 > Security Groups > sg-07df1b84434509139 - launch-wizard-12

Details

Security group name: launch-wizard-12	Security group ID: sg-07df1b84434509139	Description: launch-wizard-12 created 2023-12-15T09:47:58.007Z	VPC ID: vpc-0c5133b5e629b59fd
Owner: 085058735202	Inbound rules count: 1 Permission entry	Outbound rules count: 1 Permission entry	

Inbound rules | Outbound rules | Tags

Inbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range

Manage tags | Edit inbound rules

The screenshot displays the AWS EC2 console interface. The top window, titled 'Edit inbound rules', shows the configuration of a security group rule. A new rule is being added for port 8080 (Custom TCP) with an 'Any...' source. The 'Save rules' button is highlighted with a red box. A warning message at the bottom left cautions against using 0.0.0.0/0 or ::/0 as sources. The bottom window shows a success message: 'Inbound security group rules successfully modified on security group sg-07df1b84434509139 - launch-wizard-12'. It also displays the updated security group details, including the name 'launch-wizard-12', ID 'sg-07df1b84434509139', and a description 'launch-wizard-12 created 2023-12-15T09:47:58.007Z'. The 'Inbound rules' tab is selected, showing two entries: port 22 (SSH) and port 8080 (Custom TCP).

The screenshot shows the AWS EC2 Instances page. A green banner at the top indicates "Successfully terminated i-0bba696460e3b55d1". The main table lists two instances: "hero1" (Running, t2.micro, Initializing) and "hero" (Terminated, t2.micro). The "hero1" row is selected. Below the table, the "Details" tab is active for "hero1". A red box highlights the "Public IPv4 address copied" link next to the IP address "3.109.133.58". To the right, the instance's private IP (172.31.46.142), public DNS (ec2-3-109-133-58.ap-south-1.compute.amazonaws.com), and port (open address) are listed.

Step-5: Download and open MobaXTerm. Select **SSH terminal** and connect the SSH terminal to your instance by using public IP of instance and it's private (PEM) key.

The screenshot shows the MobaXterm application interface. On the left, the "Sessions" panel shows a session titled "3.109.133.58 (ec2-user)". The main window displays the MobaXterm Personal Edition v23.5 welcome message, which includes information about Direct SSH, SSH compression, SSH-browser, and X11-forwarding. Below this, the terminal window shows the connection details: "SSH session to ec2-user@3.109.133.58" and the Amazon Linux 2023 prompt "[ec2-user@ip-172-31-46-142 ~]\$". The status bar at the bottom indicates an unregistered version.

Step-6: Run the **\$ sudo su -** command to enter “super user” mode. Now run the following commands to install and start Jenkins server:

```
[root@ip-172-31-46-142 ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
[root@ip-172-31-46-142 ~]# sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-172-31-46-142 ~]# yum install jenkins
[root@ip-172-31-38-223 ~]# yum install git
[root@ip-172-31-46-142 ~]# service jenkins start
```

The screenshot shows a terminal window in MobaXterm with the following content:

```
Authenticating with public key "Imported-OpenSSH-Key"
  • MobaXterm Personal Edition v23.5 •
  (SSH client, X server and network tools)

  ▶ SSH session to ec2-user@3.109.133.58
    • Direct SSH : ✓
    • SSH compression : ✓
    • SSH-browser : ✓
    • X11-forwarding : ✘ (disabled or not supported by server)

  ▶ For more info, ctrl+c click on help or visit our website.

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-46-142 ~]$ sudo su -
[root@ip-172-31-46-142 ~]# sudo yum install java-17-amazon-corretto-devel
Last metadata expiration check: 0% complete on Fri Dec 15 09:49:28 2023.
Dependencies resolved.

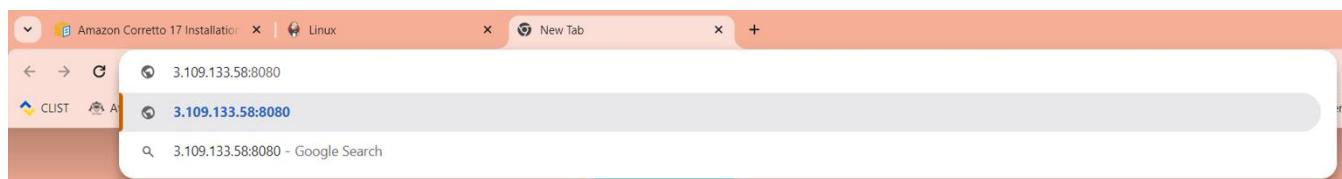
Package          Architecture Version      Repository  Size
=====
Installing:
java-17-amazon-corretto-devel x86_64       1:17.0.9+8-1.amzn2023.1  amazonlinux  149 k
Installing dependences:
alsa-lib           x86_64       1.2.7.2-1.amzn2023.0.2  amazonlinux  504 k
cairo              x86_64       1.17.6-2.amzn2023.0.1  amazonlinux  684 k
dejavu-sans-fonts noarch      2.37-16.amzn2023.0.2   amazonlinux  1.3 M
dejavu-sans-mono-fonts noarch      2.37-16.amzn2023.0.2   amazonlinux  467 K
dejavu-sig-fonts  noarch      2.37-16.amzn2023.0.2   amazonlinux  1.0 M
fontconfig         x86_64       2.13.0-2.amzn2023.0.2  amazonlinux  273 K
fonts-filesystem  noarch      1:2.0.5-12.amzn2023.0.2  amazonlinux  9.5 k
freetype            x86_64       2.13.0-2.amzn2023.0.1  amazonlinux  422 k

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

The terminal shows the user installing Java 17 using yum. The package list table includes Java, ALSA, Cairo, DejaVu fonts, and Fontconfig.

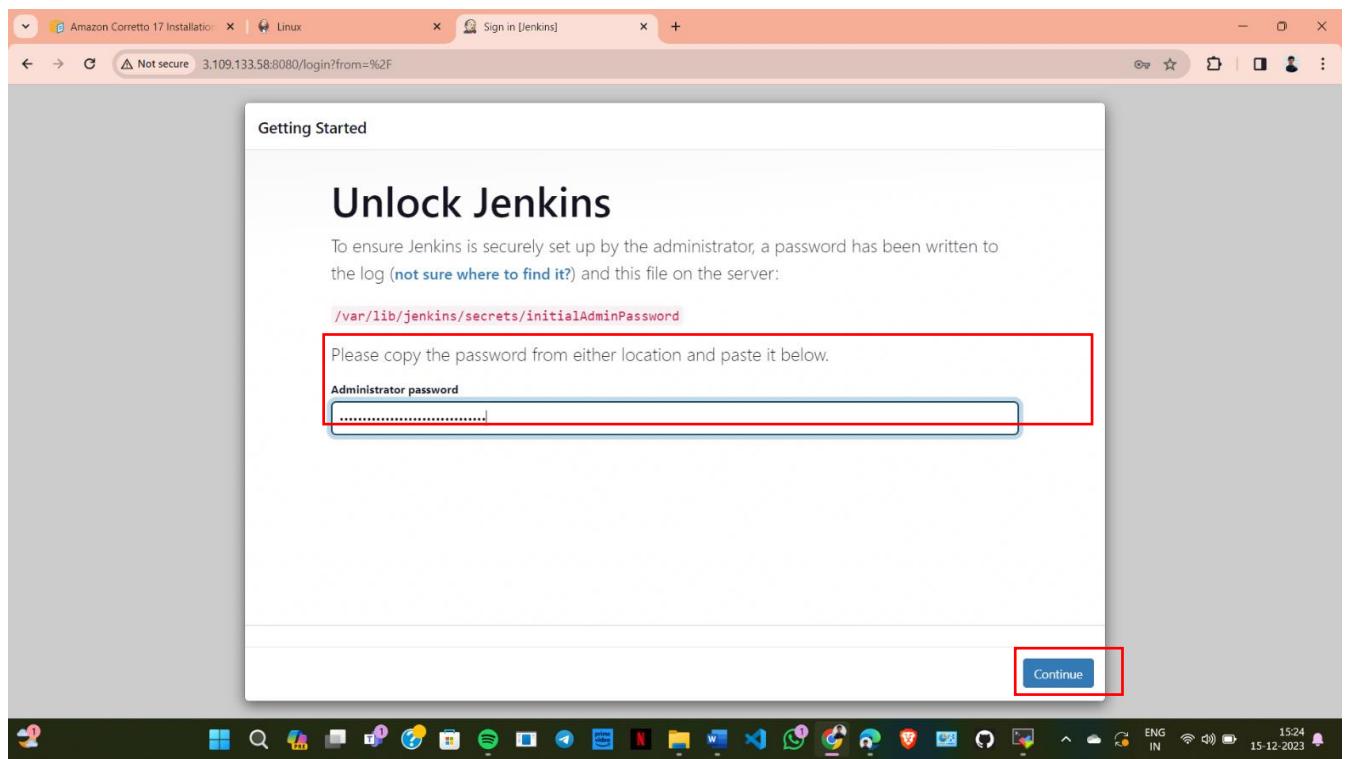
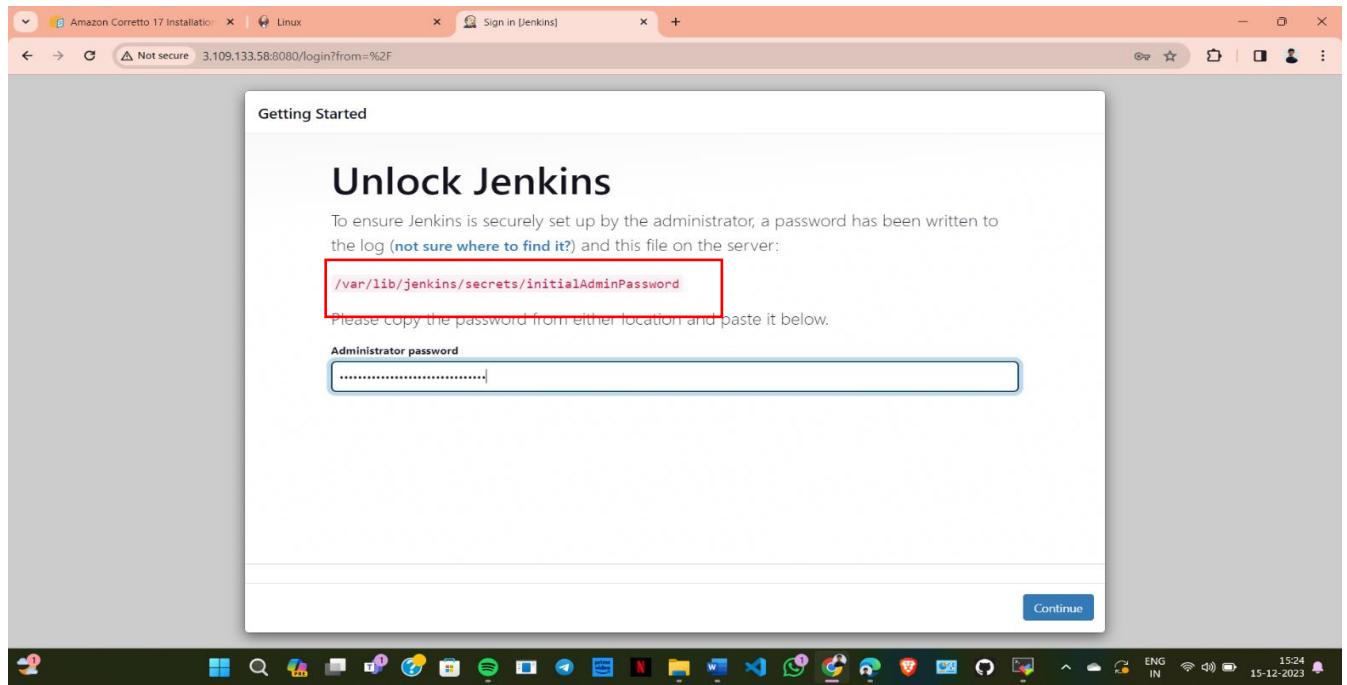
Step-7: To open Jenkins, paste the EC2 linux instance's public IP along with your port range(8080) on your browser tab.

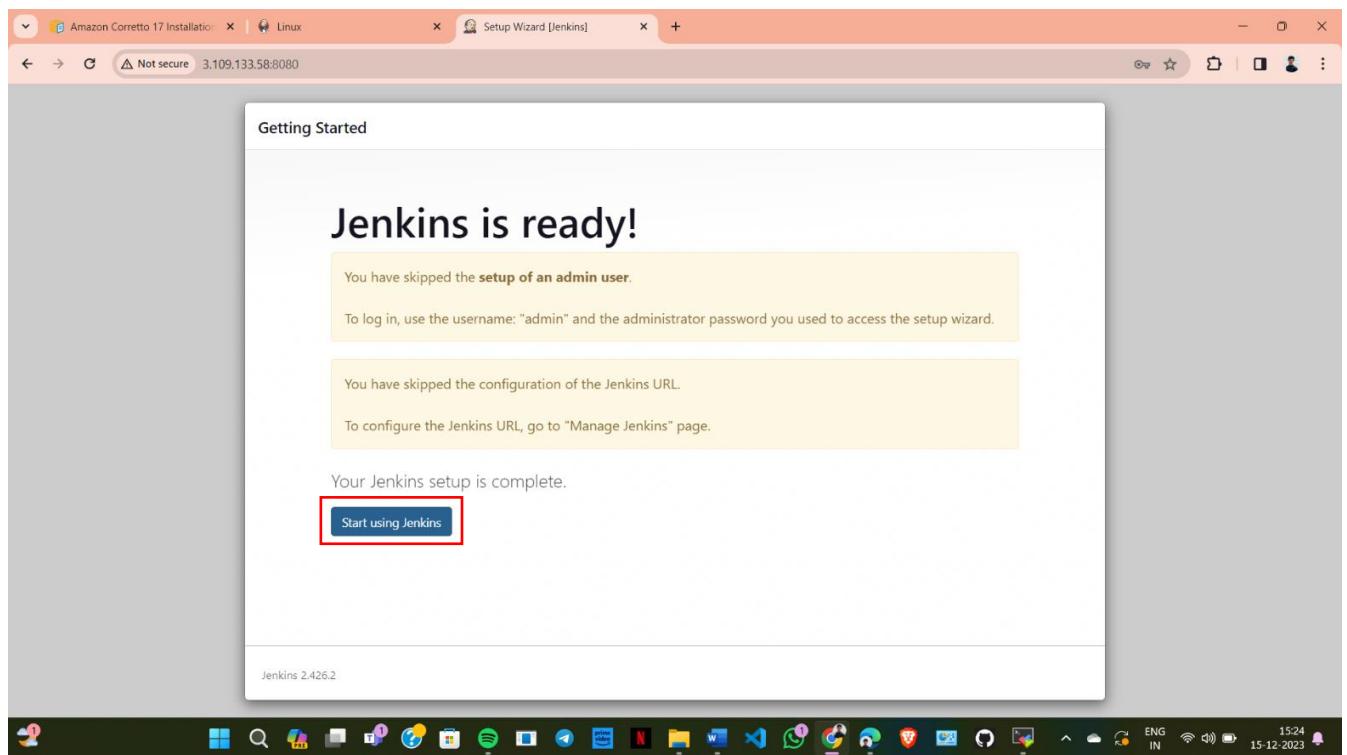
**3.109.133.58:8080**



Step-8: To get the password for opening Jenkins, copy the displaying url and run the following command on your SSH terminal

```
[root@ip-172-31-46-142 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
24b19efd4b2e43a69b59ca4b2d97d619
```

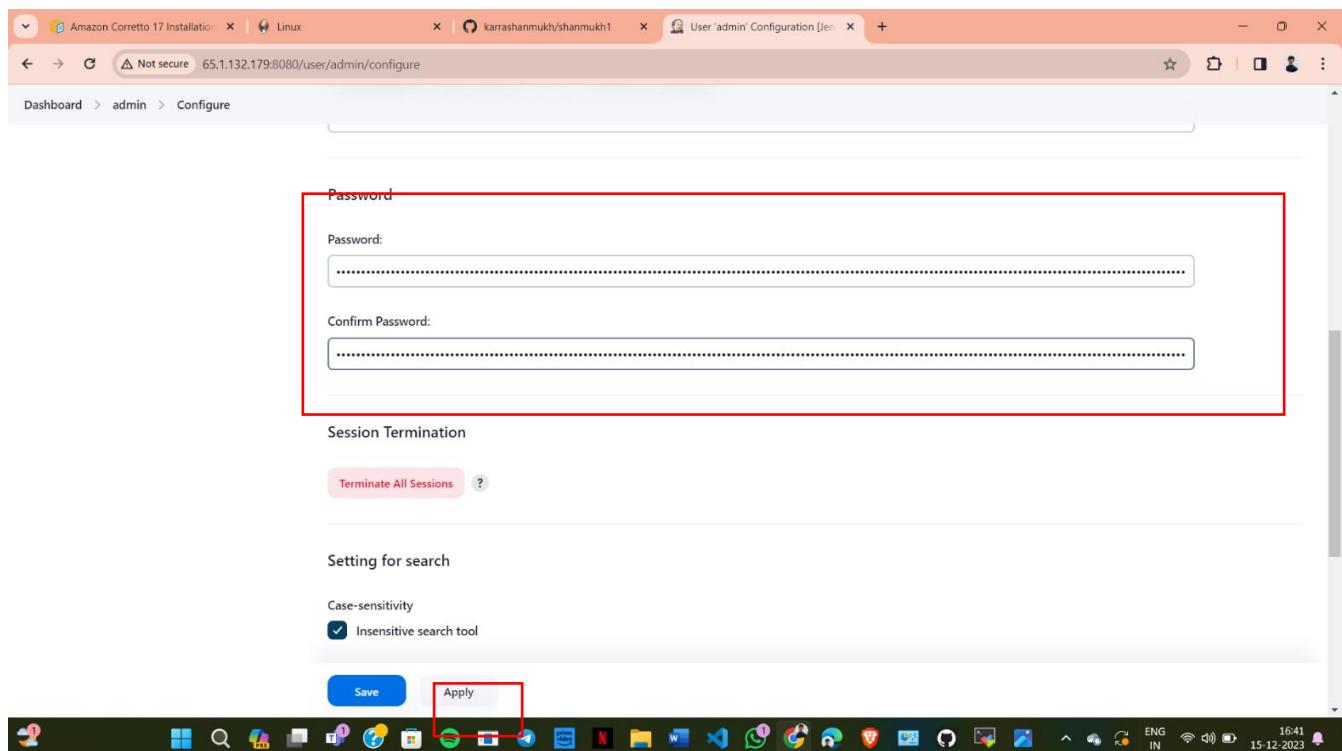




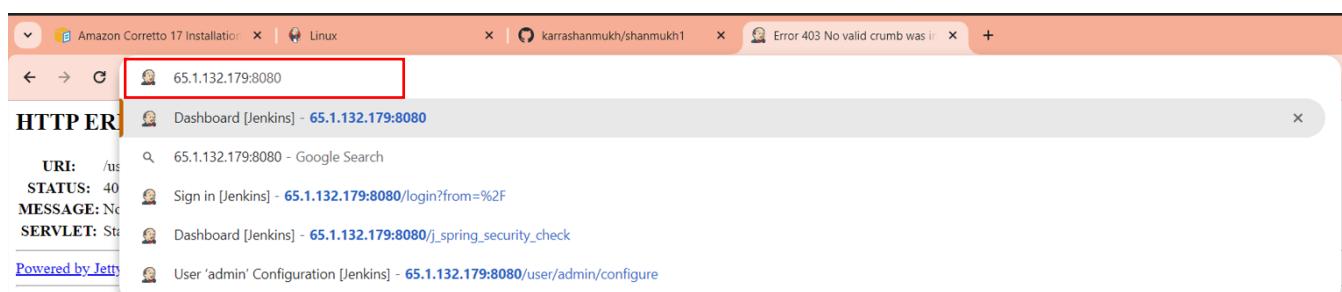
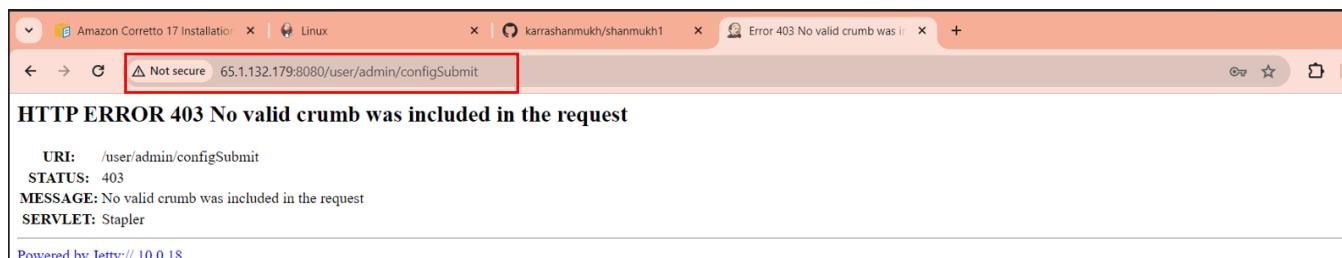
Step- 9: Now in Jenkins, goto **configure** settings and change the account password. Apply and save the changes.

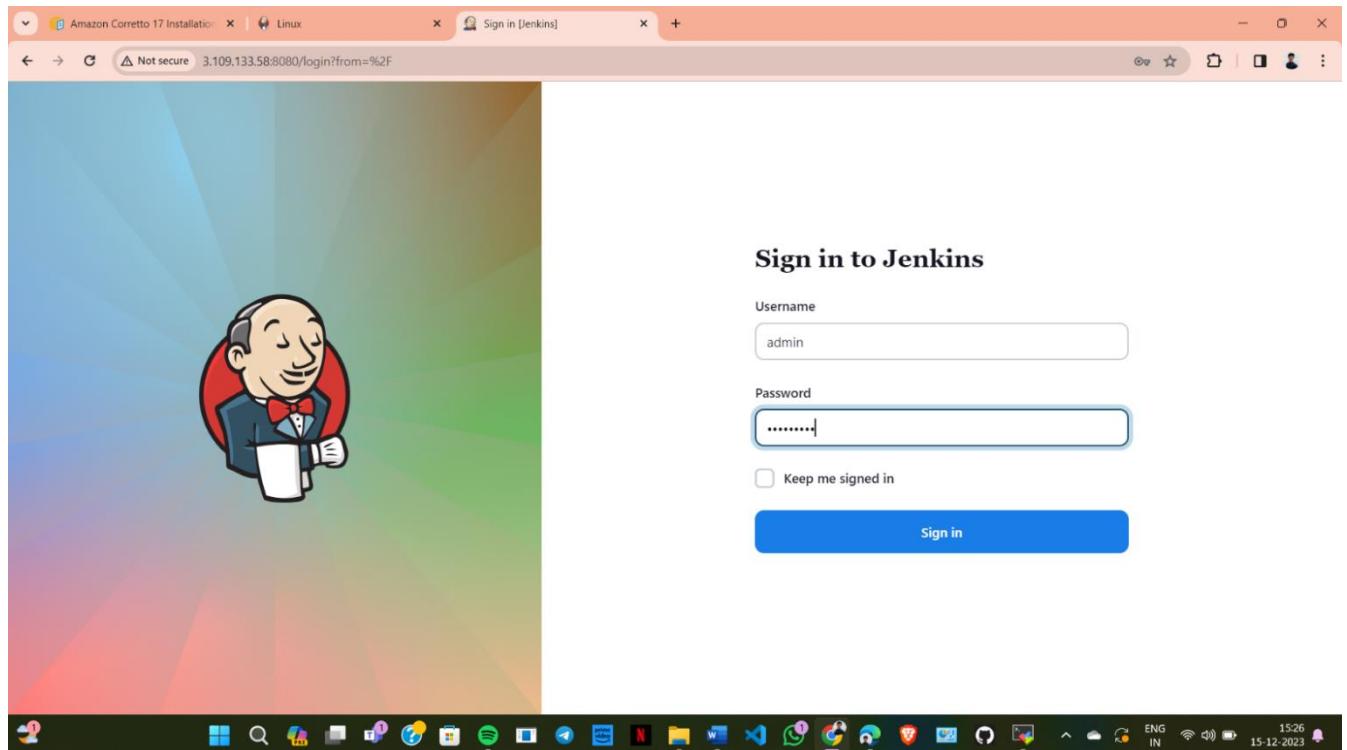
**3.109.133.58:8080**

The screenshot shows the Jenkins dashboard. The top navigation bar has a dropdown menu with "Configure" highlighted by a red box. The dashboard includes sections for "Welcome to Jenkins!", "Start building your software project", and "Set up a distributed build". On the left, there are links for "New Item", "People", "Build History", "Manage Jenkins", and "My Views". Below these are "Build Queue" and "Build Executor Status" sections. The status bar at the bottom indicates the URL is 3.109.133.58:8080/user/admin/configure.

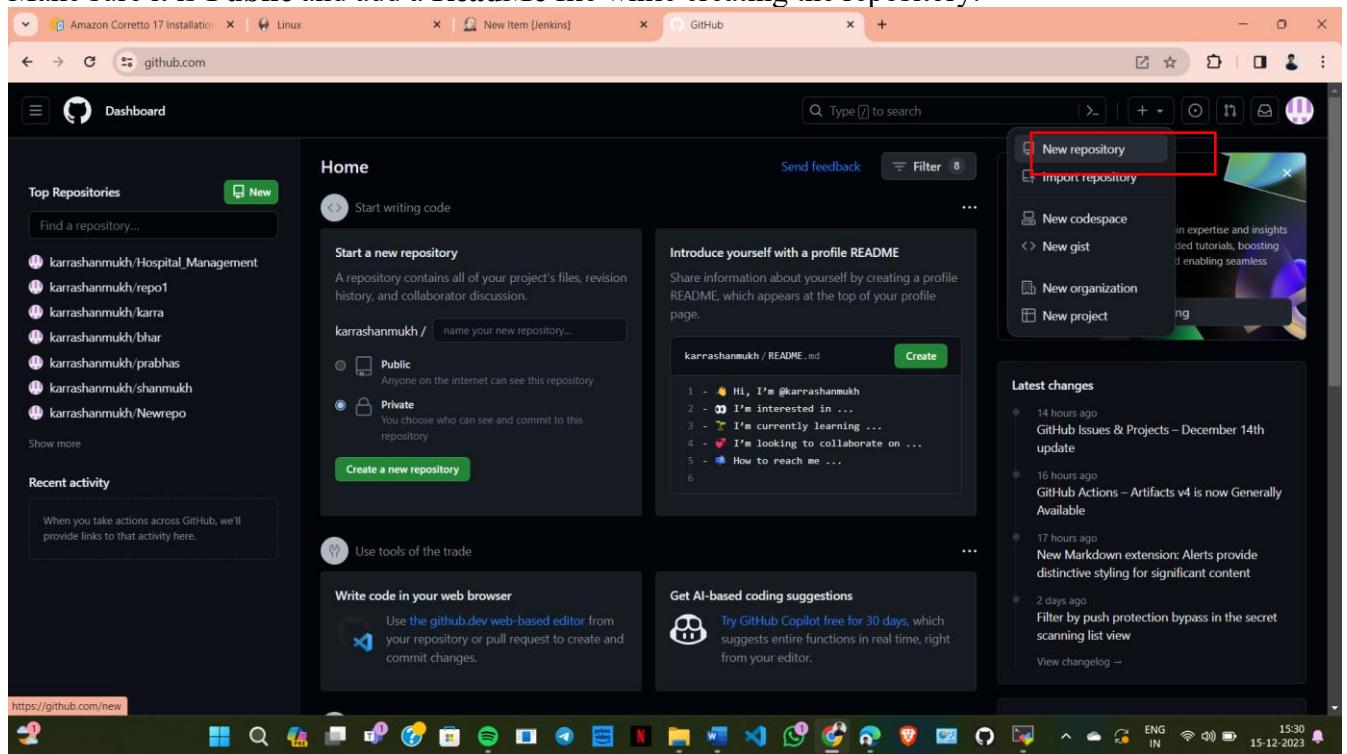


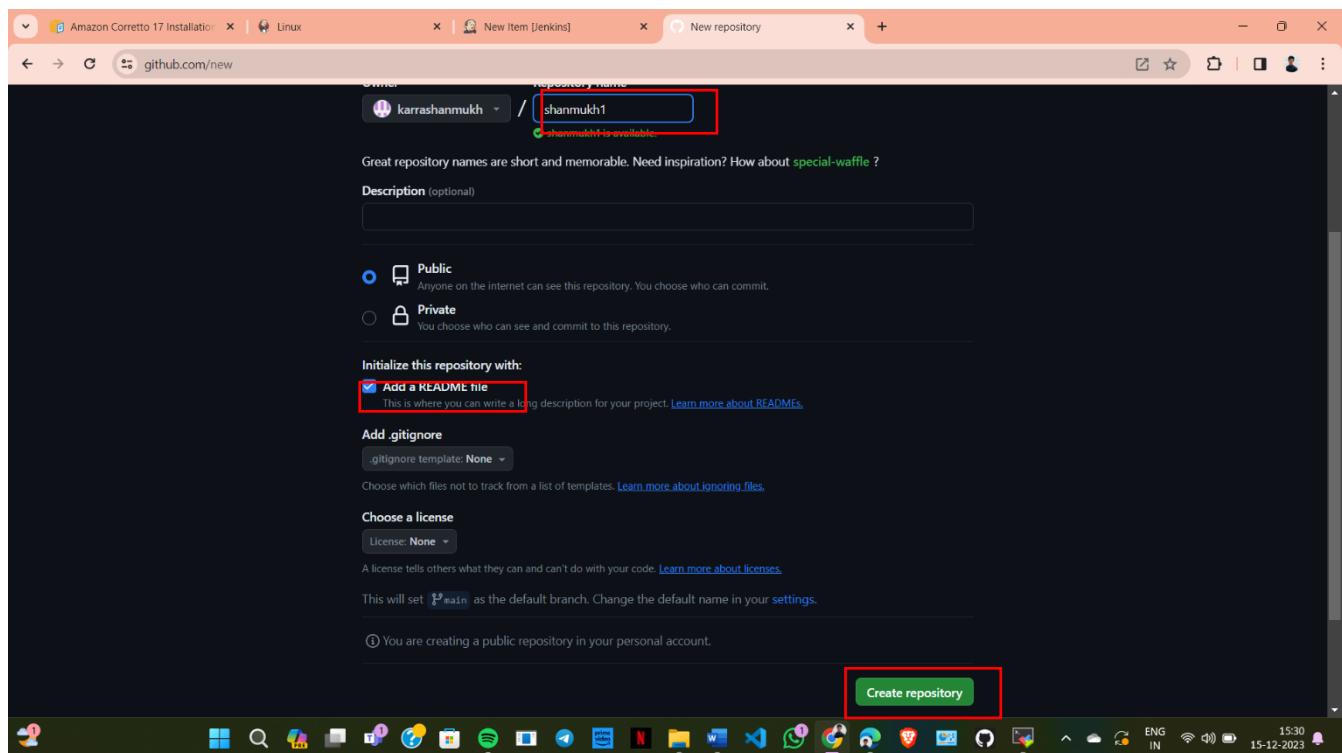
Step-10: The site now shows error after changing the password. Clear the web url and re-enter the public IP of EC2 instance along with port range(8080). Now you will be asked to enter username and password. Use **admin** as username and provide your changed password.



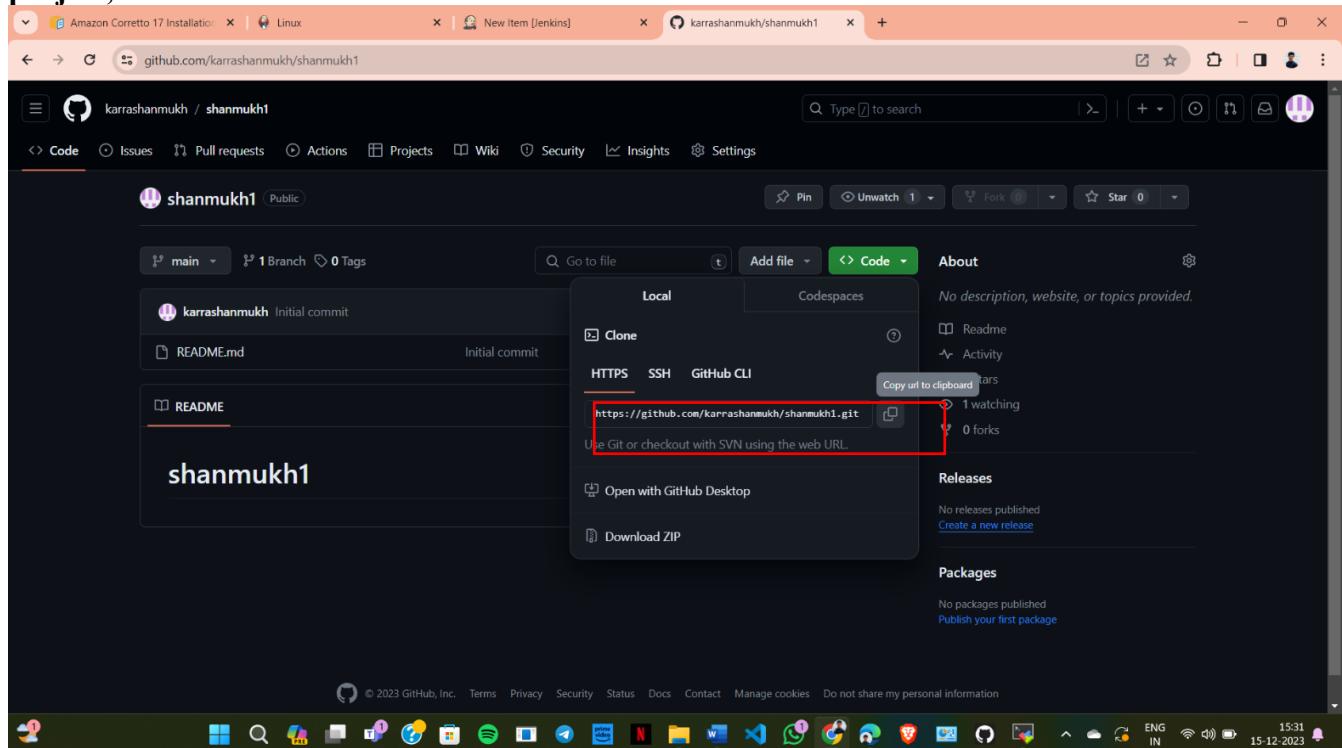


Step-11: Now open and logon to your GitHub account. Create a new repository and give it a name. Make sure it is **Public** and add a **ReadMe** file while creating the repository.





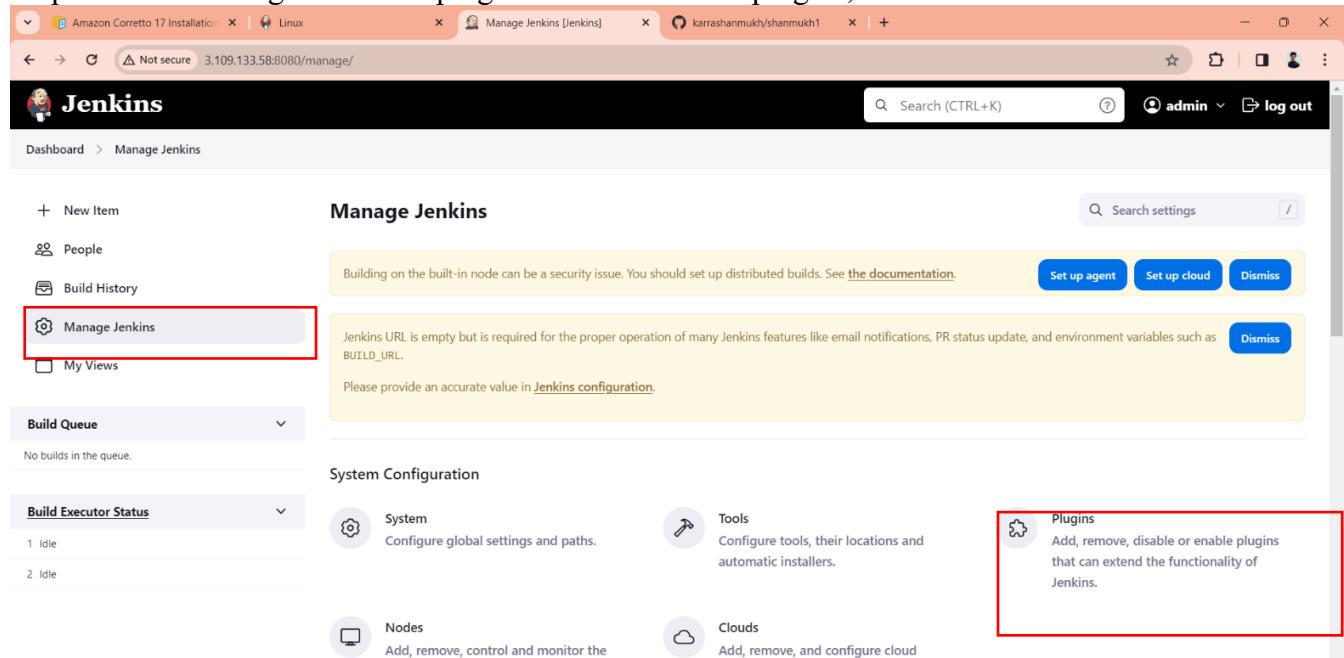
Step-12: Copy the repository's url. Goto Jenkins and click on new item, give it a name, select **free style project**, and click on OK.



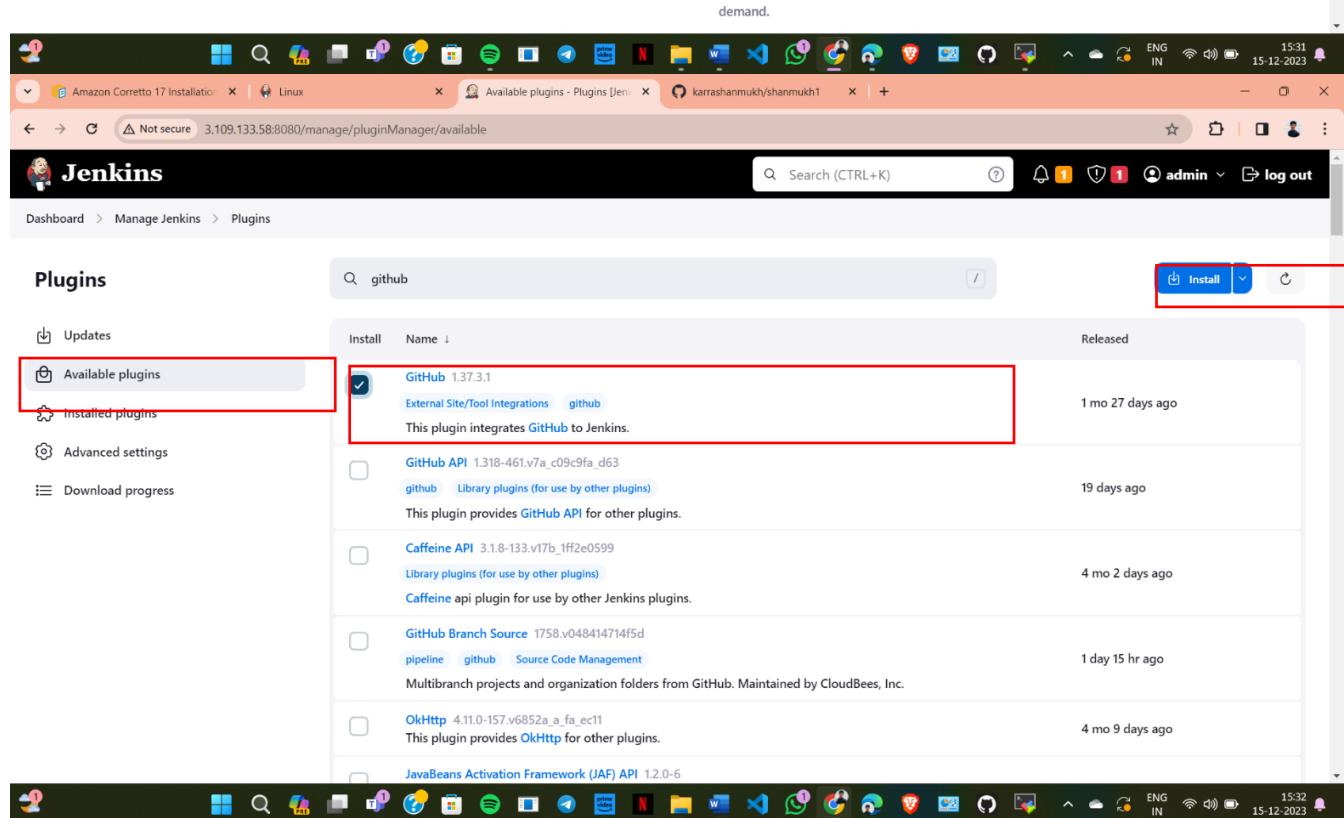
The screenshot shows the Jenkins dashboard at [3.109.133.58:8080](http://3.109.133.58:8080). The left sidebar has a red box around the '+ New Item' button. The main area displays the 'Welcome to Jenkins!' message and various management links like 'Create a job', 'Set up a distributed build', and 'Set up an agent'.

The screenshot shows the 'Enter an item name' dialog with 'shanmukh1' entered in the input field. A red box highlights the 'Required field' placeholder. Below it, a red box highlights the 'Freestyle project' section and its description. The 'OK' button at the bottom is also highlighted with a red box.

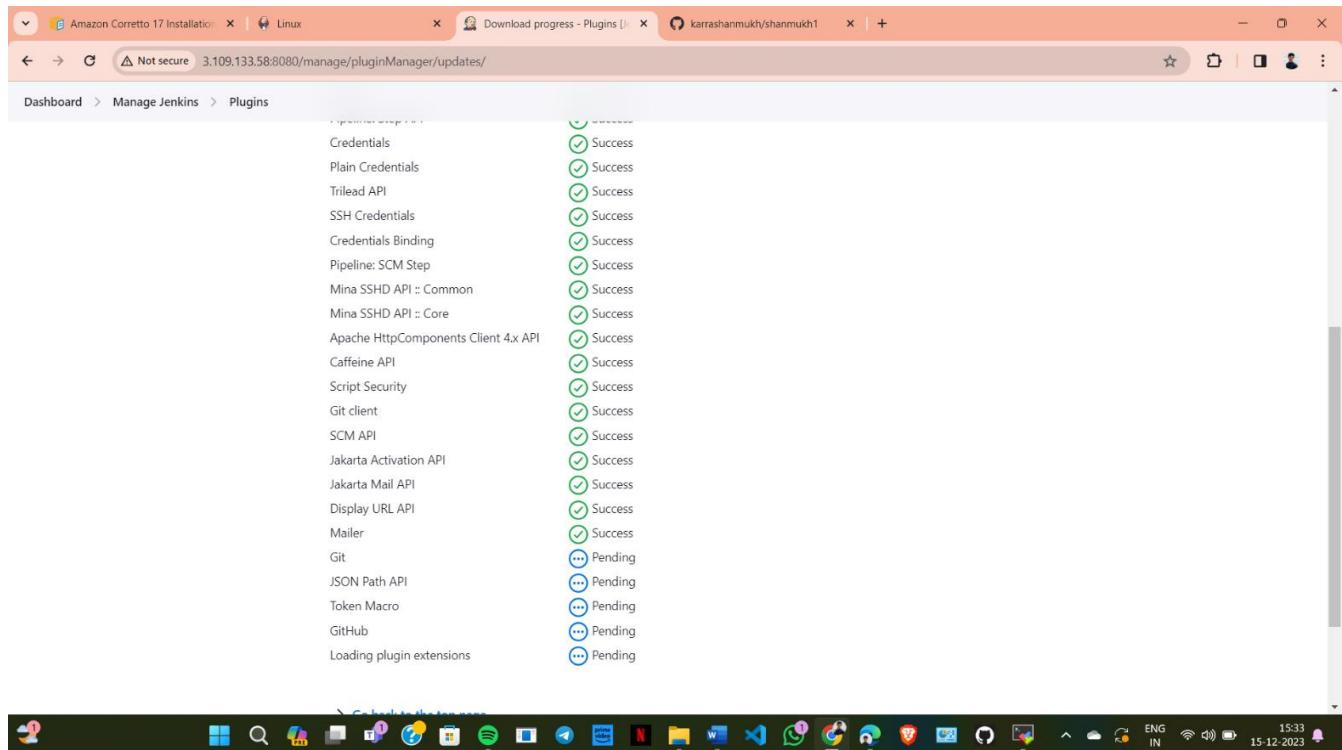
Step-13: Goto manage Jenkins > plugins. In the available plugins, search for **GitHub** and install.



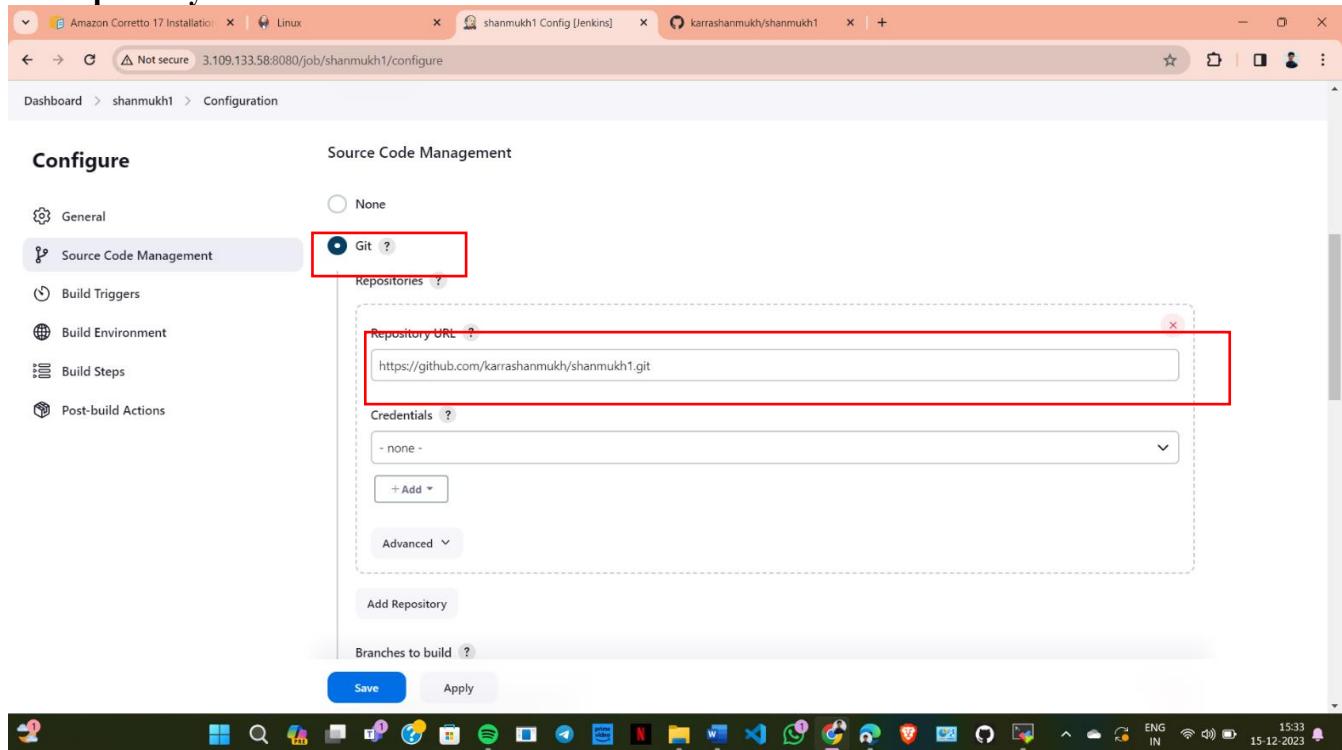
The screenshot shows the Jenkins Manage Jenkins interface. On the left sidebar, the 'Manage Jenkins' option is highlighted with a red box. The main area displays system configuration options like System, Tools, Nodes, Clouds, and Plugins. The 'Plugins' section is also highlighted with a red box. A message box at the top right says: 'Jenkins URL is empty but is required for the proper operation of many Jenkins features like email notifications, PR status update, and environment variables such as BUILD\_URL.' It also asks for an accurate value in Jenkins configuration and has a 'Dismiss' button.

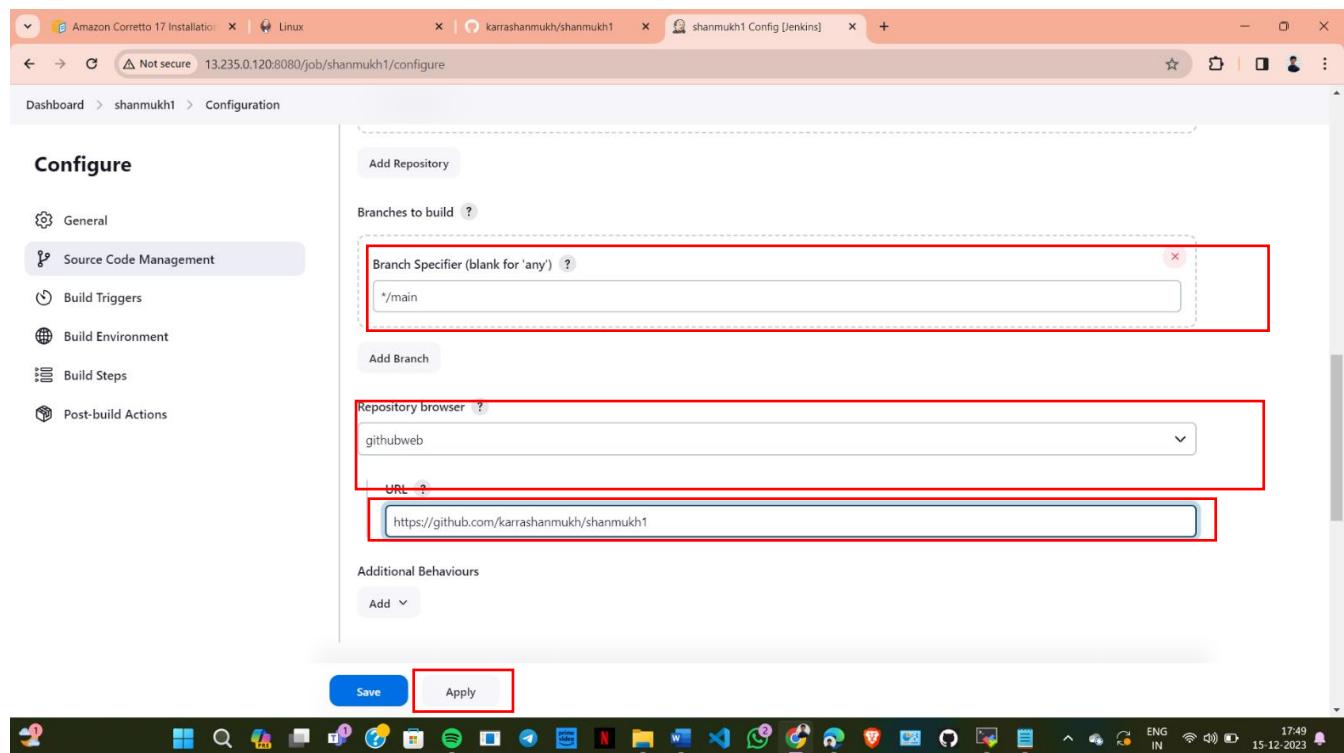
The screenshot shows the Jenkins Plugins page. The search bar at the top contains 'github'. The 'Available plugins' section is highlighted with a red box. A specific entry for the 'GitHub' plugin is selected, showing its details: 'GitHub 1.37.3.1' with a checked checkbox, 'External Site/Tool Integrations github', and a note: 'This plugin integrates GitHub to Jenkins.' The 'Install' button is highlighted with a red box. Other listed plugins include 'GitHub API', 'Caffeine API', 'GitHub Branch Source', 'OkHttp', and 'JavaBeans Activation Framework (JAF) API'.



Step-14: Goto **Source Code Management** and select git , paste the copied github repository's url in the **Repository URL** section.



Step-15: In the **Branch Specifier** section, enter `*/main` , and in **Repository browser** section, enter `githubweb`. Enter the repository's URL in the **URL** section by removing ".git" at the end of URL. Apply and save the changes. Click on Build.



## Maven:

### Maven installation commands:

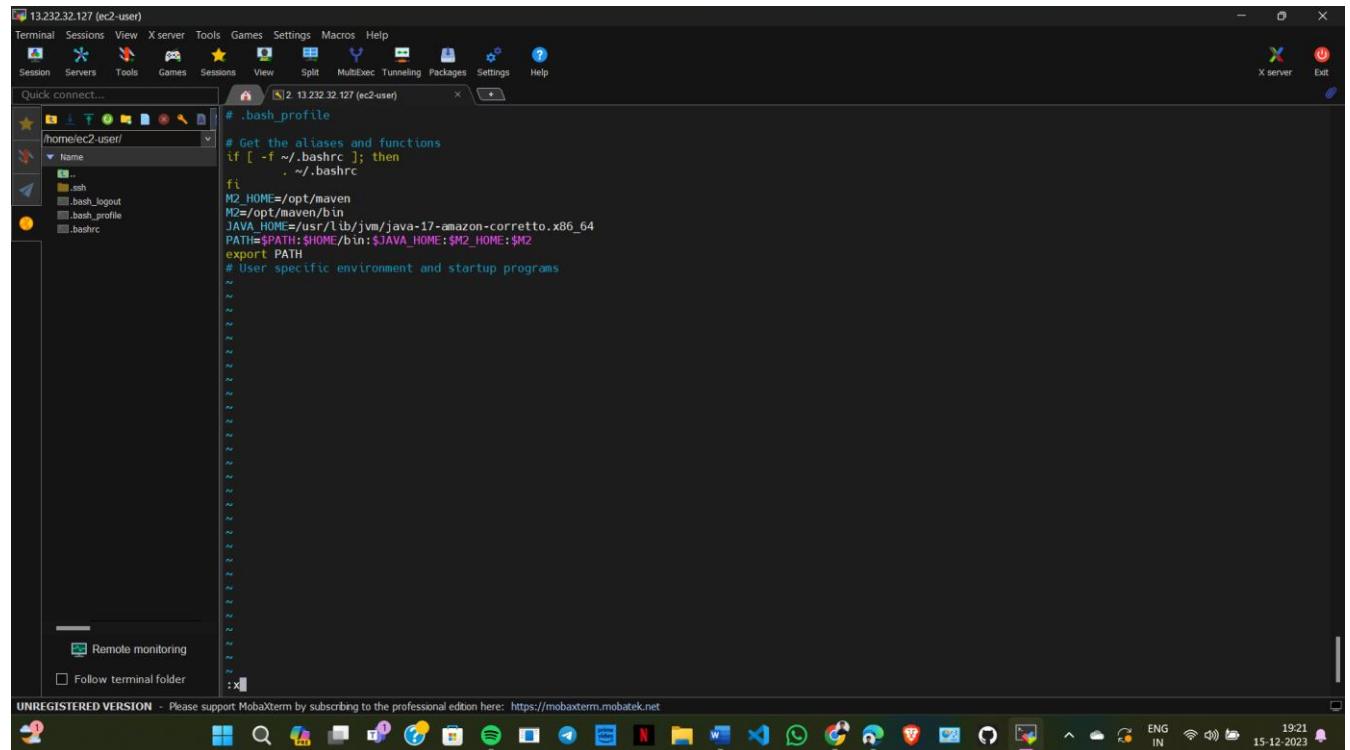
```
[root@ip-172-31-14-200 ~]# sudo su -
[root@ip-172-31-14-200 ~]# cd /opt/
[root@ip-172-31-14-200 opt]# ll
[root@ip-172-31-14-200 opt]# wget https://dlcdn.apache.org/maven/maven-3/3.9.6/binaries/apache-maven-3.9.6-bin.tar.gz
[root@ip-172-31-14-200 opt]# tar -xzvf apache-maven-3.9.6-bin.tar.gz
[root@ip-172-31-14-200 opt]# ll
[root@ip-172-31-14-200 opt]# mv apache-maven-3.9.6 maven
[root@ip-172-31-14-200 opt]# ll
[root@ip-172-31-14-200 opt]# cd maven
[root@ip-172-31-14-200 maven]# ll
[root@ip-172-31-14-200 maven]# cd bin
[root@ip-172-31-14-200 bin]# ./mvn -v
```

### output:

```
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)
Maven home: /opt/maven
Java version: 17.0.9, vendor: Amazon.com Inc., runtime:
/usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en, platform encoding: UTF-8
```

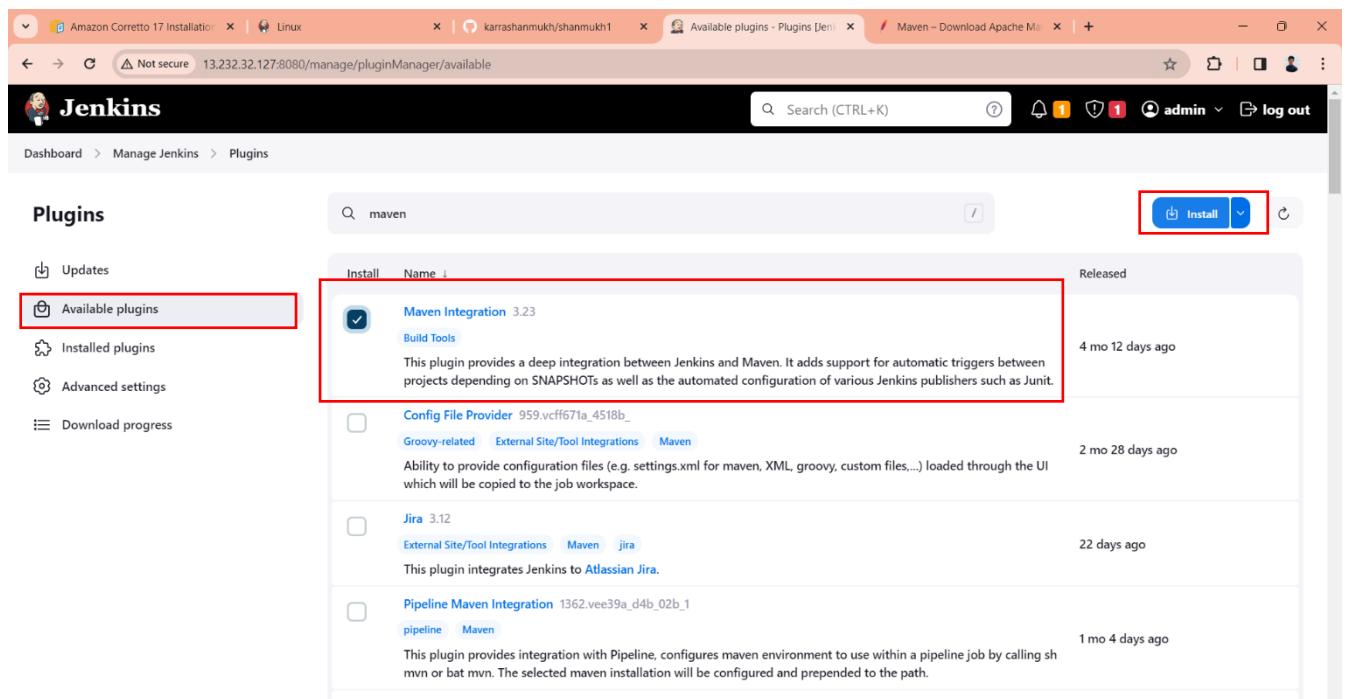
OS name: "linux", version: "6.1.66-91.160.amzn2023.x86\_64", arch: "amd64", family: "unix"

```
[root@ip-172-31-14-200 bin]# cd ~  
[root@ip-172-31-14-200 ~]# pwd  
/root  
[root@ip-172-31-14-200 ~]# vi .bash_profile
```

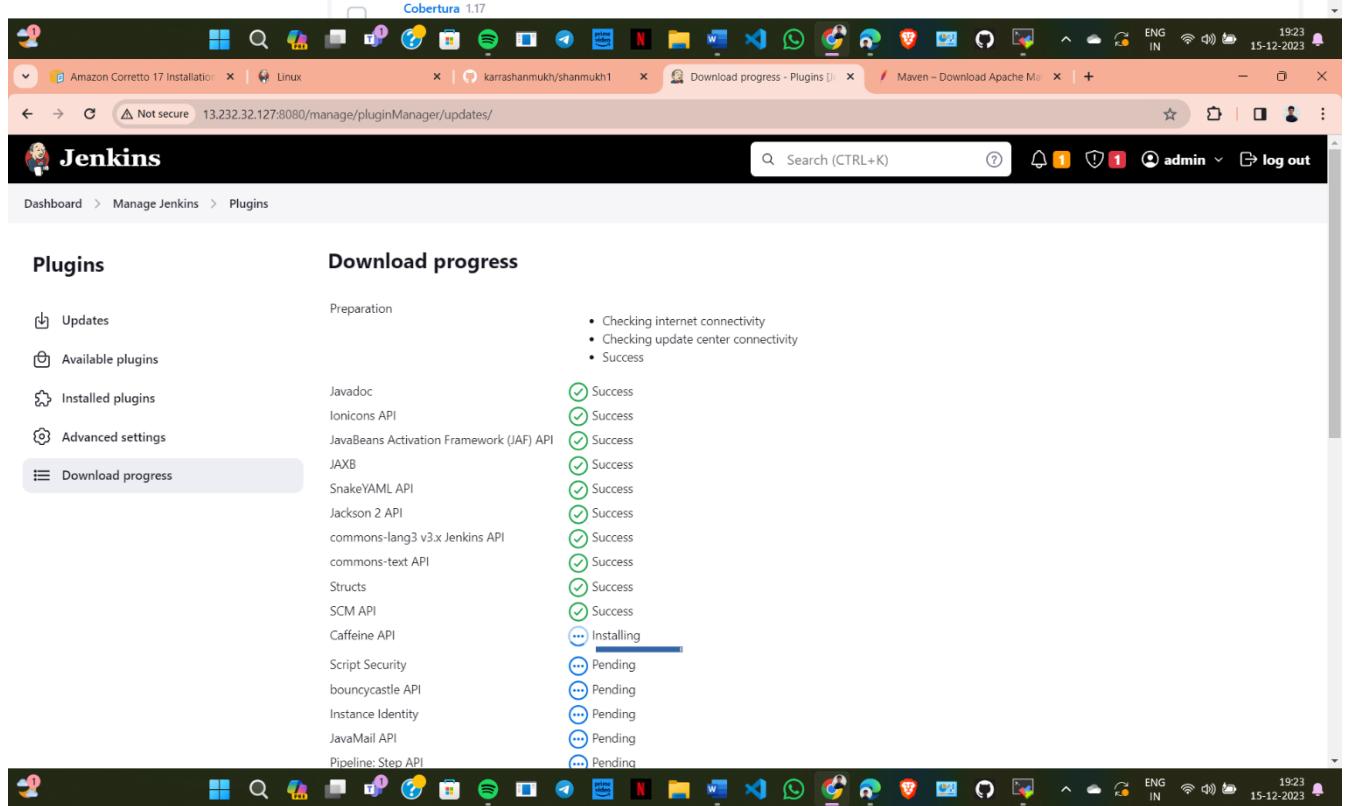


The screenshot shows a MobaXterm session titled '13.232.32.127 (ec2-user)'. The terminal window displays the contents of the '.bash\_profile' file. The file starts with a shebang '#!/bin/bash' and contains several environment variable assignments and export statements. The terminal interface includes a sidebar with session management tools like Session, Servers, Tools, Games, and X server. The taskbar at the bottom shows various application icons.

```
#!/bin/bash  
# Get the aliases and functions  
if [ -t 0 ]; then  
    . ./bashrc  
fi  
M2_HOME=/opt/maven  
M2=/opt/maven/bin  
JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64  
PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2  
export PATH  
# User specific environment and startup programs
```



The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "maven". On the left, a sidebar menu has "Available plugins" selected. In the main area, a plugin named "Maven Integration 3.23" under the "Build Tools" category is highlighted with a red box. Its description states: "This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as JUnit". To the right of the plugin's details, there is a "Released" section showing the date "4 mo 12 days ago". At the top right of the main area, there is a blue "Install" button with a red box around it.

The screenshot shows the Jenkins Plugins page with the "Download progress" section selected in the sidebar. The main area displays a table titled "Download progress" with two columns: "Preparation" and "Progress". The "Preparation" column lists several Jenkins APIs and their status: Javadoc (Success), Ionicons API (Success), JavaBeans Activation Framework (JAF) API (Success), JAXB (Success), SnakeYAML API (Success), Jackson 2 API (Success), commons-lang3 v3.x Jenkins API (Success), commons-text API (Success), Struts (Success), SCM API (Success), Caffeine API (Installing), Script Security (Pending), bouncycastle API (Pending), Instance Identity (Pending), JavaMail API (Pending), and Pipeline: Step API (Pending). The "Progress" column shows green checkmarks for successful items and a blue progress bar for the "Installing" item.

The screenshot shows the Jenkins Manage Jenkins interface. On the left sidebar, the 'Manage Jenkins' option is selected and highlighted with a red box. In the main content area, under 'System Configuration', the 'Tools' section is highlighted with a red box. This section contains a link to 'Configure tools, their locations and automatic installers.' Below it, there are sections for 'System', 'Nodes', 'Clouds', and 'Plugins'. The 'Tools' section also has a note about Jenkins URL being empty and a 'Dismiss' button. At the bottom of the page, the 'Tools' link in the breadcrumb navigation is also highlighted with a red box.

Maven installations

Add Maven

**Maven**

Name: maven-3.9.5

MAVEN\_HOME: /opt/maven

Install automatically

Add Maven

Save Apply

Enter an item name

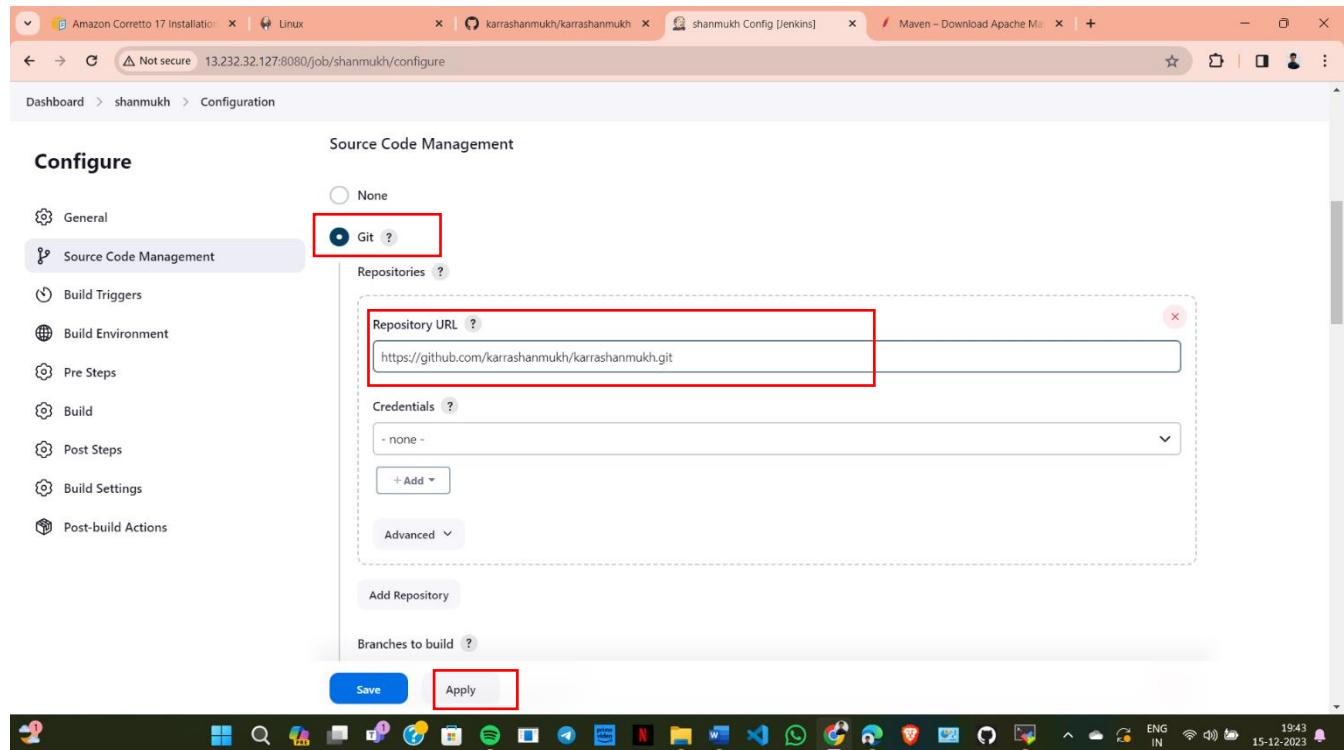
shanmukh » Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

OK





### Step3:create another instances for tomcat

- To build maven application using Jenkins and deploy it Tomcat.
- First Go to AWS webpage and Login into console. Then search EC2 click on it.
- Before, Tomcat server create Jenkins server and install maven in it and
- Create another Instance for Tomcat Server. Create a session and paste public IP address in Mobaxterm.
- Download tomcat by copy link address from Tomcat official Website. And unzip the zip file and rename it to tomcat

```
[ec2-user@ip-172-31-27-252 ~]$ sudo su -
[root@ip-172-31-27-252 ~]# cd /opt
[root@ip-172-31-27-252 opt]# wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.83/bin/apache-tomcat-9.0.83.zip
--2023-12-11 16:09:50-- https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.83/bin/apache-tomcat-9.0.83.zip
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12344614 (12M) [application/zip]
Saving to: 'apache-tomcat-9.0.83.zip'

apache-tomcat-9.0.83.zip          100%[=====] 2023-12-11 16:09:50 (229 MB/s) - 'apache-tomcat-9.0.83.zip' saved [12344614/12344614]

[root@ip-172-31-27-252 opt]# unzip apache-tomcat-9.0.83.zip
Archive:  apache-tomcat-9.0.83.zip
   creating apache-tomcat-9.0.83/
```

```
[root@ip-172-31-27-252 opt]# ll -a
total 12072
drwxr-xr-x. 4 root root 77 Dec 11 16:10 .
dr-xr-xr-x. 18 root root 237 Nov 10 19:53 ..
drwxr-xr-x. 9 root root 16384 Nov 9 20:57 apache-tomcat-9.0.83
-rw-r--r--. 1 root root 12344614 Nov 9 21:47 apache-tomcat-9.0.83.zip
drwxr-xr-x. 4 root root 33 Nov 10 19:54 aws
[root@ip-172-31-27-252 opt]# mv apache-tomcat-9.0.83 tomcat
[root@ip-172-31-27-252 opt]# cd tomcat
[root@ip-172-31-27-252 tomcat]# chmod -R u=rwx bin
[root@ip-172-31-27-252 tomcat]# ln -s /opt/tomcat/bin/startup.sh /usr/local/bin/tomcatup
[root@ip-172-31-27-252 tomcat]# ln -s /opt/tomcat/bin/shutdown.sh /usr/local/bin/tomcatdown
[root@ip-172-31-27-252 tomcat]# tomcatup
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program
[root@ip-172-31-27-252 tomcat]# sudo yum install java-17-amazon-corretto-devel
Last metadata expiration check: 0:31:50 ago on Mon Dec 11 15:41:59 2023.
Dependencies resolved.
=====
Package                                     Architecture   Version
=====
Installing:
  java-17-amazon-corretto-devel           x86_64        1:17.0.9+8-1.amzn2023.1
```

- Change the permissions for bin in tomcat. And Install JDK – 17 into the instance.
- And change the startup.sh and shutdown.sh to the tomcatup and tomcatdown.
- Now, Start Tomcat Server and Modify the context.xml in the host-manager and manager. By commenting the valve tag

```
Complete!
[root@ip-172-31-27-252 tomcat]# tomcatup
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME: /usr
Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[root@ip-172-31-27-252 tomcat]# find / -name context.xml
/opt/tomcat/conf/context.xml
/opt/tomcat/webapps/docs/META-INF/context.xml
/opt/tomcat/webapps/examples/META-INF/context.xml
/opt/tomcat/webapps/host-manager/META-INF/context.xml
/opt/tomcat/webapps/manager/META-INF/context.xml
[root@ip-172-31-27-252 tomcat]# vi /opt/tomcat/webapps/host-manager/META-INF/context.xml
[root@ip-172-31-27-252 tomcat]# vi /opt/tomcat/webapps/manager/META-INF/context.xml
[root@ip-172-31-27-252 tomcat]# cd conf
[root@ip-172-31-27-252 conf]# vi tomcat-users.xml
[root@ip-172-31-27-252 conf]#
```

- Move to conf and modify tomcat-users.xml by adding roles and users

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
  version="1.0">
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="user" password="user" roles="manager-gui"/>
```

→ Install plugin of Deploy to container. Add credentials of tomcat to Jenkins global credentials.

The screenshot shows the Jenkins 'Global credentials (unrestricted)' configuration page. A new credential is being created with the following details:

- Kind:** Username with password
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Username:** admin
- Treat username as secret:** Unchecked
- Password:** (redacted)
- ID:** tomcat-id
- Description:** (empty)

A blue 'Create' button is at the bottom left.

The screenshot shows the Jenkins 'General' configuration page for a Maven project. The 'General' tab is selected. The configuration includes:

- Description:** maven integration
- Enabled:** Enabled (checkbox checked)
- Source Code Management:** GitHub project (checkbox checked)
  - Project url:** https://github.com/csebec-github/maven
- Advanced:** (button)
- GitLab Connection:** (dropdown menu)
- Use alternative credential:** (checkbox)
- Permission to Copy Artifact:** (checkbox)

At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration interface for a Maven project. In the top navigation bar, the path is "Dashboard > maven > Configuration". The left sidebar has a "Configure" section with tabs: General, Source Code Management (selected), Build Triggers, Build Environment, Pre Steps, Build, Post Steps, Build Settings, and Post-build Actions.

**Source Code Management:**

- Repository URL: <https://github.com/csebec-github/maven.git>
- Credentials: - none -
- Add Repository: Add Branch
- Branches to build: Branch Specifier (blank for 'any')
- Repository browser: githubweb
- URL: <https://github.com/csebec-github/maven>

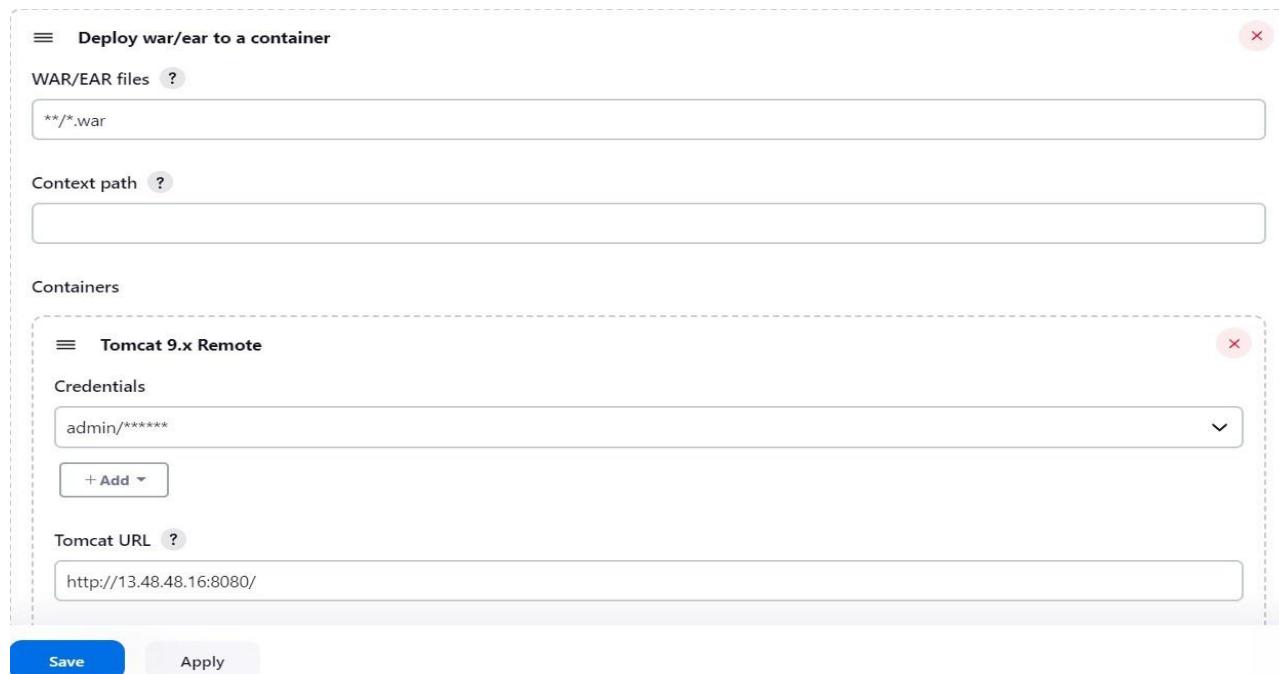
**Build Triggers:**

- Build whenever a SNAPSHOT dependency is built
- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Build when a change is pushed to BitBucket
- Build when a change is pushed to GitLab. GitLab webhook URL: <http://16.170.163.138:8080/project/maven>
- GitHub hook trigger for GITScm polling
- Perform triggered build.
- Poll SCM

**Schedule:**

```
* * * * *
```

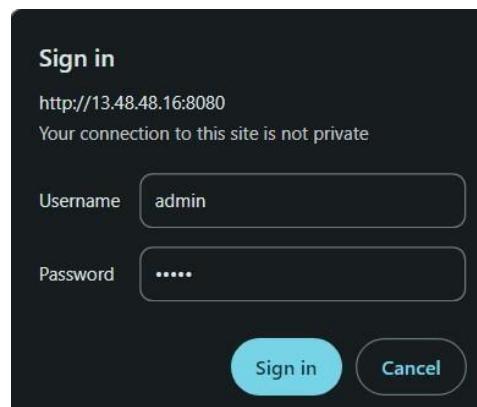
**Warning:** **⚠️ Do you really mean "every minute" when you say "\*\*\*\*\*"? Perhaps you meant "H \* \* \* \*" to poll once per hour**  
Would last have run at Monday, December 11, 2023 at 4:36:12 PM Coordinated Universal Time; would next run at Monday, December 11, 2023 at 4:37:12 PM Coordinated Universal Time



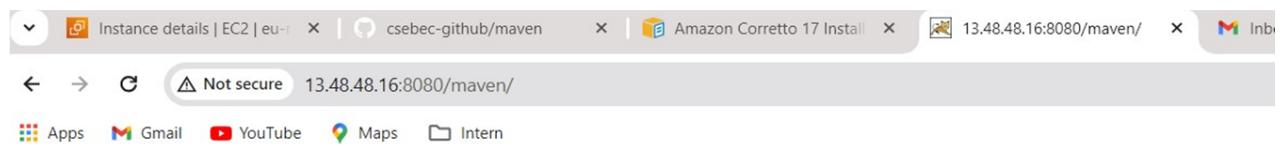
→ After all the above steps save and apply.

```
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/maven/maven/src/main/webapp]
[INFO] Building war: /var/lib/jenkins/workspace/maven/maven/target/maven.war
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ maven ---
[INFO] Installing /var/lib/jenkins/workspace/maven/pom.xml to /var/lib/jenkins/.m2/repository/project/maven/0.0.1-SNAPSHOT/maven-0.0.1-SNAPSHOT.pom
[INFO] Installing /var/lib/jenkins/workspace/maven/target/maven.war to /var/lib/jenkins/.m2/repository/project/maven/0.0.1-SNAPSHOT/maven-0.0.1-SNAPSHOT.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.083 s
[INFO] Finished at: 2023-12-11T16:39:39Z
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/maven/pom.xml to project/maven/0.0.1-SNAPSHOT/maven-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/maven/target/maven.war to project/maven/0.0.1-SNAPSHOT/maven-0.0.1-SNAPSHOT.war
channel stopped
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/maven/target/maven.war to container Tomcat 9.x Remote with context null
[/var/lib/jenkins/workspace/maven/target/maven.war] is not deployed. Doing a fresh deployment.
Deploying [/var/lib/jenkins/workspace/maven/target/maven.war]
Finished: SUCCESS
```

- Now, Open New Tab and search with tomcat IP :8080 and Redirect to tomcat page.
- There click on manager app.
- And Login to the tomcat using the credentials of tomcat user having deployer.



- After successful login. Click on the project that deployed maven project from the Jenkins after building the project.
- It will run project on tomcat server



### Login Form

Username:

Password:

## 6. Aim: Deploying a Full Stack Application on AWS EC2 with Docker.

### Source Code:

#### Step-1: Creating the aws EC2 instances:

- 1.Log in to the AWS Management Console.
- 2.Navigate to the EC2 service.
- 3.Click on “Launch Instance” to create a new EC2 instance.
- 4.Select the desired instance type, storage, and other configurations.
- 5.Configure the security group to allow inbound traffic on port 22 for SSH access.
- 6.Launch the instance and download the private key file (.pem).

The screenshot shows the AWS Management Console interface for launching an EC2 instance. The top navigation bar includes the AWS logo, Services, Search, and Mumbai/Atchuta\_Pavan dropdown. The main content area has a breadcrumb trail: EC2 > Instances > Launch an instance. The current step is 'Launch an instance' with a sub-step 'Name and tags'. Below it is 'Application and OS Images (Amazon Machine Image)'. The 'Summary' step is active, showing the configuration: Number of instances (1), Instance type (t2.micro), Firewall (security group) (New security group), and Storage (volumes) (1 volume(s) - 8 GiB). A tooltip for the 'Free tier' is displayed, stating: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS'. The 'Launch instance' button is highlighted in orange. The bottom section shows the 'Network settings' step, where a new security group 'Launch-wizard-1' is being configured with rules for SSH, HTTPS, and HTTP traffic. A note at the bottom says: 'Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting ...'.

#### Step-2: Installing Git and docker compose

1. Install Git by running the command:

```
sudo yum install git -y
```

```
Installed:
  git-2.40.1-1.amzn2023.0.1.x86_64
  git-core-2.40.1-1.amzn2023.0.1.x86_64
  git-core-doc-2.40.1-1.amzn2023.0.1.noarch
  perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  perl-File-Find-1.37-477.amzn2023.0.6.noarch
  perl-Git-2.40.1-1.amzn2023.0.1.noarch
  perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
  perl-lib-0.65-477.amzn2023.0.6.x86_64
```

Complete!

2. Install Docker by executing the following commands:

```
sudo yum install docker -y  
sudo usermod -aG docker ec2-user  
sudo service docker start
```

```
Installing      : libcgroup-3.0-1.amzn2023.0.1.x86_64          3/10
Installing      : libnftnetlink-1.0.1-19.amzn2023.0.2.x86_64        4/10
Installing      : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64      5/10
Installing      : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64          6/10
Installing      : liblfnl-1.2.2-2.amzn2023.0.2.x86_64          7/10
Installing      : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64          8/10
Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64        8/10
Installing      : pigz-2.5-1.amzn2023.0.3.x86_64          9/10
Running scriptlet: docker-24.0.5-1.amzn2023.0.3.x86_64        10/10
Installing      : docker-24.0.5-1.amzn2023.0.3.x86_64        10/10
Running scriptlet: docker-24.0.5-1.amzn2023.0.3.x86_64        10/10
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

Verifying      : pigz-2.5-1.amzn2023.0.3.x86_64          1/10
Verifying      : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64      2/10
Verifying      : containerd-1.7.2-1.amzn2023.0.4.x86_64          3/10
Verifying      : liblfnl-1.2.2-2.amzn2023.0.2.x86_64          4/10
Verifying      : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64          5/10
Verifying      : runc-1.1.7-1.amzn2023.0.3.x86_64          6/10
Verifying      : docker-24.0.5-1.amzn2023.0.3.x86_64          7/10
Verifying      : libnftnetlink-1.0.1-19.amzn2023.0.2.x86_64        8/10
Verifying      : libcgroup-3.0-1.amzn2023.0.1.x86_64          9/10
Verifying      : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64        10/10

Installed:
  containerd-1.7.2-1.amzn2023.0.4.x86_64                  docker-24.0.5-1.amzn2023.0.3.x86_64
  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  libcgroup-3.0-1.amzn2023.0.1.x86_64                  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnftnetlink-1.0.1-19.amzn2023.0.2.x86_64            liblfnl-1.2.2-2.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64                      runc-1.1.7-1.amzn2023.0.3.x86_64

Complete!
[ec2-user@ip-172-31-36-250 ~]$ sudo usermod -aG docker ec2-user
[ec2-user@ip-172-31-36-250 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-36-250 ~]$ █
```

**Step-3:** Install Docker Compose by running the commands:

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.12.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
[ec2-user@ip-172-31-36-250 ~]$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.12.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload Total Spent   Left  Speed
0       0     0      0      0      0      0 --:--:-- --:--:-- --:--:-- 0
100 42.8M 100 42.8M 0      0 26.4M      0  0:00:01  0:00:01 --:--:-- 122M
[ec2-user@ip-172-31-36-250 ~]$
```

sudo mv /usr/local/bin/docker-compose /usr/bin/docker-compose  
sudo chmod +x /usr/bin/docker-compose

```
[ec2-user@ip-172-31-36-250 ~]$ sudo mv /usr/local/bin/docker-compose /usr/bin/docker-compose
[ec2-user@ip-172-31-36-250 ~]$ sudo chmod +x /usr/bin/docker-compose
[ec2-user@ip-172-31-36-250 ~]$
```

#### **Step-4: Cloning the repository**

## 1.Change to desired path

```
cd /path/to/desired-directory
```

```
[ec2-user@ip-172-31-36-250 ~]$ sudo mkdir -p /path/to/desired-directory
[ec2-user@ip-172-31-36-250 ~]$ cd /path/to/desired-directory
[ec2-user@ip-172-31-36-250 desired-directory]$ █
```

## 2.Cloning the repository git:

```
git clone https://github.com/madhurajayashanka/docker-mysql-nodejs-reactjs-app.git
```

```
[ec2-user@ip-172-31-36-250 desired-directory]$ sudo git clone https://github.com/madhurajayashanka/docker-mysql-nodejs-reactjs-app.git
Cloning into 'docker-mysql-nodejs-reactjs-app'...
remote: Enumerating objects: 29514, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 29514 (delta 6), reused 30 (delta 6), pack-reused 29483
Receiving objects: 100% (29514/29514), 51.13 MiB | 6.46 MiB/s, done.
Resolving deltas: 100% (5281/5281), done.
Updating files: 100% (37870/37870), done.
```

## Step-5: Configuring and Running the Dockerized Application

### 1. Change into the project directory:

```
cd docker-mysql-nodejs-reactjs-app
```

```
[ec2-user@ip-172-31-36-250 desired-directory]$ cd docker-mysql-nodejs-reactjs-app
[ec2-user@ip-172-31-36-250 docker-mysql-nodejs-reactjs-app]$ █
```

### 2. Modify the configuration files (if necessary) for the application.

### 3. Build and start the containers using Docker Compose:

```
docker-compose up --build
```

```
[+] Building 259.1s (18/18) FINISHED
=> [docker-mysql-nodejs-reactjs-app-frontend internal] load build definition from Dockerfile          0.1s
=> => transferring dockerfile: 650B               0.1s
=> [docker-mysql-nodejs-reactjs-app-frontend internal] load .dockerignore                         0.1s
=> => transferring context: 2B                  0.0s
=> [docker-mysql-nodejs-reactjs-app-api internal] load build definition from Dockerfile          0.1s
=> => transferring dockerfile: 553B               0.1s
=> [docker-mysql-nodejs-reactjs-app-api internal] load .dockerignore                         0.1s
=> => transferring context: 2B                  0.1s
=> [docker-mysql-nodejs-reactjs-app-frontend internal] load metadata for docker.io/library/node:20      2.3s
=> [docker-mysql-nodejs-reactjs-app-frontend internal] load build definition from Dockerfile        53.8s
=> => resolve docker.io/library/node:20@sha256:445acd9b2ef7e9de665424053bf95
=> => sha256:27e1a8ca91d35598fbae8dee7f1c211f0f93ce529f6804a60e9301c53a604d0 24.05MB / 24.05MB
=> => sha256:d3a767d1d12e5772489f254794e359f3b04d4d5ad966006e5b5cda78cc382762 64.13MB / 64.13MB
=> => sha256:90e5e7d887a34877f61c2b86d053db1c4f440b9054cf49573e3be5d6a674a47 49.58MB / 49.58MB
=> => sha256:445acd9b2ef7e9de665424053bf95652e0b8995e36500557d48faf29300170a 1.21kB / 1.21kB
=> => sha256:b41c3ecc3dc2404464bde37f6ee28fc832b2d99b10caeaa38b250a9b9d601e8 2.00kB / 2.00kB
=> => sha256:cda45fab4fce55c63c3eb4b7f52db026ed4c6d5b977f36be324af800530843a 7.53kB / 7.53kB
=> => extracting sha256:90e5e7d8b87a34877f61c2b86d053db1c4f440b9054cf49573e3be5d6a674a47 14.5s
=> => sha256:1229565364ef2361591684e23e3b8bb5910da23197635a2b5b96a34b488d 3.37kB / 3.37kB
=> => sha256:711be5dc50448ab08ccab0b44d65962f36574d341749ab30651b78ec0d4bfd1c 211.07MB / 211.07MB
=> => sha256:53a3765121b18007f5a66660b95a49330a81912398db54d36cb1781639cb8ee5 47.93MB / 47.93MB
=> => sha256:97eae3704e79fa0eda21ac525039beeacb573c32e3c95fb6255a954f8d4cc 2.21MB / 2.21MB
=> => sha256:f52b2977463aa441447e75b516cc29ca2f36c89f2939b44aae8bb0a7f8231e21 450B / 450B
=> => extracting sha256:27e1a8ca91d35598fbae8dee7f1c211f0f93ce529f6804a60e9301c53a604d0
=> => extracting sha256:d3a767d1d12e5772489f254794e359f3b04d4d5ad966006e5b5cda78cc382762
=> => extracting sha256:711be5dc50448ab08ccab0b44d65962f36574d341749ab30651b78ec0d4bfd1c
=> => extracting sha256:22956530cc64ef2361591684e23e3b8e5bb5910da23197635a2b5b96a34b488d
=> => extracting sha256:53a3765121b18007f5a66660b95a49330a81912398db54d36cb1781639cb8ee5
=> => extracting sha256:97eae3704e79fa0eda21ac5525039beeacb573c32e3c95fb6255a954f8d4cc
=> => extracting sha256:f52b2977463aa441447e75b516cc29ca2f36c89f2939b44aae8bb0a7f8231e21
=> [docker-mysql-nodejs-reactjs-app-api internal] load build context
```

## Step-6: Configure Aws Security group

- Go to the AWS Management Console and navigate to the EC2 service.
- Select your running instance and click on “Actions” > “Networking” > “Change Security Groups.”
- Create a new security group or modify the existing one.
- Add a new inbound rule to allow traffic on port 3001 (or the desired port) from anywhere (0.0.0.0/0).

The screenshot shows the AWS Management Console interface for managing EC2 security groups. The specific view is 'Inbound rules' for a selected security group. The table lists four rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-048cd1c83b935034d	Custom TCP	TCP	8080	Cu... 0.0.0.0/0	0.0.0.0/0 X Delete
sgr-06c3dcdf9a6fb09e4	SSH	TCP	22	Cu... 0.0.0.0/0	0.0.0.0/0 X Delete
sgr-0b66944ecf4b2cb3c	HTTPS	TCP	443	Cu... 0.0.0.0/0	0.0.0.0/0 X Delete
sgr-0a66ac2245c117cb1	Custom TCP	TCP	3000	Cu... 0.0.0.0/0	0.0.0.0/0 X Delete

## Step-7: Accessing the Application

- Obtain the public IP address of your EC2 instance from the AWS Management Console.
- Open a web browser and enter the following URL: <http://<public-ip-address>:3001>.

The screenshot shows a simple web application interface. At the top, there is a header bar with a back arrow, forward arrow, refresh icon, and a search bar containing the IP address '65.2.31.23:3001'. Below the header is a form titled 'User Submit Form' with a single input field and a green 'Submit' button. Below the form is a horizontal line. Underneath the line is a section titled 'Users List' with a table:

ID	Name
1	Madhura
2	Jayashanka

## 7. Aim: Create a CI/CD pipeline by using Jenkins with AWS CodeBuild & AWS Code Deploy.

### Source Code:

#### Step-1: Creating the aws EC2 instances:

- 1.Log in to the AWS Management Console.
- 2.Navigate to the EC2 service.
- 3.Click on “Launch Instance” to create a new EC2 instance.
- 4.Select the desired instance type, storage, and other configurations.
- 5.Configure the security group to allow inbound traffic on port 22 for SSH access.
- 6.Launch the instance and download the private key file (.pem).

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, a name 'My Web Server' is entered. In the 'Application and OS Images (Amazon Machine Image)' step, an AMI is selected. On the right, the 'Summary' pane shows:

- Number of instances: 1
- Instance Type: t2.micro
- Firewall (security group): New security group
- Storage (volumes): 1 volume(s) - 8 GiB

A modal window indicates a 'Free tier: In your first year includes 750 hours of t2.micro'. At the bottom, there are 'Cancel', 'Launch instance', and 'Review commands' buttons.

The screenshot shows the 'Network settings' step of the wizard. It displays network and subnet information. Under 'Firewall (security groups)', a new security group 'launch-wizard-1' is being created with the following rules:

- Allow SSH traffic from Anywhere
- Allow HTTPS traffic from the internet
- Allow HTTP traffic from the internet

A note at the bottom states: "Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting X".

#### Step-2:SSH into EC2 Instance, Install Java and Jenkins

SSH into the EC2 instance. After securely connecting to the instance, the first thing we have to do is to install the Java application onto the instance. Jenkins runs natively runs on the Java application so it has to be properly installed in order for the Jenkins server to be operational.

### 1. sudo wget -O /etc/yum.repos.d/jenkins.repo \ https://pkg.jenkins.io/redhat-stable/jenkins.repo

```
[ec2-user@ip-172-31-13-239 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2023-12-17 07:10:03-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.154.133, 2a04:4e42:24::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.154.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.r 100%[=====] 85 --.-KB/s in 0s

2023-12-17 07:10:03 (5.74 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-13-239 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[ec2-user@ip-172-31-13-239 ~]$ sudo yum upgrade
Jenkins-stable
Dependencies resolved.
Nothing to do.
Complete!
```

### 2. sudo dnf install java-17-amazon-corretto -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo dnf install java-17-amazon-corretto -y
Last metadata expiration check: 0:00:06 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
Package           Arch    Version            Repository      Size
=====
Installing:
java-17-amazon-corretto x86_64  1:17.0.9+8-1.amzn2023.1  amazonlinux   188 k
Installing dependencies:
alsa-lib             x86_64  1.2.7.2-1.amzn2023.0.2  amazonlinux   504 k
cairo                x86_64  1.17.6-2.amzn2023.0.1  amazonlinux   684 k
dejavu-sans-fonts   noarch  2.37-16.amzn2023.0.2  amazonlinux   1.3 M
dejavu-sans-mono-fonts noarch  2.37-16.amzn2023.0.2  amazonlinux   467 k
dejavu-serif-fonts  noarch  2.37-16.amzn2023.0.2  amazonlinux   1.0 M
fontconfig           x86_64  2.13.94-2.amzn2023.0.2  amazonlinux   273 k
fonts-filesystem     noarch  1:2.0.5-12.amzn2023.0.2  amazonlinux   9.5 k
freetype              x86_64  2.13.0-2.amzn2023.0.1  amazonlinux   422 k
glib                 x86_64  5.2.1-9.amzn2023.0.1  amazonlinux   49 k
google-noto-fonts-common noarch  20201206-2.amzn2023.0.2  amazonlinux   15 k
google-noto-sans-vf-fonts noarch  20201206-2.amzn2023.0.2  amazonlinux   492 k
graphite2            x86_64  1.3.14-7.amzn2023.0.2  amazonlinux   97 k
harfbuzz              x86_64  7.0.0-2.amzn2023.0.1  amazonlinux   868 k
java-17-amazon-corretto-headless x86_64  1:17.0.9+8-1.amzn2023.1  amazonlinux   91 M
javapackages-filesystem noarch  6.0.0-7.amzn2023.0.6  amazonlinux   12 k
langpacks-core-font-en noarch  3.0-21.amzn2023.0.4  amazonlinux   10 k
libICE                x86_64  1.0.10-6.amzn2023.0.2  amazonlinux   71 k
libSM                 x86_64  1.2.3-8.amzn2023.0.2  amazonlinux   42 k
libX11                x86_64  1.7.2-3.amzn2023.0.4  amazonlinux   657 k
libX11-common          noarch  1.7.2-3.amzn2023.0.4  amazonlinux   152 k
libXau                x86_64  1.0.9-6.amzn2023.0.2  amazonlinux   31 k
libXext                x86_64  1.3.4-6.amzn2023.0.2  amazonlinux   41 k
```

### 3. sudo yum install jenkins -y

```
[ec2-user@ip-172-31-13-239 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:00:22 ago on Sun Dec 17 07:10:17 2023.
Dependencies resolved.
=====
 Package          Architecture      Version       Repository      Size
=====
Installing:
jenkins          noarch          2.426.2-1.1   jenkins        85 M
Transaction Summary
=====
Install 1 Package

Total download size: 85 M
Installed size: 85 M
Downloading Packages:
jenkins-2.426.2-1.1.noarch.rpm           7.7 MB/s | 85 MB  00:11
Total                                         7.7 MB/s | 85 MB  00:11
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :                                         1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Installing : jenkins-2.426.2-1.1.noarch        1/1
Running scriptlet: jenkins-2.426.2-1.1.noarch 1/1
Verifying   : jenkins-2.426.2-1.1.noarch        1/1

Installed:
jenkins-2.426.2-1.1.noarch
```

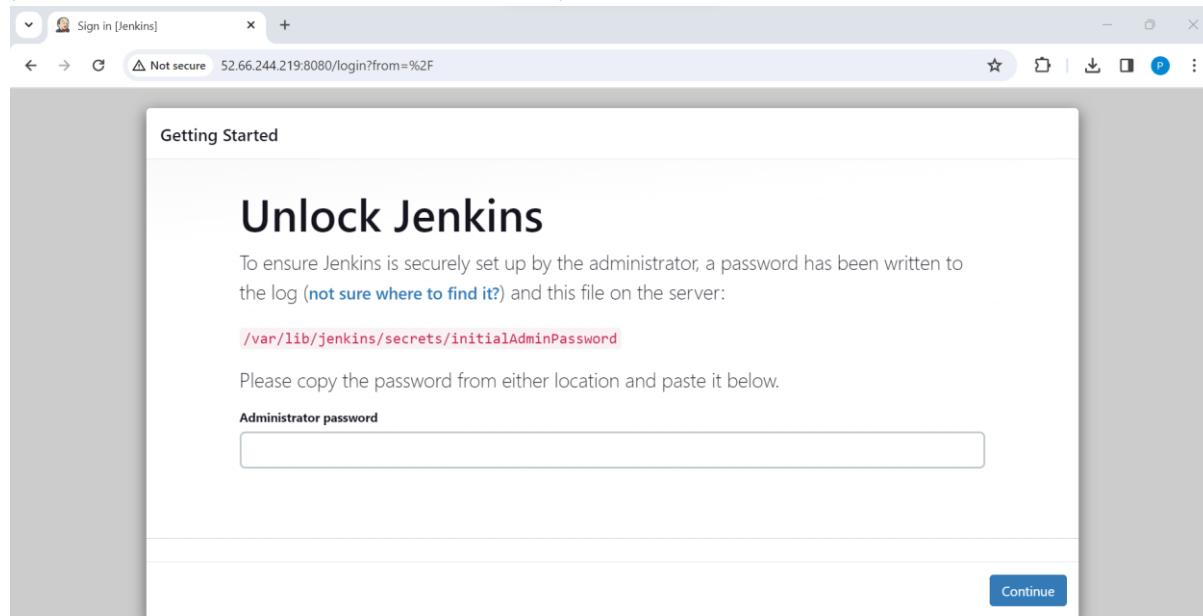
#### 4. sudo systemctl enable jenkins

```
sudo systemctl start jenkins
```

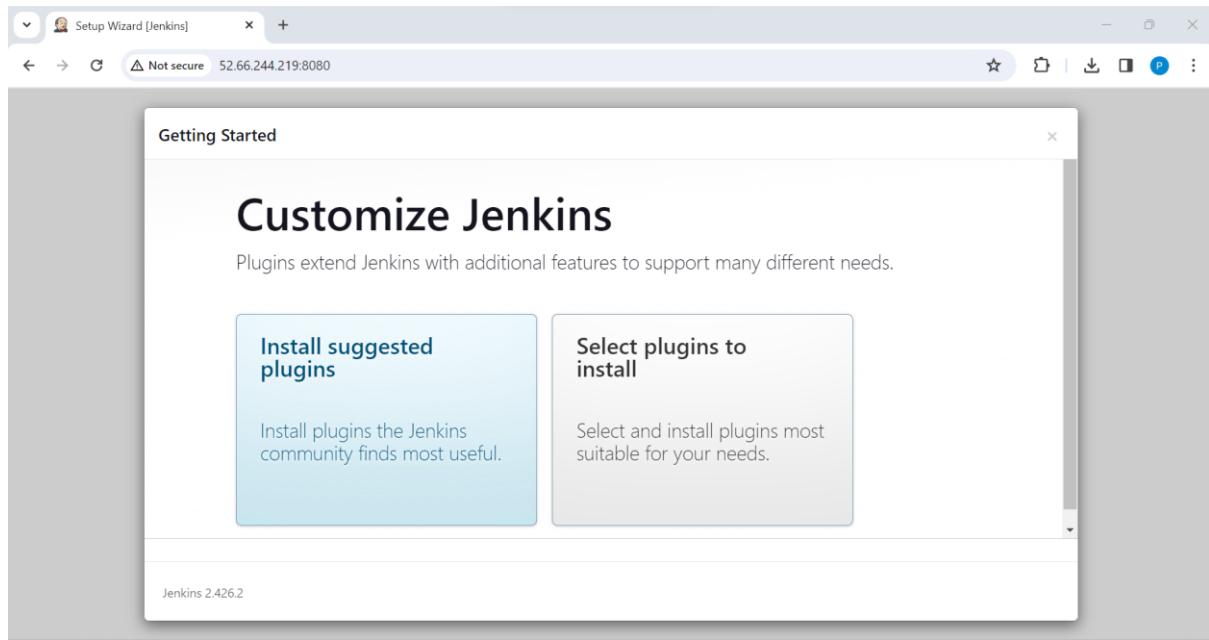
```
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-13-239 ~]$ sudo systemctl start jenkins
```

#### Step-3: Setting Up Jenkins Server

Now that the Jenkins server has been officially started on the instance, head over to the AWS Management console, grab the public IP address of the instance and paste that into your browser address bar followed by :8080



```
[ec2-user@ip-172-31-13-239 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
baaeef975c09e468a977274b59ed26a8e
[ec2-user@ip-172-31-13-239 ~]$
```



Setup an admin user. For this project we will get a dummy administrator user. In a real production you will use different credentials. Save and continue.

Username=admin

Password=Password

Full name= admin

[Email=admin@admin.com](#)

Getting Started

## Create First Admin User

Username  
admin

Password  
\*\*\*\*\*

Confirm password  
\*\*\*\*\*

Full name  
admin

E-mail address  
admin@admin.com

Jenkins 2.414.1 Skip and continue as admin Save and Continue

### Step 4: We are Live and Jenkins Ready! Time to Build

If you see the “Welcome to Jenkins” page, then you’ve successfully setup Jenkins and are ready to go to the next stage of the project, which is setting up our first Jenkins job!

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, Search bar, Notifications, User admin, Log out.
- Left Sidebar:**
  - New Item
  - People
  - Build History
  - Manage Jenkins
  - My Views
- Central Content:**

### 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 →
- Bottom Left:** Build Queue (No builds in the queue)
- Bottom Center:** Build Executor Status (1 idle, 2 idle)

Creating our first jenkin job

The screenshot shows the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo, a search bar containing "Search (%+K)", and various icons for help, security, and user management. Below the navigation bar, the left sidebar contains links for "New Item", "People", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section indicates "No builds in the queue". A "Build Executor Status" section shows 1 Idle and 2 Idle executors. The main content area features a "Welcome to Jenkins!" message, a "Start building your software project" section with a "Create a job" button (which is highlighted with a red box), and a "Set up a distributed build" section with links for "Set up an agent" and "Configure a cloud". At the bottom right, there are links for "REST API" and "Jenkins 2.414.1".

For our first job we are going to give it the name “**HelloWorldJob**” and select the option for “**Freestyle project**”. Click ok to move to the next step.

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

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

**Multi-configuration project**

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

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

**OK**



**Step-5:** Under the general configuration settings

In the description box, enter the text “HelloWorldJob”

Under the “Source Code Management” tab select “None” option

The screenshot shows the Jenkins job configuration interface for a job named 'HelloWorldJob'. The 'General' tab is selected. The job is currently 'Enabled' (indicated by a blue toggle switch). In the 'Description' field, the text 'HelloWorldJob' is entered. Below the description, there are several optional checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. An 'Advanced' dropdown menu is visible. The 'Source Code Management' section shows 'None' selected, with 'Git' as an alternative option. At the bottom of the configuration page are 'Save' and 'Apply' buttons.

- Navigate back to the Dashboard and you will see our 1st job in queue ready for the build stage.
- Click on “Build Now”
- After a few seconds you will see the the build output under the “build history”
- The console ouput should display a green checkmark indicating that the build was SUCCESSFUL.
- You can see that the echo “Hello World” and uptime commands were properly executed!

The screenshot shows the Jenkins Project HelloWorldJob dashboard. On the left, there is a sidebar with various options: Status, Changes, Workspace, Build Now (which has a green arrow pointing to it), Configure, Delete Project, and Rename. In the center, the title is "Project HelloWorldJob". To the right, there are links for "Edit description" and "Disable Project". Below the title, there is a section titled "Permalinks". At the bottom, there is a "Build History" section with a "trend" dropdown and a message "No builds".

The screenshot shows the Jenkins Console Output for build #1 of the HelloWorldJob. The sidebar on the left includes Status, Changes, Console Output (which is selected and highlighted with a grey background), View as plain text, Edit Build Information, Delete build '#1', and Next Build. The main content area shows the console output log:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorldJob
[HelloWorldJob] $ /bin/sh -xe /tmp/jenkins13823842254370167628.sh
+ echo 'Hello World'
Hello World
+ uptime
16:45:33 up 37 min, 1 user, load average: 0.03, 0.03, 0.06
Finished: SUCCESS
```

## Step-6: Integrating git

## Source Code Management

None

Git ?

### Repositories ?

#### Repository URL ?

`https://github.com/yankils/hello-world.git`



#### Credentials ?

- none -



Add ▼

#### Advanced ▼

Add Repository

### Branches to build ?

#### Branch Specifier (blank for 'any') ?

`*/master`



Add Branch

**Save**

Apply

Under Dashboard, you will see a new job for Git.

Click on “Build Now” if the build output shows a green checkmark indicator, that means our build was a SUCCESS!

The screenshot shows the Jenkins interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and various icons. Below it, the breadcrumb navigation shows: Dashboard > PullCodeFromGitHub > #1 > Console Output. On the left, a sidebar has links for Status, Changes, Console Output (which is selected and highlighted in grey), View as plain text, Edit Build Information, Delete build '#1', and Git Build Data. The main content area is titled 'Console Output' with a green checkmark icon. It displays the build log:

```

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/PullCodeFromGitHub
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/yankils/hello-world.git
> git init /var/lib/jenkins/workspace/PullCodeFromGitHub # timeout=10
Fetching upstream changes from https://github.com/yankils/hello-world.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/yankils/hello-world.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/yankils/hello-world.git #
timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* #
timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision aacc9fa39ba3ca8f46cf0019f35ce4511287375
(refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f aacc9fa39ba3ca8f46cf0019f35ce4511287375 # timeout=10
Commit message: "updated pom.xml file"
First time build. Skipping changelog.
Finished: SUCCESS

```

## Step 7: Installing and Integrating Maven to Jenkins

Maven automates the process of building and managing code in both software development as well as in DevOps. Maven manages the build, testing, and packaging of code, making it easy for DevOps teams to produce consistent, reliable, and repeatable builds.

```
[root@ip-172-31-23-228 ~]# cd /opt
[root@ip-172-31-23-228 opt]# wget https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
--2023-08-28 17:18:40-- https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9336368 (8.9M) [application/x-gzip]
Saving to: 'apache-maven-3.9.4-bin.tar.gz'

100%[=====] 9,336,368  --.-K/s   in 0.1s

2023-08-28 17:18:40 (81.3 MB/s) - 'apache-maven-3.9.4-bin.tar.gz' saved [9336368/9336368]
```

After installation, verify that Maven was indeed installed by list the contents of the directory: If you see the maven dependency files listed then you know it was successfully installed.

```
[root@ip-172-31-23-228 opt]# ll
total 9120
-rw-r--r-- 1 root root 9336368 Aug  3 07:56 apache-maven-3.9.4-bin.tar.gz
drwxr-xr-x 4 root root      33 Aug 22 18:26 aws
drwxr-xr-x 2 root root      6 Aug 16 2018 rh
[root@ip-172-31-23-228 opt]# tar -xvzf apache-maven-3.9.4-bin.tar.gz
apache-maven-3.9.4/README.txt
apache-maven-3.9.4/LICENSE
apache-maven-3.9.4/NOTICE
apache-maven-3.9.4/lib/
apache-maven-3.9.4/lib/aopalliance.license
apache-maven-3.9.4/lib/commons-cli.license
apache-maven-3.9.4/lib/commons-codec.license
apache-maven-3.9.4/lib/commons-lang3.license
apache-maven-3.9.4/lib/failureaccess.license
apache-maven-3.9.4/lib/guava.license
apache-maven-3.9.4/lib/guice.license
apache-maven-3.9.4/lib/httpclient.license
apache-maven-3.9.4/lib/httpcore.license
apache-maven-3.9.4/lib/jansi.license
apache-maven-3.9.4/lib/javax.annotation-api.license
apache-maven-3.9.4/lib/javax.inject.license
apache-maven-3.9.4/lib/jcl-over-slf4j.license
apache-maven-3.9.4/lib/org.eclipse.sisu.inject.license
apache-maven-3.9.4/lib/org.eclipse.sisu.plexus.license
apache-maven-3.9.4/lib/plexus-cipher.license
apache-maven-3.9.4/lib/plexus-component-annotations.license
apache-maven-3.9.4/lib/plexus-interpolation.license
apache-maven-3.9.4/lib/plexus-sec-dispatcher.license
apache-maven-3.9.4/lib/plexus-utils.license
apache-maven-3.9.4/lib/slf4j-api.license
apache-maven-3.9.4/boot/
apache-maven-3.9.4/boot/plexus-classworlds.license
apache-maven-3.9.4/lib/jansi-native/
apache-maven-3.9.4/lib/jansi-native/Windows/
apache-maven-3.9.4/lib/jansi-native/Windows/x86/
apache-maven-3.9.4/lib/jansi-native/Windows/x86_64/
apache-maven-3.9.4/lib/jansi-native/Windows/x86/jansi.dll
apache-maven-3.9.4/lib/jansi-native/Windows/x86_64/jansi.dll
apache-maven-3.9.4/bin/m2.conf
apache-maven-3.9.4/bin/mvn.cmd
apache-maven-3.9.4/bin/mvnDebug.cmd
apache-maven-3.9.4/bin/mvn
apache-maven-3.9.4/bin/mvnDebug
```

You can choose to keep the default maven file name that is highlighted in blue or change the name of the file. I'm going to change the name to just "maven" to keep things simple. To change the name use the "mv" command.

```
[root@ip-172-31-23-228 opt]# mv apache-maven-3.9.4 maven
[root@ip-172-31-23-228 opt]# ll
total 9120
-rw-r--r-- 1 root root 9336368 Aug  3 07:56 apache-maven-3.9.4-bin.tar.gz
drwxr-xr-x 4 root root      33 Aug 22 18:26 aws
drwxr-xr-x 6 root root      99 Aug 28 17:20 maven
drwxr-xr-x 2 root root      6 Aug 16 2018 rh
[root@ip-172-31-23-228 opt]# cd maven
[root@ip-172-31-23-228 maven]# ll
total 36
drwxr-xr-x 2 root root    97 Aug 28 17:20 bin
drwxr-xr-x 2 root root    76 Aug 28 17:20 boot
drwxr-xr-x 3 root root    63 Jul 26 09:37 conf
drwxr-xr-x 4 root root   4096 Aug 28 17:20 lib
-rw-r--r-- 1 root root 18945 Jul 26 09:37 LICENSE
-rw-r--r-- 1 root root  5034 Jul 26 09:37 NOTICE
-rw-r--r-- 1 root root  2533 Jul 26 09:37 README.txt
```

## Execute Maven server

## Check maven version

mvn – v

```
...c2-54-189-134-221.us-west-2.compute.amazonaws.com ...4-189-134-221.us-west-2.compute.amazonaws.com +
[|root@ip-172-31-23-228 ~]# mvn -v
Apache Maven 3.9.4 (dfbb324ad4a7c8fb0bf182e6d91b0ae20e3d2dd9)
Maven home: /opt/maven
Java version: 11.0.20, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.20.0.8
-1.amzn2.0.1.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.186-179.751.amzn2.x86_64", arch: "amd64", family: "unix"
[root@ip-172-31-23-228 ~]#
```

## Step-8: Maven Plugin for Jenkins:

- With Maven server installed, it is time to setup the Maven plugin in the Jenkins server to complete the integration
- Navigate back to the Jenkins Dashboard, select “Manage Jenkins” option, and select “Plugins”.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected and highlighted in blue), and 'My Views'. The main area is titled 'Manage Jenkins' and has a search bar. A yellow banner at the top right says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)'. Below this, there's a 'System Configuration' section with several options: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), and 'Security' (Secure Jenkins; define who is allowed to access/use the system). The 'Plugins' option is highlighted with a red box. At the bottom, there are sections for 'Credentials' (Configure credentials), 'Credential Providers' (Configure the credential providers and types), and 'Users' (Create/delete/modify users that can log in to this Jenkins).

- Type “Maven” and select Maven Integration. Click on install the plugin.

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with links: Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area has a search bar with 'maven' typed in. Below it, a table lists the 'Maven Integration' plugin. The table has columns for 'Install', 'Name', and 'Released'. The 'Maven Integration' row shows version 3.23, Build Tools category, a brief description, and a 'Released' date of 23 days ago. An 'Install' button is visible at the top right of the table.

This screenshot is identical to the one above, showing the Jenkins Plugins page with the 'Maven Integration' plugin listed and the 'Install' button highlighted.

Under Manage Jenkins, select the “Global Configuration” tab. Since we successfully installed JDK and Maven, you will see the options referencing JDK and Maven.

Name-java-11

JAVA\_HOME= /usr/lib/jvm/java-11-openjdk-11.0.20.0.8-1amzn2.0.1x86\_64 (Java home directory)

Deselect the “install automatically” option. We already have Java installed.

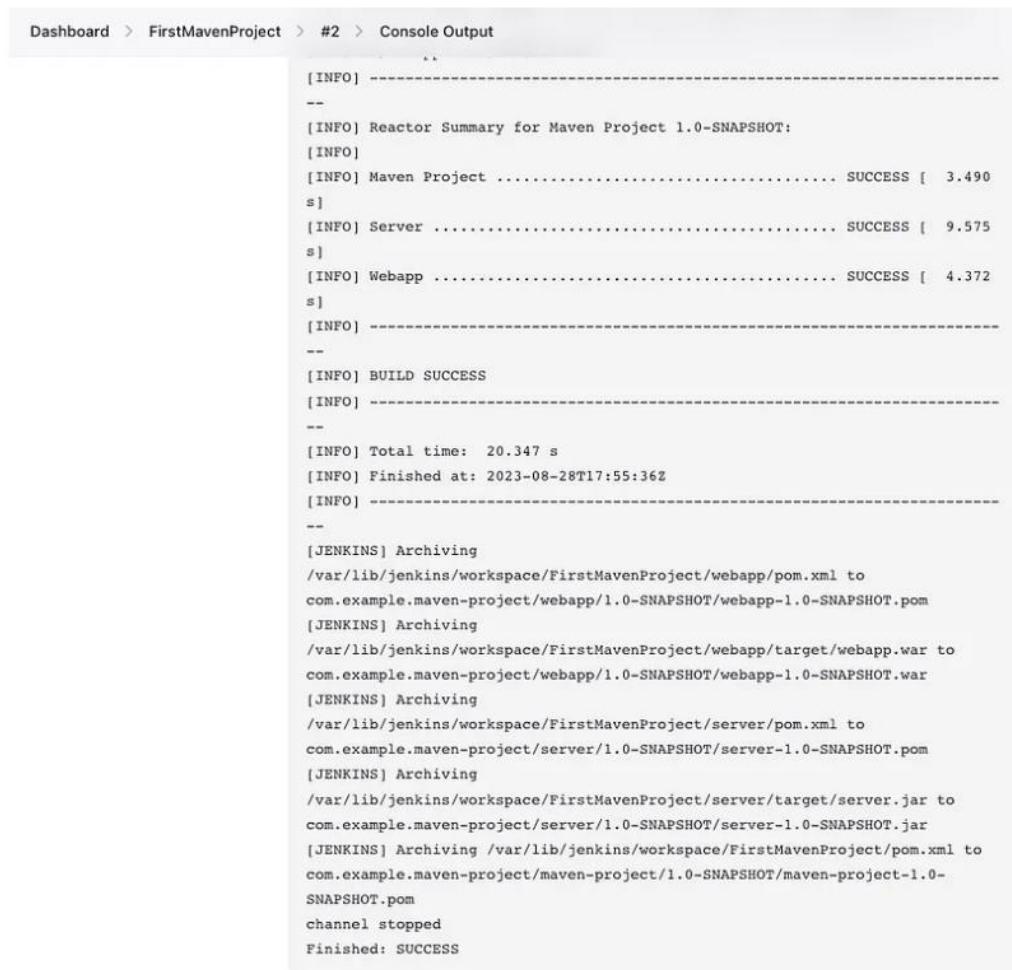
**Step-9:** Navigate to the Jenkins dashboard, select “create item”. We are going to test our CI/CD build by setting our first Maven project.

- Type “FirstMavenProject” in the name box
- Select “Maven project”. Click ok.
- Enter a description for the project build.
- Select “Git” under the Source Code Management. Copy and paste the following git repo URL into the box: <https://github.com/GeorgeBaidooJr9/hello-world>
- Under Build “Goal and Options” type “clean install”. This will result in clean copy of the git repository onto Maven.
- Apply and save

Let’s OFFICIALLY TEST THE BUILD!

- Select “Build” under configure. This build will take a few minutes since there is lots of data stored in the repo.

- After a few minutes you will see details of the build.



The screenshot shows the Jenkins console output for build #2 of the 'FirstMavenProject'. The output is as follows:

```
Dashboard > FirstMavenProject > #2 > Console Output

[INFO] -----
-- 
[INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
[INFO]
[INFO] Maven Project ..... SUCCESS [ 3.490 s]
[INFO] Server ..... SUCCESS [ 9.575 s]
[INFO] Webapp ..... SUCCESS [ 4.372 s]
[INFO] -----
-- 
[INFO] BUILD SUCCESS
[INFO] -----
-- 
[INFO] Total time: 20.347 s
[INFO] Finished at: 2023-08-28T17:55:36Z
[INFO] -----
-- 
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/webapp/pom.xml to
com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war to
com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/server/pom.xml to
com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving
/var/lib/jenkins/workspace/FirstMavenProject/server/target/server.jar to
com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/pom.xml to
com.example.maven-project/maven-project/1.0-SNAPSHOT/maven-project-1.0-
SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```