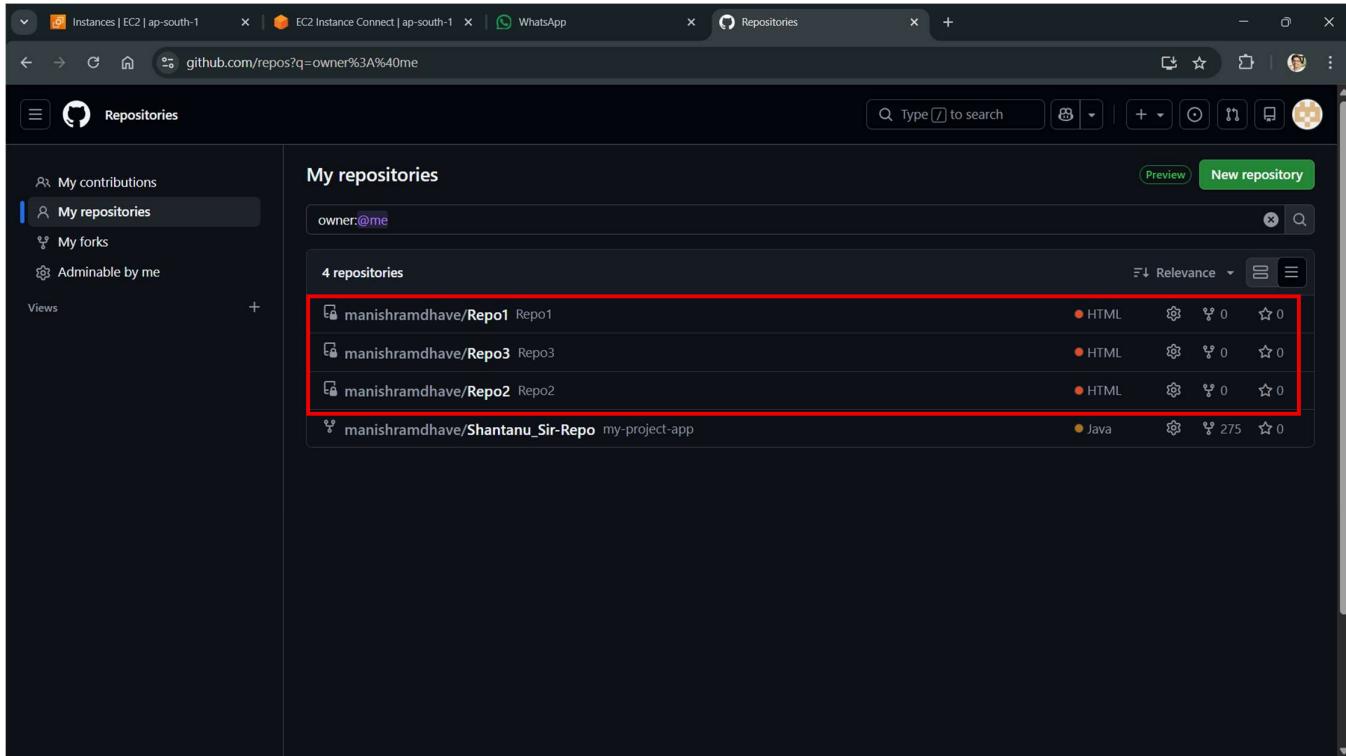


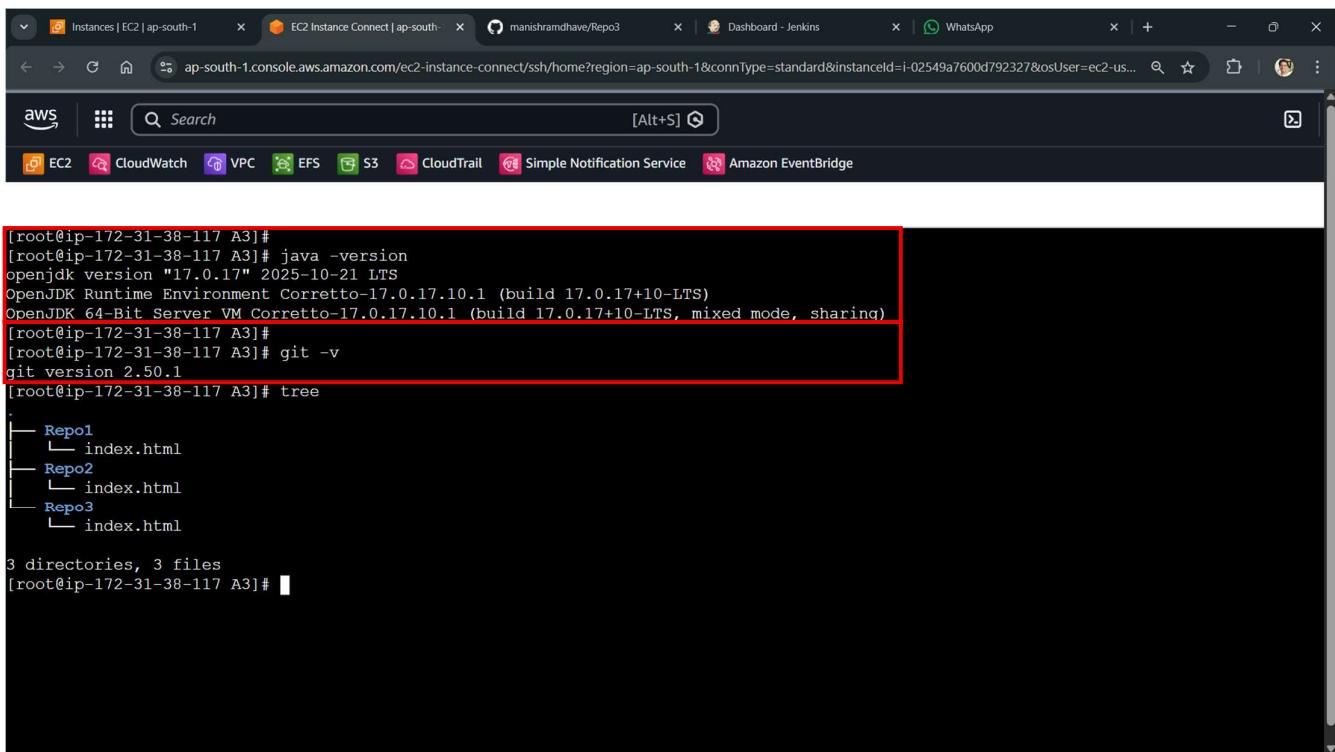
# Jenkins Assignment 3

Step 1: Made three Private Repositories Repo1, Repo2 and Repo3 in GitHub account:



The screenshot shows a browser window with multiple tabs. The active tab is 'github.com/repos?q=owner%3A%40me'. The left sidebar shows navigation options like 'My contributions', 'My repositories' (which is selected), 'My forks', and 'Admirable by me'. The main area is titled 'My repositories' and shows a search bar with 'owner:@me'. Below it, a table lists four repositories: 'Repo1', 'Repo2', and 'Repo3' under the heading 'manishramdhave/Repo1', 'manishramdhave/Repo2', and 'manishramdhave/Repo3' respectively. A fourth repository, 'Shantanu\_Sir-Repo', is listed under 'manishramdhave/Shantanu\_Sir-Repo'. Each repository entry includes a preview icon, language (HTML), license (MIT), stars (0), and issues (0). A red box highlights the first three repositories.

Step 2: Launched an instance and installed Java -17 and Git on it:



The screenshot shows a terminal window on an AWS EC2 instance. The top bar shows tabs for 'Instances | EC2 | ap-south-1', 'EC2 Instance Connect | ap-south-1', 'manishramdhave/Repo3', 'Dashboard - Jenkins', and 'WhatsApp'. The terminal itself has a red box highlighting the command output. It shows the user has installed Java 17 and Git, and has created three repositories named 'Repo1', 'Repo2', and 'Repo3', each containing an 'index.html' file. The final command shown is 'tree' which displays the directory structure.

```
[root@ip-172-31-38-117 A3]# [root@ip-172-31-38-117 A3]# java -version openjdk version "17.0.17" 2025-10-21 LTS OpenJDK Runtime Environment Corretto-17.0.17.10.1 (build 17.0.17+10-LTS) OpenJDK 64-Bit Server VM Corretto-17.0.17.10.1 (build 17.0.17+10-LTS, mixed mode, sharing) [root@ip-172-31-38-117 A3]# [root@ip-172-31-38-117 A3]# git -v git version 2.50.1 [root@ip-172-31-38-117 A3]# tree . ├── Repo1 │   └── index.html ├── Repo2 │   └── index.html └── Repo3     └── index.html 3 directories, 3 files [root@ip-172-31-38-117 A3]#
```

### Step 3: Installed Apache-Tomcat-10 and Jenkins on the instance:

The screenshot shows a terminal window on an AWS CloudWatch instance. The user has navigated to the directory /var/www/html and listed its contents. A red box highlights the command 'cd apache-tomcat-10.1.50/' and the resulting directory listing, which includes 'jenkins' and 'jenkins.war' files.

```
[root@ip-172-31-38-117 ~]# cd server
[root@ip-172-31-38-117 server]# ll
total 16
drwxrwxrwx. 9 root root 16384 Dec  2 22:57 apache-tomcat-10.1.50/
[root@ip-172-31-38-117 server]# cd apache-tomcat-10.1.50/
[root@ip-172-31-38-117 apache-tomcat-10.1.50]#
[root@ip-172-31-38-117 apache-tomcat-10.1.50]#
[root@ip-172-31-38-117 apache-tomcat-10.1.50]# cd webapps
[root@ip-172-31-38-117 webapps]#
[root@ip-172-31-38-117 webapps]# ll
total 94056
drwxrwxrwx. 3 root root 16384 Dec  2 22:57 ROOT
drwxrwxrwx. 16 root root 16384 Dec  2 22:57 docs
drwxrwxrwx. 7 root root 99 Dec  2 22:57 examples
drwxrwxrwx. 6 root root 79 Dec  2 22:57 manager
drwxr-x---. 10 root root 16384 Jan 27 17:35 jenkins
-rwxrwxrwx. 1 root root 96260165 Jan 21 09:32 jenkins.war
drwxrwxrwx. 6 root root 114 Dec  2 22:57 manager
[root@ip-172-31-38-117 webapps]#
```

### Step 4: Pushed three different index.html files using the branches 2026Q1, 2026Q2 and 2026Q3 respectively:

The screenshot shows a terminal window on an AWS CloudWatch instance. The user has navigated to three separate repositories (Rep01, Rep02, and Rep03) and pushed index.html files to the 2026Q1, 2026Q2, and 2026Q3 branches respectively. Red boxes highlight each repository's directory and the 'git log' output showing the commit messages and dates.

```
[root@ip-172-31-38-117 A3]# cd Rep01
[root@ip-172-31-38-117 Rep01]# git log
commit a42a777c322ac1c735ac7b6b1608c2e1lef3eb4 (HEAD -> 2026Q1, origin/2026Q1, master)
Author: root <root@ip-172-31-38-117.ap-south-1.compute.internal>
Date: Tue Jan 27 17:59:58 2026 +0000

    Added index.html in Rep01
[root@ip-172-31-38-117 Rep01]#
[root@ip-172-31-38-117 Rep01]# cd ..
[root@ip-172-31-38-117 A3]# cd Rep02
[root@ip-172-31-38-117 Rep02]# git log
commit c8f31040f1ba719af73e190e76270097b8211cae (HEAD -> 2026Q2, origin/2026Q2, master)
Author: root <root@ip-172-31-38-117.ap-south-1.compute.internal>
Date: Tue Jan 27 18:02:53 2026 +0000

    Added index.html in Rep02
[root@ip-172-31-38-117 Rep02]#
[root@ip-172-31-38-117 Rep02]# cd ..
[root@ip-172-31-38-117 A3]# cd Rep03
[root@ip-172-31-38-117 Rep03]# git log
commit 699fe4681b85786a1bb20d120afe5bc1160d4278 (HEAD -> 2026Q3, origin/2026Q3, master)
Author: root <root@ip-172-31-38-117.ap-south-1.compute.internal>
Date: Tue Jan 27 19:11:42 2026 +0000

    Added index.html in Rep03
[root@ip-172-31-38-117 Rep03]#
```

## Step 5: Launched the Jenkins and created three different Jobs Job1, Job2 and Job3:

The screenshot shows the Jenkins dashboard with three jobs listed in a table:

S	W	Name	Last Success	Last Failure	Last Duration
		Job1	1 hr 20 min #2	N/A	2.3 sec
		Job2	1 hr 30 min #2	N/A	2.3 sec
		Job3	1 hr 17 min #1	N/A	2.2 sec

Below the table, there are buttons for Icon: S M L.

REST API Jenkins 2.541.1

## Step 6: Integrated Git with Jenkins by creating ‘Credentials’ by using the Git Token:

The screenshot shows the Jenkins configuration page for Job1. In the Source Code Management section, the 'Git' option is selected. The 'Repository URL' field contains 'https://github.com/manishramdhave/Repo1.git'. The 'Credentials' dropdown shows 'Git-token/\*\*\*\*\* (Git-token)'. At the bottom, there are 'Save' and 'Apply' buttons.

## Step 7: Enabled ‘GitHub hook Trigger for GITScm polling’ while configuring all three Jenkins Jobs:

The screenshot shows the Jenkins configuration page for Job1. On the left, there's a sidebar with links: General, Source Code Management, Triggers (which is selected and highlighted in grey), Environment, Build Steps, and Post-build Actions. The main content area is titled 'Triggers' with the sub-instruction 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' Below this, there are several checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', and 'GitHub hook trigger for GITScm polling'. The last checkbox is checked and highlighted with a red border. There are also other options like 'Poll SCM'. At the bottom of the triggers section are 'Save' and 'Apply' buttons.

## Step 8: Executed Shell Script Commands whenever a Build is done by the user:

The screenshot shows the Jenkins configuration page for Job1. The sidebar on the left includes General, Source Code Management, Triggers, Environment, Build Steps (selected and highlighted in grey), and Post-build Actions. The main content area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' A single step is listed: 'Execute shell'. A red box highlights the 'Command' input field, which contains the following shell script commands:  
rm -rf /var/www/html/index.html  
cp -r index.html /var/www/html/  
chmod -R 777 /var/www/html/index.html  
service httpd restart

## Step 9: Created an API Connection between Jenkins to GitHub Repositories in ‘Manage Jenkins’ by creating a Secret Text (Credential) using a GitHub Token in Jenkins:

The screenshot shows the Jenkins 'Manage Jenkins' interface with the 'System' configuration selected. Under the 'GitHub' section, a 'GitHub Server' is configured with the name 'GitHub-Server' and the API URL 'https://api.github.com'. A 'Credentials' dropdown menu is open, showing a single entry 'Git' which is highlighted with a red box. Below the credentials, there is a 'Manage hooks' checkbox and an 'Advanced' dropdown. At the bottom are 'Save' and 'Apply' buttons.

## Step 10: Created an API Connection between Jenkins and every GitHub Repositories by using ‘Git Webhooks’ (i.e. setting up Jenkins URL in Git) :

The screenshot shows the GitHub repository settings for 'manishramdhave/Repo1'. In the 'Webhooks' section, a new webhook is being configured. The 'Payload URL' field contains 'http://43.205.228.3:8080/jenkins/github-webhook/'. This field is also highlighted with a red box. Other configuration options shown include 'Content type' set to 'application/json', 'Secret' (empty), 'SSL verification' (Enable SSL verification selected), event triggers ('Just the push event.' selected), and the 'Active' checkbox checked. At the bottom are 'Update webhook' and 'Delete webhook' buttons.

## Result:

- When changes are done in Gir Repo1, Build is triggered by Job1 and the index.html file is hosted which is available in Repo1:

The screenshot shows two browser windows. The left window is a GitHub repository named 'Repo1' where a file 'index.html' has been updated. The commit message is 'Update index.html'. The code in the file is:

```
1 This is Repo1.  
2 Hi there. I Have made changes
```

The right window shows the content of the 'index.html' file as it appears on the web at the URL 43.205.228.3/index.html. The content is:

This is Repo1. Hi there. I Have made changes

- When changes are done in Gir Repo2, Build is triggered by Job2 and the index.html file is hosted which is available in Repo2:

The screenshot shows two browser windows. The left window is a GitHub repository named 'Repo2' where a file 'index.html' has been updated. The commit message is 'Update index.html'. The code in the file is:

```
1 This is Repo2  
2 I have made changes
```

The right window shows the content of the 'index.html' file as it appears on the web at the URL 43.205.228.3/index.html. The content is:

This is Repo2 I have made changes

3. When changes are done in Gir Repo3, Build is triggered by Job3 and the index.html file is hosted which is available in Repo3:

