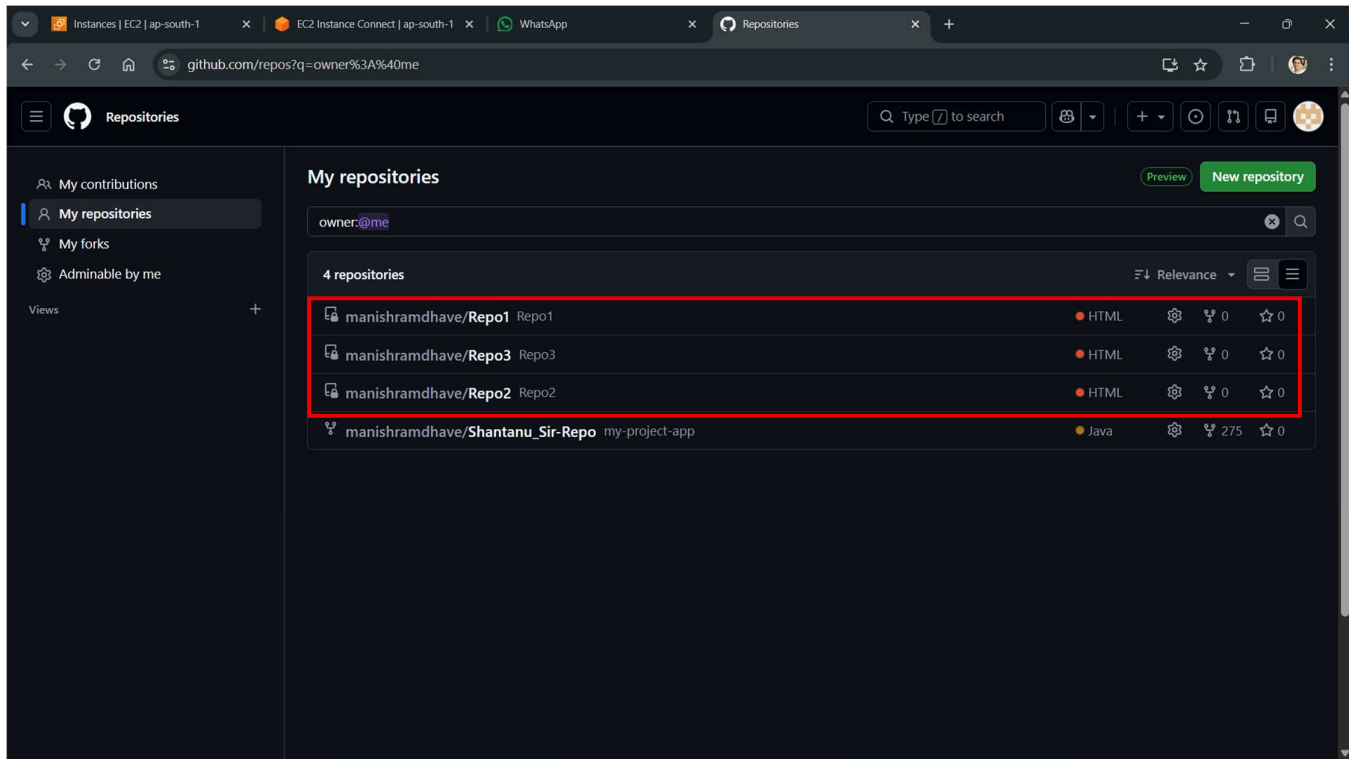
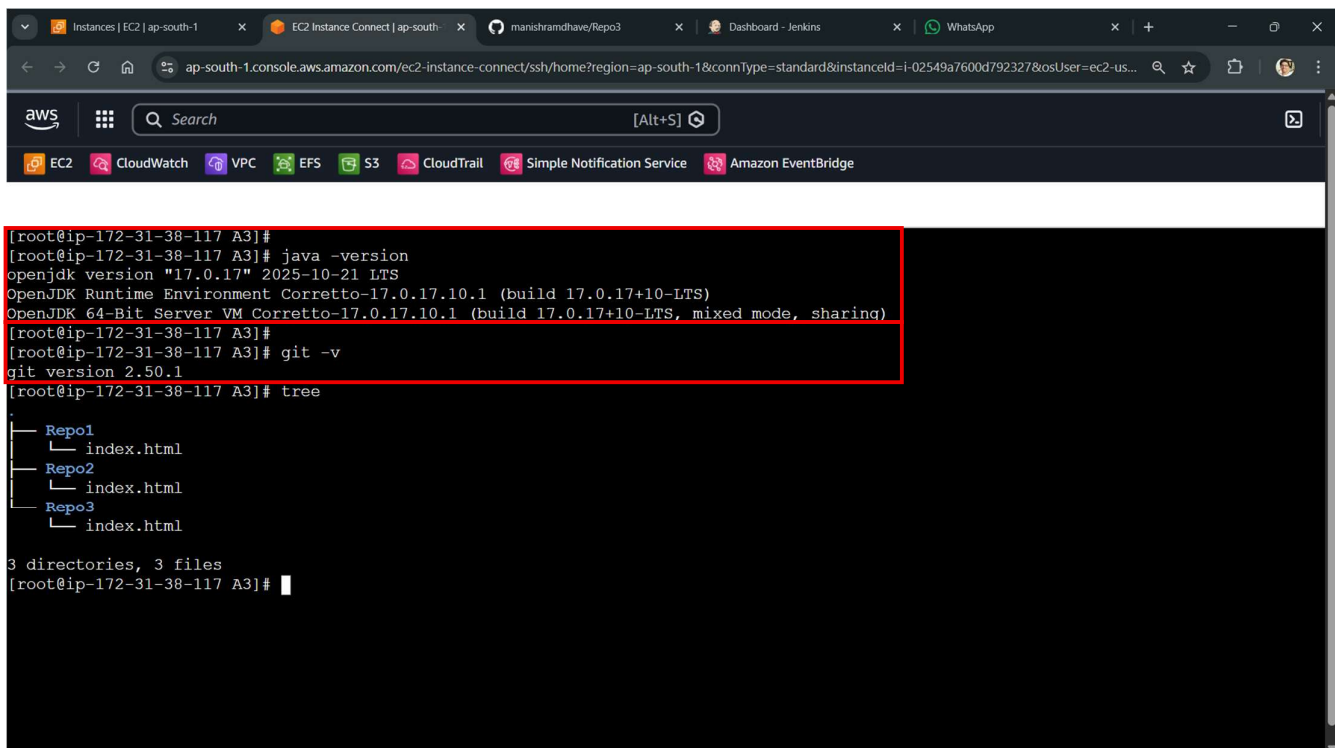


# Jenkins Assignment 3

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



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



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

```
aws
Search [Alt+S]
EC2 CloudWatch VPC EFS S3 CloudTrail Simple Notification Service Amazon EventBridge

[root@ip-172-31-38-117 ~]#
[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
drwxrwxrwx. 16 root root 16384 Dec 2 22:57
drwxrwxrwx. 7 root root 99 Dec 2 22:57
drwxrwxrwx. 6 root root 79 Dec 2 22:57
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
[root@ip-172-31-38-117 webapps]#
```

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

```
aws
Search [Alt+S] Ask Amazon Q
EC2 CloudWatch VPC EFS S3 CloudTrail Simple Notification Service Amazon EventBridge

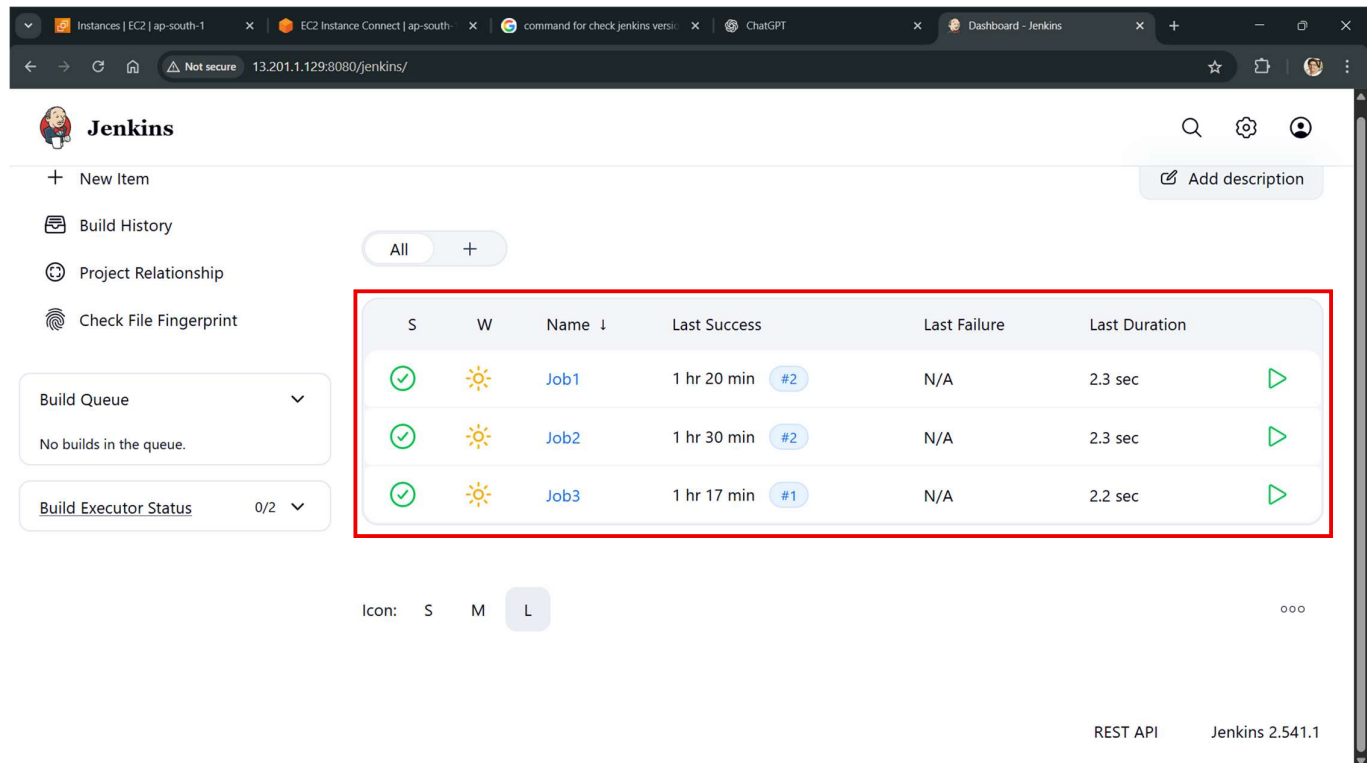
[root@ip-172-31-38-117 A3]# cd Repo1
[root@ip-172-31-38-117 Repo1]# git log
commit a42a77c322ac1c735ac7b6b1608c2e11efb3eb4 (HEAD -> 2026Q1, origin/2026Q1, master)
Author: root <root@ip-172-31-38-117.ap-south-1.compute.internal>
Date: Tue Jan 27 17:59:58 2026 +0000

    Added index.html in Repo1
[root@ip-172-31-38-117 Repo1]#
[root@ip-172-31-38-117 Repo1]# cd ..
[root@ip-172-31-38-117 A3]# cd Repo2
[root@ip-172-31-38-117 Repo2]# 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 Repo2
[root@ip-172-31-38-117 Repo2]#
[root@ip-172-31-38-117 Repo2]# cd ..
[root@ip-172-31-38-117 A3]# cd Repo3
[root@ip-172-31-38-117 Repo3]# 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 Repo3
[root@ip-172-31-38-117 Repo3]#
```

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

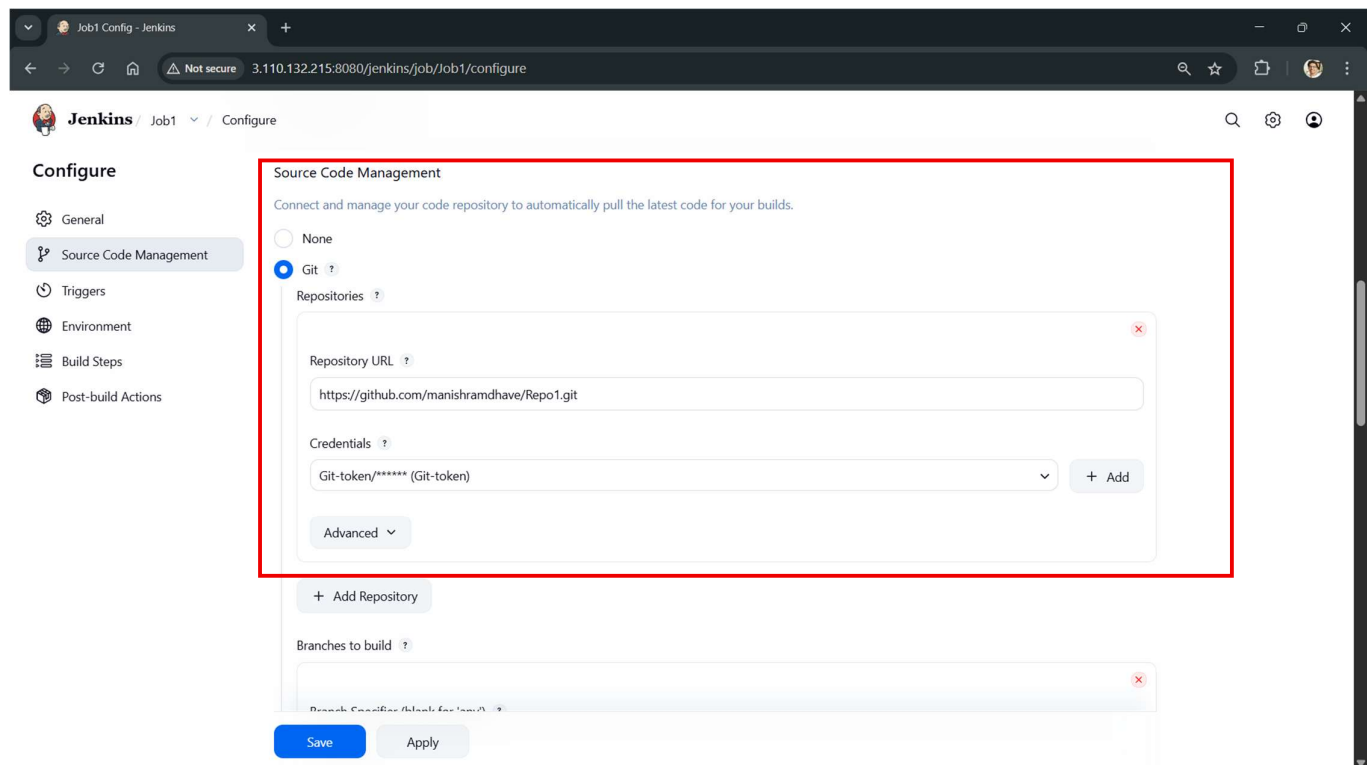


The screenshot shows the Jenkins Dashboard with the following table of jobs:

| 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       |

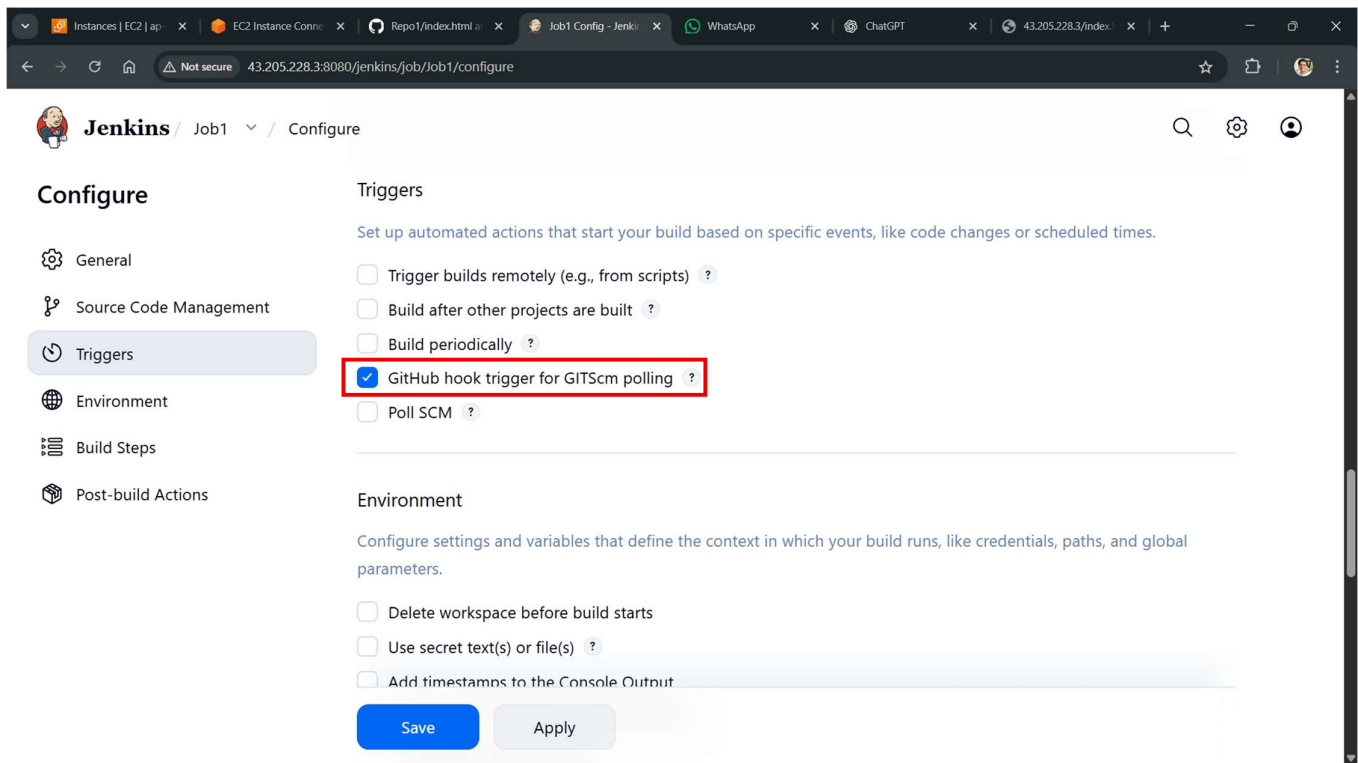
The jobs are highlighted with a red box. The dashboard also shows a 'Build Queue' with 'No builds in the queue.' and a 'Build Executor Status' of '0/2'. The Jenkins version is 2.541.1.

Step 6: Integrated Git with Jenkins by creating 'Credentials' by using the Git Token:



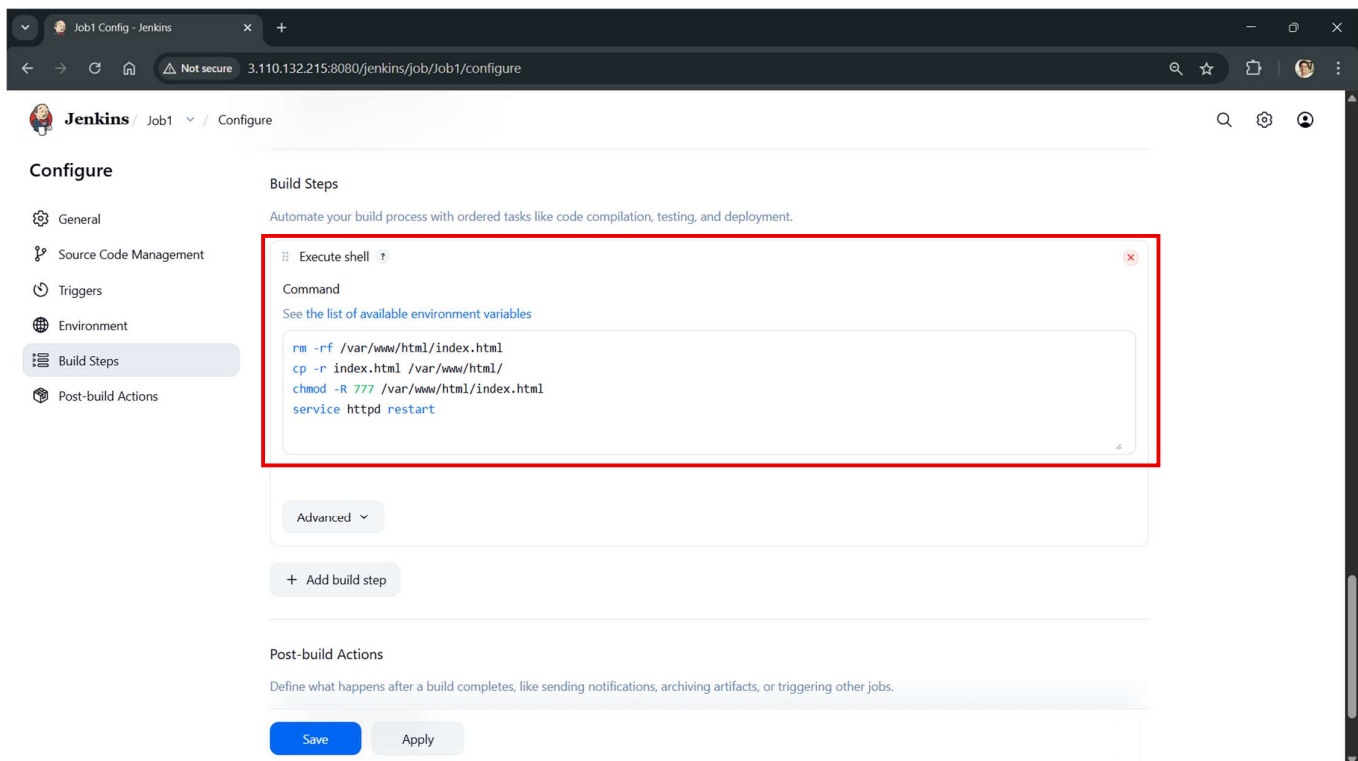
The screenshot shows the Jenkins Job Configuration page for Job1. The 'Source Code Management' section is highlighted with a red box. It shows the 'Git' option selected, with the 'Repository URL' set to 'https://github.com/manishramdhare/Repo1.git' and the 'Credentials' set to 'Git-token/\*\*\*\*\* (Git-token)'. The 'Advanced' dropdown is also visible.

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



The screenshot shows the Jenkins Job Configuration page for 'Job1', specifically the 'Triggers' tab. The left sidebar lists configuration sections: General, Source Code Management, Triggers (selected), Environment, Build Steps, and Post-build Actions. The 'Triggers' section contains several options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked and highlighted with a red box), and 'Poll SCM'. Below the triggers, the 'Environment' section is visible with options like 'Delete workspace before build starts', 'Use secret text(s) or file(s)', and 'Add timestamps to the Console Output'. At the bottom are 'Save' and 'Apply' buttons.

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

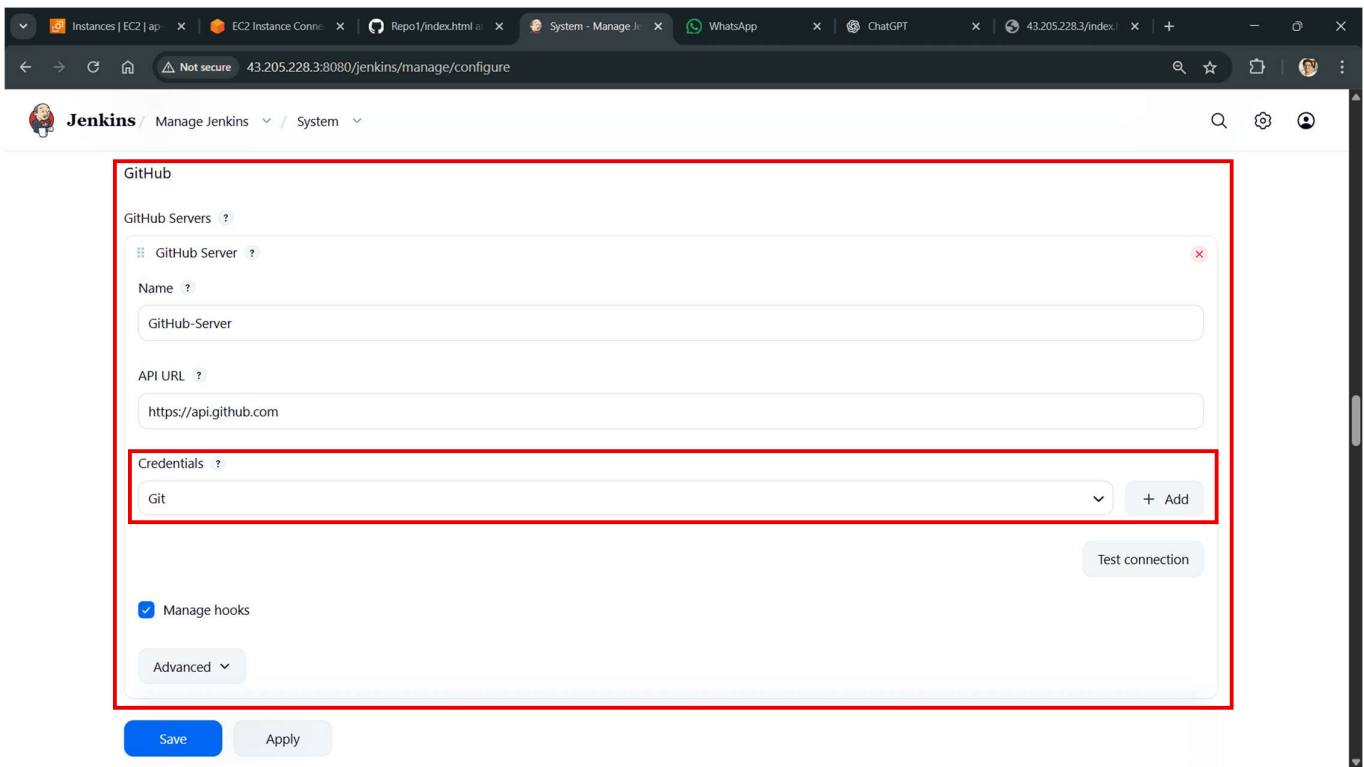


The screenshot shows the Jenkins Job Configuration page for 'Job1', specifically the 'Build Steps' tab. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (selected), and Post-build Actions. The 'Build Steps' section contains a single step named 'Execute shell'. The 'Command' field for this step contains the following shell 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
```

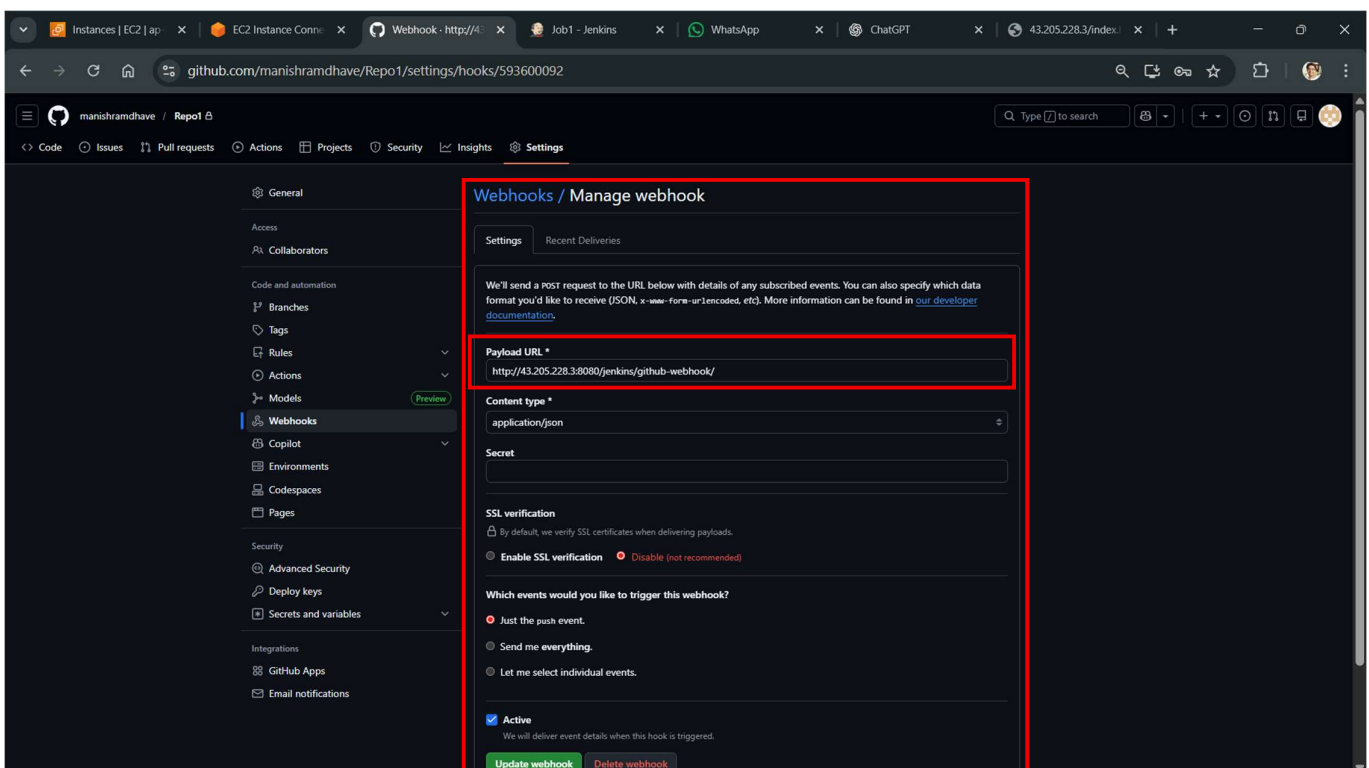
 This entire step configuration is highlighted with a red box. Below the build steps, the 'Post-build Actions' section is visible. At the bottom are 'Save' and 'Apply' buttons.

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' page with the 'System' tab selected. A red box highlights the 'GitHub' configuration section. Inside this section, the 'GitHub Servers' list contains one entry, 'GitHub Server'. The 'Name' field is 'GitHub-Server' and the 'API URL' is 'https://api.github.com'. Below this, the 'Credentials' dropdown is set to 'Git', with an '+ Add' button next to it. A 'Test connection' button is located to the right of the credentials dropdown. At the bottom of the configuration box, there is a checked checkbox for 'Manage hooks' and an 'Advanced' dropdown menu. Below the configuration box, there are 'Save' and 'Apply' buttons.

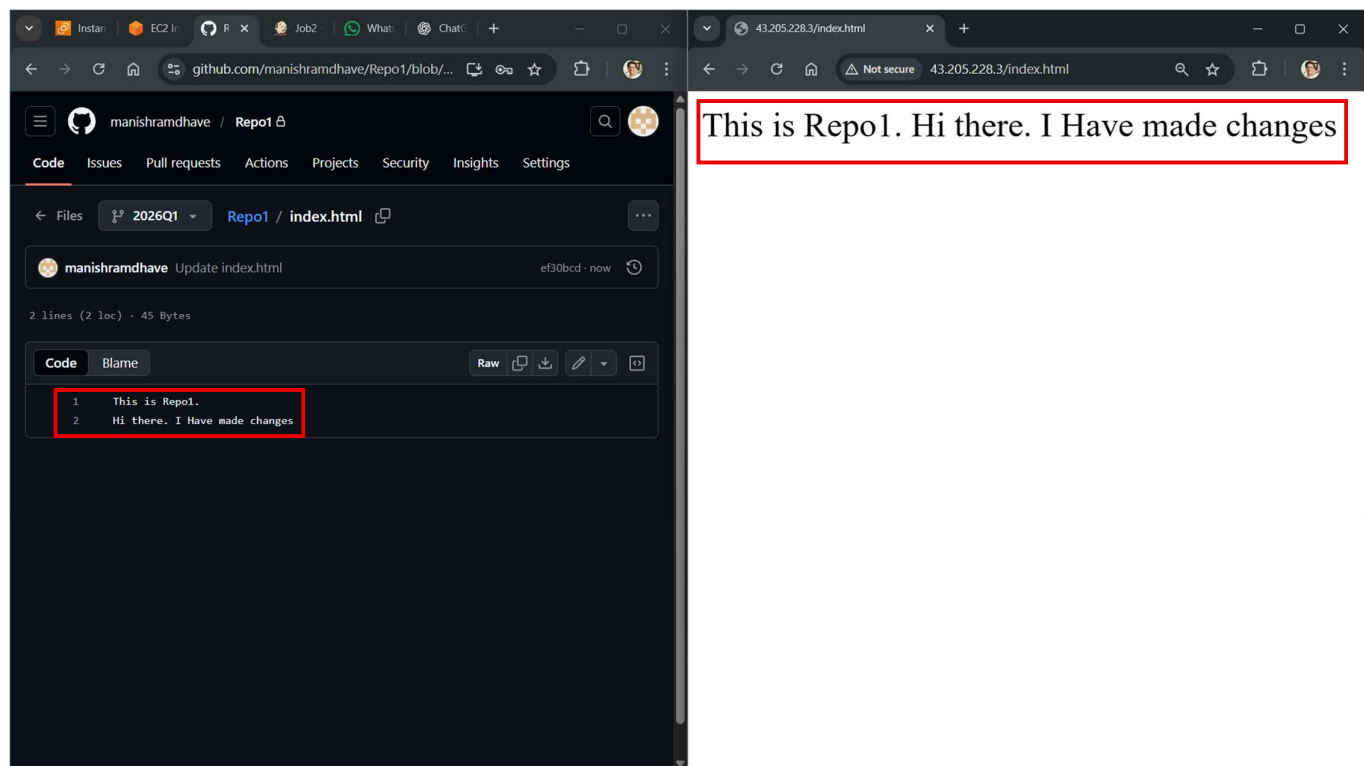
Step 10: Created an API Connection between Jenkins and every GitHub Repositories by using ‘Git Webhooks’ (i.e. setting up Jenkins URL in Git) :



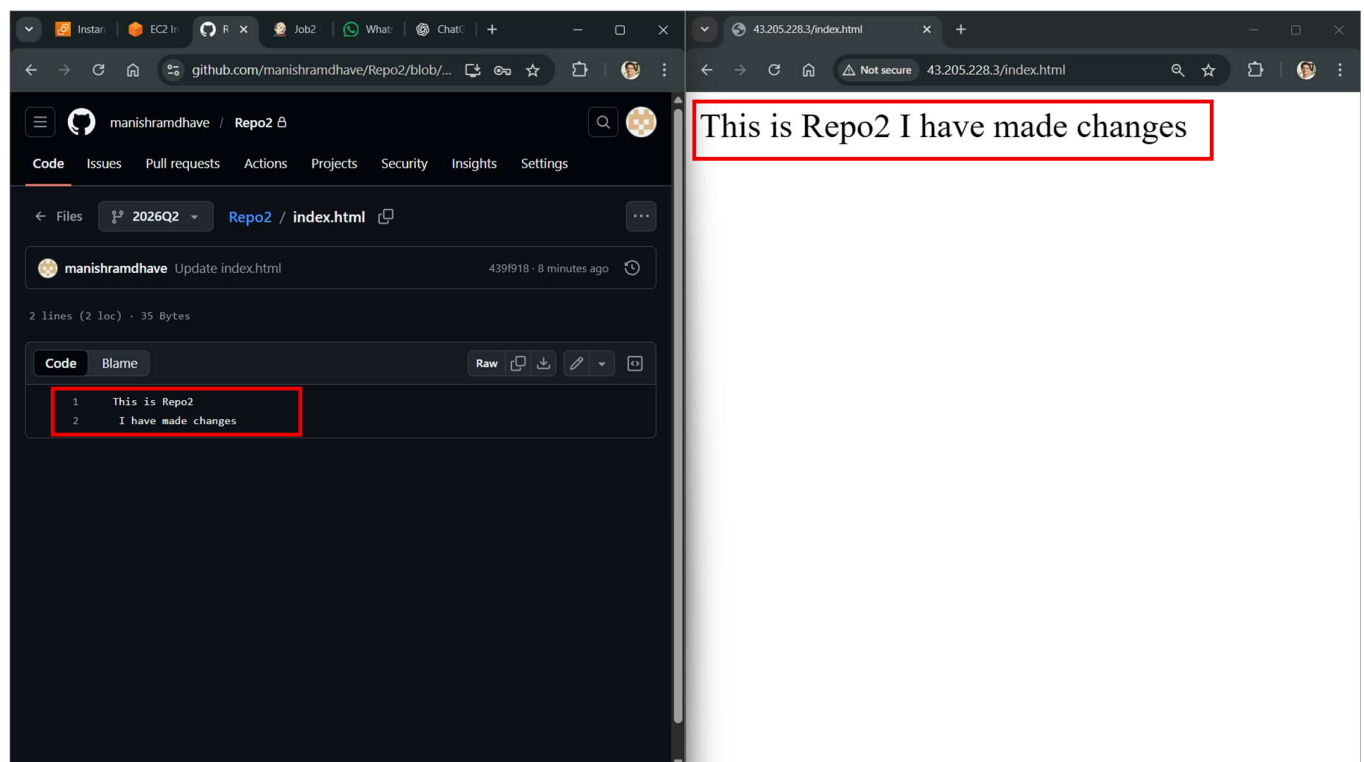
The screenshot shows the GitHub repository settings page for 'manishramdhare / Repo1'. The 'Settings' tab is selected, and the 'Webhooks / Manage webhook' section is highlighted with a red box. The 'Settings' sub-tab is active. The 'Payload URL' is set to 'http://43.205.228.3:8080/jenkins/github-webhook/'. The 'Content type' is set to 'application/json'. The 'Secret' field is empty. Under 'SSL verification', the 'Enable SSL verification' option is selected, with a note that it is not recommended. Under 'Which events would you like to trigger this webhook?', the 'Just the push event.' option is selected. The 'Active' checkbox is checked, with a note that event details will be delivered when the hook is triggered. At the bottom of the configuration box, there are 'Update webhook' and 'Delete webhook' buttons.

## Result:

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



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



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:

