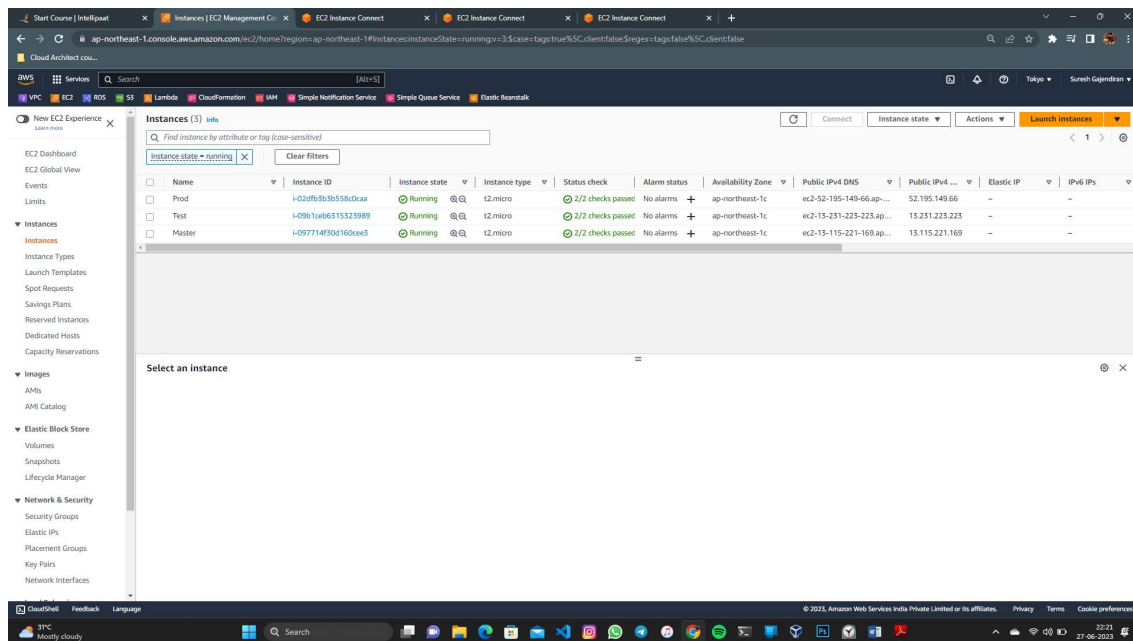


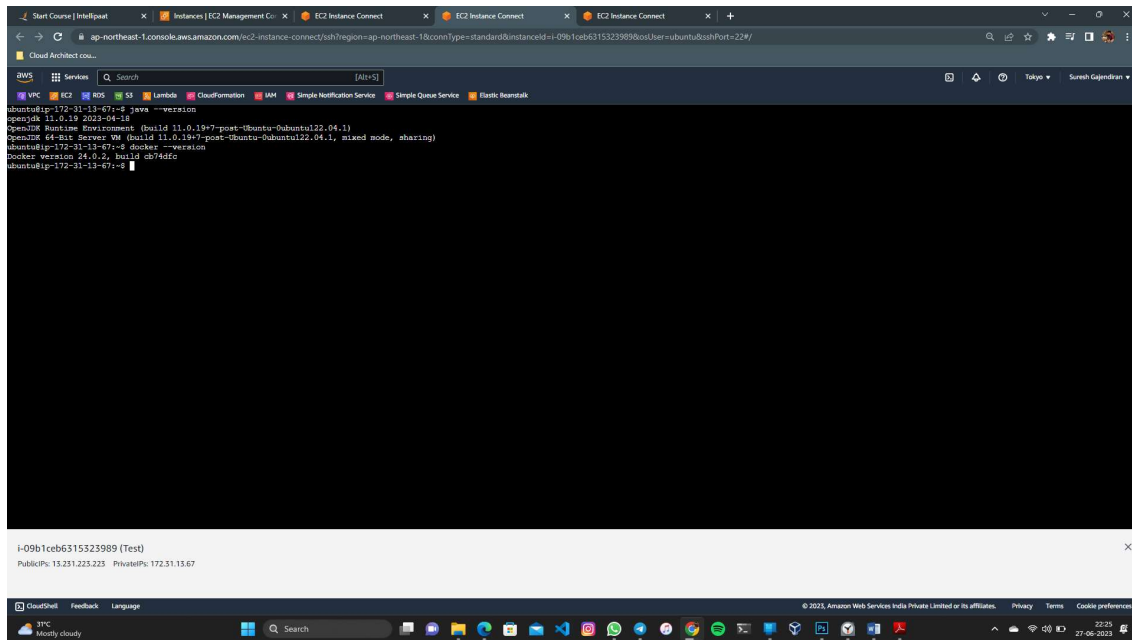
Project – Capstone 1

1. Install the necessary software on the machines using a configuration Management tool.
2. Git workflow has to be implemented.
3. Code Build should automatically be triggered once a commit is made to Master branch or develop branch.
 - a. If a commit is made to master branch, test and push to prod
 - b. If a commit is made to develop branch, just test the product, do not push to prod.
4. The code should be containerized with the help of a Docker file. The Docker file should be built every time there is a push to GitHub. Use the Following pre-built container for your application: hshar/webapp
The code should reside in '/var/www/html'



1. Installed ansible java and Jenkins in master machine

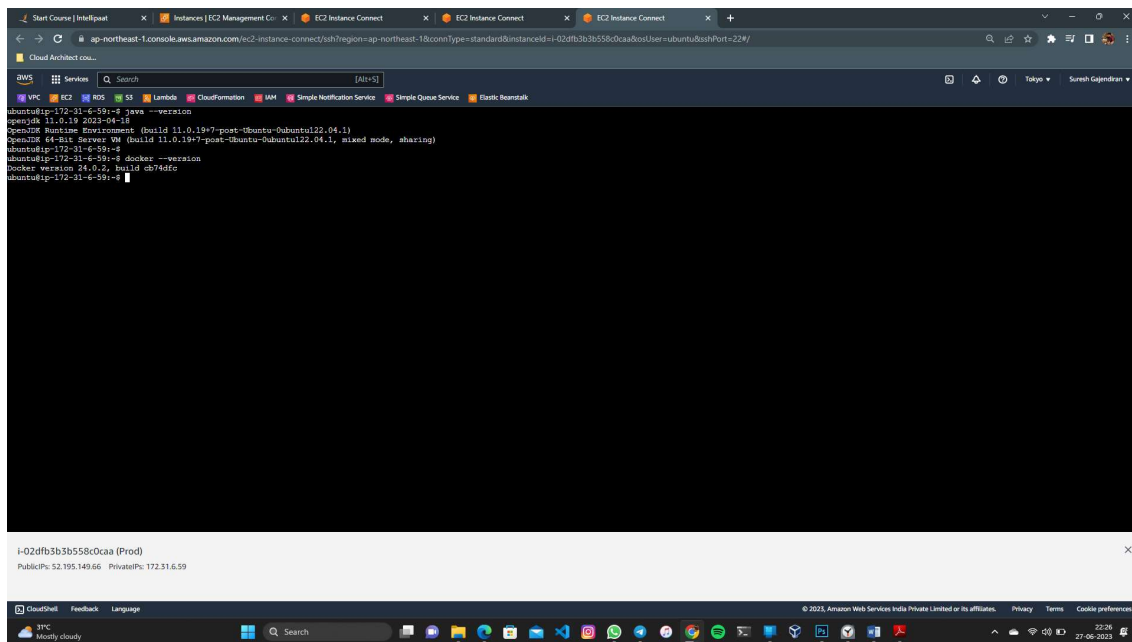
2. Installed Java and Docker in both Test and Prod machine



The screenshot shows the AWS Management Console with a terminal window open for an EC2 instance named 'i-09b1ceb6315323989 (Test)'. The terminal output shows the following commands and results:

```
ubuntu@ip-172-31-13-67:~$ java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu22.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu22.04.1, mixed mode, sharing)
ubuntu@ip-172-31-13-67:~$ docker --version
Docker version 24.0.2, build cb74dfc
ubuntu@ip-172-31-13-67:~$
```

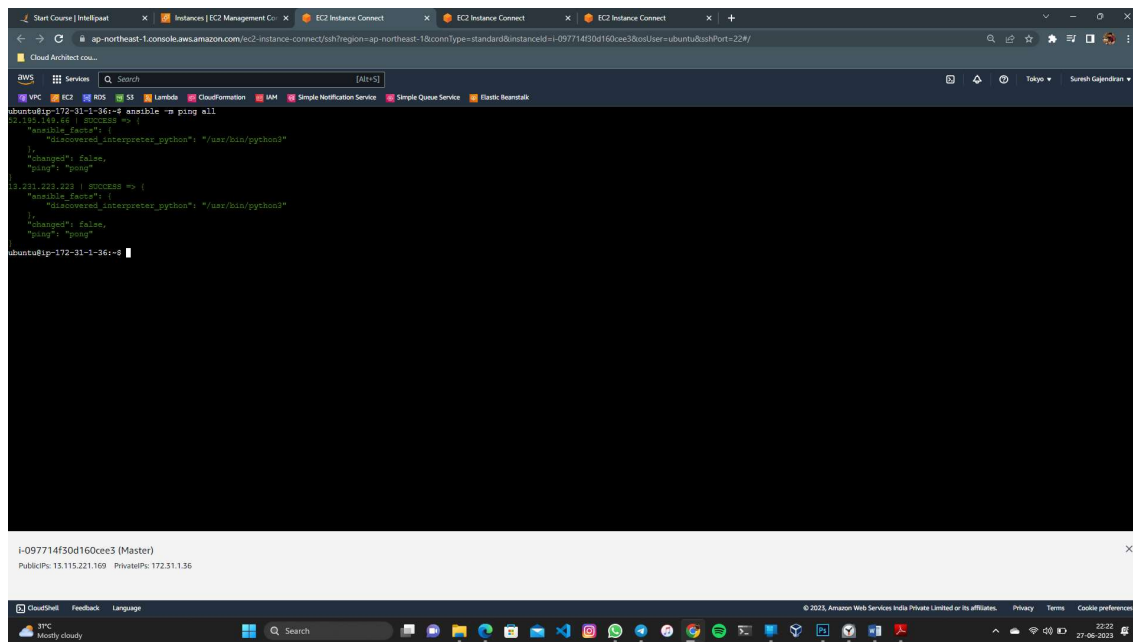
The instance details at the bottom show the Public IP as 13.231.223.223 and the Private IP as 172.31.13.67.



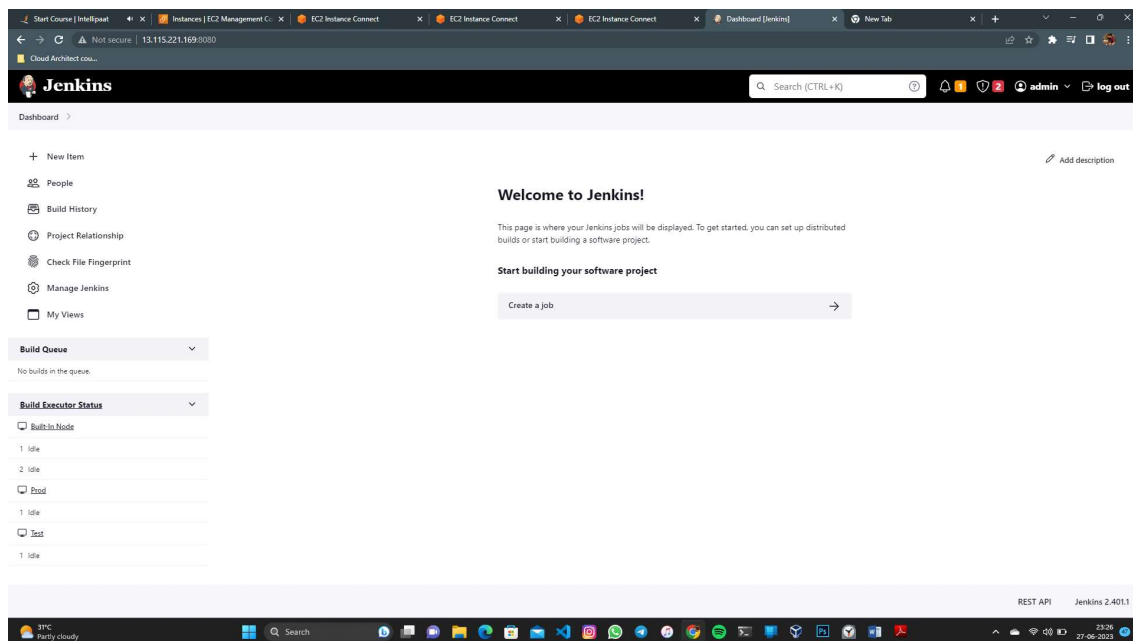
The screenshot shows the AWS Management Console with a terminal window open for an EC2 instance named 'i-02dfb3b3b5580caa (Prod)'. The terminal output shows the following commands and results:

```
ubuntu@ip-172-31-6-59:~$ java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu22.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu22.04.1, mixed mode, sharing)
ubuntu@ip-172-31-6-59:~$ docker --version
Docker version 24.0.2, build cb74dfc
ubuntu@ip-172-31-6-59:~$
```

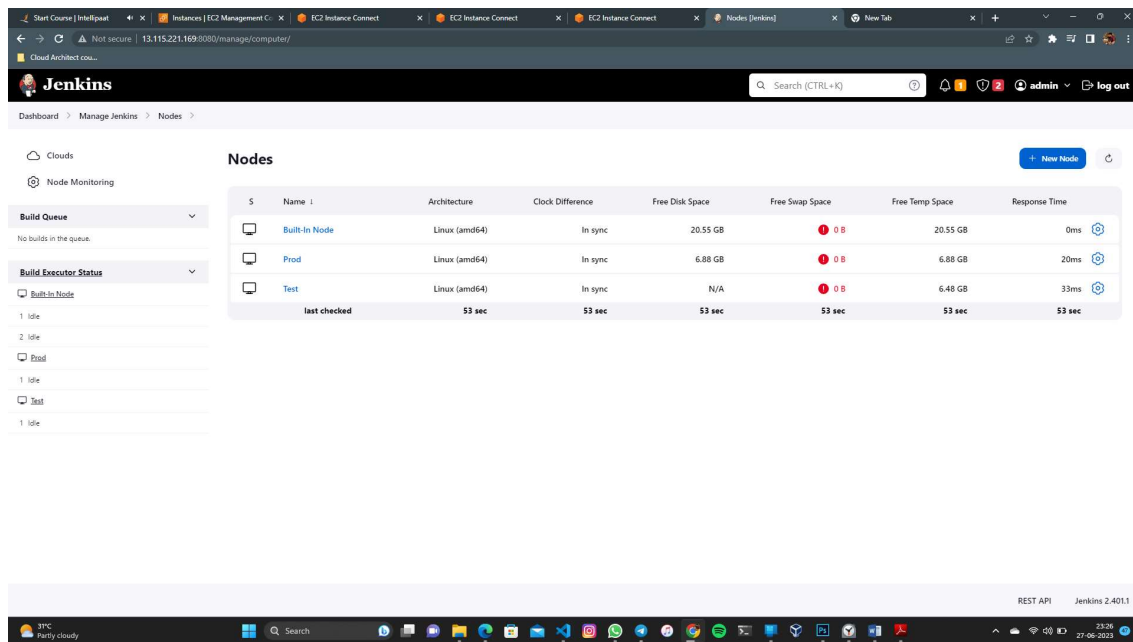
The instance details at the bottom show the Public IP as 52.195.149.66 and the Private IP as 172.31.6.59.



Both test and prod machines are connected with master machine



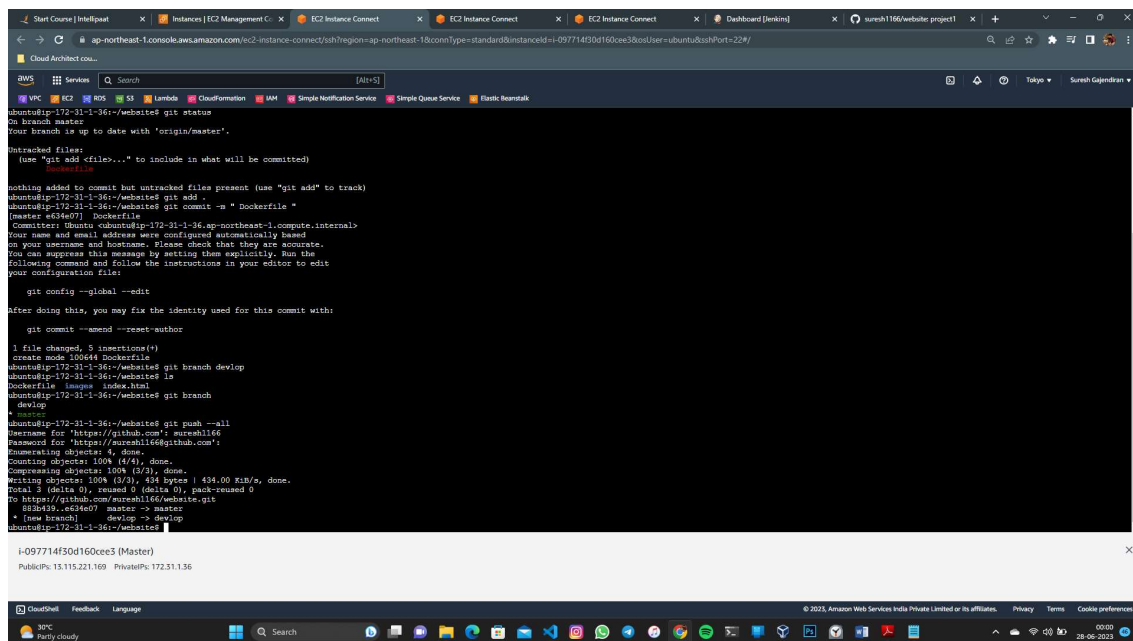
Nodes are created in jenkins



The screenshot shows the Jenkins web interface. The top navigation bar includes 'Dashboard', 'Manage Jenkins', and 'Nodes'. The 'Nodes' section is active, displaying a table of nodes. The table has columns for 'S', 'Name', 'Architecture', 'Clock Difference', 'Free Disk Space', 'Free Swap Space', 'Free Temp Space', and 'Response Time'. There are three nodes listed: 'Built-In Node', 'Prod', and 'Test'. Each node has a status icon (a green circle with a white '0' and a red 'B') and a 'last checked' timestamp of '53 sec'.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	20.55 GB	0 B	20.55 GB	0ms
	Prod	Linux (amd64)	In sync	6.88 GB	0 B	6.88 GB	20ms
	Test	Linux (amd64)	In sync	N/A	0 B	6.48 GB	33ms
	last checked	53 sec	53 sec	53 sec	53 sec	53 sec	53 sec

Two branches are created

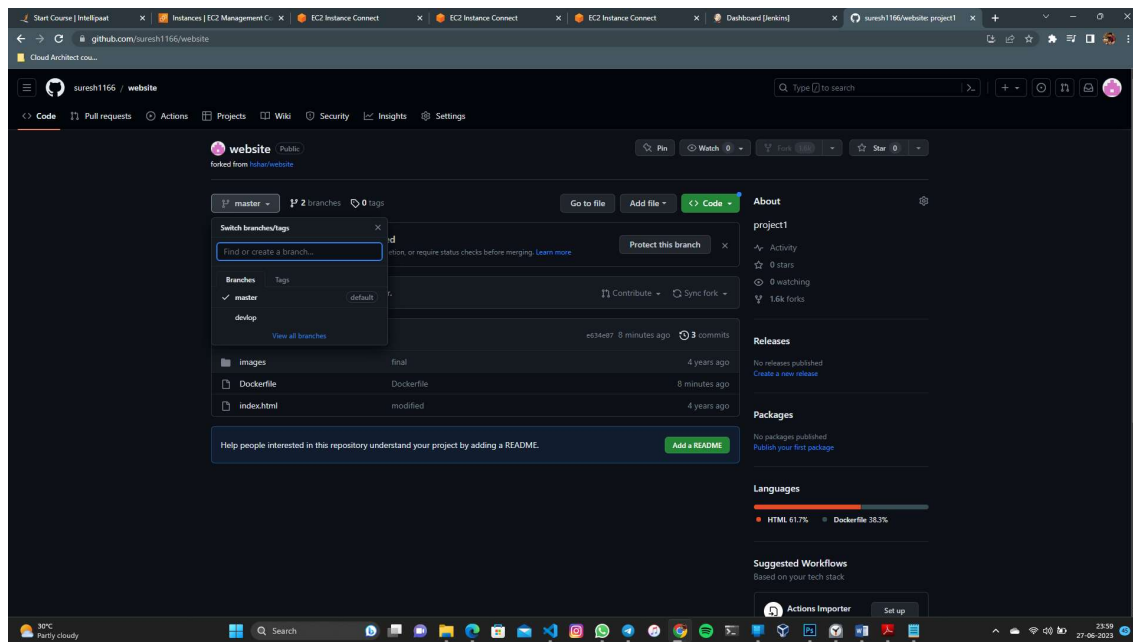


The screenshot shows a terminal window with the following commands and output:

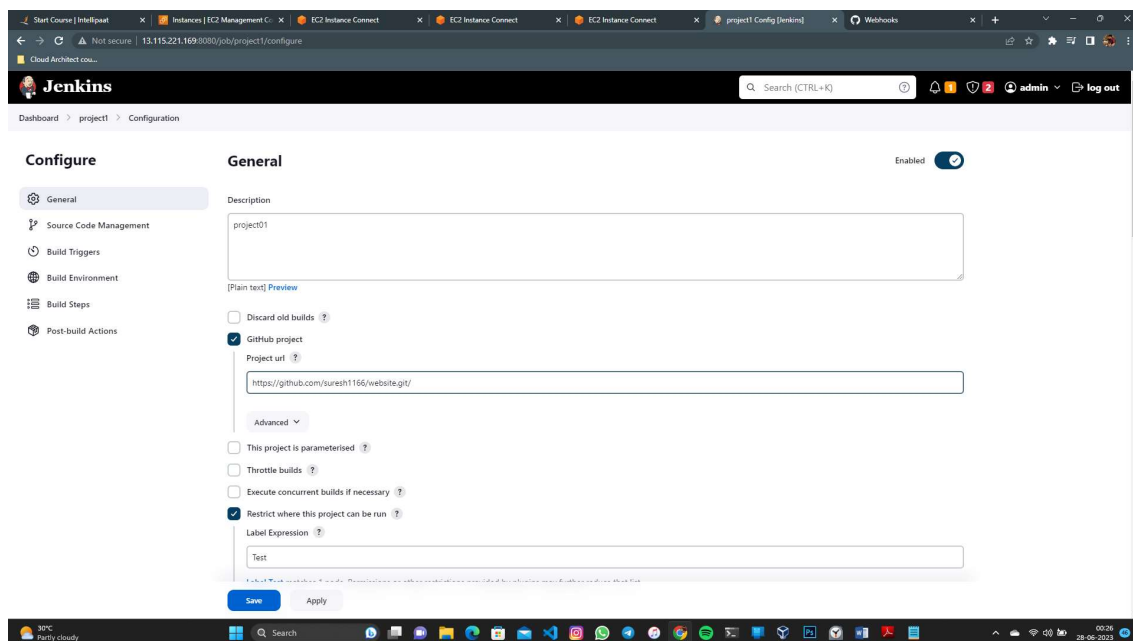
```
untracked files:
  (use "git add <file>..." to include in what will be committed)
  Dockerfile

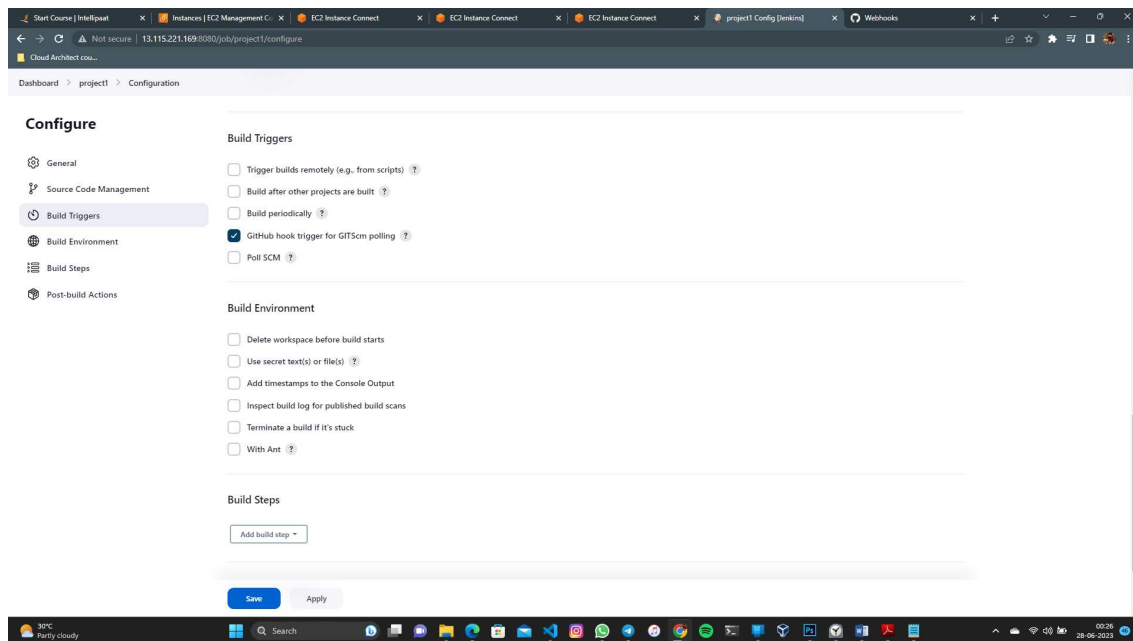
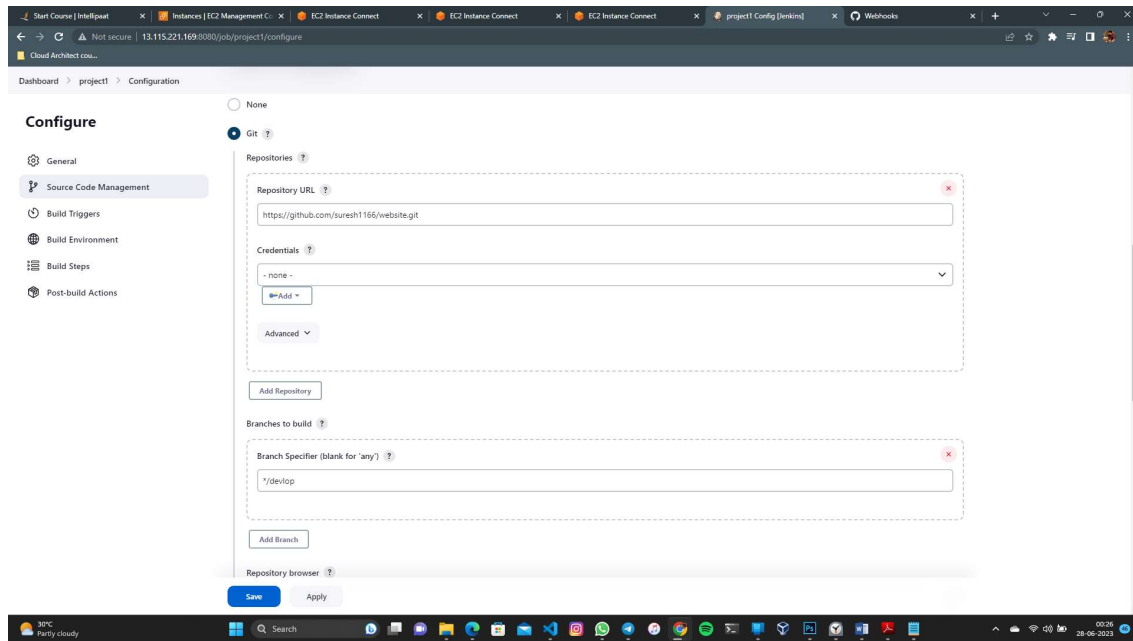
nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-1-36:/workspace$ git add .
ubuntu@ip-172-31-1-36:/workspace$ git commit -m "Dockerfile"
[master e64e07] Dockerfile
 1 file changed, 5 insertions(+)
 create mode 100644 Dockerfile
ubuntu@ip-172-31-1-36:/workspace$ git branch devlop
ubuntu@ip-172-31-1-36:/workspace$ git checkout devlop
Dockerfile images index.html
ubuntu@ip-172-31-1-36:/workspace$ git branch
* master
  devlop
ubuntu@ip-172-31-1-36:/workspace$ git push --all
Username for 'https://github.com': surekh166
Password for 'https://surekh166@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 434 bytes | 434.00 KB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com:surekh166/workspace.git
 * [new branch]      devlop -> devlop
ubuntu@ip-172-31-1-36:/workspace$
```

Now we have two branches and docker file is pushed to repo

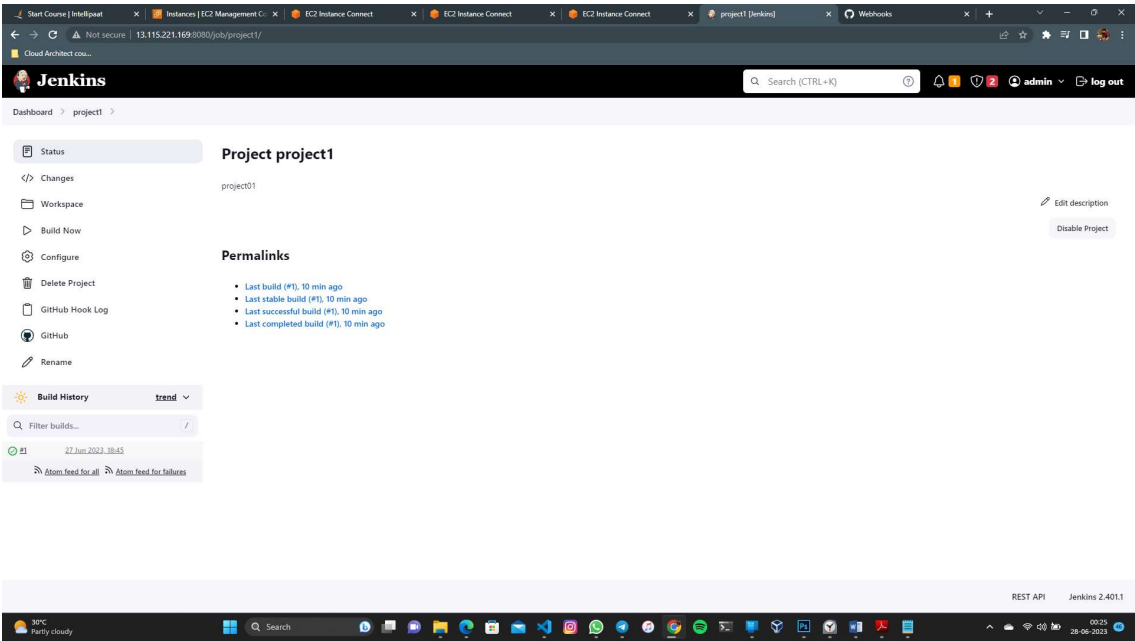
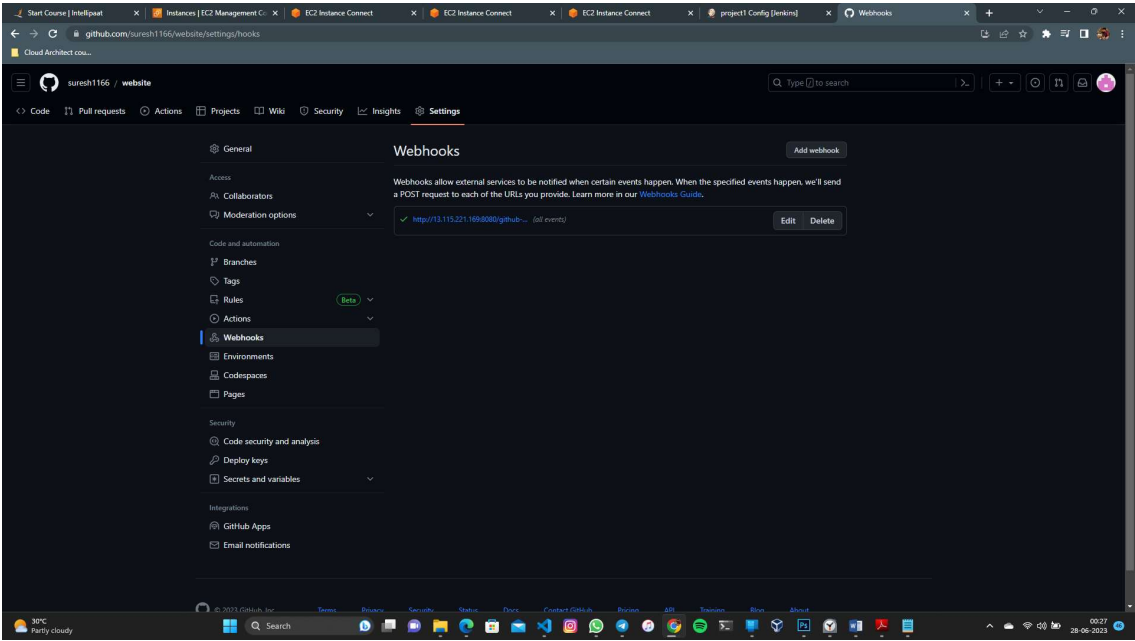


Project has been created

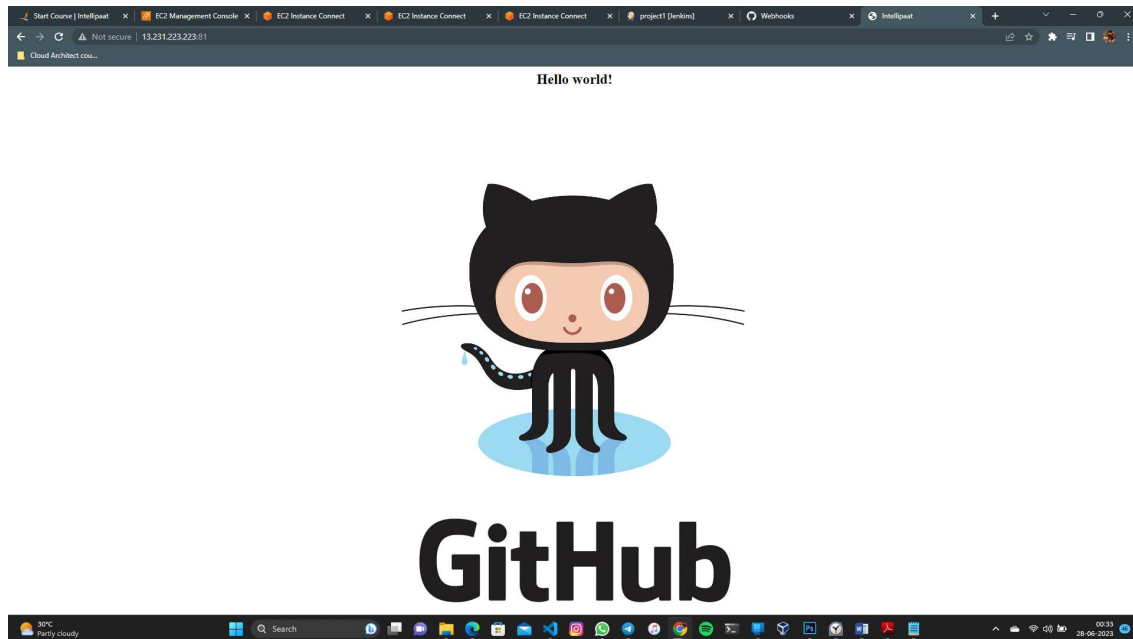




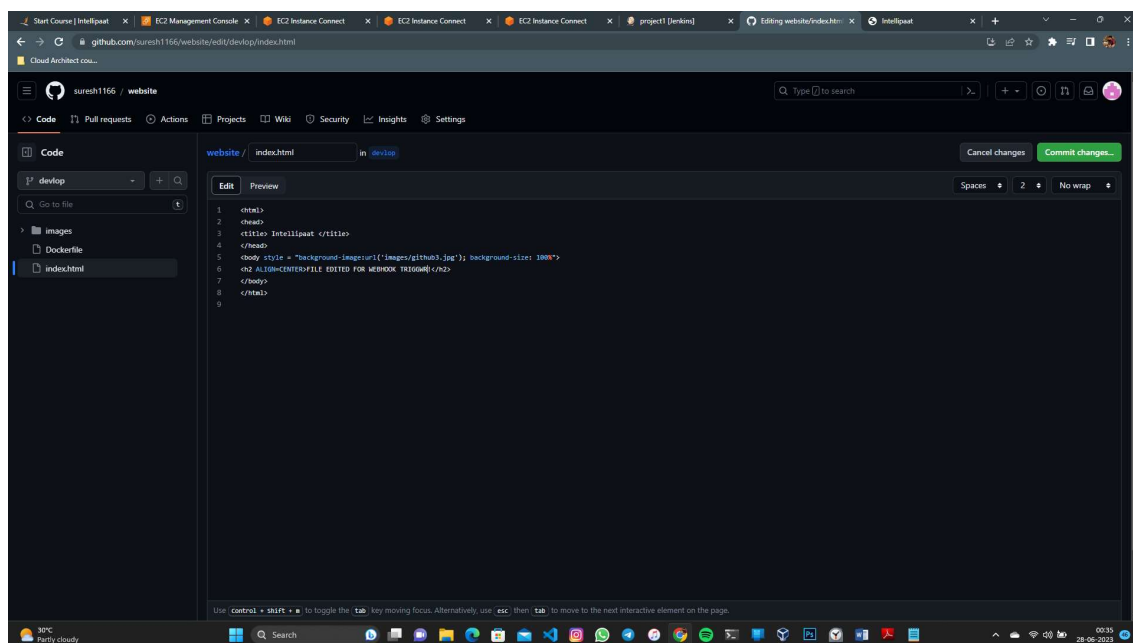
Webhook also created

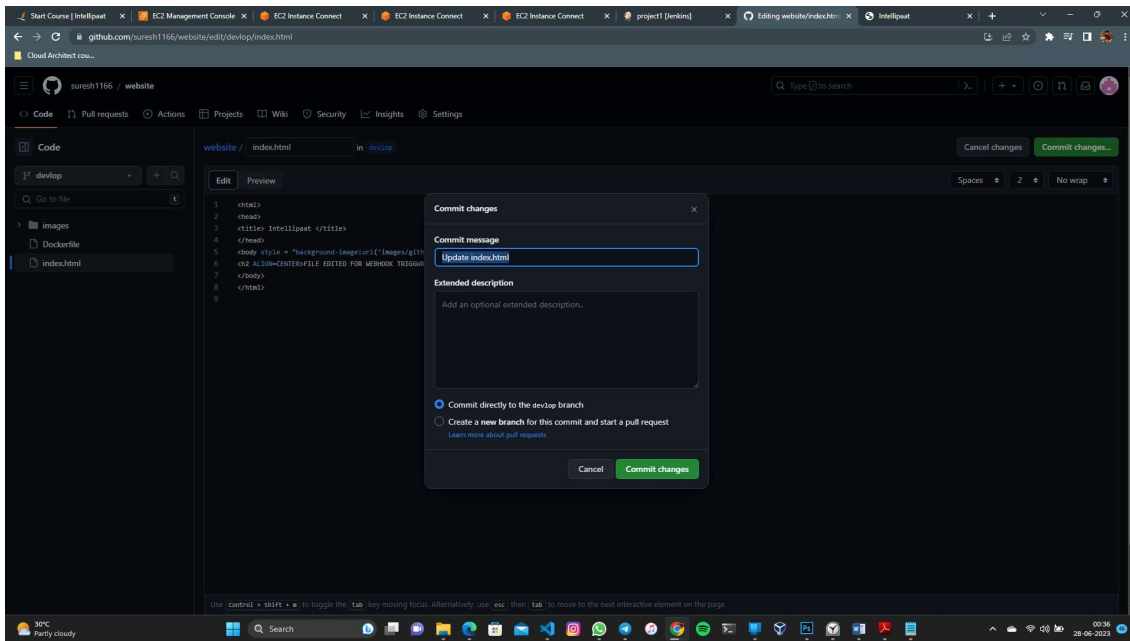


Output:

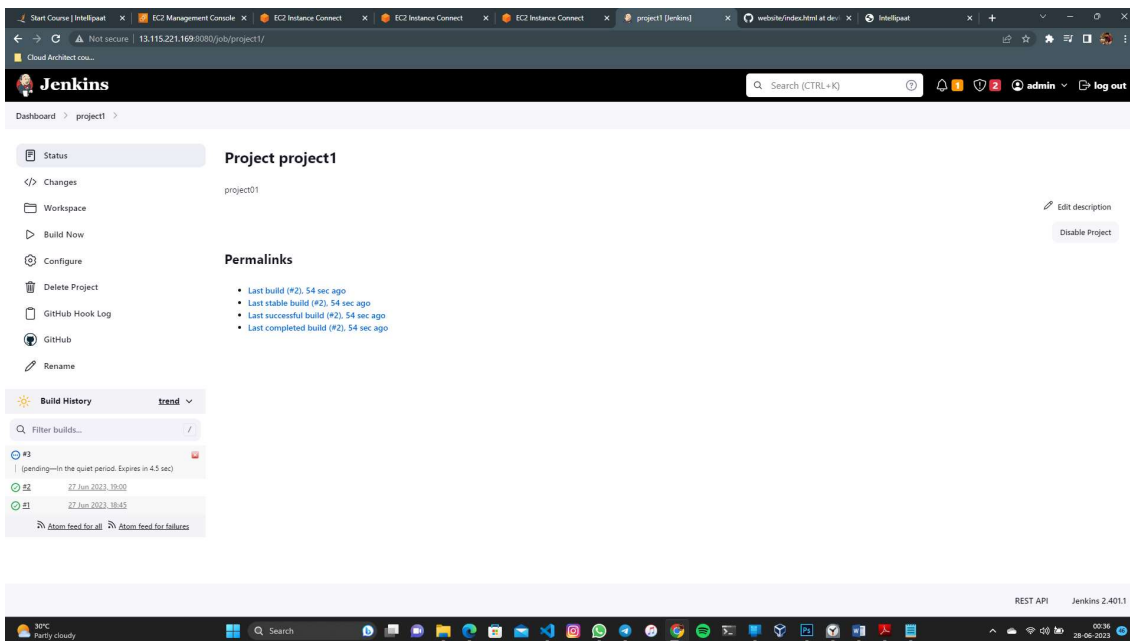


NOW TRY TO EDIT WEB FILE IN GITHUB AND WE CAN SEE WEBHOOK TRIGGERED





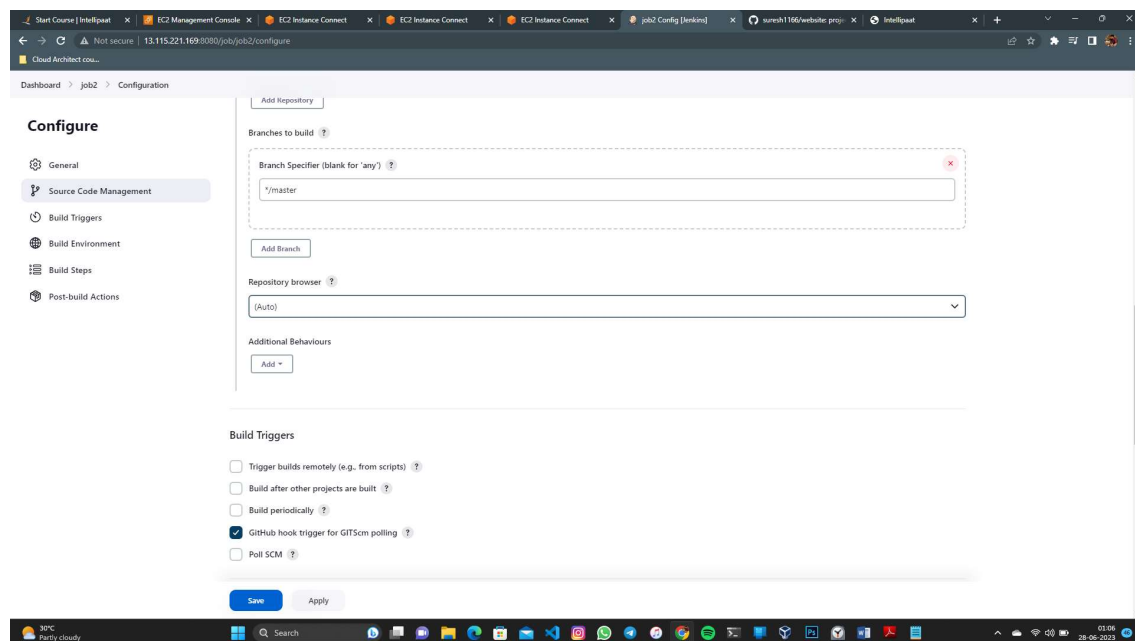
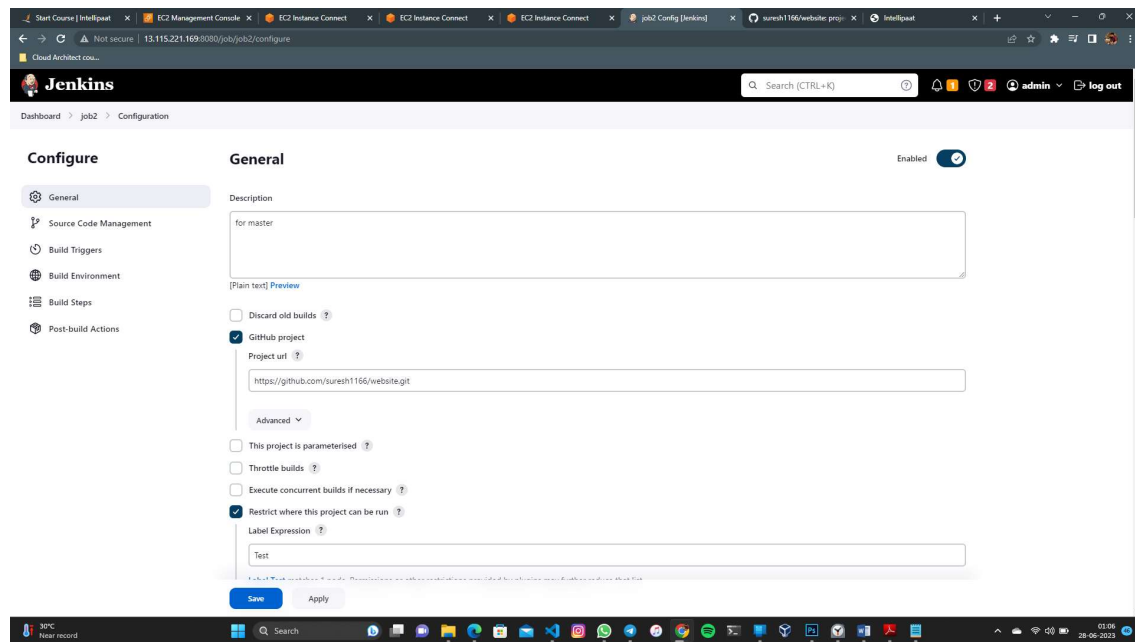
WEBHOOK TRIGRED

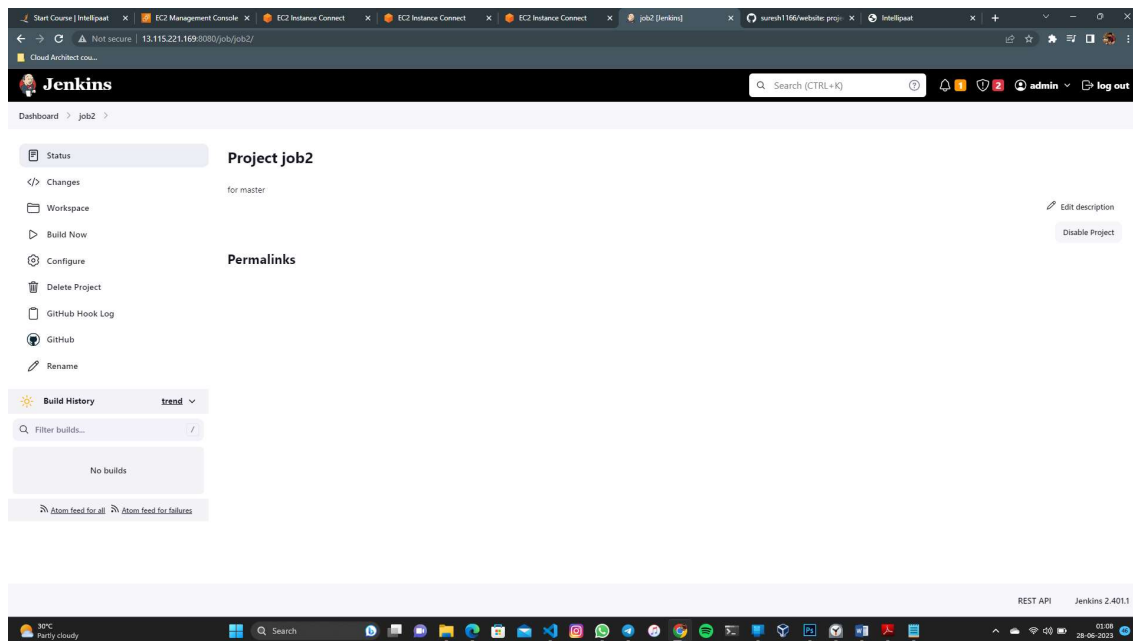
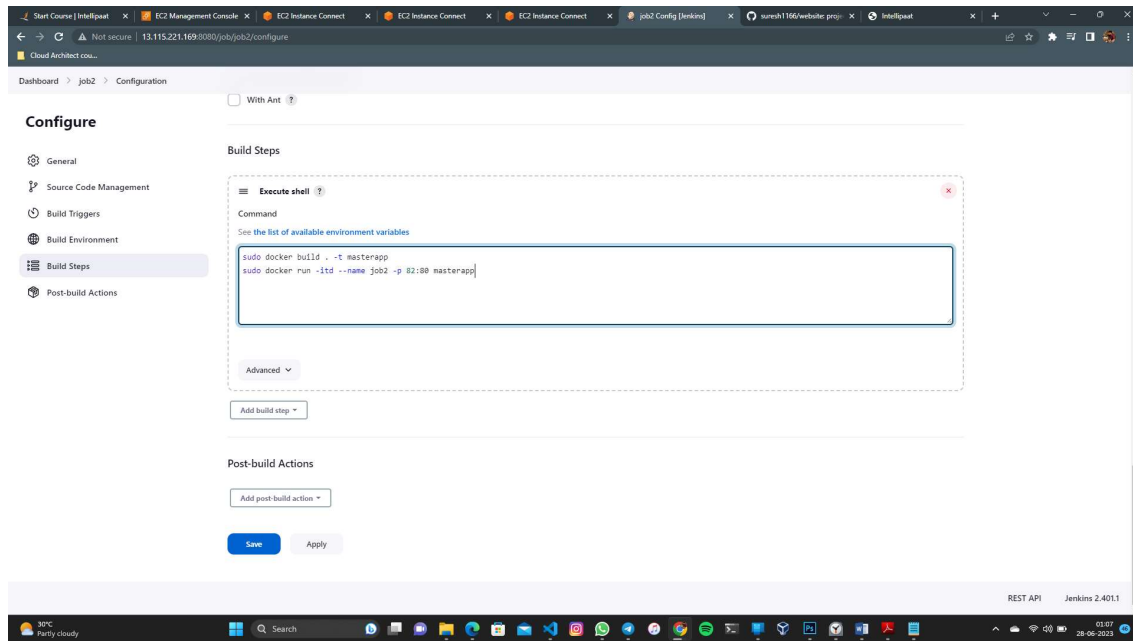


Trigred Output:

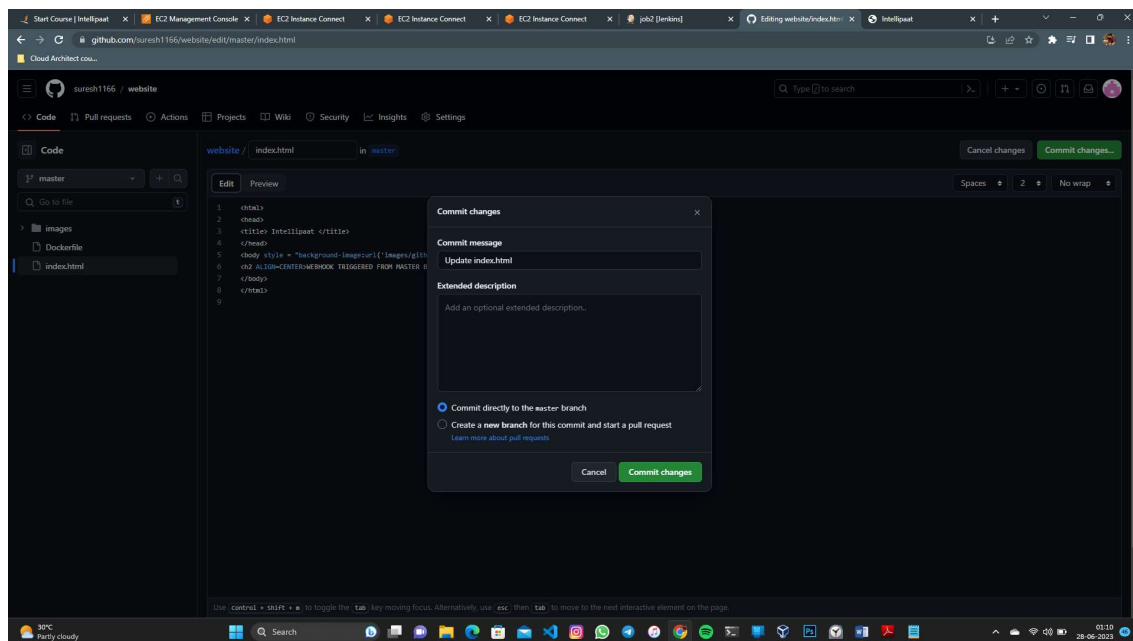
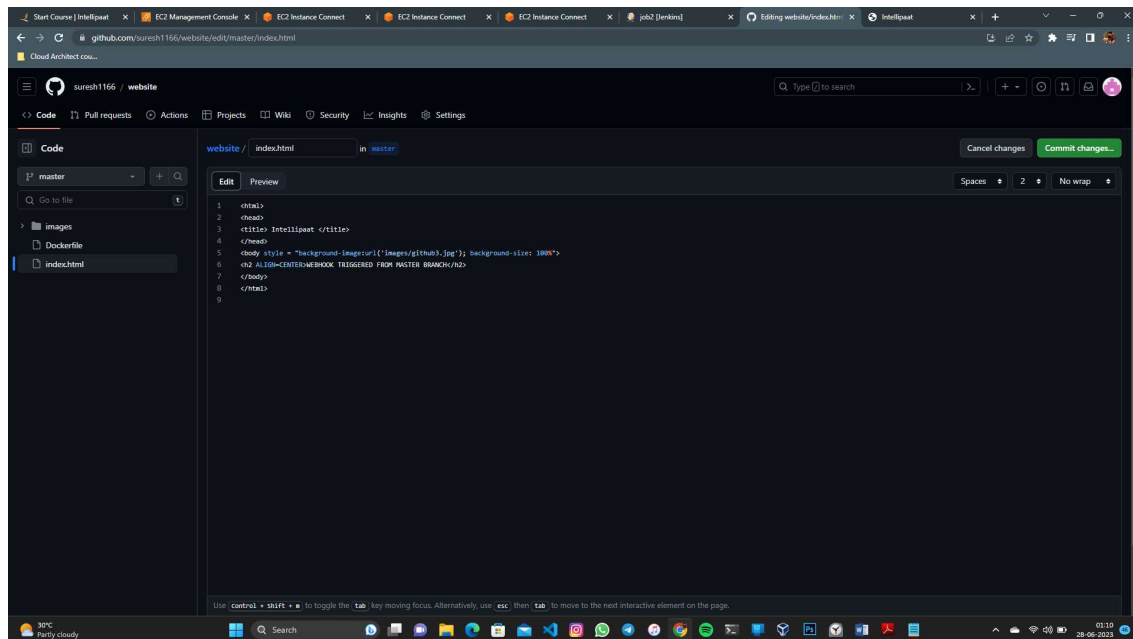


Now configure for master branch

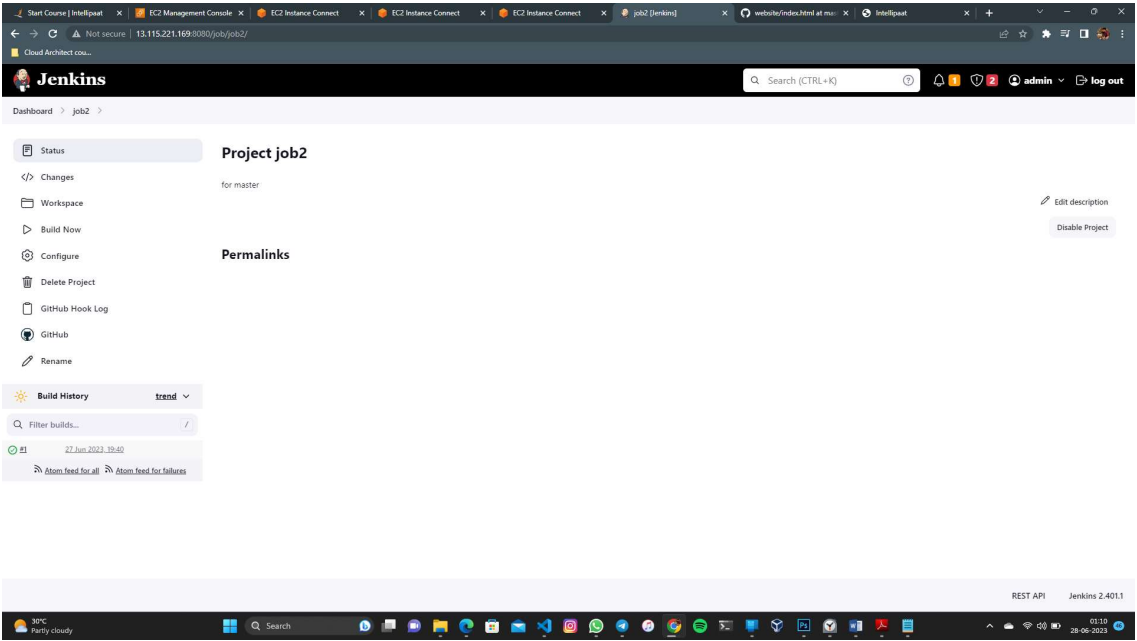




Now committing the code in master branch and see webhook triggers for master branch



Build got success:



Output:



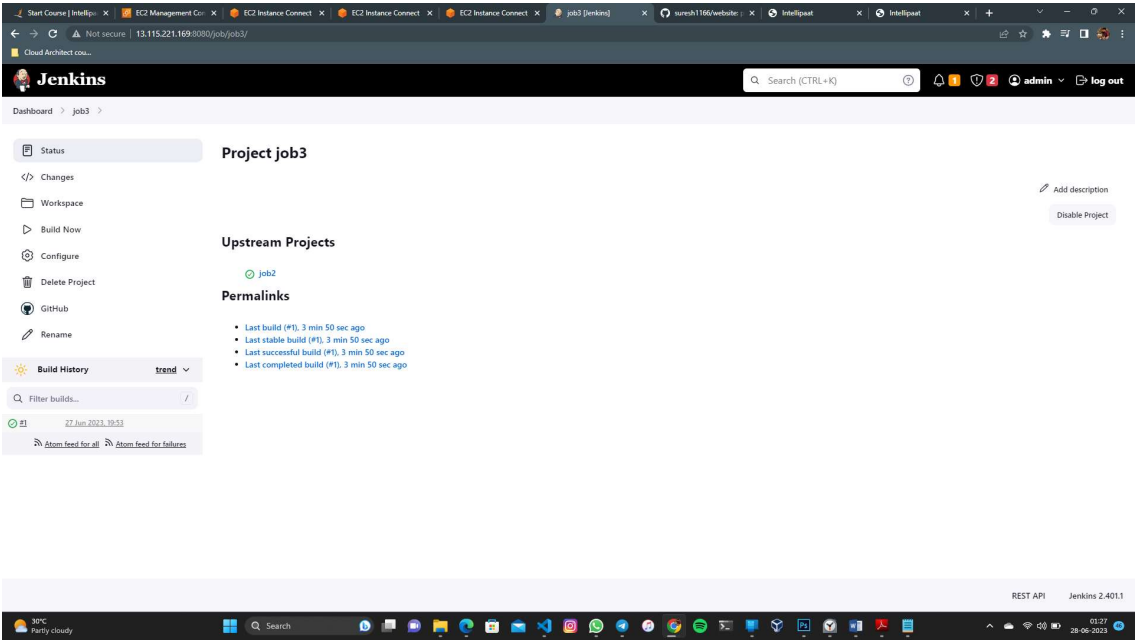
3rd Job created

The screenshot shows the Jenkins web interface for a job named 'Project job3'. The left sidebar contains navigation links: Status, Changes, Workspace, Build Now, Configure, Delete Project, GitHub, and Rename. The main area displays the job's configuration, including a 'Permalinks' section and a 'Build History' section showing 'No builds'. The top of the page features the Jenkins logo, a search bar, and user information (admin, log out). The bottom of the page shows a Windows taskbar with various application icons and system status (30°C, 28-06-2023).

Now configured job 2 the post build as job 3

The screenshot shows the Jenkins web interface for the configuration of 'job2'. The left sidebar highlights the 'Post-build Actions' section. The main area displays the 'Post-build Actions' configuration, where 'Build other projects' is selected. The 'Projects to build' field contains 'job3'. The 'Trigger only if build is stable' option is selected. The bottom of the page shows a Windows taskbar with various application icons and system status (30°C, 28-06-2023).

Once job2's post build configured job 3 automatically build and got success



Final Output:

