**Technologies used** - Jenkins, Docker, AWS EC2, GitHub Triggers, Dockerfile, Apache WebServer, Ubuntu OS

**Repository URL -** <https://github.com/Sarthak-Agarwal1410/webserver-php.git>

Firstly, I had created an EC2 instance on AWS

AMI used - Amazon Linux 2

Secondly, I had used the instance type as t2.micro

Then, in step 3 under configure the instance, I had given the following commands to configure Jenkins on the AWS ec2 instance

#!/bin/bash

sudo yum install wget -y

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

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum install jenkins java-1.8.0-openjdk-devel -y

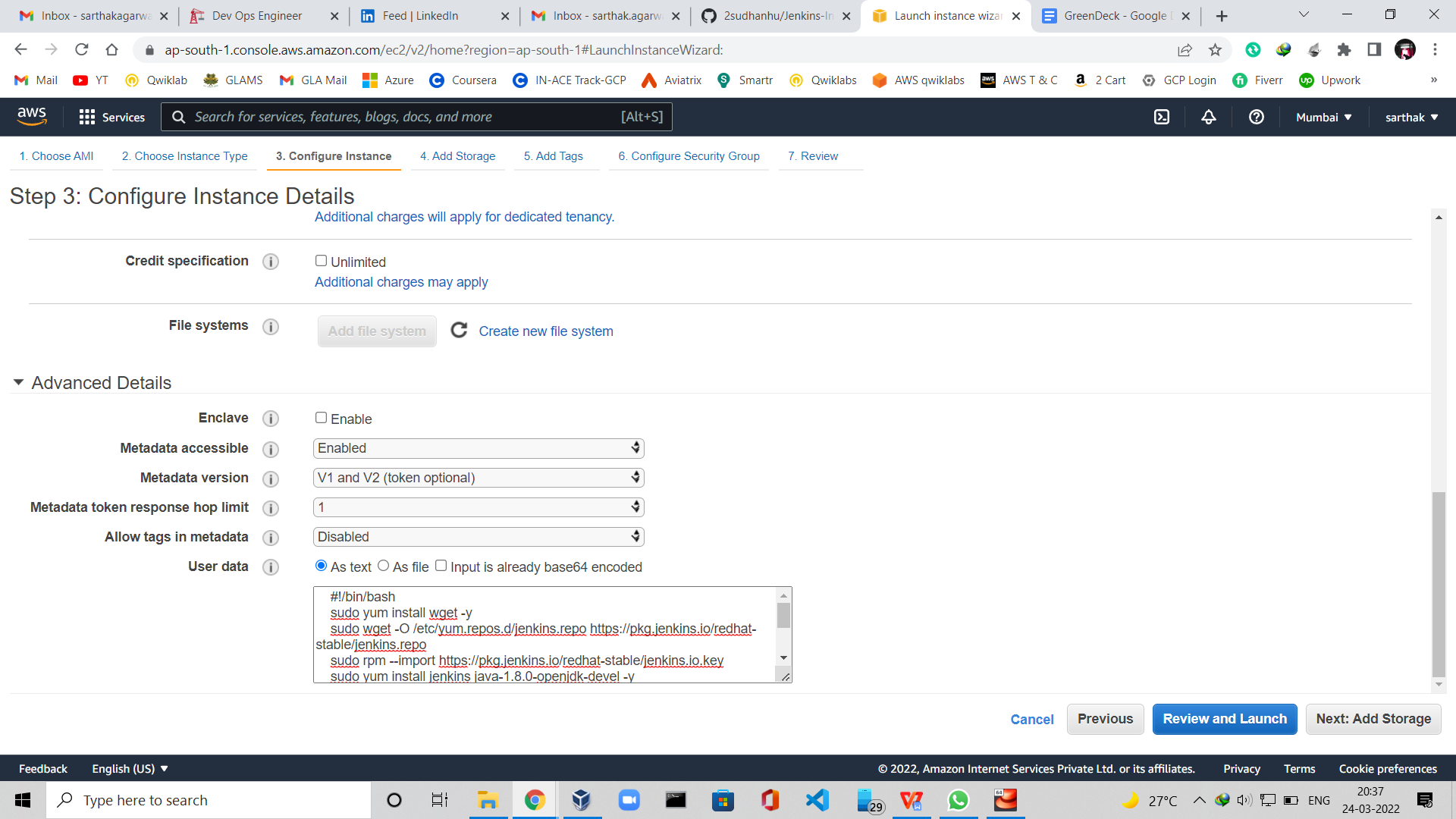
sudo systemctl daemon-reload

sudo systemctl start jenkins

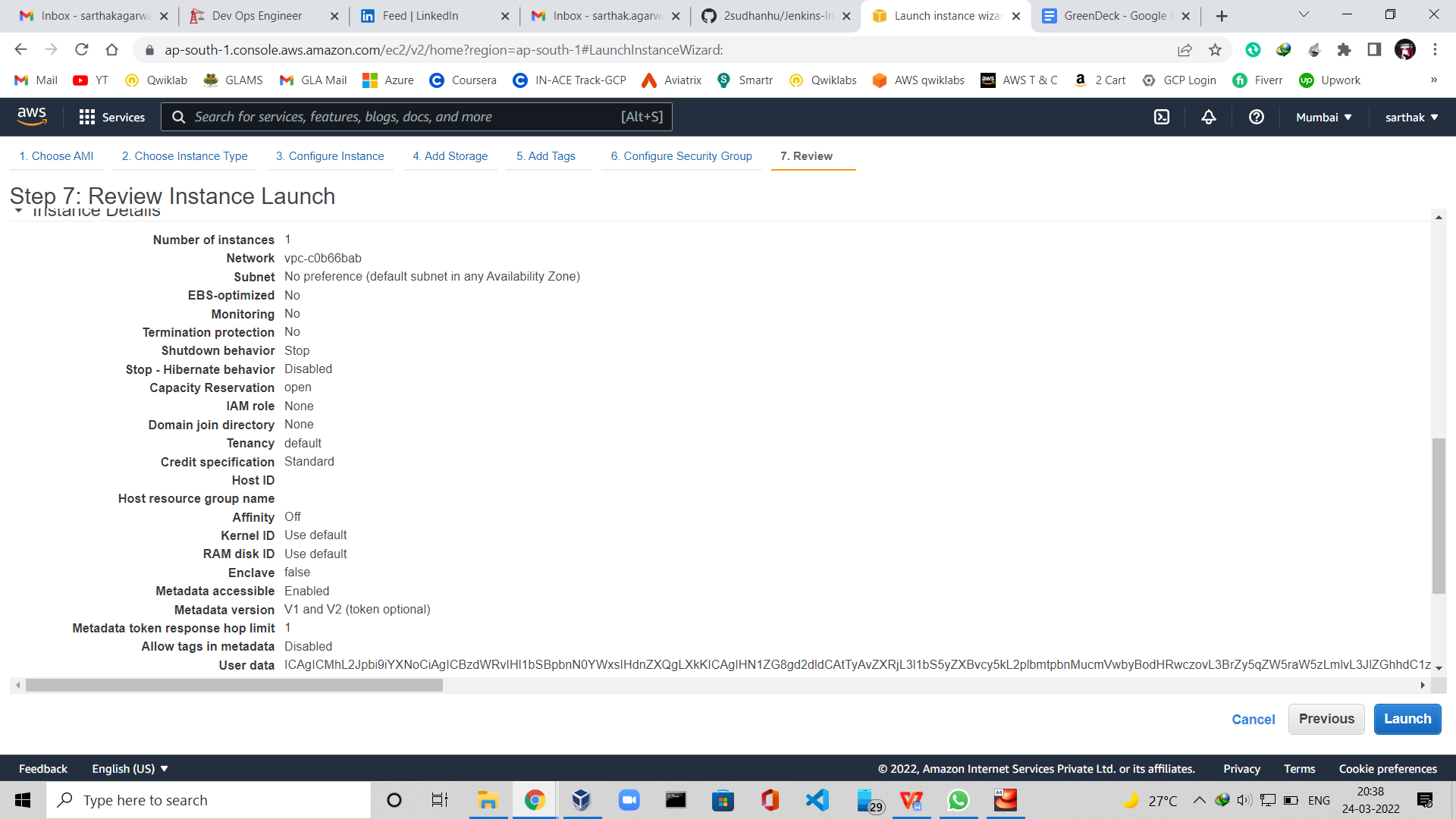
sudo systemctl enable jenkins

sudo yum install git -y

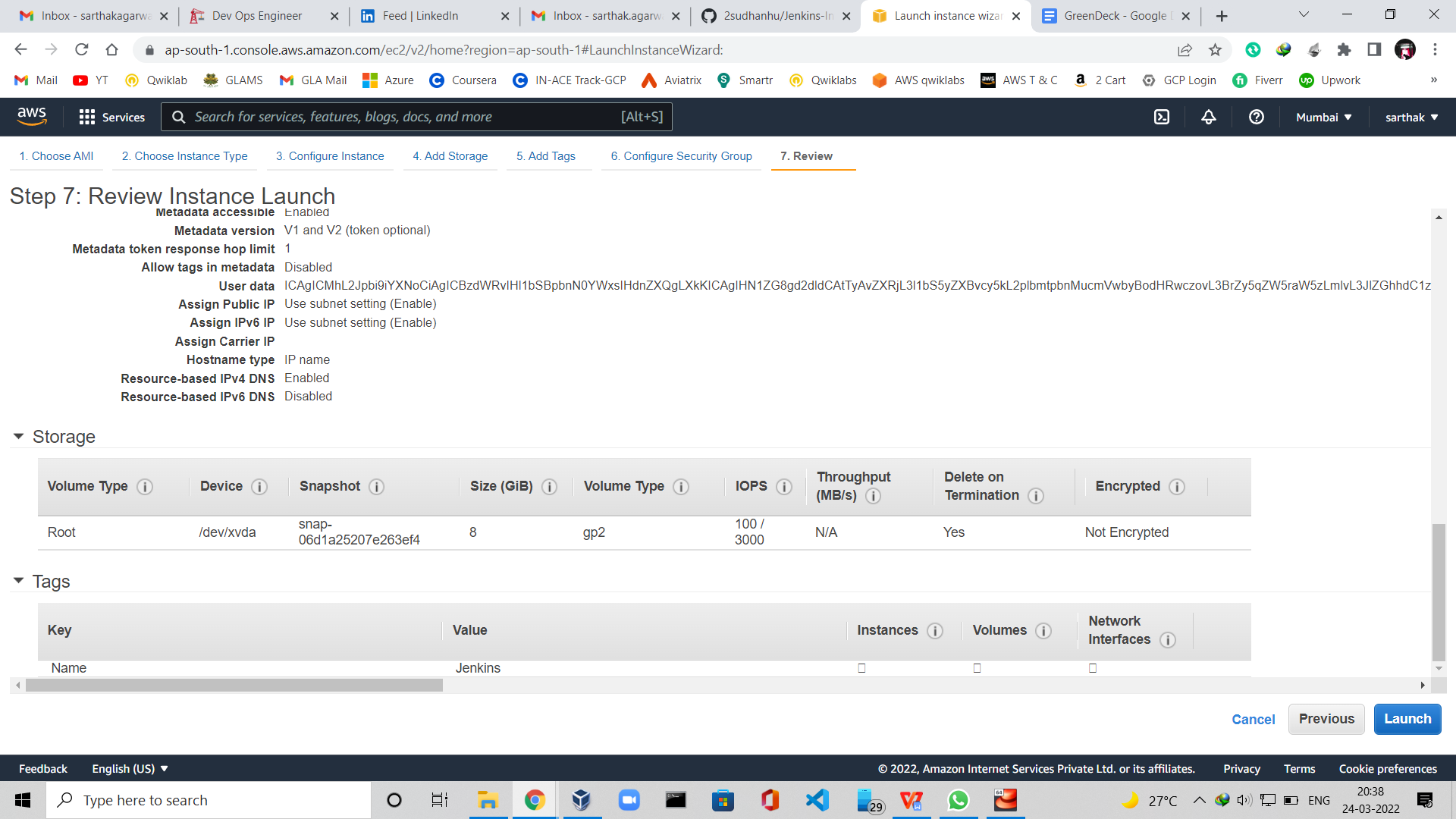
sudo yum upgrade -y



You can check more details of the instance below →

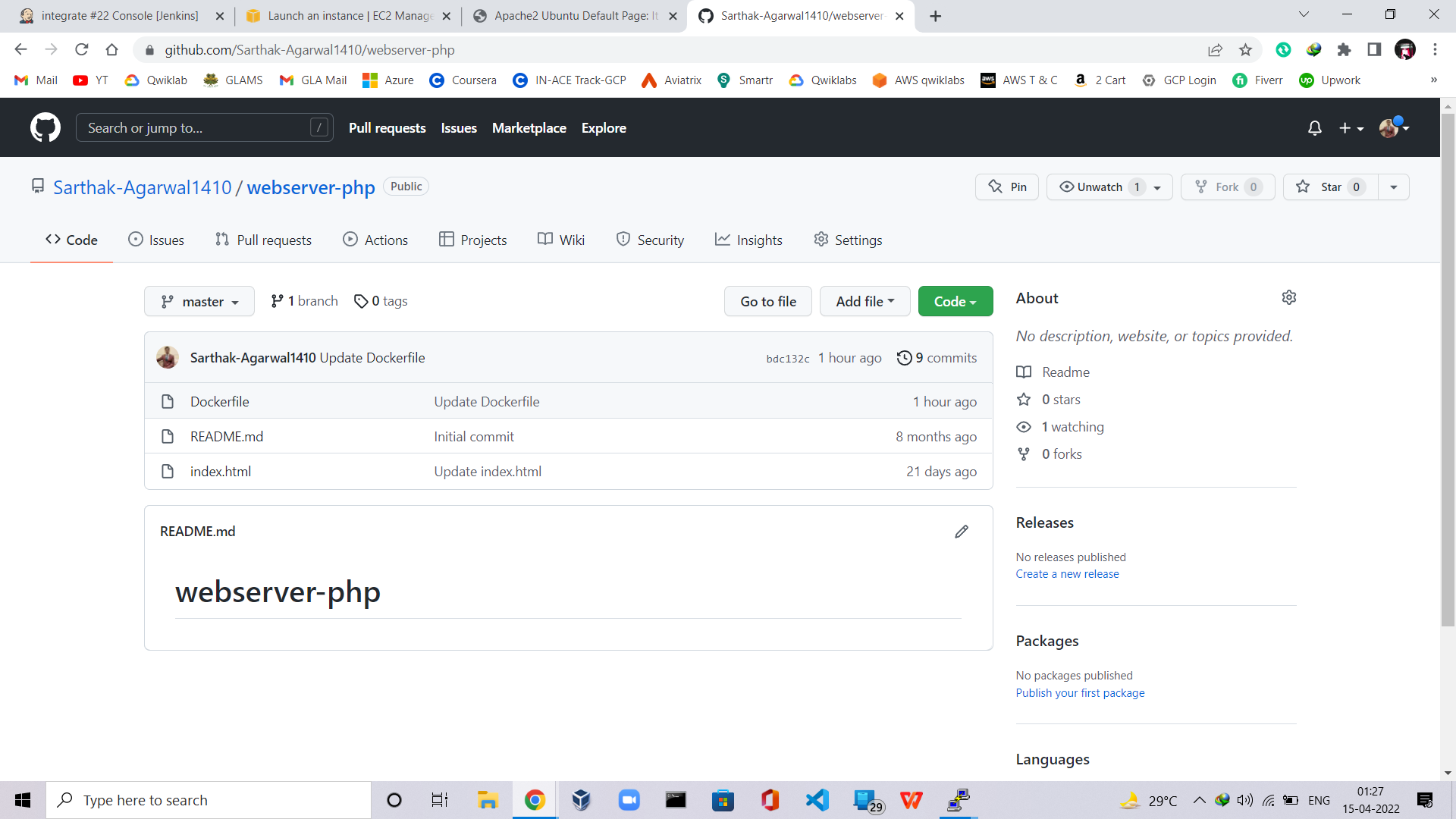


At last, I had attached the key pair with the instance



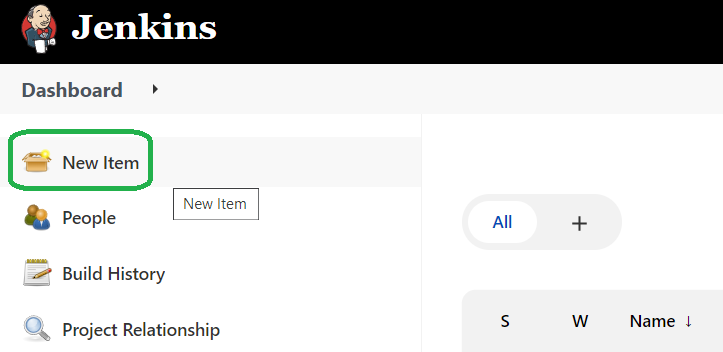
Then, I created a repository on Github and then added a simple php file.

Repository Name - webserver-php (as you can see below)



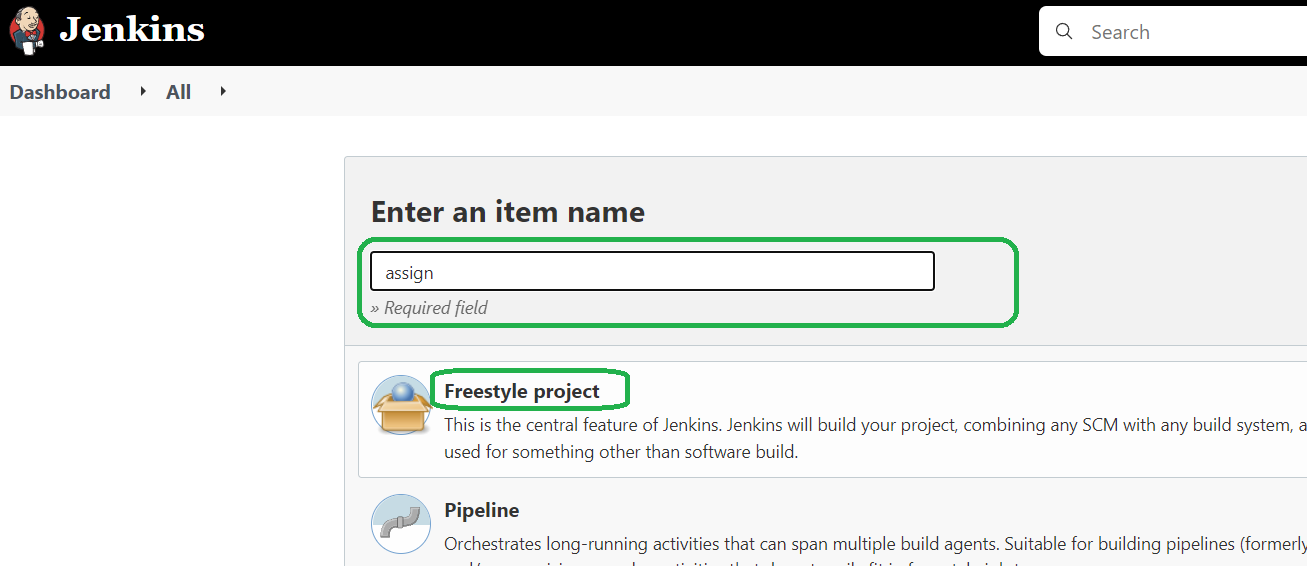
Now, I copy-pasted the public URL of the instance in the browser & set the credentials on the landing first page of Jenkins

Now lets, make a new job on the Dashboard

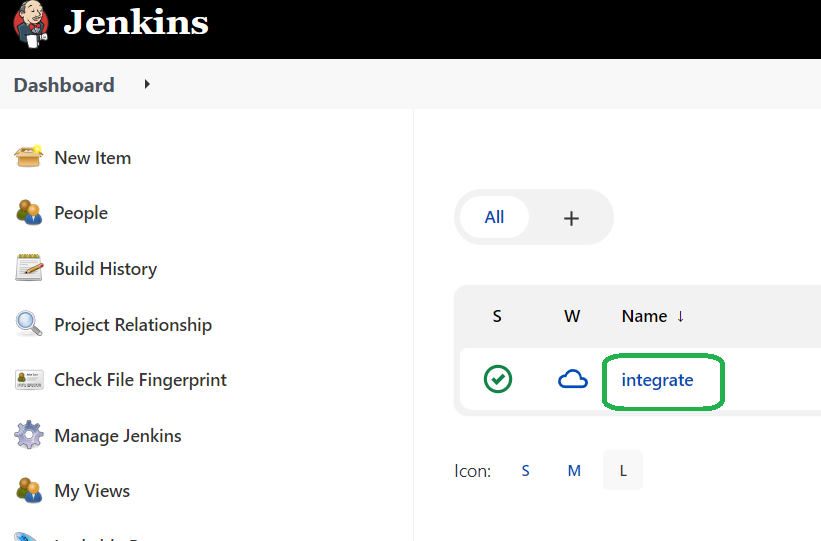


**Name** - *integrate*

And then Select *Freestyle Project*

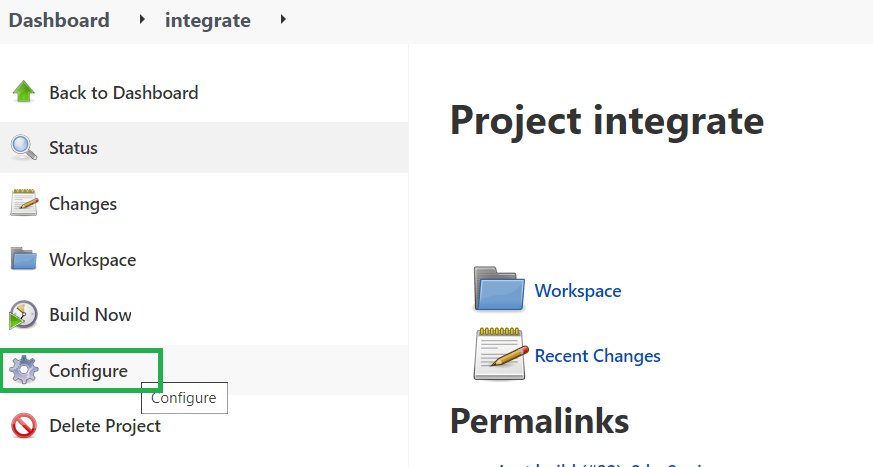


Then click on **OK**

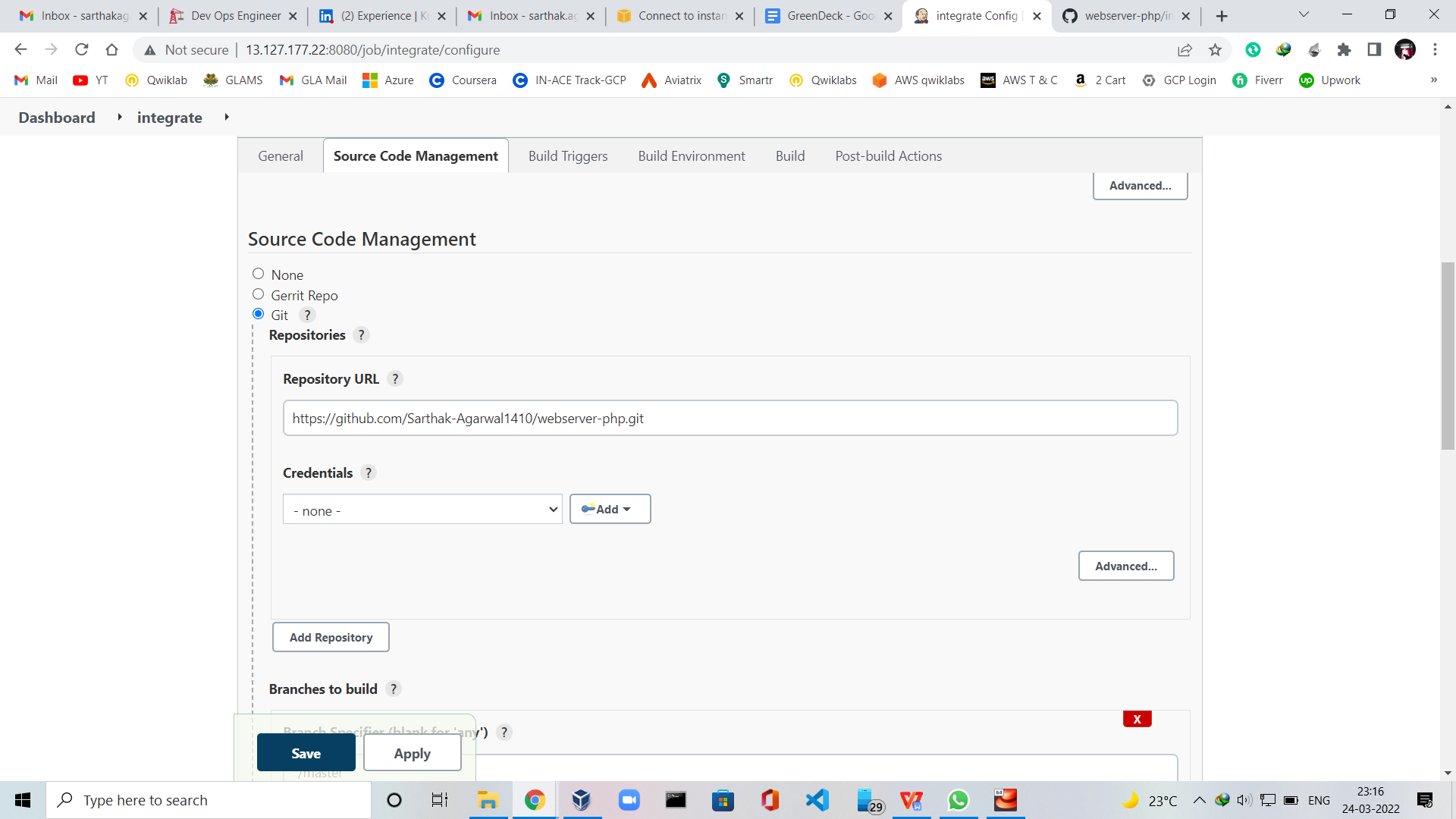


A job named Integrate will be created on the DashBoard Screen,

Now, Click on integrate → Configure

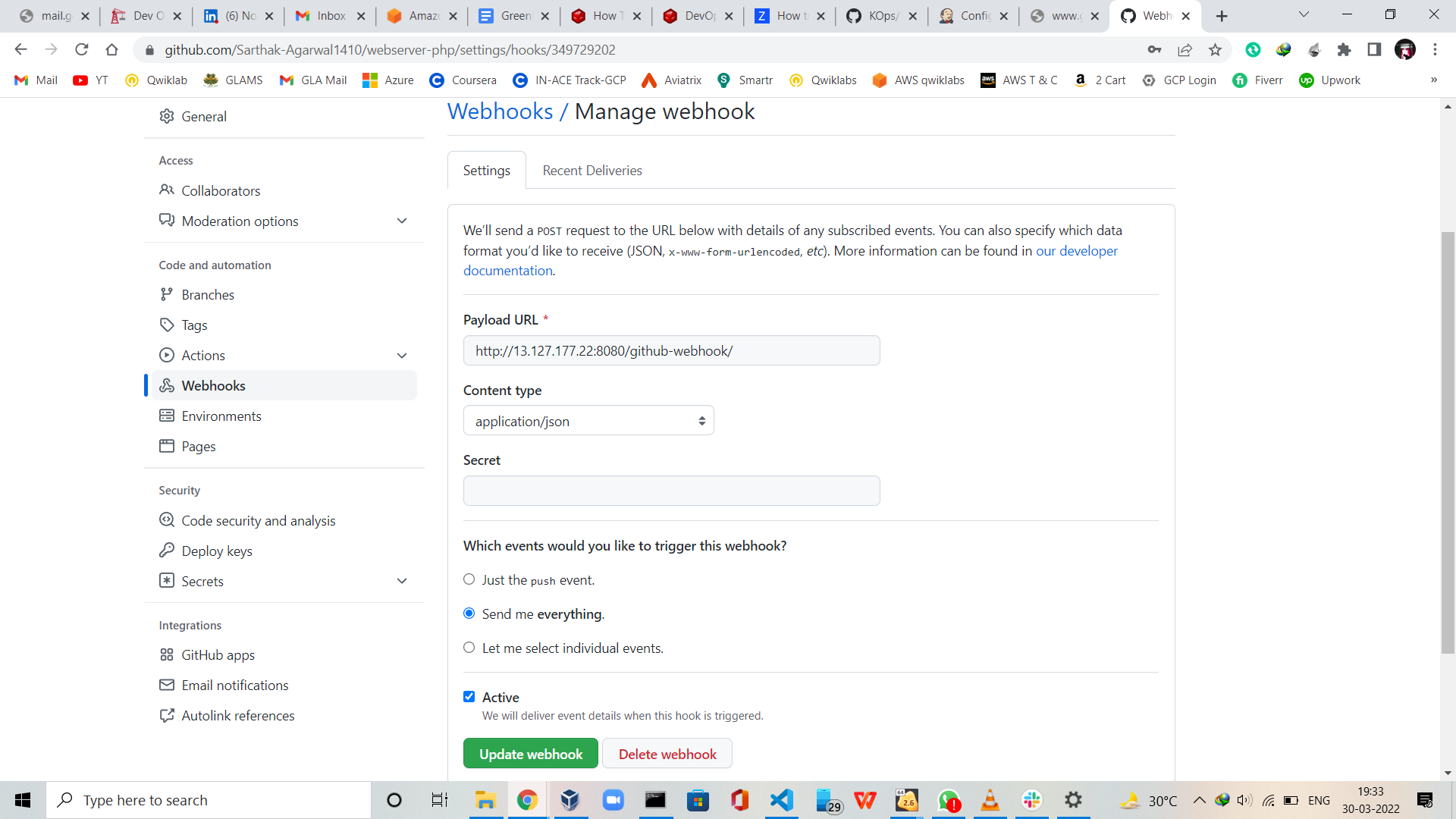


After this, I just copy-pasted the repo link in the SCM section in the Repository URL.



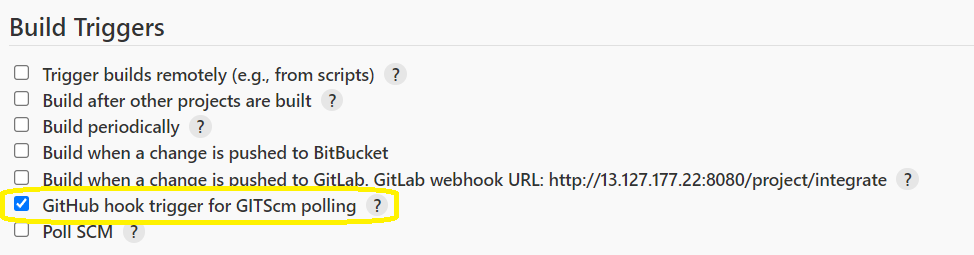
For configuring the webhooks to get triggered, I used

Payload URL = http://<public\_ip\_of\_ec2\_instance:port>/github-webhook/



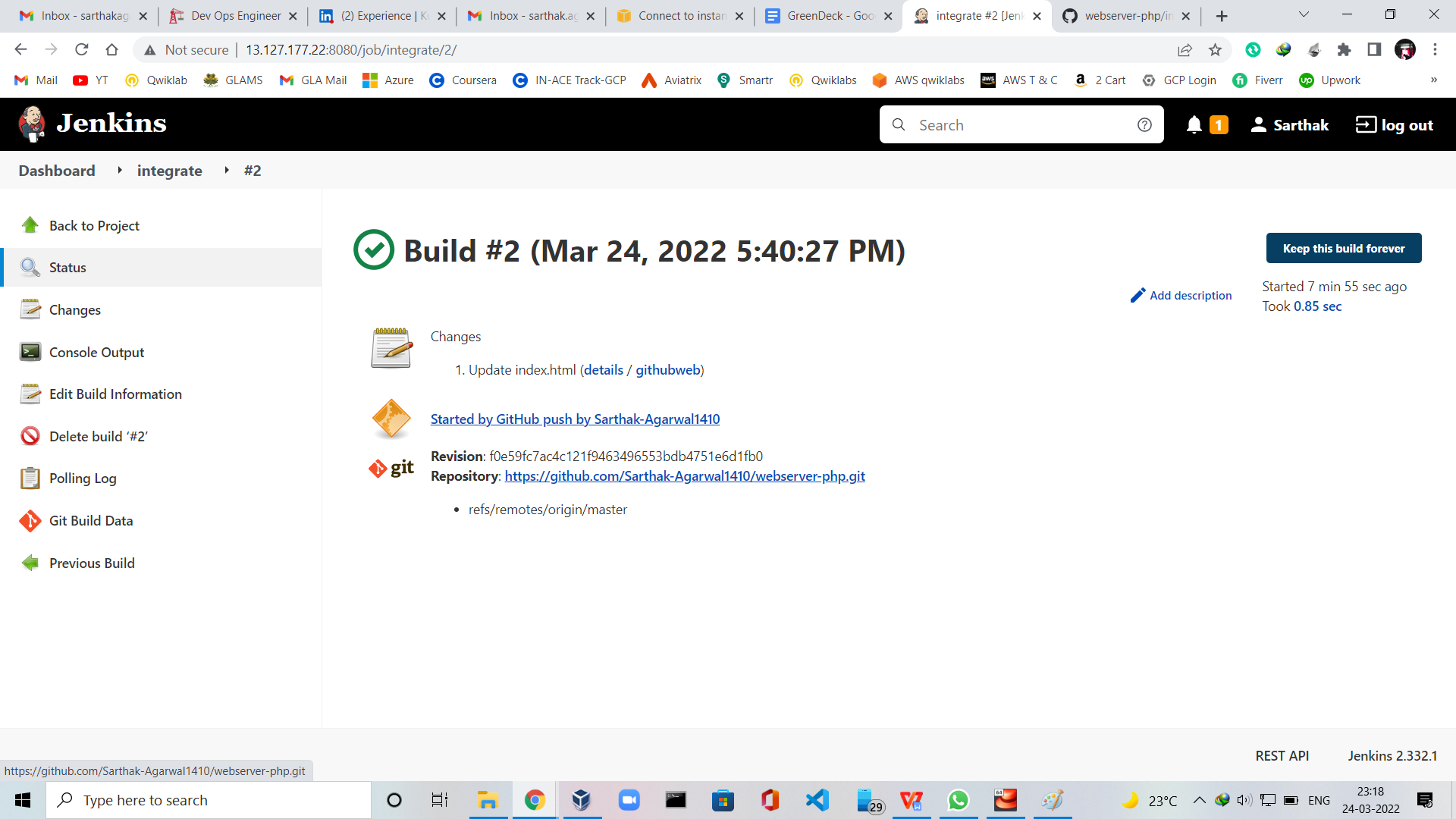
Now coming back to the Jenkins browser, I selected the *GitHub hook trigger for GitScm polling*

Under build, triggers to trigger the build.

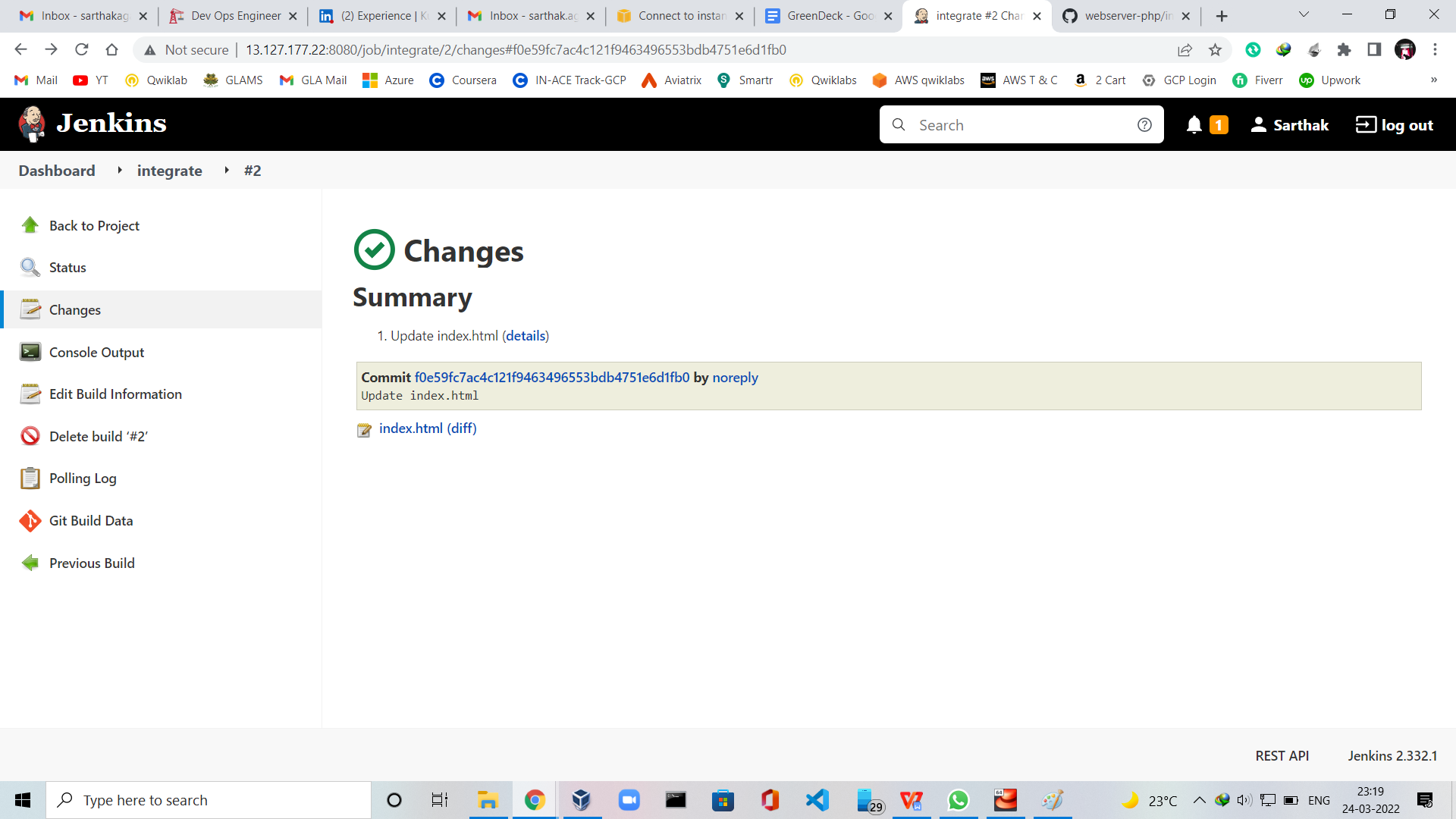


Now, As soon as I changed some code in the GitHub repo in the index.html file

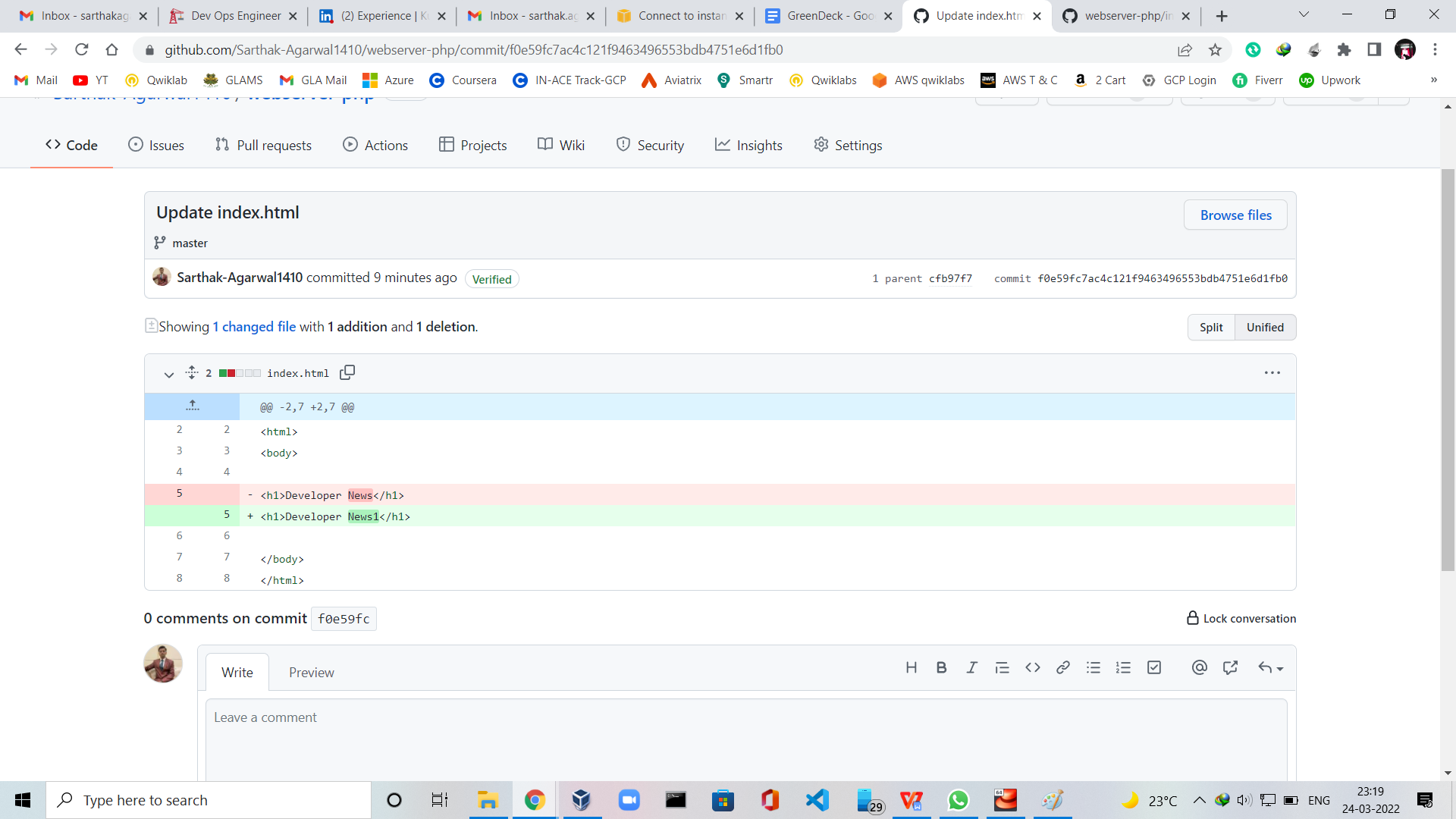
The build triggers automatically run the job as follows →



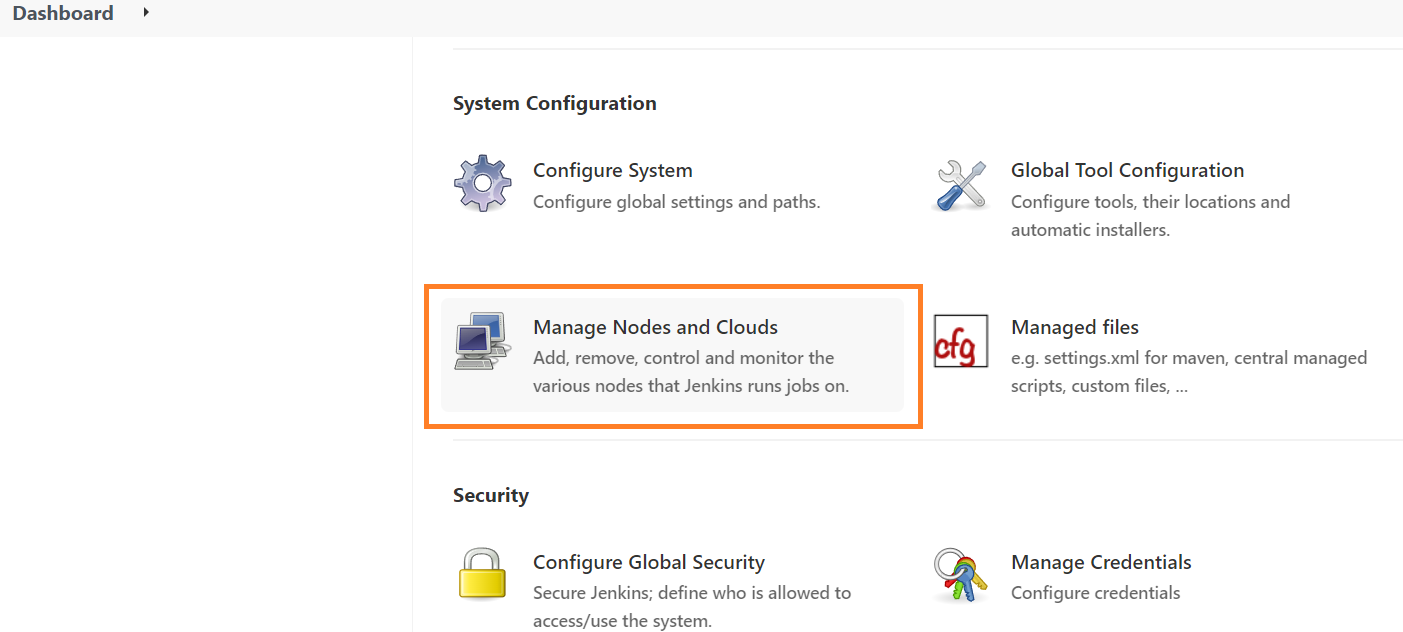
You can see the summary of the changes made below →



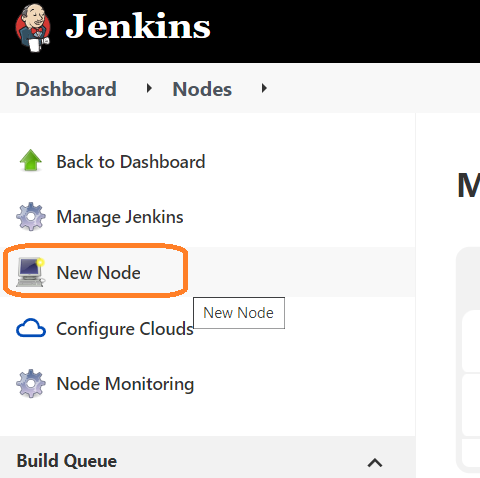
The code changes are reflected clearly in the below screenshot in green & red color →



Now, click on Manage Jenkins → Manage Nodes and Clouds



Now we need to create a need Node



**Name** - assign

**Remote root directory** - /root/

**Usage** - Use this node as much as possible

**Launch method** - Launch agents via SSH

**Host** - 65.1.148.165

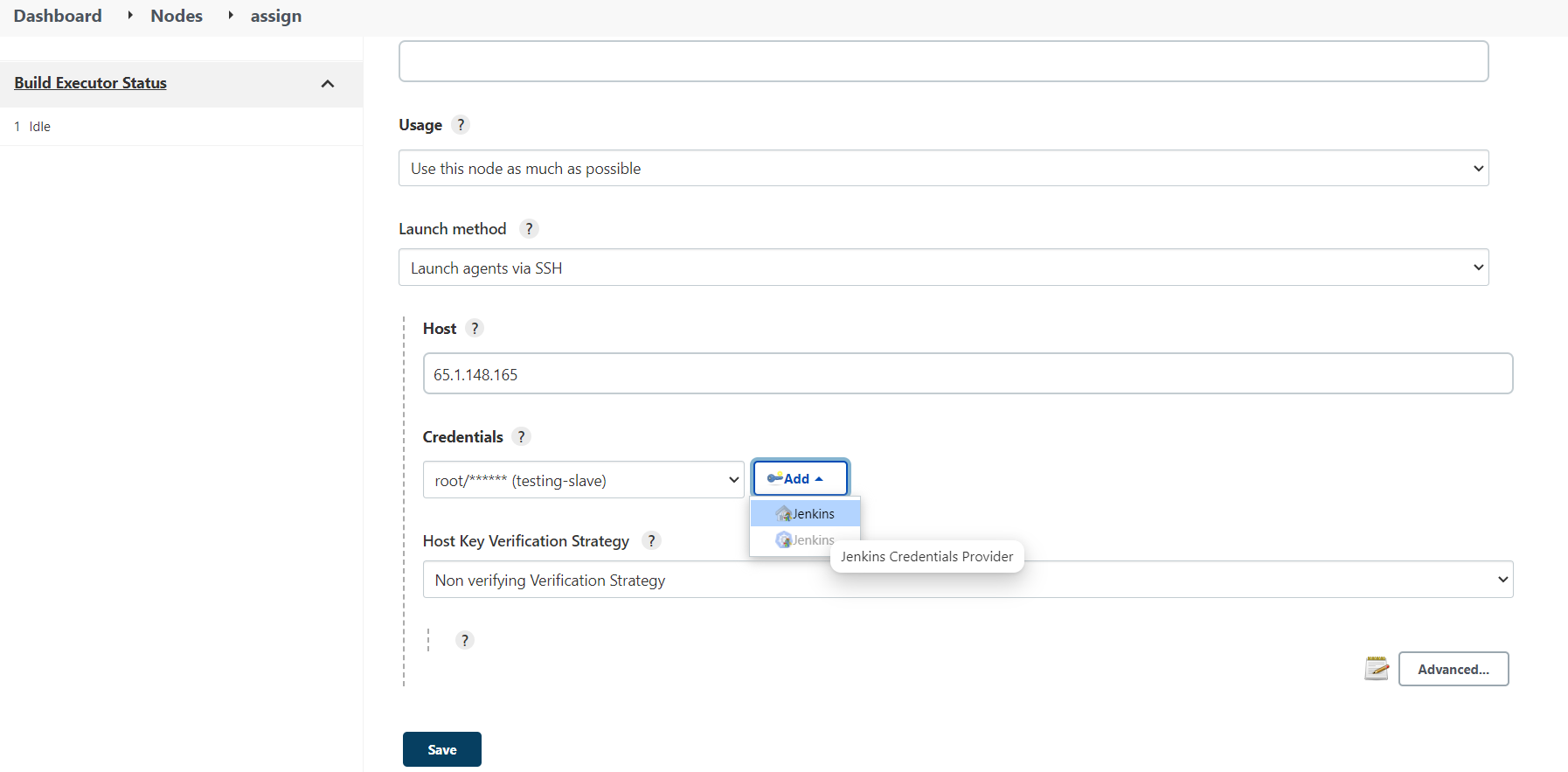
For the Host URL, I had created another instance on AWS

In that instance, I downloaded docker with the command

**# yum install docker**

To start docker services

**# systemctl start docker**



Now click on Jenkins as shown in the above screenshot, and set

Username - root

Password - <set\_any\_password>

ID - testing-slave

Click on ADD

Then select *root/\*\*\*\*\*\*(jenkins-slave)* under credentials.

Click Save

Coming back to the instance on which we installed docker just now,

Configure root sign-in

**#vim /etc/ssh/sshd\_config**

Changes Made →

*PasswordAuthentication yes*

*PermitRootLogin yes*

**#sudo service sshd restart**

Coming back to Jenkins instance,

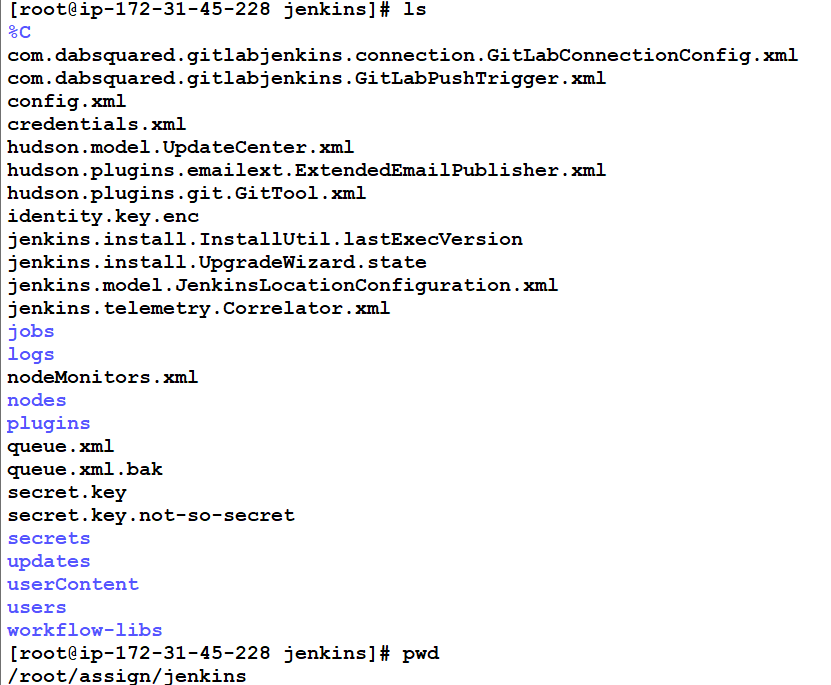
**#vim /etc/sysconfig/jenkins**

**Changed Made** → JENKINS\_HOME="/root/assign"

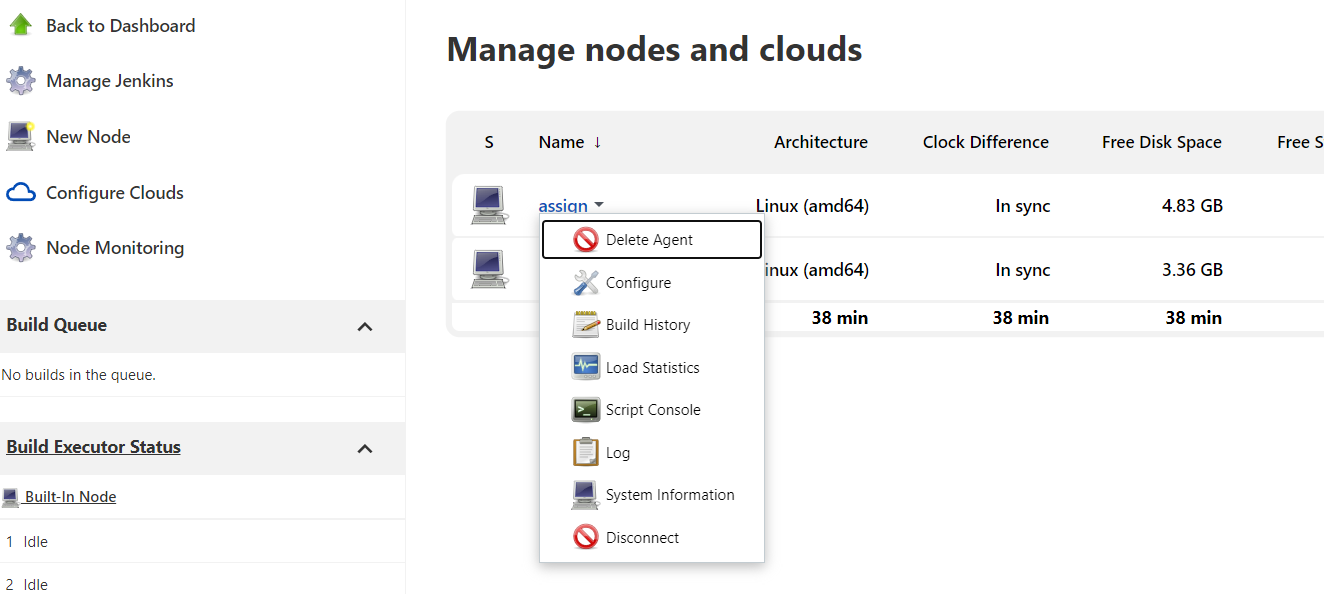
#mkdir assign

#cd assign

Make sure you copy the Jenkins folder in this directory

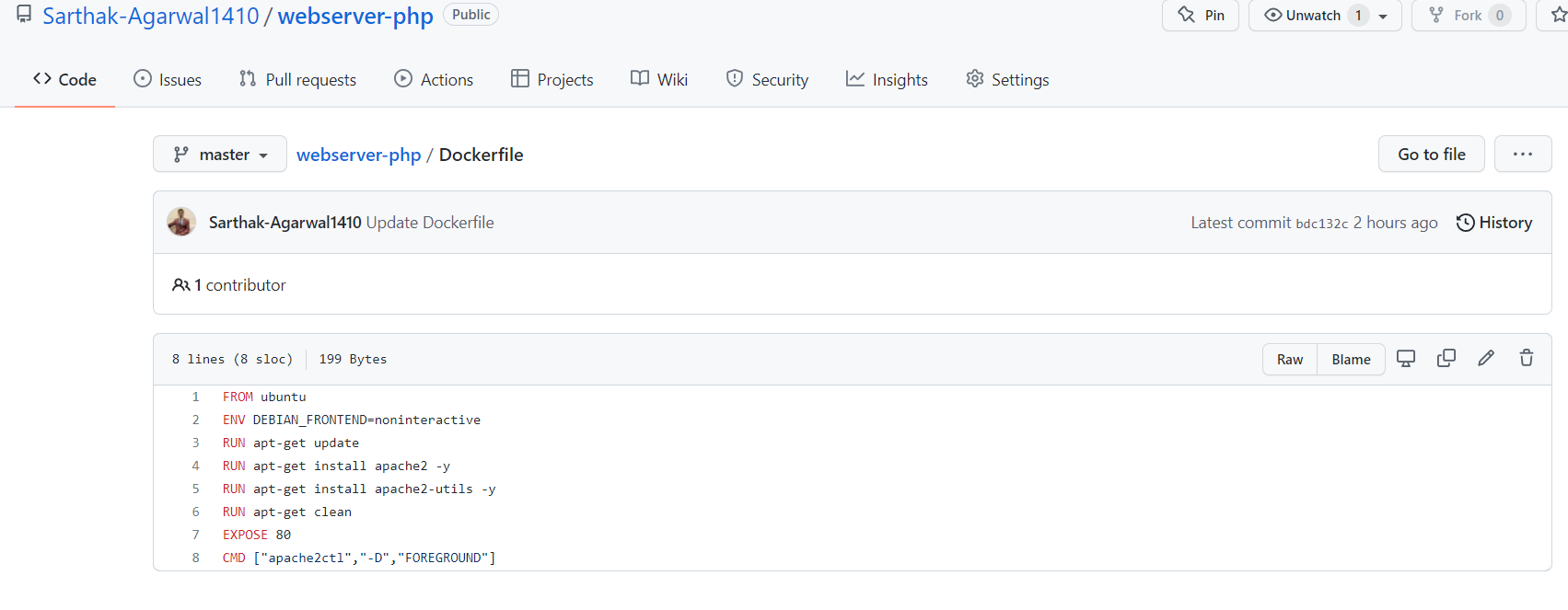


As soon as you will click on **Save** in the Jenkins console, a new node will be created by name as assigned



This will create a pipeline.

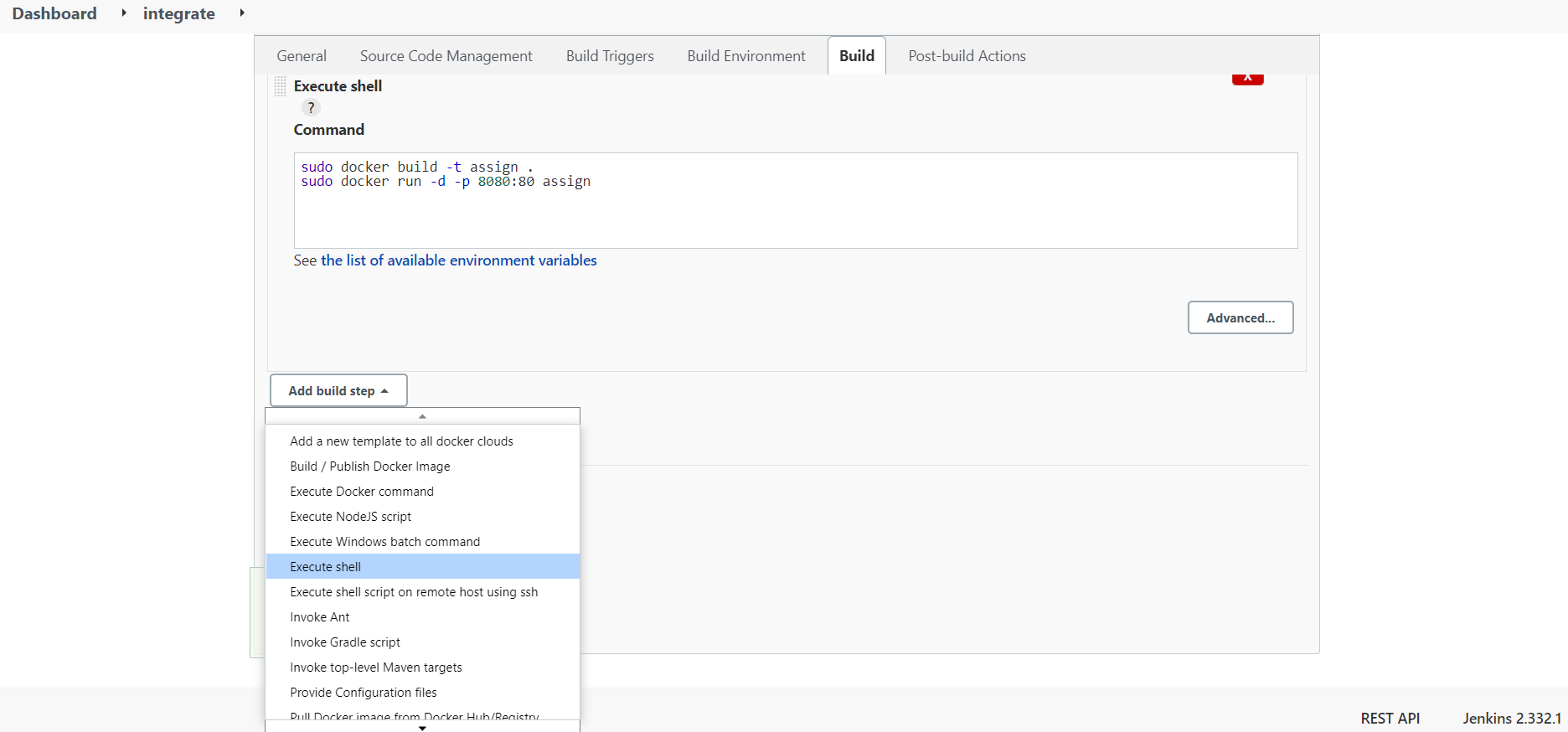
Now, go back to the Github repo and add the Dockerfile



Commit the changes that you made.

Going back to the Jenkins console,

Dashboard → integrate → Configure → Build → Add build step → Execute shell



Copy Paste the following two lines in the command box

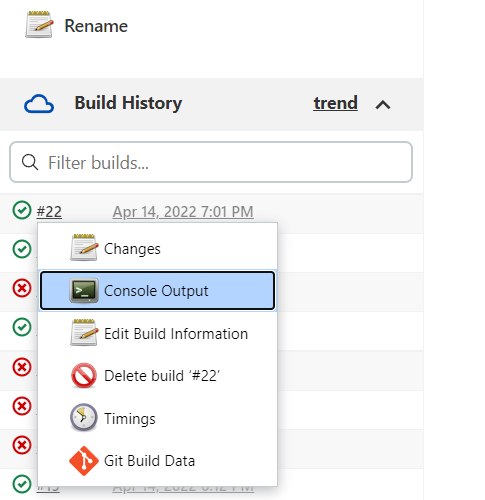
*sudo docker build -t assign .*

*sudo docker run -d -p 8080:80 assign*

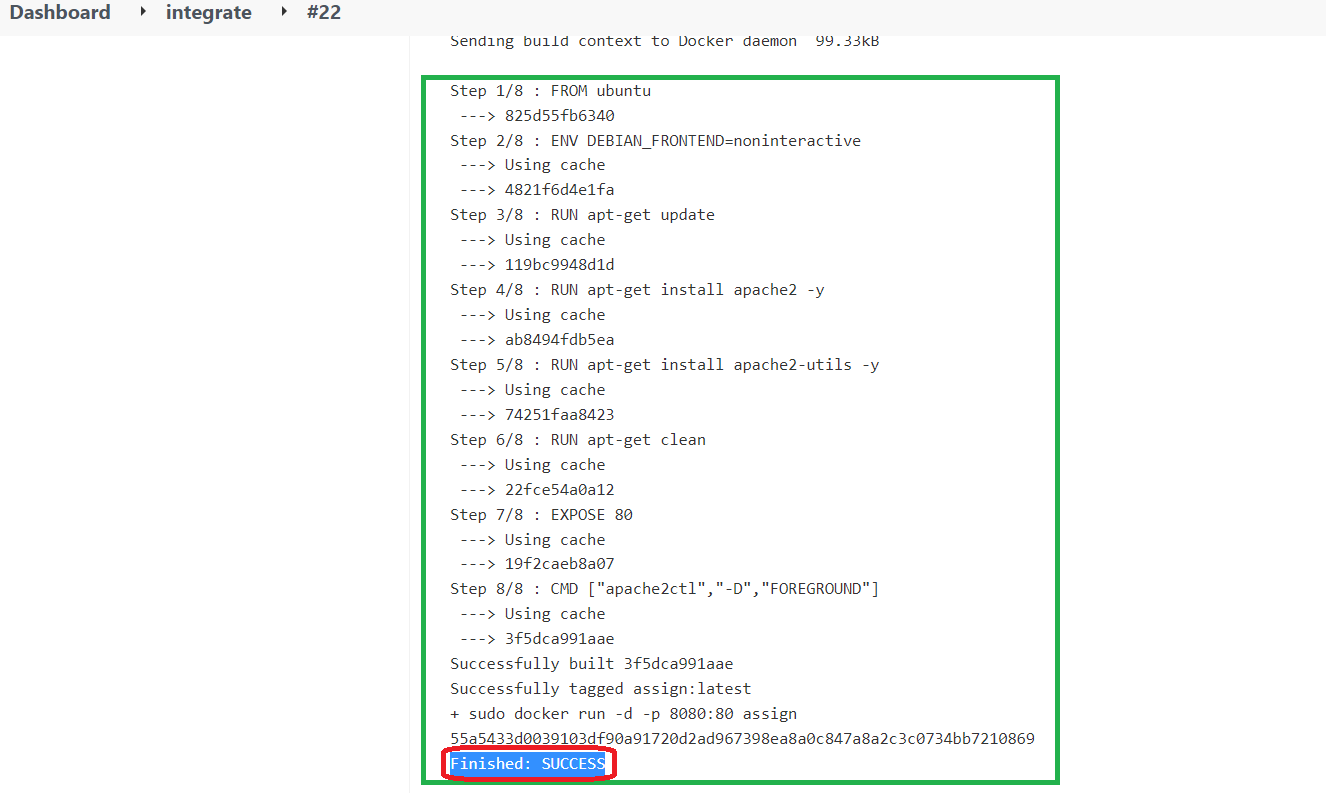
Click on **Save**

The job will be built automatically after you click on save.

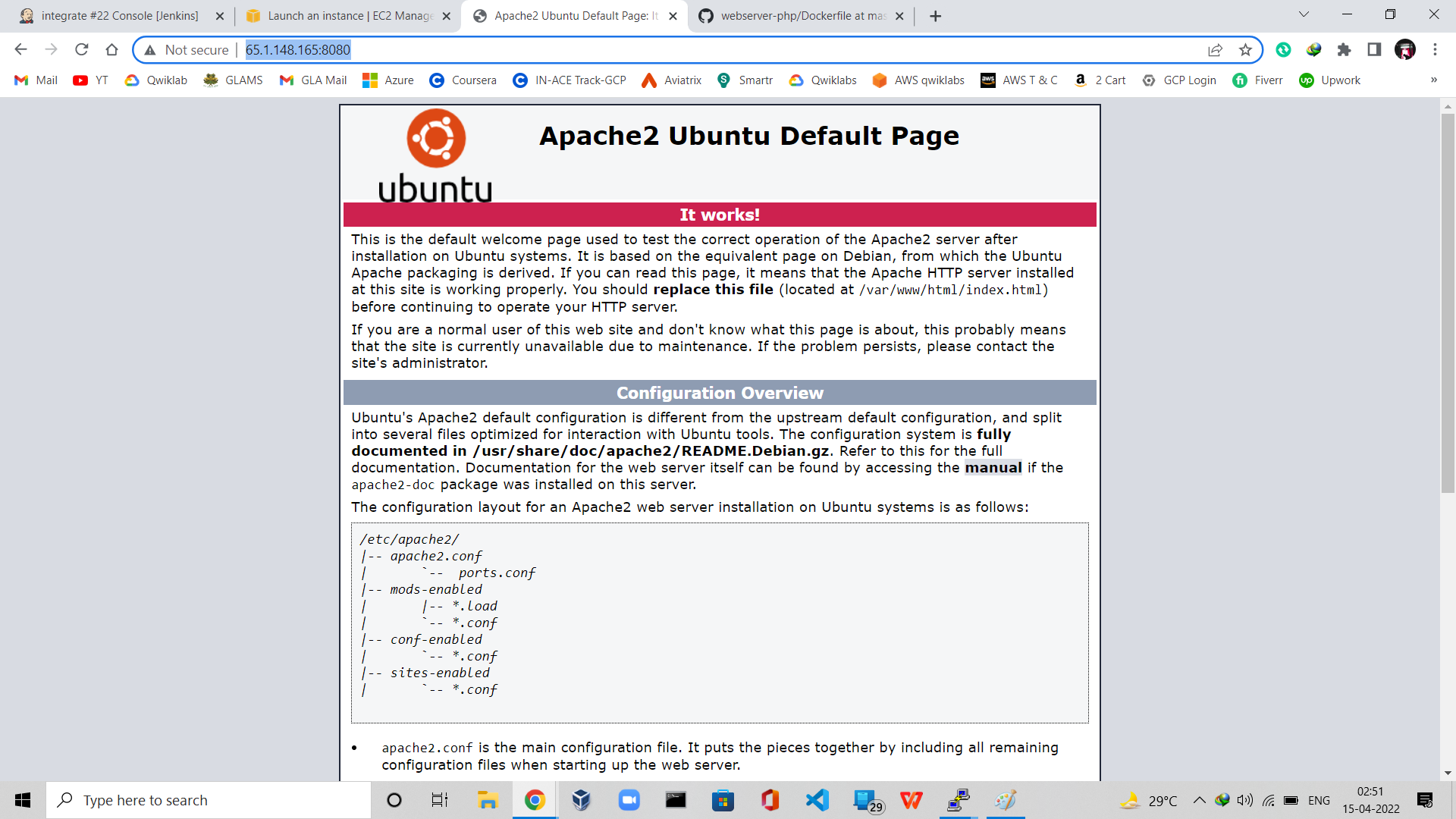
You can the console output (logs) by clicking the left new job number → Console Output



You will see the logs this way →



Copy the public IP of the instance on which you installed docker in the browser with the port number.



You will see the **OUTPUT** like this in your browser.