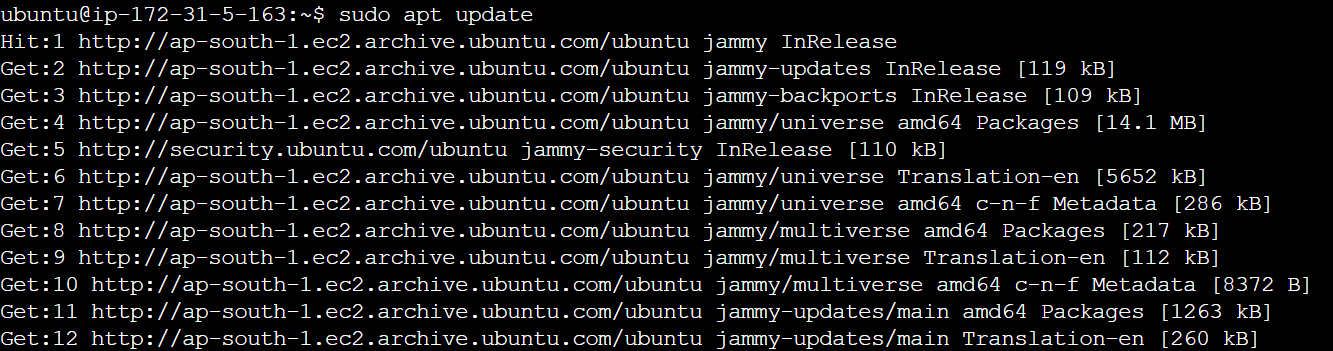
**Steps to DEPLOY a NodeJS Application by using Jenkins Pipeline along with Docker and AWS**

1. Create an AWS EC2 (Ubuntu) instance.
2. Once the instance is created, connect to the instance and update the Ubuntu (Linux) server with the command.

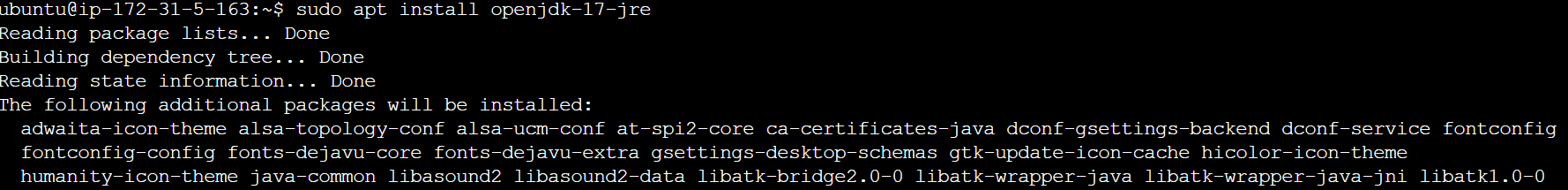
sudo apt update



**Installation and Configuration of Jenkins in Ubuntu**

1. Steps for installing and configuring Jenkins:
2. Once the server is updated, install the latest JRE (v17) package for installing Jenkins as it runs on Java platform.

sudo apt install openjdk-17-jre

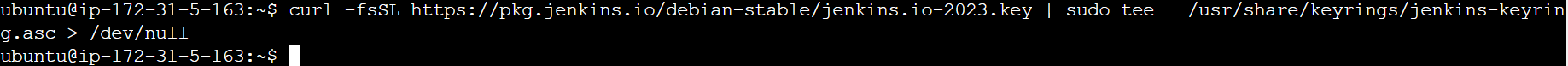


1. To check whether the JRE package is installed or not, try using this command.

java --version

1. Once the package is installed and verified, we will install Jenkins. For installing Jenkins, we need a key to install it in linux server.

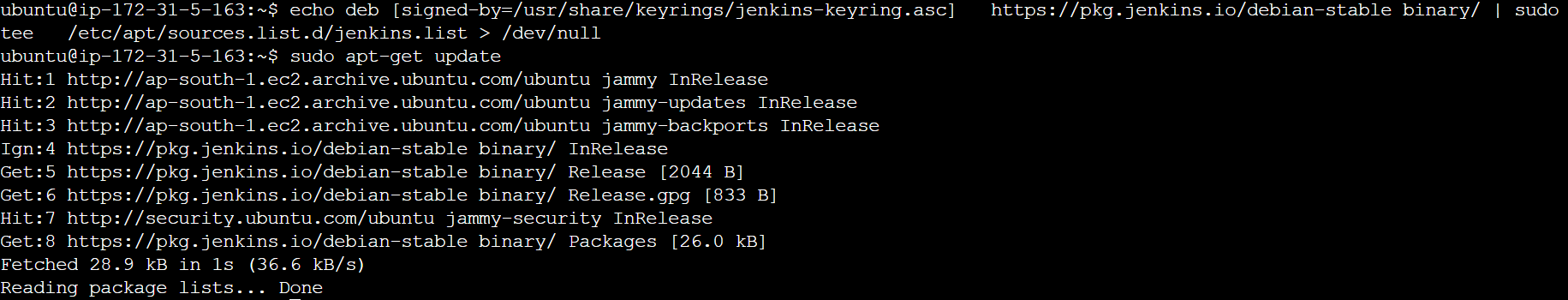
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null



1. Once the key is installed, we will push the key into the path /dev/null and we will update the server.

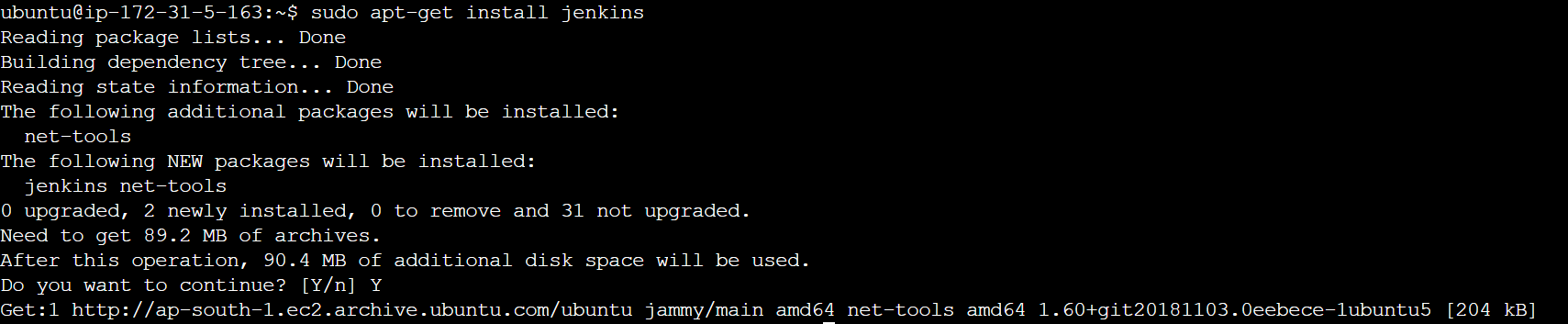
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update



1. Now we will install Jenkins as the key is installed.

sudo apt-get install jenkins



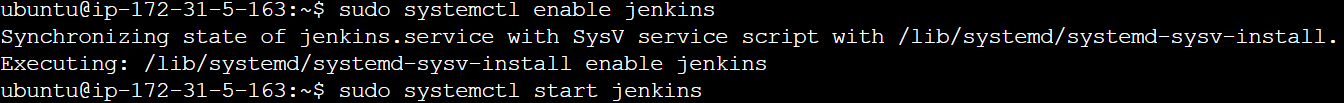
1. After installing Jenkins, we will update the Jenkins into the server.

sudo apt-get update

1. Now we will initiate the Jenkins and start the Jenkins.

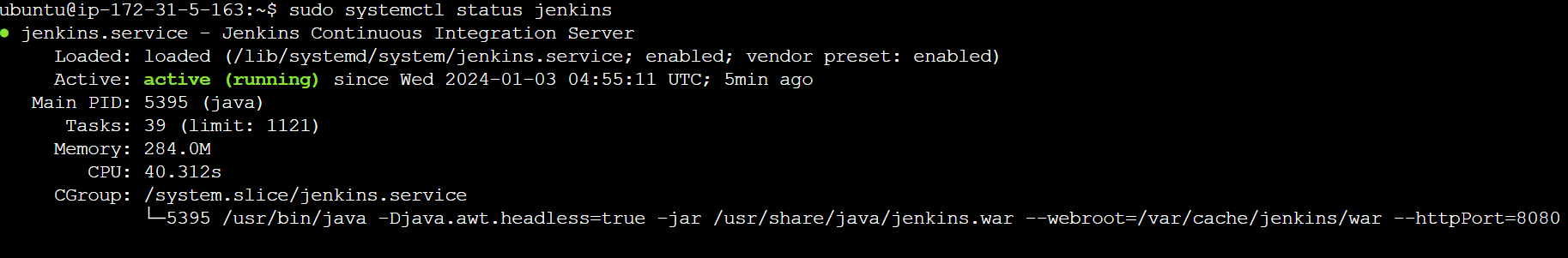
sudo systemctl enable jenkins

sudo systemctl start jenkins

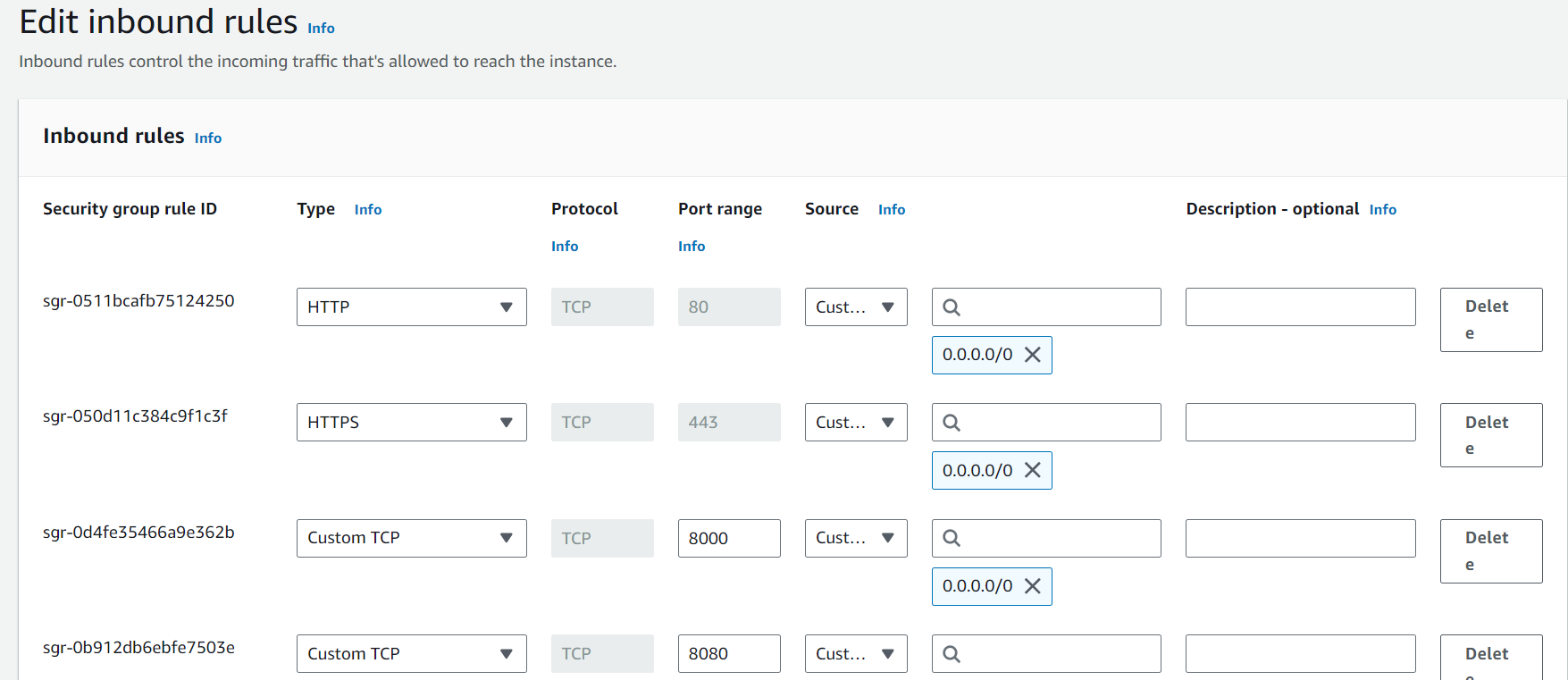


1. Try this command to check the status of Jenkins.

sudo systemctl status jenkins



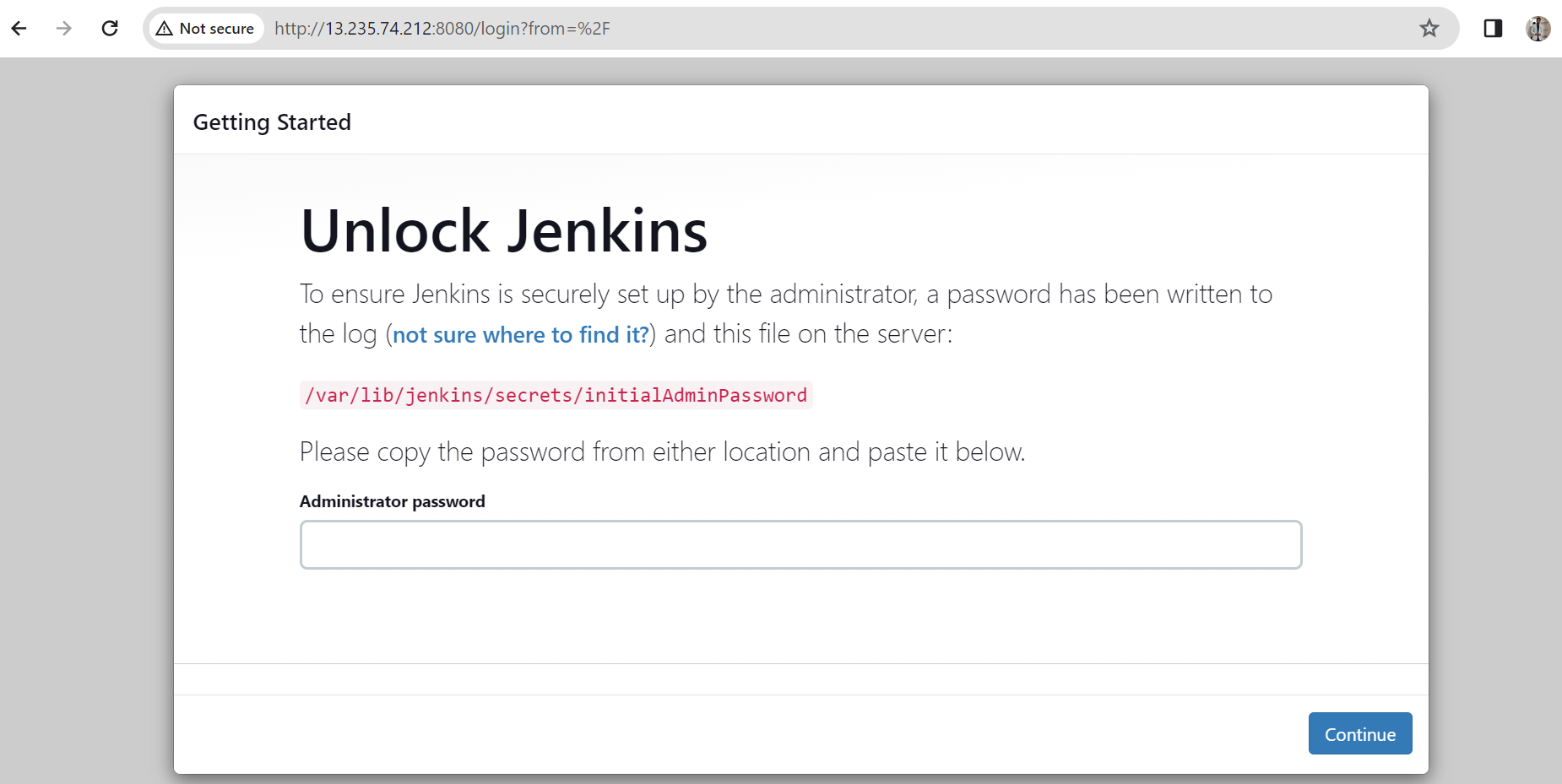
1. In the above snapshot, you can observe the port number to connect to Jenkins (8080), as it will be connecting through our EC2 instance, we need to add an inbound rule in the security group to allow the Jenkins to run.



**Note:** Try to keep the source as “My IP” as the port number should be accessed by you only for now.

1. Once you add the rule to allow the port number to run, try to run Jenkins by connecting to your instance Public IP with 8080 and a page will appear like this:

<Public IP of instance>:8080

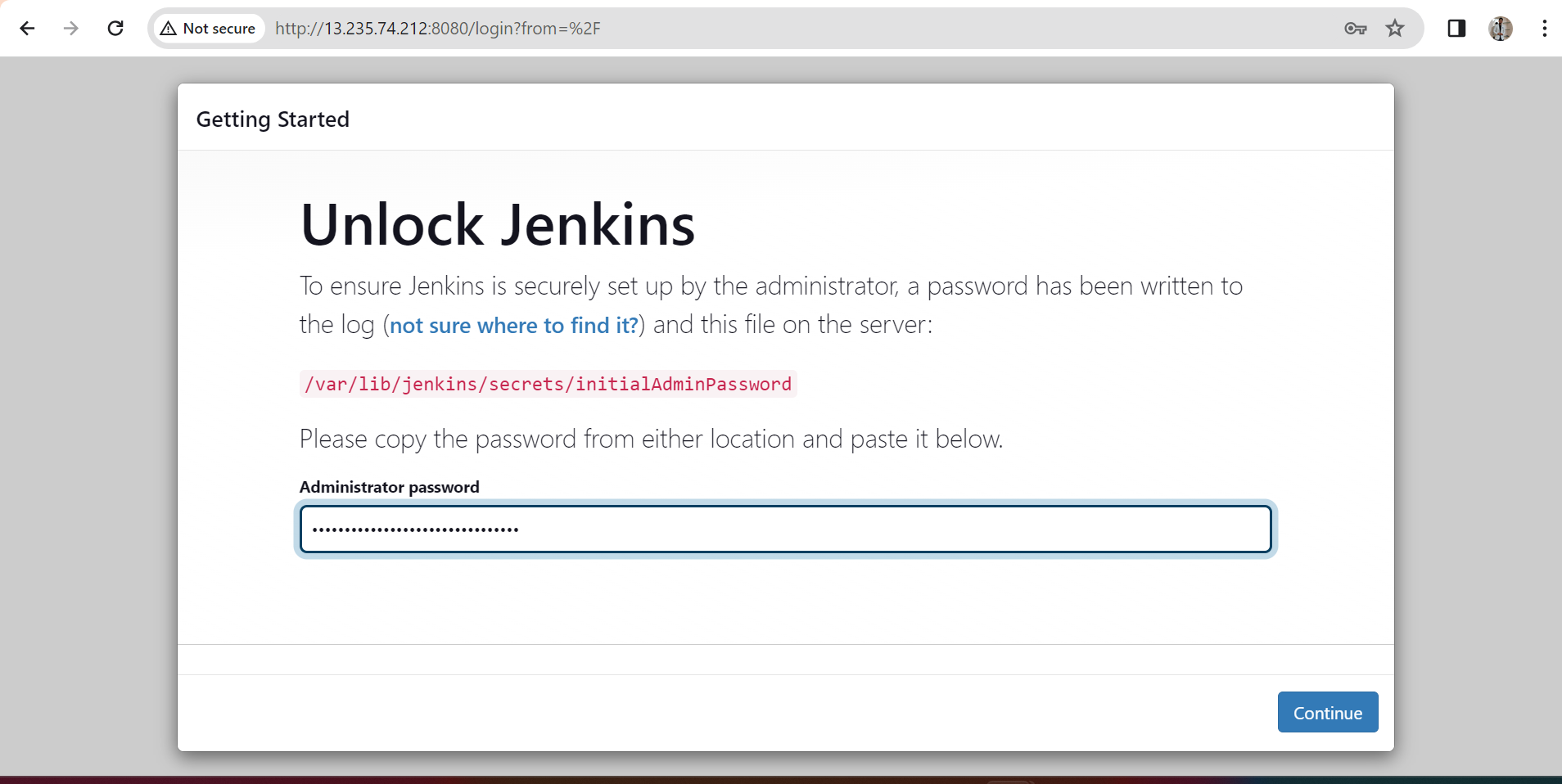


If it appears like this, then it means that you are able to connect to Jenkins.

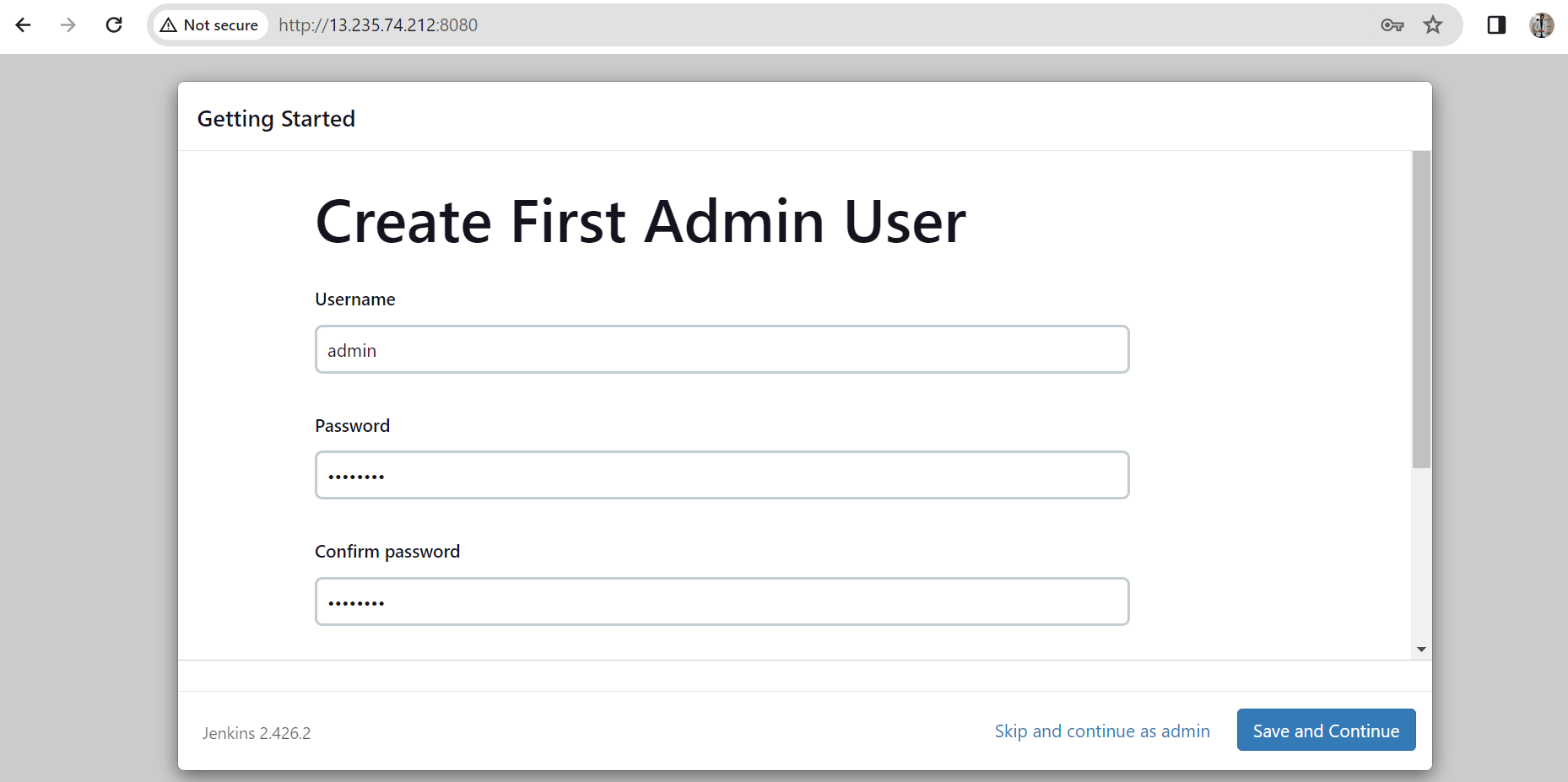
1. Try to get the password in the server from the path mentioned in the red font in the Jenkins start page and paste it in the box mentioned in the page.

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

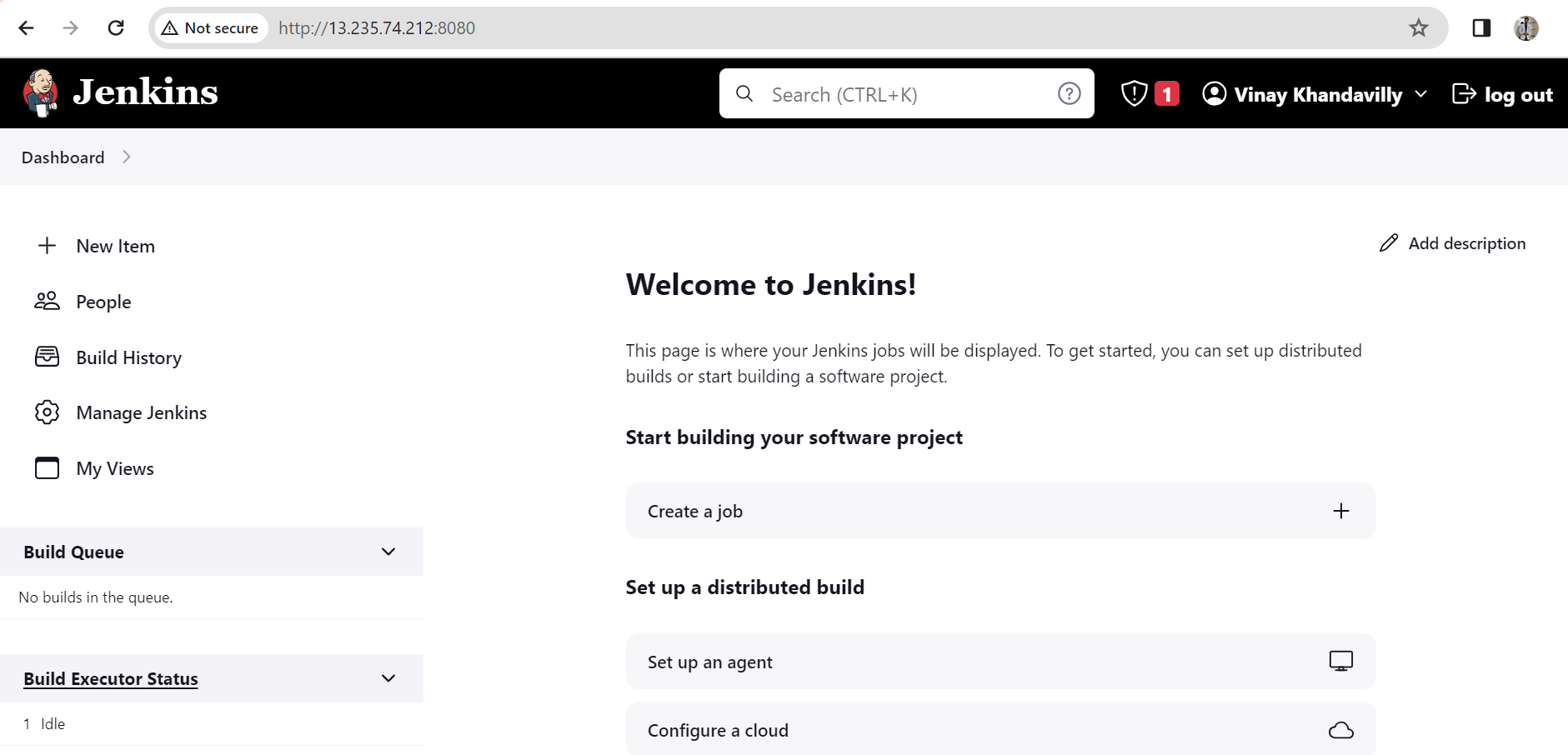




1. Once you get in to the next page, click on “Install Suggested Plugins” and continue to wait until all the plugins are installed and it asks you to change your password and save it.

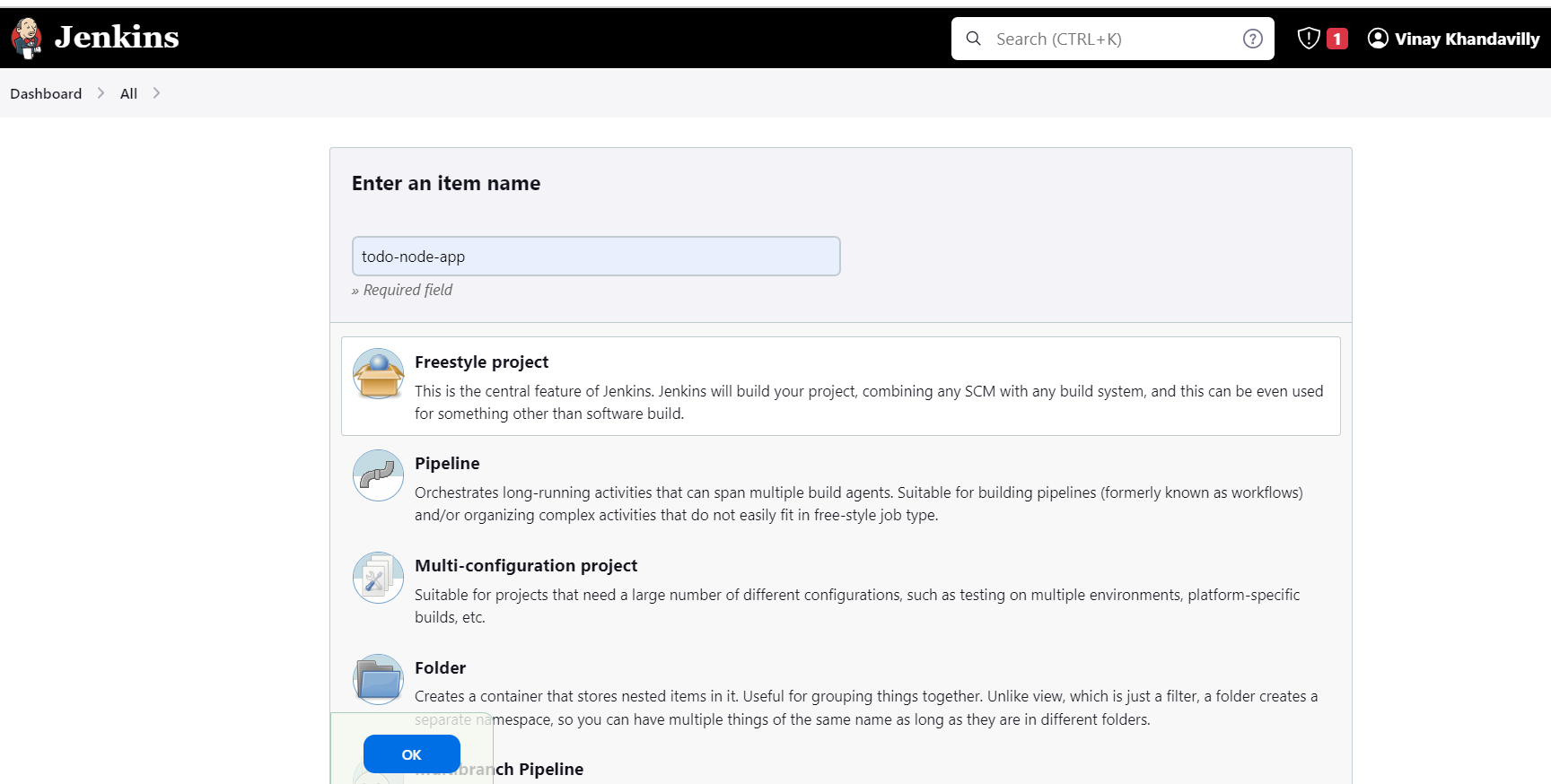


1. If everything goes well, then the page will appear like this:

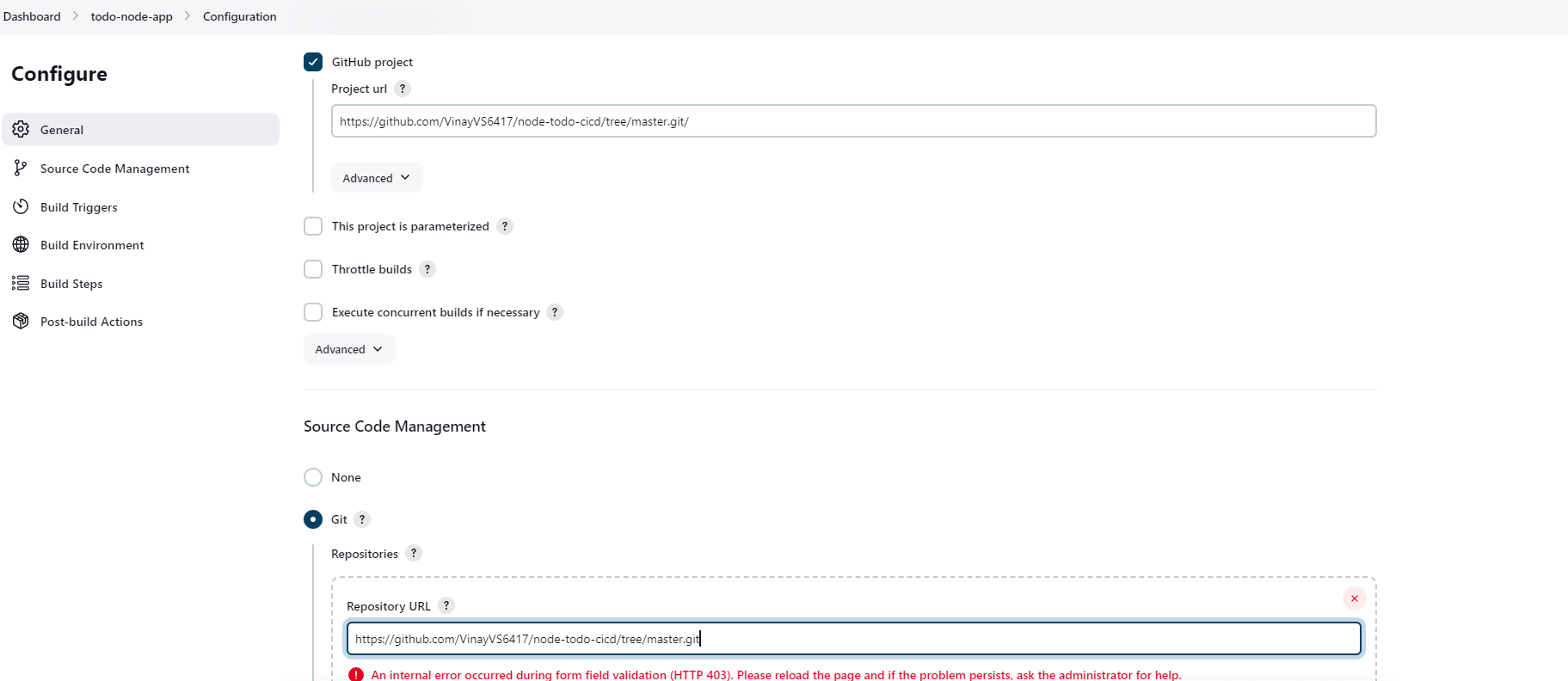


**Integration between Jenkins and GitHub**

1. Now you have to create a new item in Jenkins Dashboard where you will configure the integration between Jenkins and GitHub.
   1. You will create a Freestyle project and pull the project from GitHub.

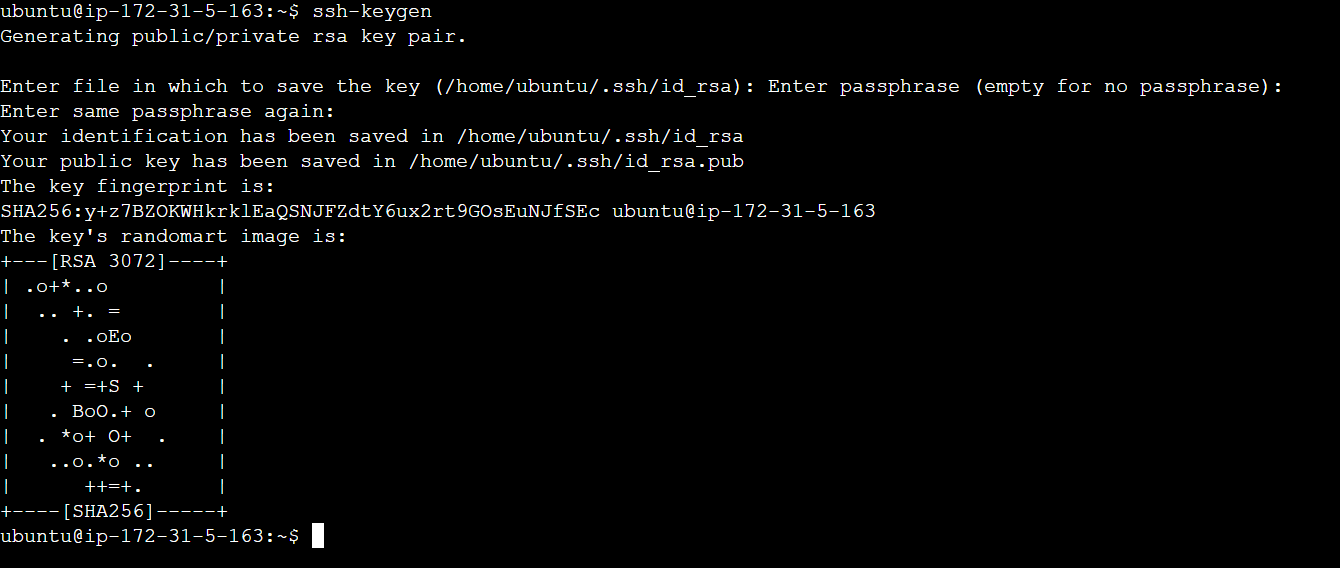


* 1. Provide required inputs as <Repository of Github URL>.git in both areas when creating the project as follows:

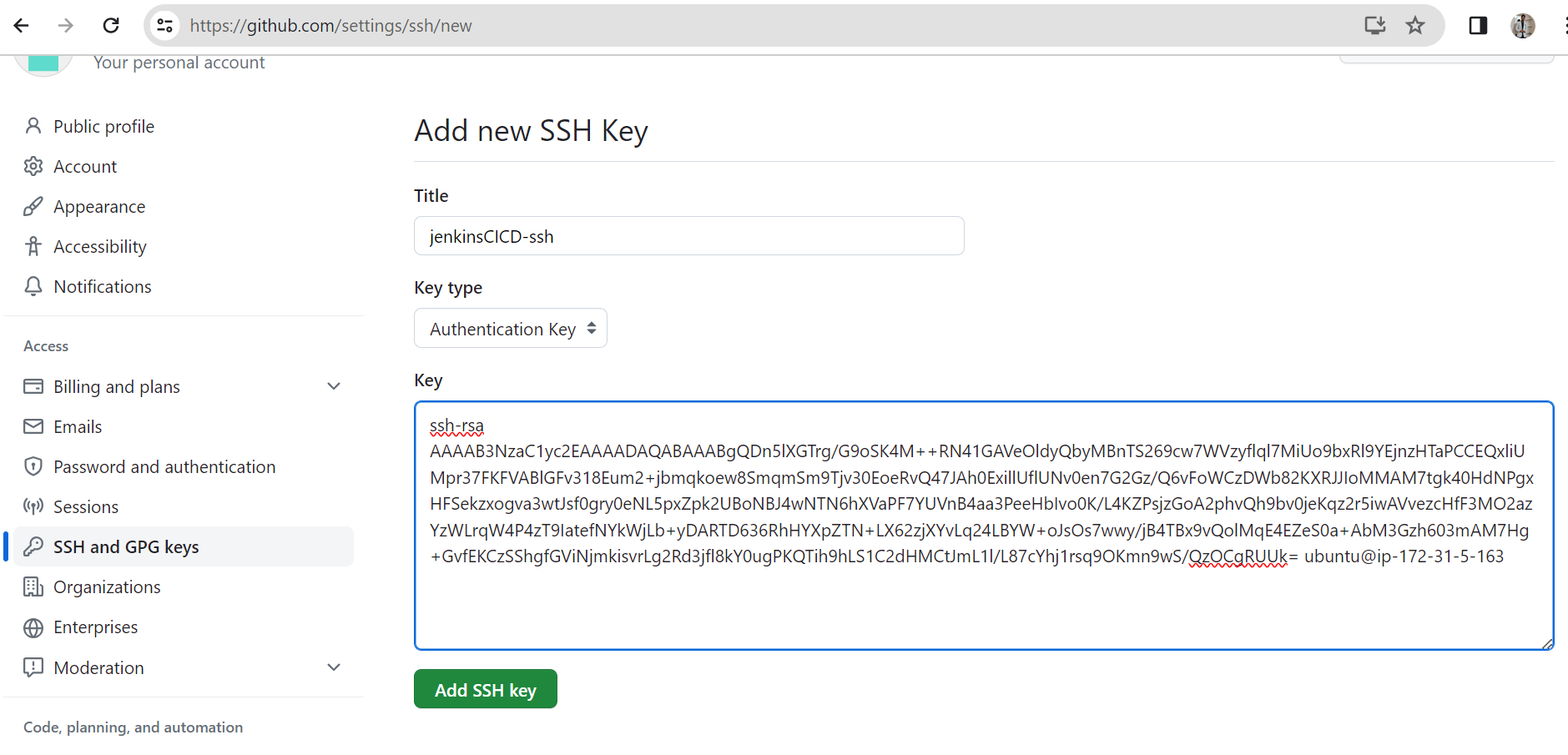


* 1. Below Repository URL in Source code Management, you need to provide the credentials for accessing the GitHub through the server. For that we need to create a SSH key in the server through which we will provide the PUBLIC\_KEY in Github and provide the PRIVATE\_KEY in the Jenkins and make the connection between them.

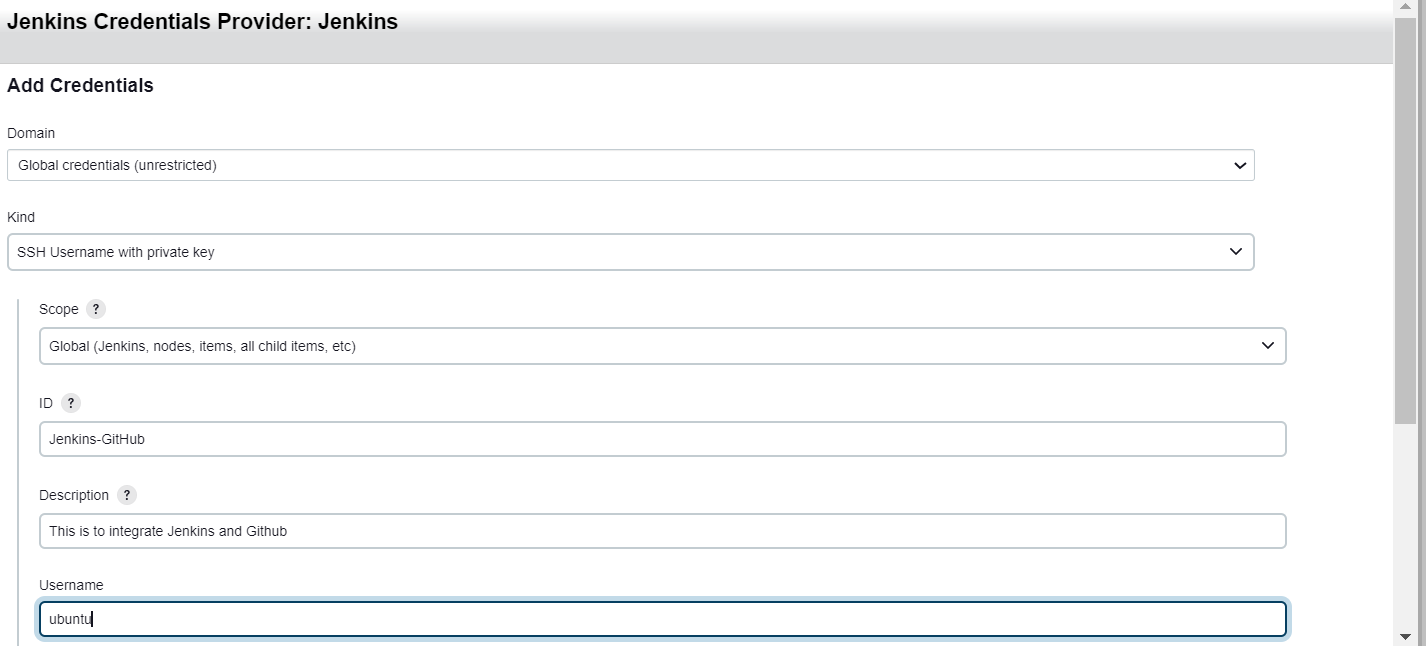
Generation of SSH key in the server

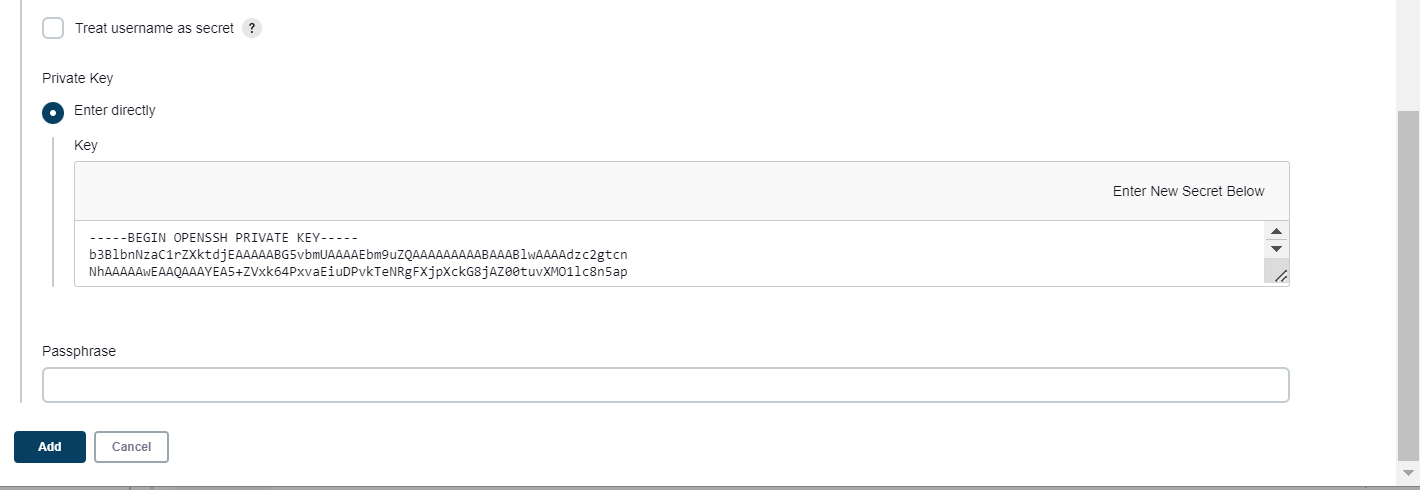


Adding the PUBLIC\_KEY in GitHub (Profile -> SSH and GPG keys -> Add new SSH key)

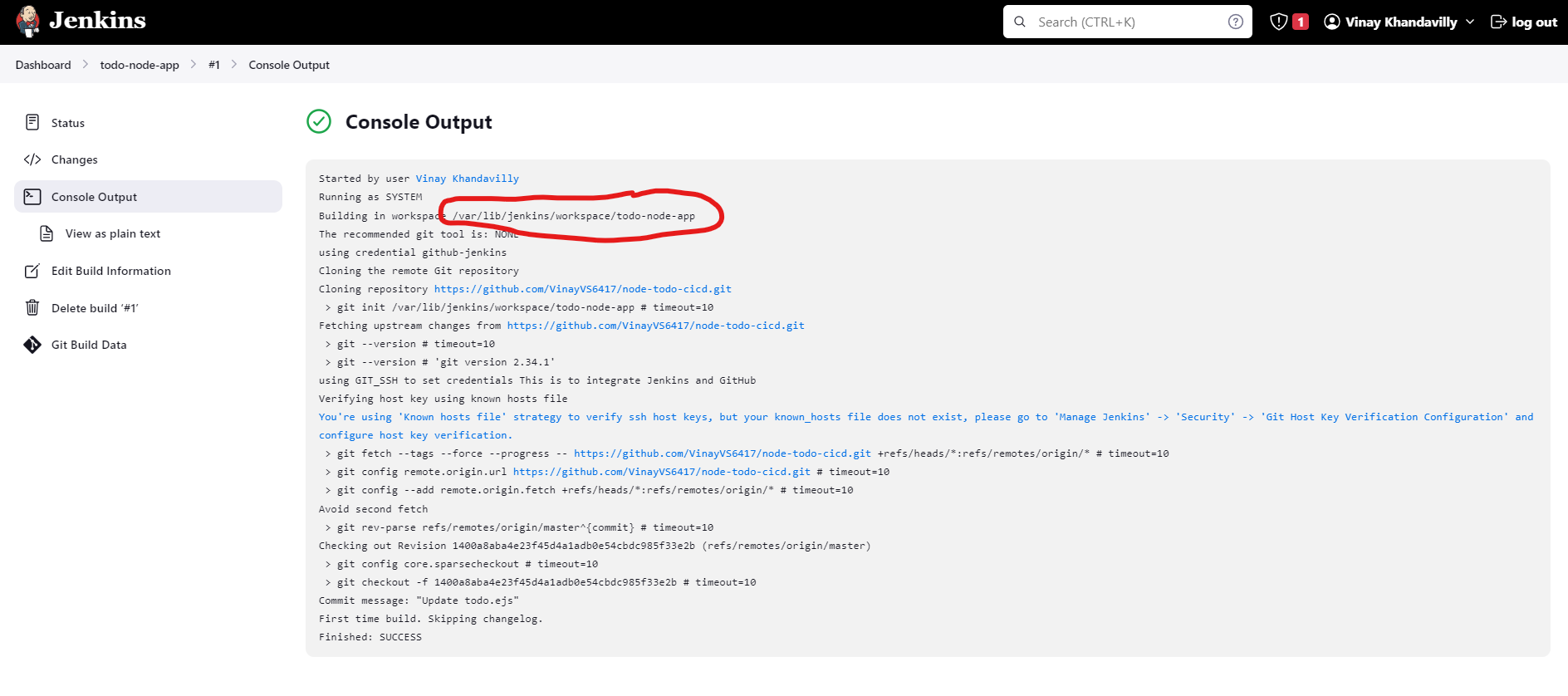


Adding PRIVATE\_KEY in Jenkins under the Repository URL in the Source Code Management.





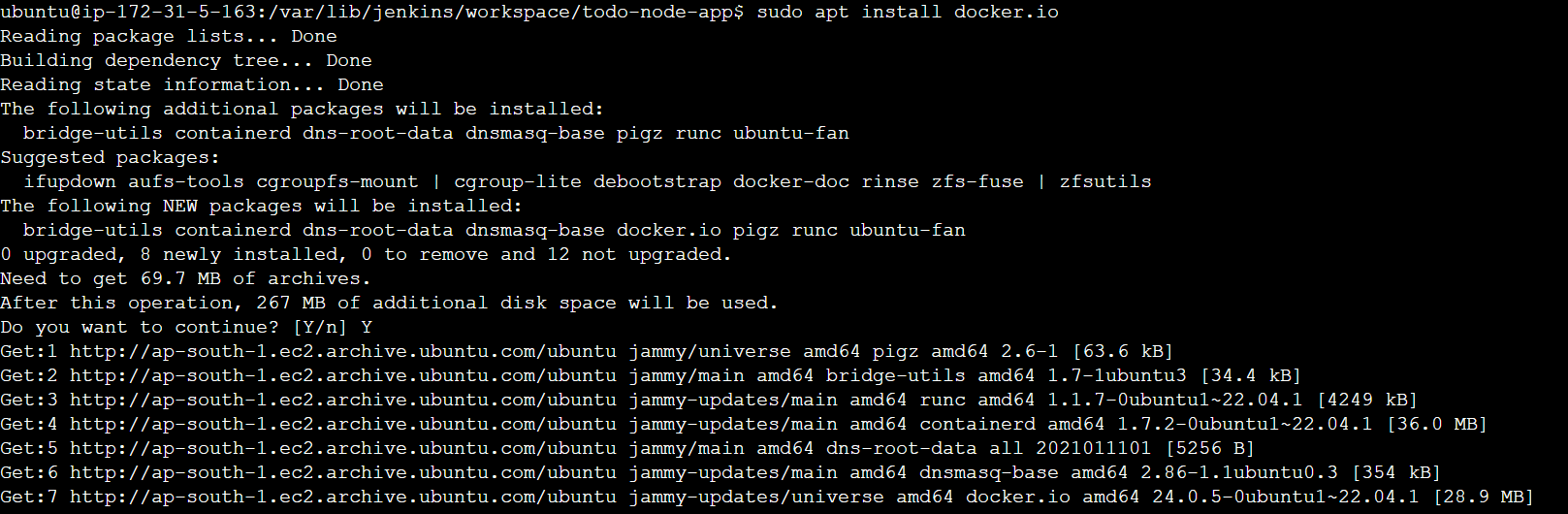
1. Once the configuration of Integration between Jenkins and Github is done, if you start building the job now in Jenkins, you will be able to see the project related files in the path mentioned in the snapshot inside the server what was present in the GitHub.



**Installation, Configuration of Docker file & Integration between Docker and Jenkins to run the application**

1. After setting up the connection between Jenkins and GitHub, we will now be able to see the files what were present in repository in our instance. Now let’s try to run the application in the instance by Docker and connecting Jenkins with Docker.

sudo apt install docker.io



1. After the docker is installed, install nodejs and npm as the application will be running on node.

sudo apt install nodejs

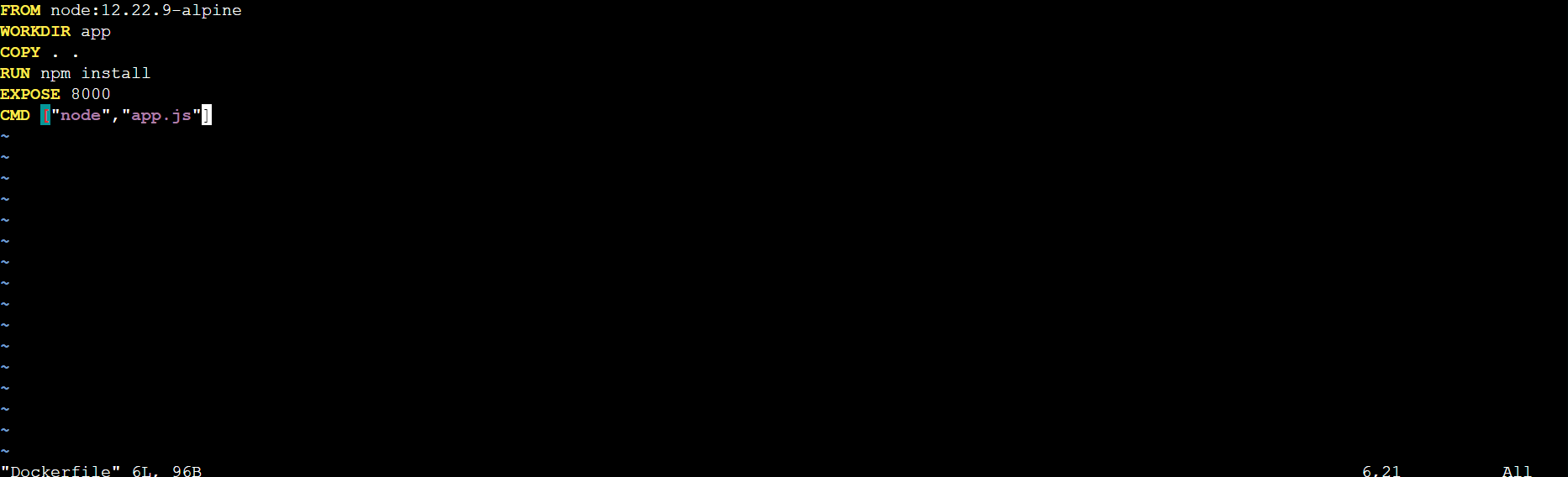
sudo apt install npm

1. We will be creating a Dockerfile with the configuration details what we will be needed to run the Node application:

sudo vim Dockerfile (to create the dockerfile in the server)



Inputs inside the DockerFile:

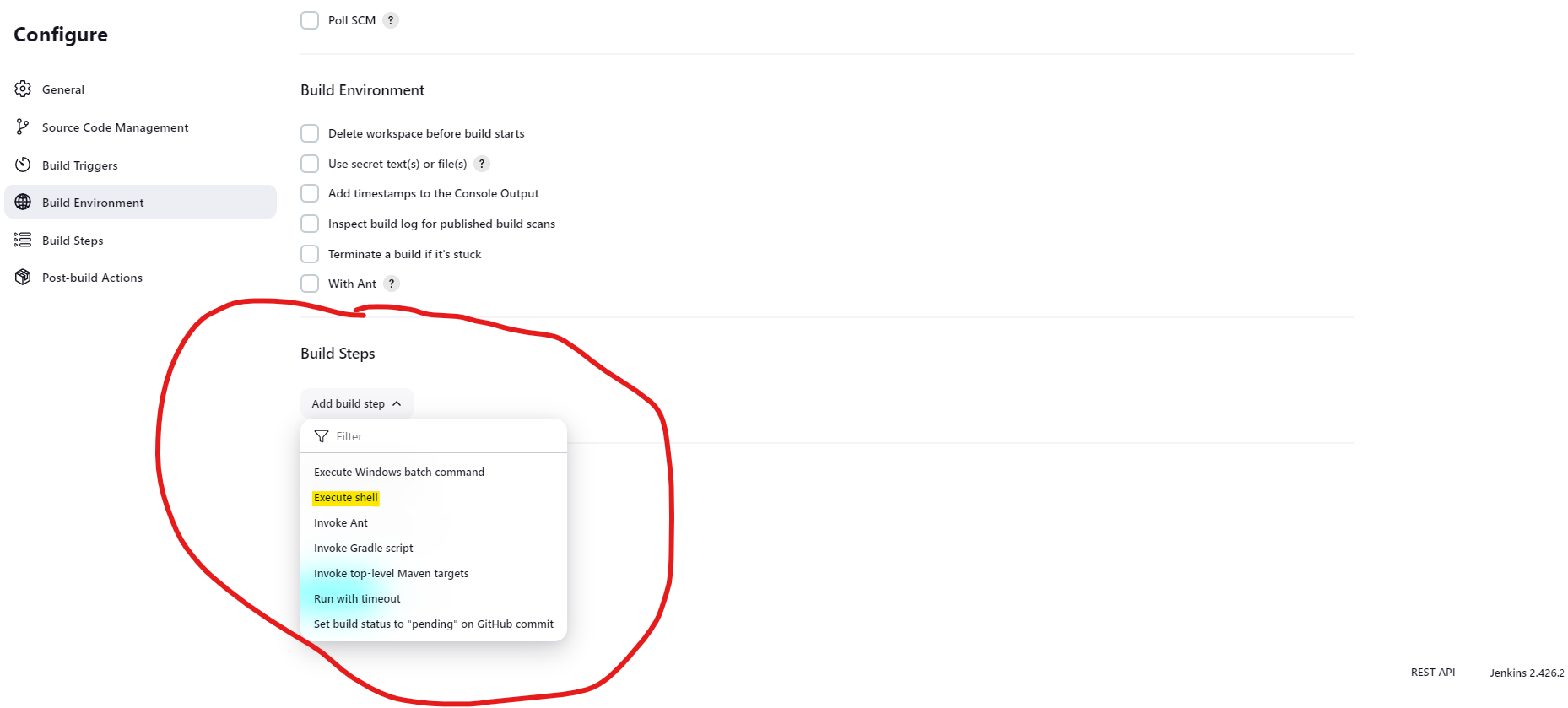


1. To run the docker file in the Jenkins, you need to give usermod permissions in the server and restart both Jenkins and server to get the command configured:

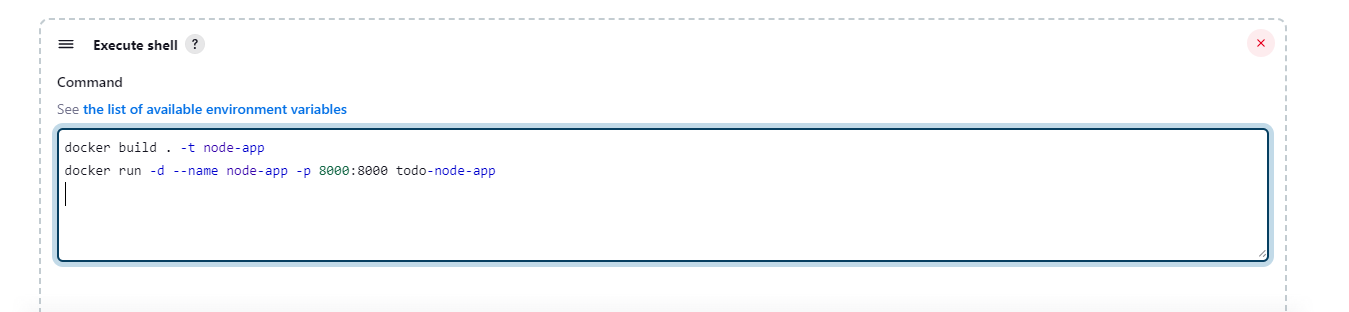
sudo usermod -a -G docker jenkins

1. Once you have given the permission to the file, now we need to add few inputs in the Jenkins to make it integrate it with the Docker as well as it will make the application run once the integration is done.

Go to Jenkins job, and click on “Add Build Steps” and click on “Execute Shell” and give the commands what you want to run the docker file.

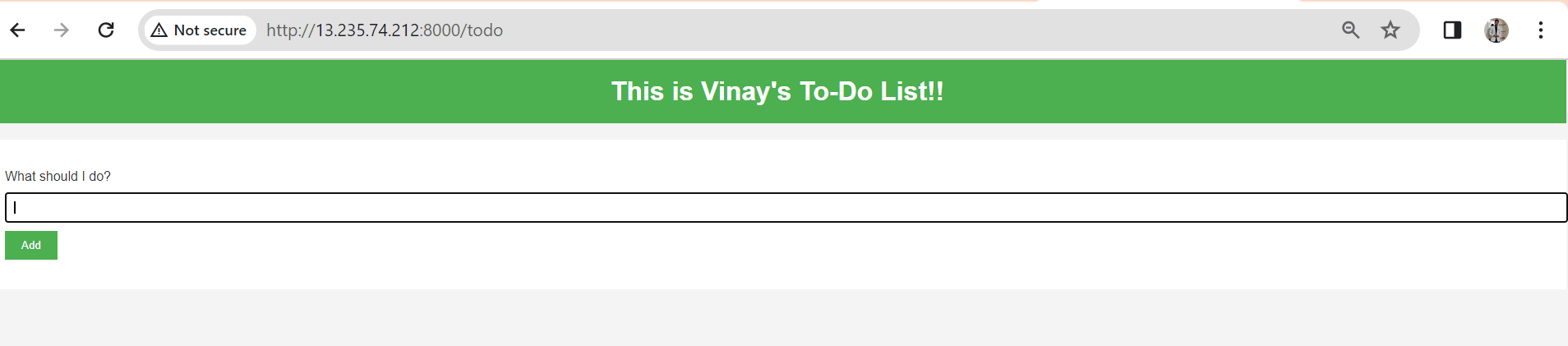


The first command (docker build . -t node-app) will build the docker image and docker container of the docker file and the second command (docker run -d --name node-app -p 8000:8000 todo-node-app) will run the docker container where “8000:8000” will bind both the ports present in the instance as well as the docker.



**Note:** Before running the “Build now” command in Jenkins job, kindly allow a inbound rule in security group for port 8000 so that it will allow you to run the application.

1. Once you are done giving all the commands, click on “Build Now” in Jenkins job and sit back and wait to see the build getting successful. Once the job is succeeded, open a new tab and type <PUBLIC\_IP>:8000 and you will see the result.

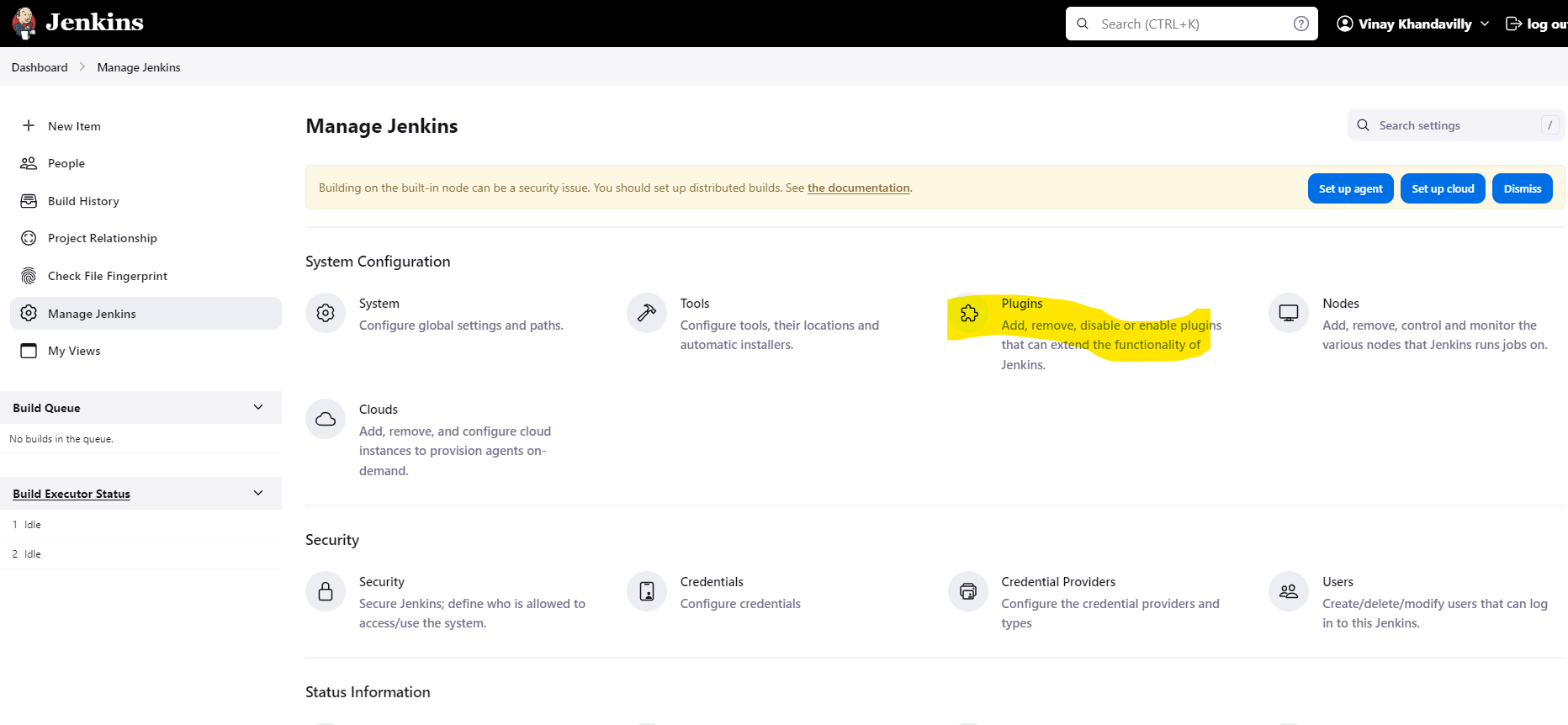


**Configuring WEBHOOK between Jenkins & Github**

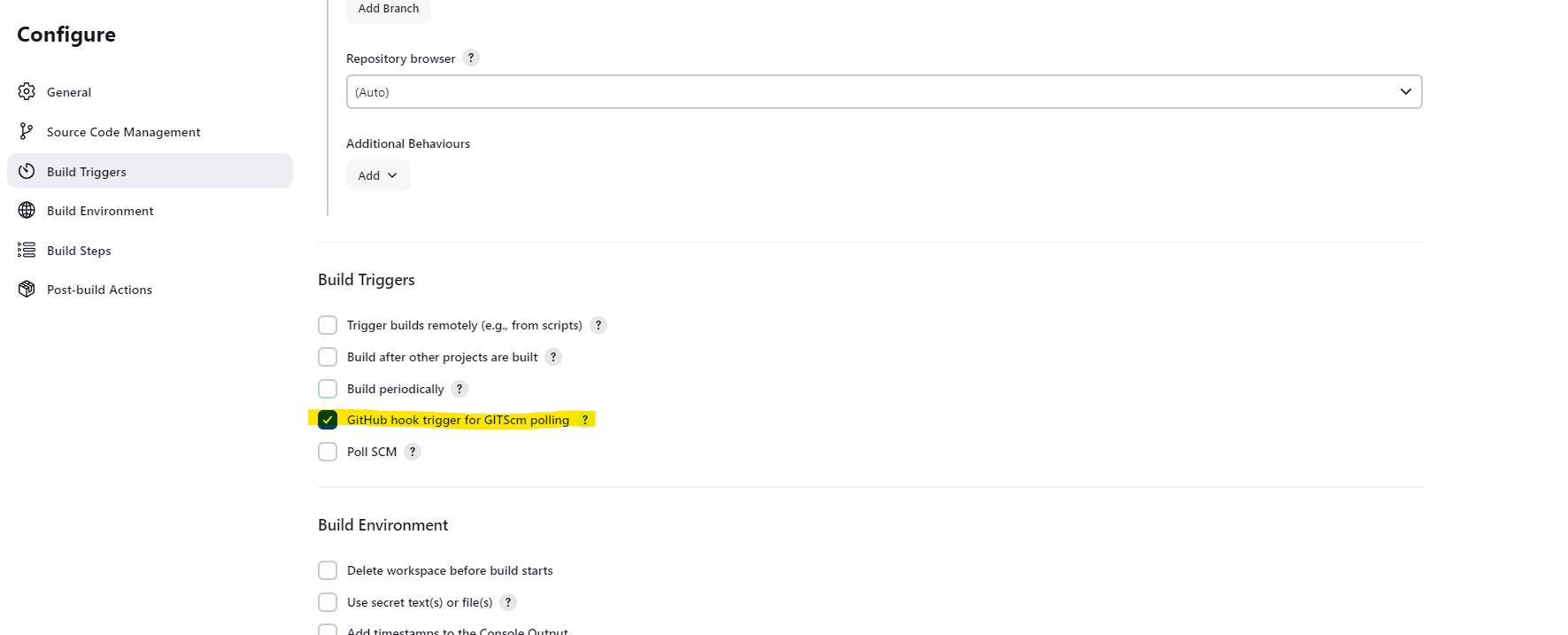
Webhook is used to connect Jenkins and Github & it is useful whenever we are making any changes in the application file in GitHub, Jenkins will get triggered and it will start building the change made in GitHub. This process we often can say as CONTINUOUS DEPLOYMENT.

To setup the webhook in both the Jenkins and Github:

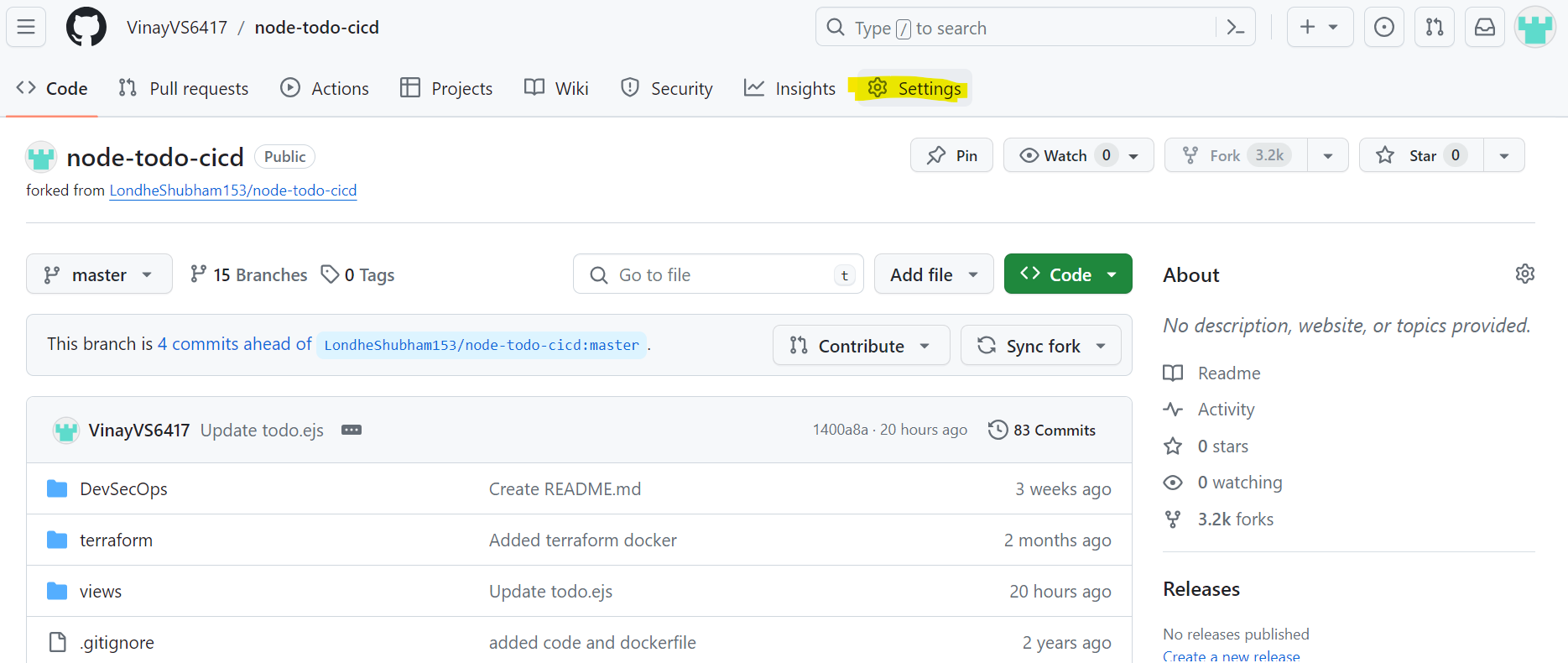
1. Go to Manage Jenkins present in Jenkins Dashoboard and click on plugins and search for GitHub Integration in available plugins. Once the plugin is installed, it will ask you for the restart of the Jenkins.



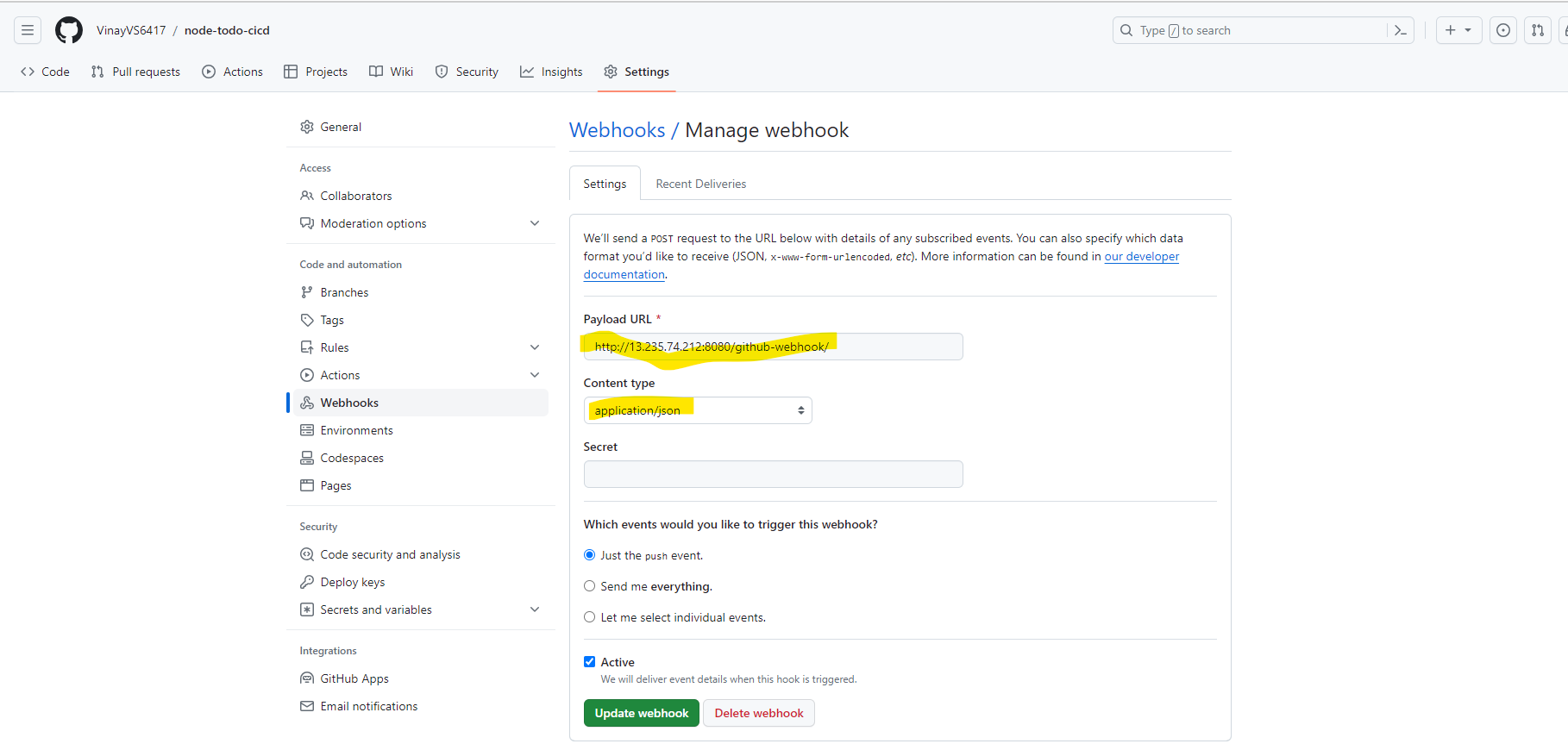
1. Go to Jenkins job and check the box “GitHub hook trigger for GitSCM polling” present in Build Triggers and save it.



1. After configuring Jenkins, go to GitHub and open the settings option present for the repository.



1. In the settings, go to Webhooks option and add the inputs to be provided and click on “Add webhook”.



Once you integrate Webhook between GitHub and Jenkins, now try to change something in the file and you can see the build is happening automatically.