**Task: CI/CD pipeline design and tool selection**

**Overview**

**CI/CD** is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment.

Therefore, CI/CD has been the main component of the software development cycle, with so many configurations and tools available. We will use Amazon Web [Services](https://www.mitrais.com/softwaredevelopment/) (AWS) as the cloud platform, GitHub for the code repository, Jenkins for the Continuous Integration (CI), and Continuous Delivery (CD).

**Requirements:**

* AWS account.
* Jenkins is to be installed on the AWS EC2 Instance
* GitHub repo (code)

## Step 1: Setup AWS EC2 Instance

We will first create an AWS Instance (Ubuntu) free-tier eligible using the AWS console.

## **Steps To launch the EC2 instance:**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose Launch Instance.
3. Once the Launch an instance window opens, provide the name of your EC2 Instance.
4. Choose the Ubuntu Image (AMI)
5. Choose an Instance Type. Select t2.micro for our use case which is also free-tier eligible.
6. Select an already existing key pair or create a new key pair. In my case, I will select an existing key pair.
7. Edit the Network Settings, Create a new Security Group, and select the default VPC with Auto-assign public IP in enable mode. Name your security group and allow ssh traffic, HTTPS, and HTTP from anywhere (later on we can modify the rules)
8. Leave the rest of the options as default and click on the Launch instance button:
9. On the screen you can see a success message after the successful creation of the EC2 instance, click on Connect to instance button
10. Now connect to instance wizard will open, go to SSH client tab and copy the provided chmod and SSH command

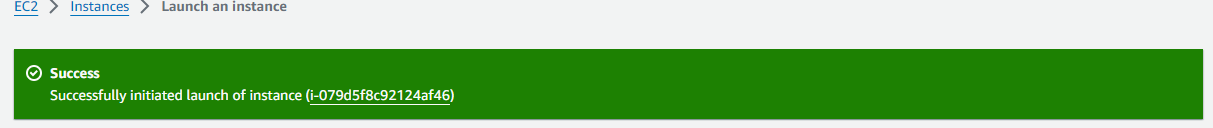


Fig. Launch Instance

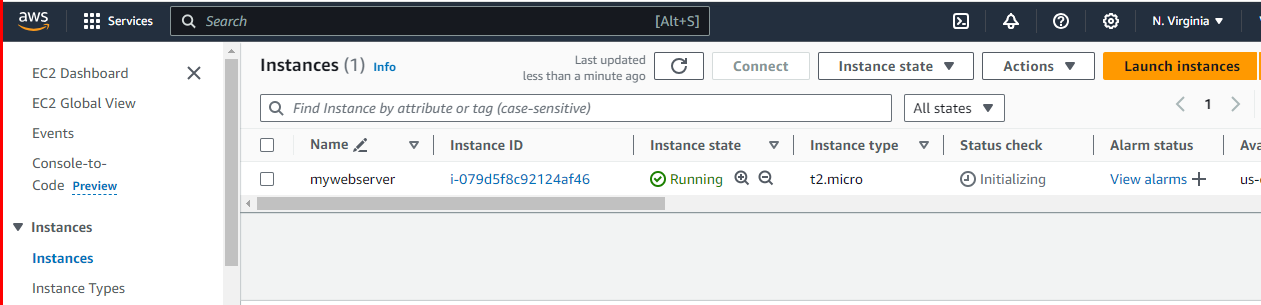


Fig. List Instance

1. Open Git bash in your local machine and paste the following two commands and you will be able to access your EC2 machine.

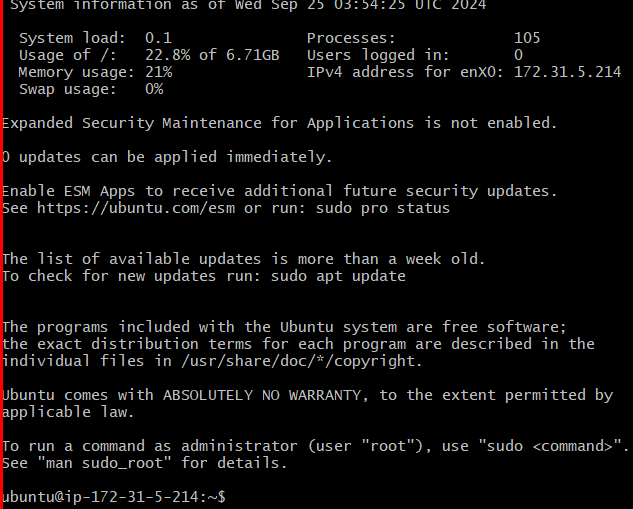


Fig. Connect to GitBash

## **Step 2: Install Jenkins on EC2 Instance:**

Jenkins installation is straightforward:

1. First Update your Server with the command

“sudo apt -get update”

1. Install Java :

sudo apt install openjdk-17-jre

1. Verify Java Installation:

java --version

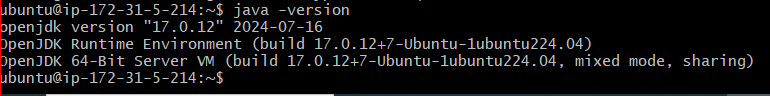


Fig. To Check Java version

**4. Install Jenkins:**

Just copy these commands and paste them onto your terminal.

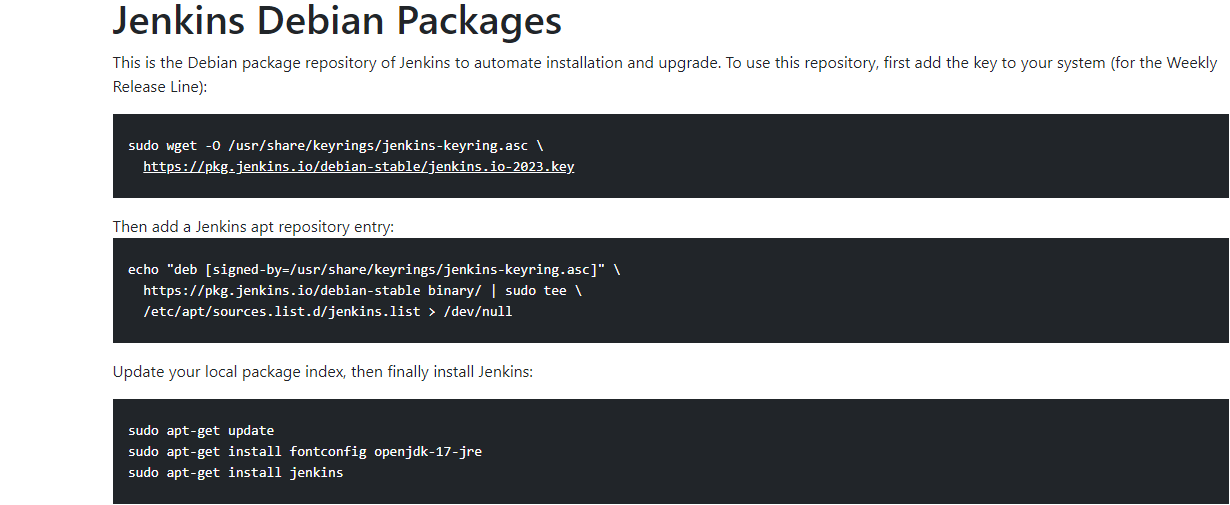
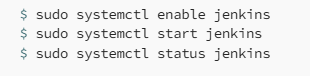
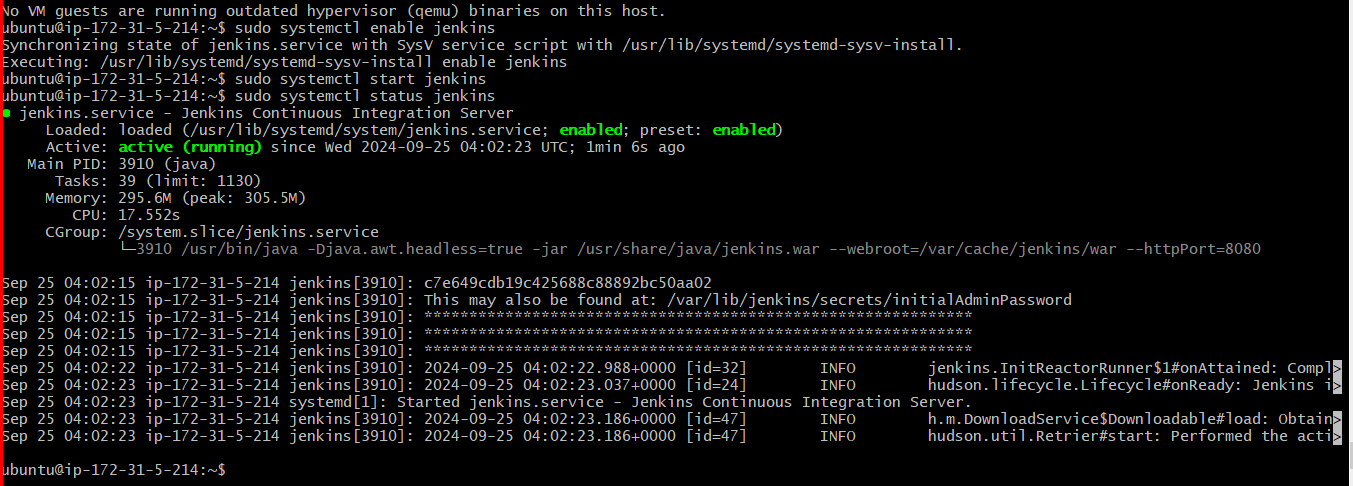


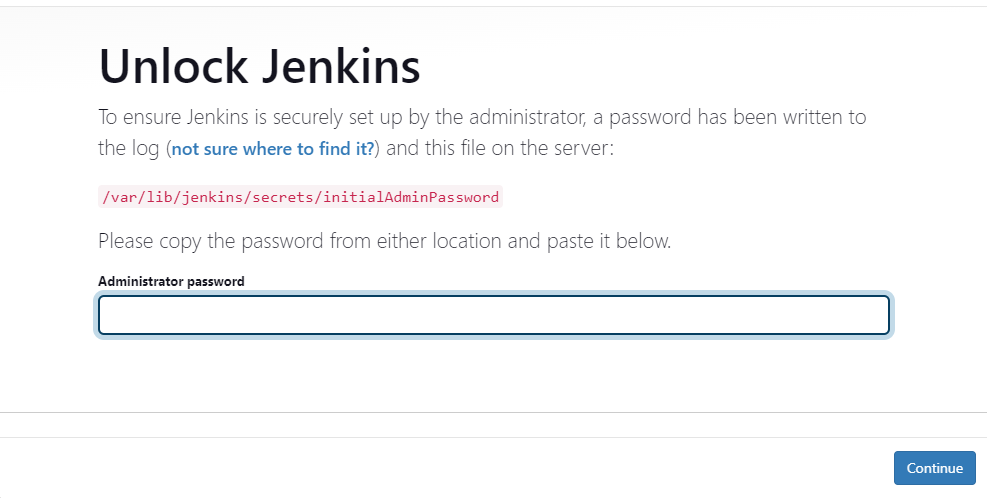
Fig. Jenkins installation packages

5. Start Jenkins:





To verify the installation of Jenkins we can take the public IP of our EC2 Instance and paste it into our browser with default port of Jenkins 8080. The Jenkins page would be appear below.



**Fig. Jenkins page**

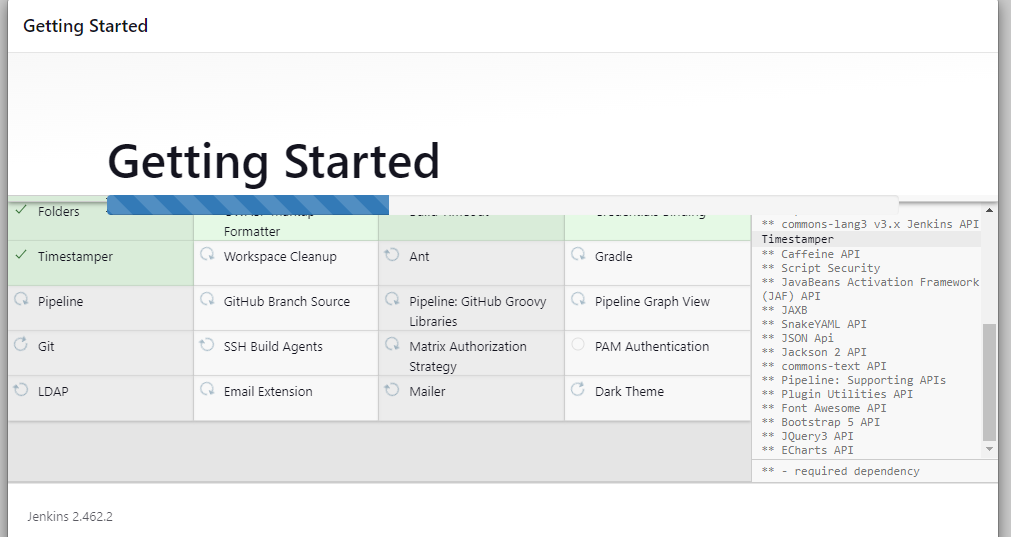
Now to unlock Jenkins we need to go to this path /var/lib/jenkins/secrets/initialAdminPassword and copy the administrative password and paste it into the above screenshot.

After pasting the password the below page will appear. Click on Install suggested plugins.

C:\Users\Lenovo T450\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5C04BF3F.tmp

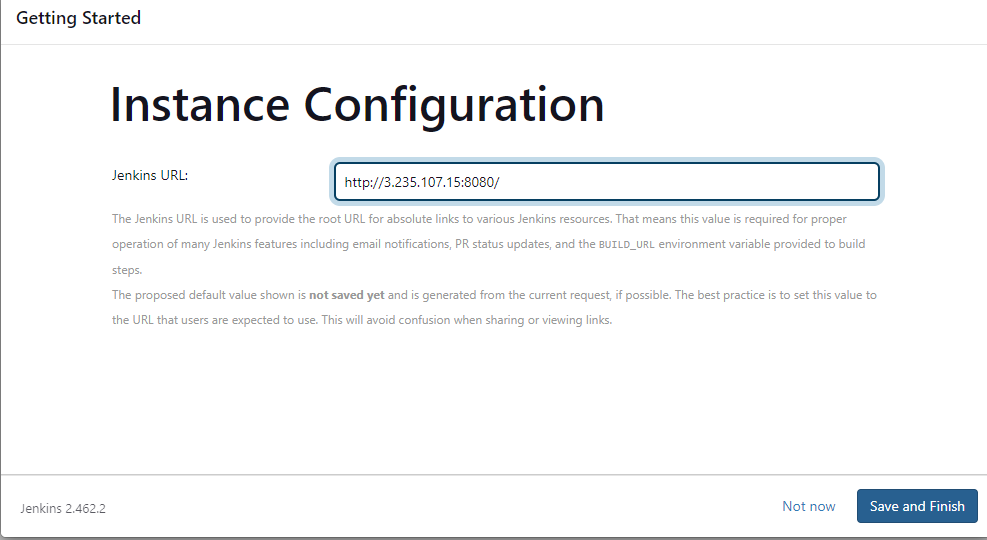
**Fig. Jenkins Admin password**

After the plugins are installed, you need to create the first Admin User. Fill in all the information and click on Save and Continue



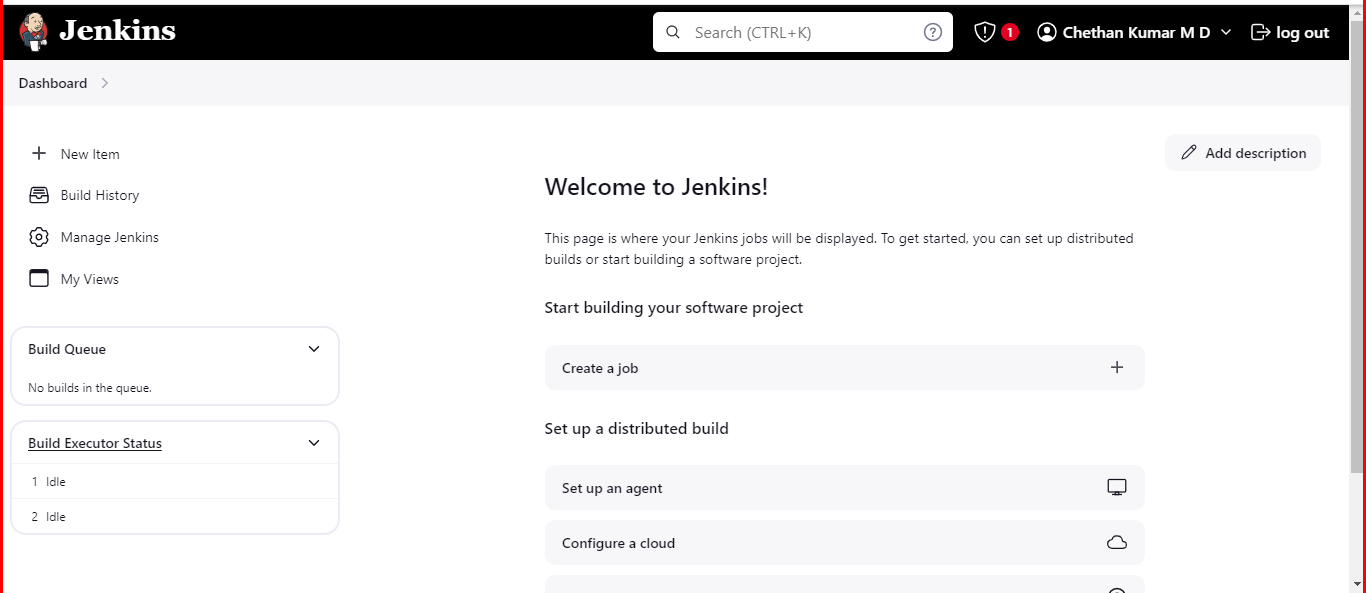
**Fig. Installing tools**

Then the wizard will display the Jenkins URL through which you will be able to access the Jenkins Server. Click on Save and Finish and start using Jenkins



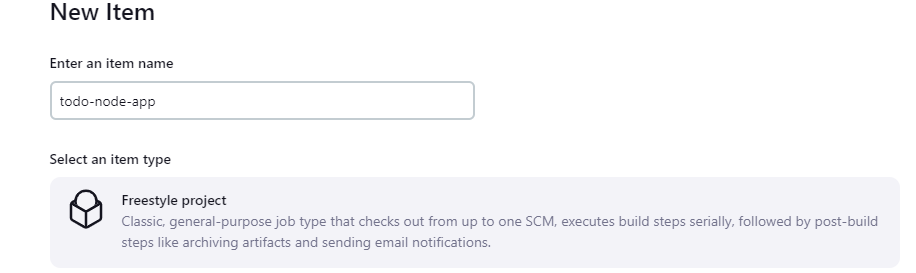
**Fig. Instance Configuration**

1. On Jenkins Dashboard, Click on New Item to create your first job or item.



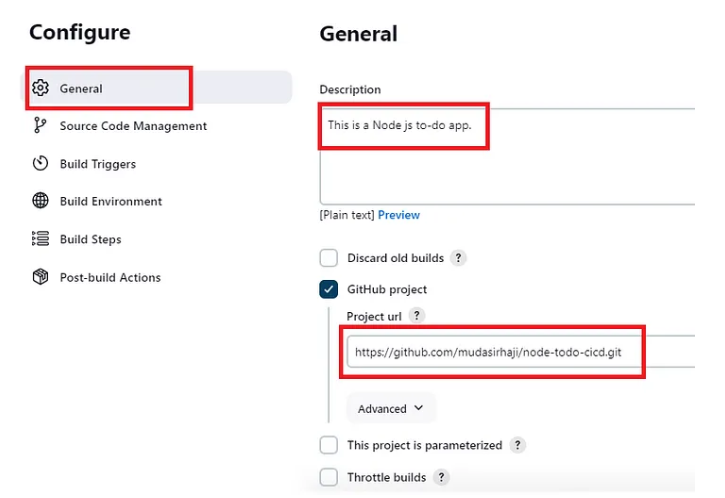
**Fig. Jenkins dashboard**

2. Enter the name of your first item and select Freestyle Project and click on Ok.



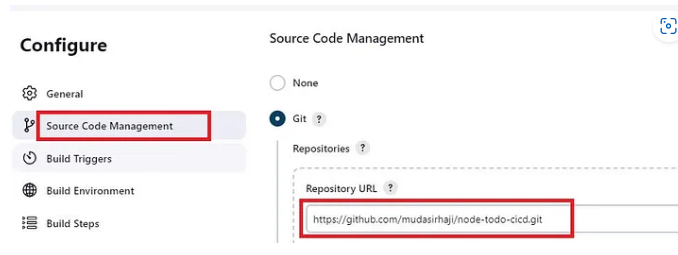
**Fig. Type of project selectrion**

.Now we need to configure various parameters of our first job in Jenkins. Under the General section, put the description of your project. Under Project Url paste the GitHub URL of the project for which we need to create the CI CD Pipeline



**Fig. Adding Git Project Url**

1. Under Source Code Management we need to paste the GitHub repository URL as shown in the below screenshot:



**Fig. Adding Git Repository Url**

Under credentials, we first need to generate the ssh keys on our EC2 Server.  
The reason to create ssh keys is to make secure integration between our Jenkins Server and Github and this way we can clone any git repo in this Jenkins instance. You do not need to provide the credentials while configuring the job in Jenkins

Take the public key and insert that under the settings in GitHub to achieve a password-less connection between Jenkins and GitHub.

For that go to the settings of your GitHub, on the left choose **SSH and GPG keys**, click on**New SSH key,** give the name unde**r Title,**and paste the public key under the **Key** section we previously generated on the EC2 server.

After adding the ssh key to our GitHub account we will add the credentials in Jenkins. For that, under credentials click on Add button and click on Jenkins in the drop-down option.

A new window will appear. Select **SSH Username with private key** under the section**Kind**. Give any name under section**ID**and**Description**

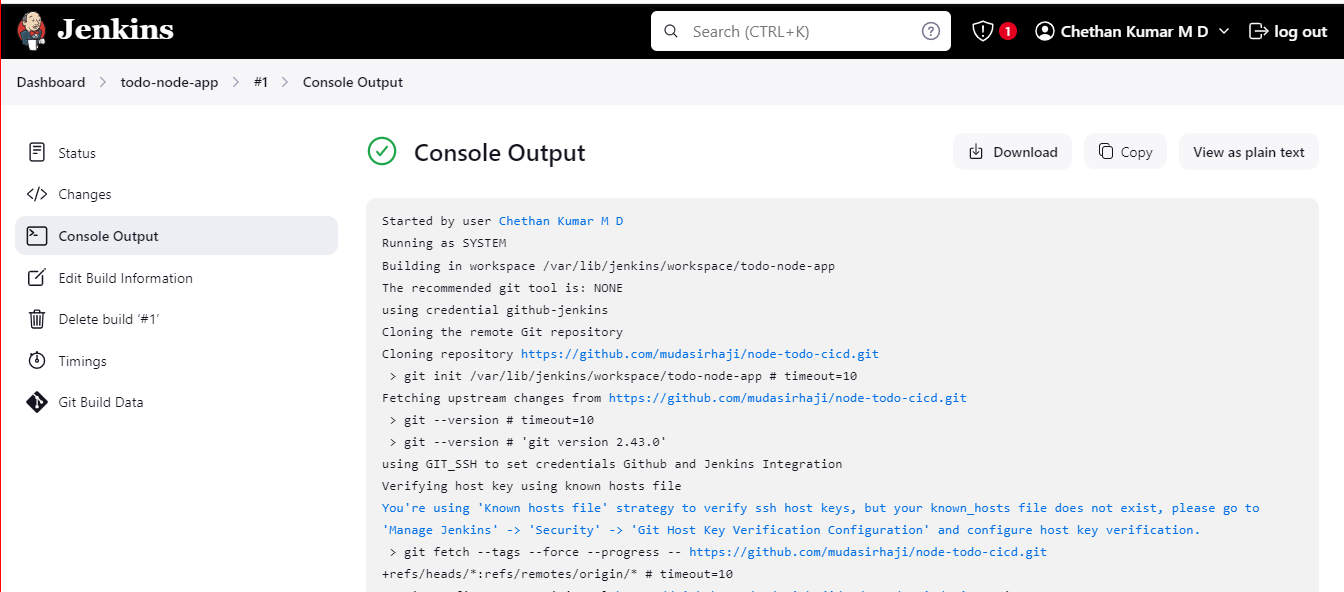
On the other half of this page type the **Username** of the EC2 machine which in our case is **chethan1725** and then paste the private ssh key copied from your EC2 server and click on **Add** button.

Now we have to specify the branch of our GitHub repository. It can be any branch the developer would have been working on. In our case, we will select the master branch

This is all we need so far as the configuration of our first job is concerned. We can save this part of the configuration and move to the next step of building the project

To check whether our GitHub integration with Jenkins is successful we will do a small test by clicking on Build Now on the left navigation bar in our first job

After clicking on Build Now we see in the output console that the build has been finished successfully and Jenkins was able to fetch the code from our GitHub repository. That means Jenkins Integration with GitHub is successful.



**Fig. Jenkins Console output**

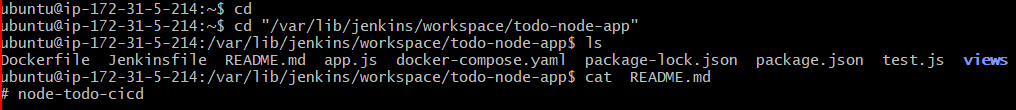


**Fig. Jenkins console output**

We can also verify by navigating to the path mentioned in the output console which is

**/var/lib/jenkins/workspace/todo-node-app**

And check if the repo is cloned there. As shown in the below screenshot we see all the files of the GitHub repo have been successfully cloned.



**Fig. Git Clone output**