CI/CD pipeline design and tool selection

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 (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

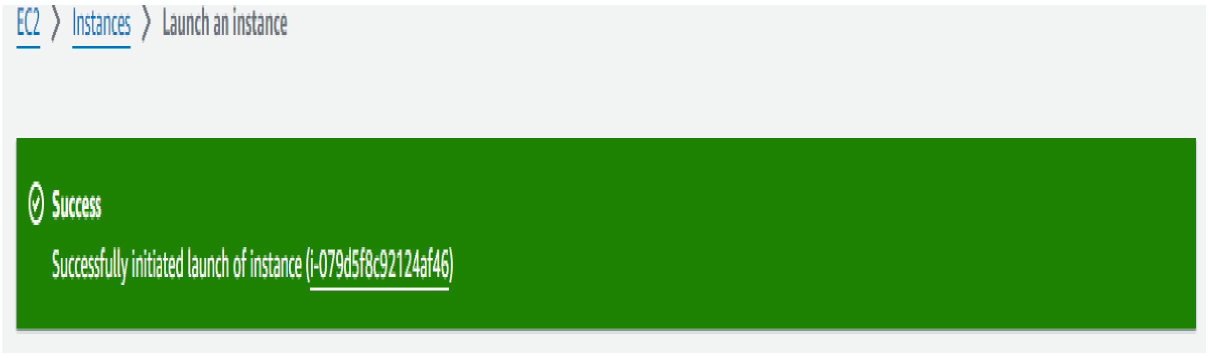
We will first create an AWS Instance (Ubuntu) free-tier eligible using the AWS console.

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

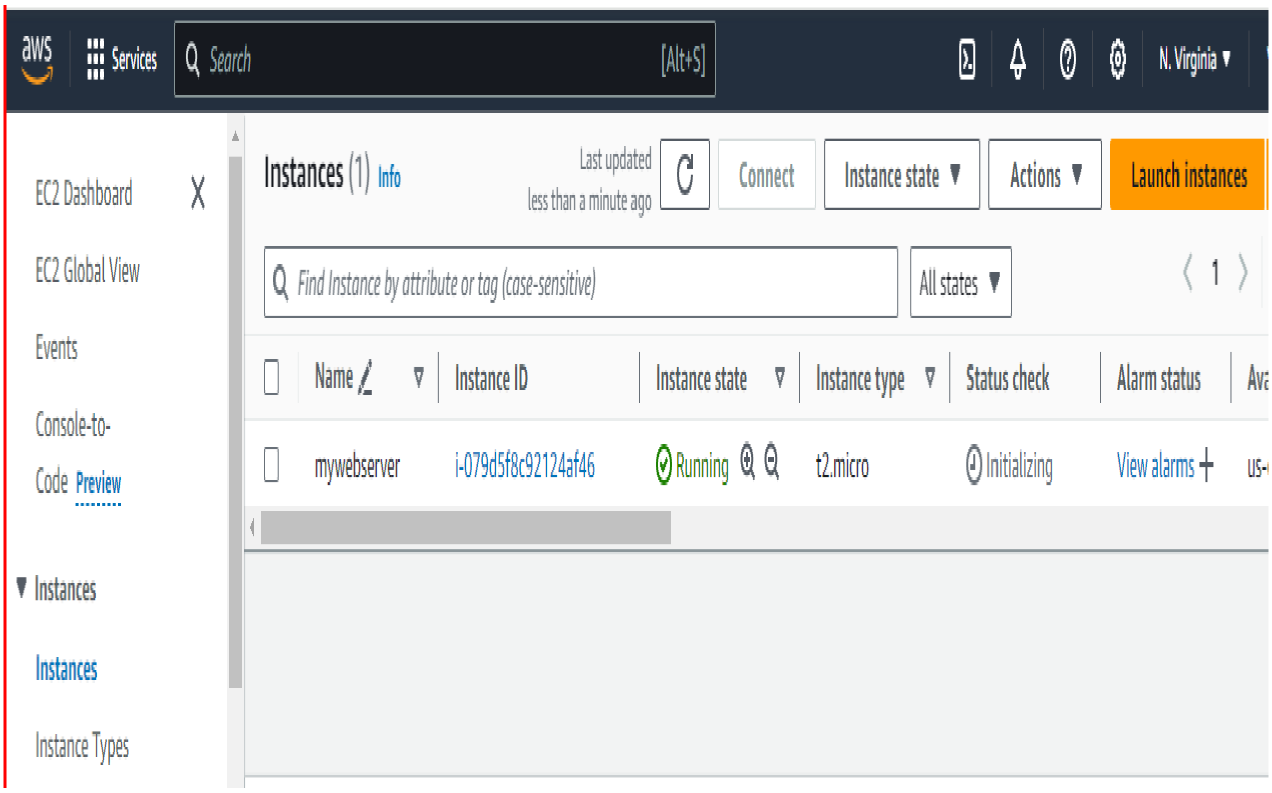
10. Now connect to instance wizard will open, go to SSH client tab and copy the provided chmod and SSH command


EC2 > Instances > Launch an instance

Success

Successfully initiated launch of instance (i-079d5f8c92124af46)

Fig. Launch Instance

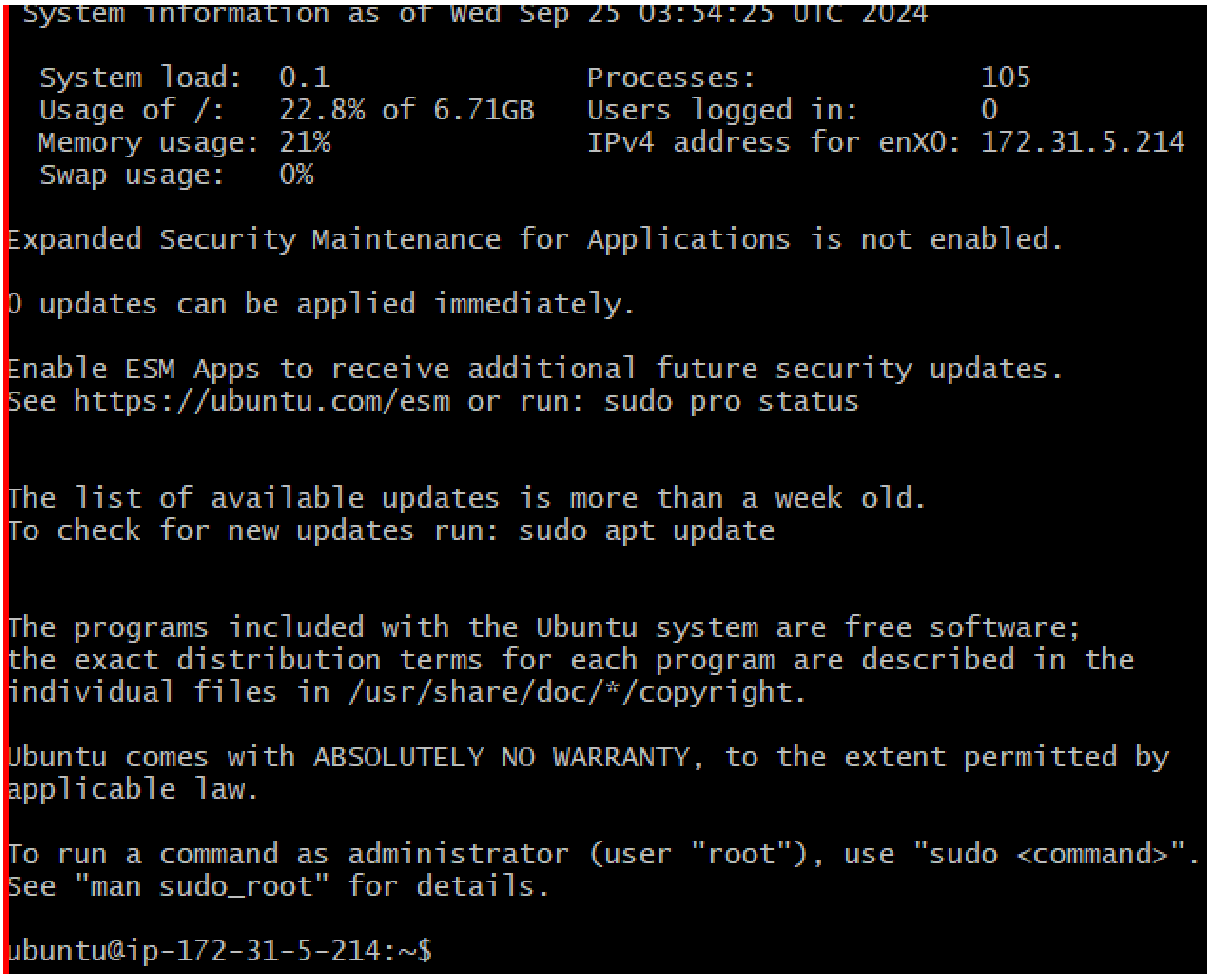


Instances (1)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
	mywebserver	i-079d5f8c92124af46	Running	t2.micro	Initializing	View alarms +	us-

Fig. List Instance

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



```
 System information as of Wed Sep 25 03:54:25 UTC 2024

  System load:  0.1               Processes:             105
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for enX0: 172.31.5.214
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-5-214:~$
```

Fig. Connect to GitBash

Step 2: Install Jenkins on EC2 Instance:

Jenkins installation is straightforward:

1. First Update your Server with the command

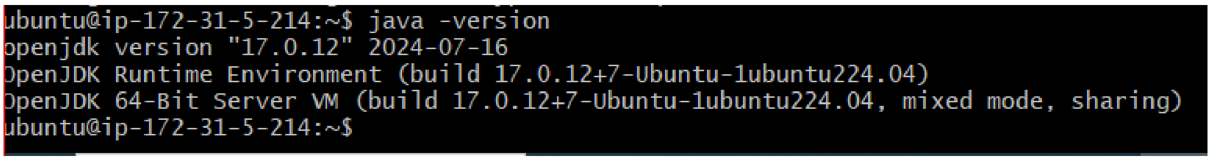
   “sudo apt -get update”

2. Install Java :

   sudo apt install openjdk-17-jre

3. Verify Java Installation:

   java --version

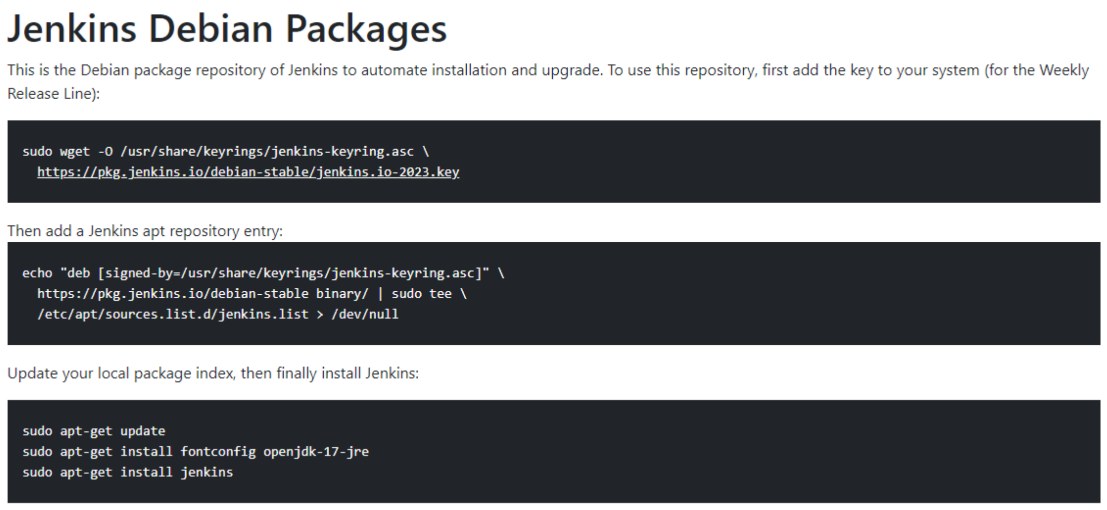


```
ubuntu@ip-172-31-5-214:~$ java -version
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu224.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu224.04, mixed mode, sharing)
ubuntu@ip-172-31-5-214:~$
```

Fig. To Check Java version

4. Install Jenkins:

Just copy these commands and paste them onto your terminal.



Jenkins Debian Packages

This is the Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system (for the Weekly Release Line):

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

Then add a Jenkins apt repository entry:

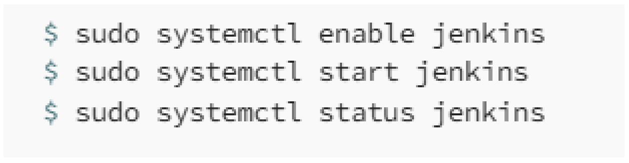
```
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

Update your local package index, then finally install Jenkins:

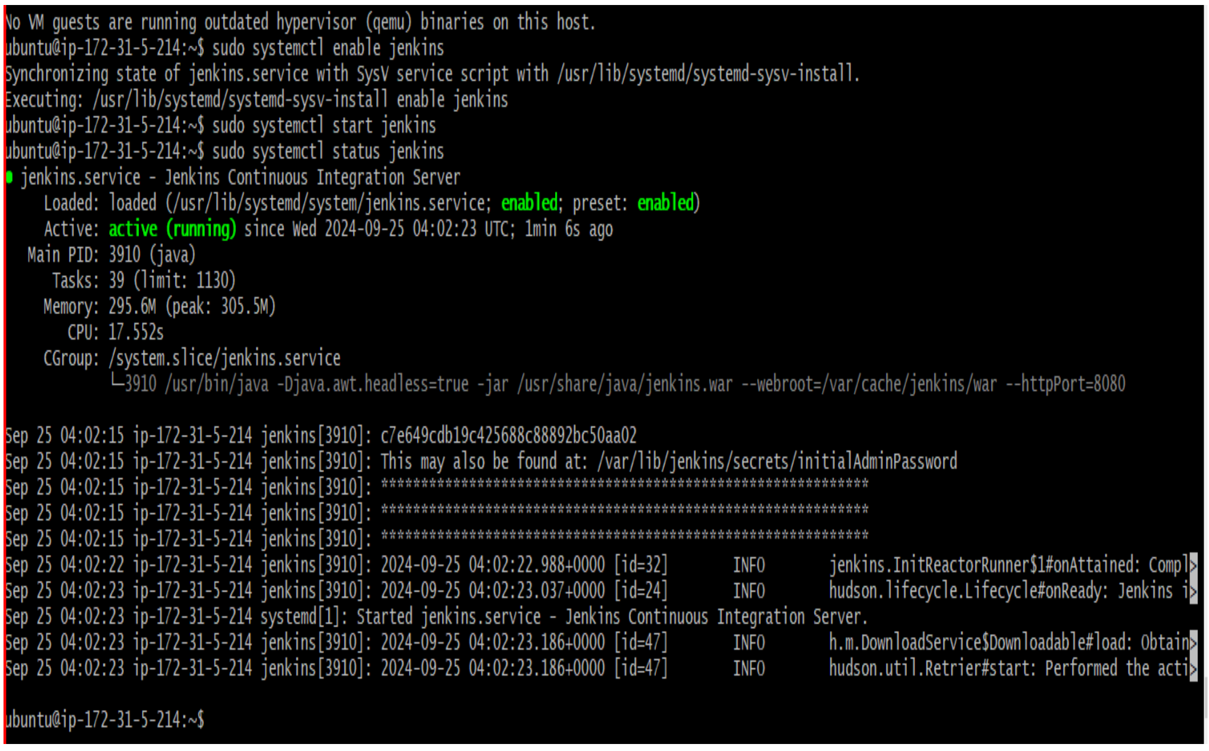
```
sudo apt-get update
sudo apt-get install fontconfig openjdk-17-jre
sudo apt-get install jenkins
```

Fig. Jenkins installation packages

5. Start Jenkins:



```
$ sudo systemctl enable jenkins
$ sudo systemctl start jenkins
$ sudo systemctl status jenkins
```

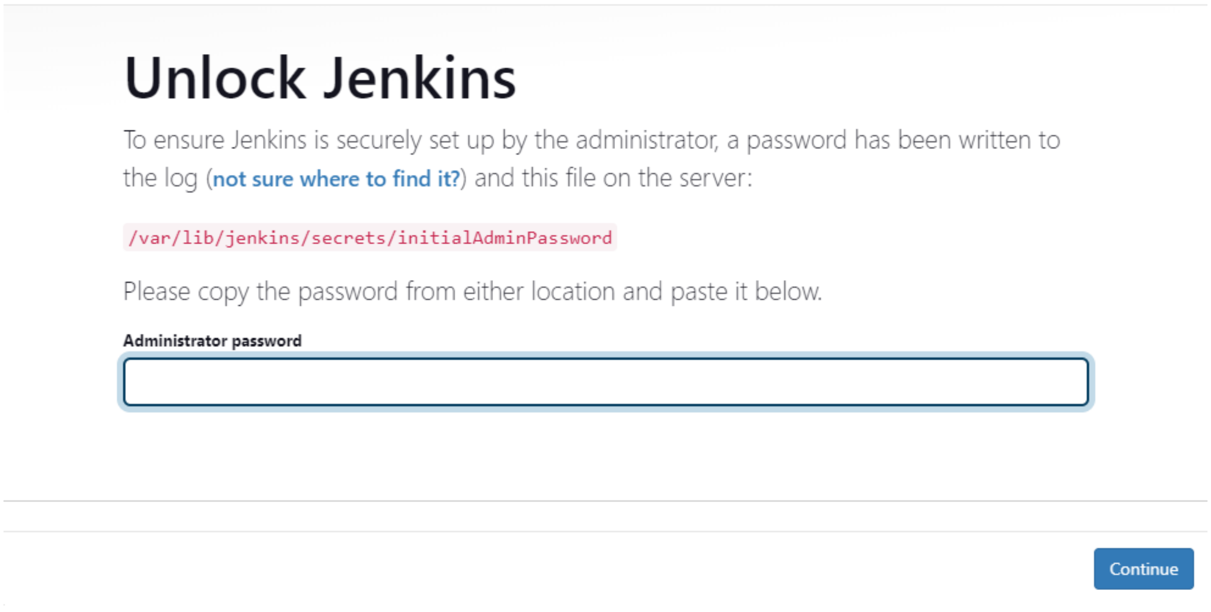


```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-5-214:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-5-214:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-5-214:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-25 04:02:23 UTC; 1min 6s ago
   Main PID: 3910 (java)
      Tasks: 39 (limit: 1130)
     Memory: 295.6M (peak: 305.5M)
        CPU: 17.552s
     CGroup: /system.slice/jenkins.service
             └─3910 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 25 04:02:15 ip-172-31-5-214 jenkins[3910]: c7e649cdb19c425688c88892bc50aa02
Sep 25 04:02:15 ip-172-31-5-214 jenkins[3910]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 25 04:02:15 ip-172-31-5-214 jenkins[3910]: *************************************************************
Sep 25 04:02:15 ip-172-31-5-214 jenkins[3910]: *************************************************************
Sep 25 04:02:15 ip-172-31-5-214 jenkins[3910]: *************************************************************
Sep 25 04:02:22 ip-172-31-5-214 jenkins[3910]: 2024-09-25 04:02:22.988+0000 [id=32]       INFO      jenkins.InitReactorRunner$1#onAttained: Compl>
Sep 25 04:02:23 ip-172-31-5-214 jenkins[3910]: 2024-09-25 04:02:23.037+0000 [id=24]       INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins i>
Sep 25 04:02:23 ip-172-31-5-214 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 25 04:02:23 ip-172-31-5-214 jenkins[3910]: 2024-09-25 04:02:23.186+0000 [id=47]       INFO      h.m.DownloadService$Downloadable#load: Obtain>
Sep 25 04:02:23 ip-172-31-5-214 jenkins[3910]: 2024-09-25 04:02:23.186+0000 [id=47]       INFO      hudson.util.Retrier#start: Performed the acti>

ubuntu@ip-172-31-5-214:~$
```

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.



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

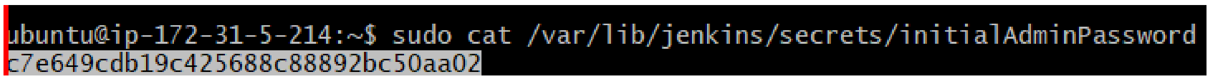
Please copy the password from either location and paste it below.

Administrator password

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.

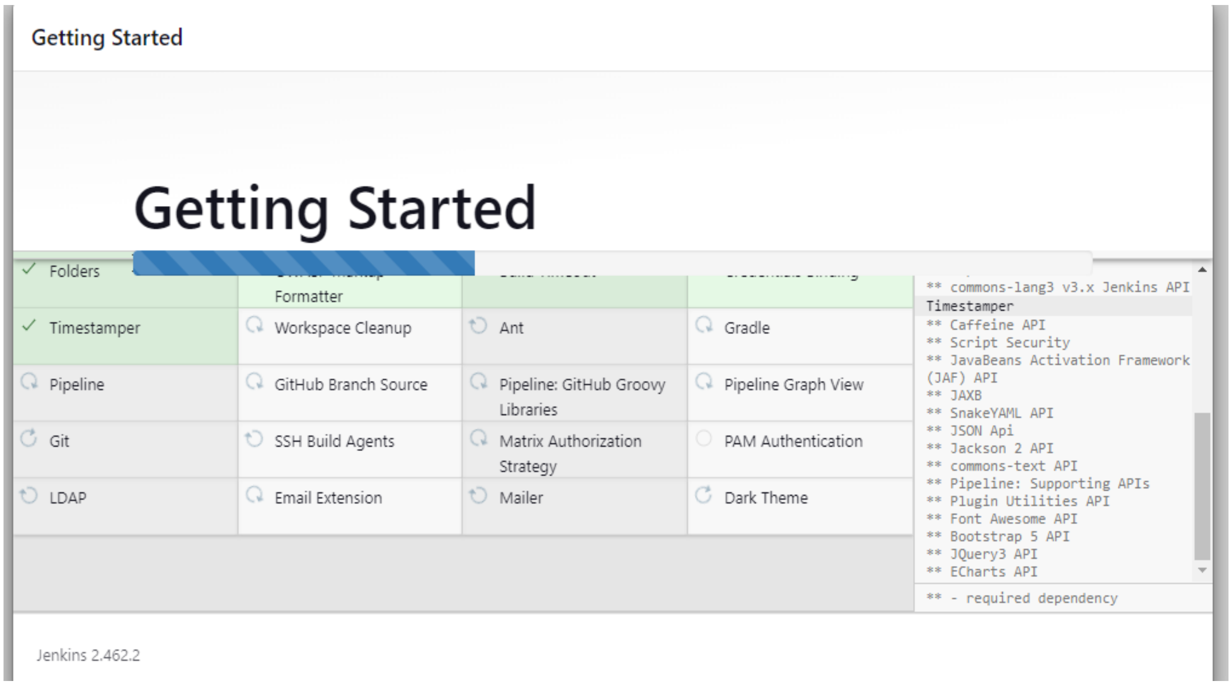
After pasting the password the below page will appear. Click on Install suggested plugins.



```
ubuntu@ip-172-31-5-214:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c7e649cdb19c425688c88892bc50aa02
```

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


Getting Started

Getting Started

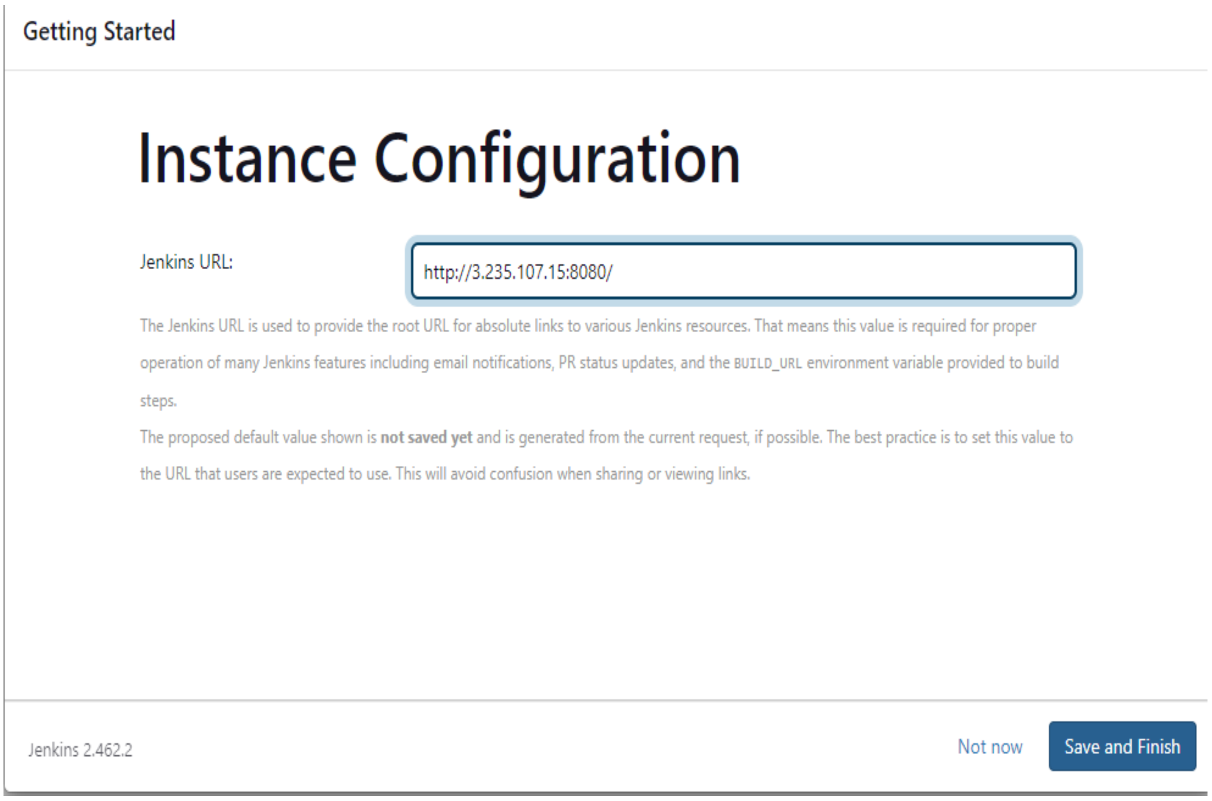
✓ Folders	Formatter		
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline Graph View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	Dark Theme

```
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JavaBeans Activation Framework
(JAF) API
** JAXB
** SnakeYAML API
** JSON Api
** Jackson 2 API
** commons-text API
** Pipeline: Supporting APIs
** Plugin Utilities API
** Font Awesome API
** Bootstrap 5 API
** JQuery3 API
** ECharts API
** - required dependency
```

Jenkins 2.462.2

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


Getting Started

Instance Configuration

Jenkins URL: http://3.235.107.15:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

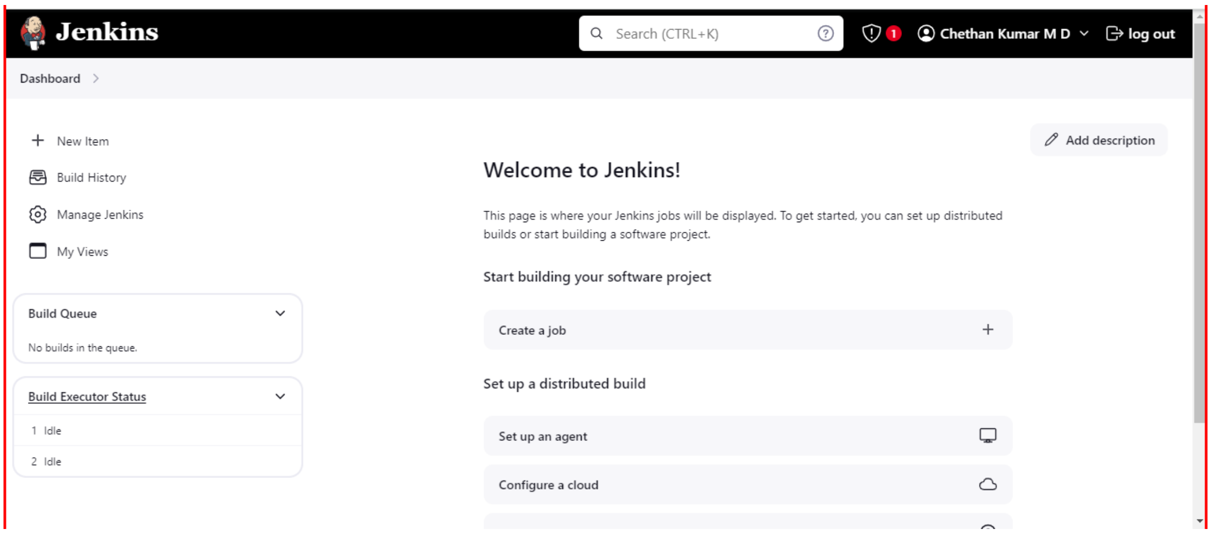
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.462.2

Fig. Instance Configuration

CI/CD pipeline design and tool selection

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



Jenkins | Search (CTRL+K) | Chethan Kumar M D | log out

Dashboard >

- New Item
- Build History
- Manage Jenkins
- My Views

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

Add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

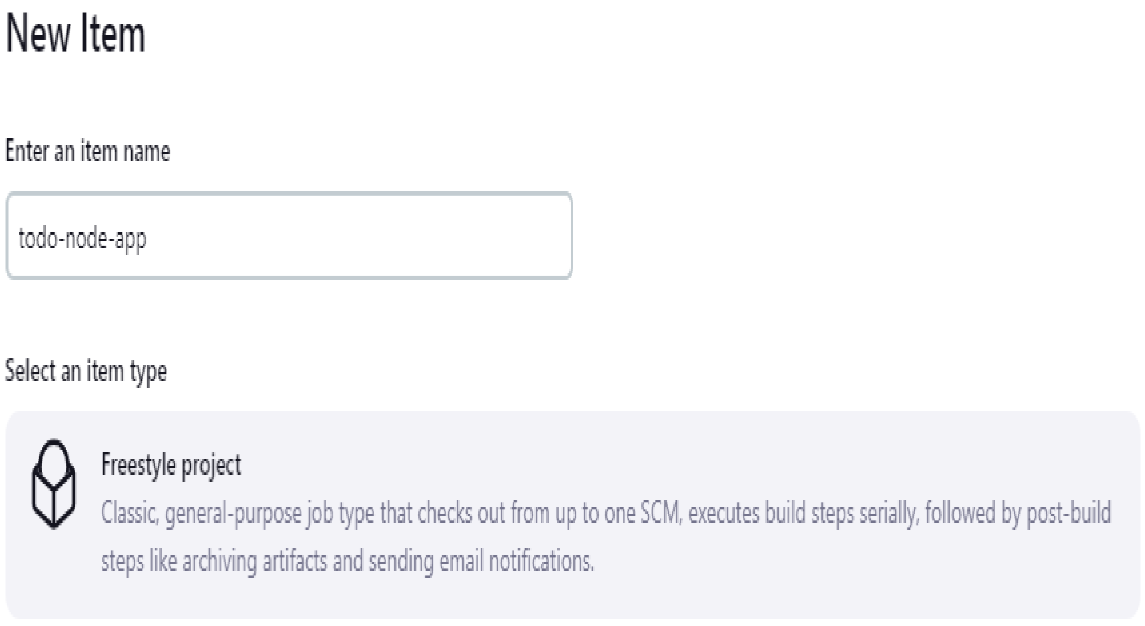
Set up a distributed build

Set up an agent

Configure a cloud

Fig. Jenkins dashboard

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



New Item

Enter an item name

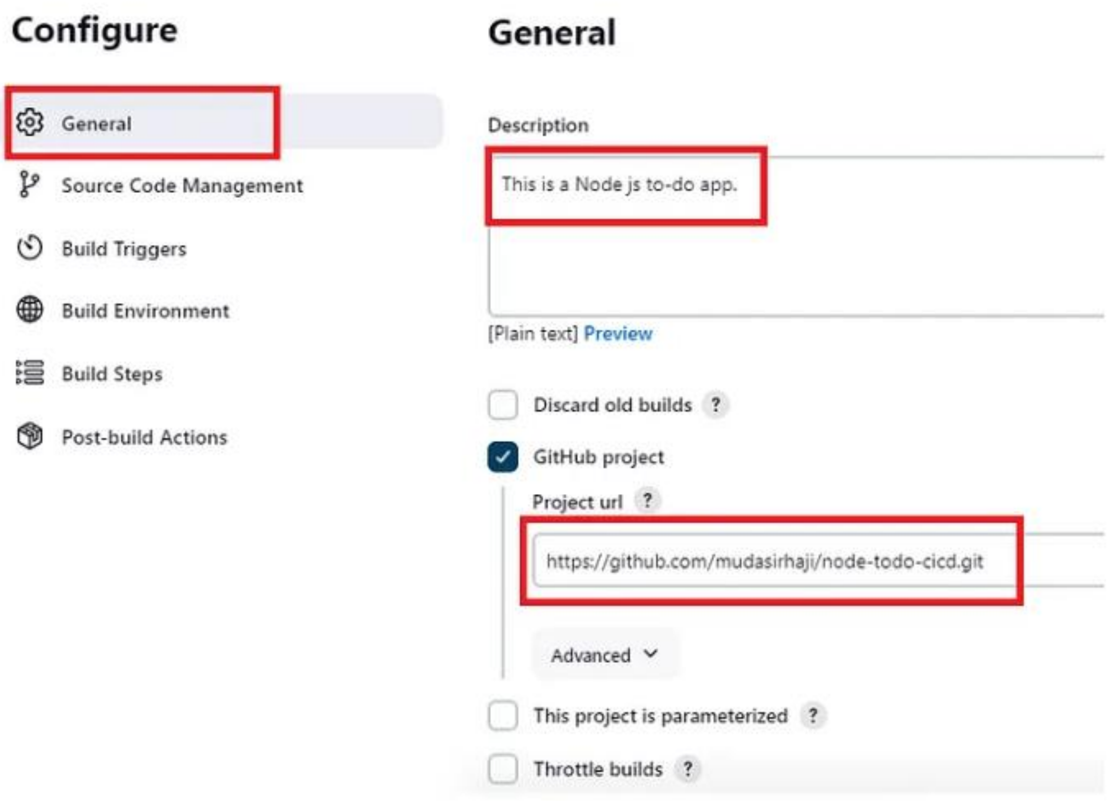
todo-node-app

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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



This is a Node js to-do app.

Discard old builds ?

☑ GitHub project

Project url ?

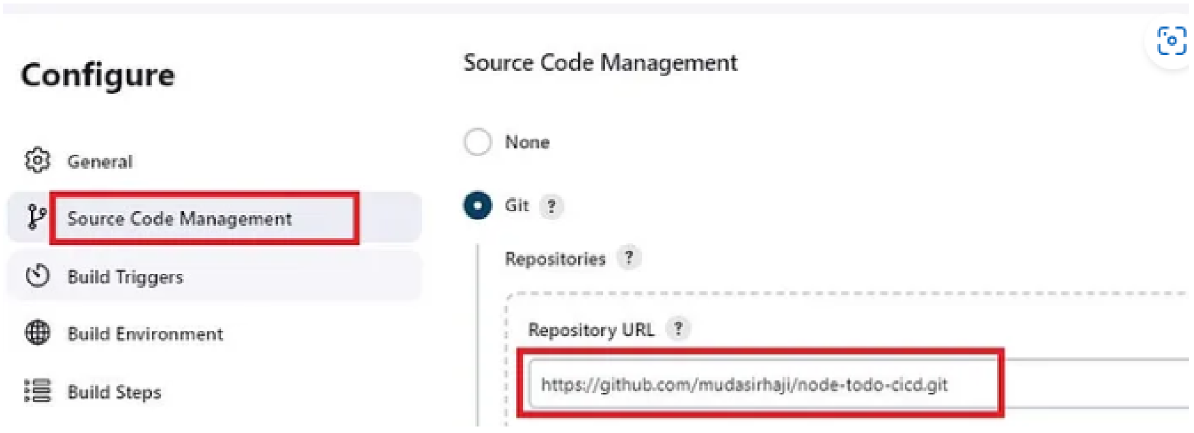
https://github.com/mudasirhaji/node-todo-cicd.git

This project is parameterized ?

Throttle builds ?

Fig. Adding Git Project Url

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



None

● Git ?

Repositories ?

Repository URL ?

https://github.com/mudasirhaji/node-todo-cicd.git

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

CI/CD pipeline design and tool selection

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**

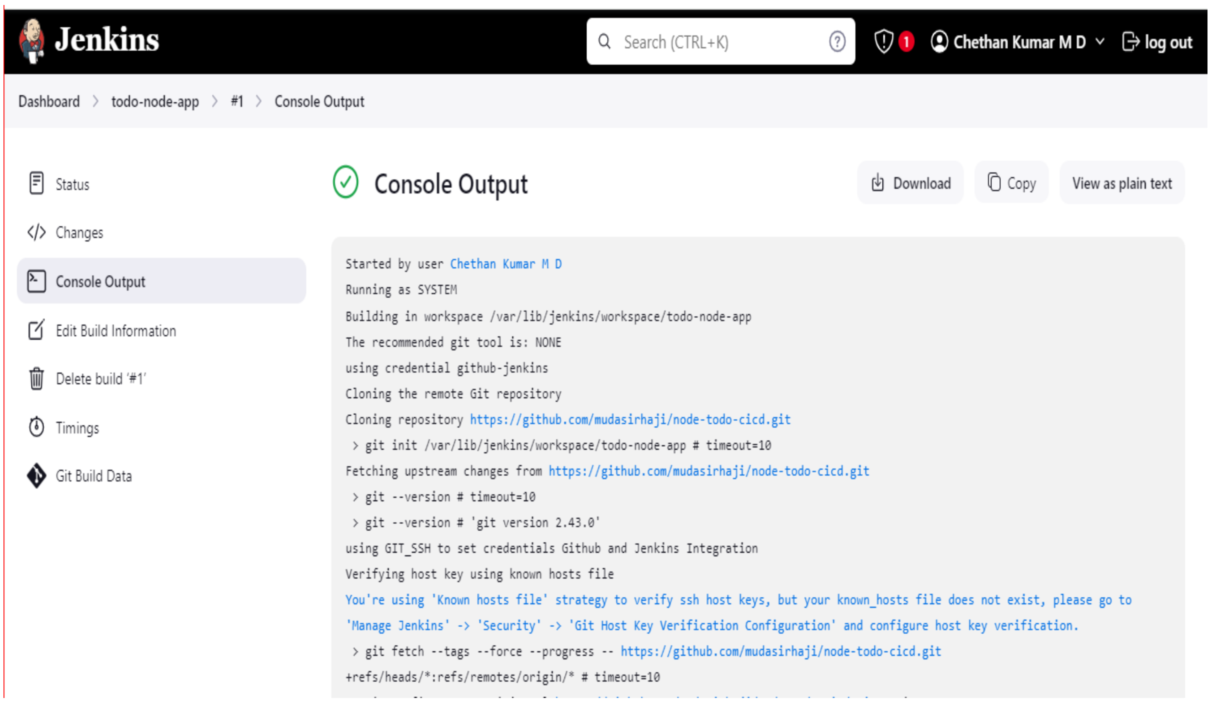
On the other half of this page type the **Username** of the EC2 machine which in our case is **chethan1725** and then paste the private ssh key copied from your EC2 server and click on **Add** button.

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

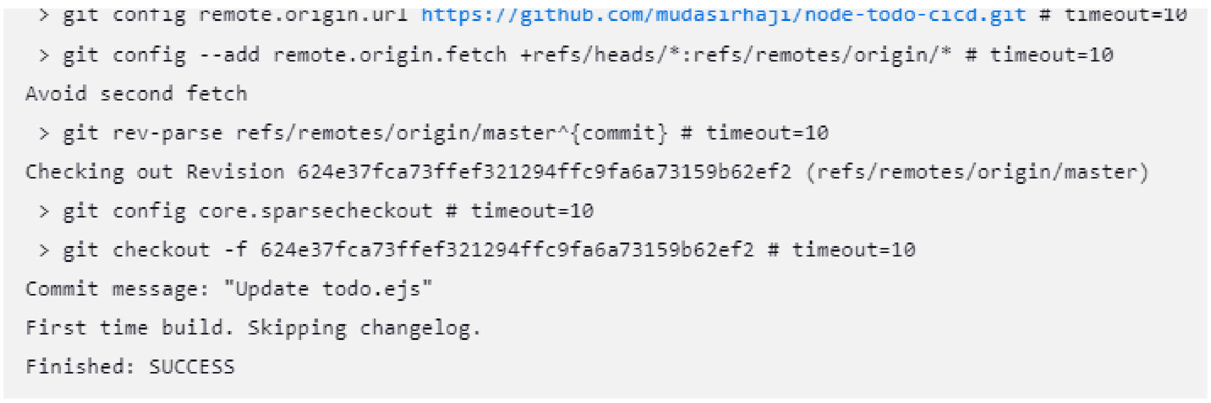
To check whether our GitHub integration with Jenkins is successful we will do a small test by clicking on Build Now on the left navigation bar in our first job

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.



```
Started by user Chethan Kumar M D
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/todo-node-app
The recommended git tool is: NONE
using credential github-jenkins
Cloning the remote Git repository
Cloning repository https://github.com/mudasirhaji/node-todo-cicd.git
 > git init /var/lib/jenkins/workspace/todo-node-app # timeout=10
Fetching upstream changes from https://github.com/mudasirhaji/node-todo-cicd.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
using GIT_SSH to set credentials Github and Jenkins Integration
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to
'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
 > git fetch --tags --force --progress -- https://github.com/mudasirhaji/node-todo-cicd.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
```

Fig. Jenkins Console output



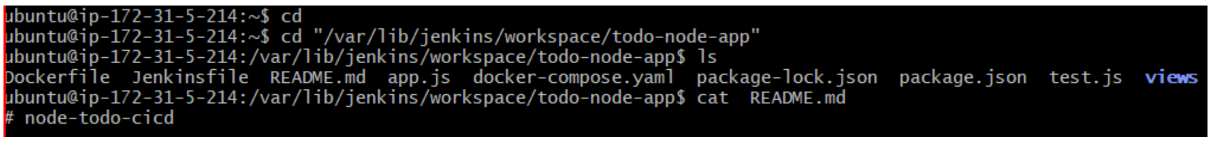
```
 > git config remote.origin.url https://github.com/mudasirhaji/node-todo-cicd.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 624e37fca73ffef321294ffc9fa6a73159b62ef2 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 624e37fca73ffef321294ffc9fa6a73159b62ef2 # timeout=10
Commit message: "Update todo.ejs"
First time build. Skipping changelog.
Finished: SUCCESS
```

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.



```
ubuntu@ip-172-31-5-214:~$ cd
ubuntu@ip-172-31-5-214:~$ cd "/var/lib/jenkins/workspace/todo-node-app"
ubuntu@ip-172-31-5-214:/var/lib/jenkins/workspace/todo-node-app$ ls
Dockerfile  Jenkinsfile  README.md  app.js  docker-compose.yaml  package-lock.json  package.json  test.js  views
ubuntu@ip-172-31-5-214:/var/lib/jenkins/workspace/todo-node-app$ cat  README.md
# node-todo-cicd
```

Fig. Git Clone output