

Guide to Setting Up Git, GitHub, Jenkins with Amazon EC2, and Building a Node.js Application

This comprehensive guide outlines the step-by-step process for setting up and integrating several key technologies crucial for modern software development and deployment. The document will cover the installation, configuration, and use of Git for version control and collaboration, GitHub for remote repository management, and Jenkins for continuous integration and continuous delivery (CI/CD) on an Amazon EC2 instance. Additionally, it will guide you through the process of building and deploying a Node.js application. Each section will provide detailed instructions, from the initial setup of each tool to the final deployment of your application, ensuring that you have a seamless and efficient development workflow. By following this guide, you will be equipped to manage your code repositories, automate your build processes, and deploy applications with ease, leveraging the power and flexibility of these industry-standard tools.

Table of Contents

1. Introduction	2
Purpose of the Document	2
Scope	2
Prerequisites	2
2. Setting Up Git and GitHub.....	2
Installing Git	2
Configuring Git	2
Creating a GitHub Repository	3
Cloning the Repository	5
3. Setting Up Amazon EC2	6
Creating an EC2 Instance	7
Connecting to the EC2 Instance	10
Setting Up Security Groups	11
4. Installing Jenkins	13
Installing Java	13
Installing Jenkins	13
Configuring Jenkins	14
5. Configuring Jenkins for Node.js Application	14
Installing Node.js on EC2	15
Setting Up Jenkins Job for Node.js Application	16
Configuring GitHub Integration.....	17
Building and Deploying the Node.js Application	18
Automate with Webhooks and pm2 Library.....	19
Master – slave concept	21
6. Conclusion	25
Summary of Steps	25
Troubleshooting Tips	25

1. Introduction

Purpose of the Document:

This document aims to provide a comprehensive guide on setting up Git, GitHub, Jenkins with Amazon EC2, and building a Node.js application.

Scope:

The documentation covers installation, configuration, and deployment steps for the mentioned technologies.

Prerequisites:

- Basic knowledge of Git, GitHub, Jenkins, AWS, and Node.js
- AWS account with access to EC2 and SSH key for EC2 access
- GitHub account

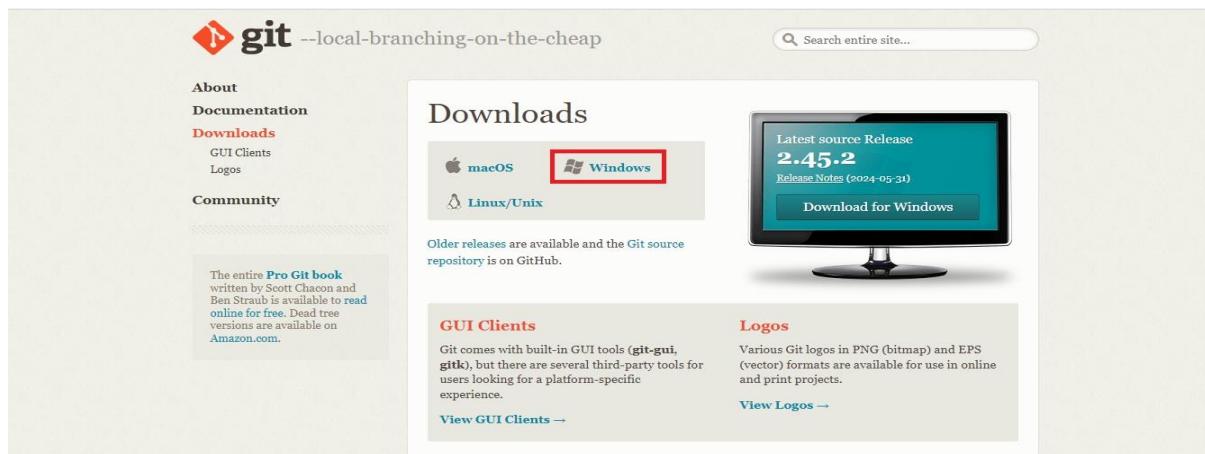
2. Setting Up Git and GitHub

Installing Git:

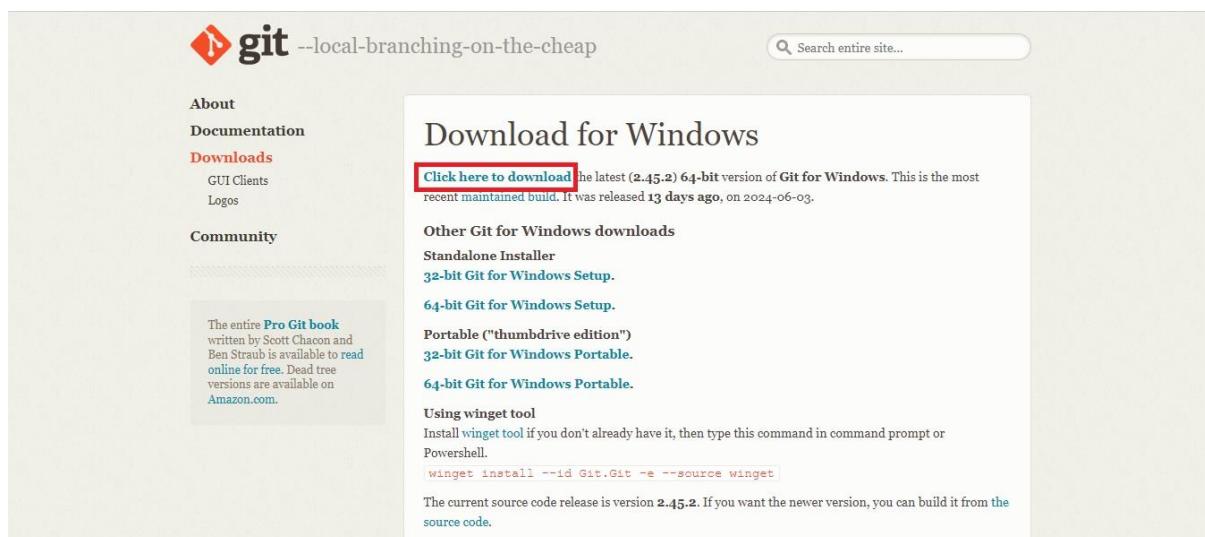
Download Git from [git-scm.com] (<https://git-scm.com/downloads>).

Follow the installation instructions for your operating system.

For windows:



Make sure that download latest version:



Reference video if needed: <https://www.youtube.com/watch?v=8HhEupU4iGU>

Configuring Git:

1. Open a terminal or command prompt.

Check the installation status to see whether it is installed or not.

The “git version” or “git -v” command displays the currently installed version of Git.

```
Command Prompt

Microsoft Windows [Version 10.0.26120.961]
(c) Microsoft Corporation. All rights reserved.

C:\Users\D.sai vamshi krishna>git version
git version 2.45.1.windows.1

C:\Users\D.sai vamshi krishna>git -v
git version 2.45.1.windows.1

C:\Users\D.sai vamshi krishna>
```

2. Set your Git username and email:

In bash/CMD:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

```
Command Prompt

Microsoft Windows [Version 10.0.26120.961]
(c) Microsoft Corporation. All rights reserved.

C:\Users\D.sai vamshi krishna>git version
git version 2.45.1.windows.1

C:\Users\D.sai vamshi krishna>git -v
git version 2.45.1.windows.1

C:\Users\D.sai vamshi krishna>git config --global user.name "Srinivasulu Dammu"
C:\Users\D.sai vamshi krishna>git config --global user.email "vasyvasf@gmail.com"

C:\Users\D.sai vamshi krishna>
```

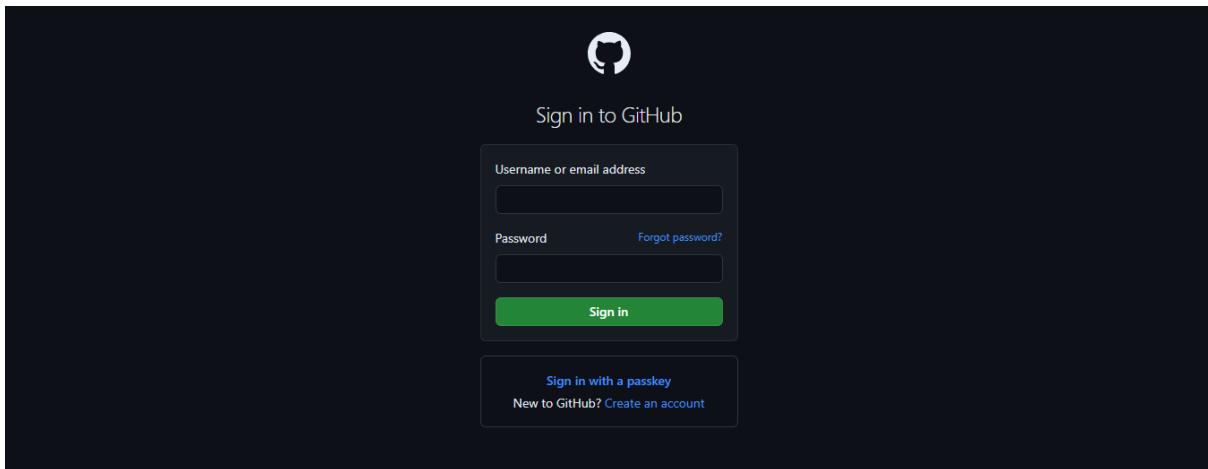
We set up our Git username and email to associate our identity with the commits we make. This information is included in each commit, allowing others to see who made changes to the code and providing accountability and traceability in the version control history.

Creating a GitHub Repository

1. Log in to GitHub.

<https://github.com/>

Note: Create a GitHub account if you don't already have one.



Sign in with our mail id and password.

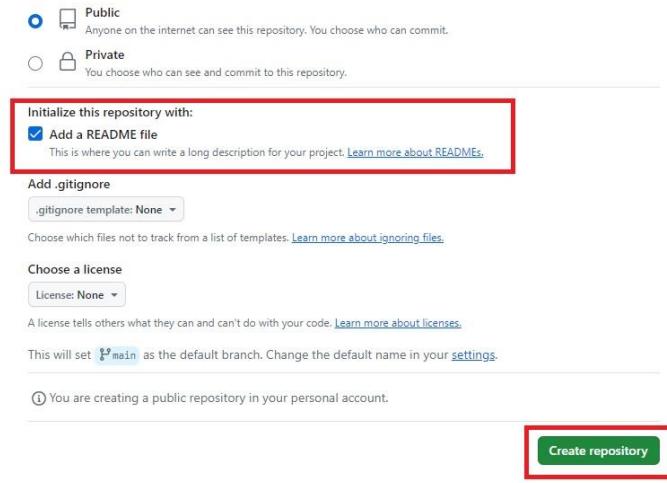
2. Click on "New Repository".

The image shows the GitHub home page. On the left, there's a sidebar with "Top Repositories" and a "New" button highlighted with a red box. The main area shows trending repositories like "togethercomputer/MoA" and "zed-industries/zed". On the right, there's a "Latest changes" section with recent updates from Microsoft and a "Explore repositories" section showing "docker-mailserver / docker-mailserver".

3. Enter a repository name and description.

The image shows the "Create a new repository" page. At the top, it says "Create a new repository" and provides a link to "Import a repository". It states that a repository contains all project files, including revision history. Below this, it says "Required fields are marked with an asterisk (*)." and shows an "Owner" dropdown set to "srinivasulu0514". The "Repository name" field is highlighted with a red box and contains "Project_1", with a note "Project_1 is available." Below the repository name, there's a description field labeled "Description (optional)" and a "Public" radio button selected, with the note "Anyone on the internet can see this repository. You choose who can commit." There's also a "Private" radio button with the note "You choose who can see and commit to this repository."

4. Initialize the repository with a README (optional).



Cloning the Repository

Steps for Forking a Repository on GitHub:

1. Navigate to the Repository:

Go to the repository that you want to fork. You can search for it or navigate directly using the URL (e.g., "https://github.com/username/repository").

URL: <https://github.com/johnpapa/node-hello>

2. Fork the Repository:

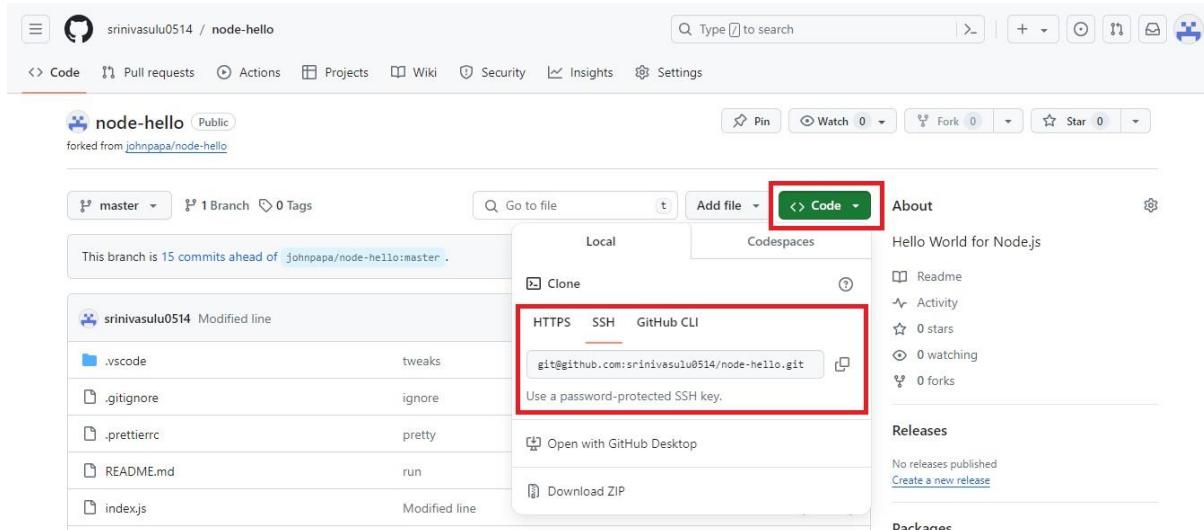
In the upper right corner of the repository page, click the “Fork” button.

The screenshot shows the GitHub repository page for 'node-hello'. The 'Fork' button in the top right is highlighted with a red box. The page displays the repository's structure with files like .vscode, .gitignore, .prettierrc, README.md, index.js, package-lock.json, and package.json. The 'About' section on the right provides details such as the repository name ('Hello World for Node.js'), tags ('nodejs', 'javascript', 'node'), commit history ('14 Commits'), and metrics ('1.7k forks', '59 stars', '5 watching'). The 'Releases' section indicates 'No releases published'.

GitHub will create a copy of the repository under your account. This may take a few moments.

3. Clone Your Forked Repository:

- After forking, navigate to your GitHub profile, and go to the `Repositories` section.
- Click on the newly forked repository.
- Click the “Code” button (usually a green button) to reveal the repository URL.
- Copy the URL (choose either HTTPS or SSH, depending on your preference and setup).



- Open a terminal or command prompt on your local machine.

- Clone the forked repository to your local machine:

```
git clone https://github.com/your-username/forked-repository.git
```

(or)

```
git clone git@github.com:your-username/forked-repository.git
```

Replace “your-username” with your GitHub username and “forked-repository” with the name of the forked repository.

By following these steps, you will have successfully forked a repository, cloned it to your local machine.

Committing and Pushing Changes

1. Make changes to your files and save it.

2. Add changes to the staging area:

```
C:\Users\D.sai vamshi krishna\Downloads> git add .
```

3. Commit the changes:

```
C:\Users\D.sai vamshi krishna\Downloads> git commit -m "Your commit message"
```

4. Push the changes to GitHub:

```
C:\Users\D.sai vamshi krishna\Downloads>git push origin main
```

3. Creating an EC2 Instance

1. Log in to AWS Management Console.
2. Navigate to EC2 Dashboard and click "Launch Instance".

The screenshot shows the AWS EC2 Instances Info page. On the left, there's a sidebar with links like EC2 Dashboard, EC2 Global View, Events, Instances (with sub-links for Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store. The main area has tabs for Instances Info and Launch requests. A search bar at the top says 'Find Instance by attribute or tag (case-sensitive)'. Below it, there are filters for Instance state (set to 'running') and Instance type. A large button labeled 'Launch instances' is highlighted with a red box. The message 'No matching instances found' is displayed below the filters.

3. Choose an Amazon Machine Image (AMI) (e.g., Amazon Linux 2023 AMI).

The screenshot shows the AWS Quick Start page. On the left, there's a 'Quick Start' section with icons for Amazon Linux (highlighted with a red box), macOS, Ubuntu, Windows, Red Hat, and SUSE. Below this is a 'Amazon Machine Image (AMI)' section for 'Amazon Linux 2023 AMI'. It shows the AMI ID 'ami-0e1d06225679bc1c5', architecture '64-bit (x86)', boot mode 'uefi-preferred', and AMI ID 'ami-0e1d06225679bc1c5'. A 'Verified provider' badge is present. To the right is a 'Summary' section with fields for 'Number of instances' (set to 2), 'Software Image (AMI)' (Amazon Linux 2023 AMI 2023.4.20240611.0 x86_64 HVM kernel-6.1), 'Virtual server type (instance type)' (t2.micro), 'Firewall (security group)' (New security group), and 'Storage (volumes)'. A 'Launch instance' button is highlighted with a red box. At the bottom, there are links for CloudShell, Feedback, and Copyright information.

4. Select an instance type (e.g., t2.micro).

The screenshot shows the AWS Launch Instances page. On the left, there's a 'Description' section for 'Amazon Linux 2023 AMI 2023.4.20240611.0 x86_64 HVM kernel-6.1' with details about architecture (64-bit (x86)), boot mode (uefi-preferred), and AMI ID (ami-0e1d06225679bc1c5). Below this is an 'Instance type' section with a dropdown menu. The 't2.micro' option is highlighted with a red box. It shows details: Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true, On-Demand Linux base pricing: 0.0124 USD per Hour, On-Demand Windows base pricing: 0.017 USD per Hour, On-Demand RHEL base pricing: 0.0724 USD per Hour, and On-Demand SUSE base pricing: 0.0124 USD per Hour. A note at the bottom says 'Additional costs apply for AMIs with pre-installed software'. To the right is a 'Summary' section with fields for 'Number of instances' (set to 2), 'Software Image (AMI)' (Amazon Linux 2023 AMI 2023.4.20240611.0 x86_64 HVM kernel-6.1), 'Virtual server type (instance type)' (t2.micro), 'Firewall (security group)' (New security group), and 'Storage (volumes)'. A 'Launch instance' button is highlighted with a red box. At the bottom, there are links for CloudShell, Feedback, and Copyright information.

5. Add a key pair.

We need a key pair in Amazon EC2 to securely connect to our instances via SSH. The key pair ensures that only authorized users can access the instance.

The screenshot shows the AWS EC2 instance creation wizard. In the 'Key pair (login)' section, there is a 'Key pair name - required' field with the placeholder 'Proceed without a key pair (Not recommended)'. A 'Create new key pair' button is highlighted with a red box. The right sidebar displays information about key pairs and a link to 'Amazon EC2 key pairs'.

6. Configure security groups (allow SSH, HTTP/HTTPS).

The screenshot shows the AWS EC2 instance creation wizard. In the 'Network settings' section, under 'Firewall (security group)', the 'Create security group' option is selected. Below it, a list of traffic rules is shown, with the first rule 'Allow SSH traffic from Anywhere (0.0.0.0/0)' highlighted with a red box. The right sidebar displays information about security groups and a link to 'Amazon EC2 key pairs'.

If we already created a security groups, choose one of the groups accordingly.

7. Configure instance details and add storage.

The screenshot shows the AWS EC2 instance creation wizard. In the 'Configure storage' section, a root volume of '1x 8 GiB gp3' is selected, which is highlighted with a red box. A note below states 'Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage'. The right sidebar displays information about storage and a link to 'Amazon EC2 key pairs'.

8. Review and launch the instance, after, download the SSH key pair.

Here, we are creating two instances “Jenkins server (Master)” and “application server”.

The screenshot shows the AWS EC2 'Launch Instance' wizard. In the 'Summary' step, the 'Number of instances' field is set to 2. The 'Launch instance' button is highlighted with a red box. The 'Key pair' sidebar indicates that no key pair is selected. The 'Learn more' section links to 'Amazon EC2 key pairs'.

The screenshot shows the 'Success' message: 'Successfully initiated launch of Instances (i-0800143abf09bb2f, i-0b7378d78fa1789b2)'. The 'Launch log' link is visible below the message.

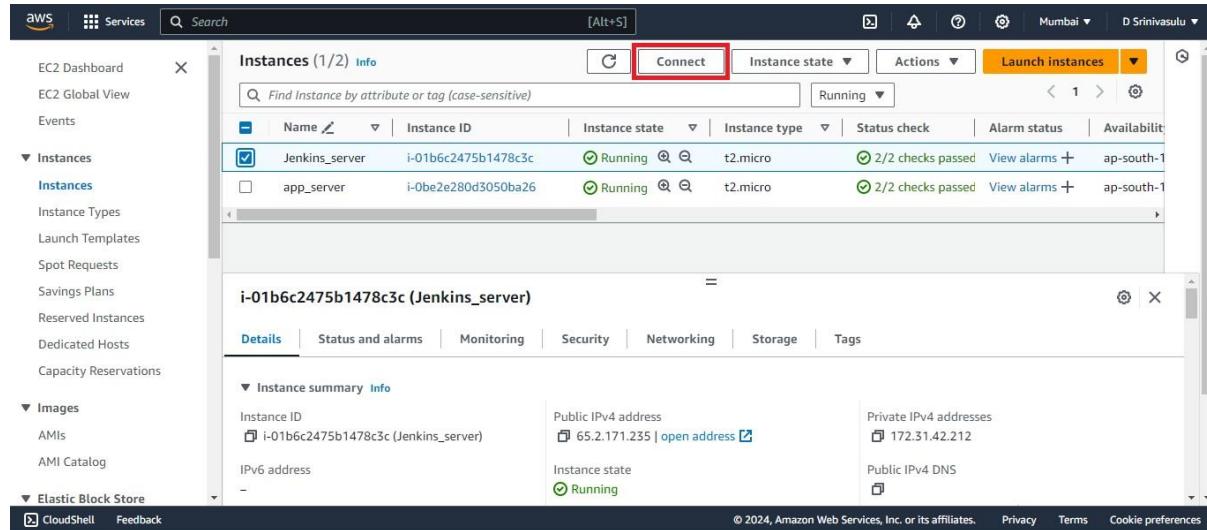
Open the instances and name the one of the instances as “Jenkins_server” and another instance as “app_server”.

The screenshot shows the AWS EC2 Instances page. Two instances are listed: 'Jenkins_server' (Instance ID: i-0800143abf09bb2f, Status: Running, Type: t2.micro) and 'app_server' (Instance ID: i-0b7378d78fa1789b2, Status: Running, Type: t2.micro). The 'Instances' sidebar is open, and the 'Actions' dropdown menu is visible.

Connecting to the EC2 Instance:

1. Open a terminal.

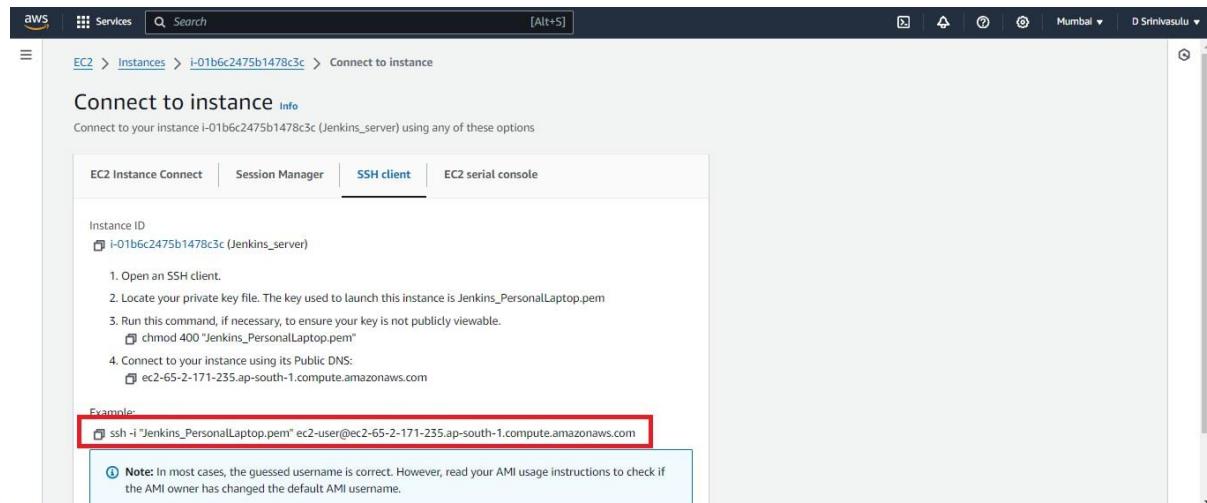
2. Go to the directory, where the private key pair file is stored and connect the Jenkins_server instance.



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. Below the sidebar are CloudShell and Feedback links. The main pane displays 'Instances (1/2) Info'. It lists two instances: 'Jenkins_server' (i-01b6c2475b1478c3c) and 'app_server' (i-0be2e280d3050ba26). Both are running, t2.micro instances. A red box highlights the 'Connect' button in the top right of the main pane. Below the instances, a detailed view for 'i-01b6c2475b1478c3c (Jenkins_server)' is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is selected. In the 'Instance summary' section, it shows Instance ID (i-01b6c2475b1478c3c), Public IPv4 address (65.2.171.235), Private IPv4 addresses (172.31.42.212), Instance state (Running), and Public IPv4 DNS (ec2-65-2-171-235.ap-south-1.compute.amazonaws.com).

3. Connect to the instance:

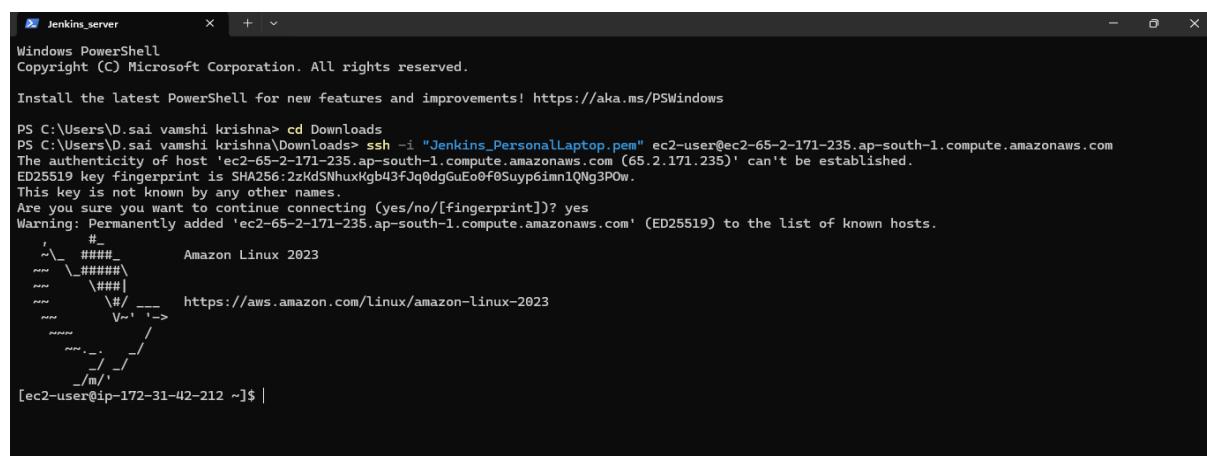
Connect the instance in the power shell/command prompt with SSH client, follow the instructions in the picture below.



The screenshot shows the 'Connect to instance' page for the Jenkins_server instance. At the top, it says 'EC2 > Instances > i-01b6c2475b1478c3c > Connect to instance'. Below that, it says 'Connect to instance' and provides instructions: 'Connect to your instance i-01b6c2475b1478c3c (Jenkins_server) using any of these options'. There are four tabs: EC2 Instance Connect, Session Manager, SSH client (selected), and EC2 serial console. Under 'SSH client', it shows the instance ID (i-01b6c2475b1478c3c) and a list of steps to open an SSH client. An example command is provided: 'ssh -i "Jenkins_PersonalLaptop.pem" ec2-user@ec2-65-2-171-235.ap-south-1.compute.amazonaws.com'. A red box highlights this command. A note below says: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' A red box highlights this note.

Follow these same steps in power shell for the “app server” too

Jenkins server:



```
Jenkins.server
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\D.sai vamsi krishna> cd Downloads
PS C:\Users\D.sai vamsi krishna\Downloads> ssh -i "Jenkins_PersonalLaptop.pem" ec2-user@ec2-65-2-171-235.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-2-171-235.ap-south-1.compute.amazonaws.com (65.2.171.235)' can't be established.
ED25519 key fingerprint is SHA256:2zKdSNhuxkgb43fJq0dgGuEo0f0Suyp6imn1QNg3POw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-2-171-235.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.

      _#_
     /###\
    /##\ 
   /# \
  /  \_> https://aws.amazon.com/linux/amazon-linux-2023
 /_/
/_/m/ [ec2-user@ip-172-31-42-212 ~]$ |
```

App server:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\D.sai vanshi krishna> cd Downloads
PS C:\Users\D.sai vanshi krishna\Downloads> ssh -i "Jenkins_PersonalLaptop.pem" ec2-user@ec2-65-0-104-199.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-0-104-199.ap-south-1.compute.amazonaws.com (65.0.104.199)' can't be established.
ED25519 key fingerprint is SHA256:lsotUtSlyiwStkii8NtBiFyy3jcxW3au0B7uSwlyNx8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-0-104-199.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.

# 
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-43-241 ~]$ |
```

Setting Up Security Groups

1. In the EC2 dashboard, select "Security Groups".

2. Edit the inbound rules to allow necessary ports (e.g., 8080 for Jenkins, 3000 for Node.js).

Navigate to the security group section within the Jenkins_server Instance and proceed to select the security group.

For Jenkins server:

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under 'Instances', 'Instances' is selected. The main pane displays two instances: 'Jenkins_server' (running, t2.micro, 2/2 checks passed, ap-south-1a, ec2-65-0-104-199.ap-south-1.compute.amazonaws.com) and 'app_server' (running, t2.micro, 2/2 checks passed, ap-south-1a, ec2-65-0-104-199.ap-south-1.compute.amazonaws.com). The 'Security' tab is selected for the Jenkins_server instance. In the 'Security groups' dropdown, the option 'sg-0052a58971519a0a3 (launch-wizard-17)' is highlighted with a red box.

Click on the "Edit Inbound rules" and Add 8080 ports to Anywhere-IP4 and IP6.

The screenshot shows the AWS Security Groups page for the 'launch-wizard-17' security group. The 'Inbound rules' tab is selected. At the top right of the rule list, there is a button labeled 'Edit inbound rules' with a red box around it. The table below lists three inbound rules:

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0836643524fbf2f29	IPv4	HTTP	TCP	80
-	sgr-06b6460ca107153...	IPv4	HTTPS	TCP	443
-	sgr-02866cd4ac13f609a	IPv4	SSH	TCP	22

Security group rule ID Type Info Protocol Info Port range Info Source Info Description - optional info

sgr-0836643524fbf2f29	HTTP	TCP	80	Custom	<input type="text" value="0.0.0.0/0"/>	<input type="text" value=""/>	Delete
sgr-06b6460ca071531a	HTTPS	TCP	443	Custom	<input type="text" value="0.0.0.0/0"/>	<input type="text" value=""/>	Delete
sgr-02866cd4ac13f609a	SSH	TCP	22	Custom	<input type="text" value="0.0.0.0/0"/>	<input type="text" value=""/>	Delete
-	Custom TCP	TCP	8080	Anywhere...	<input type="text" value="0.0.0.0/0"/>	<input type="text" value=""/>	Delete
-	Custom TCP	TCP	8080	Anywhere...	<input type="text" value="0.0.0.0/0"/>	<input type="text" value=""/>	Delete

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

For App server:

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv6 DNS
Jenkins_server	i-01b6c2475b1478c3c	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-65-2-171-235.ap-s...	65.2.1
app_server	i-0be2e280d3050ba26	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-65-0-104-199.ap-s...	65.0.1

i-0be2e280d3050ba26 (app_server)

Details Status and alarms Monitoring Security Networking Storage Tags

Security details

IAM Role: Owner ID: 471112779411 Launch time: Sun Jun 16 2024 20:08:52 GMT+0530 (India Standard Time)

Security groups: sg-0052a58971519a0a3 (launch-wizard-17)

Inbound rules

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

4. Installing Jenkins in the Jenkins server:

Installing Java

1. Update the package index:

```
[ec2-user ~]$ sudo yum update -y
```

2. Add the Jenkins repository and import the key:

```
[ec2-user ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

3. Import a key file from Jenkins-CI to enable installation from the package:

```
[ec2-user ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

```
[ec2-user ~]$ sudo yum upgrade
```

4. Install Java (Amazon Linux 2023):

```
[ec2-user ~]$ sudo dnf install java-17-amazon-corretto -y
```

```
Jenkins_server          app_server          + ~
#_###_ Amazon Linux 2023
\###_ \###_ https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-42-212 ~]$ [ec2-user@ip-172-31-42-212 ~]$ sudo yum update -y
Last metadata expiration check: 1:29:14 ago on Sun Jun 16 14:39:38 2024.
No match for argument: -y
Error: No packages marked for upgrade.
[ec2-user@ip-172-31-42-212 ~]$ [ec2-user@ip-172-31-42-212 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2024-06-16 16:09:53-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.154.133, 2a04:4e42:24::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.154.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.repo    100%[=====]     85  --.-KB/s   in 0s

2024-06-16 16:09:54 (3.08 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-42-212 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[ec2-user@ip-172-31-42-212 ~]$ sudo yum upgrade
Jenkins-stable
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-42-212 ~]$ [ec2-user ~]$ sudo dnf install java-17-amazon-corretto -y
```

Installing Jenkins

1. Install Jenkins:

```
[ec2-user ~]$ sudo yum install jenkins -y
```

Configuring Jenkins

1. Start Jenkins:

```
[ec2-user ~]$ sudo systemctl start jenkins
```

2. Enable Jenkins to start on boot:

```
[ec2-user ~]$ sudo systemctl enable Jenkins
```

3. Check the status of the Jenkins service

```
[ec2-user ~]$ sudo systemctl status Jenkins
```

```

Jenkins_server app_server
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing : 1/1
  Running scriptlet: jenkins-2.452.2-1.1.noarch 1/1
  Installing : jenkins-2.452.2-1.1.noarch 1/1
  Running scriptlet: jenkins-2.452.2-1.1.noarch 1/1
  Verifying   : jenkins-2.452.2-1.1.noarch 1/1

Installed:
  jenkins-2.452.2-1.1.noarch

Complete!
[ec2-user@ip-172-31-42-212 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-42-212 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-42-212 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
    Active: active (running) since Sun 2024-06-16 16:45:25 UTC; 1min 15s ago
      Main PID: 28792 (java)
        Tasks: 39 (limit: 1114)
       Memory: 341.5M
          CPU: 37.955s
         CGroup: /system.slice/jenkins.service
             └─28792 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jun 16 16:44:55 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 16 16:44:55 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: ****
Jun 16 16:44:55 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: ****
Jun 16 16:44:55 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: ****
Jun 16 16:45:25 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: 2024-06-16 16:45:25.133+0000 [id=31] INFO jenkins.InitRe...
Jun 16 16:45:25 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: 2024-06-16 16:45:25.155+0000 [id=24] INFO hudson.lifecycle...
Jun 16 16:45:25 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: Started jenkins.service - Jenkins Continuous Integration Server.
Jun 16 16:45:25 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: 2024-06-16 16:45:25.276+0000 [id=47] INFO h.m.DownloadSe...
Jun 16 16:45:25 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: 2024-06-16 16:45:25.276+0000 [id=47] INFO hudson.util.Re...
Jun 16 16:45:30 ip-172-31-42-212.ap-south-1.compute.internal jenkins[28792]: 2024-06-16 16:45:30.268+0000 [id=59] WARNING h.n.DiskSpa...
[lines 1-20/20 (END)]

```

Install git and NodeJS in Jenkins server to build the application:

```

Jenkins_server app_server
#_ _###_
~\_\_ #####_ Amazon Linux 2023
~\_\_ \####\_
~\_\_ \###]
~\_\_ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~\_\_ \#/\_>
~\_\_ \_/_/
~\_\_ \_/_/
~\_\_ \_/_/
Last login: Mon Jun 17 05:14:15 2024 from 136.226.233.78
[ec2-user@ip-172-31-45-40 ~]$ sudo yum install git -y
Last metadata expiration check: 1:22:49 ago on Mon Jun 17 04:14:04 2024.
Package git-2.40.1-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-45-40 ~]$ sudo yum install nodejs -y
Last metadata expiration check: 1:27:29 ago on Mon Jun 17 04:14:04 2024.
Dependencies resolved.
=====
Package           Architecture Version      Repository  Size
=====
Installing:
  nodejs           x86_64      1:18.18.2-1.amzn2023.0.4      amazonlinux 1.8 M
Installing dependencies:
  nodejs-libs      x86_64      1:18.18.2-1.amzn2023.0.4      amazonlinux 14 M
Installing weak dependencies:
  nodejs-docs      noarch     1:18.18.2-1.amzn2023.0.4      amazonlinux 7.6 M
  nodejs-full-i18n x86_64      1:18.18.2-1.amzn2023.0.4      amazonlinux 8.5 M
  nodejs-npm       x86_64      1:9.8.1-1.18.18.2.1.amzn2023.0.4      amazonlinux 2.0 M
=====
Transaction Summary

```

3. Open Jenkins in your browser:

<http://your-ec2-ip:8080>

4. Follow the setup wizard instructions.

Unlock the Jenkins (sudo cat /var/lib/jenkins/secrets/initialAdminPassword) → Create account → Use the Jenkins

5. Install git in Jenkins server (sudo yum install git -y) → to check the git version in Jenkins_server (git -v (or) git -version)

5. Installing Node.js on app server EC2:

1. Open the app server which is already created in the above steps.

2. Install Node.js:

Configure yum repository:

```
sudo yum install -y gcc-c++ make → curl -sL https://rpm.nodesource.com/setup_20.x | sudo -E bash -
```

Installing Node.js on app server

```
sudo yum install -y nodejs
```

The screenshot shows two terminal windows side-by-side. The left window is titled 'Jenkins_server' and the right is 'app_server'. Both are running Windows PowerShell.

Terminal 1 (app_server):

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10741834> cd Downloads
PS C:\Users\10741834\Downloads> ssh -i "Jenkins_login.pem" ec2-user@ec2-13-235-100-114.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-235-100-114.ap-south-1.compute.amazonaws.com (13.235.100.114)' can't be established.
ED25519 key fingerprint is SHA256:6wQjhFVJTQkRifFVRviQdqG29/39tXl8NZBY5dugH9vE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-235-100-114.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.

[...]
[ec2-user@ip-172-31-39-216 ~]$ sudo yum install -y gcc-c++ make
Last metadata expiration check: 0:25:12 ago on Mon Jun 17 04:10:39 2024.
Dependencies resolved.
=====
 Package           Architecture      Version       Repository      Size
=====
Installing:
  gcc-c++          x86_64          11.4.1-2.amzn2023.0.2   amazonlinux    12 M
  make             x86_64          1:4.3-5.amzn2023.0.2   amazonlinux    534 k
Installing dependencies:
  annobin-docs     noarch          10.93-1.amzn2023.0.1   amazonlinux    92 k
```

Terminal 2 (app_server):

```
libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64           libcrypt-devel-4.4.33-7.amzn2023.x86_64

Complete!
[ec2-user@ip-172-31-39-216 ~]$ curl -sL https://rpm.nodesource.com/setup_20.x | sudo -E bash -
2024-06-17 04:40:51 - Cleaning up old repositories...
2024-06-17 04:40:51 - Old repositories removed
2024-06-17 04:40:51 - Supported architecture: x86_64
2024-06-17 04:40:51 - Added N|Solid repository for LTS version: 20.x
2024-06-17 04:40:51 - dnf available, updating...
Node.js Packages for Linux RPM based distros - x86_64
N|Solid Packages for Linux RPM based distros - x86_64
Metadata cache created.
2024-06-17 04:40:52 - Repository is configured and updated. Run 'dnf install nodejs -y' to complete the installation.
[ec2-user@ip-172-31-39-216 ~]$ sudo yum install -y nodejs
Last metadata expiration check: 0:00:09 ago on Mon Jun 17 04:40:52 2024.
Dependencies resolved.
=====
 Package           Architecture      Version       Repository      Size
=====
Installing:
  nodejs           x86_64          2:20.14.0-1nodesource   nodesource-nodejs  36 M

Transaction Summary
=====
Install 1 Package

Total download size: 36 M
Installed size: 104 M
Downloading Packages:
nodejs-20.14.0-1nodesource.x86_64.rpm               63 MB/s | 36 MB  00:00
Total                                         63 MB/s | 36 MB  00:00
```

Create a directory "nodeapp" in the "var" directory, then give permissions to the "nodeapp" which we are created.

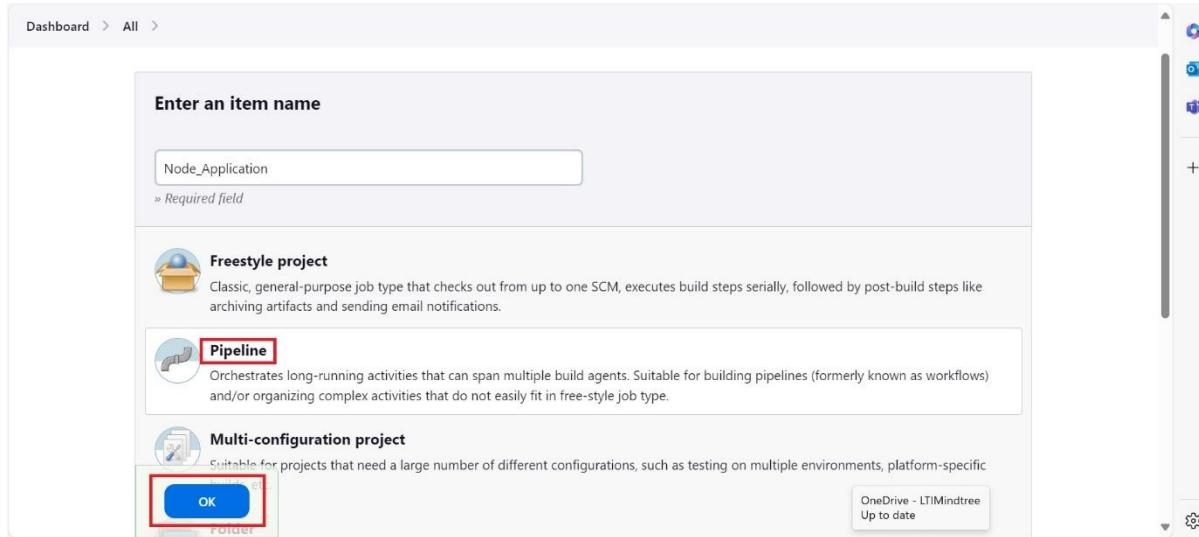
The screenshot shows a single terminal window titled 'Jenkins server' and 'app_server'. It displays the command history and output of a user performing file operations.

```
Complete!
[ec2-user@ip-172-31-39-216 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-39-216 ~]$ cd ..
[ec2-user@ip-172-31-39-216 home]$ cd /var/
[ec2-user@ip-172-31-39-216 var]$ sudo mkdir nodeapp
[ec2-user@ip-172-31-39-216 var]$ ls
account adm cache db empty ftp games kerberos lib local lock log mail nis nodeapp opt preserve run spool tmp yp
[ec2-user@ip-172-31-39-216 var]$ sudo chmod -R '777' nodeapp
[ec2-user@ip-172-31-39-216 var]$ ls -al
total 52
drwxr-xr-x. 20 root root 281 Jun 17 04:50 .
drwxr-xr-x. 18 root root 237 Jun 7 21:37 ..
-rw-r--r--. 1 root root 208 Jun 7 21:37 updated
drwxr-xr-x. 2 root root 19 Jun 7 21:38 account
drwxr-xr-x. 2 root root 6 Jan 30 2023 adm
drwxr-xr-x. 8 root root 88 Jun 7 21:38 cache
drwxr-xr-x. 3 root root 18 Jun 7 21:38 db
drwxr-xr-x. 2 root root 6 Jan 30 2023 empty
drwxr-xr-x. 2 root root 6 Jan 30 2023 ftp
drwxr-xr-x. 2 root root 6 Jan 30 2023 games
drwxr-xr-x. 3 root root 18 Jun 7 21:37 kerberos
drwxr-xr-x. 26 root root 16384 Jun 7 21:38 lib
drwxr-xr-x. 2 root root 6 Jan 30 2023 local
lrwxrwxrwx. 1 root root 11 Jun 7 21:37 lock -> ../run/lock
drwxr-xr-x. 9 root root 16384 Jun 17 04:10 log
lrwxrwxrwx. 1 root root 18 Jun 30 2023 mail -> spool/mail
drwxr-xr-x. 2 root root 6 Jan 30 2023 nis
drwxrwxrwx. 2 root root 6 Jun 17 04:50 nodeapp
drwxr-xr-x. 2 root root 6 Jan 30 2023 opt
drwxr-xr-x. 2 root root 6 Jan 30 2023 preserve
lrwxrwxrwx. 1 root root 6 Jun 7 21:36 run -> ../run
drwxr-xr-x. 5 root root 39 Jun 7 21:38 spool
```

Setting Up Jenkins Job for Node.js Application:

1. In Jenkins, create a new job (Pipeline project).

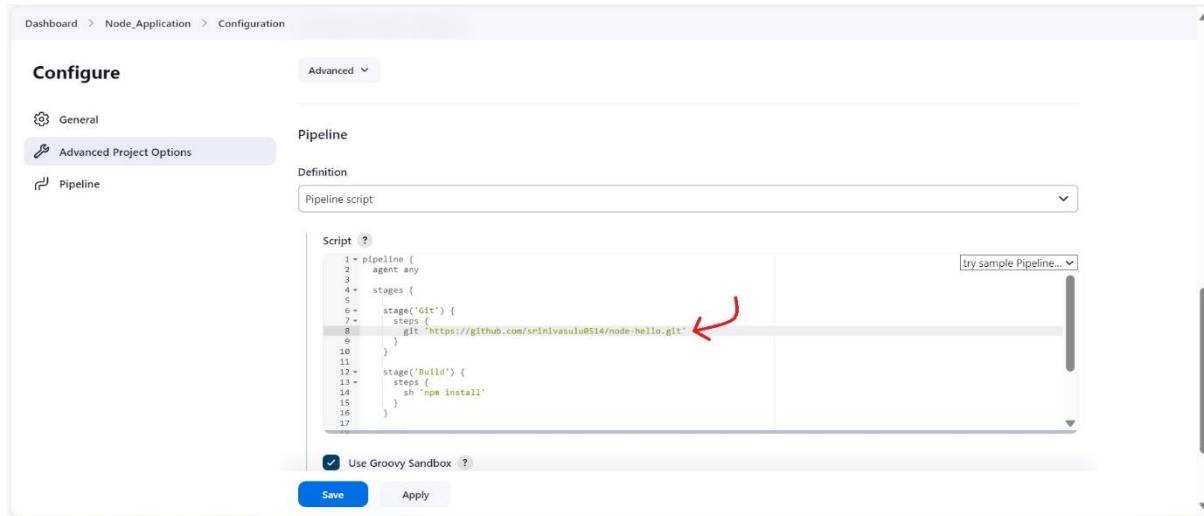
Click on “New item” → choose pipeline project



After creating pipeline → Configure the required steps for the pipeline → Choose GitHub hook trigger under “Build Triggers”

The image contains two screenshots of the Jenkins configuration interface. The top screenshot shows the 'General' configuration page for the 'Node_Application' pipeline. It includes fields for 'Description' (Deploying node App application), 'Discard old builds', and 'Do not allow concurrent builds'. The 'Save' and 'Apply' buttons are at the bottom. The bottom screenshot shows the 'Build Triggers' configuration page. It lists several triggers: 'Preserve stashes from completed builds', 'This project is parameterized', 'Throttle builds', 'Build after other projects are built', 'Build periodically', and 'GitHub hook trigger for GITScm polling'. The 'GitHub hook trigger for GITScm polling' checkbox is checked and highlighted with a red box. Both screenshots show the Jenkins sidebar on the right.

Make sure that paste the Git repository URL on which we are performing.



Before going to Deploy stage, we need to create the “jenkins” Directory (Directory name can be written as per user wish) in “opt” directory, after creating “jenkins” directory give permission to it. (Jenkins server)

```

Verifying : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
Verifying : perl-lib-0.65-477.amzn2023.0.6.x86_64

Installed:
git-2.40.1-amzn2023.0.3.x86_64          git-core-2.40.1-amzn2023.0.3.noarch
perl-Error-1.0.17v29-5.amzn2023.0.2.noarch perl-File-Find-1.37-477.amzn2023.0.6.noarch perl-Git-2.40.1-amzn2023.0.3.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
[ec2-user@ip-172-31-45-40 ~]$ git v
git version 2.40.1
[ec2-user@ip-172-31-45-40 ~]$ client-loop: send disconnect: Connection reset
C:\Users\10741834\Downloads>ssh -i "Jenkins_login.pem" ec2-user@ec2-13-235-42-2.ap-south-1.compute.amazonaws.com
#
Amazon Linux 2023
#
# https://aws.amazon.com/linux/amazon-linux-2023
#
>Last login: Mon Jun 17 04:12:56 2024 from 136.226.233.78
[ec2-user@ip-172-31-45-40 ~]$ cd /opt/
[ec2-user@ip-172-31-45-40 opt]$ ls
aws
[ec2-user@ip-172-31-45-40 opt]$ sudo mkdir jenkins
[ec2-user@ip-172-31-45-40 opt]$ sudo chmod -R 777 jenkins
[ec2-user@ip-172-31-45-40 opt]$ ls
aws
[ec2-user@ip-172-31-45-40 opt]$ 

```

Now copy the key pair data (as we created Jenkins_login.pem/Jenkins_PersonalLaptop.pem), and now move to the “jenkins” directory with “cd jenkins”, create a file aws.pem with help of “nano aws.pem” and save it.

```

[jenkins]$ sudo chmod -R 777 jenkins
[jenkins]$ ls
aws
[jenkins]$ cd jenkins
[jenkins]$ nano aws.pem
[jenkins]$ cat aws.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpIBAAQEAwhTL9Ww4nDBDizcFx7Rd0FSX1usfkfqkhHWVH60hY646
it5PdpPbm4dZjrpSUJnAEHW4TL/MMay1M90ooCJ9TvwF36j0NXSxLFyHrFZVb+g
Sv5t4xh...me6vn62d+i8+DrHwvqZo0+XB0FG2qjTS/oI9450t30Xwism/AuCl
XUhqYMz6hAjNzwzLzJTpcc8yMIDLxkvTdfzrEdcMExdr3GLfzWz6c3Gb0BIt
XzJxJwE+4alvzf3QJNG1K/ES8kGzOEH09z91wuFkiD9ctRa63MKv0o06rJscsAe
hmj/4yfMyxe8rPacFwGs/4xyfvfx1U+vqBXQIDAQABaIBAQCK80Y0tI/Yj1yY
2lJCM3YzFfaUFWS5bjPc3dmh4iFALLmEupGphM9AkWhJgQDvm+/eS04K/mJQKm
/f13zLn/wud9ugWe14AW/rx+sTQTwIGPmPoY2eYlnW8ypQRtmmwvPoST0MaxJD
PKTYtijm5Oxyjkr+cwtMmz4MKVERcmIyPmno3T7GKNEEP0bmkemckxMyU45SPv
QOsMrry0LQBtgQelmzkbjTAZ09hIgSy+r2J0CuLpv8kTApvWEka0ByJGZU5fh
5194Xy0HFPZs9PtlslJDMzjc+pUzauEMgdDXjhk0vLpm95+/TUpRp15VtbiyI4C
LNETcr4xaoGBAoL1yS1JOCfhDE01C/9lyMPiMpvyrtrlnhyBvi6zMjgTgyicCV
Mkr8KTLPDFZYolcNBIXskLvWm919+l7zm4gbJ5r1Gh07Qz2A01e8a8bAuepjNz
DLXDOLiGdAJEk22am6Eg15ulpCw6gAJNwqvK6C5wpjLQJ0wlifBBy5arAoGBANYF
yu/xTT++QclieBkv2yKDU78MbSE6VLAGInVcgQ0b1mginWuUw7tvcp2Fts4xC
eTVuMlttELT8xzhLwZwtMU51vh0hOmKOLOs++9CiGL8MgS4t6P4EzQhwMT6utfi
LUxgym5ldQIg0B8ke9pw5+DgTL2wn+KKZck3b+gXAOGBAIn8Ble50+QvzHk692xW
xy/KXuqzo6eBWXzahJFuvxIz2Blkt1mcbx07ghlrMv+Bgh1+lxIEncbz3/iFUKnX
nYmjyCwzsaCab1On3ME6j9k06xb1FibwmSsiIrr-MRoUmDxz2tPNGifJw7t0TWmW
FwFIVwvVTfc7u7iNw48E9qfLa0GAO3vfkrd/zR+6lDebu4b9JRkgGPpkA+KLKD
ixEMD+tlMfaKFcv/ie/G2FxY2kH5ScNLhRODae0cf3dlqmIzmbIt5X076MCyjAJ
MEjh033BTf+ATVFJzpwtEtkogkzCHBQ0Jc9mc9IYA8X/052FEVL/2yLuELLw1ll
Jta1/mMcgYEAlh7a2TIRily4ea9bnYm34V7wfCbkWkU/NpLU688Lgb144kYvWetzy
Y6+Yfbp+Y9tb5wU7wUZPXDcxmejAkW9Cjz07XWoQByRmhxr3sFRh1z6aJS
CZTRyGZ919cP/xnAu6odXDH82BeTpjhQsbjguzYBjvy7UYml7whLC/w=
-----END RSA PRIVATE KEY-----

```

Now, write the final step (Deploy stage) of the groovy script:

The screenshot shows the Jenkins Pipeline configuration page for a project named "Node_Application". The "Pipeline" tab is selected. The "Script" section contains the following Groovy code:

```
0 9
10
11
12 + stage('Build') {
13 +   steps {
14 +     sh 'npm install'
15 +   }
16
17
18
19 + stage('Deploy') {
20 +   steps {
21 +     sh 'scp -o StrictHostKeyChecking=no -i /opt/jenkins/aws.pem * ec2-user@ip-172-31-39-216:/var/nodeapp'
22 +   }
23
24
25 }
```

Two red arrows point to specific parts of the code: one points to the path "ip-172-31-39-216:/var/nodeapp" and another points to the EC2 server IP "ip-172-31-39-216".

Below the code, there are "Save" and "Apply" buttons. At the bottom right, it says "REST API" and "Jenkins 2.452.2".

i) First arrow indicates the path in Jenkins server. ii) Second arrow indicates the ec2 server and path in app server

Now, save the configuration and build the app.

The screenshot shows the Jenkins dashboard for the "Node_Application" pipeline. The "Build Now" button is highlighted with a red box. The pipeline status is shown as "Deploying node App application". The "Build History" section shows a single build labeled "#5" with a timestamp of "Jun 17, 2024, 5:56 AM".

To check our application, go to app server and make sure that be in "nodeapp" directory, use command "npm start".

After open the Public IP4 address/ Public IP4 DNS → make https to http and add ':3000' at the end of the URL.

Ex: <http://ec2-13-235-100-114.ap-south-1.compute.amazonaws.com:3000/>

```

PS C:\Users\1074183\Downloads> ssh -i "Jenkins_login.pem" ec2-user@ec2-13-235-100-114.ap-south-1.compute.amazonaws.com
Last login: Mon Jun 17 05:29:33 2024 from 136.226.233.105
[ec2-user@ip-172-31-39-216 ~]$ cd /var/
[ec2-user@ip-172-31-39-216 var]$ cd nodeapp
[ec2-user@ip-172-31-39-216 nodeapp]$ ls
README.md index.js package-lock.json package.json
[ec2-user@ip-172-31-39-216 nodeapp]$ npm start
> node-hello@1.0.0 start
> node index.js
Server running on http://localhost:3000/

```



To Automate the **Build** process, we need to step the GitHub webhooks in GitHub and for **refreshing** the website install pm2 library in app server (add watcher to the pm2).

Steps:

Go to the node-hello repository in GitHub → Click on settings which is beside of insights → choose Webhooks → add the Jenkins URL under Payload URL → save.

The screenshot shows the GitHub repository settings for 'node-hello'. The 'Webhooks' tab is active. On the left sidebar, the 'Webhooks' section is highlighted with a red box. The main area shows the 'Webhooks / Manage webhook' configuration. It includes fields for 'Payload URL' (containing 'http://ec2-13-235-42-2.ap-south-1.compute.amazonaws.com:8080/'), 'Content type' (set to 'application/x-www-form-urlencoded'), and 'Secret' (an empty field). A note states: 'We'll send a post request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#)'.

URL should be: Jenkins server//github-webhook/

Ex: <http://ec2-13-235-42-2.ap-south-1.compute.amazonaws.com:8080//github-webhook/>

Now, install pm2 library → make our index.js file to watch

```
Jenkins_server x app_server x + - X

To go further checkout:  
http://pm2.io/  
-----  
[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2  
[PM2] PM2 Successfully daemonized  
[PM2] Starting /var/nodeapp/index.js in fork_mode (1 instance)  
[PM2] Done.  


| id | name  | mode |   | status | cpu | memory |
|----|-------|------|---|--------|-----|--------|
| 0  | index | fork | 0 | online | 0%  | 35.6mb |

  
[ec2-user@ip-172-31-39-216 nodeapp]$ [pm2 stop index.js]  
[PM2] Applying action stopProcessId on app [index.js](ids: [ 0 ])  
[PM2] [index.js](0) ✓  

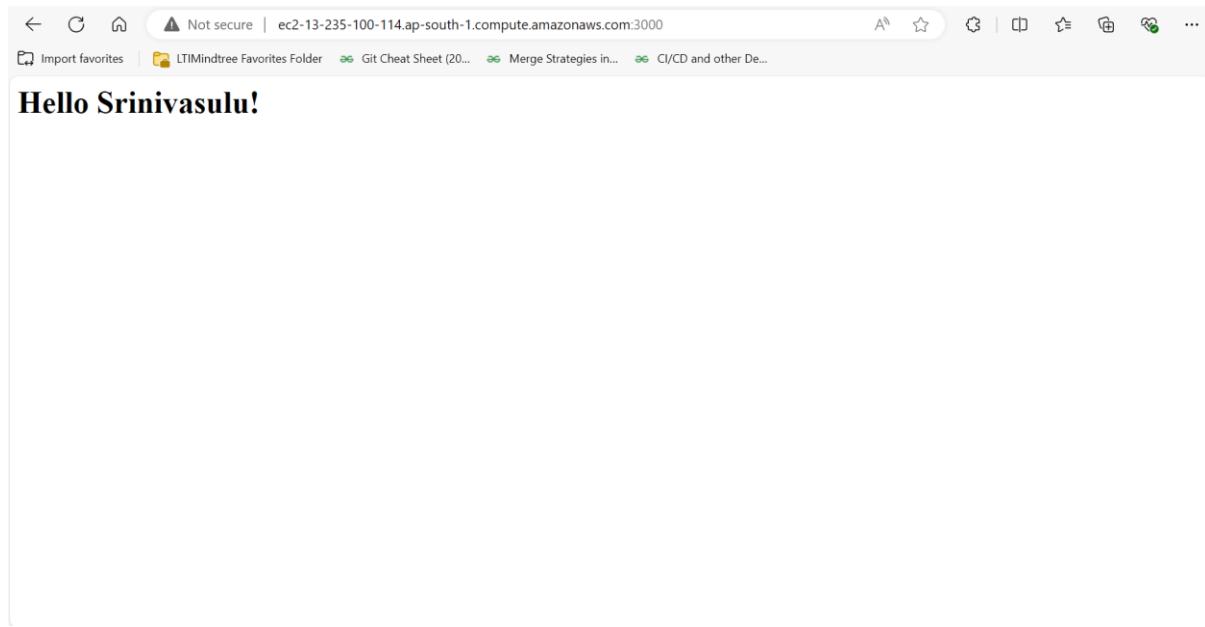

| id | name  | mode |    | status  | cpu | memory |
|----|-------|------|----|---------|-----|--------|
| 0  | index | fork | 15 | stopped | 0%  | 0b     |

  
[ec2-user@ip-172-31-39-216 nodeapp]$ [pm2 start index.js --watch]  
[PM2] Applying action restartProcessId on app [index](ids: [ 0 ])  
[PM2] [index](0) ✓  
[PM2] Process successfully started  


| id | name  | mode |    | status | cpu | memory |
|----|-------|------|----|--------|-----|--------|
| 0  | index | fork | 15 | online | 0%  | 28.8mb |

  
[ec2-user@ip-172-31-39-216 nodeapp]$
```

Now commit changes in the git through VS code/ other Editor tools → push to the GitHub → see the changes in Jenkins and application.



Master – slave:

In Jenkins, the **Master** manages build jobs and dispatches them to **Slave** nodes for execution. Slaves handle the actual build tasks.

Create a new EC2 instance for slave agent and connect the instance in command prompt.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10741834> cd Downloads
PS C:\Users\10741834\Downloads> ssh -i "Jenkins_login.pem" ec2-user@ec2-13-234-59-159.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-234-59-159.ap-south-1.compute.amazonaws.com (13.234.59.159)' can't be established.
ED25519 key fingerprint is SHA256:5NIJ9Hl0fmf6Lyys0s6n+CwmcLjU9Rpm6kCe1Cvg1UU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-234-59-159.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.

#_
~\_ #####_      Amazon Linux 2023
~~\_#####\_
~~ \###|_
~~  \|/_-->  https://aws.amazon.com/linux/amazon-linux-2023
~~  \|/_/-
~~ .-/_/-
~/m/-
[ec2-user@ip-172-31-36-227 ~]$ |
```

A screenshot of a Windows PowerShell window titled 'slaveagent'. The window shows a command being run to establish an SSH connection to an EC2 instance. It prompts for confirmation about the host's fingerprint and adds it to the list of known hosts. The output includes the Amazon Linux 2023 logo and a URL for the distribution.

Now install git, java, node.js in the slave agent server.

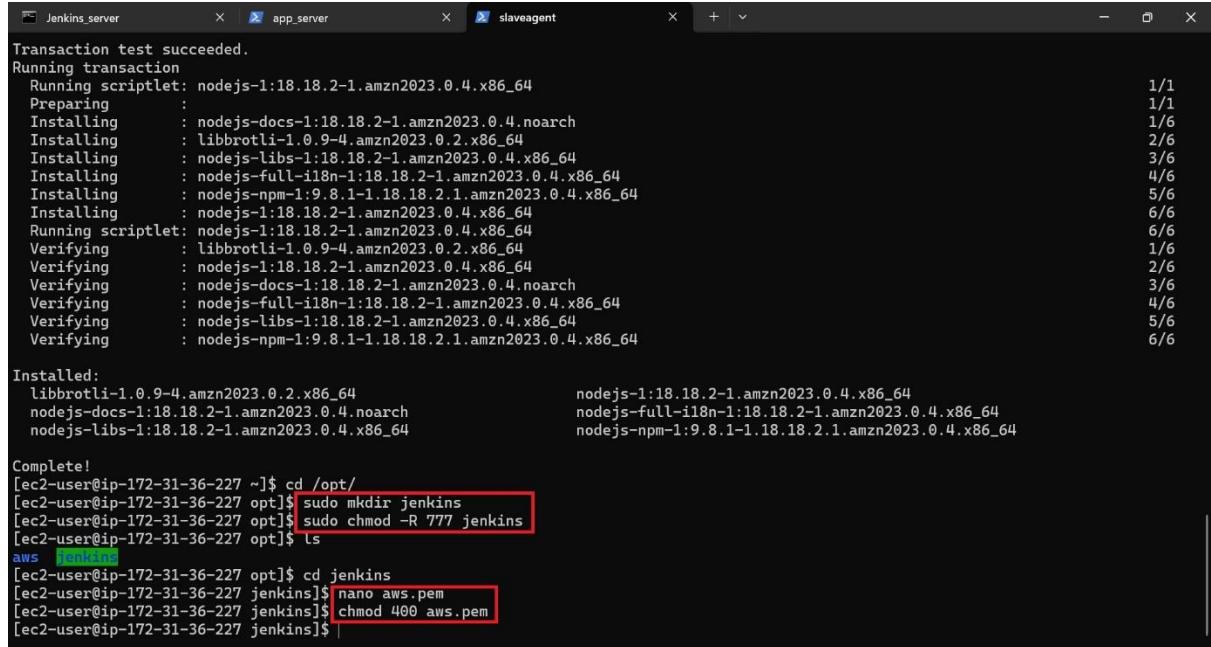
For git: sudo yum install git -y

For node.js: sudo yum install nodejs -y

For Java: sudo yum install java -y

Create “jenkins” directory in the “opt” directory and change the permissions. Create “aws.pem” key pair in the “jenkins” directory.

Note: **chmod 400 <filename>**, it sets the permissions of the file so that only the owner has read permissions. This is commonly used to secure sensitive files, such as SSH private keys.



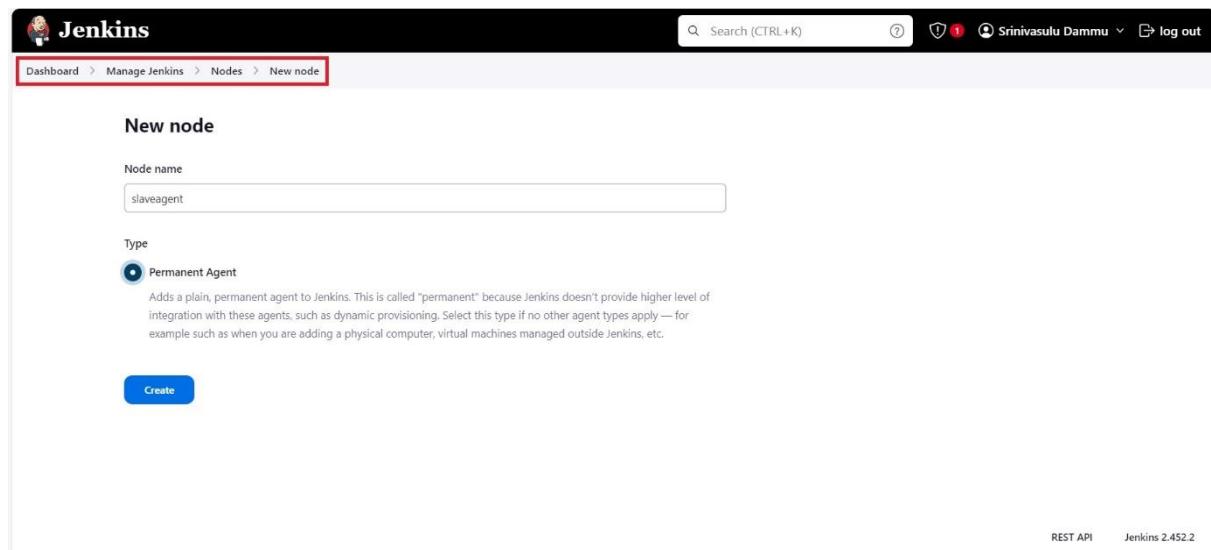
```
Transaction test succeeded.
Running transaction
  Running scriptlet: nodejs-1:18.18.2-1.amzn2023.0.4.x86_64          1/1
  Preparing : nodejs-docs-1:18.18.2-1.amzn2023.0.4.noarch                1/1
  Installing : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                   1/6
  Installing : nodejs-libs-1:18.18.2-1.amzn2023.0.4.x86_64             2/6
  Installing : nodejs-full-i18n-1:18.18.2-1.amzn2023.0.4.x86_64        3/6
  Installing : nodejs-npm-1:9.8.1-1.18.18.2.1.amzn2023.0.4.x86_64       4/6
  Installing : nodejs-1:18.18.2-1.amzn2023.0.4.x86_64                  5/6
  Installing : nodejs-1:18.18.2-1.amzn2023.0.4.x86_64                  6/6
  Running scriptlet: nodejs-1:18.18.2-1.amzn2023.0.4.x86_64          6/6
  Verifying  : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                   1/6
  Verifying  : nodejs-1:18.18.2-1.amzn2023.0.4.x86_64                 2/6
  Verifying  : nodejs-docs-1:18.18.2-1.amzn2023.0.4.noarch              3/6
  Verifying  : nodejs-full-i18n-1:18.18.2-1.amzn2023.0.4.x86_64         4/6
  Verifying  : nodejs-libs-1:18.18.2-1.amzn2023.0.4.x86_64              5/6
  Verifying  : nodejs-npm-1:9.8.1-1.18.18.2.1.amzn2023.0.4.x86_64       6/6

Installed:
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64                         nodejs-1:18.18.2-1.amzn2023.0.4.x86_64
  nodejs-docs-1:18.18.2-1.amzn2023.0.4.noarch                      nodejs-full-i18n-1:18.18.2-1.amzn2023.0.4.x86_64
  nodejs-libs-1:18.18.2-1.amzn2023.0.4.x86_64                      nodejs-npm-1:9.8.1-1.18.18.2.1.amzn2023.0.4.x86_64

Complete!
[ec2-user@ip-172-31-36-227 ~]$ cd /opt/
[ec2-user@ip-172-31-36-227 opt]$ sudo mkdir jenkins
[ec2-user@ip-172-31-36-227 opt]$ sudo chmod -R 777 jenkins
[ec2-user@ip-172-31-36-227 opt]$ ls
aws jenkins
[ec2-user@ip-172-31-36-227 opt]$ cd jenkins
[ec2-user@ip-172-31-36-227 jenkins]$ nano aws.pem
[ec2-user@ip-172-31-36-227 jenkins]$ chmod 400 aws.pem
[ec2-user@ip-172-31-36-227 jenkins]$
```

Now create a slave agent node in Jenkins:

Dashboard → Manage Jenkins → Nodes → new Node



New node

Node name

slaveagent

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called “permanent” because Jenkins doesn’t provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

Make sure that, the remote directory should be matches the path of application in the app server.

The screenshot shows the 'Configure' screen for the 'slaveagent' node. Key fields include:

- System Information: Number of executors: 1
- Build Executor Status: 1 Idle
- Remote root directory: /opt/jenkins (highlighted with a red box)
- Labels: slaveagent
- Usage: Only build jobs with label expressions matching this node (highlighted with a red box)
- Launch method: Launch agent by connecting it to the controller
- Availability: Keep this agent online as much as possible
- Save button (highlighted with a red box)

Now add the security port in the Jenkins: (Dashboard → Manage Jenkins → Security)

Adding a security port for the Jenkins slave agent ensures secure communication between the master and agents.

The screenshot shows the 'Security' configuration page. Key settings include:

- Agents: TCP port for inbound agents: Fixed (8081) (highlighted with a red box)
- Agent protocols: Agent protocols dropdown
- CSRF Protection: Default Crumb Issuer
- Save button (highlighted with a red box)

Add "8081" in security group (slave_agent EC2 instance).

In Jenkins (Dashboard → Manage Jenkins → Nodes → "slaveagent")

Copy the Unix commands and connect the slaveagent.

The screenshot shows the 'Agent slaveagent' status page. Key details include:

- Status: Agent is connected (highlighted with a red box)
- Monitoring Data: None
- Projects tied to slaveagent: None
- Build Executor Status: 1 Idle
- REST API Jenkins 2.452.2

Now, go to the (Dashboard → Node_Application (pipeline project which we have created) → configure)

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```

1 pipeline {
2   agent { label "slaveagent" } ← Red arrow points here
3 
4   stages {
5     stage('Get') {
6       steps {
7         git 'https://github.com/srinivasulu0514/node-hello.git'
8       }
9     }
10    stage('Build') {
11      steps {
12        sh 'npm install'
13      }
14    }
15  }
16}

```

Below the script, there are 'Save' and 'Apply' buttons.

Note: Name the label which we were created (ex: agentslave).

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```

1 pipeline {
2   agent { label "agentslave" } ← Red arrow points here
3 
4   stages {
5     stage('Get') {
6       steps {
7         git 'https://github.com/srinivasulu0514/node-hello.git'
8       }
9     }
10    stage('Build') {
11      steps {
12        sh 'npm install'
13      }
14    }
15    stage('Deploy') {
16      steps {
17        sh "scp -o StrictHostKeyChecking=no -i /opt/jenkins/aus.pem ec2-user@ip-172-31-39-216:/var/nodeapp"
18      }
19    }
20  }
21}

```

Below the script, there is a checkbox for 'Use Groovy Sandbox' and a 'Pipeline Syntax' section. The 'Save' button is highlighted with a red box.

We have already created automate process above, refer to it. Now commit changes in the git through VS code/ other Editor tools → push to the GitHub → see the changes in Jenkins and application.

The screenshot shows the VS Code interface with the terminal tab open. The terminal output shows the following commands being run:

```

git commit -am "Changes from Srinivasulu to Srinivasulu-Devops"
git push origin master

```

The commit message and push command are highlighted with a red box.

Auto Build is completed.

The screenshot shows the Jenkins interface for the 'Node_Application' project. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, Pipeline Syntax, and GitHub Hook Log. The main area is titled 'Node_Application' with the subtitle 'Deploying node App application'. It features a 'Permalinks' section with a list of recent builds. Below that is a 'Build History' section with a table showing the last 10 builds. The most recent build (#10) is highlighted with a red border and shows the date 'Jun 17, 2024, 9:46 AM'.

Finally, the updated application is done with help of slave agent.

The screenshot shows a web browser window with the URL 'ec2-13-235-100-114.ap-south-1.compute.amazonaws.com:3000'. The page content is 'Hello Srinivasulu - Devops !'.

6. Conclusion

Summary of Steps

- Set up Git and GitHub for version control.
- Launch and configure an Amazon EC2 instance.
- Install and configure Jenkins on EC2.
- Configure Jenkins to build and deploy a Node.js application.
- Creating slave agent to perform the task.
- Performing task using Master-slave concept.

Troubleshooting Tips

- Ensure security groups are correctly configured.
- Check Jenkins logs for build errors.
- Verify GitHub webhook configuration.