

Deploy a simple Node.js App to AWS EC2 using Jenkins CI/CD Pipeline

1. At first , Launch an Amazon Linux 2 instance in EC2 and name it as **jenkins_server**

The screenshot displays the AWS Management Console interface. The top navigation bar includes the AWS logo, a search bar, and various service icons. The left sidebar shows the 'EC2' menu with options like Dashboard, Global View, Events, and Instances. The main content area is titled 'Instances (1/1)' and shows a table with one instance, 'jenkins_server', in a 'Running' state. Below the table, the 'Details' tab for instance 'i-04ea9d4d58c5cf378' is selected, showing a summary of its configuration including IP addresses, DNS names, and instance type.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
jenkins_server	i-04ea9d4d58c5cf378	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-201-71-211.ap-...	13.201.71.211

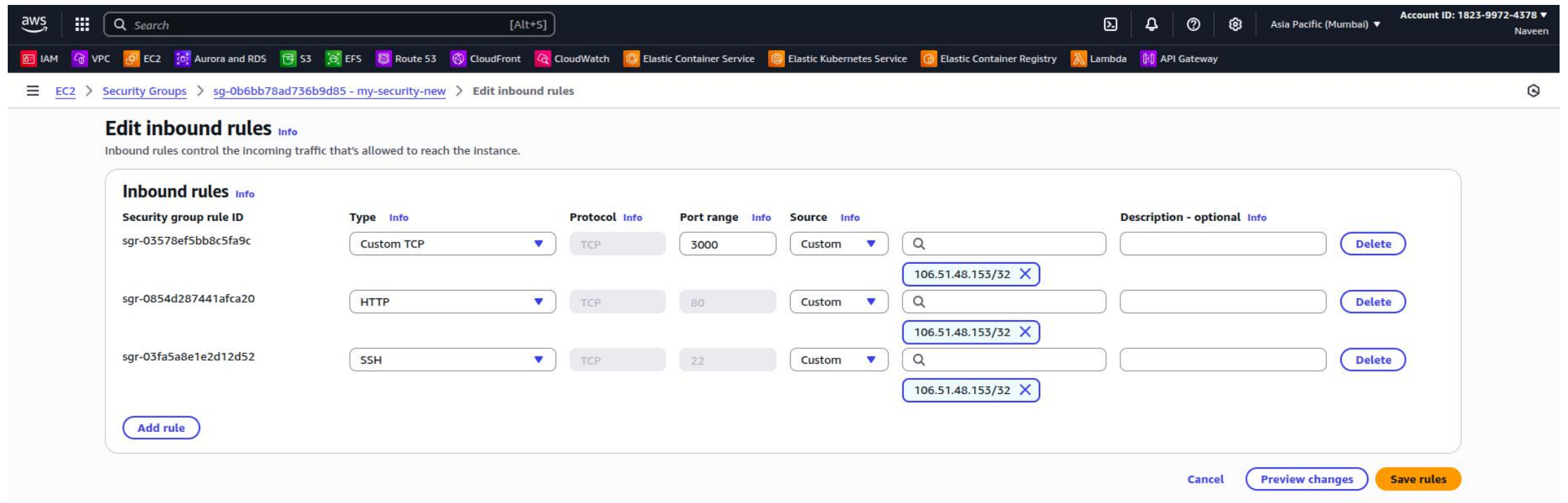
i-04ea9d4d58c5cf378 (Jenkins_server)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID i-04ea9d4d58c5cf378	Public IPv4 address 13.201.71.211 open address	Private IPv4 addresses 172.31.47.138
IPv6 address -	Instance state Running	Public DNS ec2-13-201-71-211.ap-south-1.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-47-138.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-47-138.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 13.201.71.211 [Public IP]	VPC ID vpc-099e47a9d31b97ddc	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0975fed6cbf3fd7f0	Managed
IMDSv2	Instance ARN	

2. In the Security Group inbound rules, allow the following: **SSH (22) HTTP (80) Custom TCP (3000) for Node.js**



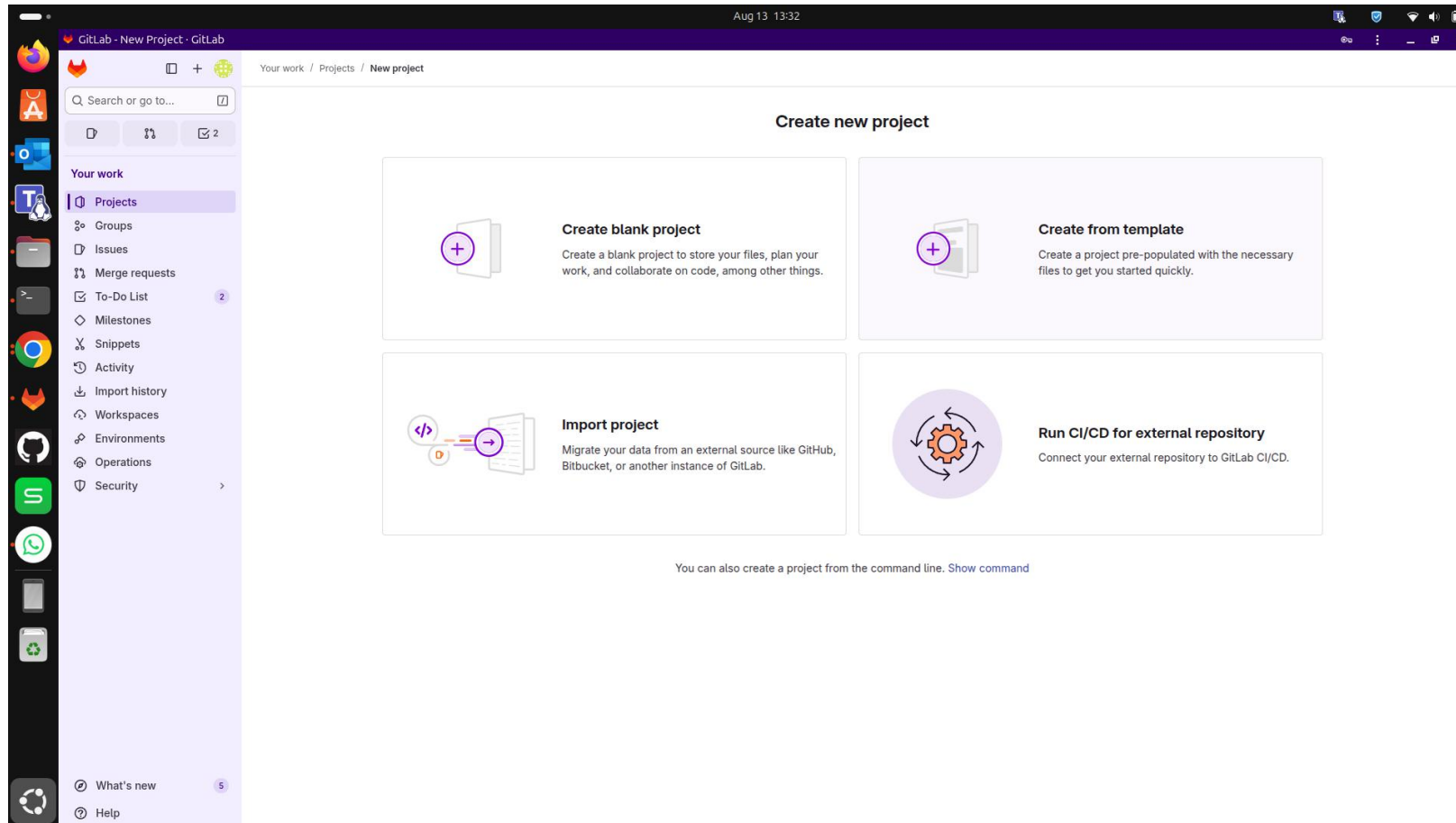
The screenshot shows the AWS Management Console interface for editing inbound rules on a Security Group. The top navigation bar includes the AWS logo, a search bar, and various service icons. The breadcrumb trail indicates the path: EC2 > Security Groups > sg-0b6bb78ad736b9d85 - my-security-new > Edit inbound rules. The main heading is 'Edit inbound rules' with an 'Info' link. Below the heading is a subheading 'Inbound rules' and a description: 'Inbound rules control the incoming traffic that's allowed to reach the instance.'

The 'Inbound rules' section contains a table with the following columns: Security group rule ID, Type, Protocol, Port range, Source, and Description - optional. There are three rules listed:

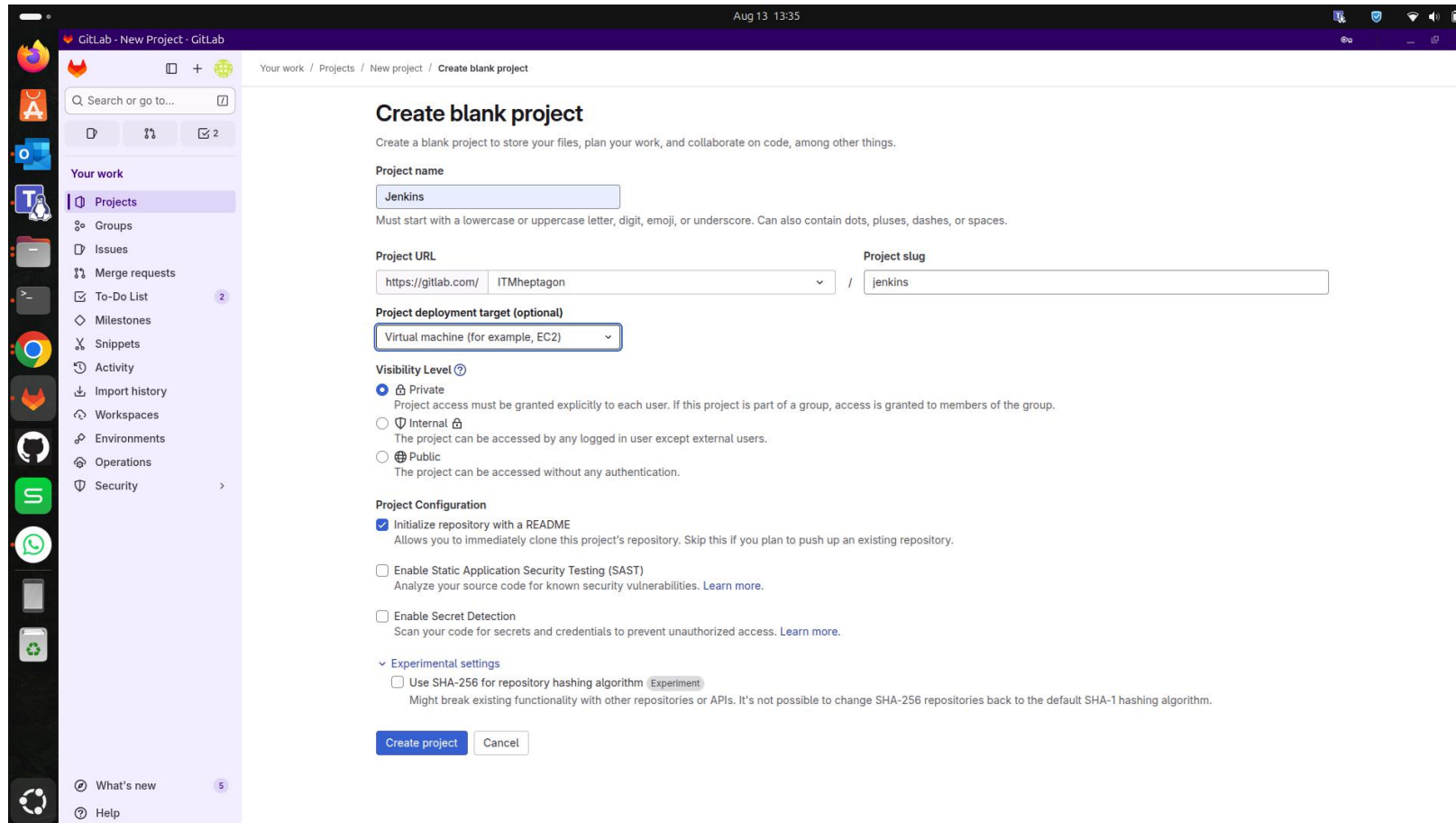
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-03578ef5bb8c5fa9c	Custom TCP	TCP	3000	Custom	
sgr-0854d287441afca20	HTTP	TCP	80	Custom	
sgr-03fa5a8e1e2d12d52	SSH	TCP	22	Custom	

Each rule has a 'Delete' button next to it. The 'Source' field for each rule is highlighted with a blue box containing the IP address 106.51.48.153/32 and a close button (X). At the bottom right, there are three buttons: 'Cancel', 'Preview changes', and 'Save rules'.

3. In GitLab, create a new project by selecting **Create Blank Project**



4. Name the project as **Jenkins**, select your **Gitlab URL** & select the project deployment target as Virtual machine (EC2). Make the repo as **private** and **create project**.



The screenshot shows the GitLab web interface for creating a new project. The left sidebar contains navigation links for 'Your work' (Projects, Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Import history, Workspaces, Environments, Operations, Security) and 'What's new' / 'Help'. The main content area is titled 'Create blank project' and includes the following fields and options:

- Project name:** A text input field containing 'Jenkins'.
- Project URL:** A dropdown menu showing 'https://gitlab.com/' and 'ITMheptagon'.
- Project slug:** A text input field containing 'jenkins'.
- Project deployment target (optional):** A dropdown menu showing 'Virtual machine (for example, EC2)'.
- Visibility Level:** Radio buttons for 'Private' (selected), 'Internal', and 'Public'.
- Project Configuration:** Checkboxes for 'Initialize repository with a README' (checked), 'Enable Static Application Security Testing (SAST)', and 'Enable Secret Detection'.
- Experimental settings:** A checkbox for 'Use SHA-256 for repository hashing algorithm'.

At the bottom of the form are two buttons: 'Create project' and 'Cancel'.

5. Once the repo is created, clone it to your local system.

The screenshot displays the GitLab web interface for a repository named 'Jenkins' under the user 'IT Heptagon'. The left sidebar contains navigation links for Project, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings, What's new, and Help. The main content area shows the 'Initial commit' by 'IT Heptagon' 3 minutes ago. Below this is a table of files with columns for Name, Last commit, and Last update. The 'README.md' file is highlighted, showing its content. The README includes sections for 'Getting started' and 'Add your files', with a code block for cloning the repository. The right sidebar provides project information, including commit, branch, and tag counts, and a list of suggested actions like 'Add LICENSE' and 'Set up CI/CD'.

IT Heptagon / Jenkins

Search or go to...

Project

- Jenkins
- Pinned
- Issues
- Merge requests
- Manage
- Plan
- Code
- Build
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings
- What's new
- Help

Jenkins

main jenkins

Initial commit
IT Heptagon authored 3 minutes ago

41908e9a History

Name	Last commit	Last update
README.md	Initial commit	3 minutes ago

README.md

Jenkins

Getting started

To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Already a pro? Just edit this README.md and make it your own. Want to make it easy? [Use the template at the bottom!](#)

Add your files

- ☐ Create or upload files
- ☐ Add files using the command line or push an existing Git repository with the following command:

```
cd existing_repo
git remote add origin https://gitlab.com/ITHeptagon/jenkins.git
git branch -M main
git push -uf origin main
```

Integrate with your tools

- ☐ Set up project integrations

Collaborate with your team

- ☐ Invite team members and collaborators
- ☐ Create a new merge request

Project information

- 1 Commit
- 1 Branch
- 0 Tags
- 4 KiB Project Storage

- README
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Enable Auto DevOps
- + Add Kubernetes cluster
- + Set up CI/CD
- + Add Wiki
- + Configure Integrations

Created on
August 13, 2025

6. Create a **Jenkinsfile** to trigger a build from GitLab and deploy the **Node.js** app to the EC2 server

```
1 pipeline {
2   agent any
3
4   tools {
5     nodejs "Node 18" // Must match Manage Jenkins → Tools → NodeJS Installations
6   }
7
8   environment {
9     EC2_USER = "ec2-user" // Change to 'ubuntu' if needed
10    EC2_HOST = "13.201.71.211" // Public IP or DNS
11    EC2_PATH = "/home/ec2-user/jenkins" // Deployment directory on EC2
12  }
13
14  stages {
15    stage('Checkout') {
16      steps {
17        git branch: 'main',
18            credentialsId: '7243f613-7608-4328-97ed-da54cc32ccb7',
19            url: 'https://gitlab.com/ITWheptagon/jenkins.git'
20      }
21    }
22
23    stage('Build') {
24      steps {
25        sh 'npm install'
26      }
27    }
28
29    stage('Test') {
30      steps {
31        sh 'npm test'
32      }
33    }
34
35    stage('Deploy') {
36      steps {
37        withCredentials([sshUserPrivateKey(credentialsId: 'cc32cf60-9256-446b-b6a7-a7accb97c597', keyFileVariable: 'SSH_KEY')]) {
38          sh """
39            ssh -i $SSH_KEY -o StrictHostKeyChecking=no $EC2_USER@$EC2_HOST '
40              mkdir -p $EC2_PATH
41              if ! command -v npm && /dev/null; then
42                curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
43                sudo yum install -y nodejs
44              fi
45              scp -i $SSH_KEY -o StrictHostKeyChecking=no index.js package.json $EC2_USER@$EC2_HOST:$EC2_PATH
46              ssh -i $SSH_KEY -o StrictHostKeyChecking=no $EC2_USER@$EC2_HOST "
47                cd $EC2_PATH && npm install && node index.js
48              "
49            """
50        }
51      }
52    }
53  }
54
55  post {
56    always {
57      echo 'Pipeline completed!'
58    }
59  }
60 }
```

```

pipeline {
  agent any

  tools {
    nodejs "Node 18" // Must match Manage Jenkins → Tools → NodeJS installations
  }

  environment {
    EC2_USER = "ec2-user"          // Change to 'ubuntu' if needed
    EC2_HOST = "13.201.71.211"     // Public IP or DNS
    EC2_PATH = "/home/ec2-user/jenkins" // Deployment directory on EC2
  }

  stages {
    stage('Checkout') {
      steps {
        git branch: 'main',
            credentialsId: '7243f613-7608-4328-97ed-da54cc32ccb7',
            url: 'https://gitlab.com/ITMheptagon/jenkins.git'
      }
    }

    stage('Build') {
      steps {
        sh 'npm install'
      }
    }

    stage('Test') {
      steps {
        sh 'npm test'
      }
    }

    stage('Deploy') {
      steps {
        withCredentials([sshUserPrivateKey(credentialsId: 'cc32cf60-9256-446b-b6a7-a7accb97c597', keyFileVariable: 'SSH_KEY')]) {
          sh """
            # Create app directory on EC2
            ssh -i $SSH_KEY -o StrictHostKeyChecking=no $EC2_USER@$EC2_HOST "mkdir -p $EC2_PATH"

            # Copy latest code to EC2
            scp -i $SSH_KEY -o StrictHostKeyChecking=no index.js package.json $EC2_USER@$EC2_HOST:$EC2_PATH

            # Install dependencies & restart app with PM2
            ssh -i $SSH_KEY -o StrictHostKeyChecking=no $EC2_USER@$EC2_HOST '
              cd $EC2_PATH && npm install
              if ! command -v pm2 &> /dev/null; then
                sudo npm install -g pm2
              fi
              pm2 start ./index.js --name hello-jenkins --update-env || pm2 restart hello-jenkins
            '
          ,

```


7. Create an **index.js** file for the Node.js server

main jenkins / index.js

index.js Find file Blame Edit

 **Edit index.js**
IT Heptagon authored 40 seconds ago

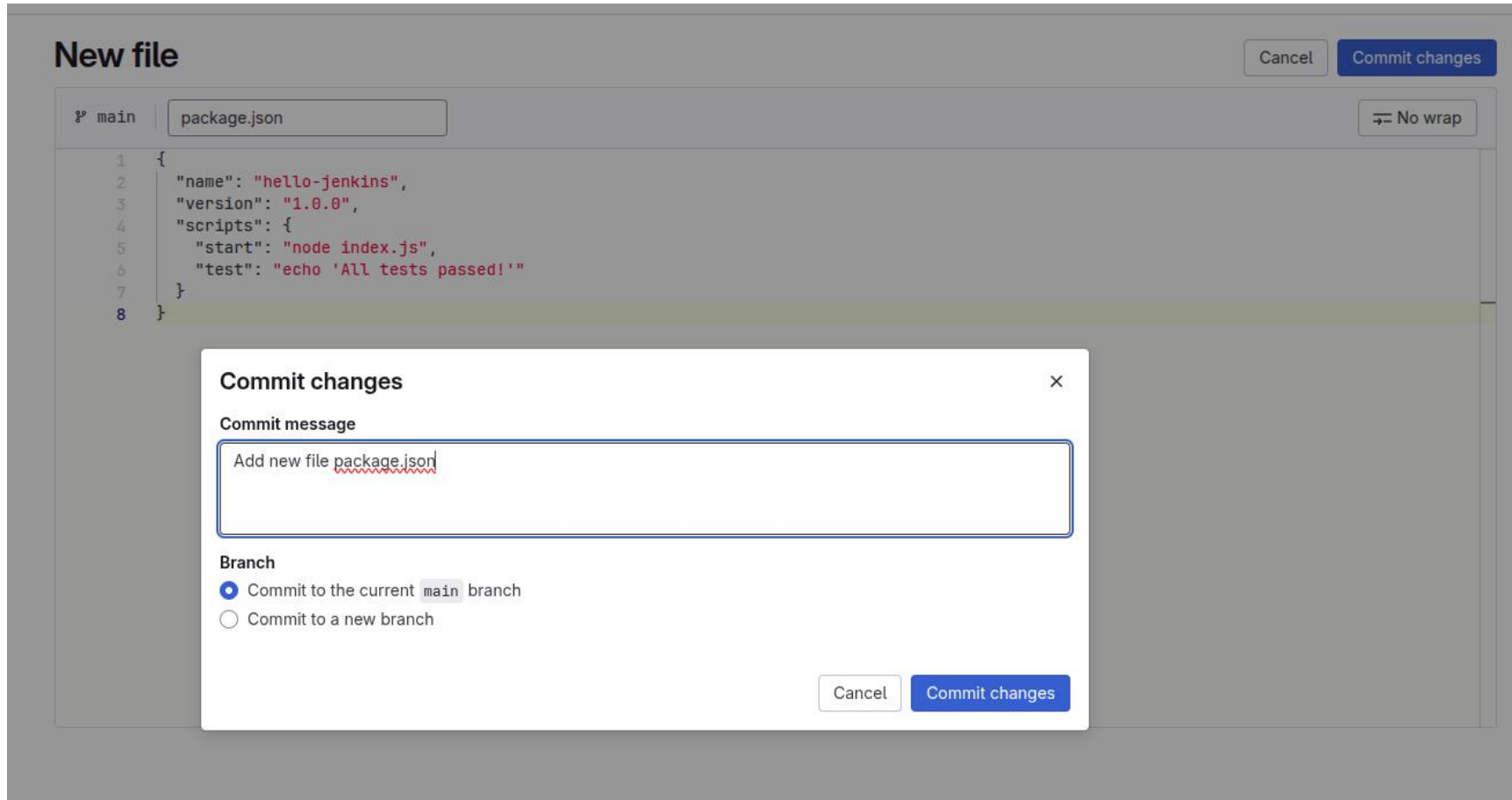
2858a583  History

index.js 342 B   

```
1  const http = require('http');
2
3  const hostname = '0.0.0.0';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    res.statusCode = 200;
8    res.setHeader('Content-Type', 'text/plain');
9    res.end('Hello, Jenkins!\n');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://${hostname}:${port}/`);
14 });
15
```

```
const http = require('http');
const hostname = '0.0.0.0';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, Jenkins!\n');
});
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

8. Also you need to create **package.json** file and commit the changes.



```
name      "hello-jenkins"  
version   "1.0.0"  
scripts  
start "node index.js"  
test  "echo 'All tests passed!'"
```

9. Now you can see all the files in the repo, copy the HTTPS url for cloning the repo.

The file has been successfully created.

main

jenkins

jenkins

Add new file package.json

IT Heptagon authored 18 seconds ago

This project manages its dependencies using **npm**. [Learn more](#)

Name	Last commit
Jenkinsfile	Edit Jenkinsfile
README.md	Initial commit
build.sh	Add new file build.sh
deploy.sh	Add new file deploy.sh
index.js	Add new file index.js
package.json	Add new file package.json
test.sh	Add new file test.sh

README.md

+

Find file

Code

Clone with SSH

git@gitlab.com:ITMheptagon/jenki

Clone with HTTPS

https://gitlab.com/ITMheptagon/j

Open with

Visual Studio Code

SSH

HTTPS

IntelliJ IDEA

SSH

HTTPS

Download source code

zip

tar.gz

tar.bz2

tar

Your workspaces

A workspace is a virtual sandbox environment for your code in GitLab.

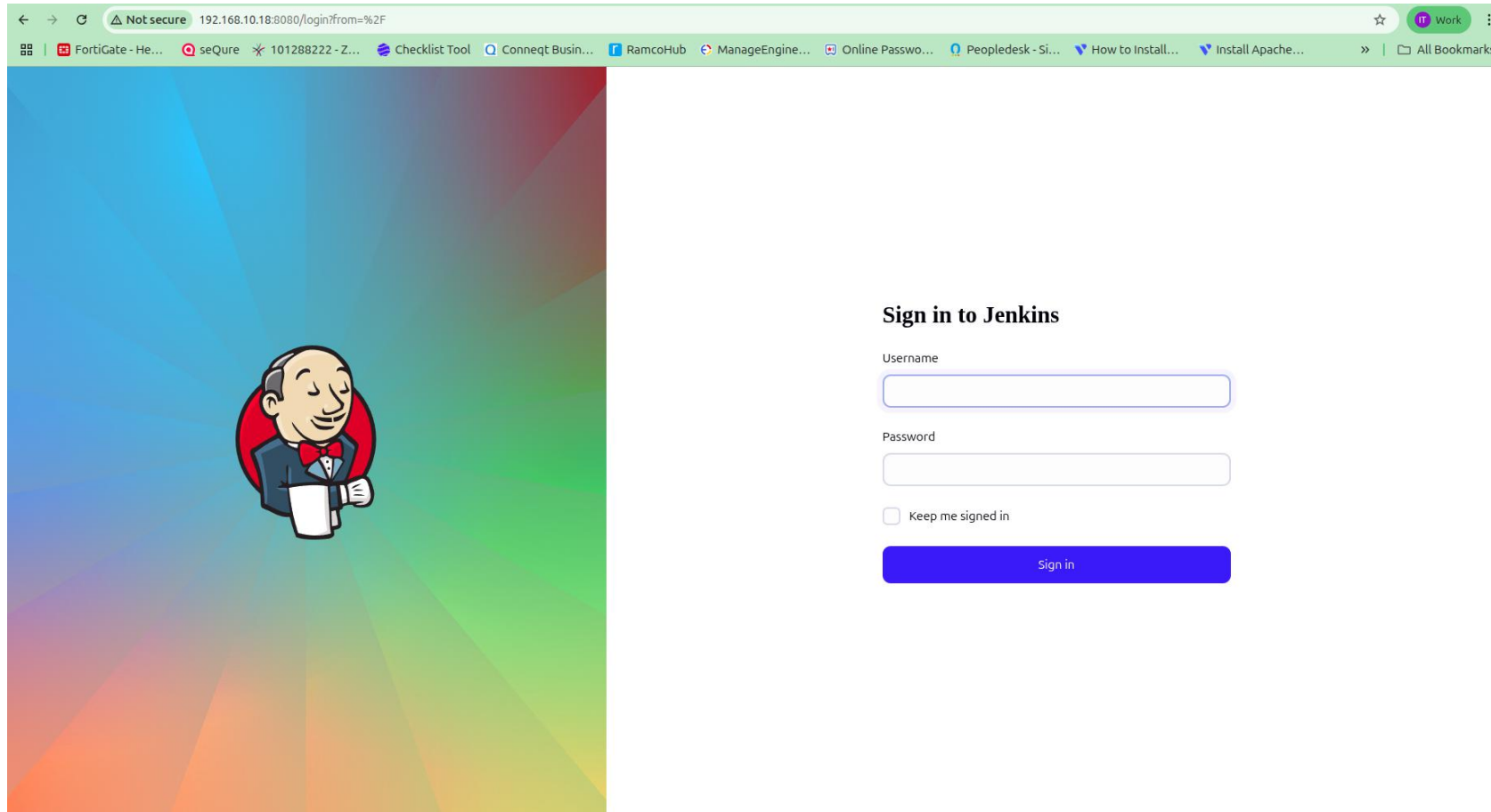
No agents available to create workspaces. Please consult [Workspaces documentation](#) for troubleshooting.

6 minutes ago

10. Make sure Git is installed in your local machine and clone the repo with the copied HTTPS url **git clone https://gitlab.com/ITMheptagon/jenkins.git**

```
root@5CD20890KH: /home/naveen/jenkins
root@5CD20890KH:/home/naveen# git --version
git version 2.43.0
root@5CD20890KH:/home/naveen# ls
aws Desktop Documents Downloads Music new-project nginx-php nginx-php.zip nodejs Pictur
root@5CD20890KH:/home/naveen# git clone https://gitlab.com/ITMheptagon/jenkins.git
Cloning into 'jenkins'...
Username for 'https://gitlab.com': ITMheptagon
Password for 'https://ITMheptagon@gitlab.com':
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 24 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (24/24), 5.39 KiB | 5.39 MiB/s, done.
Resolving deltas: 100% (6/6), done.
root@5CD20890KH:/home/naveen# ls
aws Desktop Documents Downloads jenkins Music new-project nginx-php nginx-php.zip nodejs
root@5CD20890KH:/home/naveen# cd jenkins/
root@5CD20890KH:/home/naveen/jenkins# ls
build.sh deploy.sh index.js Jenkinsfile package.json README.md test.sh
root@5CD20890KH:/home/naveen/jenkins#
```

11. Install jenkins package in your local machine and login to the URL **http://localhost:8080** (Ensure port 8080 is open)



12. In the Dashboard, go to Manage Jenkins → Plugins and install the **NodeJS plugin**.

The screenshot displays the Jenkins 'Manage Jenkins' dashboard. The left sidebar contains navigation links: Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (highlighted), and My Views. Below these are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '0/2'). The main content area features a notification bar at the top stating 'New version of Jenkins (2.516.1) is available for [download](#) ([changelog](#)).'. The dashboard is organized into four main sections: 'System Configuration' (containing System, Tools, Plugins, Clouds, Appearance, and Managed files), 'Security' (containing Security, Credentials, Credential Providers, and Users), 'Status Information' (containing System Information, System Log, Load Statistics, and About Jenkins), and 'Troubleshooting'. The 'Plugins' option in the System Configuration section is highlighted with a blue border and a red badge indicating 48 updates.

Dashboard > Manage Jenkins

Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views

Build Queue
No builds in the queue.

Build Executor Status 0/2

New version of Jenkins (2.516.1) is available for [download](#) ([changelog](#)).

System Configuration

- System**
Configure global settings and paths.
- Tools**
Configure tools, their locations and automatic installers.
- Plugins** 48
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Clouds**
Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**
Configure the look and feel of Jenkins
- Managed files**
e.g. settings.xml for maven, central managed scripts, custom files, ...
- Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

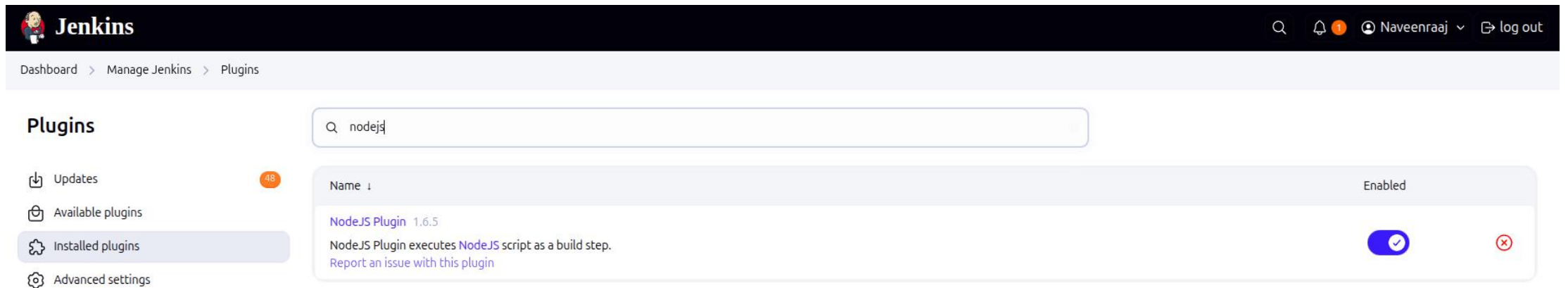
- Security**
Secure Jenkins; define who is allowed to access/use the system.
- Credentials**
Configure credentials
- Credential Providers**
Configure the credential providers and types
- Users**
Create/delete/modify users that can log in to this Jenkins.

Status Information

- System Information**
Displays various environmental information to assist trouble-shooting.
- System Log**
System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics**
Check your resource utilization and see if you need more computers for your builds.
- About Jenkins**
See the version and license information.

Troubleshooting

13. Install **NodeJS** plugin.



The screenshot shows the Jenkins web interface. At the top is a dark header with the Jenkins logo and name on the left, and search, notifications (1), user profile (Naveenraaj), and a log out button on the right. Below the header is a breadcrumb trail: Dashboard > Manage Jenkins > Plugins. The main content area is titled 'Plugins' and features a search bar with 'nodejs' entered. On the left side of the main area, there is a sidebar with four menu items: 'Updates' (with a 48 update badge), 'Available plugins', 'Installed plugins' (which is highlighted), and 'Advanced settings'. The main area displays a table of installed plugins. The table has two columns: 'Name' and 'Enabled'. One plugin is listed: 'NodeJS Plugin 1.6.5'. The description for this plugin is 'NodeJS Plugin executes NodeJS script as a build step.' with a link to 'Report an issue with this plugin'. The 'Enabled' column for this plugin shows a blue toggle switch that is turned on, with a checkmark icon and a red 'X' icon next to it.

Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Search: nodejs

48 Updates

Available plugins

Installed plugins

Advanced settings

Name	Enabled
NodeJS Plugin 1.6.5 NodeJS Plugin executes NodeJS script as a build step. Report an issue with this plugin	<input checked="" type="checkbox"/>

14. Configure **Node 18** in Manage Jenkins → Tools → NodeJS installations.

Dashboard > Manage Jenkins > Tools

NodeJS installations

NodeJS installations ^ Edited

Add NodeJS

≡ NodeJS

Name

Node 18

☒ Install automatically ?

≡ Install from nodejs.org

Version

NodeJS 18.0.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72


Save

Apply

15. Configure credentials for GitLab and EC2:

GitLab: Username + Personal Access Token

EC2: SSH Username (ec2-user) + Private Key from key pair





 Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

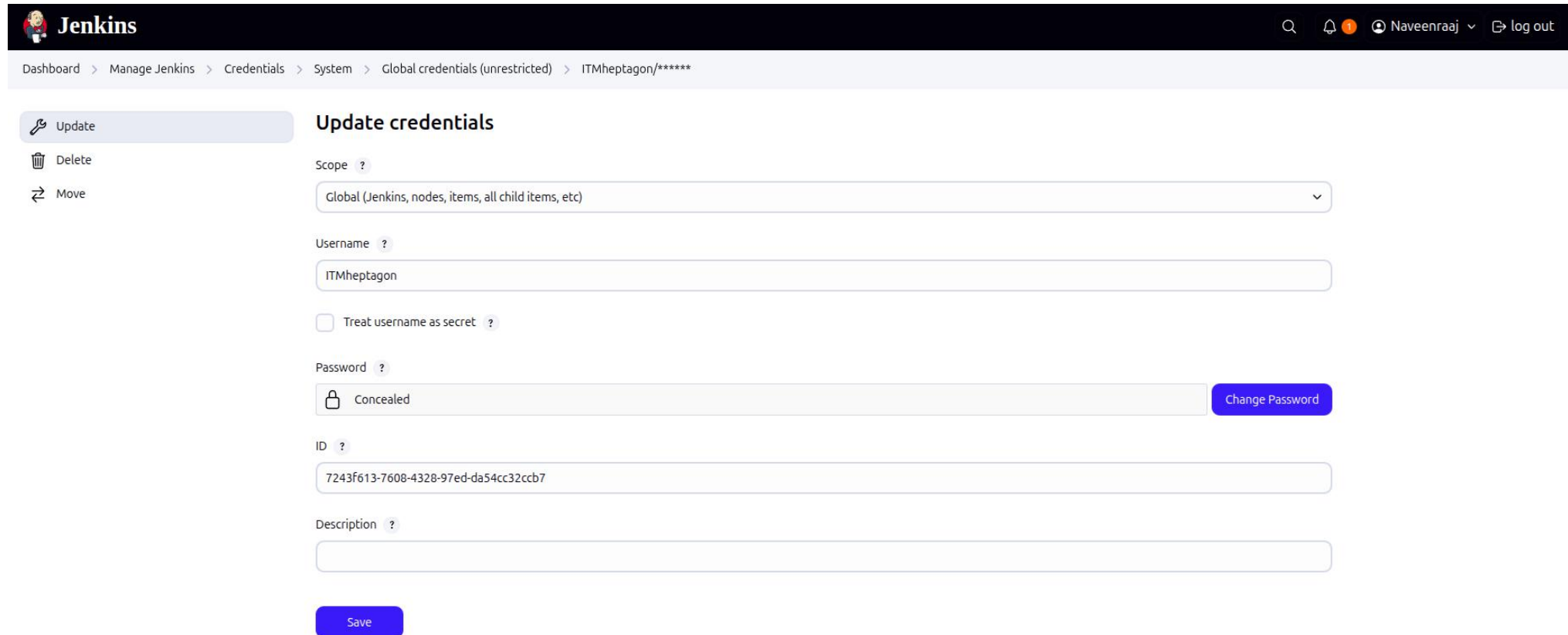
+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 7243f613-7608-4328-97ed-da54cc32ccb7	ITMheptagon/*****	Username with password	
 cc32cf60-9256-446b-b6a7-a7accb97c597	ec2-user	SSH Username with private key	

Icon: S M **L**

16. For GitLab, Kind **username & password**, type your username and copy the **Personal Access Token** from the GitLab and paste it in the password and save it.



The image shows the Jenkins web interface for updating a credential. The header bar is dark blue with the Jenkins logo and name on the left, and search, notifications, user profile (Naveenraaj), and a log out button on the right. Below the header is a breadcrumb trail: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > ITMheptagon/*****. On the left side, there is a sidebar with three buttons: 'Update' (selected, with a key icon), 'Delete' (with a trash icon), and 'Move' (with a double arrow icon). The main content area is titled 'Update credentials'. It contains several form fields: 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)'; 'Username' with a text input field containing 'ITMheptagon'; a checkbox labeled 'Treat username as secret' which is unchecked; 'Password' with a text input field showing 'Concealed' and a lock icon, and a blue 'Change Password' button; 'ID' with a text input field containing '7243f613-7608-4328-97ed-da54cc32ccb7'; and 'Description' with an empty text input field. At the bottom of the form is a blue 'Save' button.

Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > ITMheptagon/*****

Update

Delete

Move

Update credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

ITMheptagon

☐ Treat username as secret ?

Password ?

Concealed

Change Password

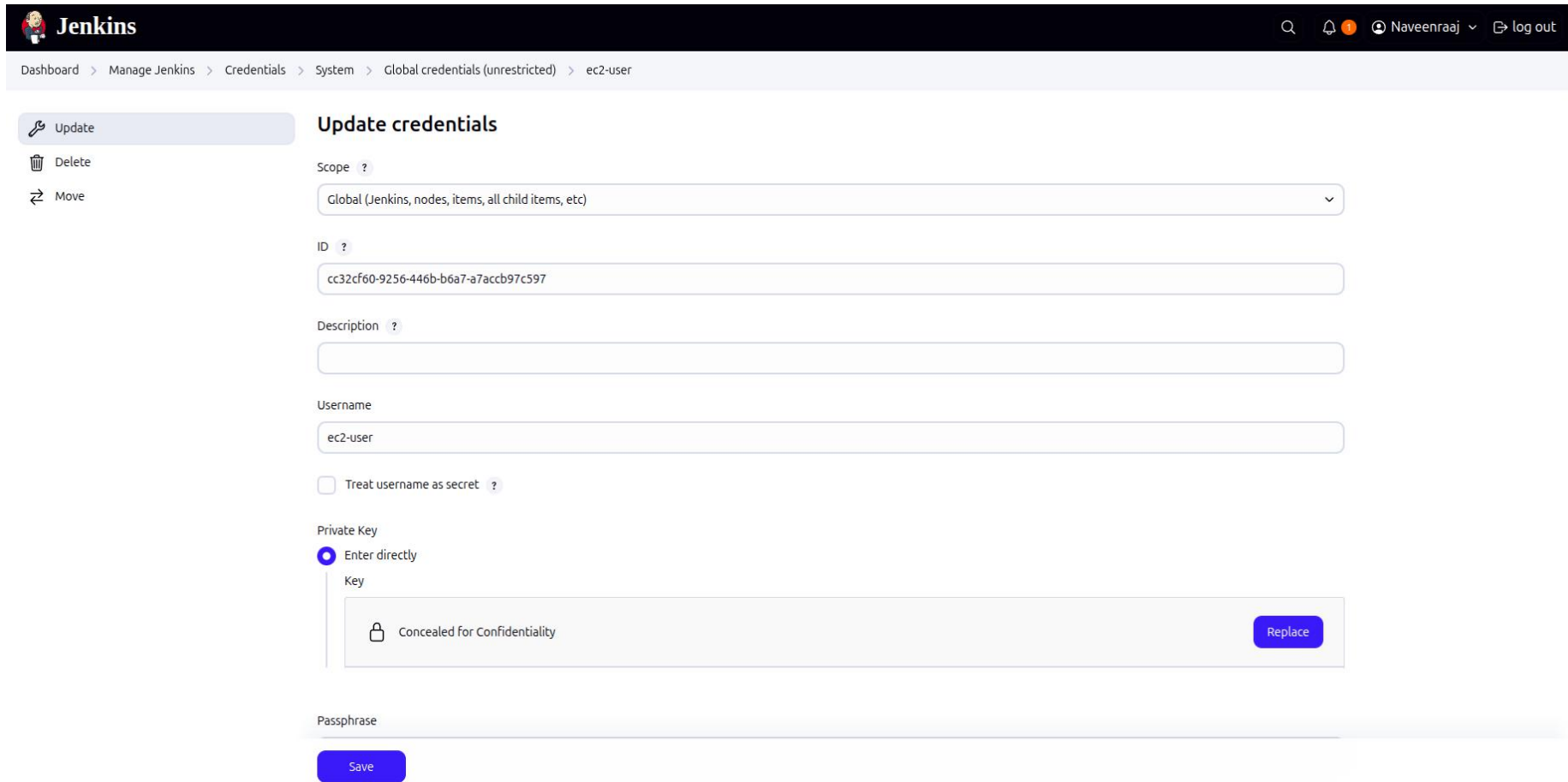
ID ?

7243f613-7608-4328-97ed-da54cc32ccb7

Description ?

Save

17. For EC2 server, select the kind as **SSH username with privatekey** and type the username as **ec2-user** then copy and paste the private key from the instance keypair.



The image shows the Jenkins 'Update credentials' page. The breadcrumb trail is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > ec2-user. On the left, there are three actions: 'Update' (selected), 'Delete', and 'Move'. The main form has the following fields: 'Scope' (dropdown menu set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'ID' (text input with value 'cc32cf60-9256-446b-b6a7-a7accb97c597'), 'Description' (empty text input), 'Username' (text input with value 'ec2-user'), and a checkbox 'Treat username as secret' which is unchecked. Under the 'Private Key' section, the 'Enter directly' radio button is selected. Below it, a 'Key' field is shown as a grey box with a lock icon and the text 'Concealed for Confidentiality', with a 'Replace' button to its right. At the bottom, there is a 'Passphrase' field and a 'Save' button.

Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > ec2-user

Update

Delete

Move

Update credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

cc32cf60-9256-446b-b6a7-a7accb97c597

Description ?

Username


ec2-user

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

 Concealed for Confidentiality

Replace

Passphrase

Save

18. Create a pipeline named **hello-jenkins-ec2**.

New Item

Enter an item name

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.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK

19. Configure the pipeline, choose **Pipeline script from SCM**, select **Git** as source code, paste the GitLab repo url **https://gitlab.com/ITMheptagon/jenkins.git** and select the GitLab credentials as saved in the global tool configuration.

Dashboard > hello-jenkins-ec2 > Configuration

Configure

- General
- Triggers
- Pipeline**
- Advanced

Pipeline
Define your Pipeline using Groovy directly or pull it from source control.

Definition
Pipeline script from SCM

SCM ?
Git

Repositories ?

Repository URL ?
https://gitlab.com/ITMheptagon/jenkins.git

Credentials ?
ITMheptagon/*****

+ Add

Advanced ▾

Add Repository

Branches to build ?

Save Apply

20. Specify the branch name as ***/Main** in the branch section. Click apply and save.

Dashboard > hello-jenkins-ec2 > Configuration

Configure

- General
- Triggers
- Pipeline**
- Advanced

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

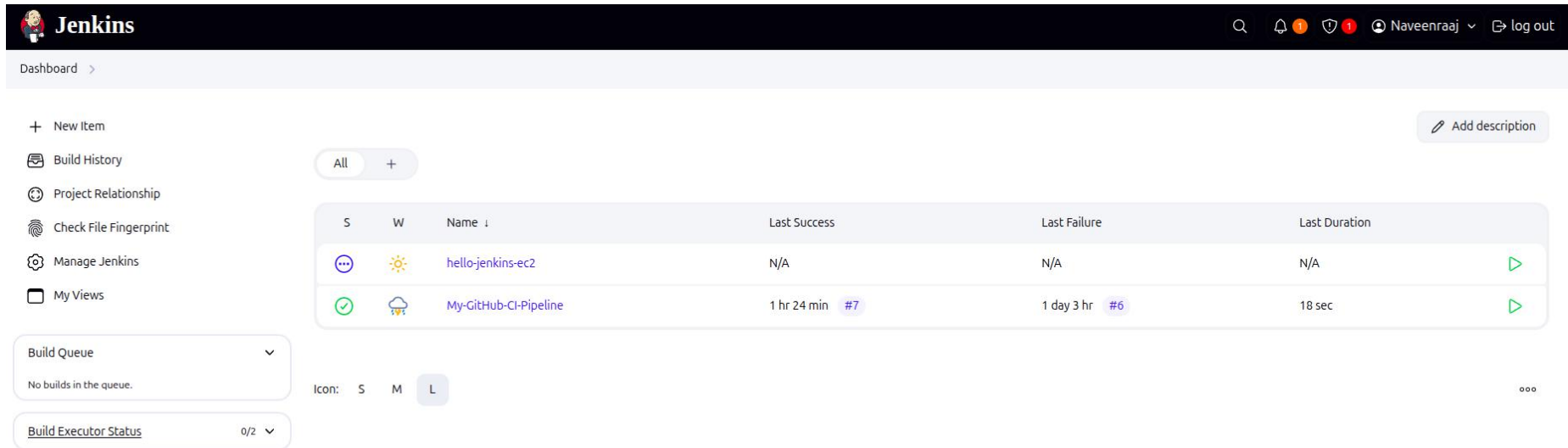
[Pipeline Syntax](#)

Advanced

Advanced

Save Apply

21. Now you can see the created pipeline project in the dashboard.



The screenshot shows the Jenkins dashboard interface. At the top, there's a dark header with the Jenkins logo, a search icon, notification icons, and the user name 'Naveenraaj' with a 'log out' button. Below the header, a light blue bar contains 'Dashboard >'. The main content area features a sidebar on the left with links: '+ New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The central part displays a table of pipeline projects. Above the table, there's a filter button 'All' and a '+ Add description' button. The table has columns: 'S' (Status), 'W' (Weather icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. Two projects are listed: 'hello-jenkins-ec2' with status 'N/A' and 'My-GitHub-CI-Pipeline' with status '1 hr 24 min #7'. At the bottom left, there are two summary boxes: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). At the bottom right, there's a footer with 'REST API' and 'Jenkins 2.504.3'.

S	W	Name	Last Success	Last Failure	Last Duration
...	☀️	hello-jenkins-ec2	N/A	N/A	N/A
✓	☁️	My-GitHub-CI-Pipeline	1 hr 24 min #7	1 day 3 hr #6	18 sec

22. Build the pipeline and monitor the console output for errors.

The screenshot shows the Jenkins web interface. At the top is a dark navigation bar with the Jenkins logo, the name 'Jenkins', and user information 'Naveenraaj' with a 'log out' button. Below this is a breadcrumb trail: 'Dashboard > hello-jenkins-ec2 >'. The main content area is divided into two columns. The left column contains a sidebar with the following items: 'Status' (selected), 'Changes', 'Build Now' (with a play button icon), 'Configure' (with a gear icon), 'Delete Pipeline' (with a trash icon), 'Stages' (with a stack icon), 'Rename' (with a pencil icon), and 'Pipeline Syntax' (with a question mark icon). The right column has the title 'hello-jenkins-ec2' and an 'Add description' button. Below the title is a 'Permalinks' section. At the bottom of the left sidebar is a 'Builds' section. It shows 'No builds' and a list of builds under the heading 'Today'. The first build is '#1' at '2:12 PM', with a status icon showing a blue bar and a red 'X'.

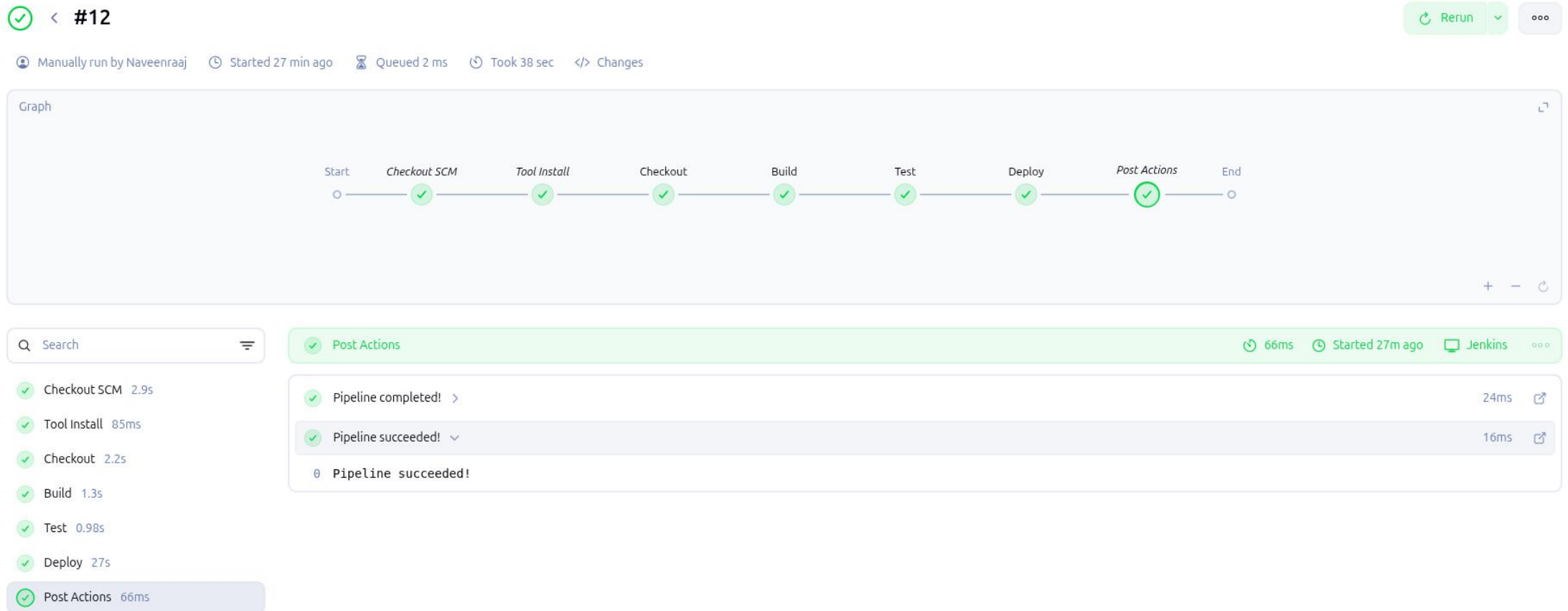
23. You can lookout for any errors in the console output

✓ Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Naveenraaj
Obtained Jenkinsfile from git https://gitlab.com/ITMheptagon/jenkins.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/hello-jenkins-ec2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential 7243f613-7608-4328-97ed-da54cc32ccb7
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/hello-jenkins-ec2/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://gitlab.com/ITMheptagon/jenkins.git # timeout=10
Fetching upstream changes from https://gitlab.com/ITMheptagon/jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
using GIT ASKPASS to set credentials
> git fetch --tags --force --progress -- https://gitlab.com/ITMheptagon/jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 81f4ded6a5472cd8a57951da4841189afa35a9b0 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 81f4ded6a5472cd8a57951da4841189afa35a9b0 # timeout=10
Commit message: "Edit Jenkinsfile"
> git rev-list --no-walk 3c30edf0366df9743ed95f321a659279a8855357 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
```

24. This is the pipeline overview



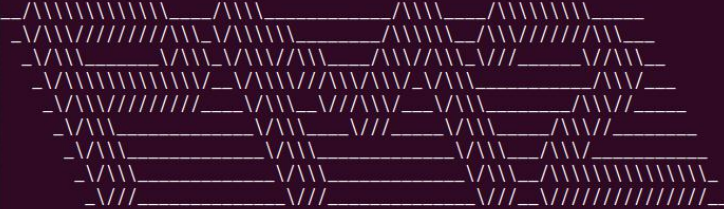
25. Log in to the EC2 server and verify the files are deployed.

```
naveen@5CD20890KH:~/Downloads$ sudo ssh -i "new_keypair.pem" ec2-user@13.201.71.211
[sudo] password for naveen:
,      #_
~\    #####_      Amazon Linux 2023
~~   \_#####\
~~      \###|
~~          \#/_____ https://aws.amazon.com/linux/amazon-linux-2023
~~          V~' '->
~~~~
~~. _ .
  _/ _/
   _/m/'
Last login: Wed Aug 13 08:18:49 2025 from 106.51.48.153
[ec2-user@ip-172-31-47-138 ~]$ ls
jenkins
```

```
[ec2-user@ip-172-31-47-138 ~]$ cd jenkins/
[ec2-user@ip-172-31-47-138 jenkins]$ ls
index.js  package.json
[ec2-user@ip-172-31-47-138 jenkins]$ nano index.js
[ec2-user@ip-172-31-47-138 jenkins]$ nano package.json
[ec2-user@ip-172-31-47-138 jenkins]$ ls
index.js  package.json
```

26. Check if the Node.js process is running **pm2 list**

```
[ec2-user@ip-172-31-47-138 ~]$ pm2 list
```



Runtime Edition

PM2 is a Production Process Manager for Node.js applications
with a built-in Load Balancer.

Start and Daemonize any application:
\$ pm2 start app.js

Load Balance 4 instances of api.js:
\$ pm2 start api.js -i 4

Monitor in production:
\$ pm2 monitor

Make pm2 auto-boot at server restart:
\$ pm2 startup

To go further checkout:
<http://pm2.io/>

```
[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2
[PM2] PM2 Successfully daemonized
```

id	name	namespace	version	mode	pid	uptime	🔄	status	cpu	mem	user	watching
----	------	-----------	---------	------	-----	--------	---	--------	-----	-----	------	----------

27. Start the process manually if needed

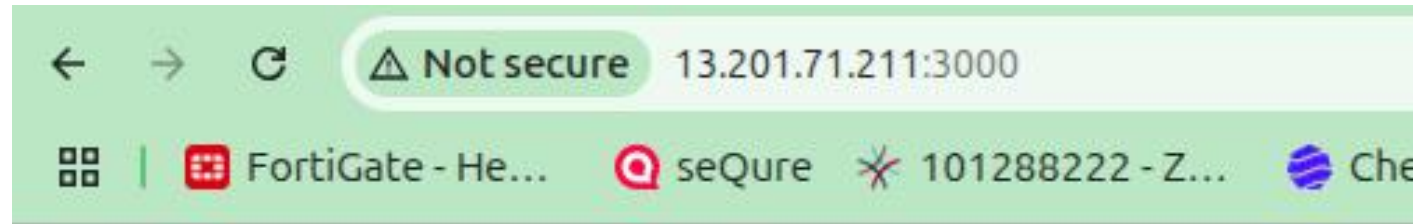
pm2 start /home/ec2-user/jenkins/index.js --name hello-jenkins

```
[ec2-user@ip-172-31-47-138 ~]$ pm2 start /home/ec2-user/jenkins/index.js --name hello-jenkins  
[PM2] Starting /home/ec2-user/jenkins/index.js in fork_mode (1 instance)  
[PM2] Done.
```

id	name	namespace	version	mode	pid	uptime	🔄	status	cpu	mem	user	watching
0	hello-jenkins	default	1.0.0	fork	32757	0s	0	online	0%	32.6mb	ec2-user	disabled

```
[ec2-user@ip-172-31-47-138 ~]$
```

28. Access the Node.js app **http://13.201.71.211:3000**



Hello, Jenkins!

Thank You