

jenkinsv4.md

Introduction to Jenkins

Jenkins is an open-source automation server that helps developers build, test, and deploy their applications efficiently. It enables Continuous Integration (CI) and Continuous Deployment (CD), automating the software development lifecycle.

Why Jenkins?

- **Automation:** Reduces manual effort in building, testing, and deploying applications.
- **Scalability:** Supports distributed builds across multiple machines.
- **Plugins:** A rich ecosystem of plugins extends Jenkins' functionality.
- **Easy Integration:** Works well with GitHub, Docker, Kubernetes, and cloud services.

How Jenkins Works

Jenkins follows a **pipeline-based** approach:

1. **Developers push code** to a repository (e.g., GitHub, GitLab).
 2. **Jenkins pulls the code**, builds it, and runs tests.
 3. **If successful**, Jenkins deploys the application to a staging or production environment.
-
-

Deploying Jenkins on Kubernetes

To deploy Jenkins on a Kubernetes cluster, follow these steps:

Prerequisites

- A running Kubernetes cluster
- Helm installed
- kubectl configured to interact with the cluster

Step 1: Add Jenkins Helm Repository

```
helm repo add jenkins https://charts.jenkins.io  
helm repo update
```

(Screenshot: Adding Helm repository)

```
devops@ubuntu:~$ helm repo add jenkins https://charts.jenkins.io  
helm repo update  
"jenkins" already exists with the same configuration, skipping
```

Step 2: Create a Namespace for Jenkins

```
kubectl create namespace jenkins
```

(Screenshot: Creating Kubernetes namespace)

```
devops@ubuntu:~$ kubectl create namespace jenkins  
namespace/jenkins created
```

Step 3: Install Jenkins Using Helm

```
helm install jenkins jenkins/jenkins --namespace jenkins --set controller.serviceType=NodePort
```

(Screenshot: Installing Jenkins with Helm)

```
devops@ubuntu:~$ helm install jenkins jenkins/jenkins --namespace jenkins --set controller.serviceType=NodePort
NAME: jenkins
LAST DEPLOYED: Tue Mar 11 19:44:04 2025
NAMESPACE: jenkins
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:
  kubectl exec --namespace jenkins -it svc/jenkins -c jenkins -- /bin/cat /run/secrets/additional/chart-admin-password && echo
2. Get the Jenkins URL to visit by running these commands in the same shell:
  export NODE_PORT=$(kubectl get --namespace jenkins -o jsonpath="{.spec.ports[0].nodePort}" services jenkins)
  export NODE_IP=$(kubectl get nodes --namespace jenkins -o jsonpath=".items[0].status.addresses[0].address")
  echo http://$NODE_IP:$NODE_PORT

3. Login with the password from step 1 and the username: admin
4. Configure security realm and authorization strategy
5. Use Jenkins Configuration as Code by specifying configScripts in your values.yaml file, see documentation: http://$NODE_IP:$NODE_PORT/configuration-as-code and examples: https://github.com/jenkinsci/configuration-as-code-plugin/tree/master/demos

For more information on running Jenkins on Kubernetes, visit:
https://cloud.google.com/solutions/jenkins-on-container-engine

For more information about Jenkins Configuration as Code, visit:
https://jenkins.io/projects/jcasc/

NOTE: Consider using a custom image with pre-installed plugins
```

Step 4: Access Jenkins

Find the Jenkins service port:

```
kubectl get svc -n jenkins
```

(Screenshot: Checking Jenkins service details)

```
devops@ubuntu:~$ kubectl get svc -n jenkins
NAME         TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
jenkins     NodePort   10.43.44.137    <none>           8080:31126/TCP  2m55s
jenkins-agent ClusterIP  10.43.147.146  <none>           50000/TCP       2m55s
```

You should see an output like this:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
jenkins	NodePort	10.43.150.139	<none>	8080:32000/TCP	5m

Step 5: Get Admin Password

```
kubectl get secret --namespace jenkins jenkins -o jsonpath=".data.jenkins-admin-password" | base64 --decode
```

(Screenshot: Retrieving admin password)

```
devops@ubuntu:~$ kubectl get secret --namespace jenkins jenkins -o jsonpath=".data.jenkins-admin-password" | base64 --decode  
Fi2r9ozCoMORE6B0wrnHdEdevops@ubuntu:~$
```

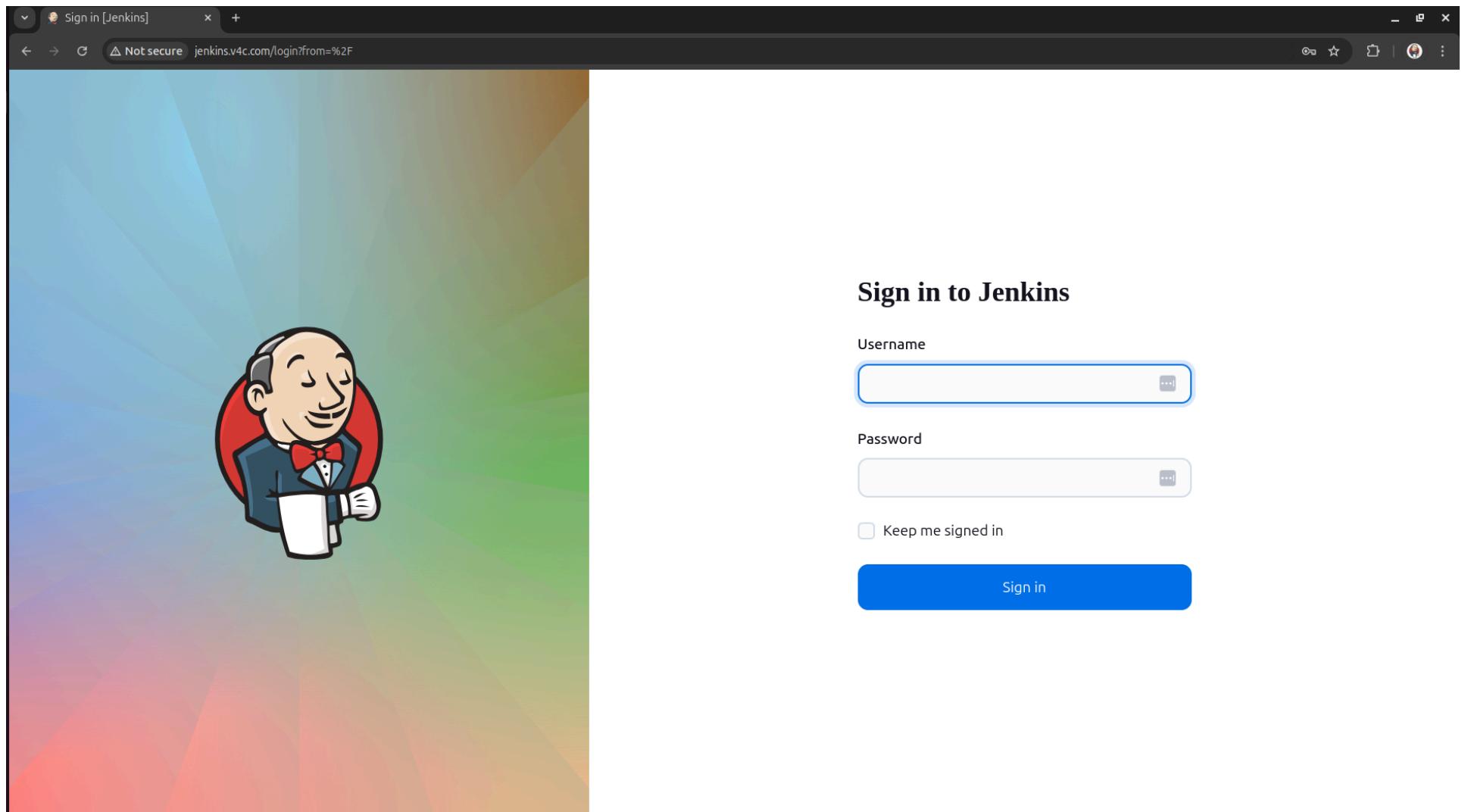
Log in to Jenkins as a Admin User

1. Open your web browser and go to your Jenkins URL.

```
http://<your-node-ip>:32000
```

2. Enter your **Admin Username** and **Password**, then click **Sign in**.

The default username is usually **admin**.



Add Plugins in Jenkins

Jenkins allows you to **extend its functionality** using plugins. Adding plugins can help with **CI/CD, integrations, reporting, notifications, security, and more.**

Installing Required Plugins for a Maven Project in Jenkins

We are deploying a **Maven application** and require three essential plugins:

1. GitHub Plugin

- Enables **integration between Jenkins and GitHub**.
- Supports **automated builds** triggered by:
 - Repository changes
 - Pull requests
 - Webhook events.

2. Maven Plugin

- Allows **building, testing, and packaging** Maven-based projects in Jenkins.
- Ensures smooth execution of the **Maven lifecycle** (`clean`, `package`, `test`, `install`).

3. Matrix Authorization Strategy Plugin

- Enables **fine-grained access control** in Jenkins.
- Lets administrators define **user-specific permissions** for jobs, builds, and configurations.
- Useful for restricting access based on **roles** (e.g., Developer, DevOps).

Here's a simple step-by-step guide for install plugins:

Go to Manage Plugins

In the left-hand menu, click "**Manage Jenkins**".

Not secure jenkins.v4c.com/manage/ Jenkins Admin log out

Jenkins

Dashboard > Manage Jenkins

+ New Item Build History Manage Jenkins My Views

Manage Jenkins

It appears that your reverse proxy set up is broken.

New version of Jenkins (2.492.2) is available for download (changelog).

Build Queue ▾ No builds in the queue.

Build Executor Status 0/0

Configure which of these warnings are shown

Warnings have been published for the following currently installed components:

Jenkins 2.492.1 core and libraries:
Multiple security vulnerabilities in Jenkins 2.499 and earlier, LTS 2.492.1 and earlier
A fix for this issue is available. Update Jenkins now.

System Configuration



System
Configure global settings and paths.



Tools
Configure tools, their locations and automatic installers.



Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Configuration as Code
Reload your configuration or update configuration source



Clouds
Add, remove, and configure cloud instances to provision



Appearance
Configure the look and feel of Jenkins

Scroll down and click "Manage Plugins".

The screenshot shows the Jenkins Manage Jenkins dashboard. The top navigation bar includes links for Dashboard, Manage Jenkins, Build Executor Status (0/0), and a Not secure warning. The main content area is titled "System Configuration" and contains several management sections:

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. This section is highlighted with a red box.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Configuration as Code**: Reload your configuration or update configuration source.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.
- 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.
 - System Log**: System log captures output from `java.util.logging`.
 - Load Statistics**: Check your resource utilization and see if you
- About Jenkins**: See the version and license information.

Install a Plugin

Click the "**Available**" tab (this shows plugins you can install).

Use the **search bar** to find the plugin you want (e.g., "GitHub", "Maven", "Matrix Authorization Strategy Plugin").

Check the box next to the plugin name.

Available plugins - Plugin x +

Not secure jenkins.v4c.com/manage/pluginManager/available

Jenkins Admin log out

Plugins

Updates 1 Available plugins Installed plugins Advanced settings

Install Name ↴ Released

Maven Integration 3.25 Build Tools 1 mo 3 days ago
This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.

GitHub 1.42.0 External Site/Tool Integrations github 18 days ago
This plugin integrates GitHub to Jenkins.

Matrix Authorization Strategy 3.2.4 Security Authentication and User Management 1 mo 13 days ago
Offers matrix-based security authorization strategies (global and per-project).

REST API Jenkins 2.492.1

Click "Install without restart" (or "Download now and install after restart").

The screenshot shows the Jenkins plugin manager interface. The left sidebar has tabs for 'Updates' (1), 'Available plugins' (selected), 'Installed plugins', and 'Advanced settings'. The main area is titled 'Plugins' and has a search bar with 'docker' typed in. Below the search bar is a red warning box containing the text: '• SSL/TLS certificate validation globally and unconditionally disabled'. A list of plugins follows:

- CrowdStrike Security** 1.2.2 - Enables assessment of Docker container images for vulnerabilities by sending them to CrowdStrike's Falcon Image Assessment module, pulling the report of vulnerabilities found, and optionally blocking the build based on configured IA Policy. Last updated 3 mo 15 days ago.
- Rancher** 1.0.13 - Deployment docker - This plugin is a rancher plugin to deploy/upgrade service to rancher. Last updated 6 yr 2 mo ago.
- Panoptica Vulnerability Scanner** 49.v106d3c954452 - Enables scan docker images for vulnerabilities in Jenkins, push images to registries, and report results to the Panoptica server. Last updated 2 yr 5 mo ago.
- Container Image Link** 11.v2fee68a69c54 - This plugin allows to add a badge and a permalink in the ui with a customizable url that redirect the traffic to your desired docker image repository. Last updated 2 yr 10 mo ago.
- Wallarm Fast** 0.4.2 - pipeline Security docker - This plugin allows you to run Wallarm FAST security tests. Last updated 4 yr 5 mo ago.

Restart Jenkins (If Needed)

- Some plugins require a restart. If prompted, **click "Restart Jenkins when installation is complete"**.
- Alternatively, you can manually restart Jenkins on **Linux** by running: `sudo systemctl restart jenkins`

This will restart the Jenkins service and apply any newly installed plugins.

User Management in Jenkins

What Are We Doing in This Section?

In this section, we will **create two users** in Jenkins and assign them different roles using **Matrix Authorization Strategy**. This is important for **role-based access control (RBAC)**, ensuring that users have the right level of access based on their responsibilities.

By the end of this section, we will have:

- A Developer (`dev_user`) who can **view and trigger builds** but **cannot modify configurations**.
- A DevOps Engineer (`deploy_user`) who can **configure jobs, approve deployments, and troubleshoot issues**.

Why Is This Important?

- **Security:** Prevents unauthorized changes to Jenkins jobs.
- **Accountability:** Ensures each action is logged under the correct user.
- **Controlled CI/CD Workflow:** Separates development work from deployment management.

Create Two Users

Navigate to User Management

click on "Manage Jenkins" from the left sidebar.

The screenshot shows the Jenkins Manage Jenkins page. On the left sidebar, there is a red box around the "Manage Jenkins" link. The main content area has a red box around the title "Manage Jenkins". Inside this box, there is a message: "It appears that your reverse proxy set up is broken." Below it, a blue box contains the text: "New version of Jenkins (2.492.2) is available for download (changelog)." Another red box surrounds a section titled "Warnings have been published for the following currently installed components:" which lists "Jenkins 2.492.1 core and libraries" and "Multiple security vulnerabilities in Jenkins 2.499 and earlier, LTS 2.492.1 and earlier". A blue button "Configure which of these warnings are shown" is to the right. At the bottom, there is a "System Configuration" section with several icons and links:

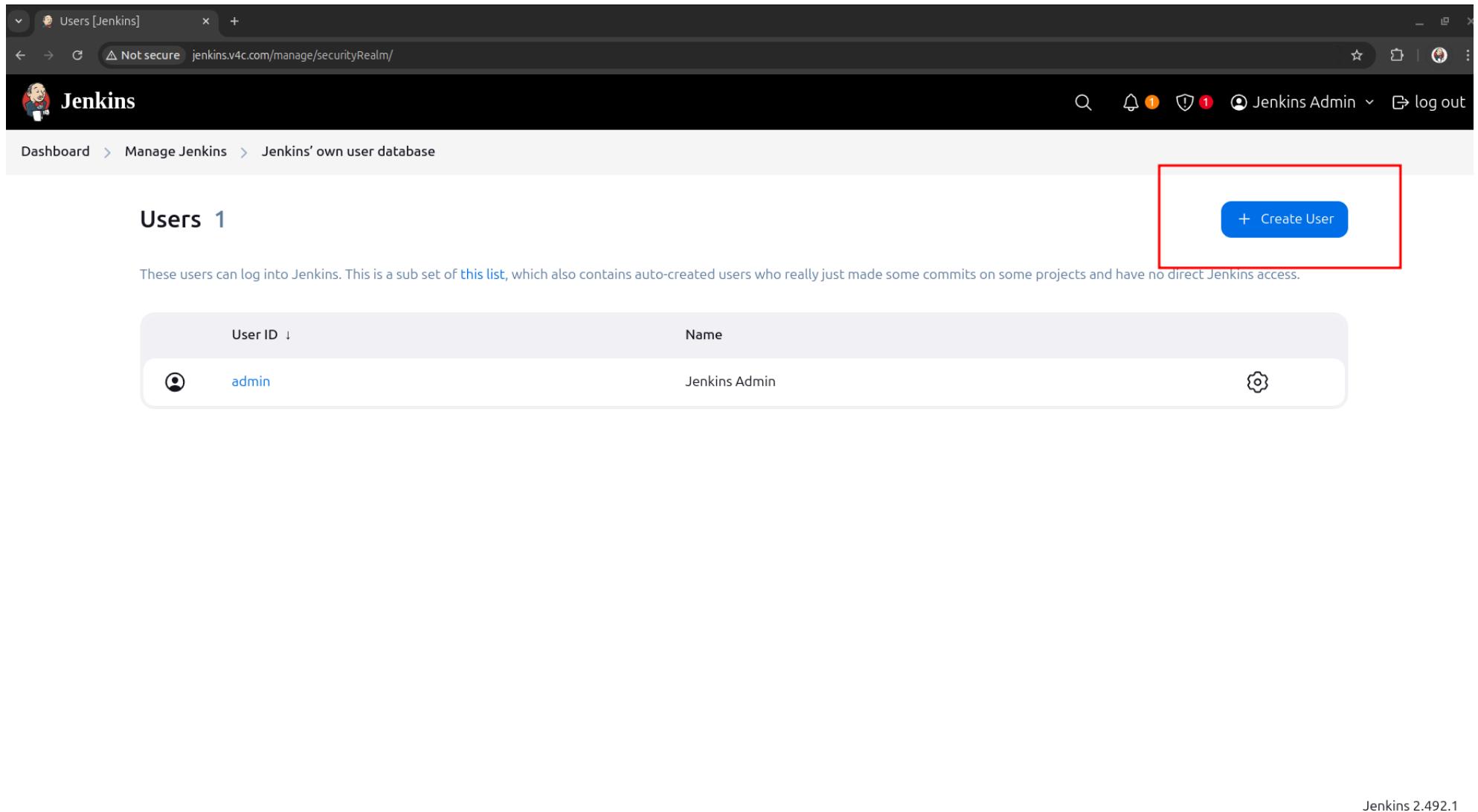
- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (1 new item)
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Configuration as Code**: Reload your configuration or update configuration source.
- Clouds**: Add, remove, and configure cloud instances to provision.
- Appearance**: Configure the look and feel of Jenkins.

Scroll down and click on "Manage Users" under the **Security** section.

The screenshot shows the Jenkins Manage Jenkins dashboard. At the top, there is a navigation bar with links for Dashboard, Manage Jenkins, and other system management options. Below the navigation bar, there are several management sections:

- Configuration as Code**: Reload your configuration or update configuration source.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.
- 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. This item is highlighted with a red rectangular box.
- 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**:
 - Manage Old Data**: Scrub configuration files to -----

Click on "Create User" on the left side.



The screenshot shows the Jenkins 'Users' page. At the top, there's a navigation bar with a 'Users [Jenkins]' tab, a 'Not secure' warning, and various Jenkins status icons. Below the navigation is the Jenkins logo and a search bar. The main content area has a breadcrumb trail: 'Dashboard > Manage Jenkins > Jenkins' own user database'. The title 'Users 1' is displayed above a table. The table has two columns: 'User ID' (sorted) and 'Name'. One row is shown, for a user named 'admin' with the name 'Jenkins Admin'. To the right of the table is a blue button labeled '+ Create User' with a red box drawn around it. Below the table, a note says: 'These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.' At the bottom right, the text 'Jenkins 2.492.1' is visible.

User ID ↓	Name
admin	Jenkins Admin

+ Create User

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

Jenkins 2.492.1

Create Developer User

1. In the **Manage Jenkins** section, navigate to **Manage Users**.
2. Click **Create User** on the left panel.

3. Fill in the following details:

- **Username:** dev_user
- **Password:** Set a secure password (e.g., mypassword123).
- **Confirm Password:** Re-enter the same password.
- **Full Name:** Developer User
- **Email Address:** dev@example.com

The screenshot shows the Jenkins 'Create User' interface. The page title is 'Create User'. It contains five input fields: 'Username' (dev_user), 'Password' (represented by a series of dots), 'Confirm password' (also represented by a series of dots), 'Full name' (Developer User), and 'E-mail address' (dev@example.com). A red box highlights the entire row of these five input fields. At the bottom of the form is a blue 'Create User' button.

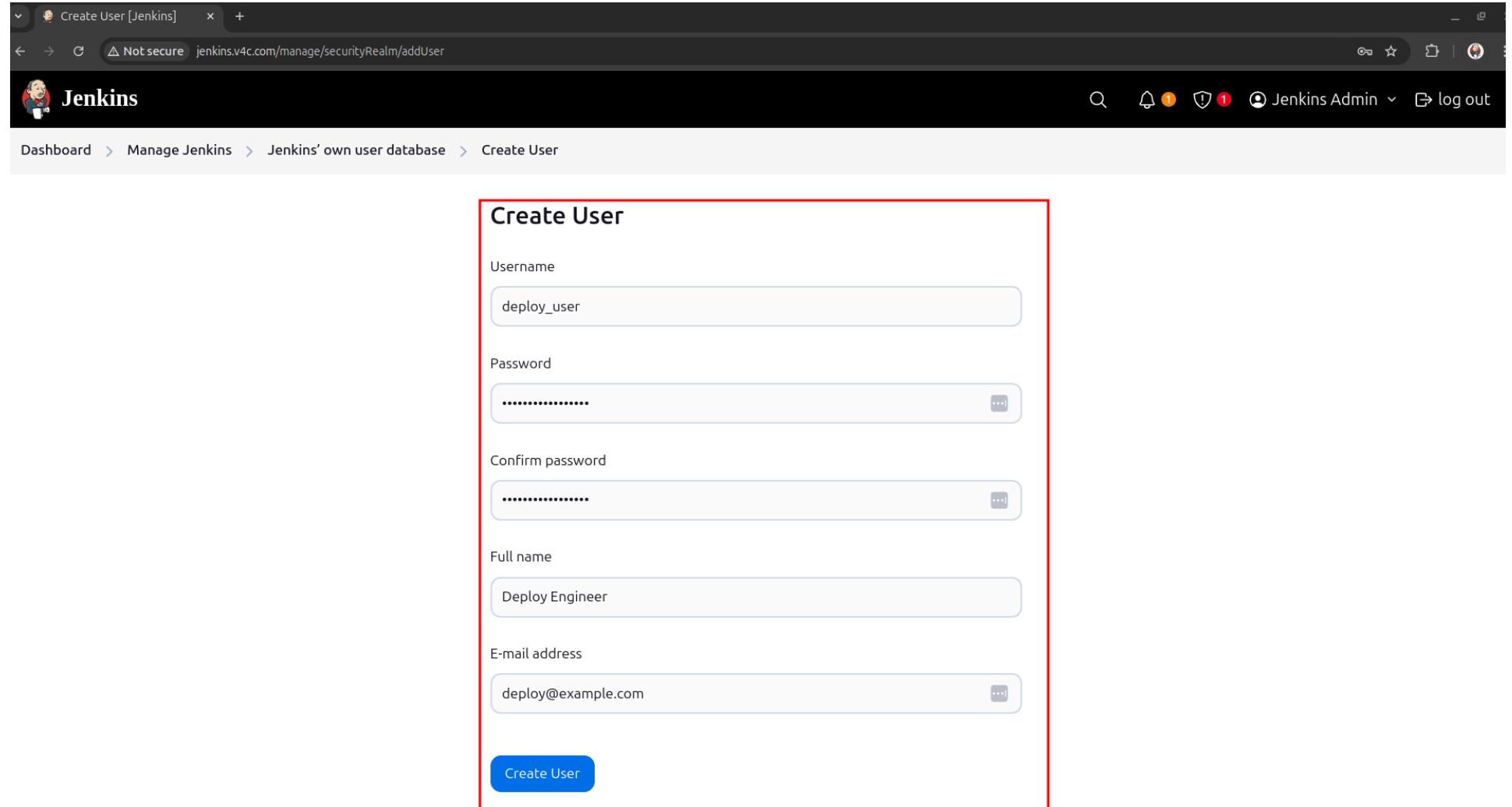
4. Click **Create User** to save the new user.

This user will have **limited permissions**, allowing them only to **view and trigger builds** but not modify configurations.

Create DevOps Engineer User

1. Go to **Manage Jenkins > Manage Users**.
2. Click **Create User**.
3. Enter the following details:
 - **Username:** deploy_user
 - **Password:** Set a secure password (e.g., deploypassword123).
 - **Confirm Password:** Re-enter the password.
 - **Full Name:** Deploy Engineer

- Email Address: deploy@example.com



The screenshot shows the Jenkins 'Create User' page. The URL in the browser is `jenkins.v4c.com/manage/securityRealm/addUser`. The page title is 'Create User'. The form fields are highlighted with a red border:

- Username: deploy_user
- Password: (redacted)
- Confirm password: (redacted)
- Full name: Deploy Engineer
- E-mail address: deploy@example.com

A blue 'Create User' button is at the bottom of the form.

4. Click **Create User** to save the new user.

This user will have **more privileges**, including the ability to **configure jobs, approve deployments, and troubleshoot issues**.

Now that both users are created, we will proceed to assign them **role-based access** using **Matrix Authorization Strategy**.

Configure User Permissions

Now that we have installed the required **plugins** and created **two users** (`dev_user` and `deploy_user`), the next step is to **configure role-based access control** using the **Matrix Authorization Strategy Plugin**.

We will now:

- Enable **Matrix Authorization Strategy** in Jenkins.
- Assign **limited permissions** to `dev_user` (Developer).
- Assign **elevated permissions** to `deploy_user` (DevOps Engineer).

Step 1: Enable Matrix Authorization Strategy

1. Go to Jenkins Dashboard → Click Manage Jenkins.

The screenshot shows the Jenkins Manage Jenkins page. The left sidebar has a 'Manage Jenkins' button highlighted with a red box. The main content area includes a warning message about a reverse proxy setup being broken, a download notice for version 2.492.2, and a system configuration section with various management links.

Manage Jenkins

It appears that your reverse proxy set up is broken.

New version of Jenkins (2.492.2) is available for download ([changelog](#)).

Warnings have been published for the following currently installed components:

Jenkins 2.492.1 core and libraries:
[Multiple security vulnerabilities in Jenkins 2.499 and earlier, LTS 2.492.1 and earlier](#)
A fix for this issue is available. Update Jenkins now.

Configure which of these warnings are shown

System Configuration

- System**
Configure global settings and paths.
- Tools**
Configure tools, their locations and automatic installers.
- Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
1 new plugin
- Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Configuration as Code**
Reload your configuration or update configuration source
- Clouds**
Add, remove, and configure cloud instances to provision
- Appearance**
Configure the look and feel of Jenkins

2. Scroll down and click **Configure Global Security**.

The screenshot shows the Jenkins Manage Jenkins dashboard. At the top, there's a header bar with the title "Manage Jenkins [Jenkins]" and a URL "Not secure jenkins.v4c.com/manage/". Below the header, the page title is "Dashboard > Manage Jenkins". The main content area is divided into several sections:

- Configuration as Code**: Reload your configuration or update configuration source.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.
- Security** (highlighted with a red box): 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**
 - Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.

3. Under the **Authorization** section, select **Matrix-based security**.

The screenshot shows the Jenkins 'Configure Security' page under the 'Manage Jenkins > Security' section. The 'Jenkins' own user database' tab is selected. The 'Authorization' section is highlighted with a red box, and the 'Matrix-based security' tab is selected within it.

Authorization Matrix:

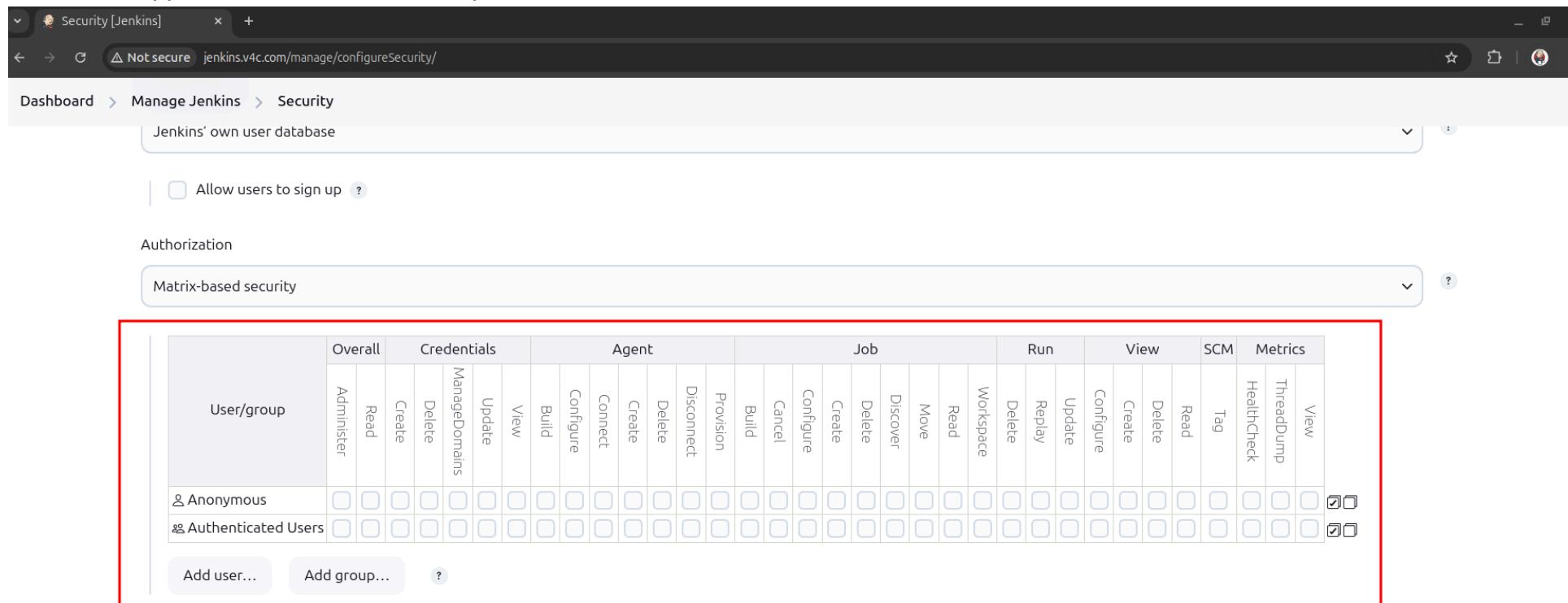
User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
Authenticated Users	<input type="checkbox"/>	<input checked="" type="checkbox"/>						

Markup Formatter:

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Save Apply

4. A table will appear where we can define permissions for each user.



The screenshot shows the Jenkins 'Configure Security' page under 'Manage Jenkins > Security'. It displays a 'Matrix-based security' configuration table. The columns represent users/groups (Anonymous, Authenticated Users) and the rows represent Jenkins features (View, ThreadDump, HealthCheck, Tag, Read, Delete, Create, Configure, Update, Replay, Delete, Workspace, Read, Move, Discover, Delete, Create, Configure, Cancel, Build, Provision, Disconnect, Delete, Create, Connect, Configure, Build, View, Update, ManageDomains, Delete, Create, Read, Administer). The 'Anonymous' row has checked checkboxes in the 'View', 'ThreadDump', 'HealthCheck', 'Tag', 'Read', 'Delete', 'Create', 'Configure', 'Update', 'Replay', 'Delete', 'Workspace', 'Move', 'Discover', 'Delete', 'Create', 'Configure', 'Cancel', 'Build', 'Provision', 'Disconnect', 'Delete', 'Create', 'Connect', 'Configure', 'Build', 'View', 'Update', and 'Administer' columns. The 'Authenticated Users' row has checked checkboxes in the 'View', 'ThreadDump', 'HealthCheck', 'Tag', 'Read', 'Delete', 'Create', 'Configure', 'Update', 'Replay', 'Delete', 'Workspace', 'Move', 'Discover', 'Delete', 'Create', 'Configure', 'Cancel', 'Build', 'Provision', 'Disconnect', 'Delete', 'Create', 'Connect', 'Configure', 'Build', 'View', 'Update', and 'Administer' columns. Buttons at the bottom include 'Add user...', 'Add group...', and a help icon.

Markup Formatter

Markup Formatter [?](#)

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

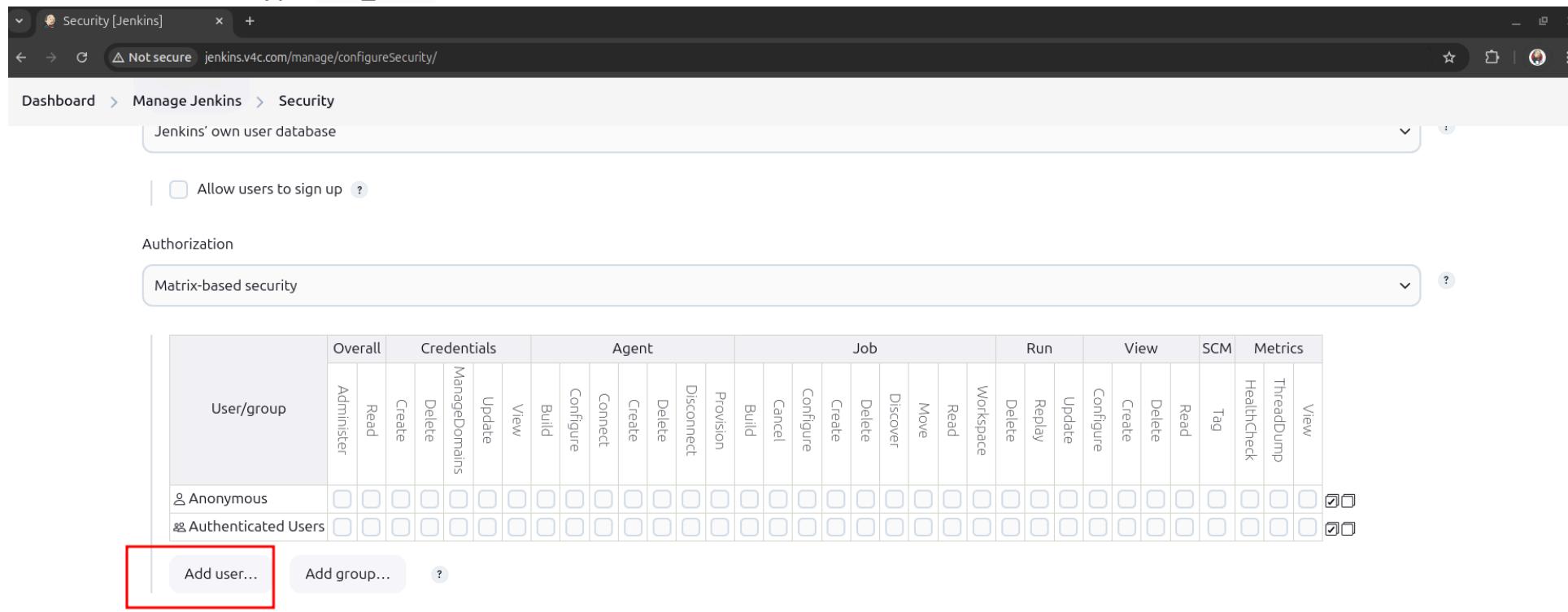
[Save](#)

[Apply](#)

Step 2: Assign Permissions

Assign Developer (dev_user) Permissions

- Click **Add user...** → Type `dev_user` → Click OK.



The screenshot shows the Jenkins Security configuration page. At the top, there's a header bar with the Jenkins logo and the URL `jenkins.v4c.com/manage/configureSecurity/`. Below the header, the breadcrumb navigation shows **Dashboard > Manage Jenkins > Security**. The main content area is titled "Jenkins' own user database". There's a checkbox labeled "Allow users to sign up". Under "Authorization", a section titled "Matrix-based security" is expanded. It contains a table where rows represent "User/group" (with entries for "Anonymous" and "Authenticated Users") and columns represent various Jenkins operations like "View", "Read", "Move", etc. The "Add user..." button is located at the bottom left of the matrix table, and it is highlighted with a red rectangular box. Other buttons visible include "Add group..." and a help icon.

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>							
Authenticated Users	<input type="checkbox"/>							

Add user...

Markup Formatter

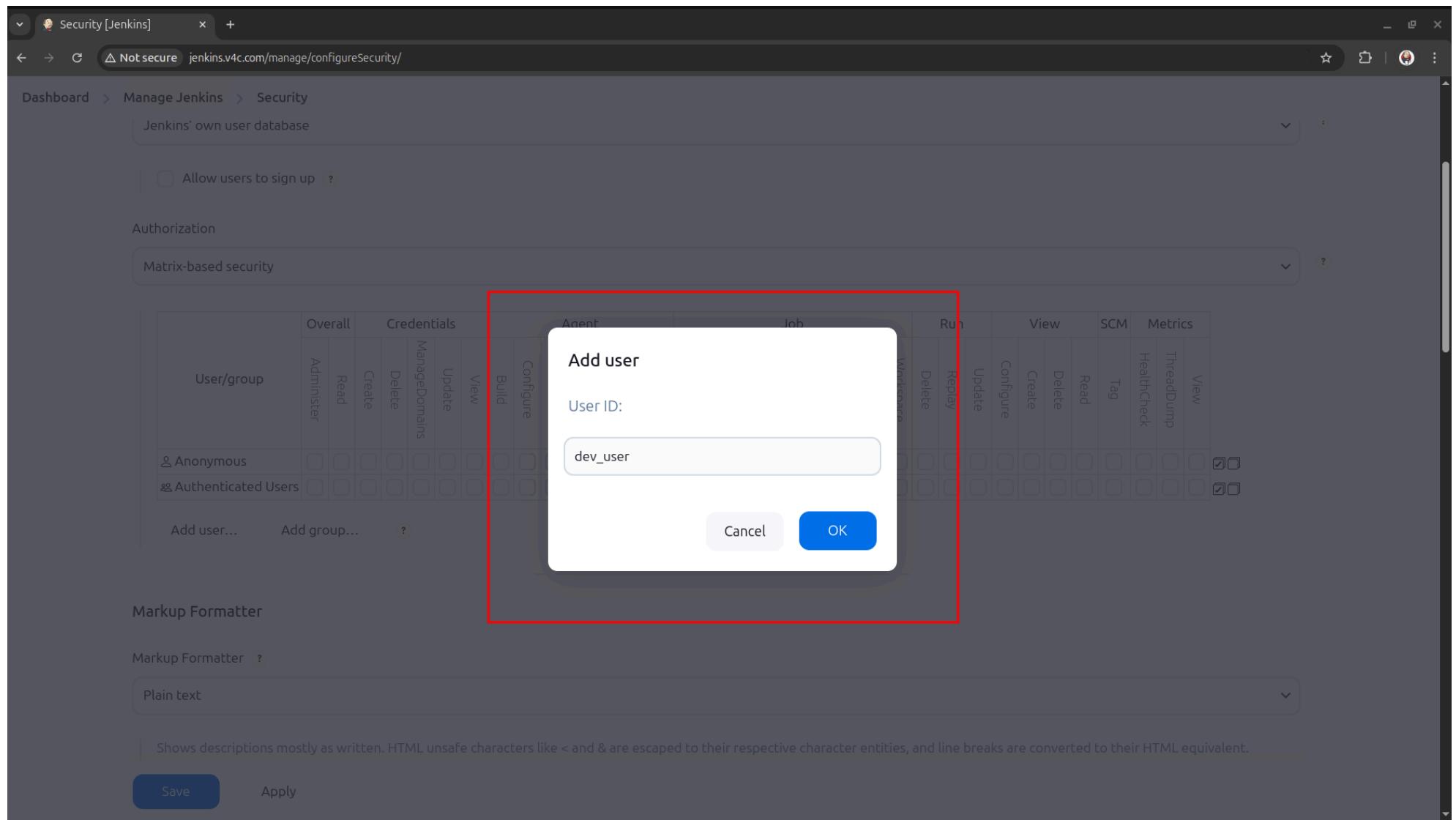
Markup Formatter [?](#)

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Save

Apply



- Assign **basic permissions**:
 - **Read**
 - **Job: Build**
 - **Job: Discover**
 - **Job: Read**

The screenshot shows the Jenkins 'Security' configuration page under 'Manage Jenkins > Security'. It displays a matrix-based security configuration for different user groups (Anonymous, Authenticated Users, Developer User) across various Jenkins features.

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> View
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> ThreadDump
Developer User	<input type="checkbox"/>	<input checked="" type="checkbox"/> Create	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Build	<input checked="" type="checkbox"/> Run	<input checked="" type="checkbox"/> View	<input checked="" type="checkbox"/> SCM

Below the matrix, there are buttons for 'Add user...', 'Add group...', and a help icon. The 'Developer User' row is highlighted with a red box.

Markup Formatter

Markup Formatter ?

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

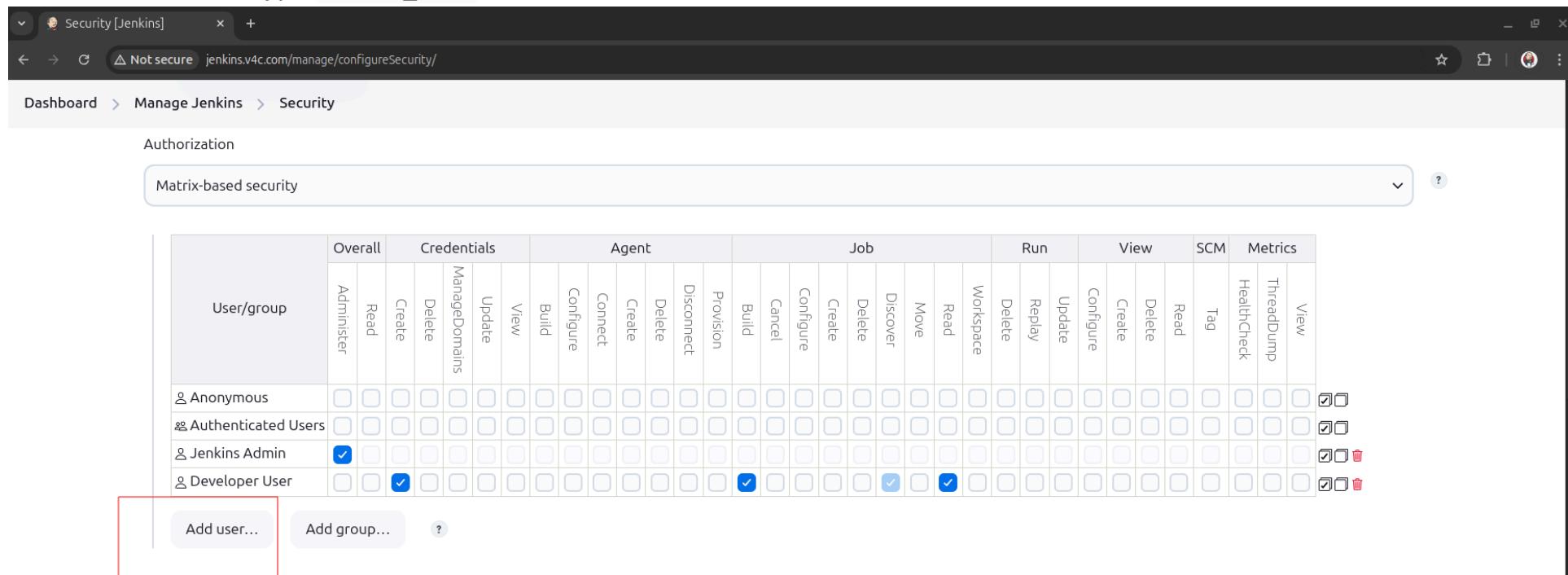
Agents

Save Apply

This ensures the **developer can trigger builds but cannot modify configurations**.

Assign DevOps Engineer (deploy_user) Permissions

- Click Add user... → Type deploy_user → Click OK.



The screenshot shows the Jenkins 'Matrix-based security' configuration page. At the top, there's a navigation bar with tabs: Dashboard, Manage Jenkins, and Security. Below that is a section titled 'Authorization' with a sub-section 'Matrix-based security'. The main area is a grid where rows represent 'User/group' (Anonymous, Authenticated Users, Jenkins Admin, Developer User) and columns represent various Jenkins operations (View, ThreadDump, HealthCheck, Tag, Read, Delete, Create, Configure, Update, Run, Repay, Workspace, Read, Move, Discover, Delete, Create, Configure, Cancel, Build, Provision, Disconnect, Delete, Create, Connect, Configure, Build, View, Update, ManageDomains, Delete, Create, Read, Administer). Most cells contain empty checkboxes. The 'Jenkins Admin' row has checked checkboxes in the 'Create' and 'Configure' columns under the 'Job' section. The 'Developer User' row has checked checkboxes in the 'Read' and 'Delete' columns under the 'Job' section. At the bottom left of the grid, there are two buttons: 'Add user...' (highlighted with a red box) and 'Add group...'. A question mark icon is also present.

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Admin	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Developer User	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Markup Formatter

Markup Formatter [?](#)

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Save

Apply

The screenshot shows the Jenkins 'Manage Jenkins > Security' page. A modal dialog titled 'Add user' is open, prompting for a 'User ID'. The input field contains 'deploy_user'. The dialog has 'Cancel' and 'OK' buttons. The background shows a matrix of checkboxes for granting permissions across various Jenkins features like View, SCM, Metrics, and Job management.

Authorization

Matrix-based security

User/group	Overall	Credentials	Agent	Job	Run	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Developer User	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

Add user... Add group... ?

Markup Formatter

Markup Formatter ?

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Save Apply

- Assign **advanced permissions:**
 - **Read**
 - **Job: Build, Configure, Delete**
 - **Job: Read, Create, Cancel**
 - **Administer (optional, if full access is needed)**

The screenshot shows the Jenkins 'Configure Security' page under 'Manage Jenkins'. It displays a matrix for 'Matrix-based security' where users and groups are assigned permissions across various Jenkins features.

Matrix-based security

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Developer User	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Deploy Engineer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add user... **Add group...** ?

Markup Formatter

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Plain text

Save **Apply**

This allows the **DevOps Engineer** to manage builds, edit jobs, and approve deployments.

Save & Apply Changes

1. Scroll down and click **Save**.

The screenshot shows the Jenkins 'Security' configuration page under 'Manage Jenkins > Security'. It displays a matrix-based security configuration for various user groups (Anonymous, Authenticated Users, Jenkins Admin, Developer User, Deploy Engineer) across different Jenkins features (Overall, Credentials, Agent, Job, Run, View, SCM, Metrics, Workspace, Provision, Build, Provision, Disconnect, Delete, Create, Configure, Connect, Configure, Build, View, Update, ManageDomains, Delete, Create, Read, Administer) and Jenkins operations (View, ThreadDump, HealthCheck, Tag, Read, Delete, Create, Configure, Update, Replay, Delete, Read, Move, Discover, Delete, Create, Configure, Cancel, Build, Provision, Disconnect, Delete, Create, Connect, Configure, Build, View, Update, ManageDomains, Delete, Create, Read, Administer). The 'Jenkins Admin' group has the most extensive permissions, while 'Anonymous' and 'Authenticated Users' have minimal permissions. The 'Developer User' and 'Deploy Engineer' groups also have specific permissions for certain Jenkins operations.

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Developer User	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Deploy Engineer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add user... Add group... ?

Markup Formatter

Markup Formatter ?

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Save **Apply**

2. Log out and test the permissions by logging in as `dev_user` and `deploy_user`.

Once this is done, we will proceed to **create a Jenkins pipeline for the Maven project**.

Create a Jenkins Pipeline for the Maven Project

We will now:

- Create a **new pipeline job** in Jenkins.
- Configure it to **fetch the Maven project from GitHub**.
- Set up **Maven build steps** to compile, test, and package the project.

Create a New Pipeline

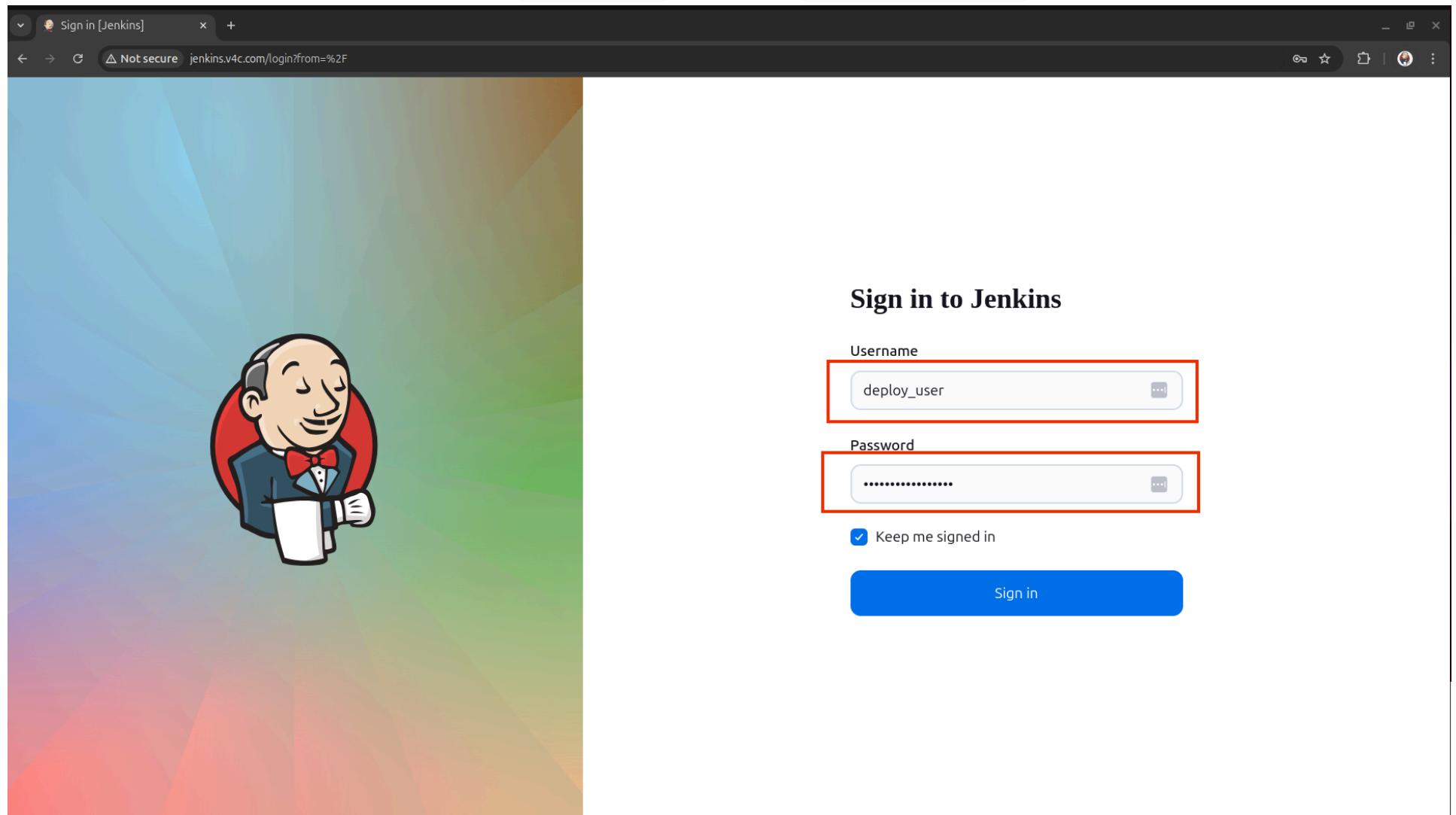
Now that we have installed the necessary plugins and created users, it's time to **set up a Jenkins Pipeline** to build and test the Maven project. Follow the steps below:

****Log in to Jenkins again**

1. Open a web browser and navigate to your Jenkins URL:

```
http://<your-jenkins-server>:8080
```

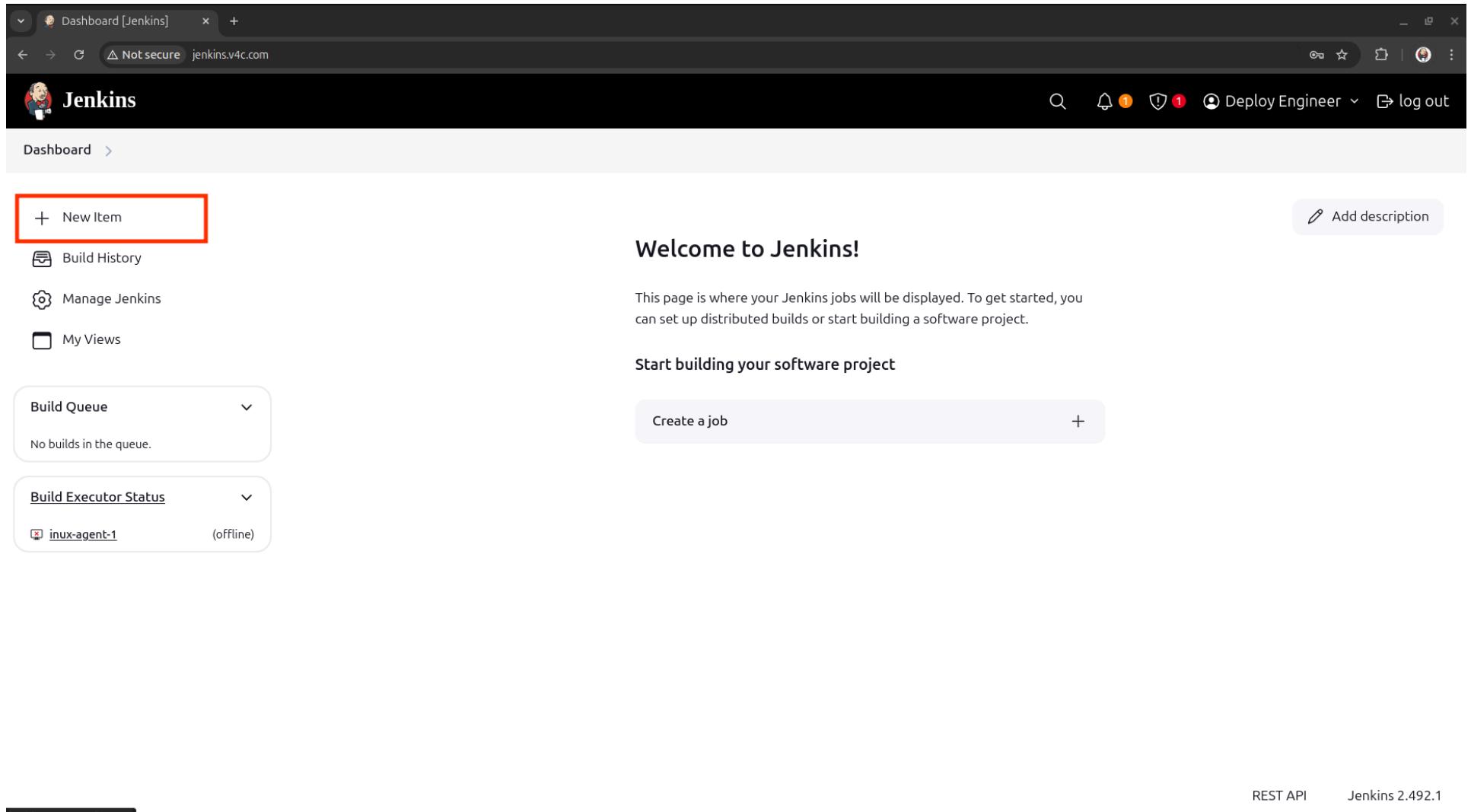
2. Enter the **deploy_user** credentials (Username: `deploy_user` , Password: `deploypassword123`).



3. Click **Sign in**.

Create a New Pipeline Job

1. After logging in, click on "**New Item**" from the Jenkins dashboard.



The screenshot shows the Jenkins dashboard. At the top left, there's a navigation bar with a back/forward button, a refresh icon, and the URL "jenkins.v4c.com". The title bar says "Dashboard [Jenkins]". On the right side of the title bar are icons for search, notifications (with 1 notification), security (with 1 issue), and user "Deploy Engineer". Below the title bar, there's a "log out" link. The main content area has a "Welcome to Jenkins!" message. To the left, there's a sidebar with links: "+ New Item" (which is highlighted with a red box), "Build History", "Manage Jenkins", and "My Views". Below the sidebar are two collapsed sections: "Build Queue" (showing "No builds in the queue.") and "Build Executor Status" (showing "linux-agent-1 (offline)"). In the center, there's a "Create a job" button with a "+" sign. At the bottom right, there are links for "REST API" and "Jenkins 2.492.1".

2. In the **Enter an item name** field, provide a descriptive name for the pipeline.

- Example: `maven-project-build`.

3. Under "Type", select **Pipeline** (since we are setting up a CI/CD pipeline for a Maven project).

The screenshot shows the Jenkins 'New Item' creation interface. In the 'Enter an item name' field, the text 'maven-project-build' is entered and highlighted with a red box. Below it, the 'Select an item type' section lists several options: 'Freestyle project', 'Maven project', 'Pipeline' (which is highlighted with a red box), 'Folder', and 'Multibranch Pipeline'. Each option has a brief description. At the bottom of the form, there is a blue 'OK' button, which is also highlighted with a red box.

4. Click **OK** to proceed.

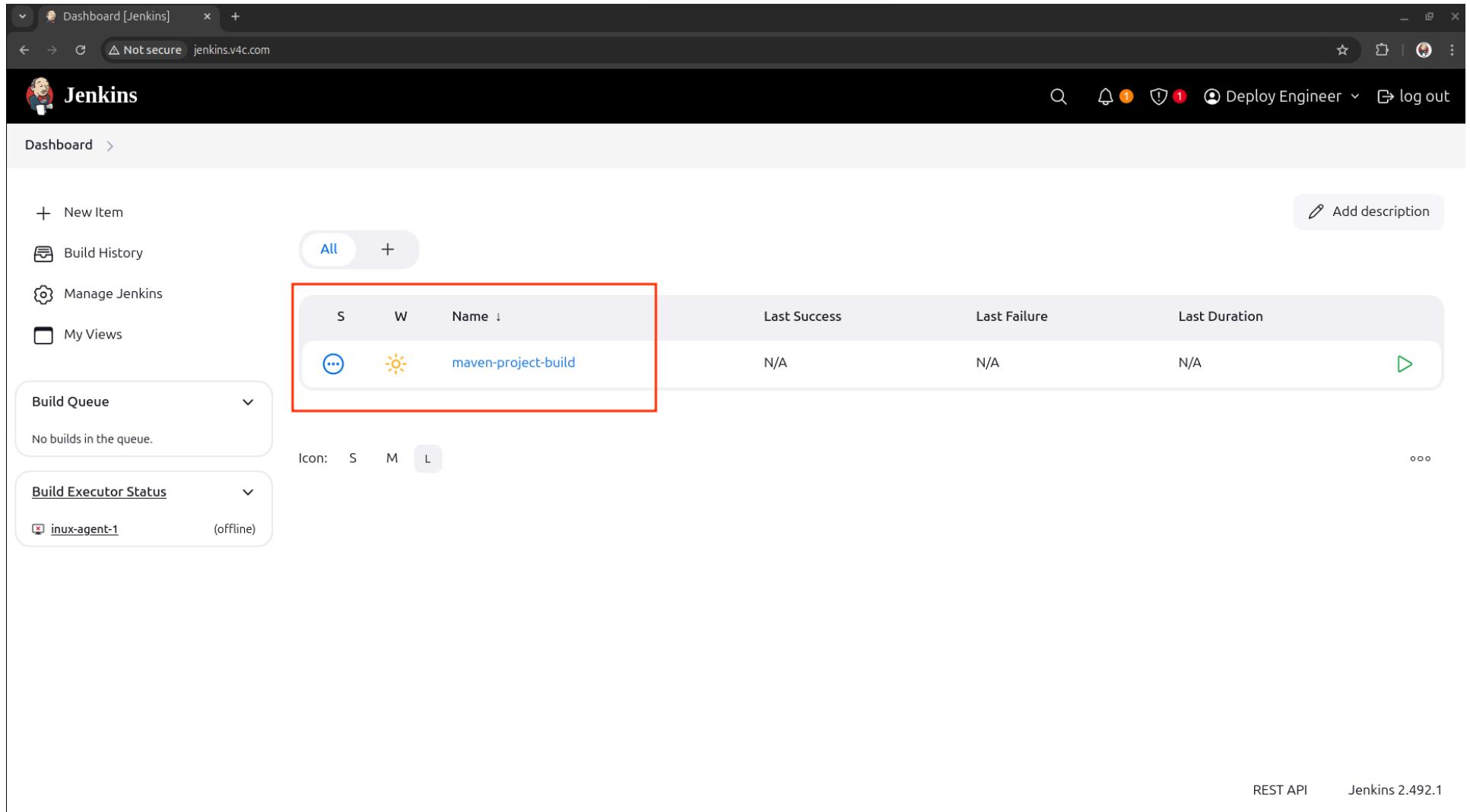
Link GitHub Repository to Jenkins Pipeline

To fetch the Maven project from GitHub, we need to configure **Source Code Management (SCM)** in Jenkins. This allows Jenkins to automatically pull the latest code before building the project.

Steps to Link GitHub Repository

Open the Pipeline Configuration Page

- From the Jenkins Dashboard, click on your Pipeline job (maven-project-build).



The screenshot shows the Jenkins Dashboard interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (one agent named 'inx-agent-1' listed as offline). The main area is titled 'All' and displays a table of pipeline jobs. A single row for 'maven-project-build' is selected and highlighted with a red box. The table columns are labeled 'S', 'W', 'Name ↓', 'Last Success', 'Last Failure', and 'Last Duration'. The 'maven-project-build' row shows an icon of three dots, a yellow sun icon, the name 'maven-project-build', and 'N/A' for all other columns. At the bottom right of the dashboard, there are links for 'REST API' and 'Jenkins 2.492.1'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
...	☀️	maven-project-build	N/A	N/A	N/A

REST API Jenkins 2.492.1

- Click **Configure** from the left sidebar.

The screenshot shows the Jenkins Pipeline configuration page for the job 'maven-project-build'. The left sidebar contains the following options:

- Status
- </> Changes
- ▷ Build Now
- Configure** (highlighted with a red box)
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

The main content area is titled 'Stage View' and displays the message: 'No data available. This Pipeline has not yet run.' Below this, there is a section titled 'Permalinks'.

At the bottom left, there is a 'Builds' section with the message 'No builds'.

At the top right, there is a Jenkins status bar showing notifications and user information: 'Flameshot Just now', 'Flameshot Info', 'Screenshot aborted.', 'Deploy Engineer', and 'log out'.

At the bottom right, there are links for 'REST API' and 'Jenkins 2.492.1'.

Enable Source Code Management (SCM)

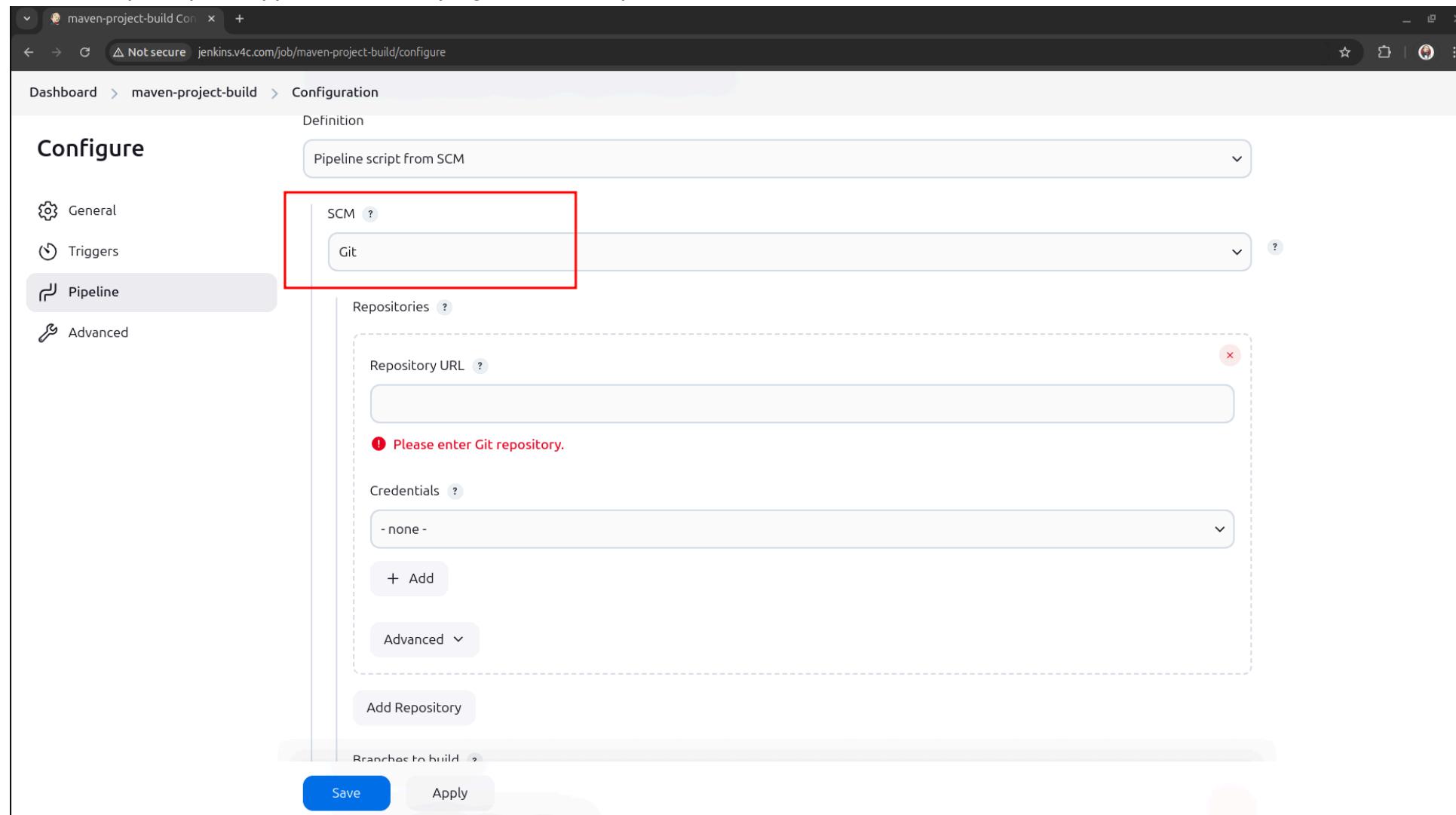
- Scroll down to the **Source Code Management** section.

The screenshot shows the Jenkins configuration interface for a job named "maven-project-build". The left sidebar lists "Configure" sections: General, Triggers, Pipeline (which is selected and highlighted in grey), and Advanced. The main content area is titled "Pipeline" and contains the following fields:

- Definition:** A dropdown menu set to "Pipeline script from SCM", which is highlighted with a red box.
- SCM:** A dropdown menu set to "None".
- Script Path:** A text input field containing "Jenkinsfile".
- Lightweight checkout:** A checked checkbox.
- Pipeline Syntax:** A link to pipeline syntax documentation.
- Advanced:** A dropdown menu set to "Advanced".

At the bottom are "Save" and "Apply" buttons.

- Select **Git** (this option appears if the Git plugin is installed).



Enter the GitHub Repository URL

- In the **Repository URL** field, enter the GitHub repository link. **Example:** `https://github.com/your-org/maven-app.git`
- If the repository is **public**, Jenkins can access it without authentication.

- If the repository is **private**, Jenkins needs credentials to access it.

The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The "Pipeline" tab is selected in the sidebar. Under the "SCM" section, "Git" is chosen. In the "Repositories" section, there is a field for "Repository URL" containing "https://github.com/your-org/maven-app.git". This URL is highlighted with a red rectangle. Below the URL, an error message is displayed:

```
⚠ Failed to connect to repository : Command "git ls-remote -h -- https://github.com/your-org/maven-app.git HEAD" returned status code 128:  
stdout:  
stderr: remote: Support for password authentication was removed on August 13, 2021.  
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls  
for information on currently recommended modes of authentication.  
fatal: Authentication failed for 'https://github.com/your-org/maven-app.git/'
```

In the "Credentials" section, a dropdown menu shows "- none -". There is also a "+ Add" button. At the bottom of the page are "Save" and "Apply" buttons.

Add GitHub Credentials (For Private Repositories)

If your GitHub repository is **private**, Jenkins needs authentication to pull the code. Follow these steps:

1. Click **Add** → **Jenkins** (next to the "Credentials" dropdown).

The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The "Pipeline" tab is selected in the sidebar. The main area displays the configuration for a GitHub repository. A red box highlights the "Credentials" section, specifically the "+ Add" button, which is the target for the first step in the list above.

Repository URL: https://github.com/your-org/maven-app.git

Failed to connect to repository : Command "git ls-remote -h -- https://github.com/your-org/maven-app.git HEAD" returned status code 128:
stdout:
stderr: remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls
for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/your-org/maven-app.git/'

Credentials: - none -

+ Add

Jenkins

Advanced

Add Repository

Branches to build:

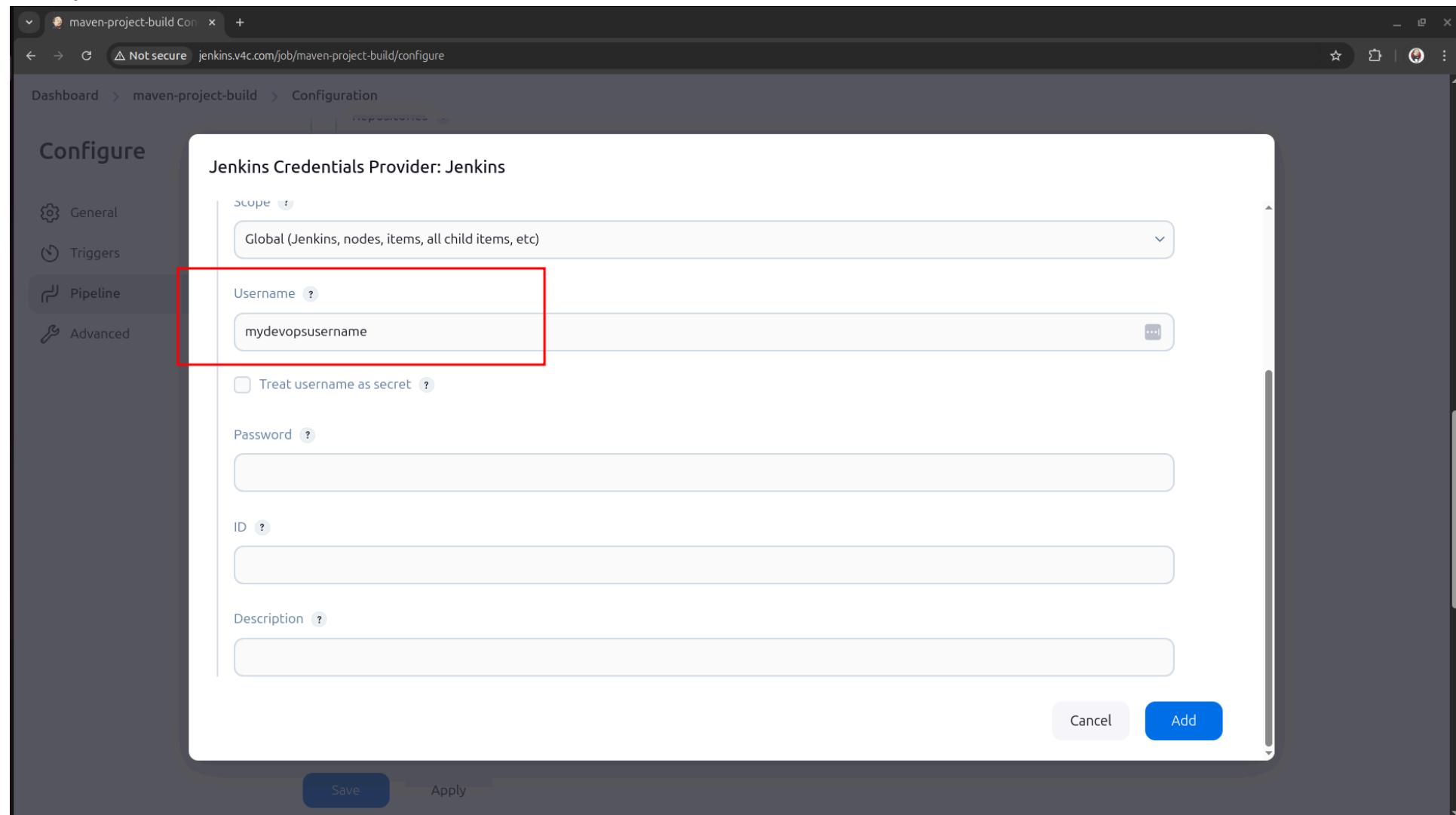
Branch Specifier (blank for 'any')

Save Apply

2. Select Kind: Username with password.

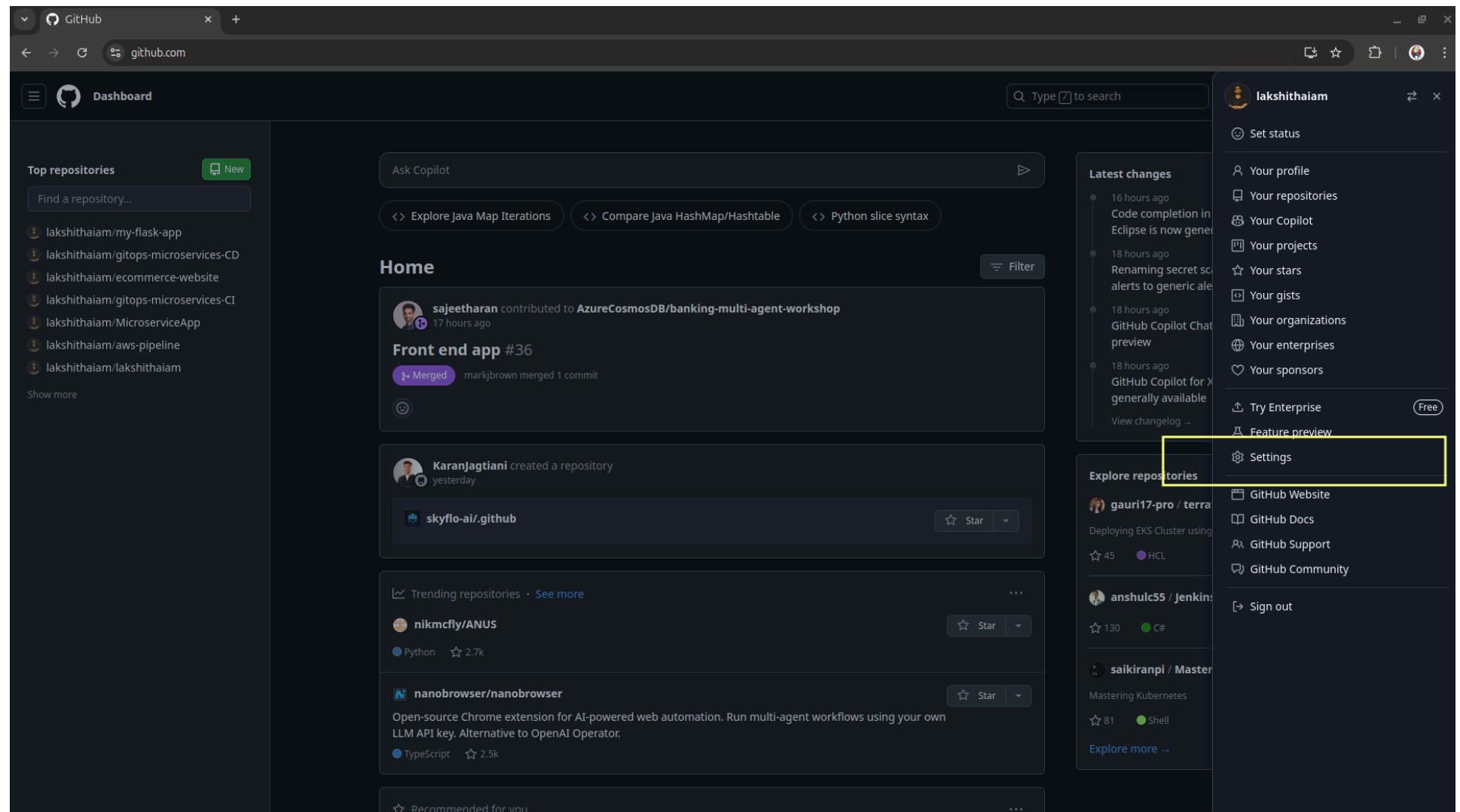
The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The "Pipeline" tab is selected in the sidebar. A modal dialog titled "Jenkins Credentials Provider: Jenkins" is open, specifically the "Add Credentials" section. The "Domain" dropdown is set to "Global credentials (unrestricted)". The "Kind" dropdown is highlighted with a red box and contains the option "Username with password". Below these, the "Scope" dropdown is set to "Global (Jenkins, nodes, items, all child items, etc)". The "Username" field is empty. There is a checkbox for "Treat username as secret" which is unchecked. The "Password" field is also empty. At the bottom of the dialog are "Save" and "Apply" buttons.

3. Enter your GitHub username.



4. Enter a Personal Access Token (PAT) instead of a password

To generate a PAT, go to **GitHub → Settings → Developer Settings → Personal Access Tokens**.

A screenshot of the GitHub Dashboard in a dark-themed browser window. The dashboard features a sidebar on the left with 'Top repositories' (lakshithaiam/my-flask-app, lakshithaiam/gitops-microservices-CD, etc.) and a central 'Home' feed showing recent contributions and repository activity. A yellow box highlights the 'Settings' link in the top right corner of the sidebar.

Top repositories

- [lakshithaiam/my-flask-app](#)
- [lakshithaiam/gitops-microservices-CD](#)
- [lakshithaiam/ecommerce-website](#)
- [lakshithaiam/gitops-microservices-CI](#)
- [lakshithaiam/MicroserviceApp](#)
- [lakshithaiam/aws-pipeline](#)
- [lakshithaiam/lakshithaiam](#)

[Find a repository...](#) [New](#)

Ask Copilot

[Explore Java Map Iterations](#) [Compare Java HashMap/Hashtable](#) [Python slice syntax](#)

Home [Filter](#)

 **sajeetharan** contributed to [AzureCosmosDB/banking-multi-agent-workshop](#) 17 hours ago

Front end app #36 [Merged](#) markjbrown merged 1 commit

 **Karanjagtiani** created a repository yesterday

[skyflo-ai/.github](#) [Star](#)

Trending repositories [See more](#)

 **nikmcfly/ANUS** Python 2.7k [Star](#)

 **nanobrowser/nanobrowser** Open-source Chrome extension for AI-powered web automation. Run multi-agent workflows using your own LLM API key. Alternative to OpenAI Operator. TypeScript 2.5k [Star](#)

Recommended for you

Latest changes

- 16 hours ago Code completion in Eclipse is now general
- 18 hours ago Renaming secret scope alerts to generic alert
- 18 hours ago GitHub Copilot Chat preview
- 18 hours ago GitHub Copilot for X generally available

[View changelog →](#)

lakshithaiam

[Set status](#)

[Your profile](#) [Your repositories](#) [Your Copilot](#) [Your projects](#) [Your stars](#) [Your gists](#) [Your organizations](#) [Your enterprises](#) [Your sponsors](#)

[Try Enterprise](#) [Feature preview](#) [Free](#)

Settings

[GitHub Website](#) [GitHub Docs](#) [GitHub Support](#) [GitHub Community](#)

[Sign out](#)

Your profile

github.com/settings/profile

Sessions

SSH and GPG keys

Organizations

Enterprises

Moderation

Code, planning, and automation

Repositories

Codespaces

Packages

Copilot

Pages

Saved replies

Security

Code security

Integrations

Applications

Scheduled reminders

Archives

Security log

Sponsorship log

Developer settings

Linux Enthusiast

You can @mention other users and organizations to link to them.

Pronouns

he/him

URL

ORCID ID

ORCID provides a persistent identifier - an ORCID iD - that distinguishes you from other researchers. Learn more at [ORCID.org](#).

Connect your ORCID iD

Social accounts

https://www.linkedin.com/in/lakshithaiam/

Link to social profile 2

Link to social profile 3

Link to social profile 4

Company

You can @mention your company's GitHub organization to link it.

Location

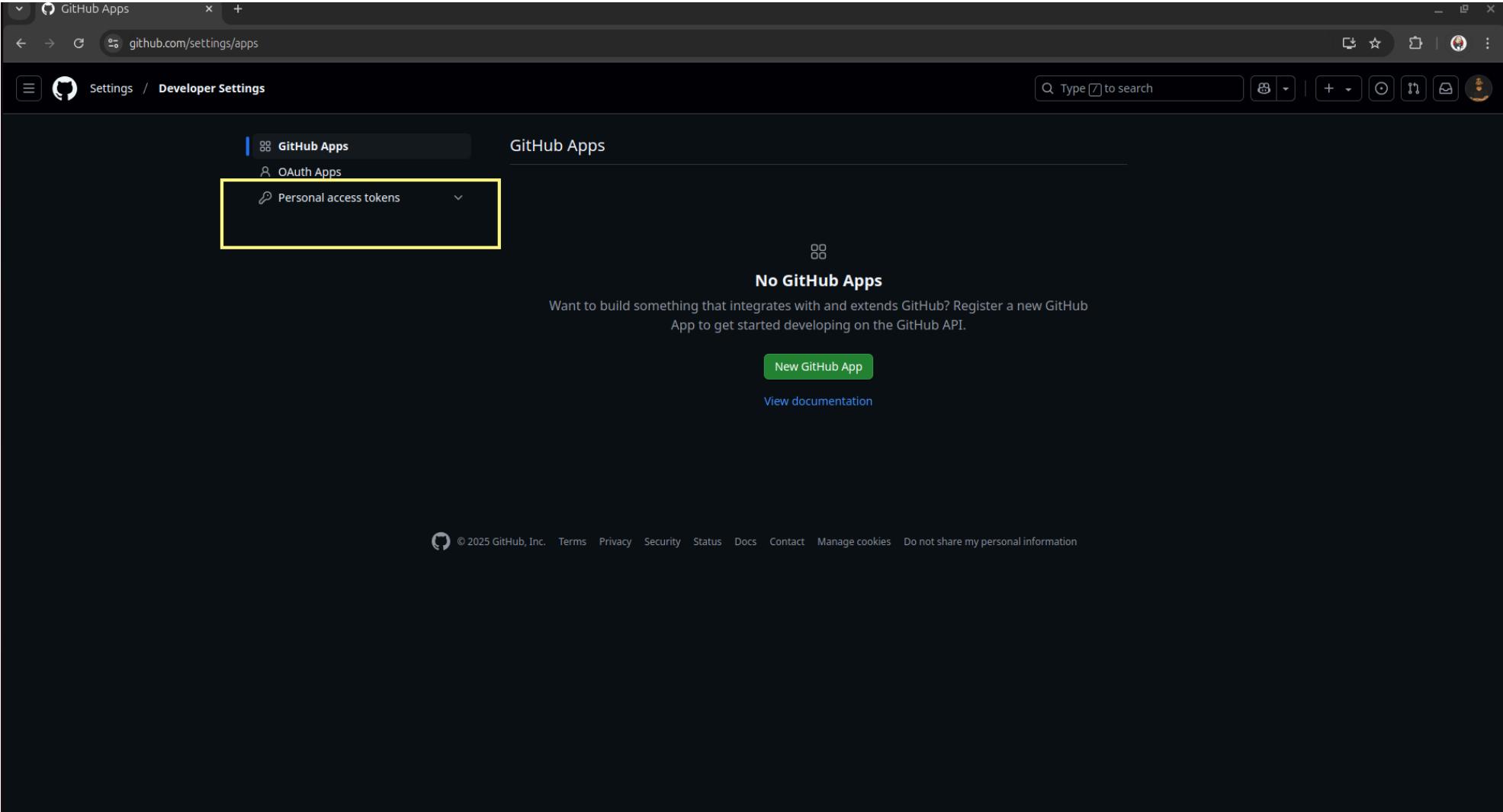
Homagama Sri Lanka

Display current local time

Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

Update profile

A screenshot of a web browser window showing the GitHub Apps settings page. The URL in the address bar is github.com/settings/apps. The page title is "GitHub Apps". On the left, there is a sidebar with "GitHub Apps" selected, followed by "OAuth Apps" and "Personal access tokens". A yellow rectangular box highlights the "Personal access tokens" section. The main content area is titled "GitHub Apps" and displays a message: "No GitHub Apps". Below this, it says "Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API." It features a green "New GitHub App" button and a blue "View documentation" link. At the bottom, there is a GitHub logo and copyright information: "© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information".

 GitHub Apps

github.com/settings/apps

Settings / Developer Settings

GitHub Apps

GitHub Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

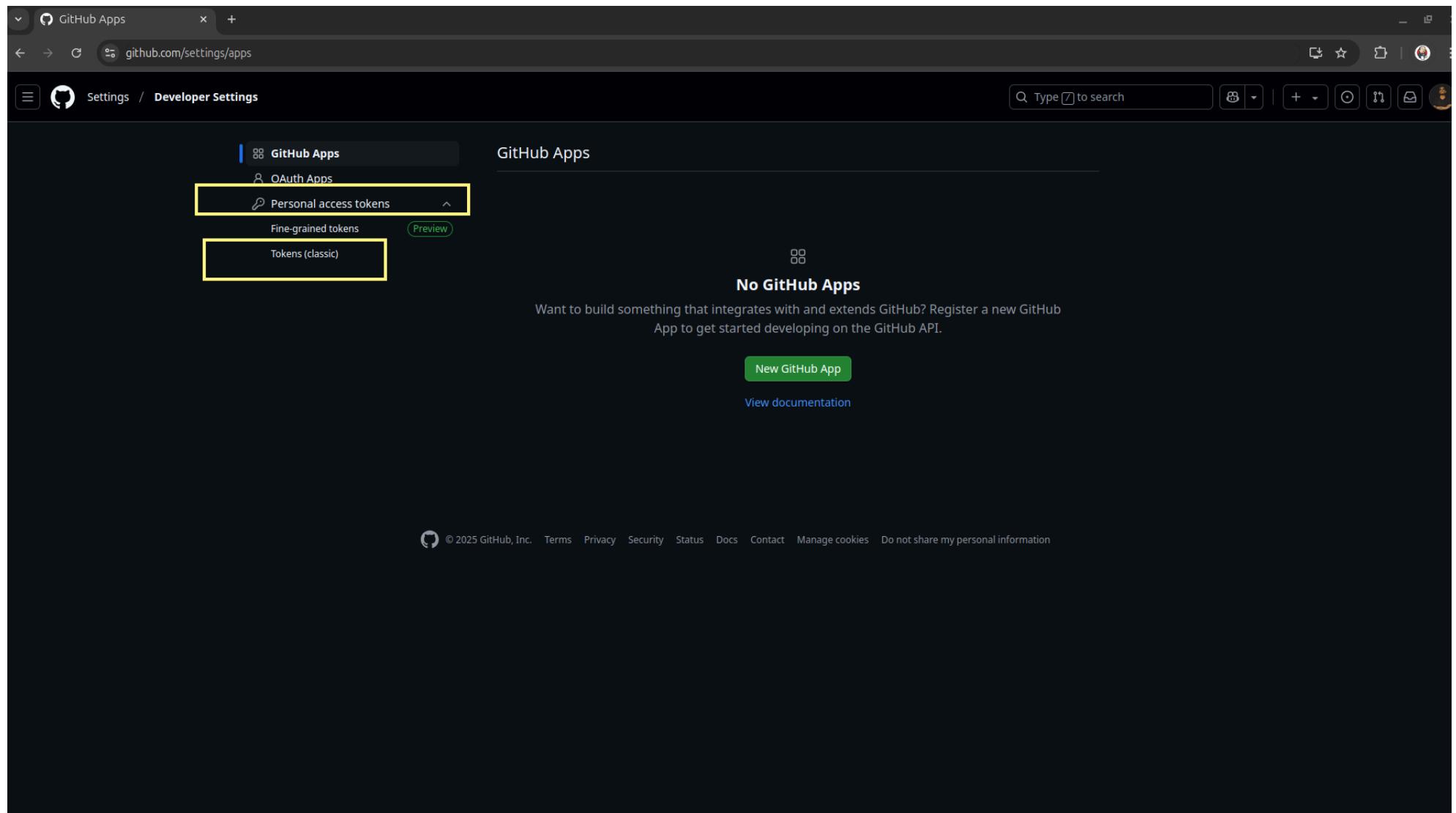
No GitHub Apps

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API.

New GitHub App

View documentation

 © 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



The screenshot shows the GitHub 'Personal Access Tokens' page under 'Developer Settings'. The left sidebar includes 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (selected), with sub-options 'Fine-grained tokens' and 'Tokens (classic)' (highlighted with a blue border). The main area is titled 'Personal access tokens (classic)' and displays two tokens:

- jenkinstoken** — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:network_configurations, write:packages Last used within the last 4 weeks Delete
- JSON Resume Registry** — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages Never used Delete

A note at the bottom states: "Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#)".

At the bottom of the page, there's a footer with links: © 2025 GitHub, Inc. | Terms | Privacy | Security | Status | Docs | Contact | Manage cookies | Do not share my personal information.

Click **Generate new token** and select scopes like:

- repo** (Full control of private repositories)
- read:packages** (Read access to GitHub Packages)

New Personal Access Token

github.com/settings/tokens/new

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic) (Preview)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

mavan-project

What's this token for?

Expiration

30 days (Apr 11, 2025) ▼

The token will expire on the selected date.

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys

Personal Access Tokens | +

github.com/settings/tokens

Settings / Developer Settings

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Preview

Tokens (classic)

Personal access tokens (classic)

Generate new token ▾

Make sure to copy your personal access token now. You won't be able to see it again!

ghp_rjkoYER1zrGd8WOieP8jpcCXvC101F0MZhge  Delete

jenkinstoken — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, Last used within the last 4 weeks   This token has no expiration date.

JSON Resume Registry — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, Never used   This token has no expiration date.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

5. Click **Add** to save the credentials.

The screenshot shows the Jenkins configuration interface for a job named "maven-project-build". A modal dialog titled "Jenkins Credentials Provider: Jenkins" is open. Inside the dialog, there are fields for "Scope" (set to "Global (Jenkins, nodes, items, all child items, etc)"), "Username" ("mydevopsusername"), and "Password" (represented by a series of dots). A checkbox for "Treat username as secret" is unchecked. Below these are fields for "ID" ("git-cred") and "Description" ("git-cred"). At the bottom right of the dialog are "Cancel" and "Add" buttons. A red box highlights the "Password" input field. The main Jenkins dashboard is visible in the background, showing the "General", "Triggers", "Pipeline", and "Advanced" tabs.

6. Select the saved credentials from the **Credentials** dropdown.

The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Pipeline script from SCM" and shows a "Git" configuration. Under "Repositories", there is a "Repository URL" field containing "https://github.com/lakshithaiam/maven-app.git". Below it is a "Credentials" dropdown menu with the option "mydevopsusername/******** (git-cred)" selected. This entire "Credentials" section is highlighted with a red rectangle. At the bottom of the SCM section are buttons for "+ Add", "Advanced", "Add Repository", and "Branches to build". At the very bottom of the page are "Save" and "Apply" buttons.

Specify the Branch

- If you want to build a specific branch (e.g., `main` or `develop`), enter the branch name in the **Branches to build** field.

Example:

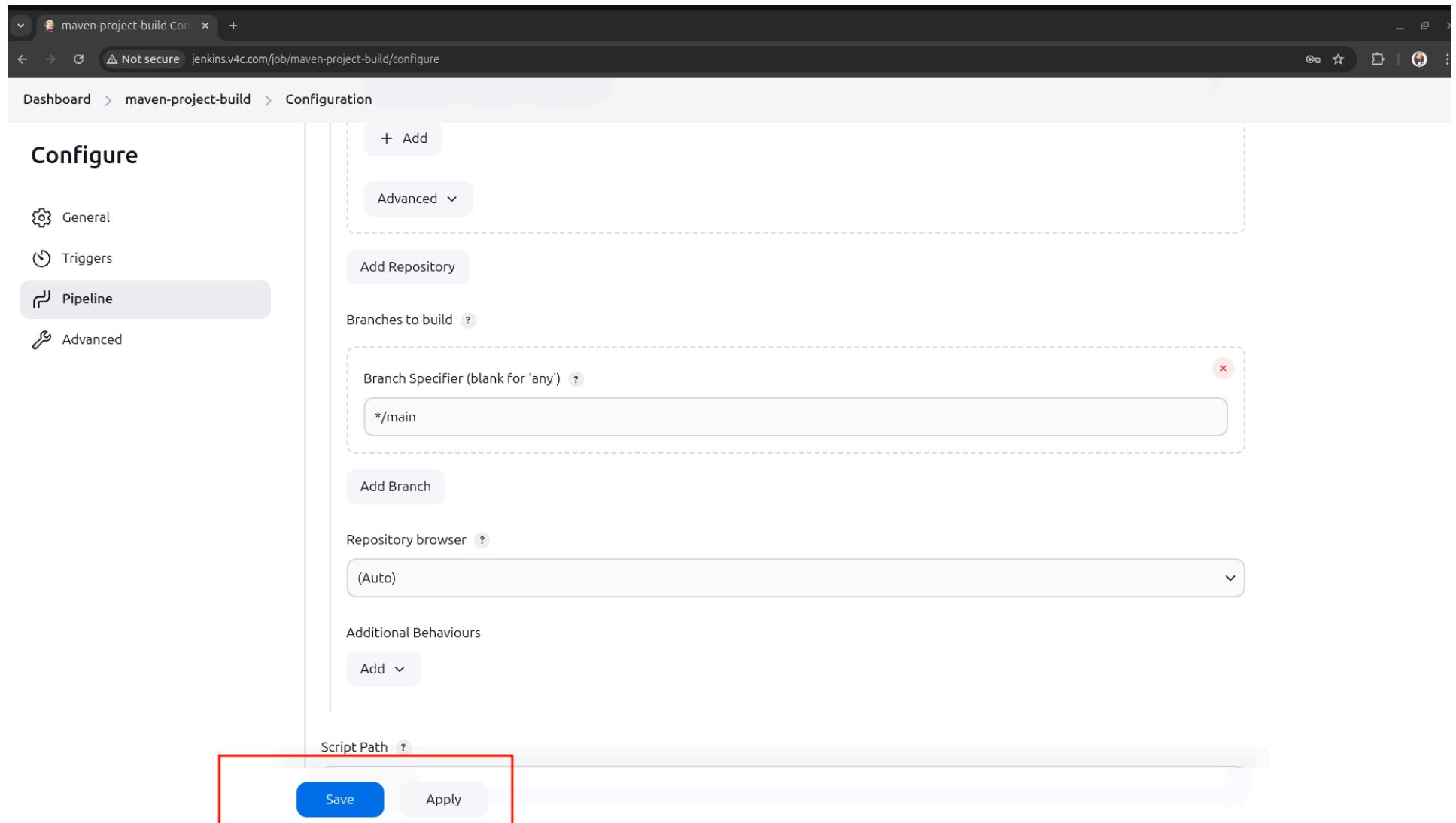
`:main`

- If left blank, Jenkins will pull the **default branch** of the repository.

The screenshot shows the Jenkins job configuration interface for a job named "maven-project-build". The "Pipeline" tab is selected in the sidebar. In the main configuration area, under the "Branches to build" section, there is a text input field labeled "Branch Specifier (blank for 'any')". This field contains the value "/main", which is highlighted with a red rectangular box. Below this field is a "Save" button and an "Apply" button.

Save the Configuration

- Click **Save** to apply the changes.



Now, your Jenkins pipeline is successfully linked to the GitHub repository!

Set Up the Pipeline Script

In this step, we will define a **Jenkins pipeline script** to automate the build, test, and package process for our **Maven project**.

Where to Add the Script?

1. Scroll down to the **Pipeline** section in the job configuration page.

The screenshot shows the Jenkins job configuration page for a job named "maven-project-build". The left sidebar has sections for General, Triggers, Pipeline (which is highlighted with a red box), and Advanced. The main content area is titled "Pipeline" and contains instructions to "Define your Pipeline using Groovy directly or pull it from source control." It features a "Definition" dropdown set to "Pipeline script", a "Script" code editor with a "try sample Pipeline..." button, and a checked "Use Groovy Sandbox" checkbox. Below the editor is a "Pipeline Syntax" link. At the bottom, there are "Advanced" settings, a "Save" button, and an "Apply" button.

2. Select **Pipeline Script** from the dropdown menu.

The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The "Pipeline" tab is selected. In the "Definition" section, the "Pipeline script" tab is highlighted with a red box. Below it is a code editor area with a "Script" placeholder and a "try sample Pipeline..." dropdown. A "Use Groovy Sandbox" checkbox is checked. At the bottom, there are "Save" and "Apply" buttons.

3. Copy and paste the following script into the text box:

Pipeline Script Breakdown

```
pipeline {  
    agent any
```

- `pipeline {}` → Defines the start of the Jenkins pipeline.
- `agent any` → Runs the pipeline on any available Jenkins agent.

Checkout Code from GitHub

```
stage('Checkout') {  
    steps {  
        git branch: 'main', url: 'https://github.com/your-org/maven-app.git'  
    }  
}
```

- **Purpose:** Retrieves the latest code from the **GitHub repository**.
- `git branch: 'main'` → Specifies the branch to clone (change `'main'` if using another branch like `'develop'`).
- `url: 'https://github.com/your-org/maven-app.git'` → Replace with your actual repository URL.
- **Outcome:** The repository is cloned onto the Jenkins workspace.

Build the Application

```
stage('Build') {  
    steps {  
        sh 'mvn clean install'  
    }  
}
```

- **Purpose:** Compiles the Java code and downloads project dependencies.

- `mvn clean install` →
 - `clean` → Removes old build artifacts.
 - `install` → Builds the project and installs the generated JAR/WAR file in the local Maven repository.
- **Outcome:** The application is successfully compiled and ready for testing.

Run Unit Tests

```
stage('Test') {  
    steps {  
        sh 'mvn test'  
    }  
}
```

- **Purpose:** Runs the **unit tests** written for the project.
- `mvn test` → Executes all test cases defined in the project.
- **Outcome:** Ensures the code is functional and does not break any existing features.

Package the Application

```
stage('Package') {  
    steps {  
        sh 'mvn package'  
    }  
}
```

- **Purpose:** Bundles the application into a **deployable artifact** (JAR or WAR).
- `mvn package` → Creates a package (`.jar` or `.war`) inside the `target/` directory.

- **Outcome:** The final artifact is built and ready for deployment.

Final Pipeline Script (Full View)

```
pipeline {  
    agent any  
  
    stages {  
        stage('Checkout') {  
            steps {  
                git branch: 'main', url: 'https://github.com/your-org/maven-app.git'  
            }  
        }  
  
        stage('Build') {  
            steps {  
                sh 'mvn clean install'  
            }  
        }  
  
        stage('Test') {  
            steps {  
                sh 'mvn test'  
            }  
        }  
  
        stage('Package') {  
            steps {  
                sh 'mvn package'  
            }  
        }  
    }  
}
```

```
        }
    }
}
}
```

Next Steps After Adding the Pipeline Script

Now that you've copied the pipeline script into Jenkins, follow these steps:

Save and Apply the Pipeline Job

1. Scroll down to the bottom of the pipeline configuration page.

2. Click **Apply** and then **Save**.

The screenshot shows the Jenkins Pipeline configuration page for a job named "maven-project-build". The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Pipeline" and contains a "Definition" section with a "Pipeline script" input field. A red box highlights the pipeline script code:

```
11  stage('Build') {  
12      steps {  
13          sh 'mvn clean install'  
14      }  
15  }  
16  
17  stage('Test') {  
18      steps {  
19          sh 'mvn test'  
20      }  
21  }  
22  
23  stage('Package') {  
24      steps {  
25          sh 'mvn package'  
26      }  
27  }  
28 }
```

Below the script is a "try sample Pipeline..." dropdown and a "Use Groovy Sandbox" checkbox (which is checked). A "Pipeline Syntax" link is also present. At the bottom, there is an "Advanced" section with an "Advanced" dropdown and a "Save" button highlighted with a red box, followed by an "Apply" button.

Trigger the First Build

1. On the Jenkins job page, click "**Build Now**".

2. Jenkins will start executing the pipeline.

The screenshot shows the Jenkins Pipeline interface for the 'maven-project-build' job. On the left, a sidebar lists options: Status, Changes, Build Now (which is highlighted with a red box), Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area is titled 'Stage View' and contains the message 'No data available. This Pipeline has not yet run.' Below this is a 'Permalinks' section. At the bottom left is a 'Builds' section with the message 'No builds'. At the top right, there are navigation links for Deploy Engineer and log out, along with a search bar and notification icons.

Conclusion

Jenkins is a powerful tool for automating software builds and deployments. In this guide, we covered:

- ✓ What Jenkins is and why it is useful
- ✓ Deploying Jenkins on Kubernetes
- ✓ Creating a simple Jenkins job
- ✓ Running a basic Jenkins pipeline

