

Lab: Enable and configure Jenkins security

6 minute read

The goal of this exercise is to set up a basic security model on your Jenkins instance and manipulate credentials.

Enabling authentication

The **Security Realm** defines the authentication method to be used. For this lab, we are going to use the Mock Security Realm plugin, which simulates the information provided by LDAP, Active Directory, etc. We must install that plugin and populate its (very simple) database.

Install the Mock Security Realm plugin

To install the Mock Security Realm plugin in your lab:

1. Go to **Manage Jenkins Manage Plugins**.
2. Select the **Available** tab.
3. Start to type "Mock Security Realm" into the search box until the plugin is listed.

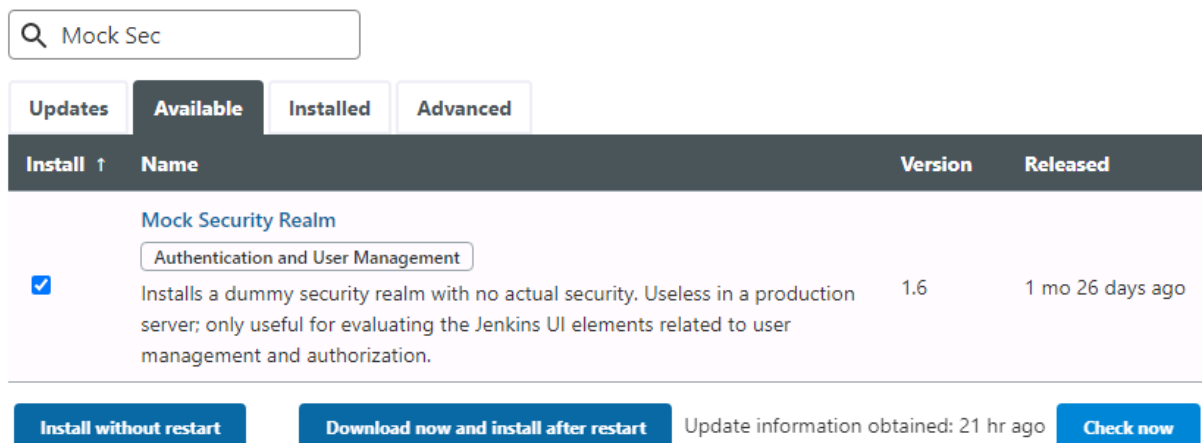


Figure 1. Install Mock Security Realm plugin

4. Check the box in the **Install** column.
5. Select **Download now and install after restart**.
6. Wait for Jenkins to restart.

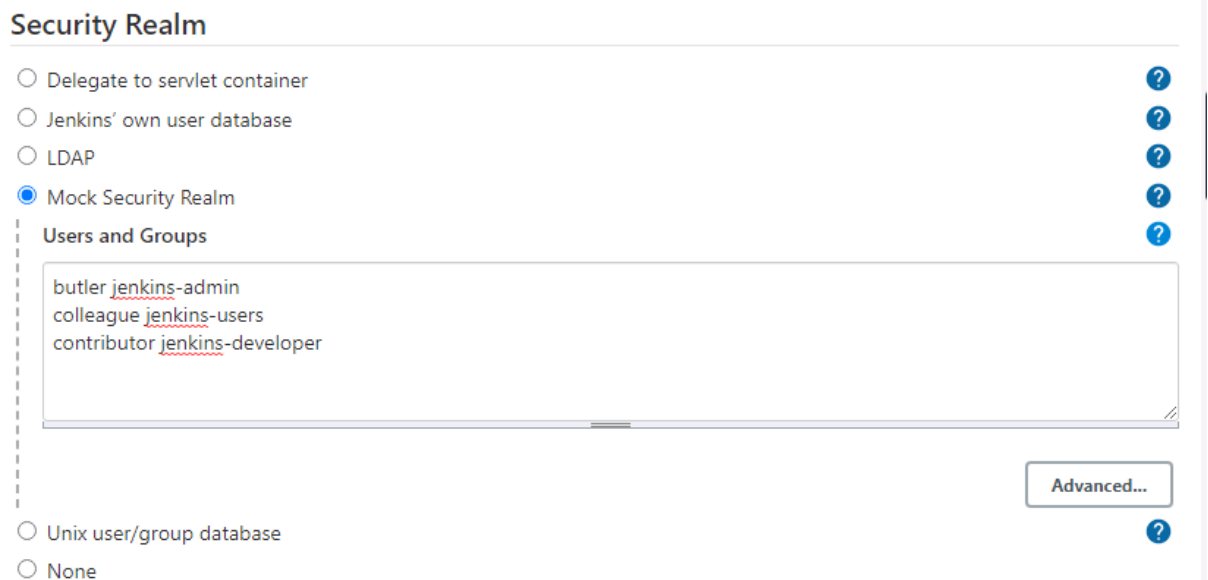
Installing the Mock Security Realm plugin automatically adds that as an option to the **Security Realm** section of **Securing Jenkins**.

The [Mock Security Realm](#) provides **no actual security** and should never be used for production systems! It is, however, useful when prototyping authorization schemes because it lets you simulate having multiple users and external groups that can be bound to local groups without the overhead of actually configuring a standard security realm.

Implement the security realm

To implement the Mock Security Realm for your lab:

1. Go to **Manage Jenkins** **Configure Global Security**.
2. Select **Mock Security Realm**.
3. This opens a window where you can define the users.
 - Each user is defined by username and an external group, which emulates the equivalent of the LDAP user group. A space separates the username from the name of the external group.
 - Each user's password is the same as the username.
4. Add the following three users:
5. butler jenkins-admin
6. colleague jenkins-users
- contributor jenkins-developer



The screenshot shows the 'Security Realm' configuration page in Jenkins. The 'Mock Security Realm' option is selected with a blue radio button. Below this, the 'Users and Groups' section is expanded, showing a text area with three entries: 'butler jenkins-admin', 'colleague jenkins-users', and 'contributor jenkins-developer'. The text area has a scrollbar. To the right of the radio buttons are five question mark icons. At the bottom right, there is an 'Advanced...' button with a question mark icon.

Figure 2. Populate the Mock Security database

7. Select **Save** to enable Authentication.

This returns you to the **Manage Jenkins** screen.

Sign in to confirm authentication

To confirm that authentication is working:

1. Sign out of Jenkins, then sign on to Jenkins as **butler**. The password is same as the username. You are now logged in as the **butler** user, so authentication works.
2. Verify that the buttons in the upper right corner of the screen show **Butler** as the display name and show he **log out** link:

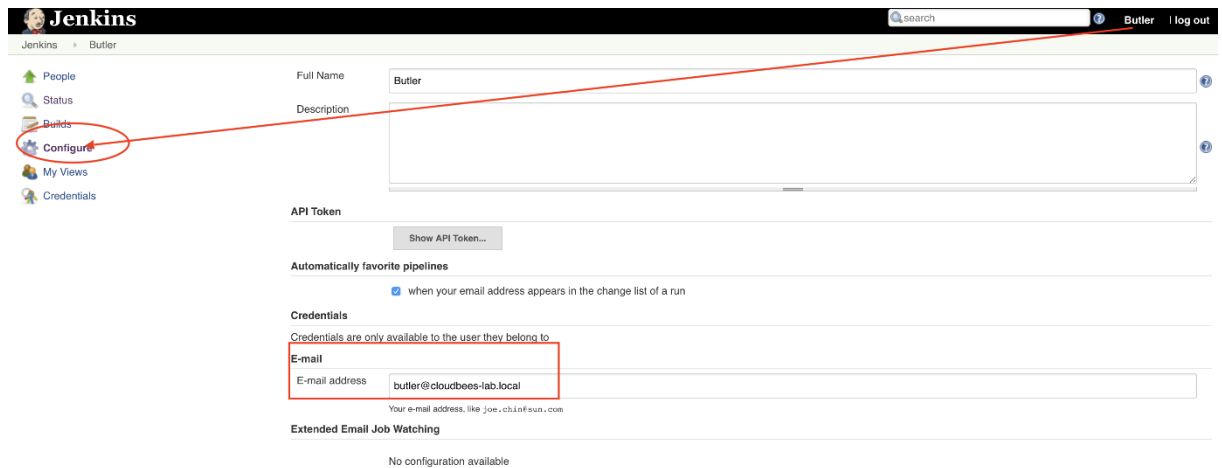


Figure 3. Sign in as butler

3. Select **Configure** to check your settings.

Some configuration fields (such as the **Email** field) may be populated using data from the external authentication service such as LDAP. Remember that updating these values through Jenkins only modifies the Jenkins database. It does not affect the corporate LDAP Directory Service.

Configuring authorization policy

Now that we have configured the security realm to authenticate users, we are ready to configure the authorization policy and assign permissions to the authenticated users.

Sign in as **butler** and go to the **Manage Jenkins Configure Global Security** page. The authorization policy for the lab instance is set to the default value of **Logged-in users can do anything**. This policy is only appropriate for a small set of users in a trusted environment.

By default, Jenkins provides two authorization schemes that define rights based on group membership:

- Project-based Matrix Authorization Strategy
- Role-Based Strategy

Using one of these schemes makes it easier to scale Jenkins as the organization grows.

For this exercise, we are going to assign permissions by project. To do this, go to the **Manage Jenkins Configure Global Security** page and enable the **Project-based Matrix Authorization Strategy** policy. This displays the matrix with only the default values. Save this configuration, then sign out and sign back in as **butler**.

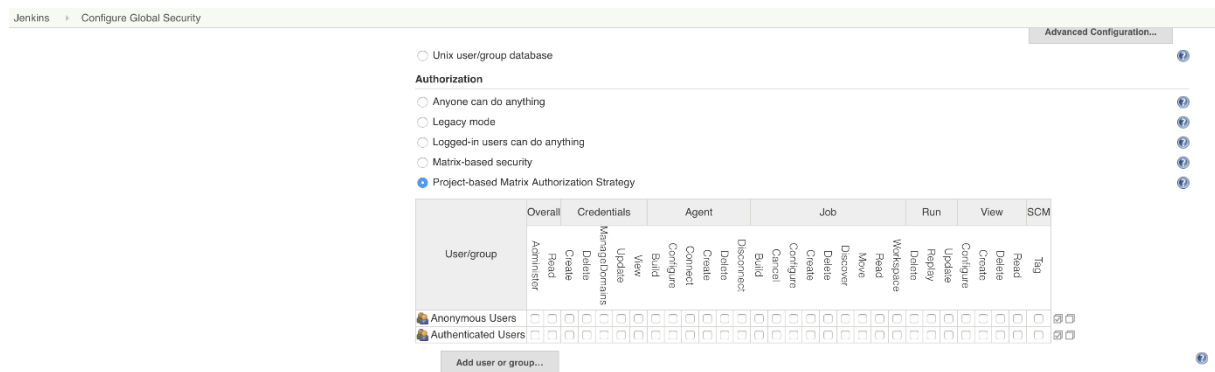


Figure 4. Enabling Project-Matrix Policy with default settings

Recovering if you get locked out of Jenkins

When modifying the authorization, one small mistake may leave you locked out of Jenkins. If this happens, do the following to recover:

1. Spawn a shell in the Jenkins machine:

```
$ docker exec -ti jenkins /bin/bash
```

2. Change to the \$JENKINS_HOME directory:

```
$ cd /var/jenkins_home
```

3. Show contents of the *config.xml* file:

```
4. $ cat config.xml
...
```

5. Show status of the "Enable Security" setting in the *config.xml* file:

```
6. $ grep useSecurity config.xml
...
```

7. Show status of the "Authorization Strategy" setting in the *config.xml* file:

```
8. $ grep authorizationStrategy config.xml
...
```

9. Set the XML attribute useSecurity to false in the *config.xml* file:

```
10. $ sed -i \
11. 's#<useSecurity>true#<useSecurity>false#g' \
    config.xml
```

12. Disable security by deleting the lines containing "authorizationStrategy" from the *config.xml* file:

```
13. $ sed -i \
14. '/authorizationStrategy/d' \
    config.xml
```

15. Validate the changes:

```
16. $ cat config.xml
...
```

17. Exit the Jenkins machine:

```
18. $ exit
    exit
```

19. Restart Jenkins to apply the configuration change:

```
$ docker restart jenkins
```

Configure the security matrix

When you select the **project-based security** authorization scheme, Jenkins displays a matrix. We will now populate that with information about our users and groups.

We want to apply the following rights:

- Non-authenticated users (Jenkins system's **anonymous** user) have no access to Jenkins.
- Authenticated users who are members of the **jenkins-users** LDAP group can view the Jenkins dashboard and non-private jobs.
- Authenticated users who are members of the **jenkins-developers** LDAP group can Launch, Create/View/Update/Delete (CRUD), Configure, Move jobs as well as use CRUD Credentials.
- Authenticated users who are members of the **jenkins-admin** LDAP group can administer Jenkins.

To assign these permissions, use the **User/group to add** field add your three groups (jenkins-users, jenkins-developers, and jenkins-admin) to the matrix.

Assign permissions to each group based on the permissions table below.

Hover your cursor over each permission to display a description popup. Read **Job:Discover**, **Job:Cancel** and **Run:Update** as mandatory examples.

Matrix Group/User	Members	Permissions (section, checkbox)	
anonymous	None (unauthenticated users)	None	None
jenkins-users	butler, colleague, contributor	Overall Job View	Read Read Read
jenkins-developers	butler, colleague	Overall Credentials Credentials Credentials Credentials Job Job Job Job Job Job Job	Read Create Delete Update View Build Cancel Configure Create Delete Discover Read

		Job Run Run Run View View View View	Workspace Delete Replay Update Configure Create Delete Read
jenkins-admin	butler	Overall	Administer

Table 1. Groups, Users and Permissions summary

Verifying your authorization settings

Your matrix should now look like this:

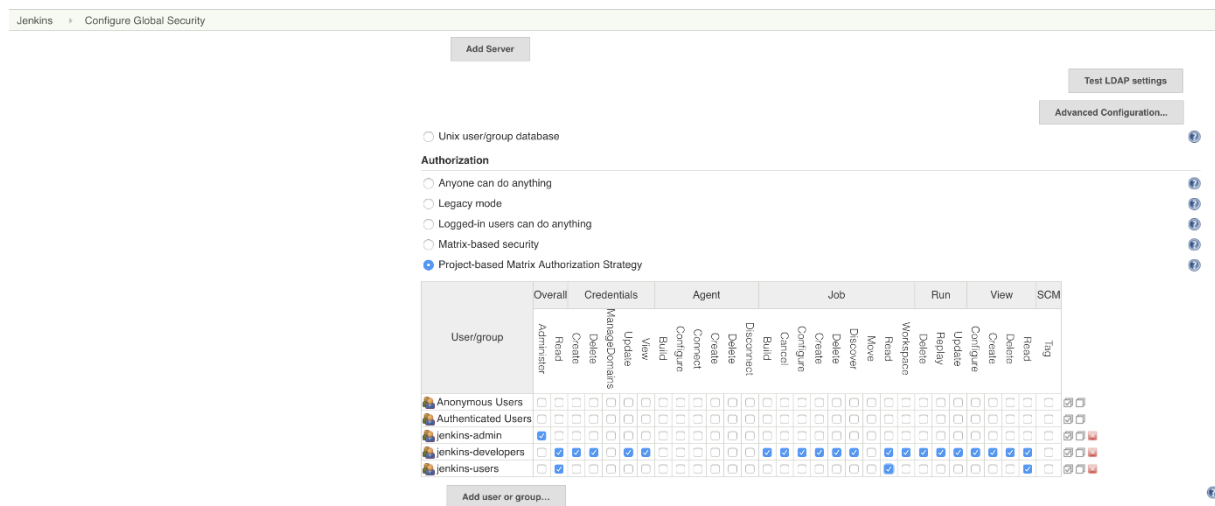


Figure 5. Project Matrix Settings

If you have set the correct permissions, select **Save** to activate the configuration.

Testing authorization settings

After saving the authorization settings, test this new configuration by doing the following:

1. Sign in as `butler` and verify that you still have rights to do everything.
2. Sign in as `colleague`. You should see the following:

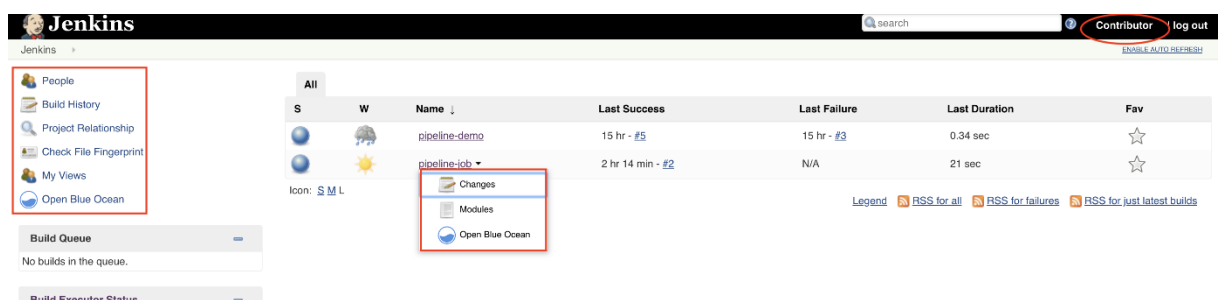


Figure 6. Expected dashboard for Contributor

3. Sign in as contributor. You should see the following:

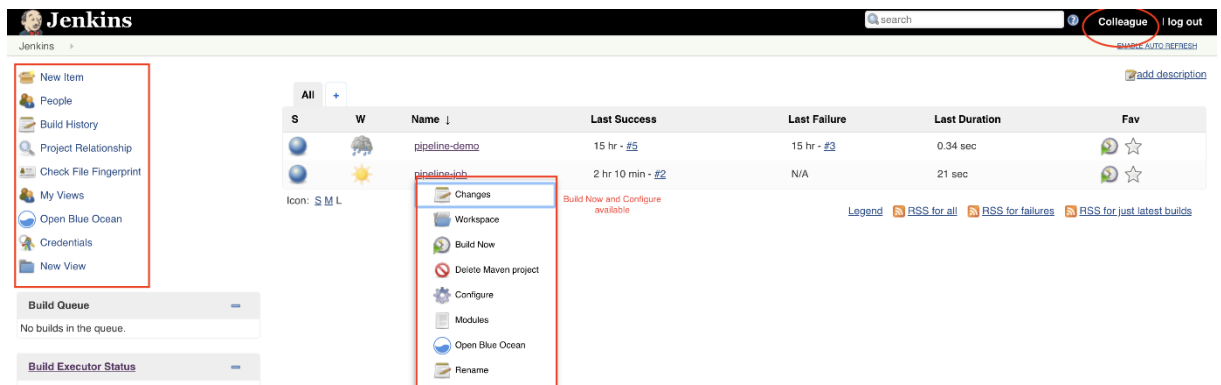


Figure 7. Expected dashboard for Colleague

Accounting: Privacy and credentials

The goal of this exercise is to create an administrator **private** job that is hidden from ALL users except administrators. Credentials are also used to access the related **private** repository.

Creating a new repository

To create a new repository in Gitea:

1. Select the **Gitserver (A local Git service)** link on your lab's home page and log into the Gitserver as `butler`.
2. Create a new repository by selecting **New Repository** from the drop-down next to the **+** in the upper toolbar:

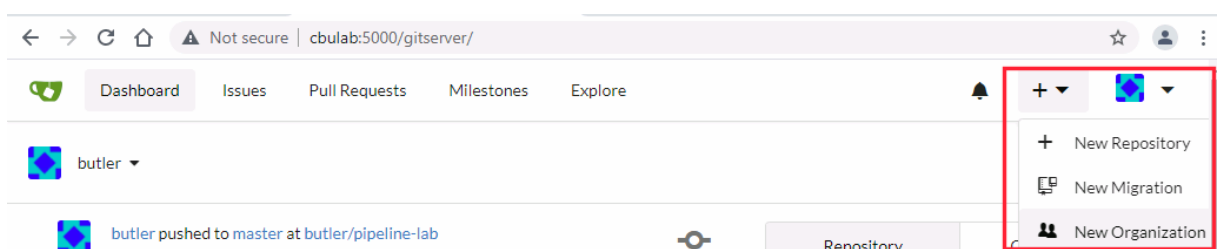


Figure 8. Create new repository in Gitea

3. Configure this repository with these settings:
 - **Owner:** `butler`
 - **Repository Name:** `admin-scripts`
 - **Visibility:** Select **Make Repository Private**
 - **Description:** A set of admin scripts that must remain private
 - Use the **default** values for the other options:

Figure 9. Configure new private repository

4. After the repository is created, verify that it is empty and private.
5. Copy the HTTP URL by selecting the clipboard icon to the right of the repository:

butler / admin-scripts

Unwatch 1 Star 0

Quick Guide Settings

Clone this repository Need help cloning? Visit [Help!](#)

HTTP `http://localhost/gitserver/butler/admin-scripts.git`

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin http://localhost/gitserver/butler/admin-scripts.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin http://localhost/gitserver/butler/admin-scripts.git
git push -u origin master
```

Figure 10. A private and empty repository in Gitea

Creating a new project in the new repository

1. Select **WebIDE (a Codiad Web IDE service)** link on your lab's home page. Go to the new tab that is created and create a new project based on this repository:
 - o **Project Name:** admin-scripts
 - o **Folder Name:** admin-scripts
2. Select **From Git Repo** to open the **Git Repository** field. Populate it by pasting the HTTP URL for the repository that you copied in the previous section.
3. Set the **Branch** field to the official branch for the repository. This defaults to **master** but another value (such as **main**) can be used.

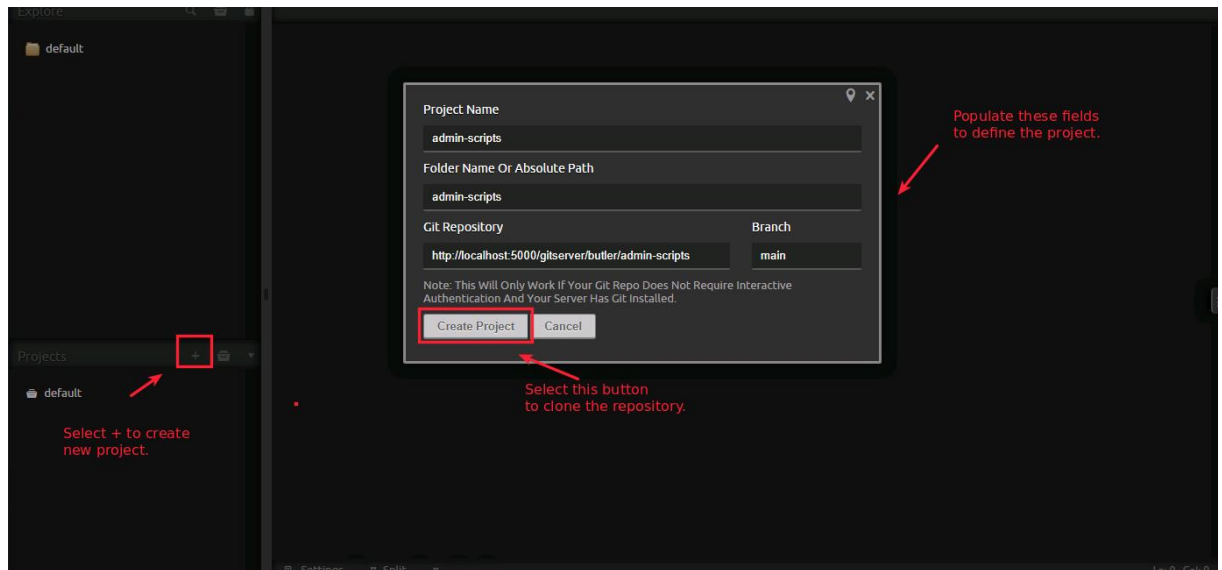


Figure 11. Create WebIDE project for admin-scripts

4. Select **Create Project** to clone the project. It will be cloned empty.

Create shell script in WebIDE project

In the new screen that is displayed, create a new file named `run-admin-task.sh` at the root, with the following content:

```
#!/bin/bash

set -e
set -u

echo "Some secret admin task to run there"
```

1. Add the new project to git tracking
2. Commit and push the project (providing credentials). Use `butler/butler` for the Git authentication when pushing.

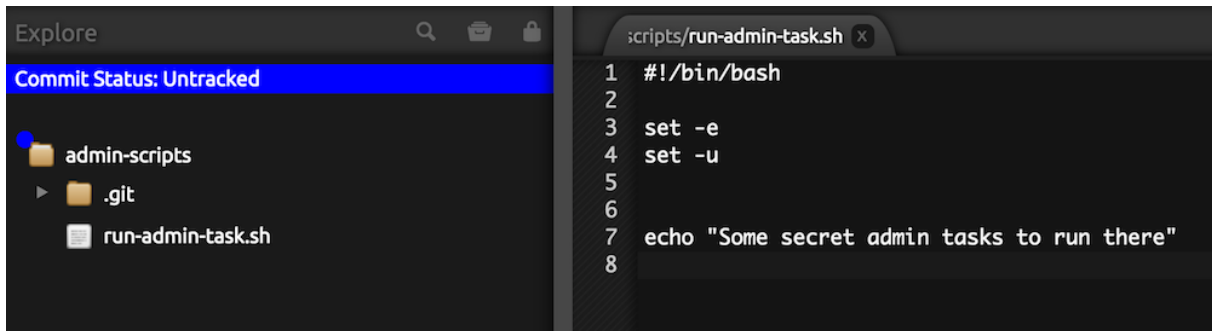


Figure 12. New script in private repository

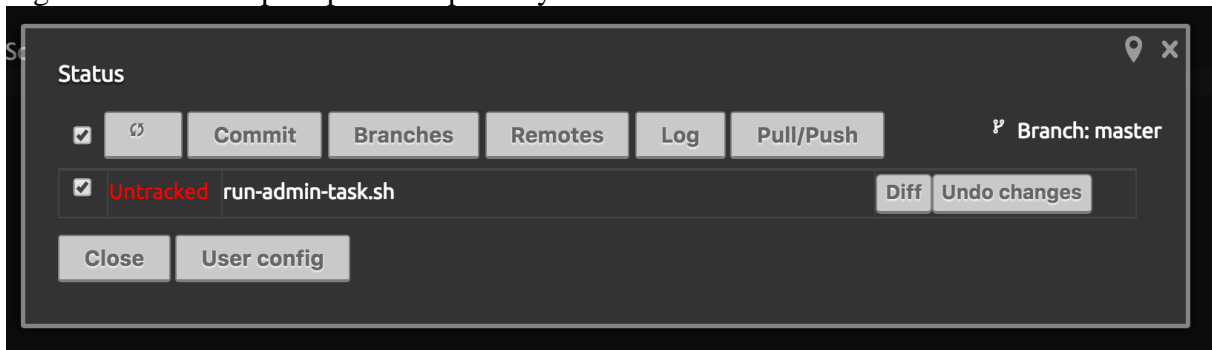


Figure 13. Git-tracking the new script

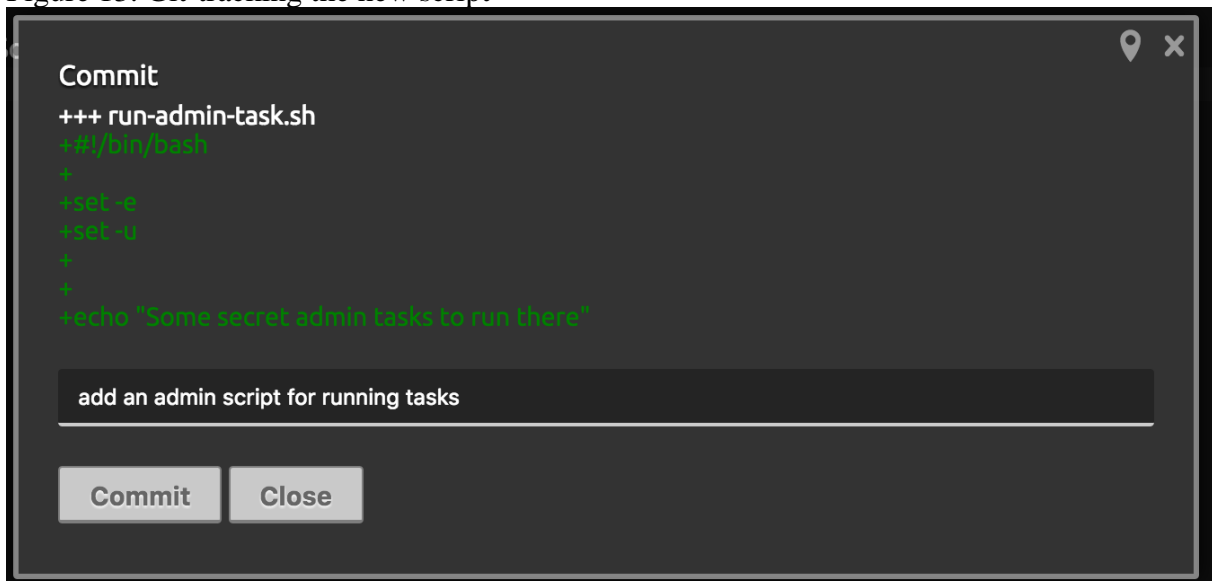


Figure 14. Git-committing the new script

Go back to Jenkins and sign in as **butler**. Create a new **Freestyle** job named `admin-tasks` and configure it as:

- Select the checkbox **Enable project-based security**.
 - Select the **Do not inherit permission grants from the other ACLs** checkbox.
 - Ensure that **only** the `jenkins-admin` group has all rights.

☒ Enable project-based security

Inheritance Strategy: **Do not inherit permission grants from other ACLs**

This object will **not** inherit the [global security settings](#), or any permissions from its ancestors. Only permissions explicitly enabled here will be granted. To ensure that users are not inadvertently locked out from Jenkins, an exception is made for the **Overall/Administer** permission: Administrators of Jenkins will still have access to this object even if not explicitly granted here.

User/group	Credentials		Job										Run		SCM			
	Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Delete	Discover	Move	Read	Workspace	Delete	Replay		Update	Tag
Anonymous 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"/>	<input type="checkbox"/>	<input type="checkbox"/>	<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"/> <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"/>	<input type="checkbox"/>	<input type="checkbox"/>	<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"/> <input type="checkbox"/>
jenkins-admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

[Add user or group...](#)

Figure 15. Correct rights for admin project

- To build the project, execute the following shell script:

```
bash -x run-admin-task.sh
```

- SCM:** Git repository, from the previously created **private** repository
 - Notice the **Red** error message that should appear: username and password are required to access this repository



Figure 16. No access to this repository without authentication

- To add Credentials, select **Add** and select **Jenkins** from the drop-down menu. Generate the Credential with these properties:
 - Domain:** Global credentials (unrestricted)
 - Kind:** Username with password
 - Username:** butler
 - Password:** butler
 - ID:** butler-gogs-creds
 - Description:** Butler's credentials for Gitea



The screenshot shows the Jenkins 'Add Credentials' form. The title is 'Jenkins Credentials Provider: Jenkins'. Below the title is a section 'Add Credentials' with a key icon. The form contains the following fields:

- Domain: Global credentials (unrestricted)
- Kind: Username with password
- Scope: Global (Jenkins, nodes, items, all child items, etc)
- Username: butler
- Password:
- ID: butler-gogs-creds
- Description: Butler's credentials for Gogs

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

Figure 17. Adding a credential for the private repository

- Make the **Red** message go away by selecting the newly created **Credential**.
- Save the job and build it. It should end in success, thanks to this credential.
- Sign out from **butler** then sign in as `colleague` in Jenkins. The job should not be visible.