Credentials

Contents

Overview

Keeping our private data private is a challenge faced by most, if not all, companies in IT.
In CI/CD there are many different pieces of private information that may be required as part of your pipeline, but that cannot be stored as plain text.
This is where the `Credentials` plugin is very useful.

Creating Credentials

To ensure that our data is not stored as plain text we can create credentials for Jenkins to use.
These credentials will have private data stored inside them and **will not display that data** in plain text.

Credentials can only be created by an Admin user, or by a user who has been given specific permission to create credentials.

Credentials are created in the following way:

From the `Jenkins Dashboard`, we head to `Manage Jenkins`.
Then, click on `Manage Credentials` under the `Security` heading.

Scope

We can choose the `scope` of our credentials.

The main scope we shall be using is `Global`.
These credentials are stored on the Jenkins instance and are accessible globally within Jenkins; this means that they can be used by any current or

future pipelines we create.

There is another option for the scope, which we can choose when we create a credential, this is called `System` and if we choose this then the credential will only be available to the Jenkins system processes, and not available within any pipelines.

Types

There are many different types of credentials available to us.
Depending on the format of our data, or what we need to use it for, we can choose a credential type to suit.

Some of the common credential types are listed below:

Username with Password

This allows you to store both the `username` and `password` in the same credential.
This is great for organising your credentials, it can reduce clutter if you have many usernames and passwords stored on Jenkins.

GitHub App

A `GitHub App` is a tool that adds extra functionality to your GitHub web application.
For example, you can install an app to your GitHub account/organisation that can check the information you have provided as part of a `pull request`, and if there isn't much, it will prompt you for more.

This type of credential is used for communicating directly with Apps you have installed on GitHub.

You are required to enter the `App ID` which points at the specific app you want to communicate with, this information is provided by GitHub.
You are also required to enter the `Key` which is how you authenticate with the App.

SSH Username with private key

This is for allowing Jenkins to connect to other machines via the SSH protocol.

SSH requires the `username` of the user to which we want Jenkins to connect and a `private key` for authentication.

The private key, as the name suggests, should be kept securely.
The username can either be stored as plain text or not, this is up to us.

Secret file

We can upload a file and store it as a credential.

The type of files we may want to upload are configuration files.
These often contain sensitive information, such as how to authenticate with a cloud provider.
This kind of information should not be stored in plain text!

Secret Text

Secret Text is the most versatile type of credential.
All you need to provide is a `string` which you would like to store as a credential, this string can be a *username*, *password*, *access token*, etc.

Certificate

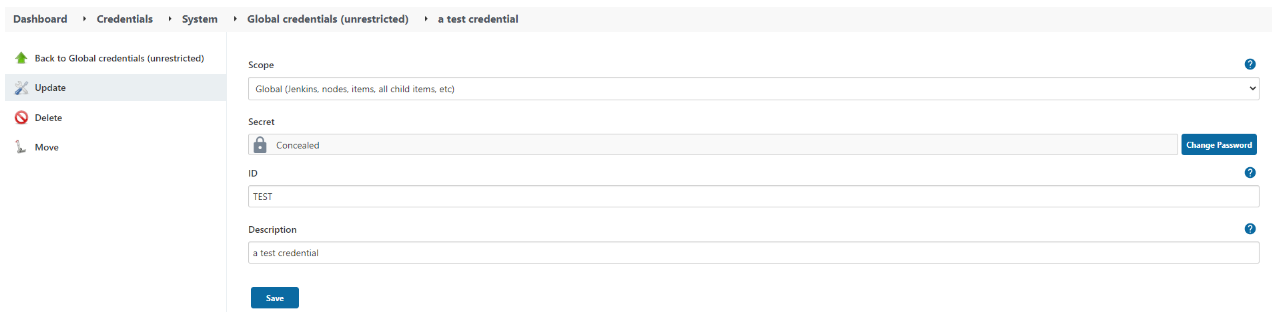
The certificate credential is used to store a PKCS #12 file.
This allows us to store many encrypted objects within a file, a password can also be stored for accessing the contents of a file.

Other required information

There are two fields that are available for every credential we create, these are `ID` and `Description`.

- ID: The ID is the unique key that Jenkins uses to identify each credential. The ID should be upper/lowercase characters and hyphens can be used to separate words.
  E.g. `github-authentication-token`
- Description: The description is an optional field that we can use to provide some extra detail about what is stored within the credential.

This is what a completed credential may look like:



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

Secret: Concealed

ID: TEST

Description: a test credential

Using Credentials

The handling of credentials is handled by yet another plugin.
The specific process of handling credentials is slightly different depending on whether you are working with a `Freestyle project` or `Pipeline`, but the general process is the same.

First, we need to create a `variable` for the `Build Environment`, then we assign a value to that variable using Jenkins credentials.

▶ Freestyle Project
▶ Pipeline

Limitations

Jenkins Credentials are encrypted and stored on the Jenkins instance, and credentials are never outputted as plain text.

Although credentials are encrypted, credential values stored on Jenkins can still be compromised if someone gains access to the instance that is running Jenkins.

In reality, there are many credential stores, that are not associated with Jenkins, which store credentials far more securely than Jenkins does.
Most of these third-party credential providers have plugins to allow them to work almost seamlessly with Jenkins.

Tutorial

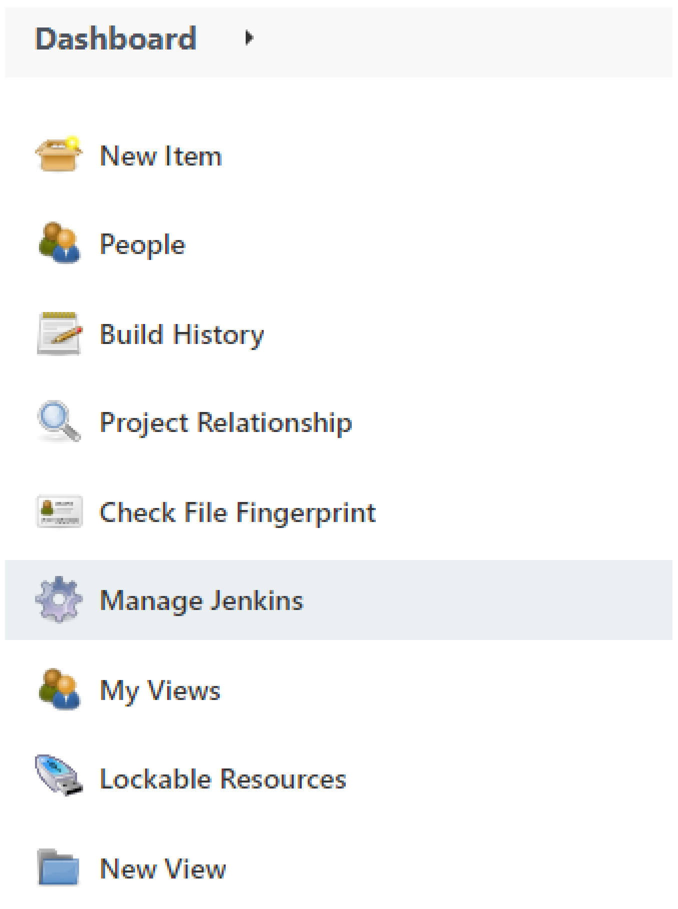
In this tutorial, we will be creating a freestyle project and making use of Jenkins credentials within that project.

Prerequisites

To complete this tutorial, you must make sure you have access to a Jenkins server **AND** you must be an admin user.

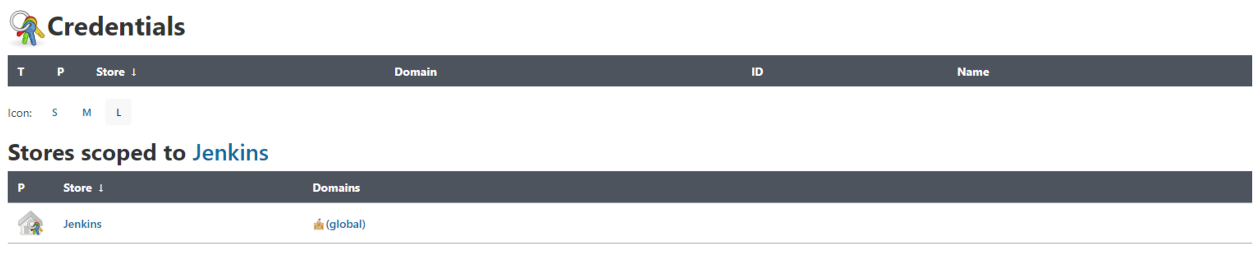
Create the Credential

From your `Jenkins Dashboard` navigate to `Manage Jenkins`.



Select `Manage Credentials`.

We are going to set a `Global credential`, so select the `global` button.


Credentials

T	P	Store ↓	Domain	ID	Name

Icon: S M L

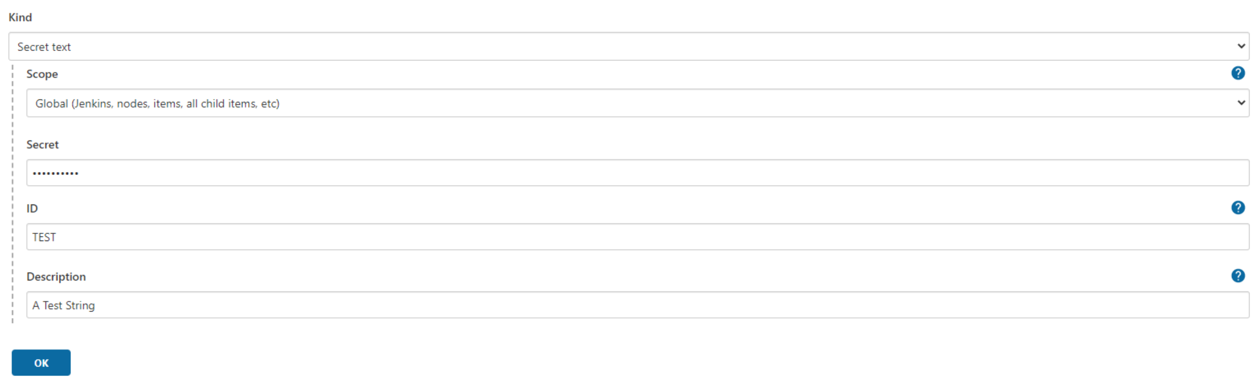
Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	(global)

Now we can add our credentials. Select the `Add Credentials` from the left-hand side.

We are going to add the following credential:

- Type: Secret Text
- Scope: Global
- Secret: TestString
- ID: TEST
- Description: A Test String


Kind: Secret text

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

Secret: ••••••••••

ID: TEST

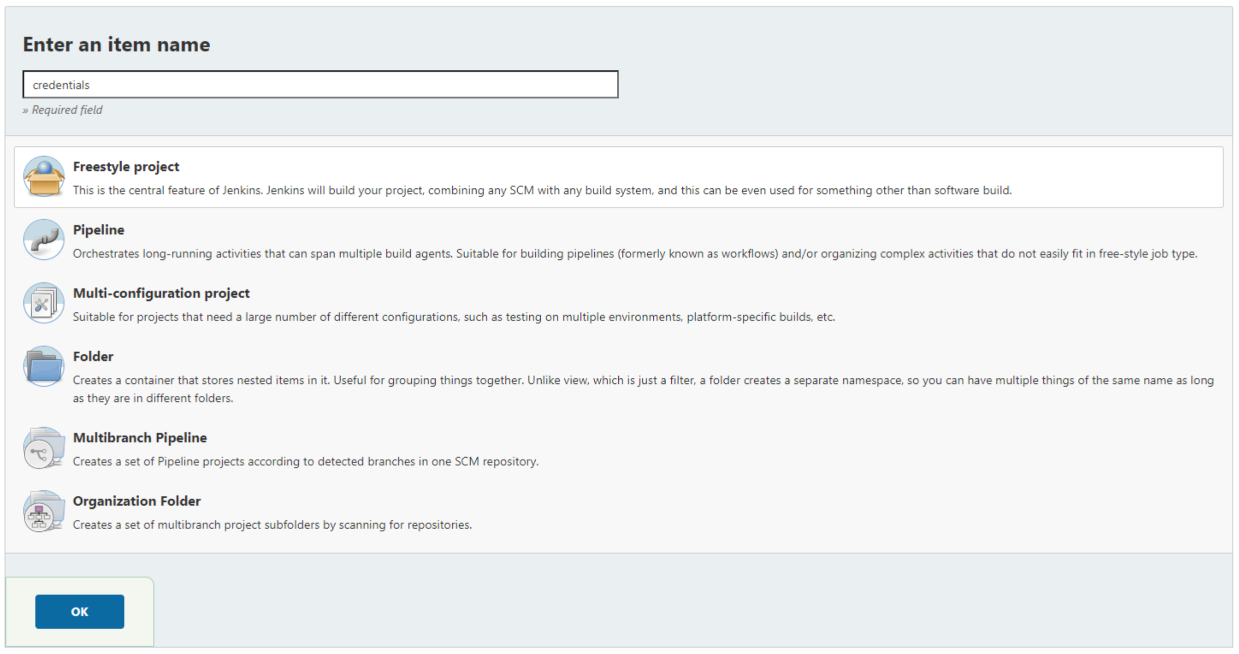
Description: A Test String

OK

You should now be able to see your credential stored on Jenkins.

Use the Credential

Head back to the Jenkins Dashboard and this time, click `New Item` from the left-hand menu.
We will be creating a `freestyle project` with the name `credentials`.


Enter an item name

credentials

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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.

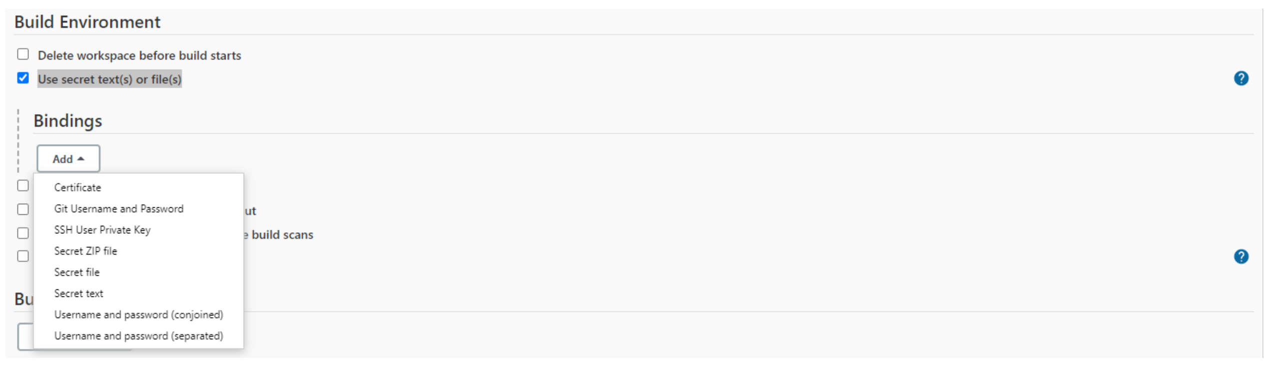
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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.

You can set up this project however you want but, importantly, we must configure the `Build Environment` to have a variable.

Select `Use secret text(s) or file(s)` then click `add` and finally `Secret text`.

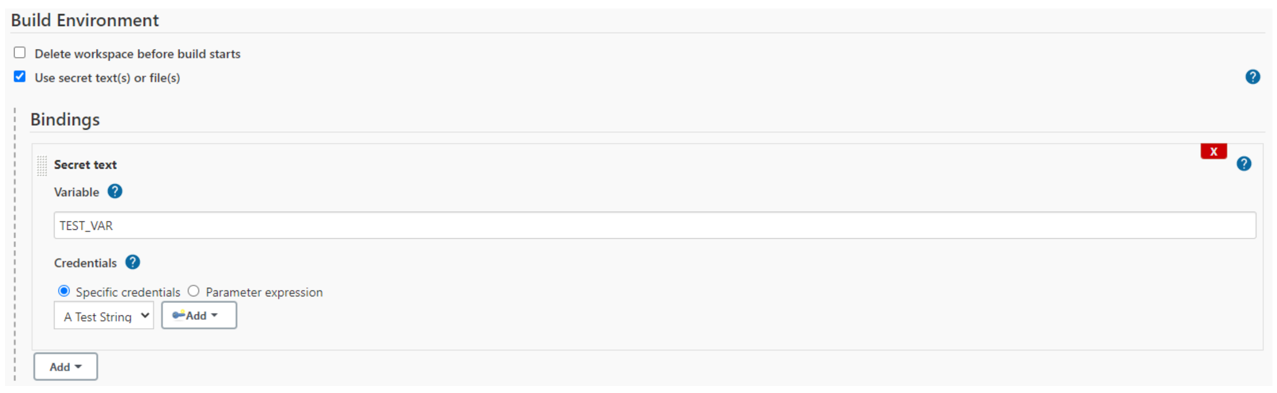

Build Environment

☐ Delete workspace before build starts
☑ Use secret text(s) or file(s)

Bindings

Add
Certificate
Git Username and Password
SSH User Private Key
Secret ZIP file
Secret file
Secret text
Username and password (conjoined)
Username and password (separated)

Then create an environment variable with the name `TEST_VAR` and make sure you select the value to be from the credential we just created (it should show the description).


Build Environment

☐ Delete workspace before build starts
☑ Use secret text(s) or file(s)

Bindings

Secret text

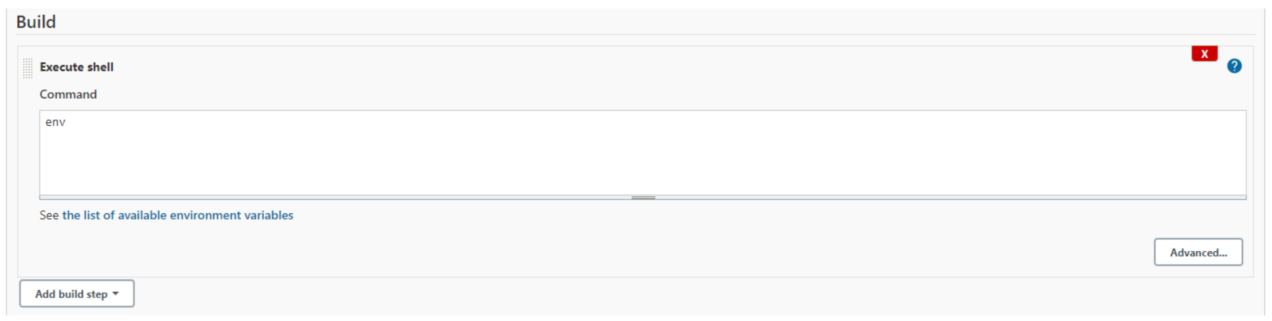
Variable: TEST_VAR

Credentials: [✓ Specific credentials] Parameter expression

A Test String

The last bit of setup is to configure the job to output all environment variables we can use.

Head to the `Build step` and include an `Execute shell`.
In that, enter the command `env`.


Build

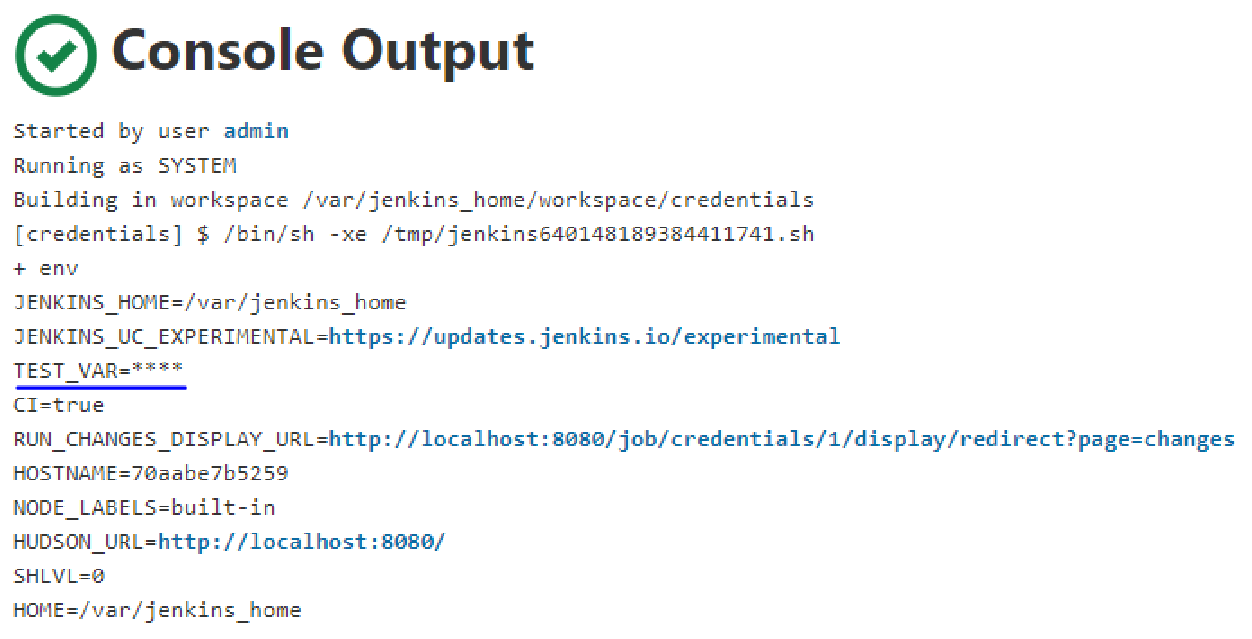
Execute shell

Command

```
env
```

See the list of available environment variables

Finally, we can test that the variable is available to us.
The command `env` shows all exported environment variables that we can use during our build step.
Scroll down until you find the variable you just created.



Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/credentials
[credentials] $ /bin/sh -xe /tmp/jenkins640148189384411741.sh
+ env
JENKINS_HOME=/var/jenkins_home
JENKINS_UC_EXPERIMENTAL=https://updates.jenkins.io/experimental
TEST_VAR=****
CI=true
RUN_CHANGES_DISPLAY_URL=http://localhost:8080/job/credentials/1/display/redirect?page=changes
HOSTNAME=70aabe7b5259
NODE_LABELS=built-in
HUDSON_URL=http://localhost:8080/
SHLVL=0
HOME=/var/jenkins_home
```

Clean Up

Head back to the `Project Dashboard` and select `Delete Project`.

Head back to the `Manage Jenkins` > `Manage Credentials`, select your `TEST` credential and then select `Delete` from the left-hand menu.

Exercises

Create a new freestyle project and set it up to use a **private** GitHub repository.

To do this Jenkins will need to authenticate with GitHub.
Authentication with GitHub must now be done with either `SSH keys` or `Personal Access Tokens`, as of August 2021.

If we want Jenkins to authenticate with an SSH key, we will want to store the credential as an `SSH Username with private key`.

If we want Jenkins to authenticate with a Personal Access Token, we will want to store the credential as `Username with password`.

> Note: When you have finished this exercise, please remove the unused SSH Key or Personal Access Token from GitHub.