

**NAME**

gitcredentials - providing usernames and passwords to Git

**SYNOPSIS**

```
git config credential.https://example.com.username
myusername
git config credential.helper $helper $options
```

**DESCRIPTION**

Git will sometimes need credentials from the user in order to perform operations; for example, it may need to ask for a username and password in order to access a remote repository over HTTP. This manual describes the mechanisms Git uses to request these credentials, as well as some features to avoid inputting these credentials repeatedly.

**REQUESTING CREDENTIALS**

Without any credential helpers defined, Git will try the following strategies to ask the user for usernames and passwords:

1. If the `GIT_ASKPASS` environment variable is set, the program specified by the variable is invoked. A suitable prompt is provided to the program on the command line, and the user's input is read from its standard output.
2. Otherwise, if the `core.askpass` configuration variable is set, its value is used as above.
3. Otherwise, if the `SSH_ASKPASS` environment variable is set, its value is used as above.
4. Otherwise, the user is prompted on the terminal.

**AVOIDING REPETITION**

It can be cumbersome to input the same credentials over and over. Git provides two methods to reduce this annoyance:

1. Static configuration of usernames for a given authentication context.
2. Credential helpers to cache or store passwords, or to interact with a system password wallet or keychain.

The first is simple and appropriate if you do not have secure storage available for a password. It is generally configured by adding this to your config:

```
[credential ] -P -- https://example.com
username = me
```

Credential helpers, on the other hand, are external programs from which Git can request both usernames and passwords; they typically interface with secure storage provided by the OS or other programs.

To use a helper, you must first select one to use. Git currently includes the following helpers:

**cache**

Cache credentials in memory for a short period of time. See [git-credential-cache\(1\)](#) for details.

**store**

Store credentials indefinitely on disk. See [git-credential-store\(1\)](#) for details.

You may also have third-party helpers installed; search for `credential-*` in the output of `git help -a`, and consult the documentation of individual helpers. Once you have selected a helper, you can tell Git to use it by putting its name into the `credential.helper` variable.

1. Find a helper.
 

```
$ git help -a | grep credential-
credential-foo
```
2. Read its description.
 

```
$ git help credential-foo
```

3. Tell Git to use it.

```
$ git config --global credential.helper foo
```

If there are multiple instances of the `credential.helper` configuration variable, each helper will be tried in turn, and may provide a username, password, or nothing. Once Git has acquired both a username and a password, no more helpers will be tried.

## CREDENTIAL CONTEXTS

Git considers each credential to have a context defined by a URL. This context is used to look up context-specific configuration, and is passed to any helpers, which may use it as an index into secure storage.

For instance, imagine we are accessing <https://example.com/foo.git>. When Git looks into a config file to see if a section matches this context, it will consider the two a match if the context is a more-specific subset of the pattern in the config file. For example, if you have this in your config file:

```
[credential ] -P -- https://example.com
username = foo
```

then we will match: both protocols are the same, both hosts are the same, and the pattern URL does not care about the path component at all. However, this context would not match:

```
[credential ] -P -- https://kernel.org
username = foo
```

because the hostnames differ. Nor would it match `foo.example.com`; Git compares hostnames exactly, without considering whether two hosts are part of the same domain. Likewise, a config entry for <http://example.com> would not match: Git compares the protocols exactly.

## CONFIGURATION OPTIONS

Options for a credential context can be configured either in `credential.*` (which applies to all credentials), or `credential.<url>.*`, where `<url>` matches the context as described above.

The following options are available in either location:

### helper

The name of an external credential helper, and any associated options. If the helper name is not an absolute path, then the string `git credential-` is prepended. The resulting string is executed by the shell (so, for example, setting this to `foo --option=bar` will execute `git credential-foo --option=bar` via the shell. See the manual of specific helpers for examples of their use.

### username

A default username, if one is not provided in the URL.

### useHttpPath

By default, Git does not consider the path component of an http URL to be worth matching via external helpers. This means that a credential stored for <https://example.com/foo.git> will also be used for <https://example.com/bar.git>. If you do want to distinguish these cases, set this option to true.

## CUSTOM HELPERS

You can write your own custom helpers to interface with any system in which you keep credentials. See the documentation for Git's [credentials API](#)<sup>[1]</sup> for details.

## GIT

Part of the [git\(1\)](#) suite

## NOTES

1. `credentials API`  
<file:///usr/share/doc/git/html/technical/api-credentials.html>

