



Architecture & Deployment

2025-2026 v0.1.0 on branch main Rev: bf5a3ed8baf85ebafdf2c8031e836d37fa6b3121

Git Hooks

Learn the basics of Git hooks.

You will need

- A Unix CLI
- [Git](#)

Recommended reading

- [Git Introduction](#)
- [Git Branching](#)
- [Collaborating with Git](#)
- [Collaborating with Git](#)
- [Shell Scripting](#)

What is a Git hook?

A Git hook is **an executable file** that can be **triggered when certain actions occur**. Hooks can be any kind of executable file, e.g. a compiled binary program, a shell script, a Ruby script, a Python script, etc.

Hooks must be executable files with a specific name in the `hooks` directory inside a repository's Git directory (inside the `.git` directory in a normal repository).

You can find example hook files in most Git repositories:

```
$> cd /path/to/a/git/repository
```

```
$> cat .git/hooks/pre-commit.sample
```

```
#!/bin/sh
```

```
#
```

```
# An example hook script to verify what is about to be committed.
```

```
...
```

More information

These example files are not triggered because they are not executable and do not have the right name.

What kind of hooks are there?

Hook scripts must have a specific name (and be executable) to be triggered.

These hook scripts are only limited by a developer's imagination. Some example hook scripts include:

Hook	Example usage
<code>pre-commit</code>	Check the commit message for spelling errors.
<code>pre-receive</code>	Enforce project coding standards.
<code>post-commit</code>	Email/SMS team members of a new commit.
<code>post-receive</code>	Push the code to production.

You can type the `git help hooks` command to get the full list of hooks and their documentation. A full list is also available [here](#).

How can I create a hook?

Create a new repository:

```
$> cd /path/to/projects
$> mkdir git-hook-example
$> cd git-hook-example
$> git init
```

Create a hook script named `pre-commit` that will be executed before every commit:

```
$> echo '#!/bin/bash' > .git/hooks/pre-commit
$> echo 'echo You are about to commit...' >> .git/hooks/pre-commit
$> chmod +x .git/hooks/pre-commit
```

You now have a `pre-commit` hook script:

```
$> cat .git/hooks/pre-commit
#!/bin/bash
echo You are about to commit...
```

Hooks are not under version control

Note that hooks are not part of the working tree and thus **cannot be put under version control**:

```
$> git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```



More information

This is a good thing, as you don't want people to be able to send malicious hook scripts to your machine every time you clone a repository from the Internet.

How do I use a hook?

Following the previous example, make a commit and the `pre-commit` hook will be automatically triggered:

```
$> echo content > file.txt
$> git add .
$> git commit -m "Test"
```

You are about to commit...

```
[main 14b306d] Test
1 file changed, 1 insertion(+), 1 deletion(-)
```

What are hooks good for?

Hooks can be a way to automate repetitive or mundane tasks, like removing trailing spaces at each commit.

Or they can be used to automate deployments, for example to run the latest version of a server-side application every time new commits are pushed.

References

- [Pro Git Book](#)
 - [Customizing Git - Git Hooks](#)

- [Git Hooks](#)

Table of contents

- [What is a Git hook?](#)
 - [What kind of hooks are there?](#)
 - [How can I create a hook?](#)
 - [Hooks are not under version control](#)
 - [How do I use a hook?](#)
 - [What are hooks good for?](#)
- [References](#)

[↑](#) Back to top