< Back to Terminus

GIT

Git Clone, Push, And Pull Over SSH

Last Updated on 2023-04-26

CONTENTS

CONTRIBUTORS

Razvan Ludosanu
Founder, learnbackend.dev

RELATED TOPICS

Adding a Submodule in Git
Undo a git push
Undo a Git Merge
Undo Git Add
Undo a Git Rebase

The short answer

To clone a Git repository using the SSH protocol, you can use the `git clone` command with a valid SSH URL as follows:

```
$ git clone git@host:username/repository.git
```

Where:

- `host` represents the domain name or the IP address of the hosting server.
- `username` represents your user account.
- `repository` represents the name of the Git repository you want to clone.

For example:

```
$ git clone git@github.com:johndoe/my-app.git
```

Cloning with SSH vs. HTTPS

The main difference between cloning a remote repository with SSH and HTTPS is the way the authentication is handled.

When using HTTPS, Git will prompt you for your username and password during the authentication process.

On the other hand, when using SSH, Git uses your SSH key to authenticate, which means that you don't need to send your credentials over the network.

In that sense, SSH is a more secure method for cloning repositories and pushing / pulling commits, as only the machines with the key file on disk are able to access the repositories.

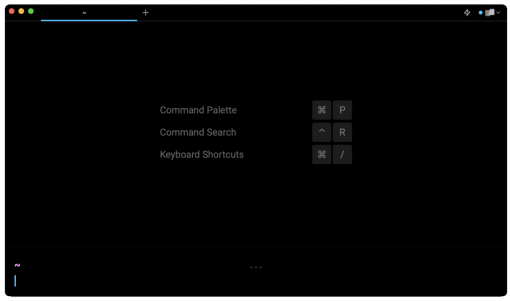
Moreover if the SSH key file was to be stolen, it won't give access to the account itself (unlike the credentials) and can be easily revoked.

Set up SSH for Git

In order to be able to clone a remote Git repository using the SSH protocol, you will have to create a new SSH key pair on your local machine, and add this key to your Git hosting service. We will be using GitHub as our Git hosting service in the following examples, but others will work just as well.

Using Warp's AI to quickly retrieve these steps

If you're using Warp as your terminal, you can easily retrieve an approximation of the steps described below using Warp's AI feature:



For that:

1. Click on the bolt icon on the top right of the terminal
2. Type in your question; for example: "How do I add a new SSH key to my Github account".
3. Press `ENTER` to generate an answer.

As with any AI-powered tool, use it as a starting point and not a final answer. We'll dig into more depth in our human-powered writeup below.

Step 1: Generate a new SSH key with ssh-keygen

To generate a new SSH key pair on your local machine, you can use the `ssh-keygen` command as follows:

```
$ ssh-keygen -t ed25519 -C “user@example.com”
```

Where

- The `-t` flag is used to specify the type of key to create, in this case, an ed25519, which is a popular type of public-key cryptography.
- The `-C` flag is used to provide additional information about the key, such as its purpose or the user who generated it.

When prompted with this question, simply press `ENTER` to validate the recommended file path the key pair will be saved at, or type in another path if a similar file already exists in the `.ssh` directory.

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/johndoe/.ssh/id_ed25519):
```

To the following questions, type in a passphrase to secure your key, and press `ENTER` once again to complete the process.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

You should now see a similar output in your terminal window confirming that the key was successfully generated:

```
SHA256:ToTEp33dDV8Sokslnx568DC5ABPQTmvlBjGx+r/W0k8 user@example.com
The key's randomart image is:
+--[ED25519 256]--+
|o.+o ..          |
|. +o....+        |
| +.=  .o.        |
|o.+ o . .  .      |
|o+ . . .So .     |
|*.    oo+ o .    |
|o*.  o =E= o     |
|ooo.o =..o o      |
|.o+o++ ..        |
+----[SHA256]-----+
```

And you should be able to find the following files in your `.ssh` directory:

```
$ ls ~/.ssh
id_ed25519  id_ed25519.pub
```

Where:

- The `id_ed25519` file is your private key whose content should be kept confidential.
- The `id_ed25519.pub` file is your public key that we'll use in the next step.

Step 2: Add a public key to your GitHub account

To add a public key to your GitHub account, first display the content of the **public** key file you've just created using the `cat` command, and copy it to your clipboard using `CTRL+C` (or `CMD+C` on MacOS).

```
$ cat ~/.ssh/id_ed25519.pub
```

Alternatively, you can use the `pbcopy` command on MacOS as follows:

```
$ pbcopy < ~/.ssh/id_ed25519.pub
```

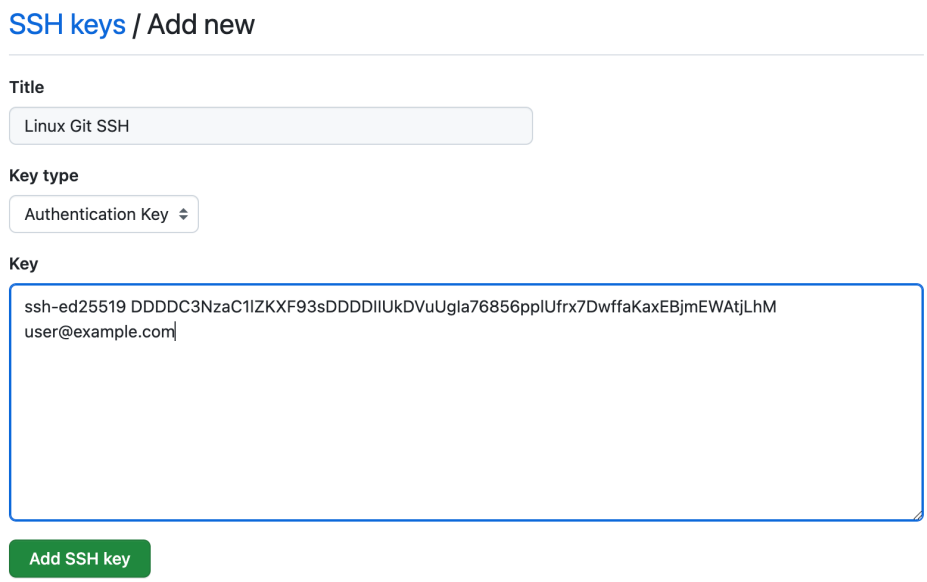
Once you've done that:

1. Log in to your GitHub account.
2. Navigate to “Settings”.
3. Click on “SSH and GPG keys” in the left menu
4. Click on the “New SSH key” button.

Or directly follow this link https://github.com/settings/ssh/new.

From here:

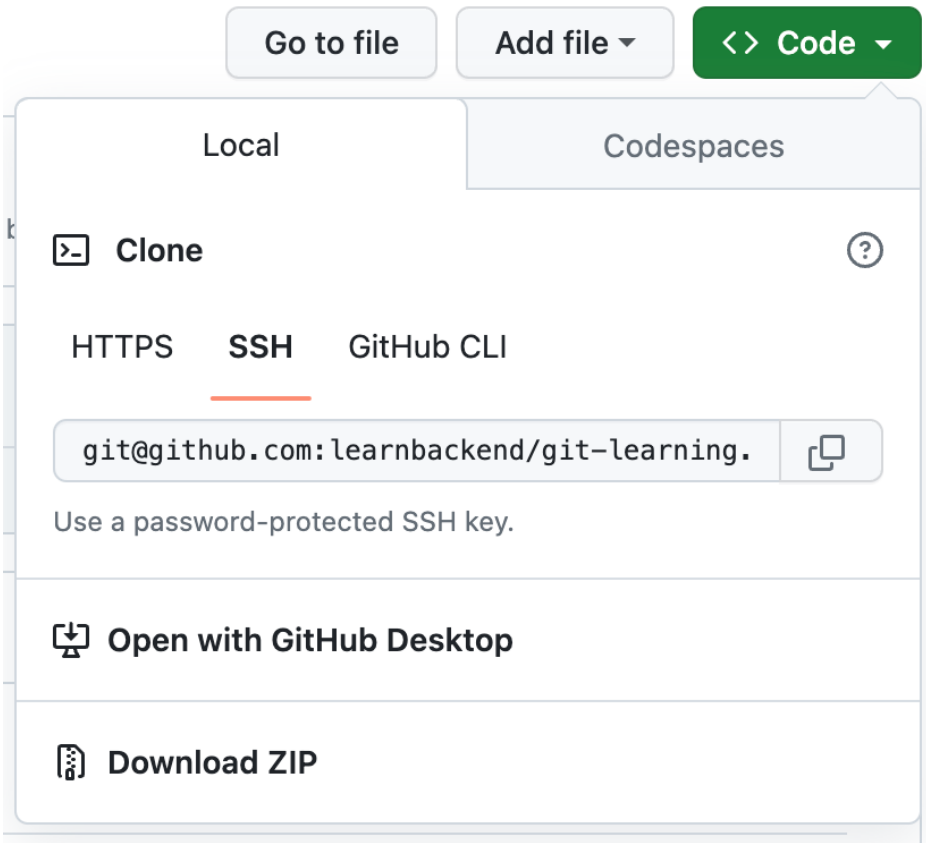
1. Add a short descriptive title in the `Title` field.
2. Paste the public key in the `Key` field
3. Click on the "Add SSH key" button to finalize the process.



Step 3: Clone a repository

To verify that the SSH key you've just added to your account works:

1. Navigate to the repository you wish to clone.
2. Click on the "Code" button.
3. Copy the URL located under the "SSH" tab.



4. Run the `git clone` command in your terminal with the URL you just copied.

Step 4: Specify the SSH key to the ssh-agent

If you don't want to type in your password every time your SSH key is used by Git, you can add your key to the list of keys managed by the SSH agent. To do so, first make sure that the SSH agent is running using the following command:

```
$ eval "$(ssh-agent -s)"
```

Then add your private key file using the following `ssh-add` command:

```
$ ssh-add ~/.ssh/id_ed25519
```

`git` pushing and pulling commits with SSH

Once you've cloned a Git repository on your local machine with SSH, every commit you push and pull will automatically go through the SSH protocol, with no additional configuration steps required.