

[kamarada.github.io](https://kamarada.github.io)

# Using Git with SSH keys

*Antônio Vinícius*

13-16 minutos

Today's post is for developers. If you use the [Git](#) version control system with a service such as [GitHub](#), [GitLab](#) or [Bitbucket](#) to host and manage your projects source codes, you know that by default Git connects to remotes using the [HTTPS](#) protocol, which requires you to enter username and password every time you run a command such as `git pull` or `git push`.

Using the [SSH](#) protocol, you can connect and authenticate to servers to use their services. The three mentioned services allow Git to connect via SSH instead of HTTPS. Connecting with public key encryption dispenses typing username and password for every Git command.

You are going to see in this post how to use GitHub, GitLab and Bitbucket with SSH.



## Make sure an SSH client is installed

In order to connect using the SSH protocol, an SSH client must be

installed on your system. If you use [openSUSE](#), it should be already installed by default.

Just to make sure, open the terminal and run:

That command should output the version number of the SSH client being used:

```
1 OpenSSH_7.9p1, OpenSSL 1.1.0i-fips 14 Aug 2018
```

In case the system informs that the **ssh** command was not found, you can install the [OpenSSH](#) client running:

## Check for existing SSH keys

To connect using the SSH protocol, you need an SSH key pair (one private and the other public). If you have never used SSH, you can safely skip this topic and move on to the next. If you have ever used SSH (for instance, to [remotely access a server](#)), probably you already have an SSH key pair, in which case you don't need to generate a new key pair.

To see if existing SSH keys are present, run:

That command should list the contents of the `~/ .ssh` folder, in which the SSH client stores its configuration files:

```
1 total 28K
2 drwx----- 2 vinicius users 94 Mar 17
3 14:55 .
4 drwxr-xr-x 54 vinicius users 4.0K Jul 14
5 02:44 ..
6 -rw----- 1 vinicius users 2.5K Mar 1
7 23:41 authorized_keys
8 -rw-r--r-- 1 vinicius users 39 Dec 19
   2018 config
   -rw----- 1 vinicius users 3.3K Jul 18
   2018 id_rsa
```

```
-rw-r--r--  1 vinicius users  748 Jul 18
2018 id_rsa.pub
-rw-r--r--  1 vinicius users 4.7K Jul  5
01:57 known_hosts
```

If you receive an error that there is no `~/ .ssh` directory or there are no files in it, don't worry: it means you haven't created an SSH key pair yet. If that is the case, proceed to the next topic.

By default, public SSH keys are named:

- `id_dsa.pub`;
- `id_ecdsa.pub`;
- `id_ed25519.pub`; or
- `id_rsa.pub`.

Inside my `~/ .ssh` folder, I have an SSH key pair (`id_rsa.pub` is the public key and `id_rsa` is the private key) created a year ago (Jul 18 2018).

For security reasons, it is recommended that you generate a new SSH key pair at least once a year. If you already have an SSH key pair that was created more than a year ago, it is recommended that you proceed to the next topic.

If you already have an SSH key pair and want to reuse it, you can skip the next topic.

## Generate a new SSH key pair

To generate a new SSH key pair, run the following command (replace `your_email@example.com` with your email address):

```
1 $ ssh-keygen -t rsa -b 4096 -C
2 "your_email@example.com"
3
4 Generating public/private rsa key pair.
   Enter file in which to save the key (/home
```

```
/your_user_name/.ssh/id_rsa):
```

It asks you where to save the private key (`id_rsa`).

Press **Enter** to accept the default location.

If you already have a private key, it asks whether it should overwrite:

```
1 /home/your_user_name/.ssh/id_rsa already
2 exists.
  Overwrite (y/n)?
```

If that happens, type `y` and press **Enter**.

Then, enter and re-enter a [passphrase](#) (think of it as a kind of password):

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

The SSH key pair is created in `~/.ssh`.

The whole interaction should look similar to the following:

```
1 your_user_name@your_host_name:~> ssh-keygen -t rsa
2 -b 4096 -C "your_email@example.com"
3 Generating public/private rsa key pair.
4 Enter file in which to save the key (/home
5 /your_user_name/.ssh/id_rsa):
6 /home/your_user_name/.ssh/id_rsa already exists.
7 Overwrite (y/n)? y
8 Enter passphrase (empty for no passphrase):
9 Enter same passphrase again:
10 Your identification has been saved in
11 /home/your_user_name/.ssh/id_rsa.
12 Your public key has been saved in
13 /home/your_user_name/.ssh/id_rsa.pub.
14 The key fingerprint is:
```

```
15 SHA256:CEnY8F0QmvISJpVp6oAlITemk1aWKRdVi0FePP6/CKk
16 your_email@example.com
17 The key's randomart image is:
18 +---[RSA 4096]----+
19 |o.=@X++          |
20 |o*@0++           |
21 |=Bo+==+         |
22 |0o+ oo..        |
23 |=+ . . . S      |
24 |...  o           |
   | .    o .       |
   |      . . o      |
   |   E    . o.     |
   +----[SHA256]-----+
your_user_name@your_host_name:~>
```

## Add the private SSH key to the ssh-agent

If you don't want to type your passphrase each time you use your SSH keys, you need to add it to the [ssh-agent](#), which is a program that runs in background while you are logged in to the system and stores your keys in memory.

To start the **ssh-agent** in background, run the following:

```
1 $ eval "$(ssh-agent -s)"
```

That command outputs the **ssh-agent** [process identifier](#):

Then, add your SSH private key to the **ssh-agent**:

```
1 $ ssh-add ~/.ssh/id_rsa
```

Type your passphrase and press **Enter**:

```
1 Enter passphrase for /home/your_user_name
  ~/.ssh/id_rsa:
```

The command confirms that the private SSH key has been added to the **ssh-agent**:

```
1 Identity added: /home/your_user_name  
  /.ssh/id_rsa (your_email@example.com)
```

## Add the public SSH key to your account

Once you have an SSH key and have added it to the **ssh-agent**, you can set up connecting via SSH. Let's see how to do that for each of the three servers: GitHub, GitLab and Bitbucket.

In all the three cases, the process is similar. Start by copying your public SSH key (`~/.ssh/id_rsa.pub`) file contents to the [clipboard](#) using the **xclip** command:

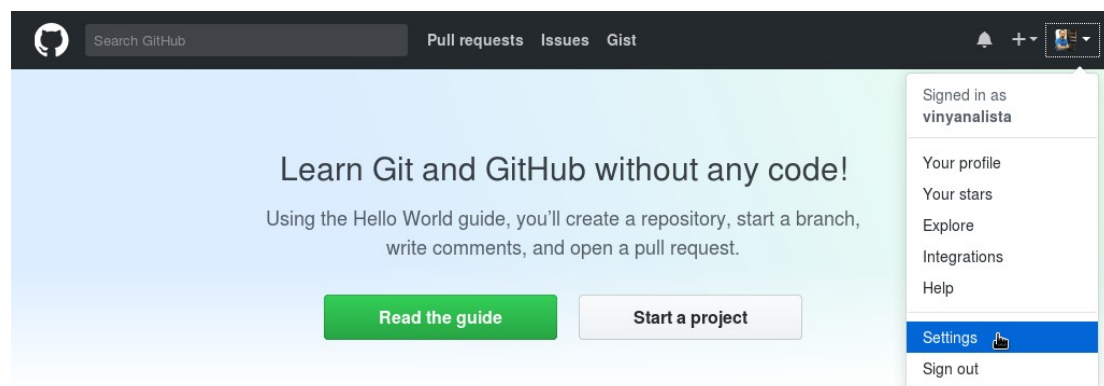
```
1 $ xclip -sel clip < ~/.ssh/id_rsa.pub
```

[xclip](#) is a command line utility that allows access to the graphical interface clipboard from the terminal. If it is not installed, you can install it running:

## GitHub

Using a browser, go to the GitHub home page at [github.com](https://github.com) and sign in to your account.

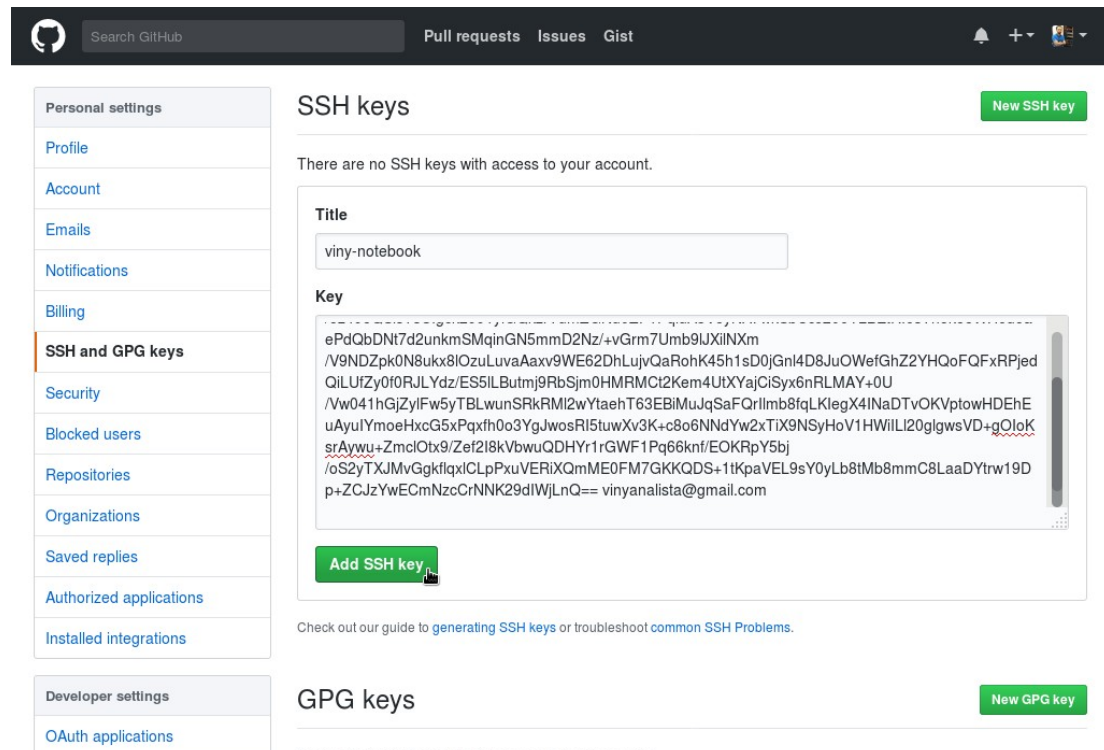
In the upper-right corner of the page, click your profile photo, then click **Settings**:



In the user settings sidebar, click **SSH and GPG keys**. Then click

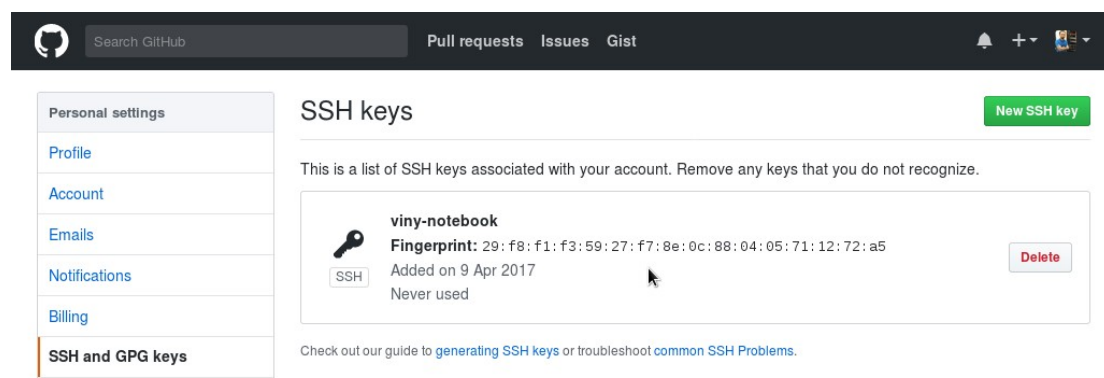
## New SSH key.

Fill in the **Title** field with a descriptive label for the new key (for example, the name of your computer) and paste your public key into the **Key** field. Finally, click **Add SSH key**:



The screenshot shows the GitHub 'SSH keys' page. On the left is a sidebar with 'Personal settings' (Profile, Account, Emails, Notifications, Billing, SSH and GPG keys, Security, Blocked users, Repositories, Organizations, Saved replies, Authorized applications, Installed integrations) and 'Developer settings' (OAuth applications). The main content area is titled 'SSH keys' and has a 'New SSH key' button. It states 'There are no SSH keys with access to your account.' Below this is a form with a 'Title' field containing 'viny-notebook' and a 'Key' field containing a long public key. An 'Add SSH key' button is at the bottom of the form. Below the form is a link to a guide: 'Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).' Below the SSH keys section is a 'GPG keys' section with a 'New GPG key' button and the text 'There are no GPG keys with access to your account.'

Now the key appears in the list of SSH keys associated with your account:



The screenshot shows the GitHub 'SSH keys' page after adding a key. The sidebar is the same. The main content area is titled 'SSH keys' and has a 'New SSH key' button. It states 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' Below this is a list of keys. One key is shown: 'viny-notebook' with a key icon, fingerprint '29:f8:f1:f3:59:27:f7:8e:0c:88:04:05:71:12:72:a5', added on '9 Apr 2017', and status 'Never used'. There is a 'Delete' button next to it. Below the list is the same link to the guide: 'Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).'

## GitLab

Using a browser, go to the GitLab home page at [gitlab.com](https://gitlab.com) and sign in to your account.

In the upper-right corner of the page, click your profile photo, then click **Settings**:



The screenshot shows the top navigation bar of the GitLab website. It includes the GitLab logo, a 'Projects' dropdown menu, and links for 'Groups', 'Activity', 'Milestones', and 'Snippets'. There is a search bar with the text 'Search or jump to...'. On the right side, there are icons for notifications, a user profile photo, and a dropdown menu.

**Welcome to GitLab**

Code, test, and deploy together

**Create a project**

Projects are where you store your code, access issues, wiki and other features of GitLab.

**Create a group**

Groups are the best way to manage your members.

Medeiros  
@vinyanalista

Set status

Profile

Settings

Sign out

In the **User Settings** sidebar, click **SSH Keys**.

Paste your public key in the **Key** field. Fill in the **Title** field with a descriptive label for the new key (for example, the name of your computer). Finally, click **Add key**:

**GitLab** Projects Groups Activity Milestones Snippets

User Settings > SSH Keys

### SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

**Add an SSH key**  
To add an SSH key you need to [generate one](#) or use an [existing key](#).

**Key**  
Paste your public SSH key, which is usually contained in the file '~/.ssh/id\_ed25519.pub' or '~/.ssh/id\_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

```
KKiUmWYgqJ09Cw3PwqB+qL1yzCUNzKsbGjVrbatH/gSty86U/LXnN334KAH/CUX8L+MX3
7req5t8lQT86R/gw/rFxQl6HFq0B/wjdgCE4C+EqJC4BqVPxGNLqUxxFVDD50aXVik2vtIWx
/Vg2p9v4oEaYrZAPcxHv58RRHq3bHaWioty+lp+oQ4yVQaLMnmWuHvf7uZrPQOS9ujSkdR
G0tYcu5RVL+3N584PO5W0D/ej5Lz38FzoJ8b98mCjaRzOopGJoVtjko7JQROUpbZsbV9bba9
XTCsAZsgip9szMyncxjgLB8vVGF5lI68H3D+dfleHPH0mxxEBdsav0S1ReeJMxQCC80L53tbFp
ymAyiQMpsXPYXnH4xwWkS7MsfKZRUAUqjvLmxiUW6kKxj9VPFf23LBafJgTj8M+KS7yQK+
SPEES7KswqoFn9rTKfDRazE/ORT+JbDI4KuzbVilIG/s6+4R5ZDF5Tcmw==
vinyanalista@gmail.com
```

**Title**  
viny-notebook  
Name your individual key via a title

**Add key**

**Your SSH keys (0)**  
There are no SSH keys with access to your account

Now the key appears in the list of SSH keys associated with your account:

**GitLab** Projects Groups Activity Milestones Snippets

User Settings > SSH Keys > viny-notebook

SSH Key
<p><b>Title:</b> viny-notebook</p> <p><b>Created on:</b> Jul 2, 2019 11:57am</p> <p><b>Last used on:</b> N/A</p>

**Fingerprint:** e7:8f:c0:2c:1e:ad:11:35:f2:30:4f:80:33:72:0f:42

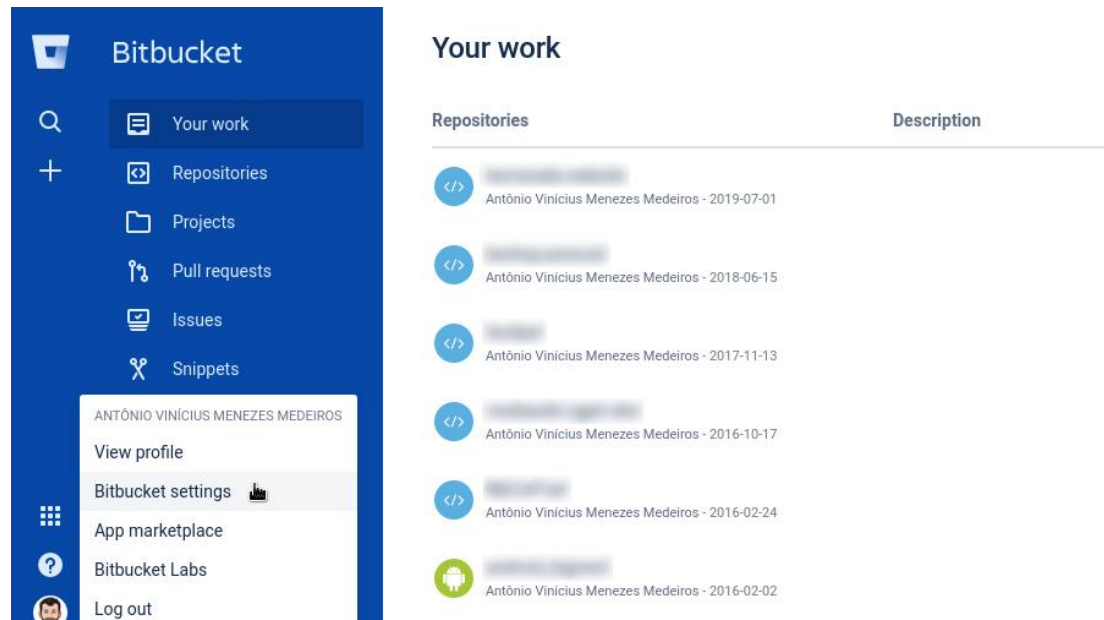
```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCsQVMjT8jBB5T3J4a0yA8YHFz7y79Da3t1Vh:
```

**Remove**



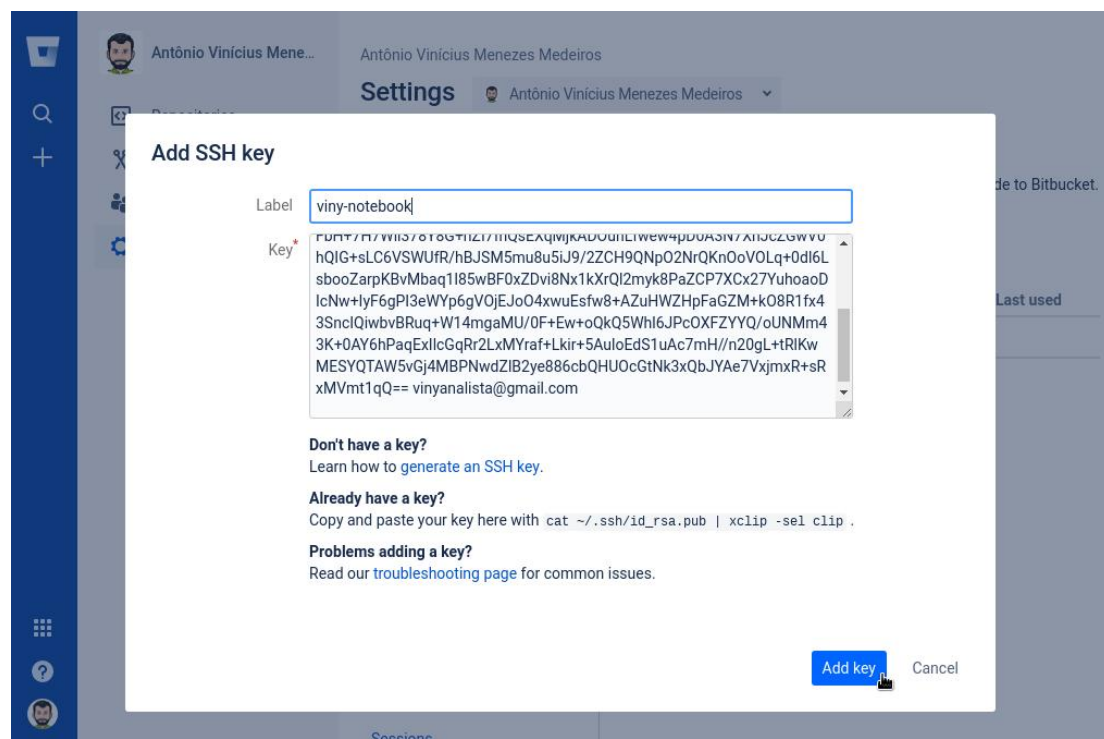
Using a browser, go to the Bitbucket home page at [bitbucket.org](https://bitbucket.org) and log in to your account.

In the lower-left corner of the page, click your profile photo, then click **Bitbucket settings**:

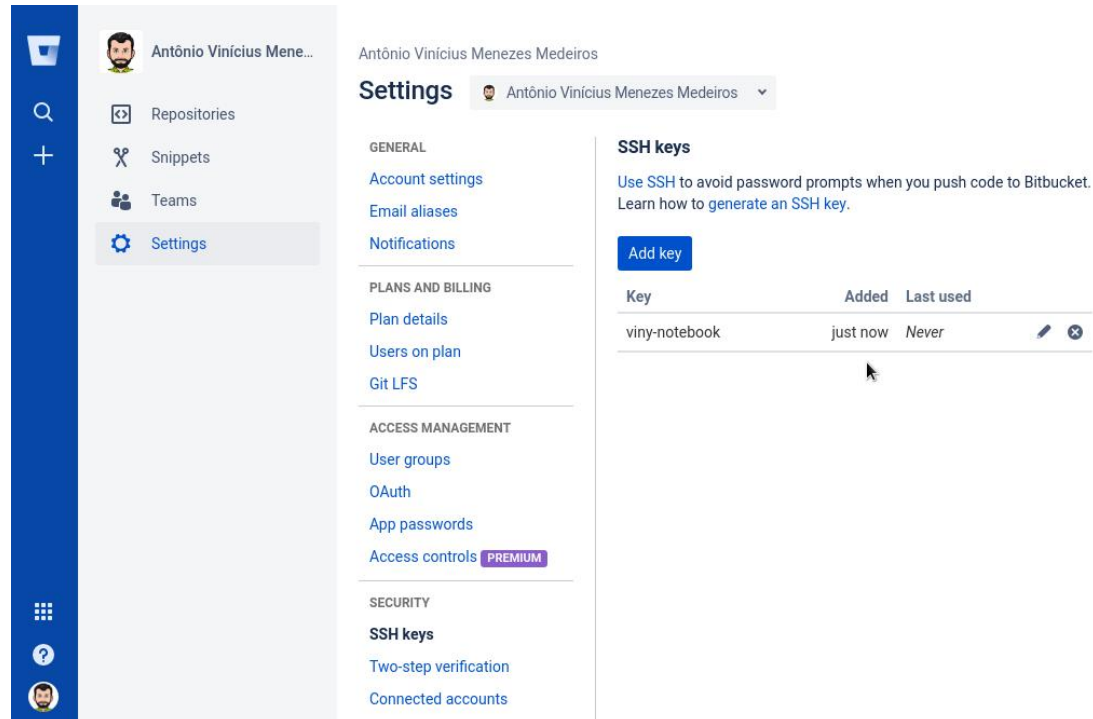


In the **Settings** sidebar, in the **Security** section, click **SSH keys**. Then, click **Add key**.

Fill in the **Label** field with a descriptive label for the new key (for example, the name of your computer) and paste your public key into the **Key** field. Finally, click **Add key**:



Now the key appears in the list of SSH keys associated with your account:



## Test connecting via SSH

GitHub, GitLab and Bitbucket allow you to test whether SSH connection has been set up correctly before actually using it with Git.

### GitHub

After you've added your SSH key to your GitHub account, open the terminal and run:

```
1 $ ssh -T git@github.com
```

That command attempts an [SSH remote access](#) to GitHub.

If that is the first time you connect to GitHub via SSH, the SSH client asks you if it can trust the public key of the GitHub server:

```
1 The authenticity of host 'github.com
2 (140.82.113.4)' can't be established.
3 RSA key fingerprint is
  SHA256:nThbg6kXUpJWGL7E1IG0CspRomTxdCARLviKw6E5SY8.
```

```
Are you sure you want to continue connecting
(yes/no)?
```

Type **yes** and press **Enter**. The SSH client adds GitHub to the list of trusted hosts:

```
1 Warning: Permanently added
  'github.com,140.82.113.4' (RSA) to the list
  of known hosts.
```

Once added to the list of known hosts, you won't be asked about GitHub's public key again.

As this remote access via SSH is provided by GitHub just for testing, not for actual use, the server informs that you have successfully authenticated and terminates the connection:

```
1 Hi your_user_name! You've successfully
  authenticated, but GitHub does not provide
  shell access.
```

If you completed the test successfully, now you can use SSH with GitHub.

The whole interaction should look similar to the following:

```
1 your_user_name@your_host_name:~> ssh -T
2 git@github.com
3 The authenticity of host 'github.com
4 (140.82.113.4)' can't be established.
5 RSA key fingerprint is
6 SHA256:nThbg6kXUpJWG17E1IG0CspRomTxdCARLviKw6E5SY8.
7 Are you sure you want to continue connecting
  (yes/no)? yes
Warning: Permanently added
  'github.com,140.82.113.4' (RSA) to the list of
  known hosts.
Hi your_user_name! You've successfully
```

```
authenticated, but GitHub does not provide shell
access.
your_user_name@your_host_name:~>
```

## GitLab

If you have added your SSH key to your GitLab account, the test is very similar:

```
1 $ ssh -T git@gitlab.com
2
3 The authenticity of host 'gitlab.com
4 (35.231.145.151)' can't be established.
5 ECDSA key fingerprint is
6 SHA256:HbW3g8zUjNSksFbqTiUWPWg2Bq1x8xdGUrliXFzSnUw.
7 Are you sure you want to continue connecting
  (yes/no)? yes
Warning: Permanently added
'gitlab.com,35.231.145.151' (ECDSA) to the list of
known hosts.
Welcome to GitLab, @your_user_name!
```

If you completed the test successfully, now you can use SSH with GitLab.

## Bitbucket

If you have added your SSH key to your Bitbucket account, the test is very similar:

```
1 $ ssh -T git@bitbucket.org
2
3 The authenticity of host 'bitbucket.org
4 (104.192.143.1)' can't be established.
5 RSA key fingerprint is
6 SHA256:zzXQ0XSRBEiUtuE8AikJYKwbHaxvSc0ojez9YXaGp1A.
```

```
7 Are you sure you want to continue connecting
8 (yes/no)? yes
9 Warning: Permanently added
  'bitbucket.org,104.192.143.1' (RSA) to the list of
  known hosts.
  logged in as your_user_name.

You can use git or hg to connect to Bitbucket.
Shell access is disabled.
```

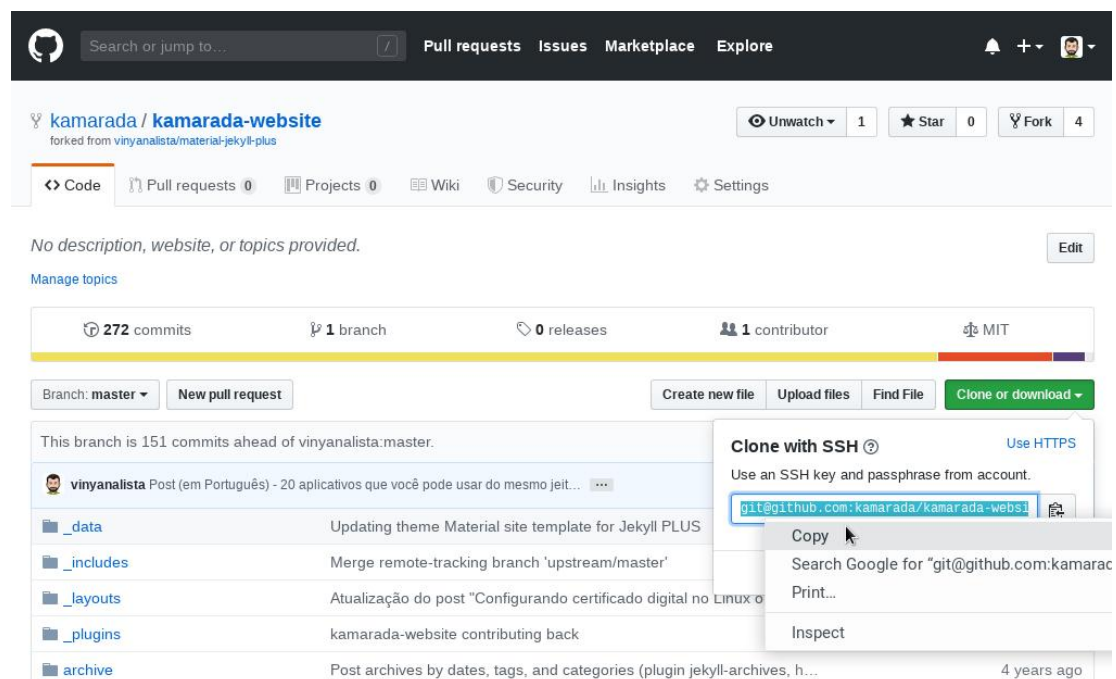
If you completed the test successfully, now you can use SSH with Bitbucket.

## Clone a repository using SSH

Now that we've got our SSH keys set up, let's see how to clone a Git repository using SSH instead of HTTPS.

### GitHub

At [GitHub](#), go to a project's repository, click **Clone or download** and copy the [URL](#) to clone the repository using SSH:



The URL of a GitHub repository looks like:

```
1 git@github.com:your_user_name/your_project_name.git
```

Open the terminal and run the `git clone` command passing the copied URL as argument.

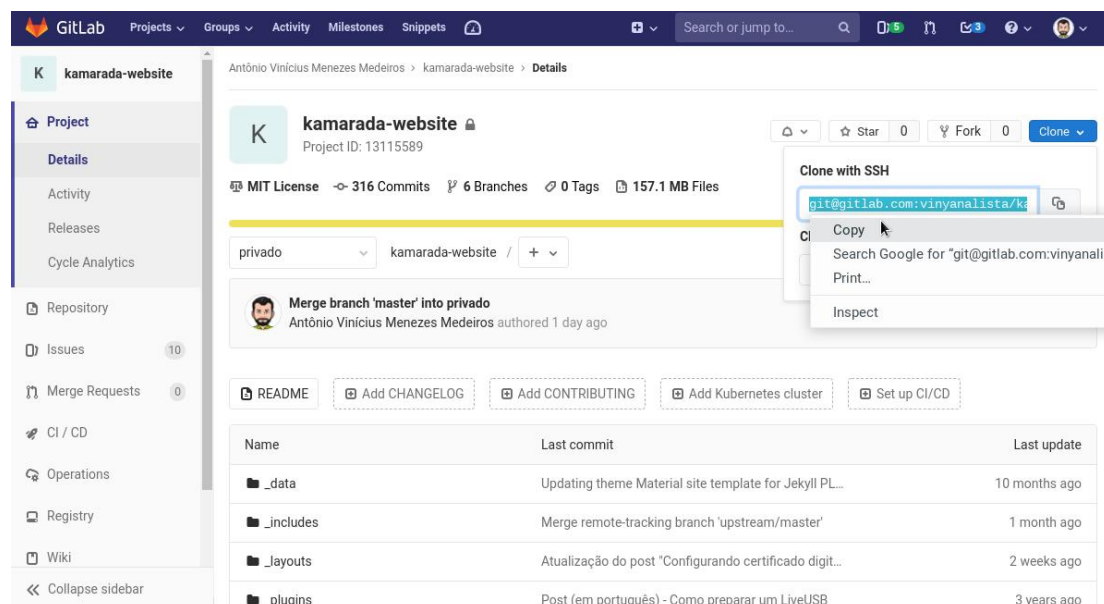
**Tip:** to paste into the terminal, use **Ctrl + Shift + V**.

Note that now Git clones the repository without asking for a password:

```
vinicius@viny-notebook:~  
File Edit View Search Terminal Help  
vinicius@viny-notebook:~> git clone git@github.com:kamarada/kamarada-website.git  
Cloning into 'kamarada-website'...  
remote: Enumerating objects: 351, done.  
remote: Counting objects: 100% (351/351), done.  
remote: Compressing objects: 100% (284/284), done.  
remote: Total 8257 (delta 113), reused 285 (delta 63), pack-reused 7906  
Receiving objects: 100% (8257/8257), 57.79 MiB | 4.19 MiB/s, done.  
Resolving deltas: 100% (4391/4391), done.  
vinicius@viny-notebook:~>
```

## GitLab

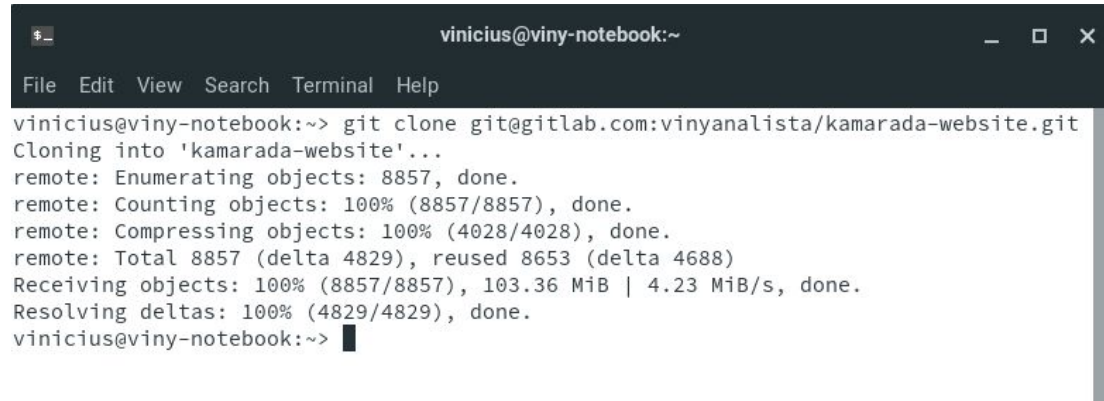
At [GitLab](#), go to a project's repository, click **Clone** and copy the URL to clone the repository using SSH:



The URL of a GitLab repository looks like:

```
1 git@gitlab.com:your_user_name/your_project_name.git
```

Open the terminal and run the `git clone` command passing the copied URL as argument:

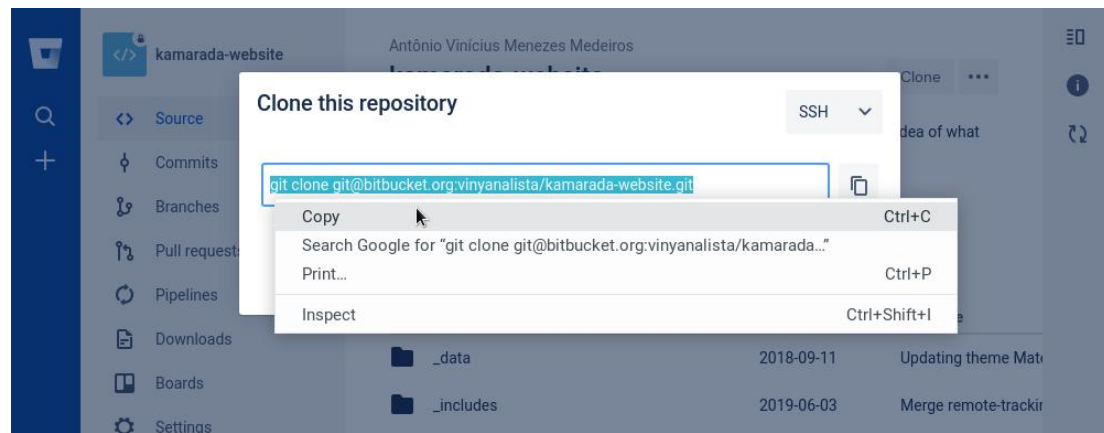
A terminal window titled 'vinicius@viny-notebook:~' showing the execution of the command 'git clone git@gitlab.com:vinyanalista/kamarada-website.git'. The output shows the cloning process: 'Cloning into 'kamarada-website'...', 'remote: Enumerating objects: 8857, done.', 'remote: Counting objects: 100% (8857/8857), done.', 'remote: Compressing objects: 100% (4028/4028), done.', 'remote: Total 8857 (delta 4829), reused 8653 (delta 4688)', 'Receiving objects: 100% (8857/8857), 103.36 MiB | 4.23 MiB/s, done.', 'Resolving deltas: 100% (4829/4829), done.', and the prompt returns to 'vinicius@viny-notebook:~>'.

```
vinicius@viny-notebook:~> git clone git@gitlab.com:vinyanalista/kamarada-website.git
Cloning into 'kamarada-website'...
remote: Enumerating objects: 8857, done.
remote: Counting objects: 100% (8857/8857), done.
remote: Compressing objects: 100% (4028/4028), done.
remote: Total 8857 (delta 4829), reused 8653 (delta 4688)
Receiving objects: 100% (8857/8857), 103.36 MiB | 4.23 MiB/s, done.
Resolving deltas: 100% (4829/4829), done.
vinicius@viny-notebook:~>
```

Note that now Git clones the repository without asking for a password.

## Bitbucket

At [Bitbucket](#), go to a project's repository, click **Clone** and copy the command to clone the repository using SSH:




Note that, differently from GitHub and GitLab that present the URL, Bitbucket presents the entire `git clone` command, including the URL.

The URL of a Bitbucket repository looks like:

1	<code>git@bitbucket.org:your_user_name/your_project_name.git</code>
---	---

Open the terminal, paste and run the command you copied from Bitbucket:

A terminal window titled 'vinicius@viny-notebook:~' showing the execution of the command 'git clone git@bitbucket.org:vinyanalista/kamarada-website.git'. The command is partially visible at the bottom of the terminal.

```
vinicius@viny-notebook:~> git clone git@bitbucket.org:vinyanalista/kamarada-website.git
```



```
vinicius@viny-notebook:~$ git clone git@redacted.org:vinicius/kamarada-website.git
Cloning into 'kamarada-website'...
remote: Counting objects: 8802, done.
remote: Compressing objects: 100% (4379/4379), done.
remote: Total 8802 (delta 4812), reused 7880 (delta 4281)
Receiving objects: 100% (8802/8802), 121.15 MiB | 4.15 MiB/s, done.
Resolving deltas: 100% (4812/4812), done.
vinicius@viny-notebook:~$
```

Note that now Git clones the repository without asking for a password.

## Reconfigure existing repositories to use SSH

The repositories we clone from now on using SSH will continue to use SSH for future Git commands such as `git pull` and `git push`. But existing local repositories, previously cloned with HTTPS, will continue to use HTTPS, unless we set them up to use SSH.

To do that, open the terminal and change the current directory to a local repository.

List the existing remote repositories and their URLs with:

That command should output something like:

```
1 origin https://your_server/your_user_name
2 /your_project_name.git (fetch)
origin https://your_server/your_user_name
/your_project_name.git (push)
```

Change your remote repository's URL with:

```
1 $ git remote set-url origin
git@your_server:your_user_name/your_project_name.git
```

Run `git remote -v` once more to verify that the remote repository's URL has changed:

```
1 origin
2 git@your_server:your_user_name/your_project_name.git
(fetch)
```



```
origin  
git@your_server:your_user_name/your_project_name.git  
(push)
```

Great. That done, Git will use SSH, instead of HTTPS, to synchronize that local repository with its remote equivalent.

## References

I hope those tips can be useful to you as they have been to me since I started using Git. If you have any questions or trouble, don't hesitate to comment! See you!

And always remember: have a lot of fun...

- [Connecting to GitHub with SSH - GitHub Help](#)
- [Set up an SSH key - Atlassian Documentation](#)
- [Changing a remote's URL - GitHub Help](#)
- [GitLab and SSH keys - GitLab](#)