Error: Permission denied (publickey)

A "Permission denied" error means that the server rejected your connection. There could be several reasons why, and the most common examples are explained below.

In this article

Should the `sudo` command be used with Git?

You should not be using the `sudo` command with Git. If you have a *very good reason* you must use `sudo`, then ensure you are using it with every command (it's probably just better to use `su` to get a shell as root at that point). If you generate SSH keys without `sudo` and then try to use a command like `sudo git push`, you won't be using the same keys that you generated.

Check that you are connecting to the correct server

Typing is hard, we all know it. Pay attention to what you type; you won't be able to connect to "githib.com" or "guthub.com". In some cases, a corporate network may cause issues resolving the DNS record as well.

To make sure you are connecting to the right domain, you can enter the following command:

```
$ ssh -vT git@github.com
> OpenSSH_8.1p1, LibreSSL 2.7.3
> debug1: Reading configuration data /Users/you/.ssh/config
> debug1: Reading configuration data /etc/ssh/ssh_config
> debug1: /etc/ssh/ssh_config line 47: Applying options for *
> debug1: Connecting to github.com port 22.
```

The connection should be made on port 22, unless you're overriding settings to use SSH over HTTPS.

Always use the "git" user

All connections, including those for remote URLs, must be made as the "git" user. If you try to connect with your GitHub username, it will fail:

```
$ ssh -T GITHUB_USERNAME@github.com
```

```
$ ssh -T GITHUB-USERNAME@github.com
> Permission denied (publickey).
```

If your connection failed and you're using a remote URL with your GitHub username, you can change the remote URL to use the "git" user.

You should verify your connection by typing:

```
$ ssh -T git@github.com
> Hi username! You've successfully authenticated...
```

Make sure you have a key that is being used

If you have GitHub Desktop installed, you can use it to clone repositories and not deal with SSH keys.

1. **If you are using Git Bash**, turn on ssh-agent:

   ```
   # start the ssh-agent in the background
   $ eval "$(ssh-agent -s)"
   > Agent pid 59566
   ```

   If you are using another terminal prompt, such as Git for Windows, turn on ssh-agent:

   ```
   # start the ssh-agent in the background
   $ eval $(ssh-agent -s)
   > Agent pid 59566
   ```

2. Verify that you have a private key generated and loaded into SSH.

   ```
   $ ssh-add -l -E sha256
   > 2048 SHA256:274ffWxgaxq/tSINAykStUL7XWyRNcRTlcST1Ei7gBQ /Users/USERNAME/.ssh/id_rsa (RS
   ```

The `ssh-add` command *should* print out a long string of numbers and letters. If it does not print anything, you will need to generate a new SSH key and associate it with GitHub.

> **Tip**: On most systems the default private keys (`~/.ssh/id_rsa` and `~/.ssh/identity`) are automatically added to the SSH authentication agent. You shouldn't need to run `ssh-add path/to/key` unless you override the file name when you generate a key.

Getting more details

You can also check that the key is being used by trying to connect to `git@github.com` :

```
$ ssh -vT git@github.com
> ...
> debug1: identity file /Users/you/.ssh/id_rsa type -1
> debug1: identity file /Users/you/.ssh/id_rsa-cert type -1
> debug1: identity file /Users/you/.ssh/id_dsa type -1
> debug1: identity file /Users/you/.ssh/id_dsa-cert type -1
> ...
> debug1: Authentications that can continue: publickey
> debug1: Next authentication method: publickey
```

```
> debug1: Trying private key: /Users/you/.ssh/id_rsa
> debug1: Trying private key: /Users/you/.ssh/id_dsa
> debug1: No more authentication methods to try.
> Permission denied (publickey).
```

In that example, we did not have any keys for SSH to use. The "-1" at the end of the "identity file" lines means SSH couldn't find a file to use. Later on, the "Trying private key" lines also indicate that no file was found. If a file existed, those lines would be "1" and "Offering public key", respectively:

```
$ ssh -vT git@github.com
> ...
> debug1: identity file /Users/you/.ssh/id_rsa type 1
> ...
> debug1: Authentications that can continue: publickey
> debug1: Next authentication method: publickey
> debug1: Offering RSA public key: /Users/you/.ssh/id_rsa
```

Verify the public key is attached to your account

You must provide your public key to GitHub to establish a secure connection.

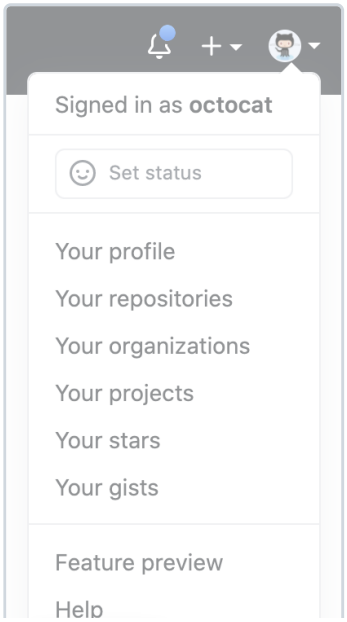
1. Open the command line.
2. Start SSH agent in the background.

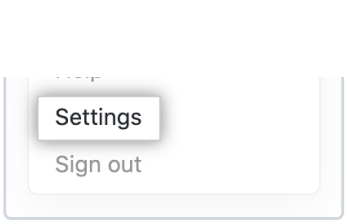
   ```
   $ ssh-agent -s
   > Agent pid 59566
   ```

3. Find and take a note of your public key fingerprint.

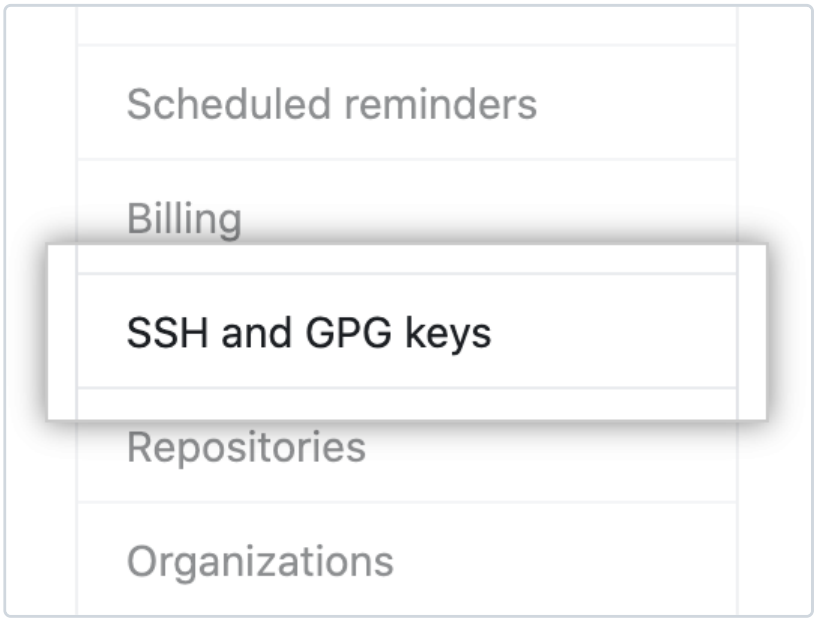
   ```
   $ ssh-add -l -E sha256
   > 2048 SHA256:274ffWxgaxq/tSINAykStUL7XWyRNcRTlcST1Ei7gBQ /Users/USERNAME/.ssh/id_rsa (RS
   ```

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

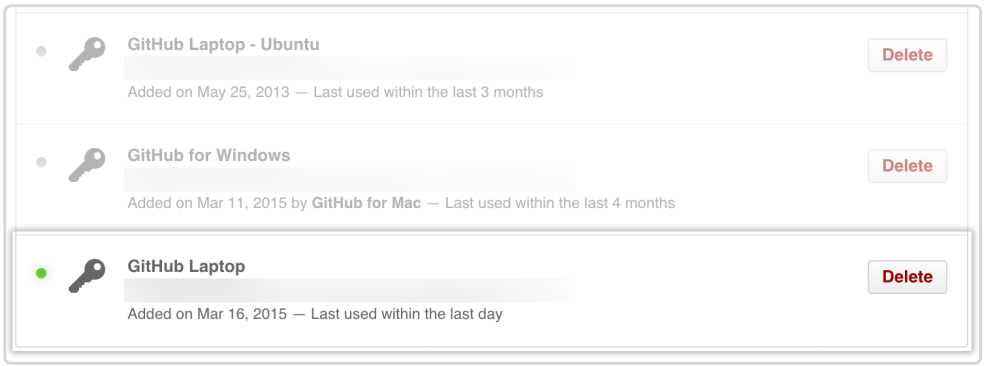




5 In the user settings sidebar, click **SSH and GPG keys**.



6 Compare the list of SSH keys with the output from the `ssh-add` command.



If you don't see your public key in GitHub, you'll need to add your SSH key to GitHub to associate it with your computer.

Warning: If you see an SSH key you're not familiar with on GitHub, delete it immediately and contact GitHub Support, for further help. An unidentified public key may indicate a possible security concern. For more information, see "Reviewing your SSH keys."