

Data 607 Context Presentation

Leticia Salazar

2021-10-26

Contents

<i>GitHub Shortcuts on Terminal and Command Prompt</i>	1
<i>SSH Key:</i>	1
<i>Additional steps to use GitHub remotely through your Terminal / Command Prompt</i>	4
<i>Simple Operations in Terminal / Command Prompt:</i>	5
<i>How to use GitHub and Terminal / Command Prompt:</i>	5
<i>Practice Examples</i>	6
<i>Feature Branches:</i>	7

GitHub Shortcuts on Terminal and Command Prompt

SSH Key:

By generating SSH keys it allows developers to work remotely with services and transfer information without having to open a new window and type out login information.

SSH Keys can be set up in GitHub to facilitate uploading files and updating your repos, especially when working in groups. It will help you push your code through your Terminal for Mac users or your Command Prompt for Windows users.

Steps to set up your SSH Key(s):

1. Sign up for a GitHub account.
2. Open up your Terminal (Command Prompt)
3. Before setting up your SSH key, make sure you do not have a key set up in your system by typing:

```
ls -al ~/.ssh
```

If no key, move to step 4, otherwise check that none of the keys are listed under `id_rsa`, see im-

age below:

```

MacBook-Pro Documents % ls -al ~/.ssh
total 56
drwx----- 7          staff   224 Aug  3 16:07 .
drwxr-xr-x+ 66          staff  2112 Oct 20 21:36 ..
-rw-r--r--  1          staff  12288 Aug  3 2019 .config.swp
-rw-r--r--@ 1          staff    78 Aug  3 16:07 config
-rw----- 1          staff  3401 Oct 19 13:59 id_rsa
-rw-r--r-- 1          staff   760 Oct 19 13:59 id_rsa.pub
-rw-r--r-- 1          staff   3370 Jul 31 20:35 known_hosts

```

If you do find keys with a matching name you can overwrite them by following the steps below. If you do decide to stay with the same keys be aware that you will have to remember the password connected to your key.

4. Type the following along with your email to generate your keys:

```
ssh-keygen -t rsa -b 4096 -C "yourgithubemail@email.com"
```

5. You will be asked to enter a file to save the key, just hit the *Return/Enter* key.

* Enter a password for your key {you can opt for not adding a password but your key will not be encrypted.}

```

MacBook-Pro ~ % ssh-keygen -t rsa -b 4096 -C "
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/ /ssh/id_rsa):
/Users/ /ssh/id_rsa already exists.
Overwrite (y/n)? Y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/ /ssh/id_rsa.
Your public key has been saved in /Users/ /ssh/id_rsa.pub.
The key fingerprint is:
Your Key
The key's randomart image is:
+---[RSA 4096]---+
|+=+0o.          |
|=&.. + .         |
|B==oE =         |
|+=+oB.B =       |
|.o0 B B S .     |
|.+.o + . o      |
|..              |
|                |
|                |
+---[SHA256]-----+

```

6. Your window should look like this:

7. Now we test whether the SSH agent is working on your machine, run the following command in your Terminal/Command Prompt:

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

* Your Terminal/Command Prompt must look like this: ![(/Users/letiix3/Desktop/Data-607/Context_Presentation/Pictures/agent_pid.png)]

8. Next, run the following command:

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

9. Enter the password associated with the SSH key and if you have forgotten your password go back to step 4 to create a new one.

For the next steps please make sure to have logged into your GitHub account.

10. We will add the key to GitHub but first enter the following command to copy the key to your clipboard:

```
pbcopy < ~/.ssh/id_rsa.pub
```

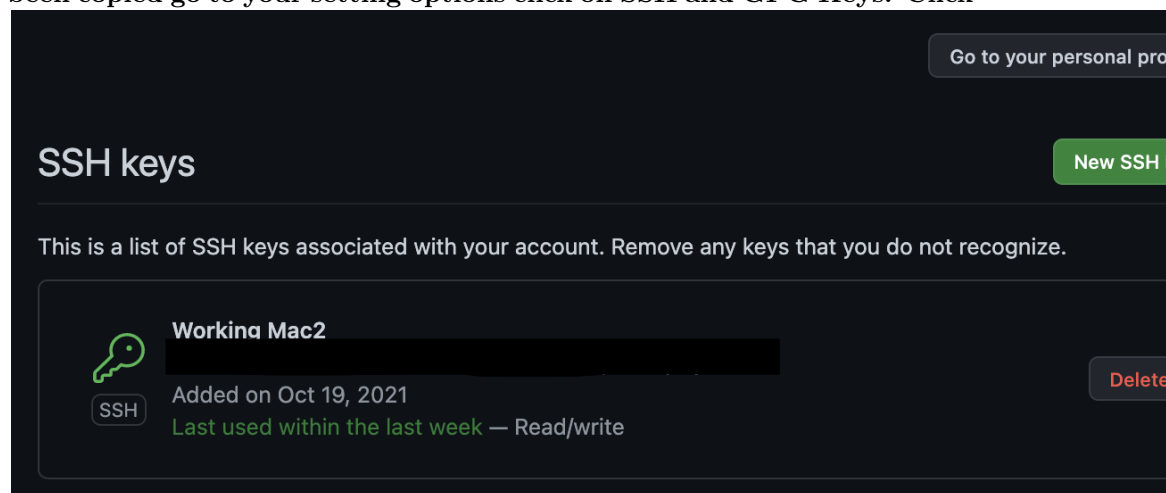
* For Windows users use this command:

```
clip < ~/.ssh/id_rsa.pub
```

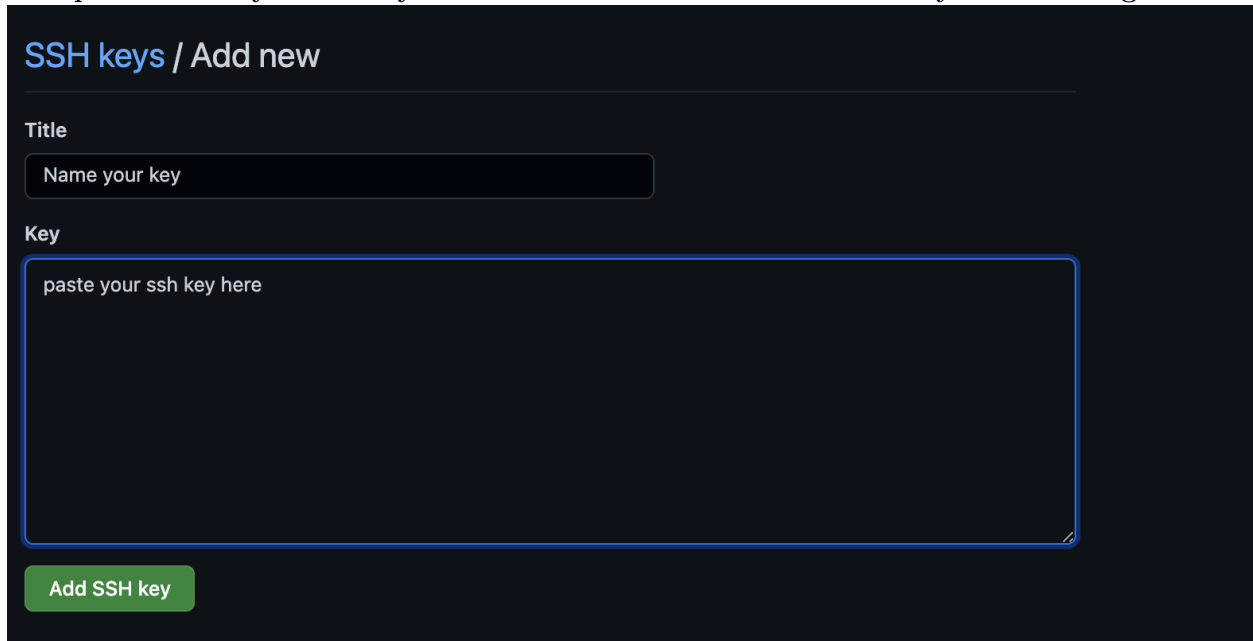
* Note: no message should appear, if there is please re-enter the command once more.

11. Once the key has been copied go to your setting options click on SSH and GPG Keys. Click

on “New SSH Key”.



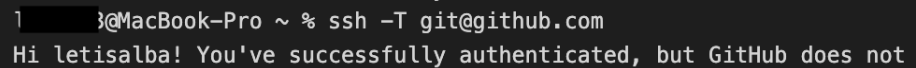
12. You will then see a form where you can enter a name for your working computer and paste the key. Once you are done click on “Add SSH Key”. See image below:



13. Finally we add GitHub to your computer’s list of acceptable SSH hosts by entering the following command on your Terminal/Command Prompt:

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

You should see the following message:



```
1 letisalba@MacBook-Pro ~ % ssh -T git@github.com
Hi letisalba! You've successfully authenticated, but GitHub does not
```

Additional steps to use GitHub remotely through your Terminal / Command Prompt

Git associates a username / email with any commit you make. For that there’s two simple steps you can do using the `git config` command.

Setting your Git username for your repositories on your computer

Open your Terminal / Command Prompt

Set a Git username:

```
git config --global user.name "Jane Doe"
```

Confirm that you have set the Git username correctly:

```
git config --global user.name
```

```
Jane Doe
```

Setting your email address for your repositories on your computer

Open your Terminal / Command Prompt

Set an email address in Git [You can use any email address]:

```
git config --global user.email "youremail@email.com"
```

Confirm that you have set the Git username correctly:

```
git config --global user.email
```

```
youremail@email.com
```

Simple Operations in Terminal / Command Prompt:

- `cd` changes directory
- `cd ~` changes to home directory
- `cd ..` moves up one directory
- `ls` lists files in folder
- `pwd` shows current directory
- `mkdir <FOLDERNAME>` creates new directory
- `touch <FILENAME>` creates a file
- `open .` opens the current folder for Mac
- `explore .` opens the specific file for Windows

For more commands check out this [Git Cheat Sheet](#)

Using TAB can autocomplete the file or directory name. Autocomplete only works for unique files or directory names.

How to use GitHub and Terminal / Command Prompt:

- Open the Terminal / Command Prompt and (using only the command line) create the following:
 - Navigate to your Desktop or Directory: `cd Desktop`
 - Create a new directory names test: `mkdir test`

NOTE: When naming your directories and files, we generally want to avoid having spaces in the name. If you wish to have a descriptive name, it's recommended you use underscores. So, instead of naming a directory "directory name", name it "directory_name".

- Create a new txt file called sample: `touch sample.txt`
- Navigate into the test directory: `cd test`
- Check to see what's inside (it should be empty): `ls`
- Navigate back to Desktop: `cd ..`
- Move the sample.txt file into the test directory: `mv sample.txt test`
- Make sure this was successful: `cd test` press enter and then `ls`

You can also rename your text file name from sample.txt to example.txt by using the `mv` command

- Navigate into the test directory: `cd test`
- Rename the text file: `mv old-file-name new-file-name`
- Make sure this was successful: `ls *.txt`

Practice Examples

- Create a new repository on GitHub and copy the link ex:<https://github.com/letisalba/your-repo.git>
- In your Terminal / Command Prompt, `cd` into the directory where you want your repository to live (i.e Desktop, Documents, etc)
- Clone the repository using `git clone <repo name>`
- `cd` into the repository and add a file
- Add the changes in your current working directory by using `git add .`
- Commit the changes using `git commit -m "some message"`
- Push the changes using `git push origin main`

`git push origin main` will push your changes directly into the main branch which is the main source for your project.

When working in a team, you might not have access to the main branch so pushing directly to the main branch might not be the best approach. By using Feature Branching, it allows to push code into a feature branch rather than the main branch.

How to work in a Feature Branch Workflow:

- Start with the main branch
 - `git checkout main`
 - `git fetch origin`
 - `git reset --hard origin/main`

This switches to the main branch, pull the latest changes and resets the repo's local copy of main to match the latest version

- Create a new branch
 - All changes to our repository (additions, modifications, or deletion) are made inside this branch.
- Commit the changes, and push them to GitHub
- Once the branch has been pushed, we can open a Pull Request.
 - Once the Pull Request has been made, there will very often be automated tests that run to ensure that your modifications do not break the working application.
 - Pull Requests also initiate a discussion about the changes you've committed, and generally have to be approved by someone else on the team. That person (or entire team) will review your changes to make sure that everything is in order.
- Once any concerns have been addressed and the Pull Request has been approved, the modified code is merged into the main branch.
- A great benefit of this process is that, once merged, Pull Requests act as a record of changes to the codebase, just like a save file!

Feature Branches:

To create a new branch:

- `cd` into the repo directory
- Create and switch to a new branch using `git checkout -b new-branch-name`
- Make the changes to your repository that you desire
- Stage the changes using `git add -A`
- Commit the changes using `git commit -m "some message"`
- Push the changes using `git push origin new-branch-name`
- Navigate to your GitHub repository in your browser
- On the Pull requests tab, create a new pull request
- To switch between existing branches, use `git checkout branch-name`. *Note that you do not need the -b flag in your git checkout command this time because we are not creating a new branch.*

To merge your changes into main, and update your local repository with those changes:

- On GitHub, merge the Pull Request you created.
- On your computer, switch back to your main branch using `git checkout main`
- Pull the updates from the remote server with `git pull`