How to Properly Setup Your Github Repository — Windows Version

Alex Aklson Follow
Jan 30 · 11 min read ★

As a data scientist, I interact frequently with people who are aspiring data scientists, and usually one of my recommendations to them is to start working on projects and publishing their work and code on a public platform in order to be able to showcase their work to prospective employers. One such public platform is Github.

If you do not yet know what Github is, suffice to know that it is a development platform that provides teams, developers, and even individuals with the ability to host, review, and maintain their code and control it with version control. Also, one can fork a repository, meaning copy the source code, and then customize it for their own applications without impacting the original source code. In summary, it is an awesome tool and a must-have skill for any aspiring data scientist.

However, I often get approached by aspiring data scientists having trouble with creating a Github repository and setting it up properly so that they are able to continuously save their work to the repository, or to sync it with the local version of the repository, without any issues. Therefore, I decided to write this post to walk those having difficulties with Github through the process of creating a Github repository and setting it up properly so that they are able to seamlessly sync it with its local version.

Prerequisites:

Please make sure that you have a Github account. If you do not, head over to github.com and click on the **Sign up** button on the top right corner and follow the instructions to create your account.

Please note that the steps and instructions described in this post are relevant to a Windows computer, so if you are using a Mac computer, please check the Mac version of this post.

Also, this post assumes that you already have Git installed. If that is not the case, then please take a moment and install **Git Bash** for Windows. You can select the default options throughout the installation wizard.

Step 1: Authenticate Yourself and Your Machine

Now that you have your Github account created, the first step is to connect your machine to your Github account through an **SSH** key. With an **SSH** key, Github is able to authenticate you automatically every time you try to manipulate your repository without the need to provide your username or password, which is super convenient.

To create an **SSH** key, simply follow the steps outlined below:

1. Launch Git Bash.

2. Type the following, replacing <your email address> with the email address that is linked to your Github account, and hit Enter:

```
ssh-keygen -t rsa -b 4096 -C "<your email address>"
```

This will generate a new **SSH** key.

3. Next, you will be prompted to enter a directory to save the key. I simply press Enter to accept the default location, which is a **.ssh** folder in the home directory. In ther words, you will be able to locate the key in “~/.ssh/id_rsa”.

4. You will then be prompted to choose a passphrase. I prefer not to have a passphrase; so just press Enter and Enter again to confirm the empty passphrase.

Now, if you navigate to the **.ssh** directory, i.e., if you run the following in the Git Bash terminal,

```
cd ~/.ssh
```

and then,

```
ls
```

to the list the contents of the **.ssh** directory, you should find “id_rsa” and “id_rsa.pub” in the list of contents, where “id_rsa” is the private version of your key and “id_rsa.pub” is the public version of your key.

5. Finally, you will need to add the **SSH** key to the ssh-agent, which is meant to help with the authentication process. To do that, first you need to start the ssh-agent, so run the following in the Git Bash terminal:

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

And then add the key to the agent by running the following in the Git Bash terminal:

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

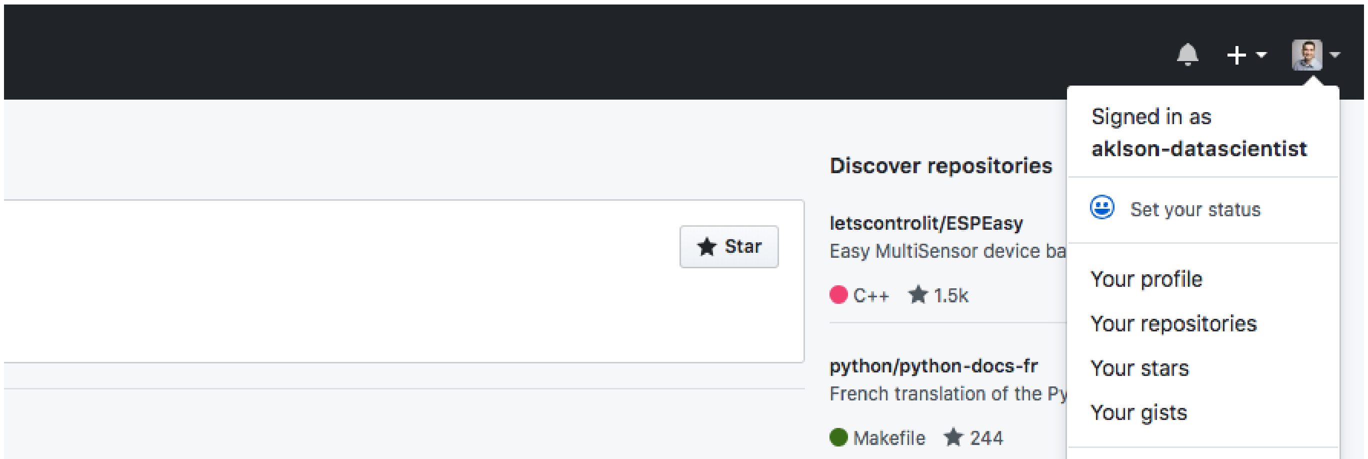
Step 2: Add the Key to Your Github Account

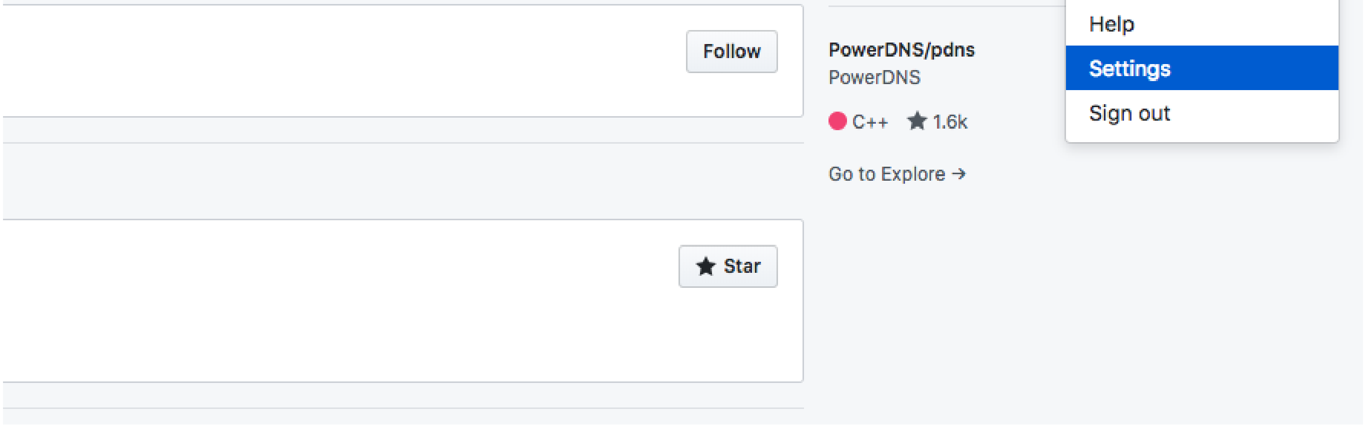
The next step is to add the **SSH** key to your Github account. The first thing to do, is to run the following in the Git Bash terminal:

```
cat ~/.ssh/id_rsa.pub | clip
```

What this does is it copies the public version of your key to the clipboard buffer.

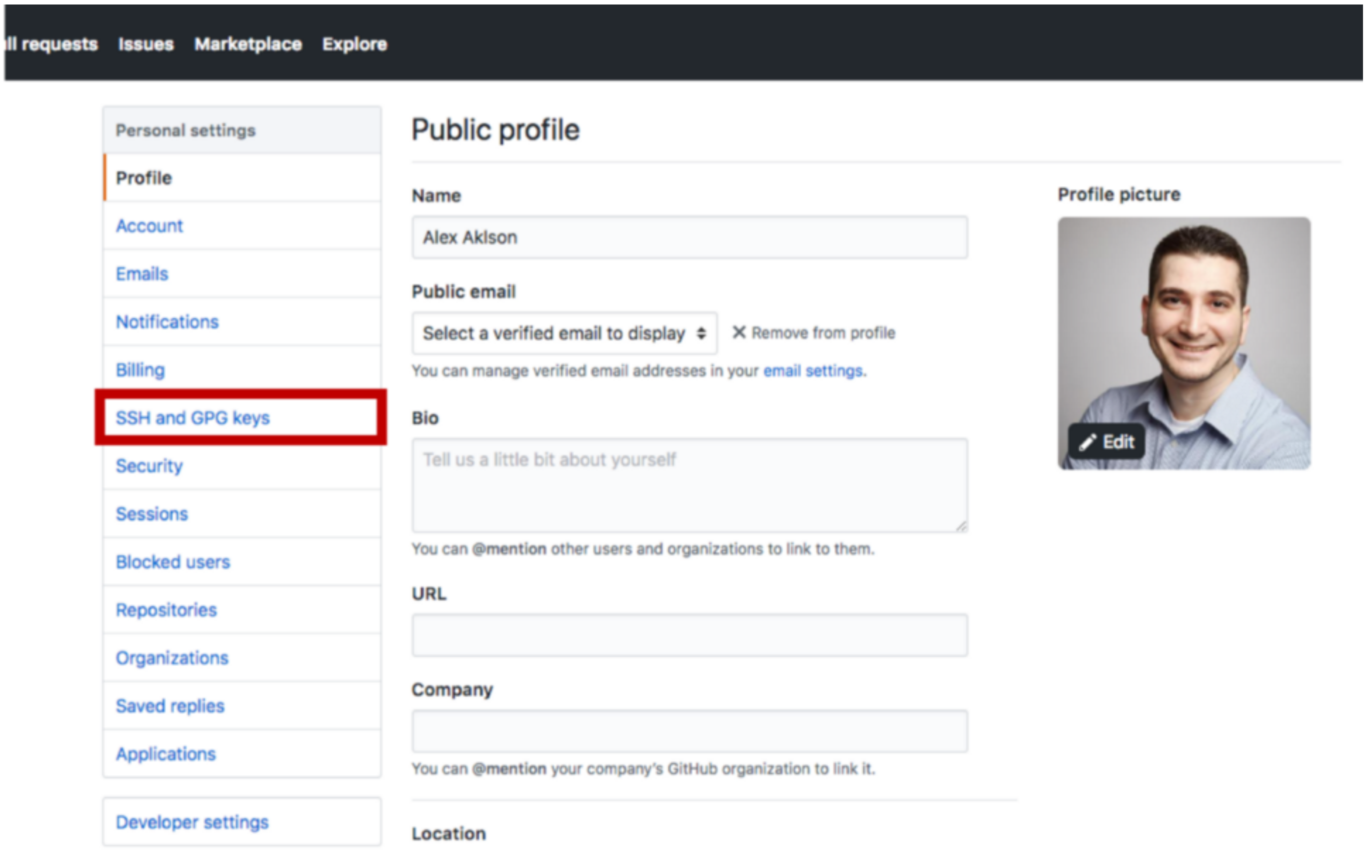
Then, go to your Github account, and click your profile pic in the top right corner of your Github account and select “Settings” from the drop-down menu, as shown below:





Browsing to the Settings page.

Under **Personal settings**, select "SSH and GPG keys", as shown below:



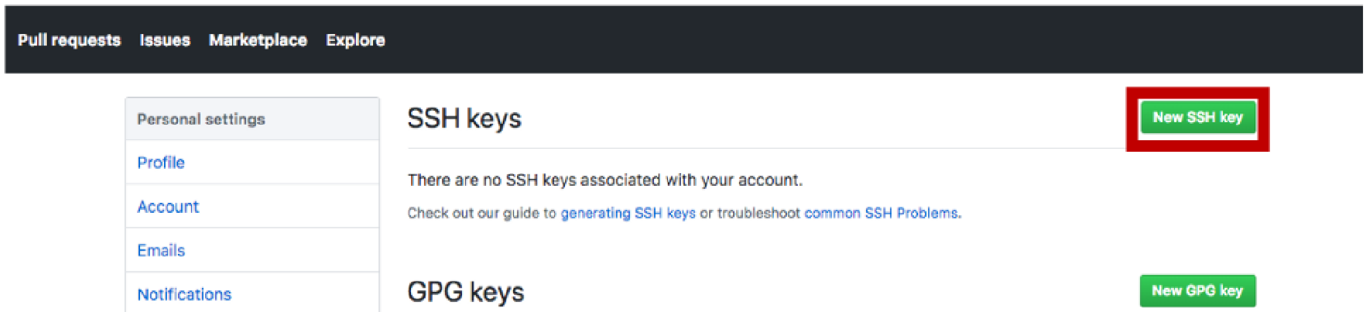
You can manage verified email addresses in your email settings.

You can @mention other users and organizations to link to them.

You can @mention your company's GitHub organization to link it.

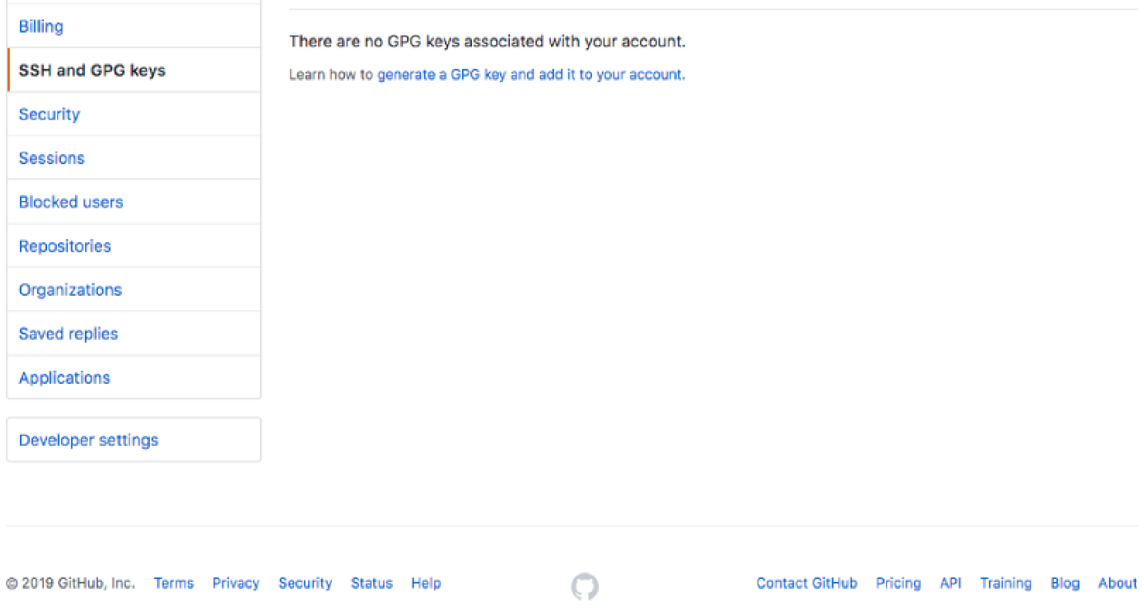
Selecting SSH and GPG keys section.

On the next page, click the button to add a new **SSH** key to your account, as shown below:



There are no SSH keys associated with your account.

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.


Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings

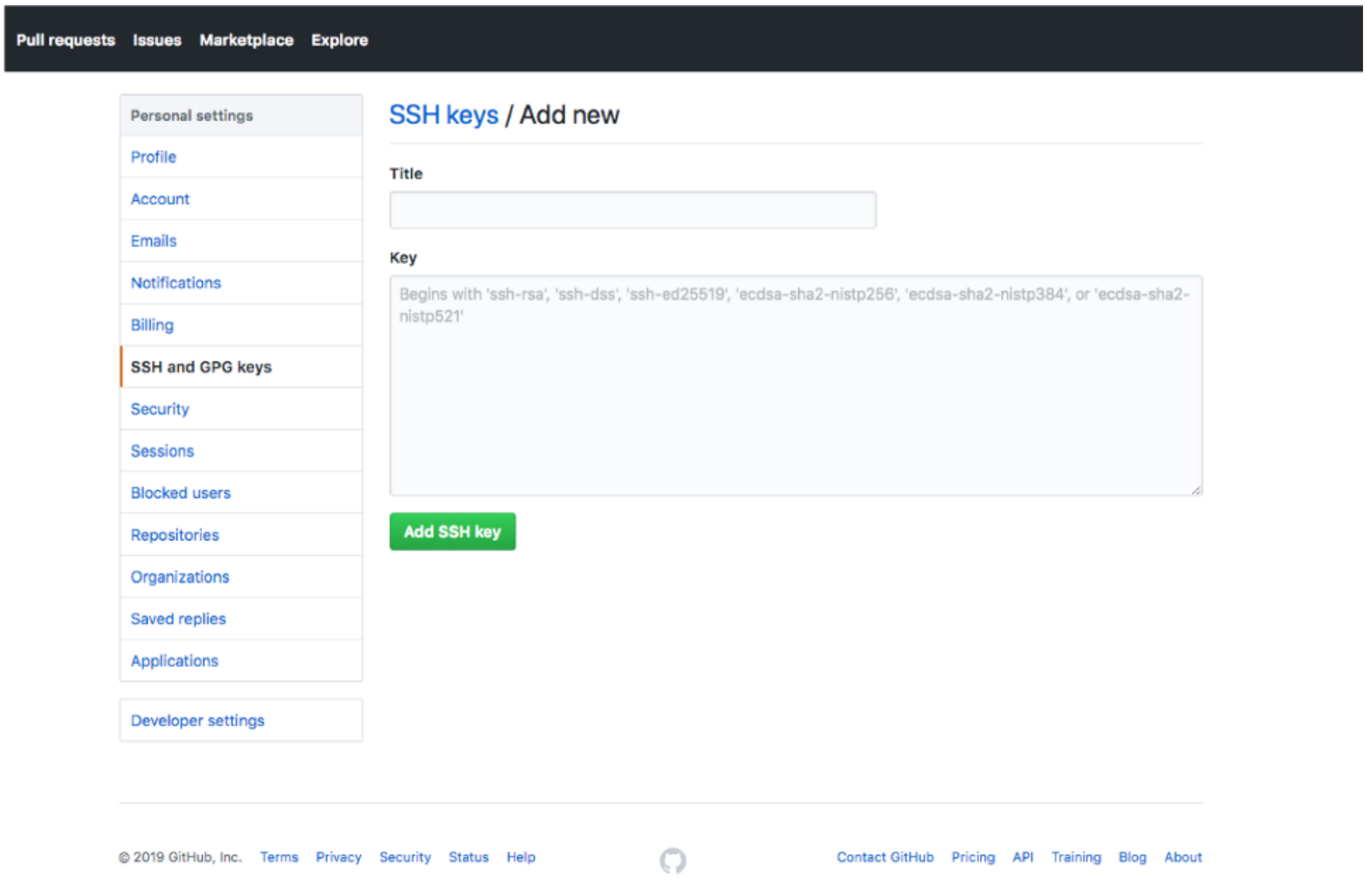
There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Selecting to add a new SSH key.

This will start a new form where you will have to add the key and give it a name as well.



SSH keys / Add new

Title

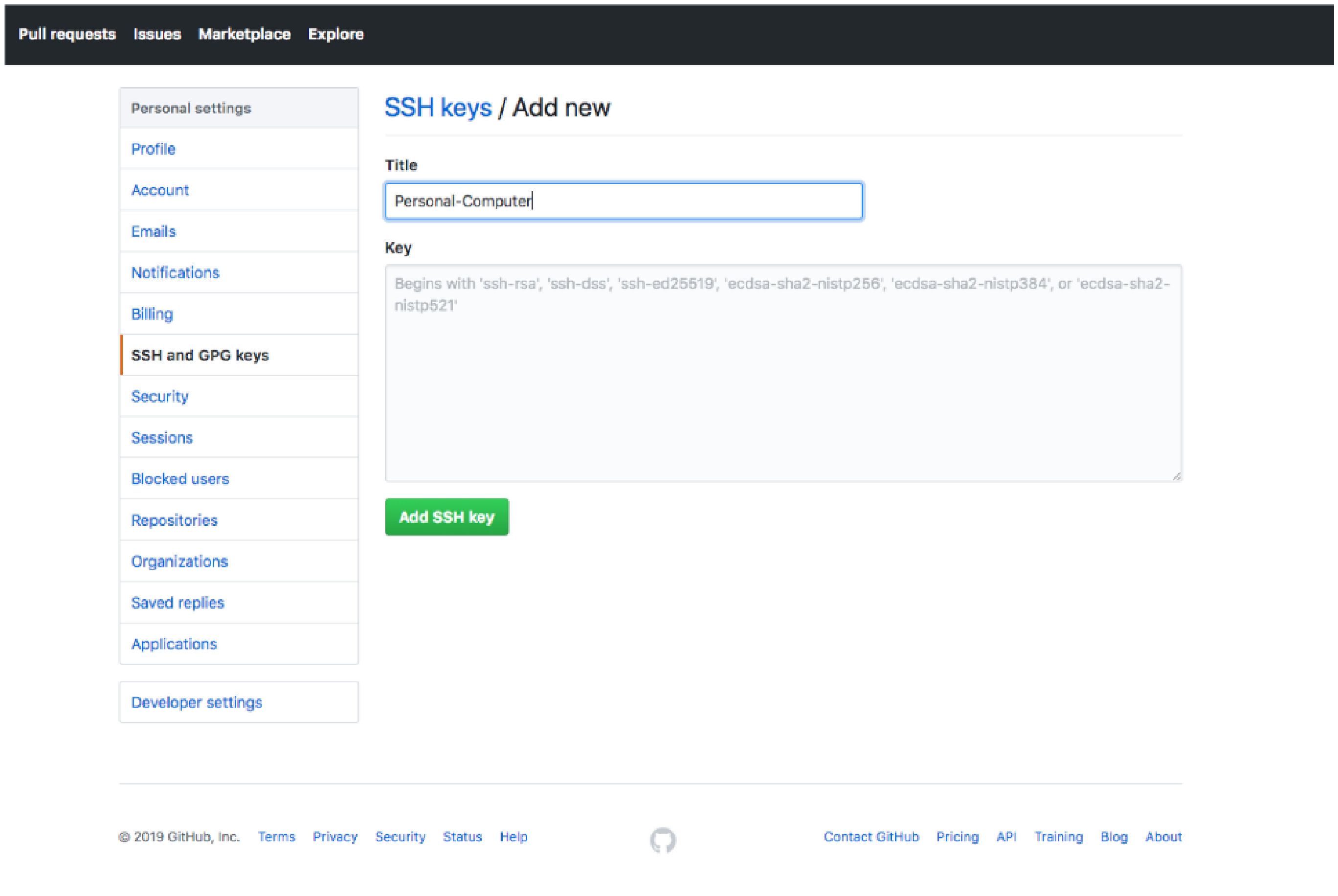
Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Adding a new SSH key page.

For the title, you give the key a name that describes the machine associated with it. So, if it is your personal computer, then you can name it “Personal-Computer”, as shown below:



Title

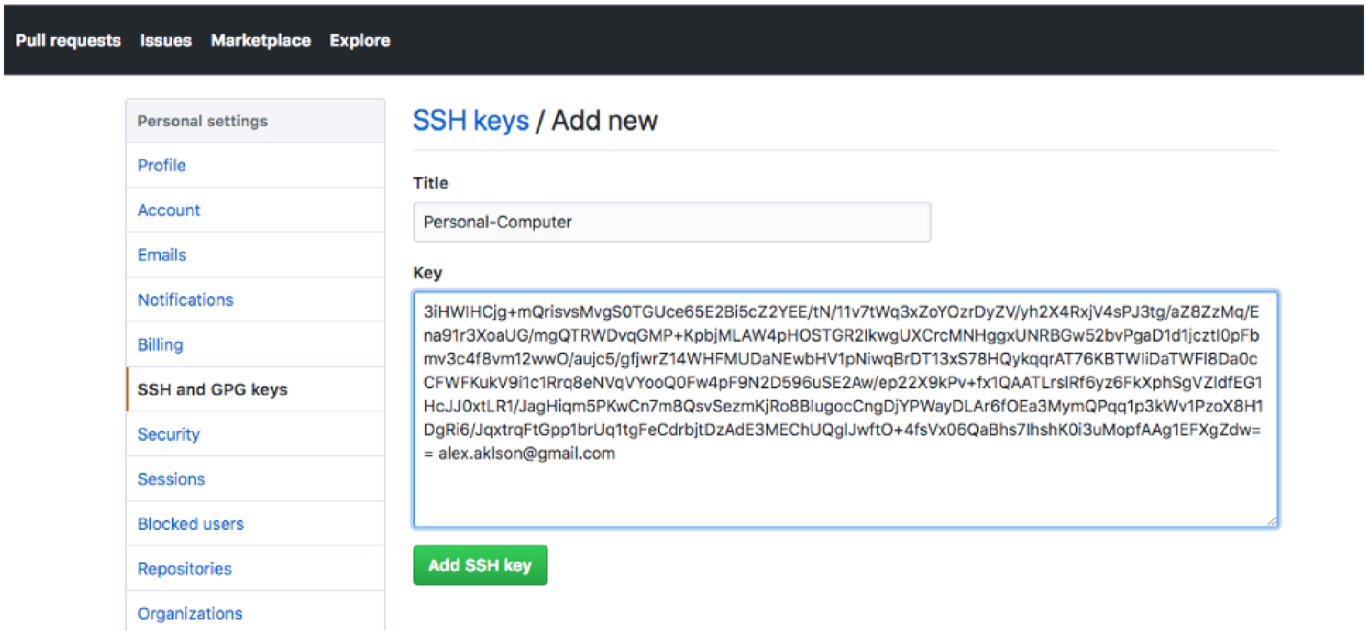
Personal-Computer

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Naming the SSH key.

Then, select the **Key** field, and press **Ctrl-v** to paste the key from the clipboard buffer. The pasted key should have your email address at the end, as shown below:

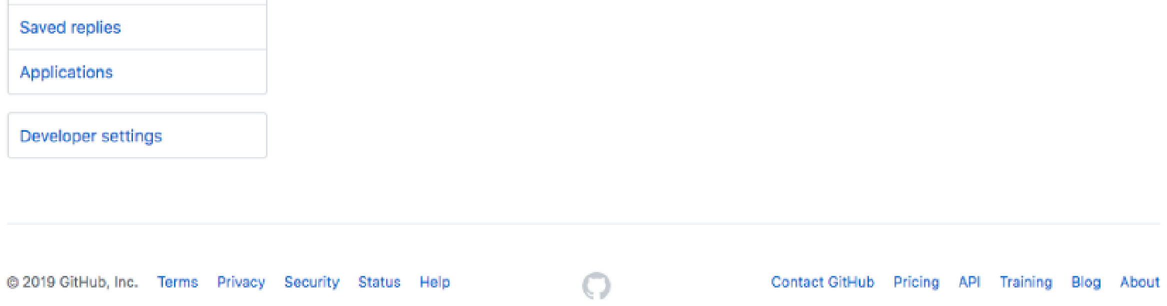


Title

Personal-Computer

Key

```
3iHWIHCjg+mQrisvsMvgS0TGUce65E2Bi5cZ2YEE/tN/11v7tWq3xZoYOzrDyZV/yh2X4RxjV4sPJ3tg/aZ8ZzMq/E
na91r3XoaUG/mgQTRWDvqGMP+KpbjMLAW4pHOSTGR2lkwgUXCrcMNHggxUNRBGw52bvPgaD1d1jcztl0pFb
mv3c4f8vm12wwO/aujc5/gfjwrZ14WHFMUDaNEwbHV1pNiwqBrDT13xS78HQykqqrAT76KBTWliDaTWFl8Da0c
CFWFKukV9i1c1Rrq8eNVqVYooQ0Fw4pF9N2D596uSE2Aw/ep22X9kPv+fx1QAATLrslRf6yz6FkXphSgVZldfEG1
HcJJ0xtLR1/JagHiqm5PKwCn7m8QsvSezmKjRo8BlugocCngDjYPWayDLAr6fOEa3MymQPqq1p3kWv1PzoX8H1
DgRi6/JqxtrqFtGpp1brUq1tgFeCdrbjtDzAdE3MEChUQglJwftO+4fsVx06QaBhs7lhshK0i3uMopfAAg1EFXgZdw=
= alex.aklson@gmail.com
```


Saved replies
Applications
Developer settings
© 2019 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub Pricing API Training Blog About

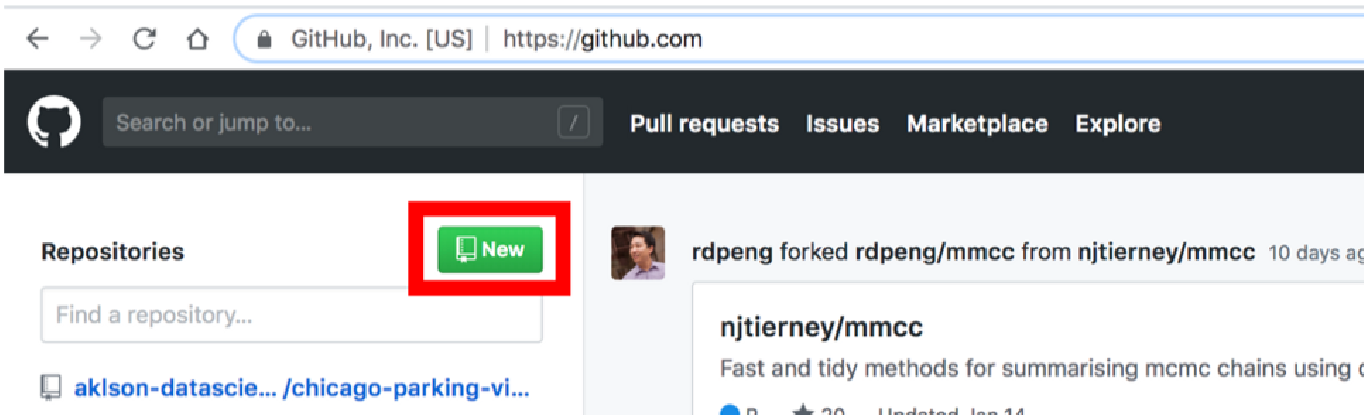
Pasting the new SSH key into the key field.

Finally, click the “Add SSH key” to complete the process of adding the **SSH** key to your Github account.

Step 3: Create a Github Repository

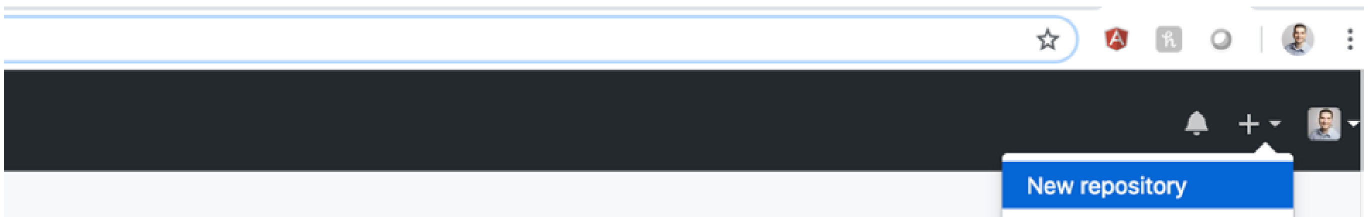
Now that your machine is linked to your Github account, and Github is confident that any manipulation of any of your repositories coming from your machine would be initiated by you, let’s go ahead and create a Github repository.

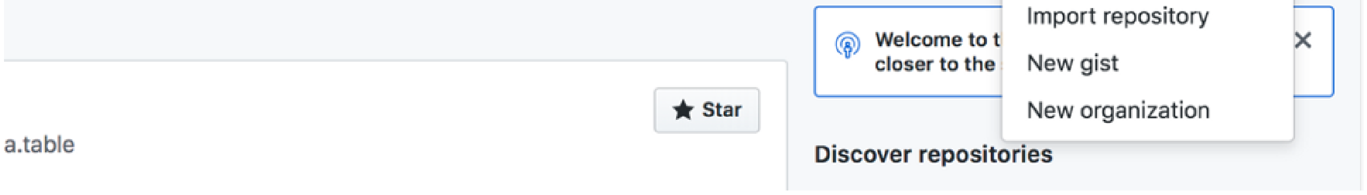
On the homepage of your Github account, you will see all your repositories listed on the left side of the page under **Repositories**, if you have any repositories. You can simply create a new repository by clicking on the green “New” button, as shown below:


GitHub, Inc. [US] | https://github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Repositories
New
Find a repository...
aklson-datascie... /chicago-parking-vi...
rdpeng forked rdpeng/mmcc from njtierney/mmcc 10 days ag
njtierney/mmcc
Fast and tidy methods for summarising mcmc chains using

Creating a new Github repository — option 1.

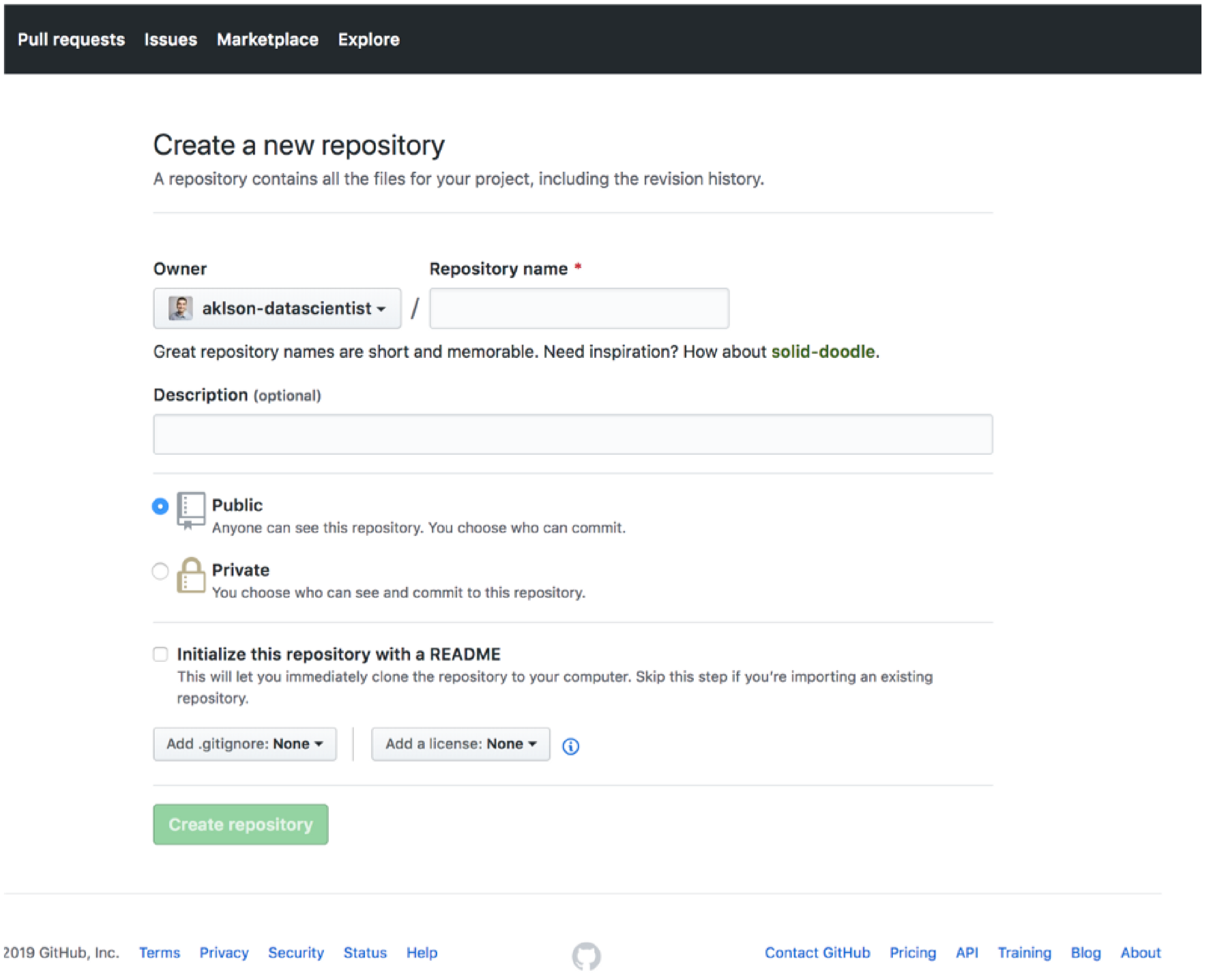
Another option to create a new repository is by clicking the “+” sign next to your profile pic and selecting “New repository” from the drop-down menu, as shown below:





Creating a new Github repository — option 2.

Either option will take you to the following page, where you will have to fill out a form to create the new repository:



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: aklson-datascientist / Repository name *

Great repository names are short and memorable. Need inspiration? How about solid-doodle.

Description (optional)

Public — Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

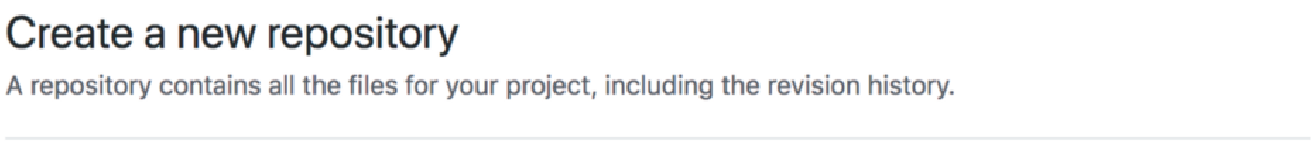
Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

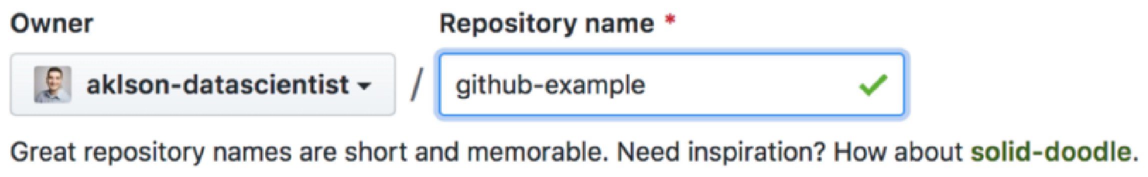
Creating a new repository page.

The first field you will need to fill out is the repository name. Here, I am naming the repository “github-example”, as shown below.



Create a new repository

A repository contains all the files for your project, including the revision history.

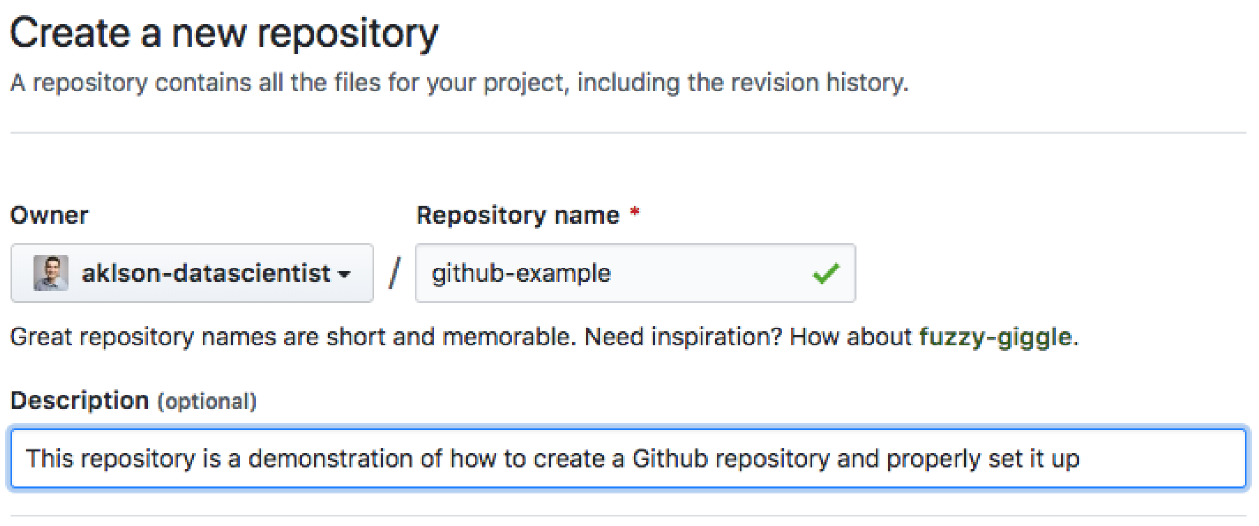

Owner

aklson-datascientist / Repository name * github-example

Great repository names are short and memorable. Need inspiration? How about **solid-doodle**.

Naming the new repository.

The next field is a description of the repository. Although the field is optional, it is highly recommended in order to give other users an idea of what the repository is about, and it reflects a professional image of you.


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

aklson-datascientist / Repository name * github-example

Great repository names are short and memorable. Need inspiration? How about **fuzzy-giggle**.

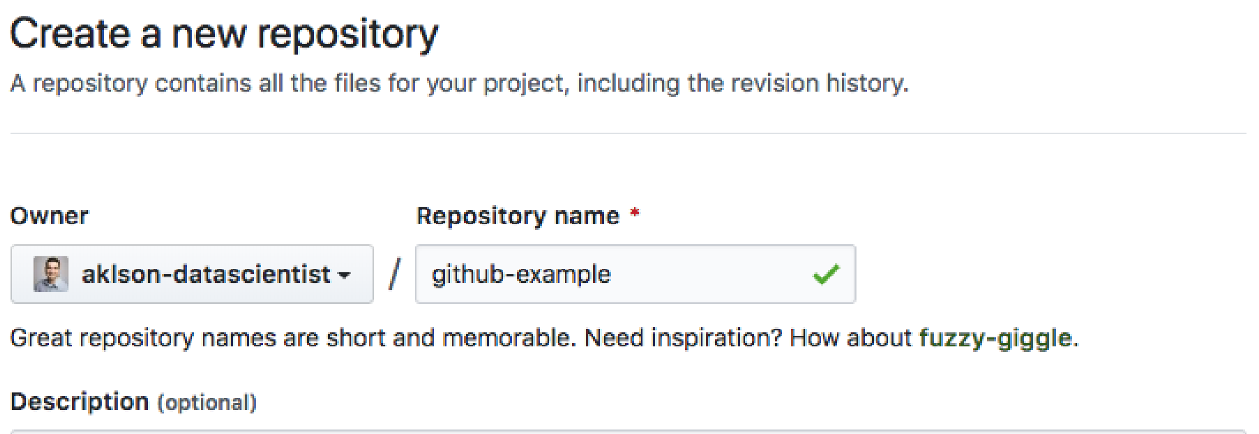
Description (optional)

This repository is a demonstration of how to create a Github repository and properly set it up

Adding Description of the new repository.

Next, you can choose whether you want the repository to be public or private. Private repositories are not free, but let's say you you are working with a client, and you sign an NDA where you cannot disclose that you are working with them. In this case, you should select the private option if you wish to use Github to share your work with them.

On the other hand, if your goal is to showcase your work to potential employers or to share your work with other data scientists, then you want your repositories to be public.


Create a new repository

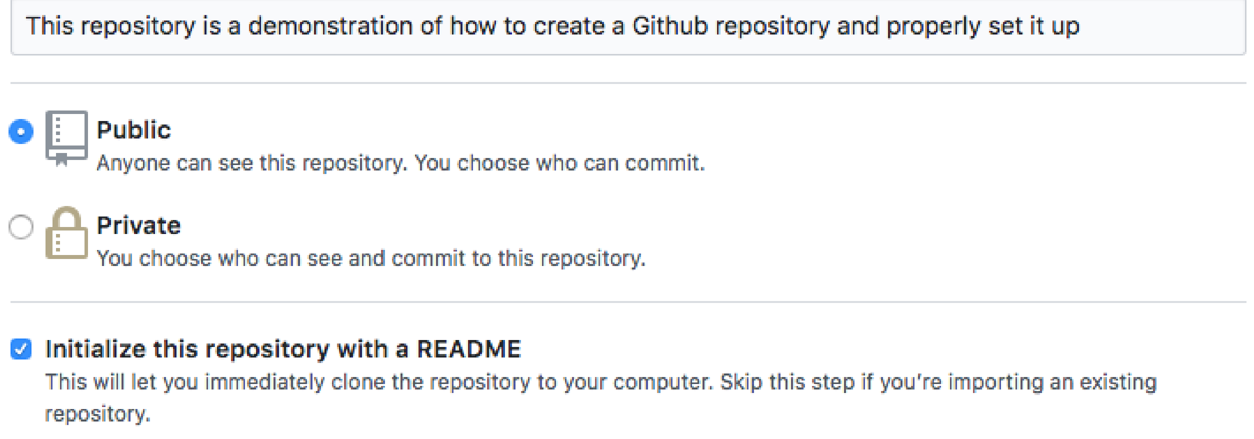
A repository contains all the files for your project, including the revision history.

Owner

aklson-datascientist / Repository name * github-example

Great repository names are short and memorable. Need inspiration? How about **fuzzy-giggle**.

Description (optional)



This repository is a demonstration of how to create a Github repository and properly set it up

● Public
Anyone can see this repository. You choose who can commit.

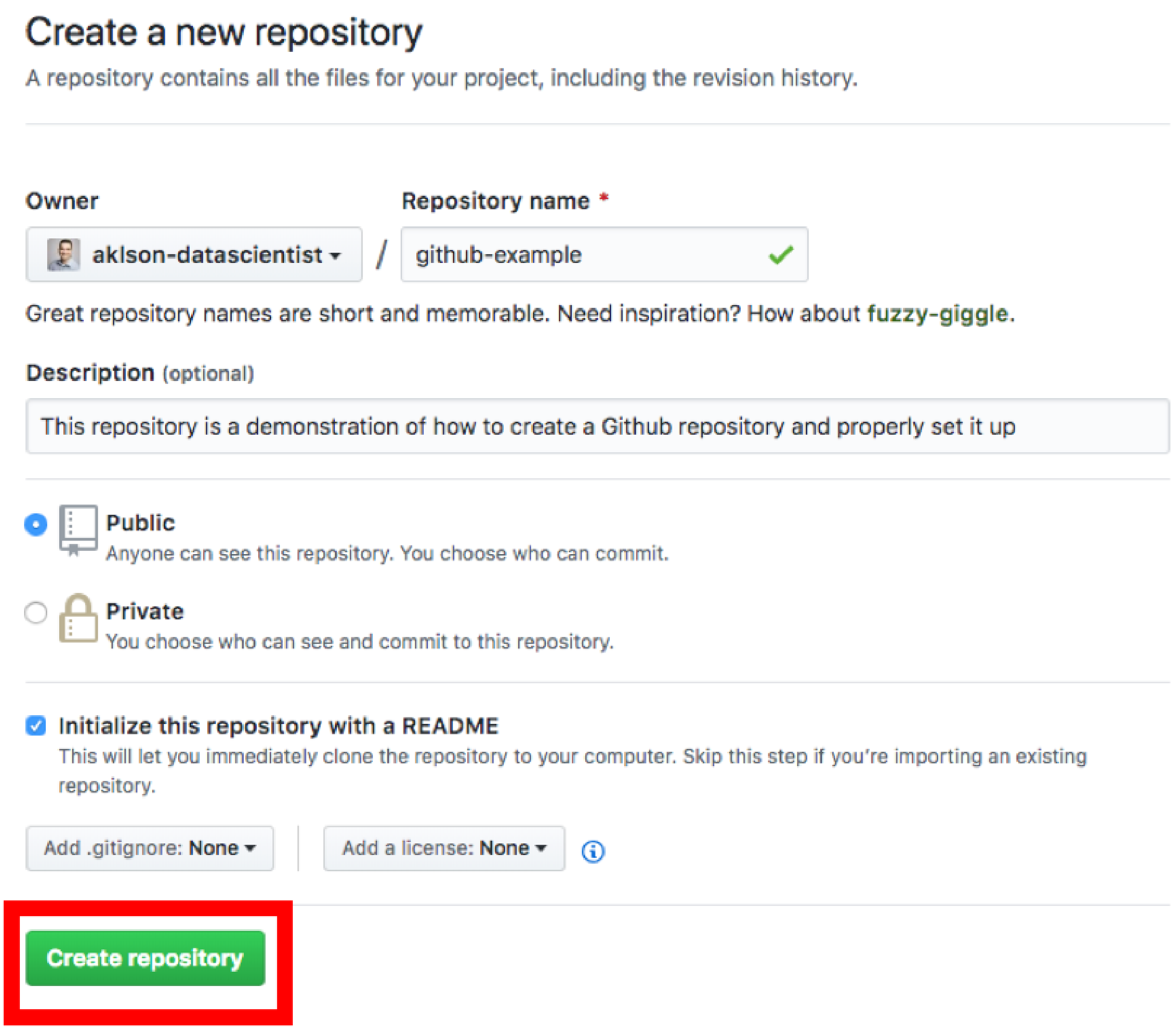
○ Private
You choose who can see and commit to this repository.

✓ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Selecting whether to make the repository private or keep it public and initializing it with a README file.

And make sure to check the option of initializing your repository with a README file, as shown above. The description that you added earlier would automatically become part of your README file, as you will see later.

Finally, go ahead and click the button to create the repository, as shown below.



Create a new repository
A repository contains all the files for your project, including the revision history.

Owner: aklson-datascientist / Repository name *: github-example ✓

Great repository names are short and memorable. Need inspiration? How about fuzzy-giggle.

Description (optional)
This repository is a demonstration of how to create a Github repository and properly set it up

● Public
Anyone can see this repository. You choose who can commit.

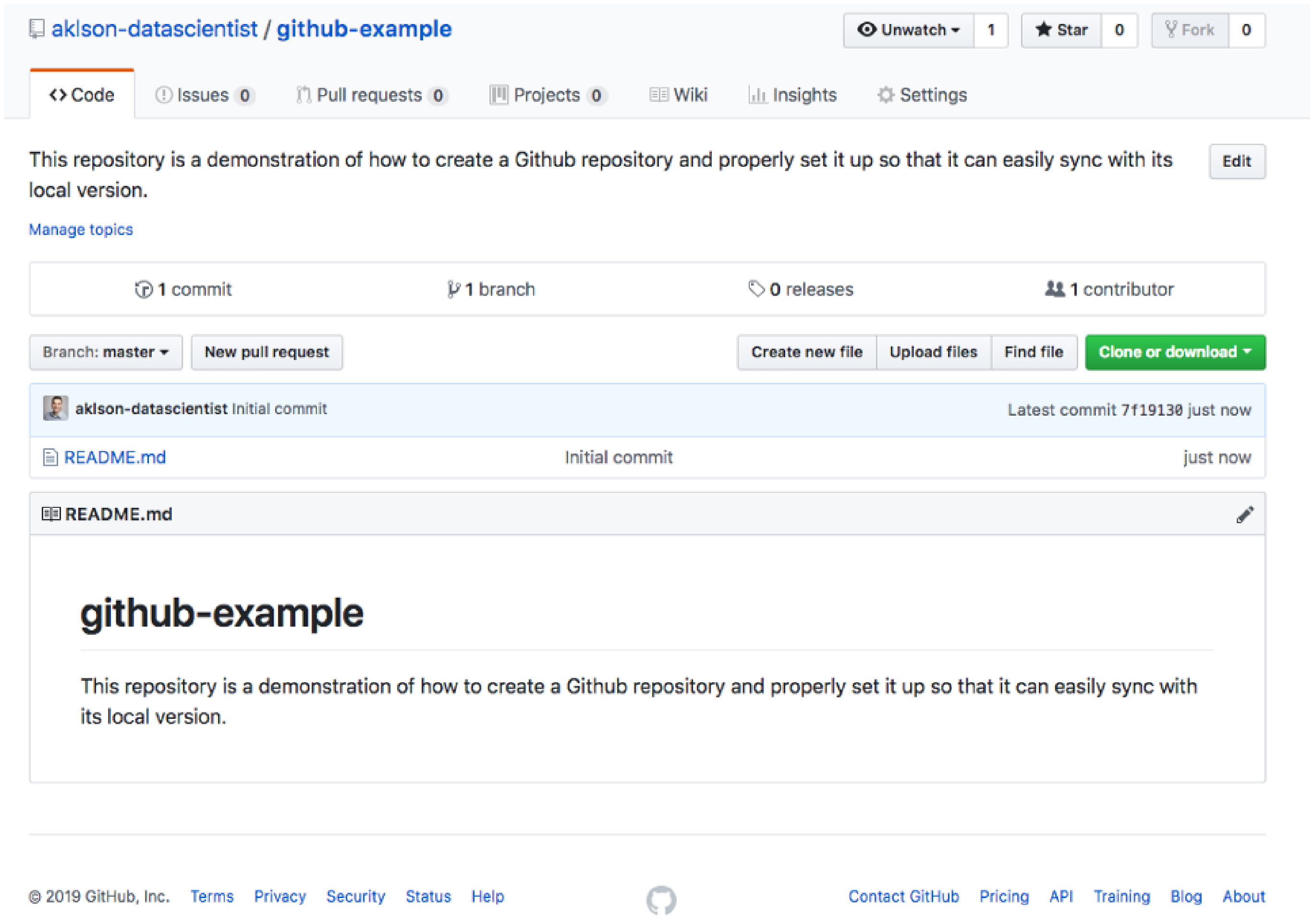
○ Private
You choose who can see and commit to this repository.

✓ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Completing the process of creating a new repository.

Here is how your repository would look like.


aklson-datascientist / github-example

This repository is a demonstration of how to create a Github repository and properly set it up so that it can easily sync with its local version.

aklson-datascientist Initial commit — Latest commit 7f19130 just now

README.md — Initial commit — just now

README.md

github-example

This repository is a demonstration of how to create a Github repository and properly set it up so that it can easily sync with its local version.

© 2019 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub Pricing API Training Blog About

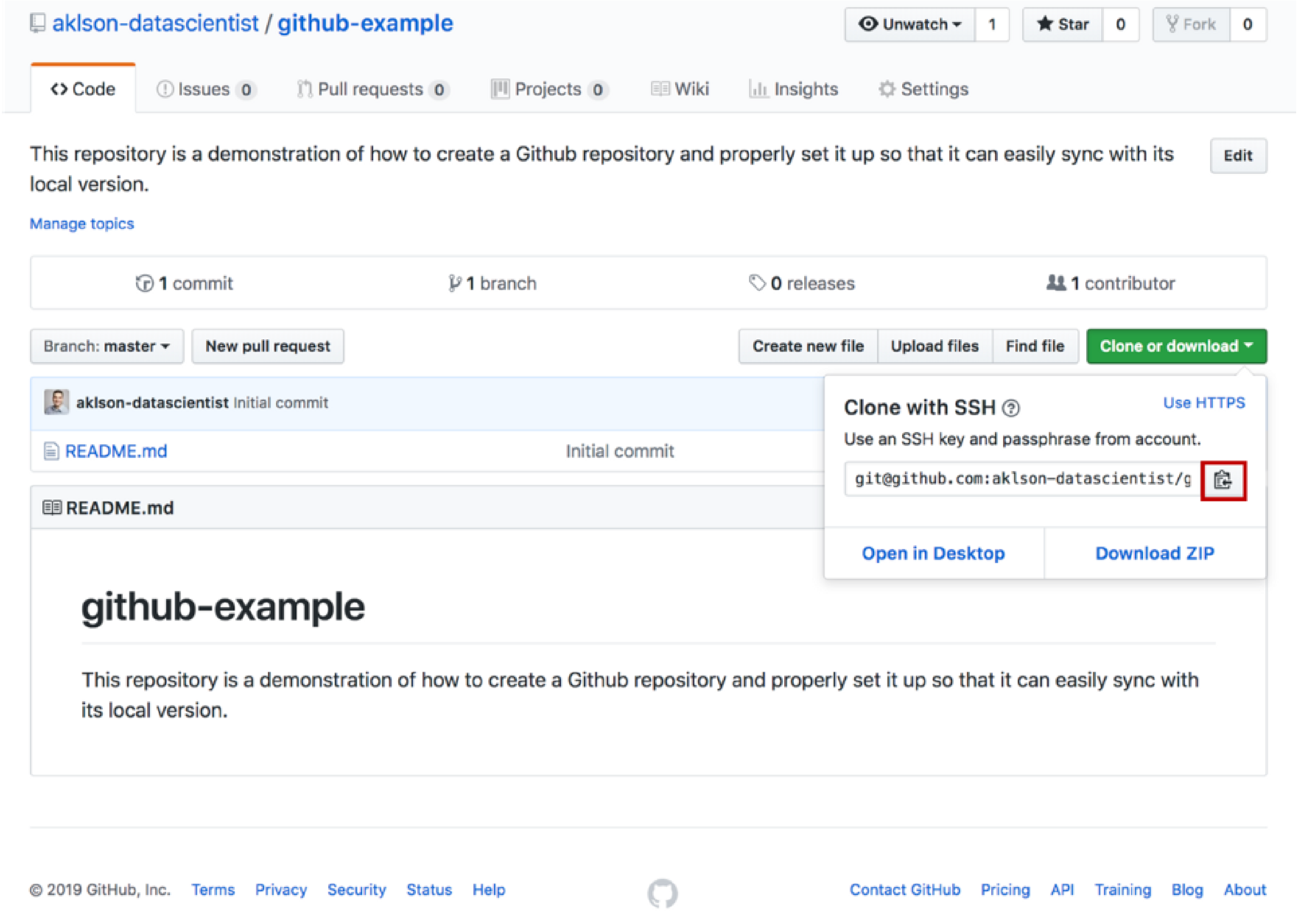
The resulting Github repository.

The name and the description of the repository appear at the top. Next, you would see a list of the files and folders that are in the repository. Finally, you would see what is in the README file.

Since the README file is the first thing users see when they land on your repository, it is good practice to populate it with useful information to users. Using the pencil icon on the right, you can edit the README file and include, for example, a detailed description of the repository. You can also add special instructions or prerequisites to use the repository, and perhaps some examples of how to use the repository.

Step 4: Clone Your Repository

Now that you have your repository created, let's create a local version of it. This is called cloning. To do that, click the green "Clone or download" button and click the copy icon to copy the repository URL to the clipboard buffer. Make sure that you are cloning with SSH and not with HTTPS.


aklson-datascientist / github-example

This repository is a demonstration of how to create a Github repository and properly set it up so that it can easily sync with its local version.

Clone with SSH

Use an SSH key and passphrase from account.

```
git@github.com:aklson-datascientist/g
```

Then, switch back to Git Bash, and navigate to the directory where you want the local version of the repository to reside. For example, you can create a directory called “projects”, that houses all the projects that you plan to work on. To do that, run the following in the Git Bash terminal:

```
mkdir ~/Desktop/projects
```

This will create a folder called “projects” on your Desktop. Then navigate to this directory by running the following in the Git Bash terminal:

```
cd ~/Desktop/projects
```

Now that you are inside the “projects” directory, run the following to clone your repository, replacing <*repository_URL*> with the repository URL that you copied earlier. You would simply press **Ctrl-v** to paste the URL:

```
git clone <repository_URL>
```

For the **github-example** repository, I would execute:

```
git clone git@github.com:aklson-datascientist/github-example.git
```

Then, if you run the following in the Git Bash terminal:

```
ls
```

you should see a new folder with the name of your repository appear in the list of contents of the “projects” directory. So, in my case, I should see **github-example** appear as a folder in my “projects” directory.

Step 5: Push Your First Commit!

Now that you have your repository cloned, let’s go ahead and add a python script to our project. First let’s navigate to the cloned repository. In my case, I would run the following in the Git Bash terminal:

```
cd ~/Desktop/projects/github-example
```

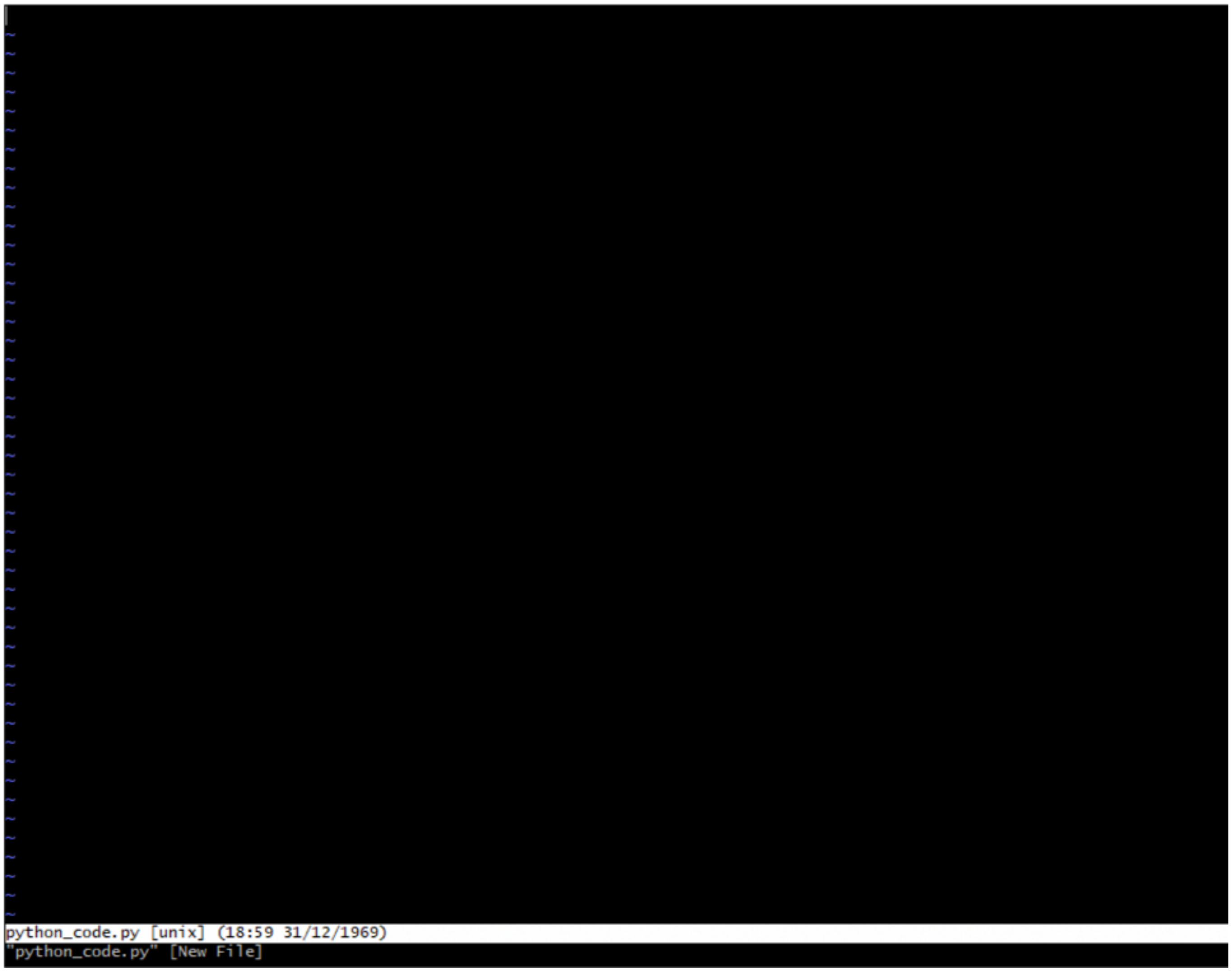
Now that you are inside the cloned repository, let’s create a Python script with a simple print statement:

```
print("Hello Github!")
```

And let’s name the file, *python_code.py*. If you selected Vim as the default code editor when installing Git, then you can create this simple Python file using Vim. To do that, you can type the following in the Git Bash terminal:

```
vim python_code.py
```

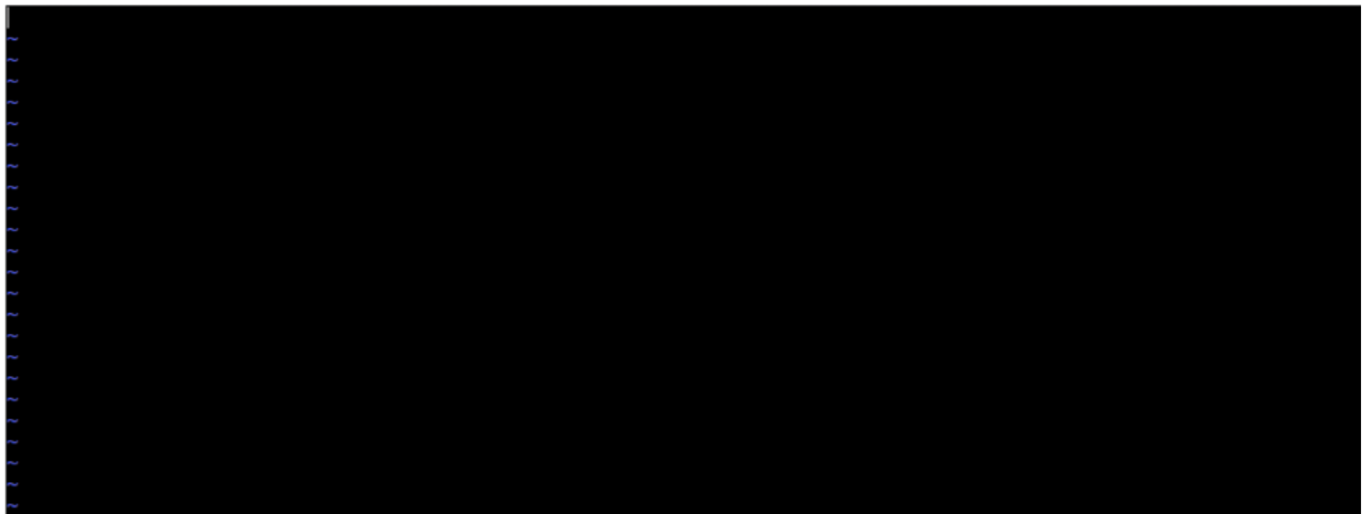
This will initialize an empty file for your Python script in Vim named *python_code.py* and then Git Bash would look like this:

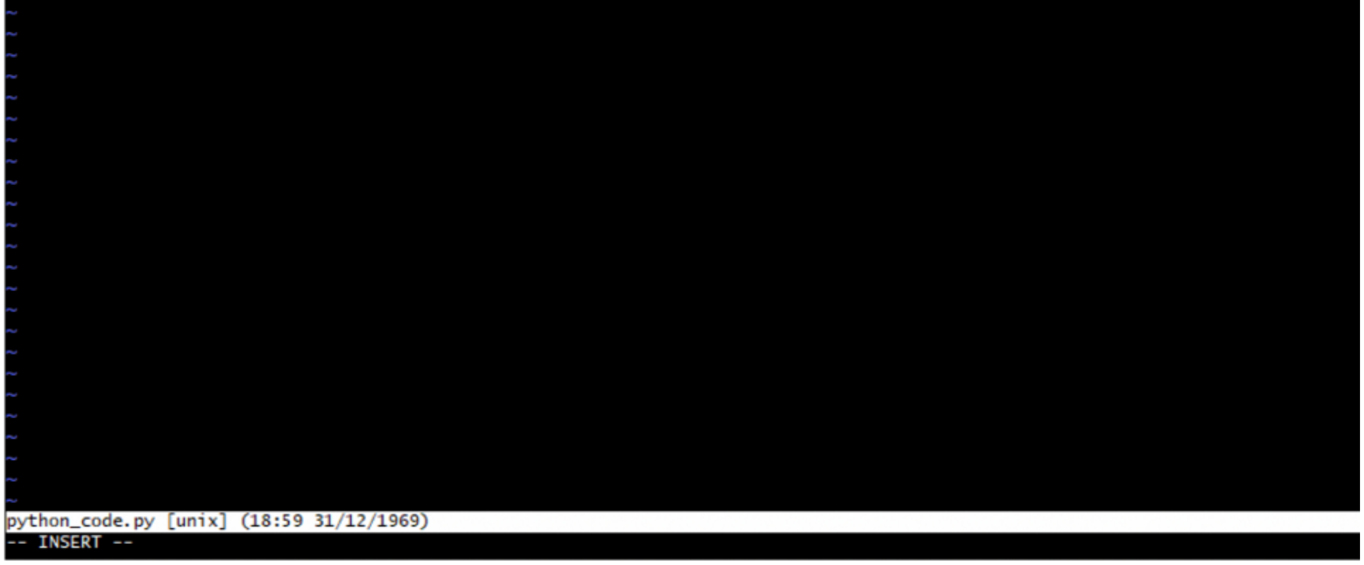


```
~
python_code.py [unix] (18:59 31/12/1969)
"python_code.py" [New File]
```

Vim interface for the Python script.

Start by pressing *i,* in order to switch to the **Insert** mode. You should see the word “INSERT” appear on the bottom of the window.

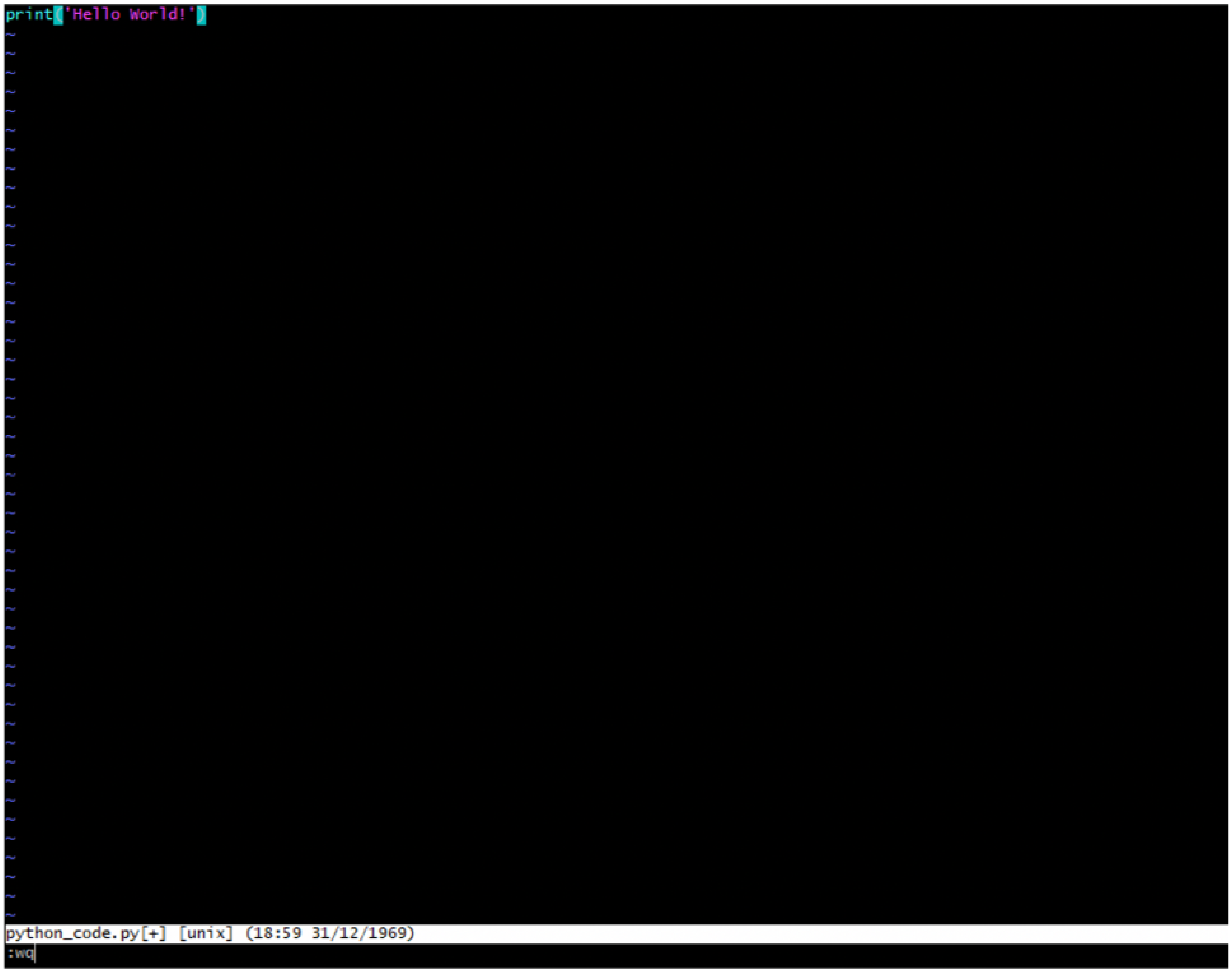




```
python_code.py [unix] (18:59 31/12/1969)
-- INSERT --
```

Switching to Inset mode.

Now, you can go ahead and add the Python code to print “Hello Github!”, as shown below:



```
print('Hello World!')
python_code.py[+] [unix] (18:59 31/12/1969)
:wq
```

Adding print statement.

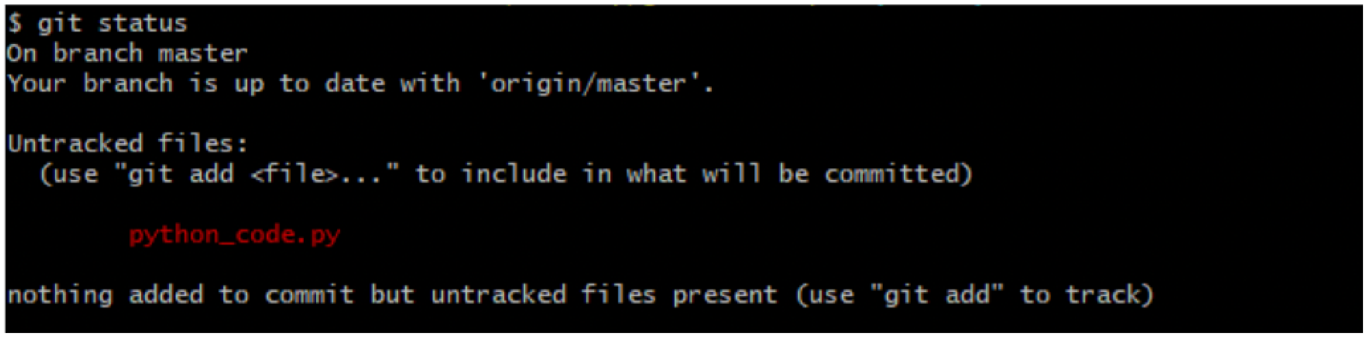
To save the file, press ESC and then type ":wq" and hit Enter to save and return to the Git Bash terminal.

Here, I am just using the Vim editor here because I am creating a file with only one line of code, and it let's me make changes to the repository quickly. For real projects, you should use an Integrated Development Environment (IDE), such as **Sublime** or **Atom**. Just make sure that the files are saved to your repository directory.

Now our local repository contains a new file, and therefore, it has changed and is not the same as the remote repository on Github. So if you run the following in the Git Bash terminal,

```
git status
```

the changes in the repository should be displayed in red, as shown below:



```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        python_code.py

nothing added to commit but untracked files present (use "git add" to track)
```

Changes in the local repository.

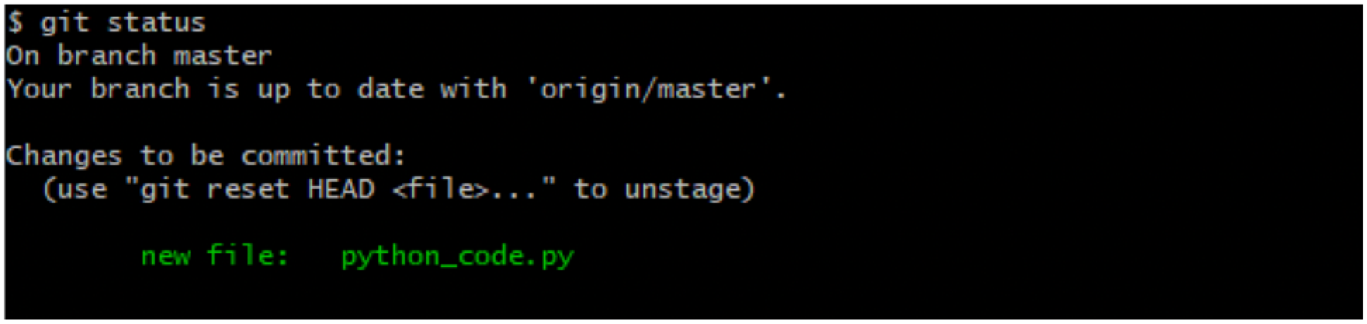
Updating the remote repository or sync-ing it with the local version consists of three steps:

1. Staging the changes.
2. Committing them.
3. Pushing them to the remote repository.

In order to stage the changes, you use the **git add** command. So, to stage the *python_code.py* file, we run the following in the Git Bash terminal:

```
git add python_code.py
```

If we run **git status** again, the staged changes will be displayed in green, as shown below:



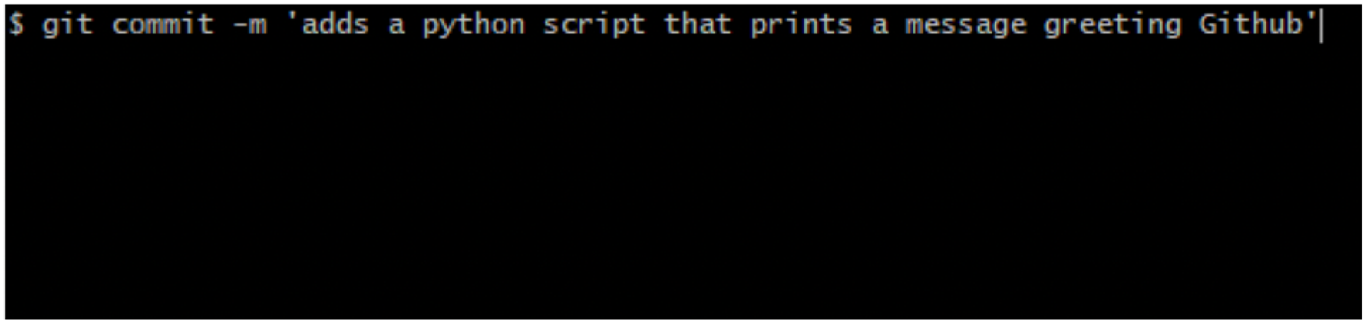
```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   python_code.py
```

Staging the changes to be committed.

The next step is committing the changes. In order to do that, we use the **git commit** command. So, to commit the *python_code.py* file, we run the following in the Git Bash terminal:

```
git commit -m <message>
```

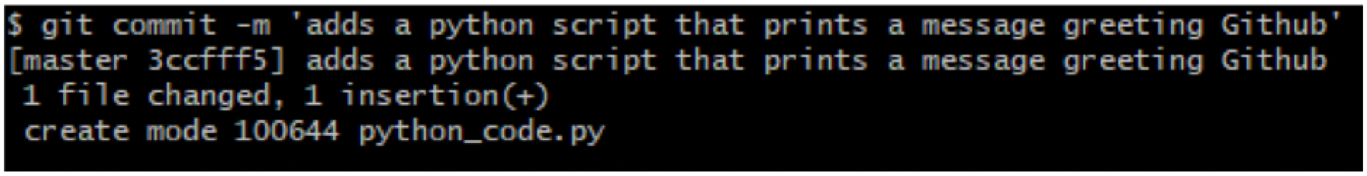
and you replace <message> with a brief description that explains what the changes are. So, for committing *python_code.py*, I am using the message 'adds a python script that prints a message greeting Github', as shown below:



```
$ git commit -m 'adds a python script that prints a message greeting Github'
```

Committing the changes.

And once the command is executed, a summary describing the changes committed should appear, as shown below.



```
$ git commit -m 'adds a python script that prints a message greeting Github'
[master 3ccfff5] adds a python script that prints a message greeting Github
 1 file changed, 1 insertion(+)
 create mode 100644 python_code.py
```

Summary of changes committed.

In the *python_code.py* case, the summary shows that it is just one file, with one insertion, where the insertion is the print statement that we added to the file. Had we performed more modifications such as adding or deleting more files or codes, then all of them would appear in the displayed summary as insertions and deletions.

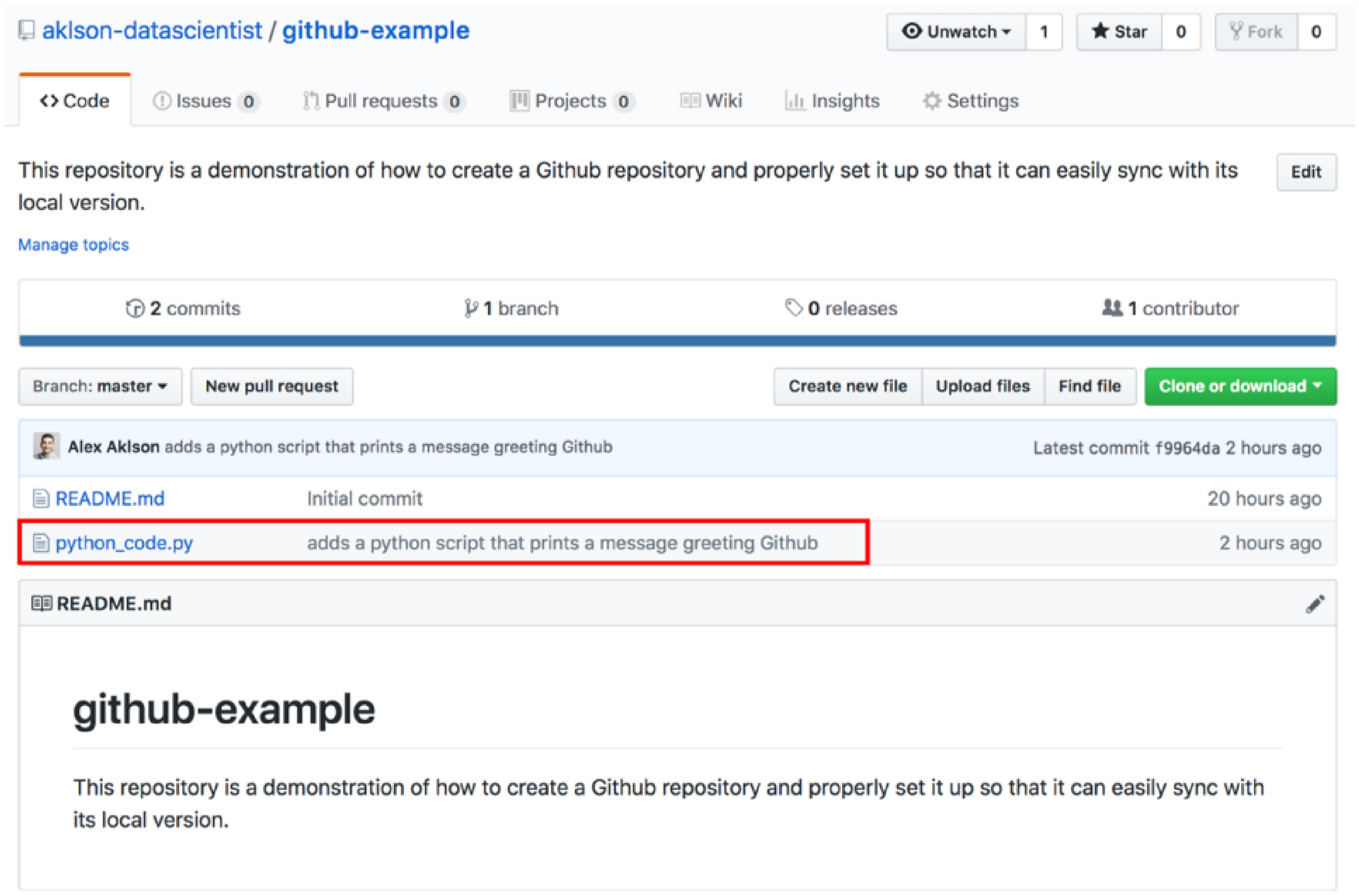
The final step is to push the changes to the remote repository. In order to do that, we run the following in the Gib Bash terminal:

```
git push origin master
```

or just

```
git push
```

And now if I check my **github-example** repository on Github, I should see the *python_code.py* file along with the message that I included when I ran the **git commit** command, as shown below:



This repository is a demonstration of how to create a Github repository and properly set it up so that it can easily sync with its local version.

Alex Aklson adds a python script that prints a message greeting Github   Latest commit f9964da 2 hours ago

README.md	Initial commit	20 hours ago
python_code.py	adds a python script that prints a message greeting Github	2 hours ago

README.md

github-example

This repository is a demonstration of how to create a Github repository and properly set it up so that it can easily sync with its local version.

Step 6: Congratulations!

Congratulations on successfully creating a Github repository and properly setting it up so that you are able to seamlessly update it as build your code and add more files to your project.

I hope you found this post helpful in at least getting you started with Github and understanding how to create a repository and push changes and commits to your repository in order to save your work and keep it up-to-date!

Git Github Windows