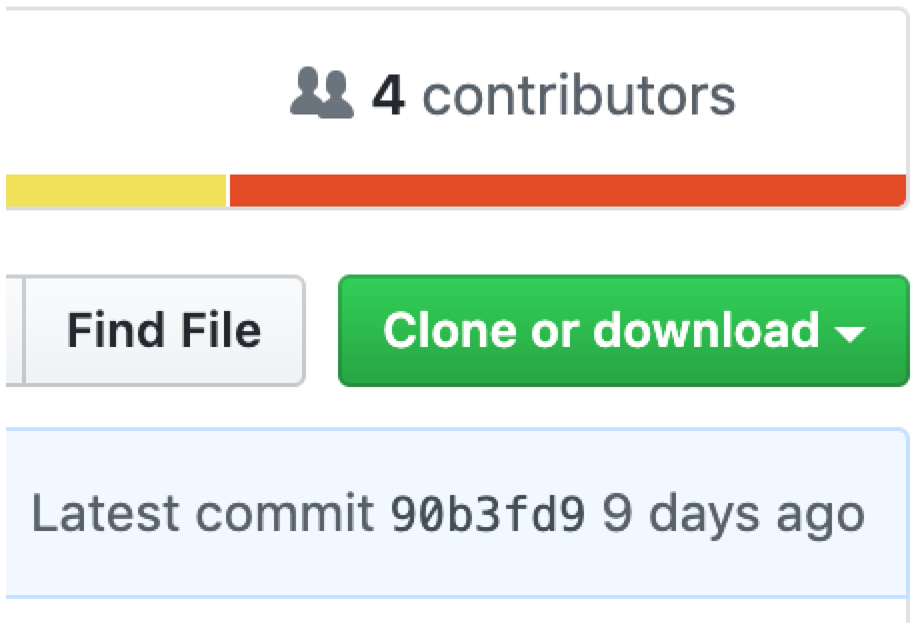
GitHub Tutorial

Part 1 Starting From Scratch

You want your own brand new branch!!!!

Steps

1. Find your repo on github.com
2. Then you gota click the green button in the below image and copy the link to a clipboard



Latest commit 90b3fd9 9 days ago

3. Then open a terminal and navigate to where you want to place the file and type in the following command followed by the link you just copied

   ```
   git clone <link you copied>
   ```
   JavaScript

   Note: exclude the <>

4. Now you can do the following command to see what branch you are on

   ```
   git branch
   ```
   JavaScript

   and you should get an output that looks like so

   ```
   Ricardos-MacBook-Pro:src ricardo$ git branch * WCTest
   codeClimateTest esther-showroom gh-pages hello-world-code lit-
   element-baseclass master r3molina/ch820/test-cafe
   ```
   JavaScript

   The * shows what branch I'm currently on (yours should show the * next to the master)

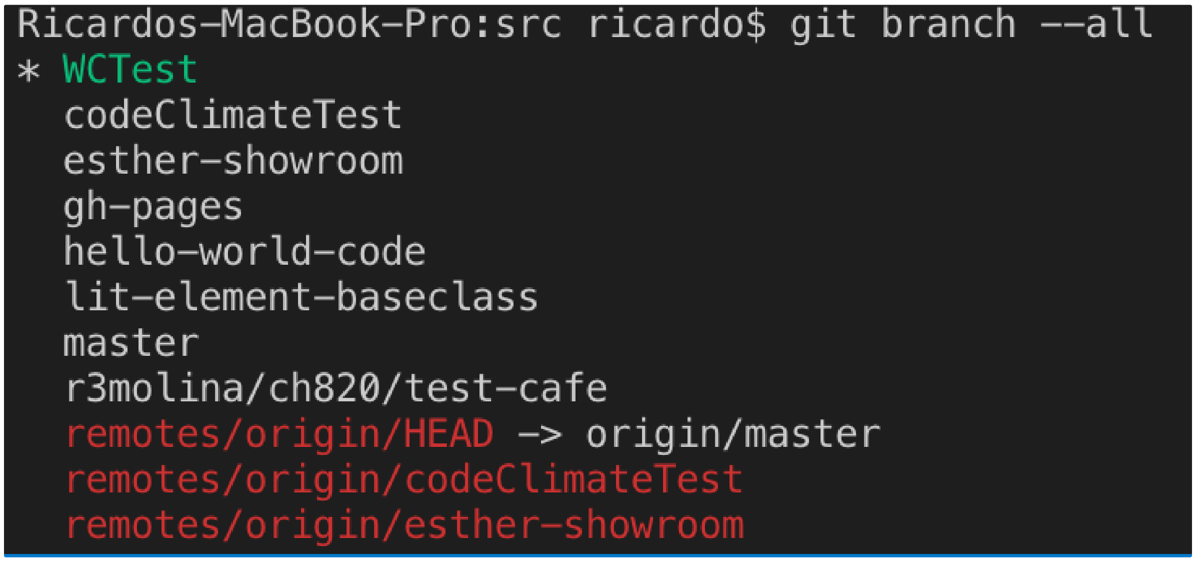
5. Done

Part 2. Pulling From a Non Master Branch

Steps

1. git fetch

2. use git branch - - all (the all flag will show you all the branches on the github) (theres no space between the dashes but this thing formats it as a single line if I remove it)



```
Ricardos-MacBook-Pro:src ricardo$ git branch --all
* WCTest
  codeClimateTest
  esther-showroom
  gh-pages
  hello-world-code
  lit-element-baseclass
  master
  r3molina/ch820/test-cafe
  remotes/origin/HEAD -> origin/master
  remotes/origin/codeClimateTest
  remotes/origin/esther-showroom
```

The Ones in white are the ones you currently have and can switch between freely the red ones on the other hand are the ones we are after

3. git checkout

```
git checkout --track origin/<branch your after>
```

JavaScript

for example if I wanted to use esther-showroom branch

```
git checkout --track origin/esther-showroom
```

JavaScript

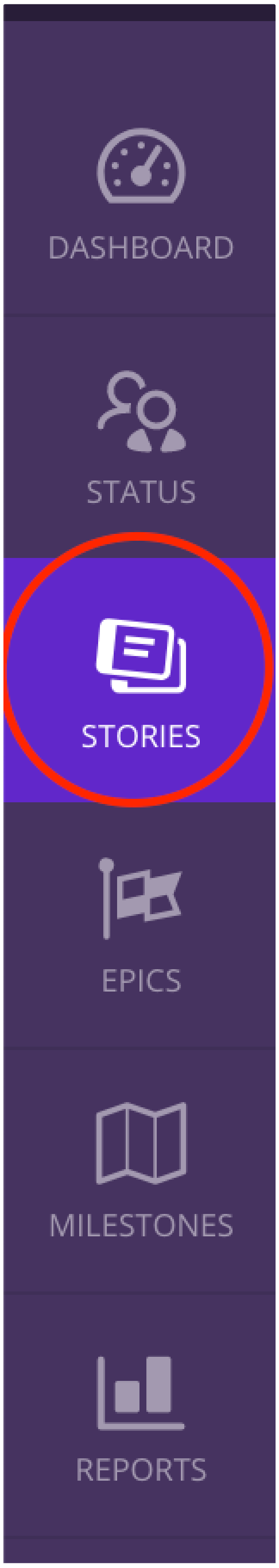
4. Then create a new branch

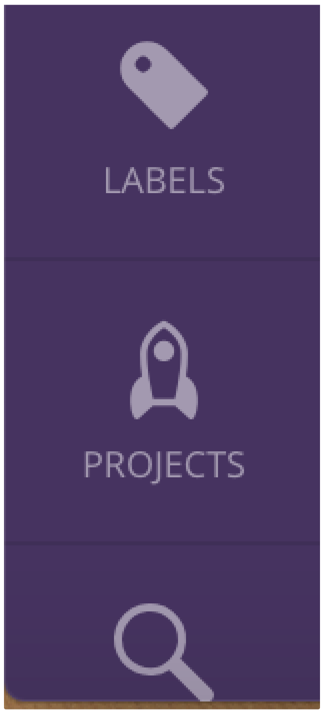
Part 3 Creating a new branch

1. At this point you should be on some kinda branch
2. run git checkout -b <follow Part 4> (note if you are just testing you can name this whatever you want.

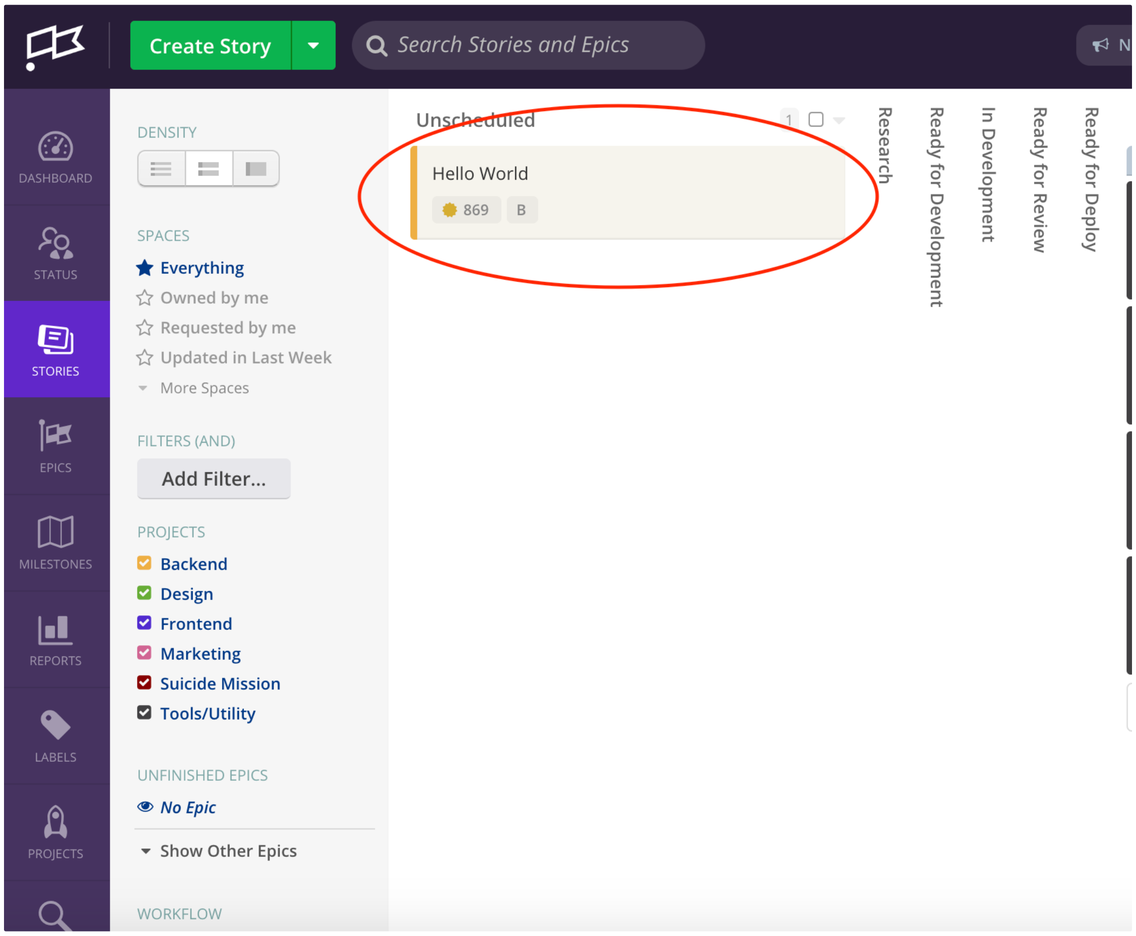
Part 4 Selecting branch name

1. Think of a name you want to call this branch it should let everyone know what you are currently working on. For Our project you will do the following steps to select a name.
    1. You need to open clubhouse so follow this link https://app.clubhouse.io/ucsdcse112sp19teameleven/projects
    2. Select stories from the left bar

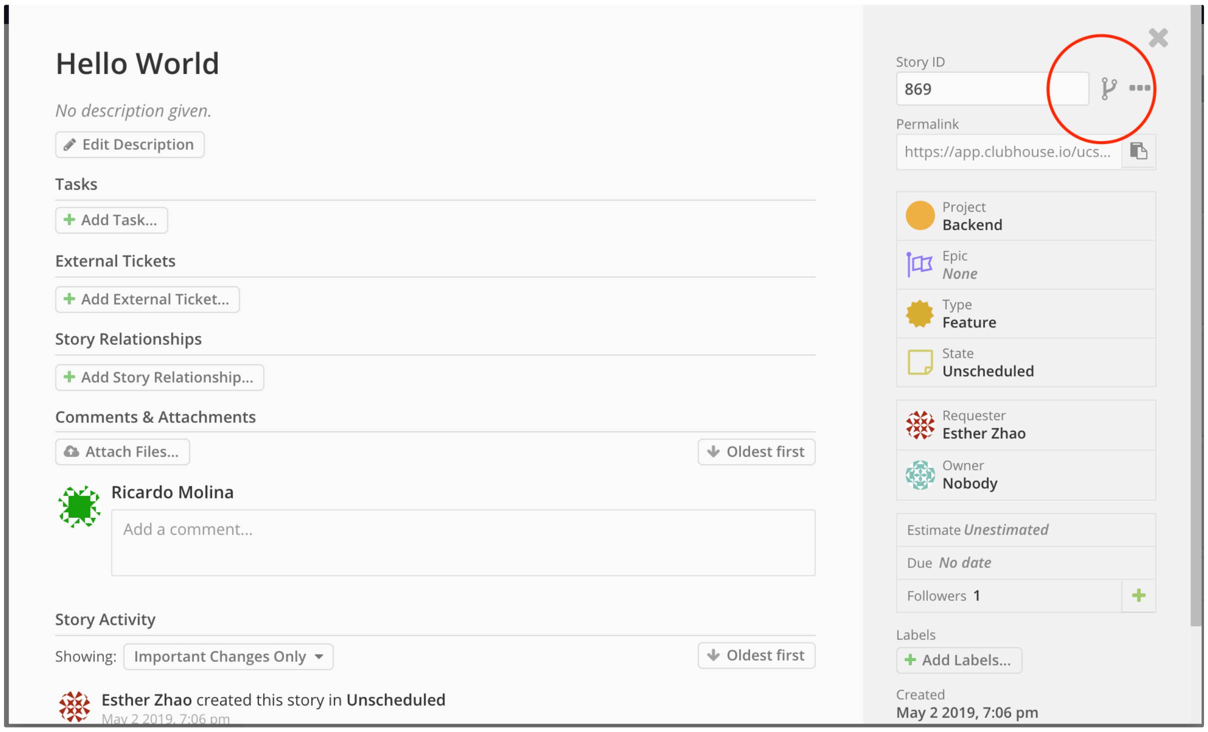

DASHBOARD
STATUS
STORIES
EPICS
MILESTONES
REPORTS



3. Select the Story you want to work on



4. Click the Git helpers button



Hello World

No description given.

Edit Description

Tasks

Add Task...

External Tickets

Add External Ticket...

Story Relationships

Add Story Relationship...

Comments & Attachments

Attach Files... Oldest first

Ricardo Molina

Add a comment...

Story Activity

Showing: Important Changes Only Oldest first

Esther Zhao created this story in **Unscheduled**

May 2 2019, 7:06 pm

Story ID: 869

Permalink: https://app.clubhouse.io/ucs...

Project: **Backend**

Epic: ***None***

Type: **Feature**

State: **Unscheduled**

Requester: **Esther Zhao**

Owner: **Nobody**

Estimate ***Unestimated***

Due ***No date***

Followers **1**

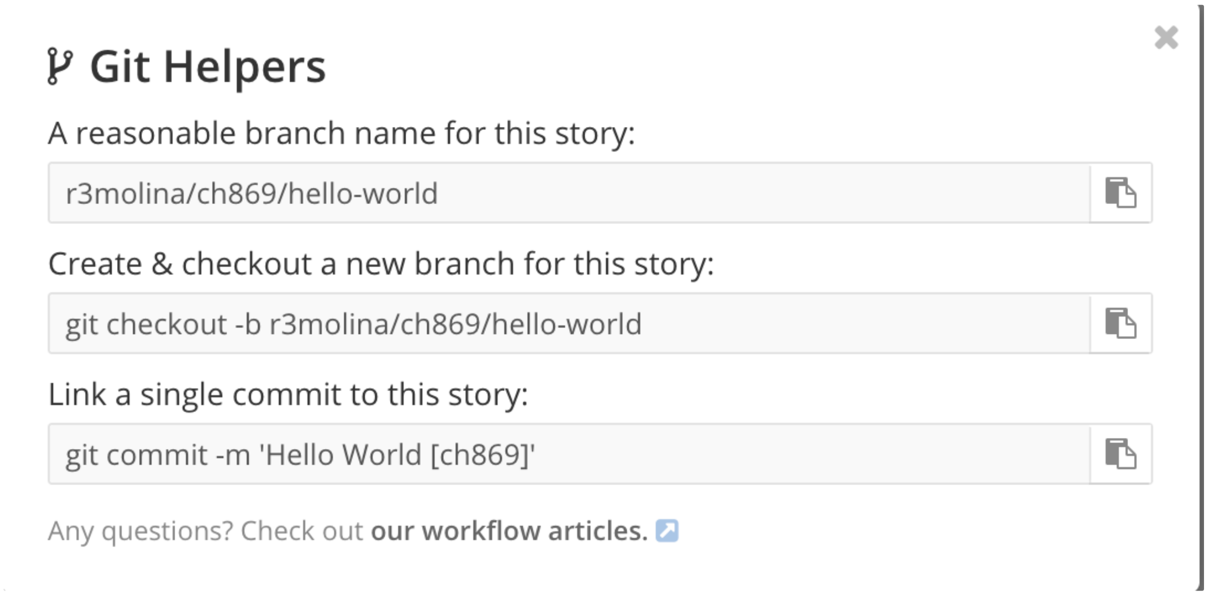
Labels

Add Labels...

Created

May 2 2019, 7:06 pm

5. A new window will appear that looks like this



Git Helpers

A reasonable branch name for this story:

```
r3molina/ch869/hello-world
```

Create & checkout a new branch for this story:

```
git checkout -b r3molina/ch869/hello-world
```

Link a single commit to this story:

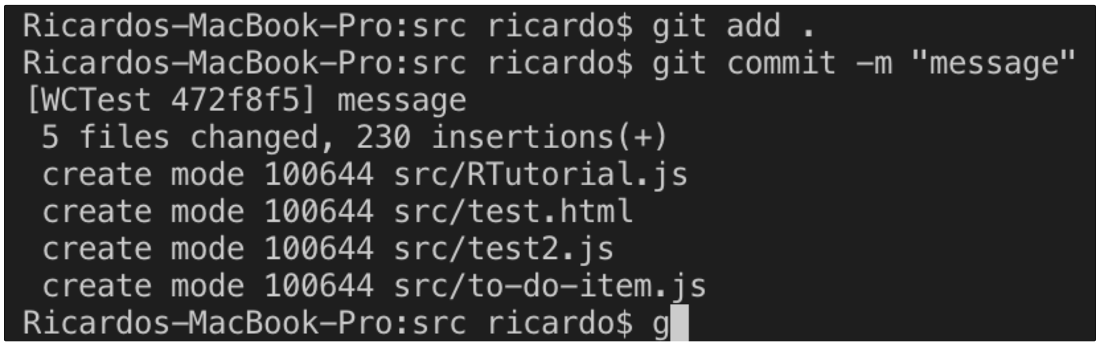
```
git commit -m 'Hello World [ch869]'
```

Any questions? Check out **our workflow articles.**

Option 1. is kinda pointless unless you want to rename your current branch but I've never done this and have no idea if its even possible

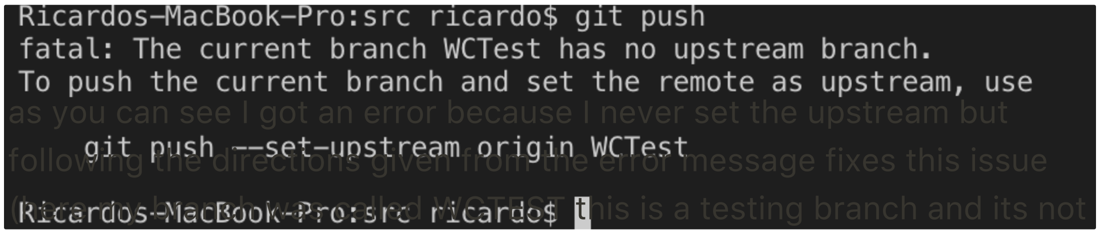
Option 2. Is the one you want to use if you copy and past that into the terminal it will create a new branch for you and then from there you simply enter the following commands into the terminal

1. double check your actually on the new branch so do a git branch (Instructions are in part 1 step 4)
2. then you will enter the commands git add, git commit -m <option 3 from above> (this is the line that reads git commit -m 'Hello World [ch869]'
   - it should like like this



```
Ricardos-MacBook-Pro:src ricardo$ git add .
Ricardos-MacBook-Pro:src ricardo$ git commit -m "message"
[WCTest 472f8f5] message
 5 files changed, 230 insertions(+)
 create mode 100644 src/RTutorial.js
 create mode 100644 src/test.html
 create mode 100644 src/test2.js
 create mode 100644 src/to-do-item.js
Ricardos-MacBook-Pro:src ricardo$ g
```

3. now do a git push



```
Ricardos-MacBook-Pro:src ricardo$ git push
fatal: The current branch WCTest has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin WCTest

Ricardos-MacBook-Pro:src ricardo$ 
```

as you can see I got an error because I never set the upstream but following the directions given from the error message fixes this issue (here my branch was called WCTEST this is a testing branch and its not on the github as I've never pushed it and plan to just keep it local so if you want to work and play around just never push it)

```
git push --set-upstream origin <your branch name>
```

JavaScript

4. Done

Part 5 Git Merge

Steps

1. checkout the branch that you want the changes to appear in
2. then enter the following command

```
git merge < branch you want to merge with (this one will not
be changed)>
```

JavaScript

3. fix merge conflicts
4. done

Part 6 Cleaning UP

When your all Done with a branch and its fully merged use

```
git branch -d <branch name>
```

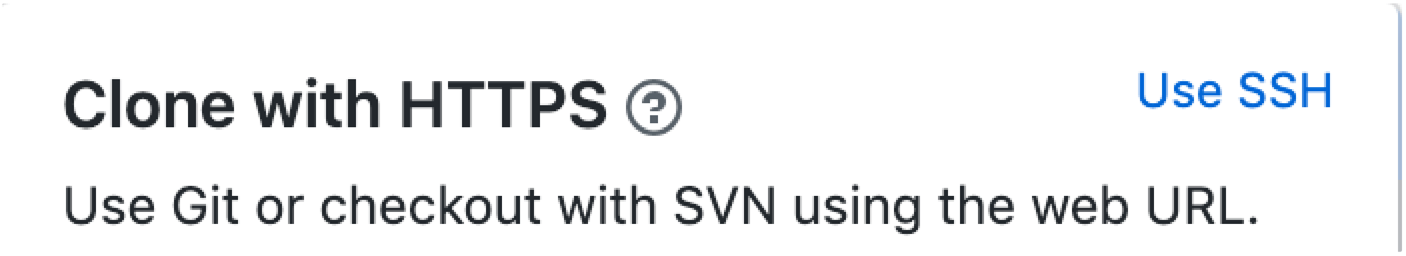
JavaScript

to Remove the branch

Set up SSH Keys

SSH vs. HTTPS

SSH and HTTPS are the two most commonly used protocols to transfer data between the local and server (GitHub), as you can see when you want to clone a repo:


Clone with HTTPS

Use Git or checkout with SVN using the web URL.

The biggest advantage of SSH over HTTPS is that once configured, you don't need to login every time to push code. Though you can also save credentials of HTTPS, people tend to use SSH more in a private repo. SSH seems to be faster too.

Steps

1. Generate a new ssh key. Use default settings by just hitting enter if you never do this before.

```bash
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Bash

2. Add key to ssh-agent. ssh-agent is a key management software that can automatically manage your authorization process.

```bash
$ eval "$(ssh-agent -s)" $ open ~/.ssh/config # Use vim or other editor
if you're not using Mac. $ # Add following text to ~/.ssh/config Host *
AddKeysToAgent yes UseKeychain yes IdentityFile ~/.ssh/id_rsa $ ssh-add
-K ~/.ssh/id_rsa
```

Bash

3. Add public key to GitHub.

```
# See your public key. $ cat ~/.ssh/id_rsa.pub > ssh-rsa some-
encrypted-text your_email@example.com # Copy it. This command is for
Mac too. # If it doesn't work, use other commands instead or just copy
it manually. $ pbcopy < ~/.ssh/id_rsa.pub
```

Bash

Now open GitHub Settings https://github.com/settings/keys. Click `New SSH Key`, and paste the key here.

4. Test connection.

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

Bash

Reference

https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent