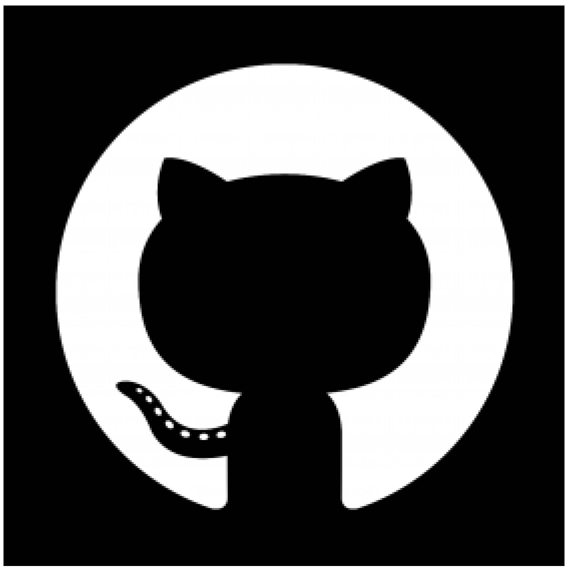
The Ultimate Github Collaboration Guide - Jonathan Mines - Medium

This is just one of many ways to collaborate on a project using GitHub. But it's one I would suggest if you're just starting out working...

By Jonathan Mines

May 30, 2018 06:18 PM · 6 min. read · View original



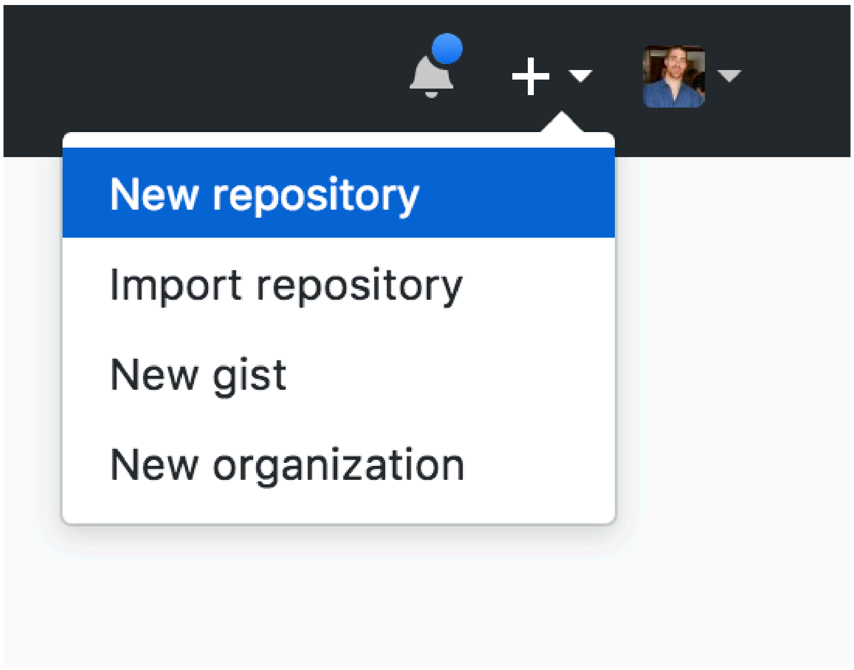
This is just one of many ways to collaborate on a project using GitHub. But it's one I would suggest if you're just starting out working with a team and haven't established a git flow yet or know where to start in establishing one.

Step 1: Initialize a New Project

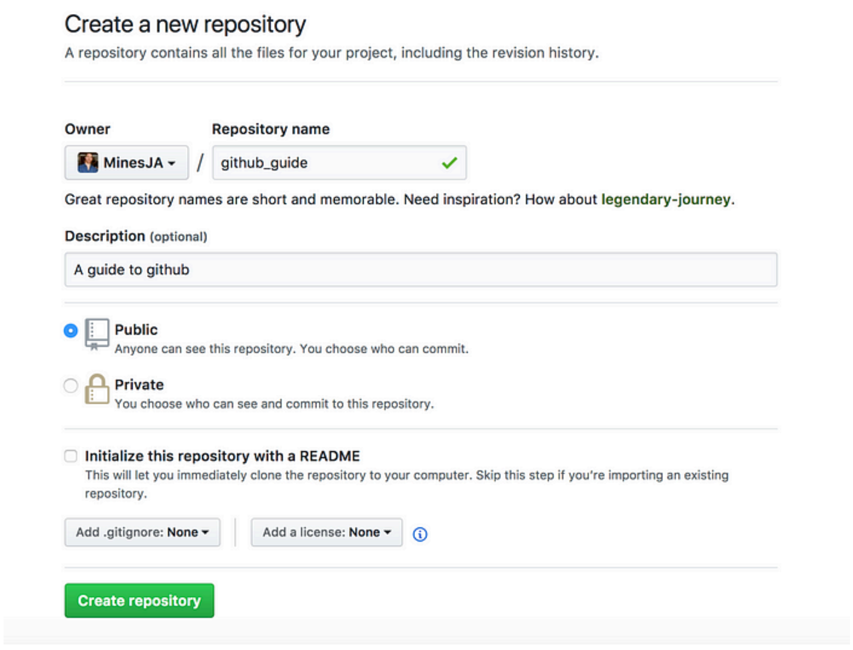
Create a new project/directory from the command line

```
$ rails new github_guide
```

Go to Github and click the '+' button in the rop right corner and select 'New Repository'.



Then fill out the Repository name and the Description fields. Keep it public, and do not "Initialize this repository with a README". Don't change anything else. Click "Create repository".



Create a new repository

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

Owner: MinesJA / Repository name: github_guide ✓

Great repository names are short and memorable. Need inspiration? How about legendary-journey.

Description (optional): A guide to github

[✓ Public] Anyone can see this repository. You choose who can commit.

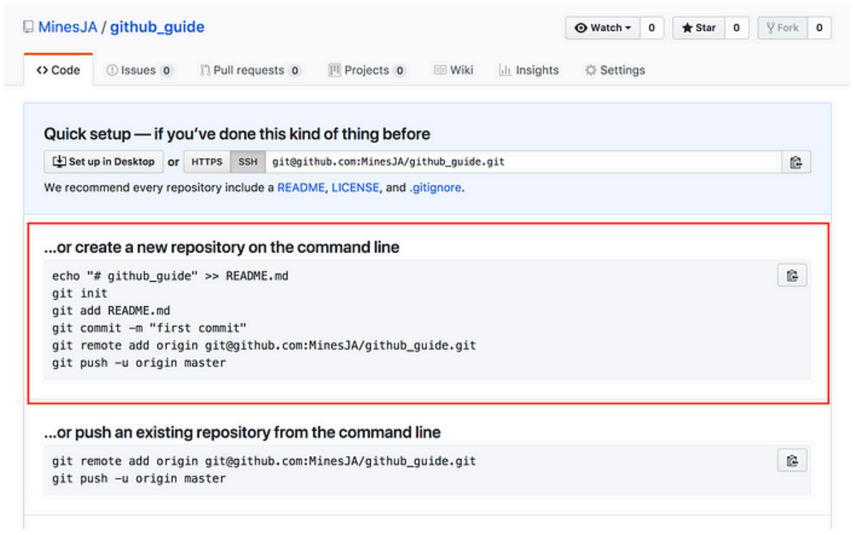
Private — You choose who can see and commit to this repository.

☐ 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

Next you'll see the setup page. These are the instructions for connecting the Repo you just created in Github (Remote) to the directory you created in your terminal (Local).



MinesJA / github_guide

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:MinesJA/github_guide.git

We recommend every repository include a README, LICENSE, and .gitignore.

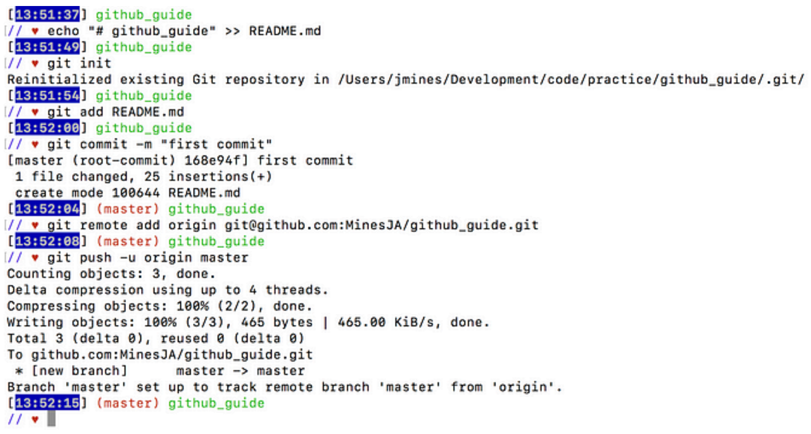
...or create a new repository on the command line

```
echo "# github_guide" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:MinesJA/github_guide.git
git push -u origin master
```

...or push an existing repository from the command line

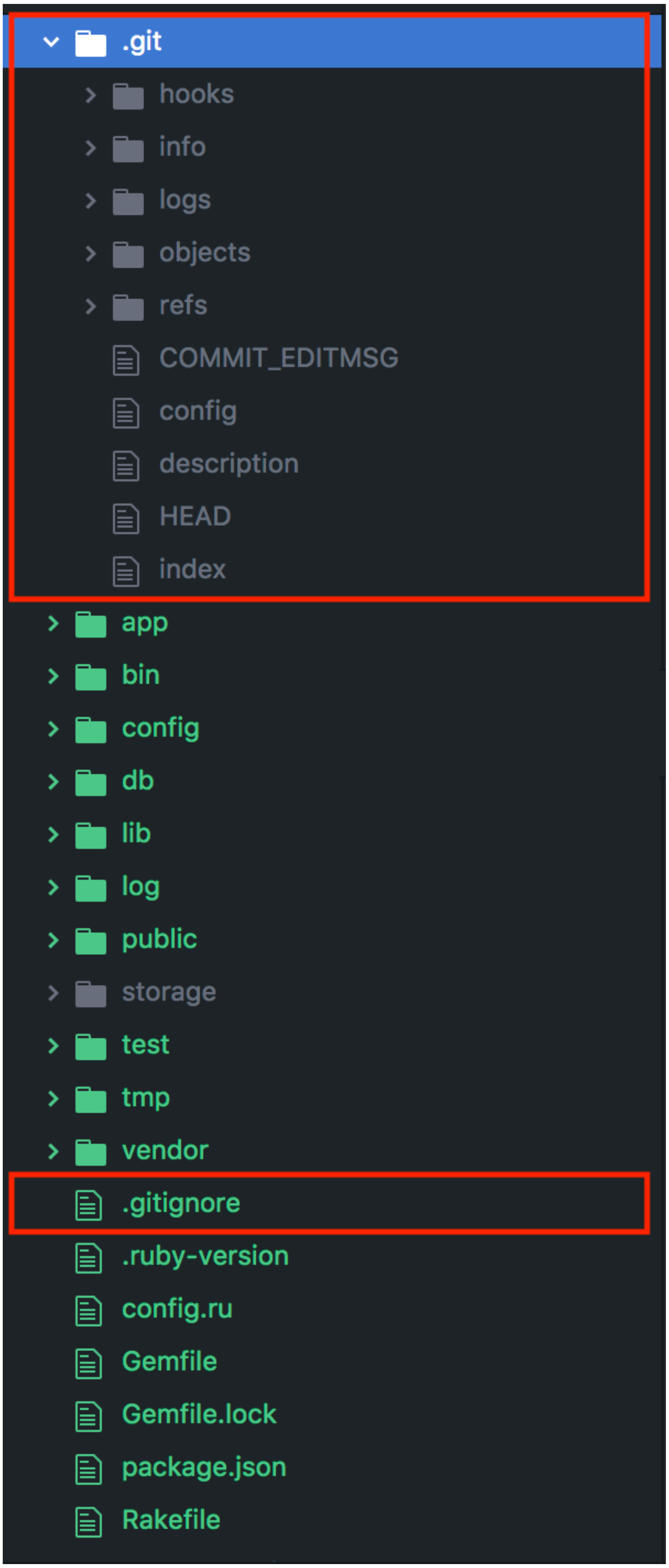
```
git remote add origin git@github.com:MinesJA/github_guide.git
git push -u origin master
```

Paste the lines in the red box line by line starting with "echo..." into the terminal while you're cd'ed into the directory you just created locally. Your terminal should look like this when you're done:



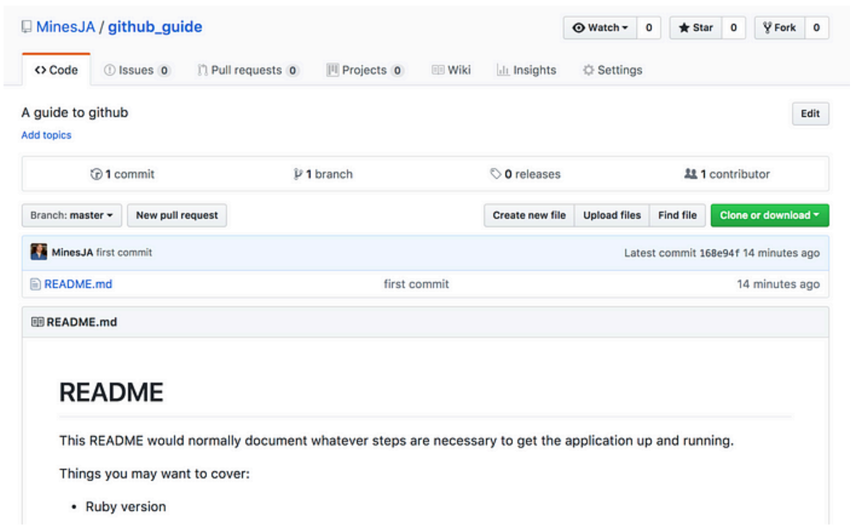
```
[13:51:37] github_guide
// ♥ echo "# github_guide" >> README.md
[13:51:49] github_guide
// ♥ git init
Reinitialized existing Git repository in /Users/jmines/Development/code/practice/github_guide/.git/
[13:51:54] github_guide
// ♥ git add README.md
[13:52:00] github_guide
// ♥ git commit -m "first commit"
[master (root-commit) 168e94f] first commit
 1 file changed, 25 insertions(+)
 create mode 100644 README.md
[13:52:04] (master) github_guide
// ♥ git remote add origin git@github.com:MinesJA/github_guide.git
[13:52:08] (master) github_guide
// ♥ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 465 bytes | 465.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:MinesJA/github_guide.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[13:52:15] (master) github_guide
// ♥
```

This adds a ‘.git’ folder to your repo, connects you to your remote Github Repo and also gives you a ‘.gitignore’ file.



```
.git
    hooks
    info
    logs
    objects
    refs
    COMMIT_EDITMSG
    config
    description
    HEAD
    index
app
bin
config
db
lib
log
public
storage
test
tmp
vendor
.gitignore
.ruby-version
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
```

And if you go to your Github Repo page, you'll see the ReadMe that you intialized with and the reference to the first commit you made.



A guide to github

1 commit | 1 branch | 0 releases | 1 contributor

MinesJA first commit — Latest commit 168e94f 14 minutes ago

README.md	first commit	14 minutes ago

README.md

README

This README would normally document whatever steps are necessary to get the application up and running.

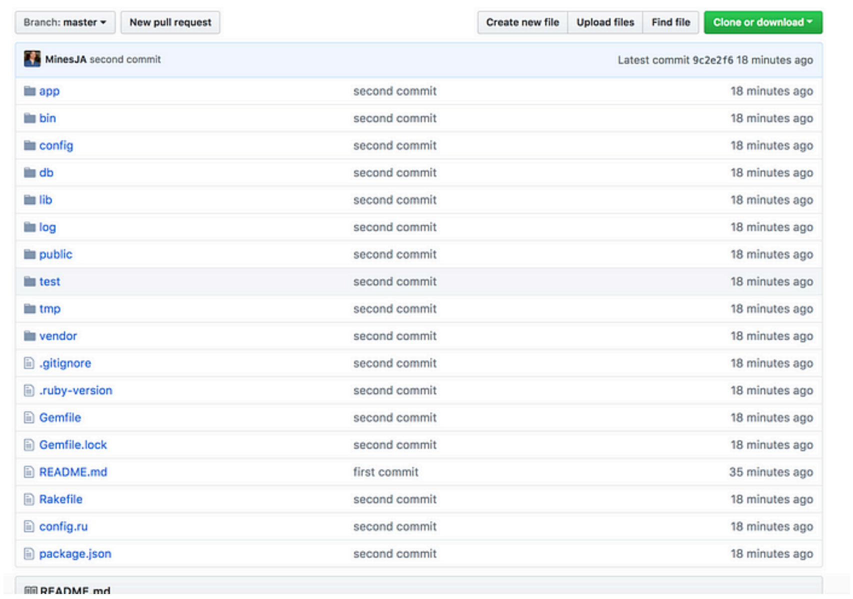
Things you may want to cover:

- Ruby version

Now let's get this Repo up to date. Go back to your terminal and git add, git commit, and git push:

```
$ git add .$ git commit -m "Second commit"$ git push
```

Now check out your repo. It should have all the files you created your local directory with along with a new commit id (9c2e2f6):



MinesJA second commit — Latest commit 9c2e2f6 18 minutes ago

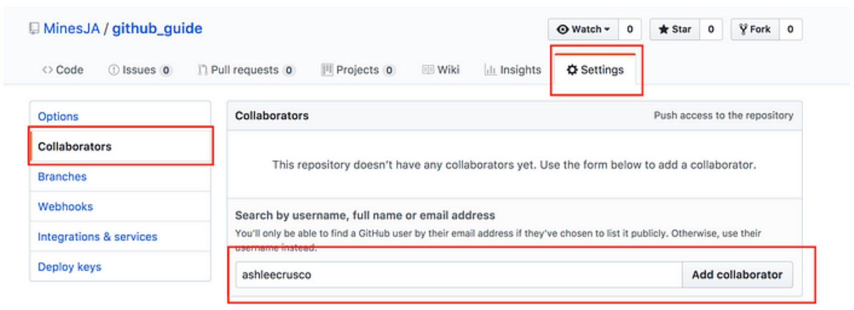
app	second commit	18 minutes ago
bin	second commit	18 minutes ago
config	second commit	18 minutes ago
db	second commit	18 minutes ago
lib	second commit	18 minutes ago
log	second commit	18 minutes ago
public	second commit	18 minutes ago
test	second commit	18 minutes ago
tmp	second commit	18 minutes ago
vendor	second commit	18 minutes ago
.gitignore	second commit	18 minutes ago
.ruby-version	second commit	18 minutes ago
Gemfile	second commit	18 minutes ago
Gemfile.lock	second commit	18 minutes ago
README.md	first commit	35 minutes ago
Rakefile	second commit	18 minutes ago
config.ru	second commit	18 minutes ago
package.json	second commit	18 minutes ago

You’re initialized and ready to start working!

Step 2: Setup Your Team

You’re a team player, so you’re going to need to add your team to your repo so they can collaborate. Once your team is added as collaborators they’ll be able to push, merge, and a ton of other destructive things so make sure you’re only adding your teammates.

Click on the “Settings” tab of your rep, then “Collaborators” then search for Github users and add them by clicking “Add Collaborator”:



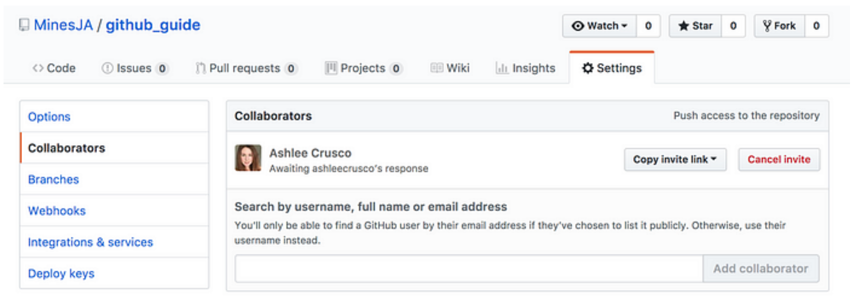
This repository doesn’t have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You’ll only be able to find a GitHub user by their email address if they’ve chosen to list it publicly. Otherwise, use their username instead.

ashleecrusco

They’ll receive an email letting them know you added them and will be listed as a collaborator.

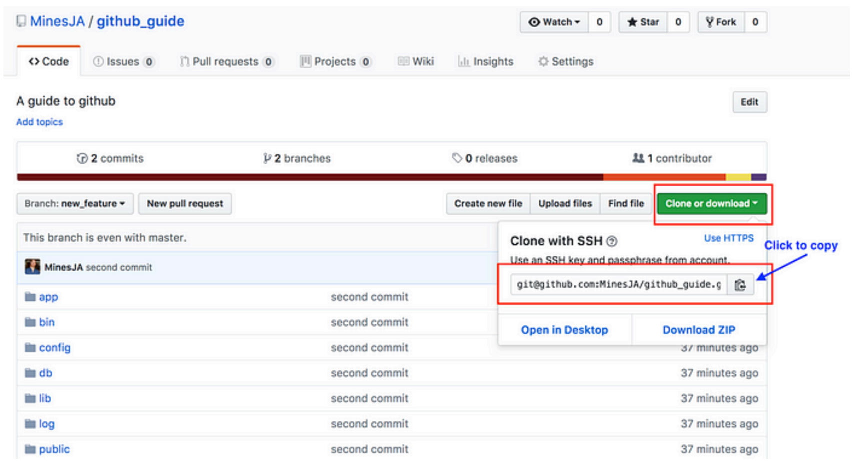


Ashlee Crusco
Awaiting ashleecrusco’s response

Search by username, full name or email address

You’ll only be able to find a GitHub user by their email address if they’ve chosen to list it publicly. Otherwise, use their username instead.

If you’re a collaborator, go to the Github Repo page, Git Clone the project, and cd into the directory. **Don’t fork it!** Forking will copy it in a new Repo to your Github page, but you don’t want that — you want to collaborate on the same Github Repo with your teammates.



A guide to github

This branch is even with master.

Use an SSH key and passphrase from account.

git@github.com:MinesJA/github_guide.g

MinesJA second commit

app	second commit	
bin	second commit	
config	second commit	37 minutes ago
db	second commit	37 minutes ago
lib	second commit	37 minutes ago
log	second commit	37 minutes ago
public	second commit	37 minutes ago

```
$ git clone
git@github.com:MinesJA/github_guide.git$ cd
github_guide/
```

And now you're ready to collaborate!

Step 3: Collaborating

When you're using git to work on the same project with multiple people, there's one central rule you must follow:

THE MASTER BRANCH SHOULD ALWAYS BE DEPLOYABLE

The way to keep Master deployable is to create new branches for new features and merge them into Master when they're completed. Here's how that works.

Step 3a: Branches

To start, branches should always represent features. For example, if you want to add the ability for a user to login you should probably create a branch called "user_authentication" and in that branch you should only update what you need to to enable a user to login.

It's also important when collaborating that your team picks features that don't have overlapping code. For example, you shouldn't be working on a "user_login" branch at the same time that your teammate is working on a "user_logout" branch because the chances that you're working on the same files and are writing overlapping code are very high.

So let's say you want to create the User model. In your terminal create a new branch:

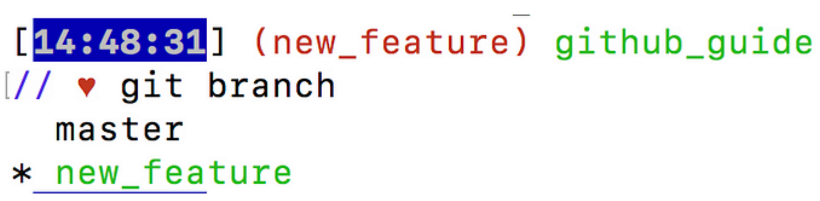
```
$ git co -b create_user
```

"co" is short for "checkout" which is used to switch between branches. Adding the "-b" and a name at the end creates a new branch and then moves into that new branch for us.

You should be able to verify this with the command:

```
$ git branch
```

Which should produce:



```
[14:48:31] (new_feature) github_guide
// ♥ git branch
  master
* new_feature
```

You're now in your new branch and can start coding away.

Note: As a general rule, you should git add frequently and git commit when you finish something that allows your code to work (ends up being a couple times an hour). For example, when you finish a method and the code base works, git commit like so:

```
$ git commit -m "Added function to allow Users to
say 'Hello World'"
```

Step 3b: Submitting Pull Requests

Your team spent all day and night working on their separate features in their various branches. They come back the next day with their completed features and want to merge them back into Master to be deployed.

Determining your Git Flow is a huge part of working in a team, but here's one Git Flow you could adopt for now:

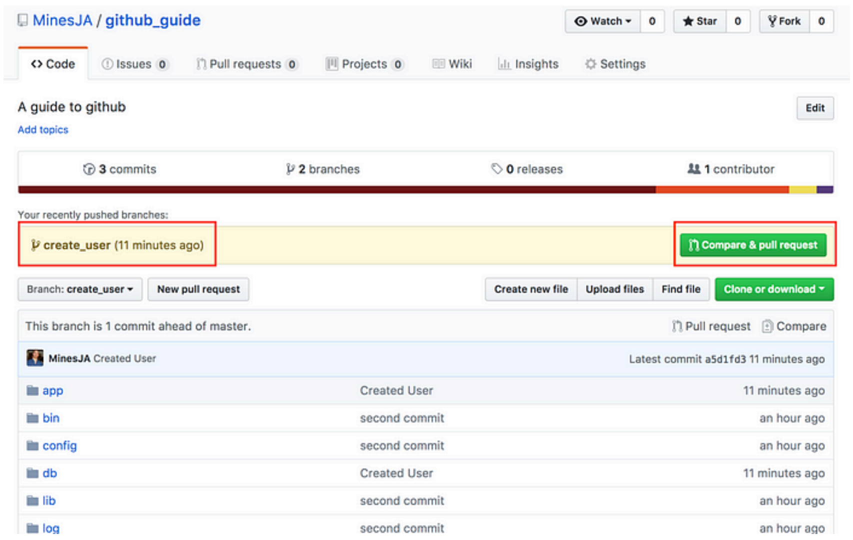
First, determine who's going to be in charge of handling merging. The less people acting independently on merging the better so for a team of 4 it would probably behoove you to have one official "Reviewer" or "Merge Master".

Next, have everyone git push their branches:

```
$ git push
```

Now go to the Github Repo page. You should see the branch you pushed up in a yellow bar at the top of the page with a button to "Compare & pull request".

Note: Alternatively, you can select the branch in the drop-down "Branch:" menu and select the branch you just pushed up. You'll then have a "Pull request" and "Compare" button on the right hand side.



A guide to github

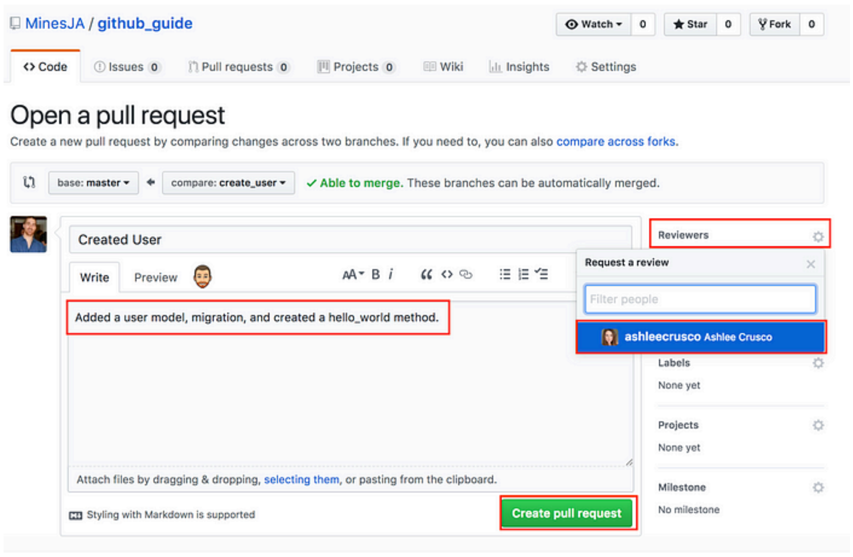
create_user (11 minutes ago)

This branch is 1 commit ahead of master.

MinesJA Created User — Latest commit a5d1fd3 11 minutes ago

app	Created User	11 minutes ago
bin	second commit	an hour ago
config	second commit	an hour ago
db	Created User	11 minutes ago
lib	second commit	an hour ago
log	second commit	an hour ago

Click "Compare & pull request". This will take you to the "Open a pull request" page. From here, you should write a brief description of what you actually changed. And you should click the "Reviewers" tab and select whoever your team decided would be the "Merge Master". When you're done, click "Create pull request".



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

✓ Able to merge. These branches can be automatically merged.

Created User

Added a user model, migration, and created a hello_world method.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

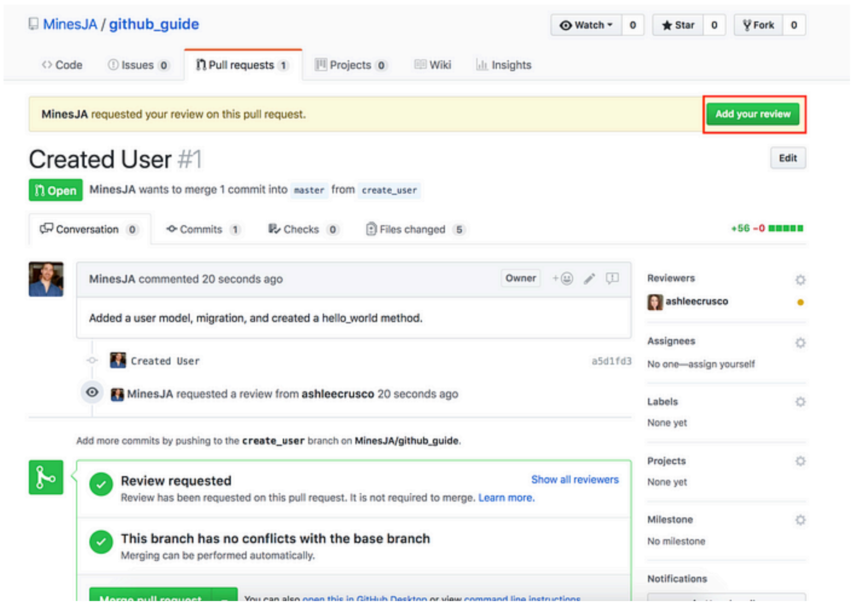
ashleecrusco Ashlee Crusco

Step 3c: Merging Pull Requests

Note that if you're a collaborator, you can merge your own pull requests. But, again, if you're working on a team, it makes more sense to have one person do all the merging and everyone else submit "Pull Requests" and assign the "Master Merger" as a reviewer just to make sure you're dealing with any merge conflicts one branch at a time.

So, assuming you are the one charged with taking care of all merges and someone has assigned you as "Reviewer" on a pull request, when you login to your Github you'll notice you have a notification letting you know that someone has assigned you as a reviewer. You'll also noticed a yellow bar indicating one of your teammates as "requested your review on this pull request."

Go ahead and click the "Add your review" button.



MinesJA requested your review on this pull request.

Created User #1

MinesJA wants to merge 1 commit into `master` from `create_user`

MinesJA commented 20 seconds ago

Added a user model, migration, and created a hello_world method.

`Created User` `a5d1fd3`

MinesJA requested a review from **ashleecrusco** 20 seconds ago

Add more commits by pushing to the **`create_user`** branch on **MinesJA/github_guide**.

Review requested Show all reviewers
Review has been requested on this pull request. It is not required to merge. Learn more.

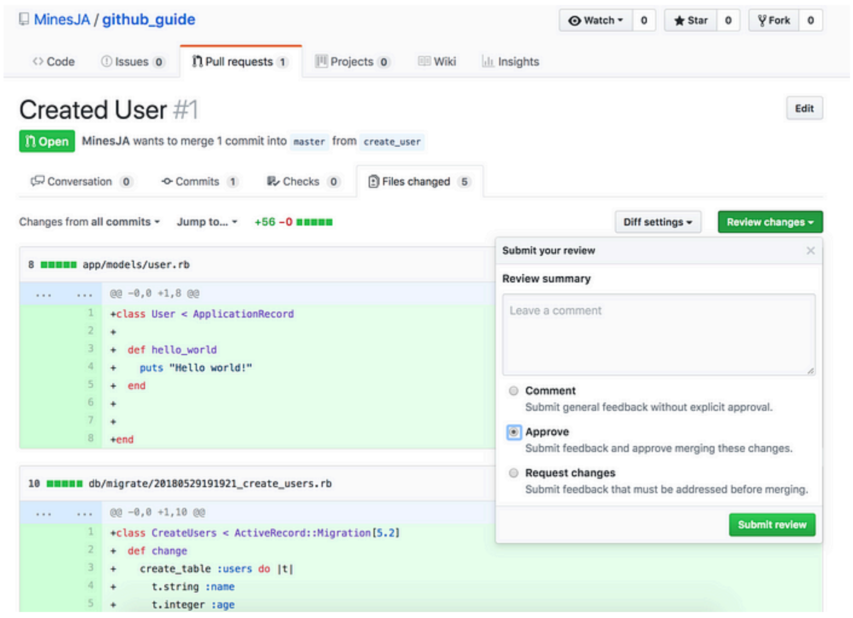
This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions

Reviewers: ashleecrusco
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone

This will take you to the Pull Request page. How you move forward from here is up to you and your team. If you're working together I would use every morning to sit down as a team and go through pull requests together. If you do that, you won't have to necessarily leave long-winded, detailed Review Summaries.

However, if you're working remotely, this will be your main tool for letting the requester know if they need to make changes or if you're going to merge their request.



Created User #1

MinesJA wants to merge 1 commit into `master` from `create_user`

+56 −0

8 app/models/user.rb

```
@@ -0,0 +1,8 @@
+class User < ApplicationRecord
+
+  def hello_world
+    puts "Hello world!"
+  end
+
+
+end
```

10 db/migrate/20180529191921_create_users.rb

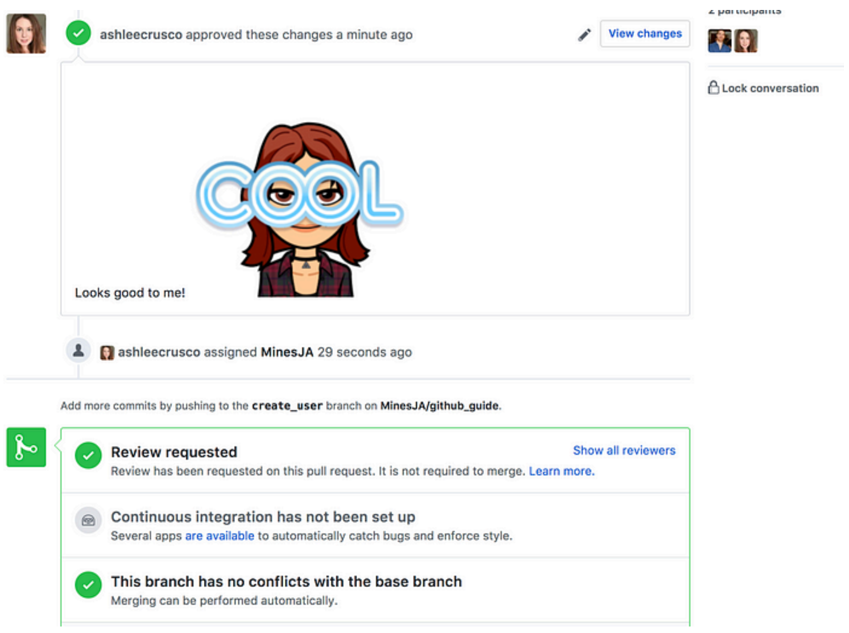
```
@@ -0,0 +1,10 @@
+class CreateUsers < ActiveRecord::Migration[5.2]
+  def change
+    create_table :users do |t|
+      t.string :name
+      t.integer :age
```

Comment
Submit general feedback without explicit approval.

[✓ Approve]
Submit feedback and approve merging these changes.

Request changes
Submit feedback that must be addressed before merging.

When you click "Submit review" on the "Review changes" drop-drown your review will now exist as a comment on the pull request thread.



ashleecrusco approved these changes a minute ago

View changes

Lock conversation

Looks good to me!

ashleecrusco assigned MinesJA 29 seconds ago

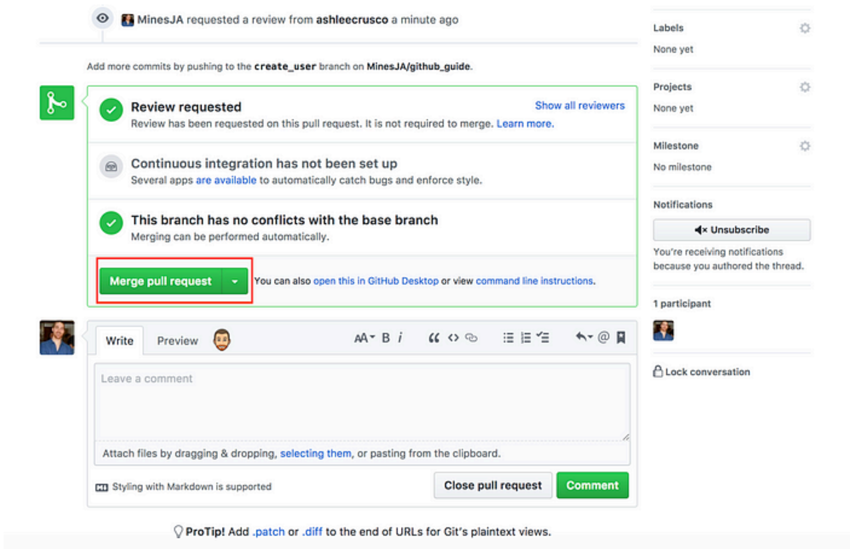
Add more commits by pushing to the **create_user** branch on **MinesJA/github_guide**.

Review requested Show all reviewers
Review has been requested on this pull request. It is not required to merge. Learn more.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

When you’re satisfied with the pull request, go to the bottom of the pull request and click “Merge pull request”.



MinesJA requested a review from **ashleecrusco** a minute ago

Add more commits by pushing to the **create_user** branch on **MinesJA/github_guide**.

Review requested Show all reviewers
Review has been requested on this pull request. It is not required to merge. Learn more.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request | You can also open this in GitHub Desktop or view command line instructions.

Write | Preview

Leave a comment

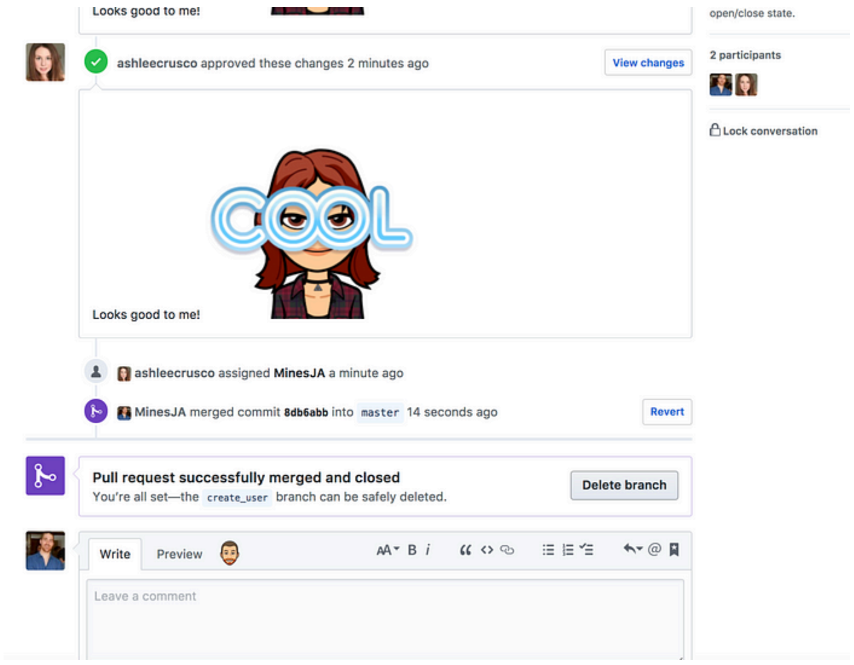
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported | Close pull request | Comment

Labels: None yet
Projects: None yet
Milestone: No milestone
Notifications: Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation

ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

You’ll then see a “Pull request successfully merged and closed” message and a button to “Delete branch” which you should click.


Looks good to me!

open/close state.

ashleecrusco approved these changes 2 minutes ago

View changes

2 participants

Lock conversation

Looks good to me!

ashleecrusco assigned MinesJA a minute ago

MinesJA merged commit 8db6abb into master 14 seconds ago

Revert

Pull request successfully merged and closed
You're all set—the create_user branch can be safely deleted.

Delete branch

Write Preview

Leave a comment

Step 4: Rinse, Repeat

And that's pretty much it! Keep adding new branches for new features and then coming together as a team to merge them into master. Keep master clean and deployable and don't try to merge more than one branch at a time and you should be good to go.

****

Here's the repo I used for this demonstration. I'll add the text of this article into the ReadMe. Feel free to make a pull request if there's anything you want to change or if you just want to test out your pull request skills!

https://github.com/MinesJA/github_guide