Simple Learn to use Git.

1. Installing Git:
2. Windows: <https://git-scm.com/download/win>

Download the correct version for your OS (32bit or 64bit) Defaults to 64bit

Download.

Follow the prompts - the default install options are probably good.

B. Linux (Debian/Ubuntu): sudo apt-get install git

2. Setup an account on <https://github.com>

I will need your user name to invite you to use the Automation Technology Club

Repository.

<https://github.com/automation-technology-club>

Hopefully you’ll learn how to create a repository, clone a repository, make a new branch, and push changes to github. There are really just a few things you need to know to get started.

And github tells you most of them, Jeremy created a small README.md file that also gives some hints as to what to do. This document should cover the rest.

Git is best used at the command line (at least I found it easier to deal with). The Windows version does install a GUI if you’d rather use it. I’m not going to use for these lessons. But it should be similar to use, once you understand the basic terminology used.

# Part 1: (Create a local repository)

Open your file manager, and goto the “documents” folder - you’ll want to create a temporary folder just for learning. Double click the new folder you just created, and create another folder with your name - “LeRoy” folder. Double click that folder.

Now create a text file called - “README.md” -

Open that file, and just put some text in it and save it.

Something like “I was here - Today Aug 14, 2017 - LeRoy” and save.

Right click anywhere in that folder and click on “open Git Bash” if you are using windows or “open in terminal” if your on linux.

A BASH shell prompt should open up.

We are about to create a new repository.

Type:

git init

You should see something like:

“Initialized empty Git repository in …..” where … is the directory you just created.

Git init - is used to create a new repository, it is going to be the first thing you type if you are creating a new repository for yourself.

Next Type:

git status

Status will tell you various information about the current state of your repository. You can use this to find files to add to the repository, or what branch you are on. It’s a very helpful command.

You should see something like:

“Untracked files:

(use “git add <file>....” to include in what will be committed)

README.md

Nothing added to commit but untracked files present (use “git add” to track)”

A tracked file is one that the repository will keep track of changes made.

(you might notice, GIT tells you what to do here…..)

Next We are going to add the untracked file to the repository.

Type:

git add README.md

Add doesn’t display anything unless there is an error - the error I have seen is if you tell it to track a file that doesn’t exist.

Wild cards can also be used, and you can track directories as well.

So if you had a lot of files:

git add \* is valid.

Sometimes at this point I recheck the status (git status) - Let do that just so we can see what has changed.

Type:

git status

You should see something like:

“Changes to be committed:

(use “git rm --cached <file>....” to unstage)

New file: README.md “

The new file is now GREEN. And what it’s tell us is a way to untrack the file before we commit it. Once it’s commited, the repository will remember it, even if it’s been removed.

We are happy now, but still need to make the commit to make the changes permanent.

Type:

git commit -am ‘first commit’

You should see the “commit” hash, how many files changed, the number of insertions, and other

Information.

The -am switch tells Git that you also want to stage already tracked files, (-a), and that you want to leave a message for the commit (-m). Leaving a message is alway a good idea, it gives an idea of what you were doing at the time of the commit.

Congratulations you just made your first GIT repository.

STEPS:

git init

git status

git add <files>.....

git commit -am ‘message….’

Easy - 4 steps.

## 1a Make Changes to local repository.

Now let’s make a change to the README.md file - it can be anything but you’ll want to change it so you know it’s been changed.

I will add “Hello World” to my file. Save the change, and go back to the BASH prompt

Type:

git status

You should see:

…… Modified: README.md …..

Again, tells you want what to do:

“git add <files> and/or git commit -a”

So type:

git commit -am “2nd commit added Hello World”

You should see the commit hash.

You have already seen both of those commands - so lets learn something new here:

Type

git log

You should see a list of the all the commits, and the comments, who created the commits, and the date.

If you ever want to go back in time you just use the git checkout command.

You will need that long hash commit number - Let’s go back to the first commit.

Type:

git checkout 123455949034xxxxxxx

You should see a message now telling you that you are in a Detached space and that you can look around, make changes, a new branch ect.

If you open your README.md file now, you’ll notice that it is from the first commit.

Let’s not make any changes at this point, and go back to the master branch.

Type:

git checkout master

New Command:

git checkout

git log

\*git branch will show you what your current working branch of the tree is.

After making changes to your program, your to your work, and saving those changes. You should go into the repository, and commit the changes with a message that tells what you did.

This could help if something you change in your code breaks - you can go back in time and figure out what happend. If you don’t commit each change that ability could be lost.

The above just creates just creates a local copy (repository).

In my mind the best part of GIT is the ability to keep your code online and being able to grab a current copy of it anywhere on any machine.

# PART 2: (Create a online repository)

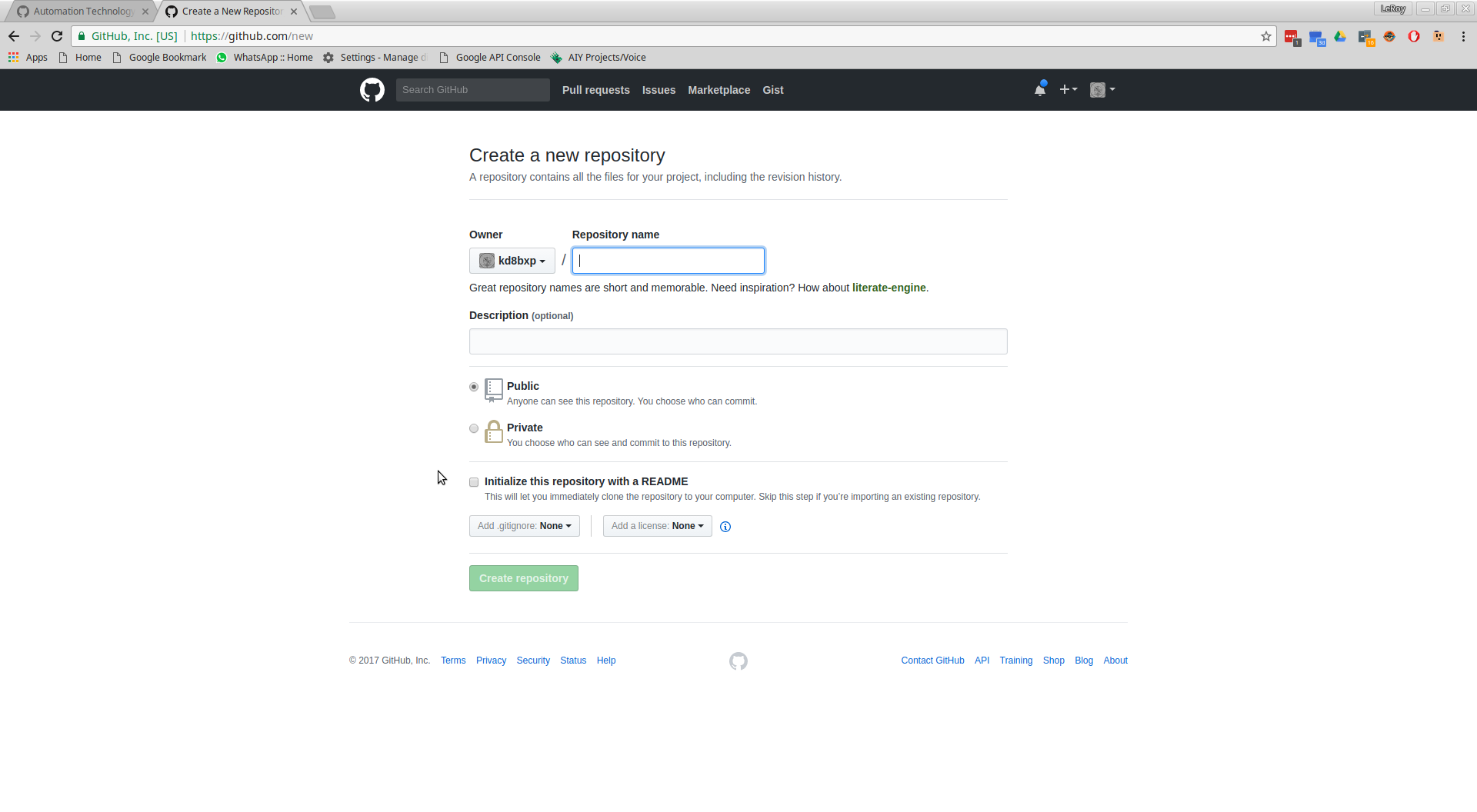
So we are now going to setup a online repository.

Goto <https://github.com> if you haven’t setup your account yet, or need to log in, do so now.

You should see a “plus” sign up in the right hand corner of your screen, click this.

And click “new repository”

You should see a screen that looks like this:



You can call your new repository anything you want - I’m calling mine “test”

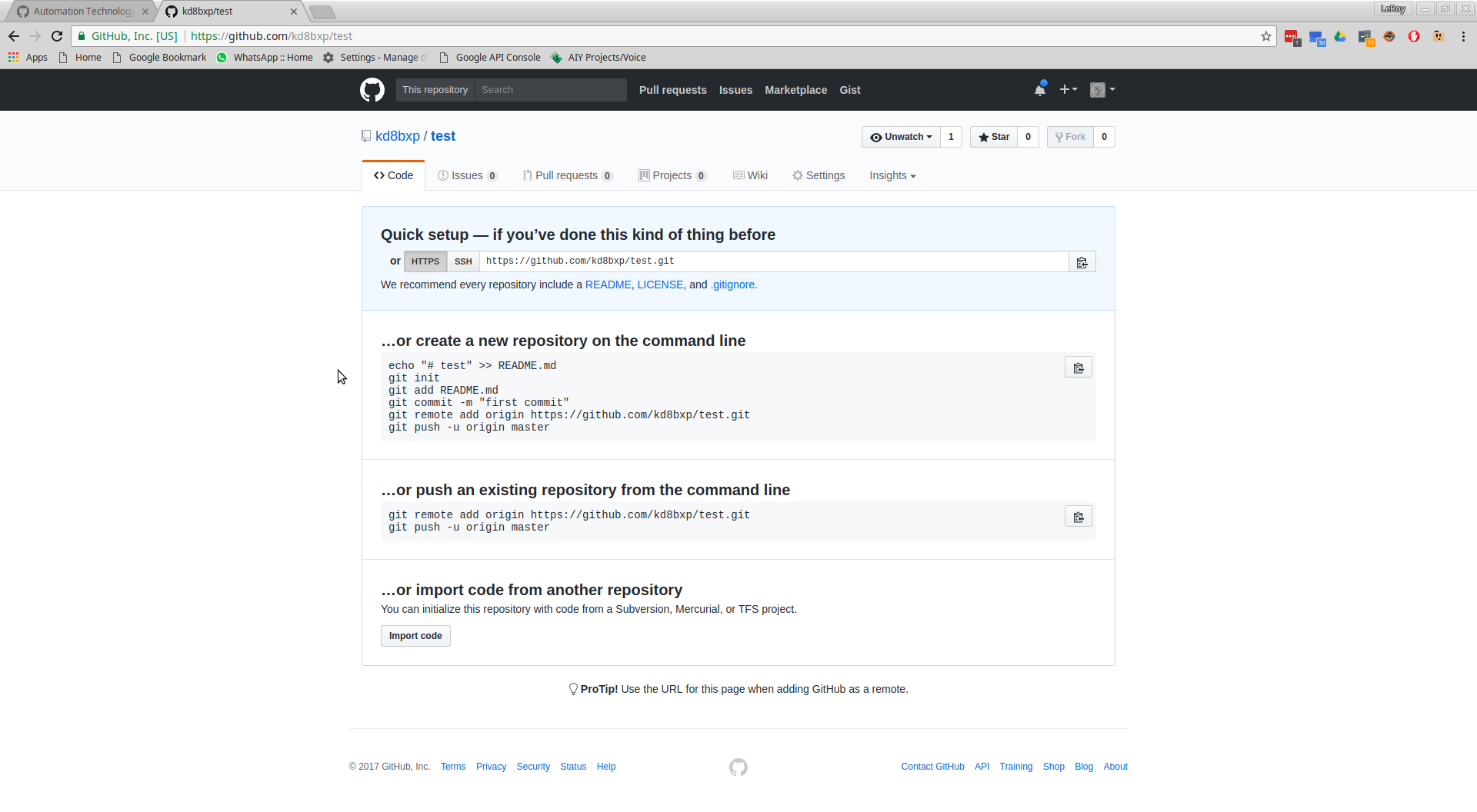
You can put a description if you like or leave it blank.

Everything else on the screen should be left alone at this point.

(Public will let everyone find your repository, And we already have a README file so no need to create a new one).

Click the green “Create repository”

And you should see a screen like this:



On this screen we see a few options:

We need the 2nd option since we already have a LOCAL repository.

(But if you look at the first option, the first 4 lines are just what we did to create our local repository)

So you can see github even tells you what you need to know to create or push a current repository to it.

USING OPTION 2: (or push an existing repository from the command line).

Go back to your bash prompt and type the 1st line that starts with:

Type:

git remote add origin <https://github.com/yourswillbedifferentfrommine.git>

If all goes right you shouldn’t see anything on your screen.

Next Type the 2nd line:

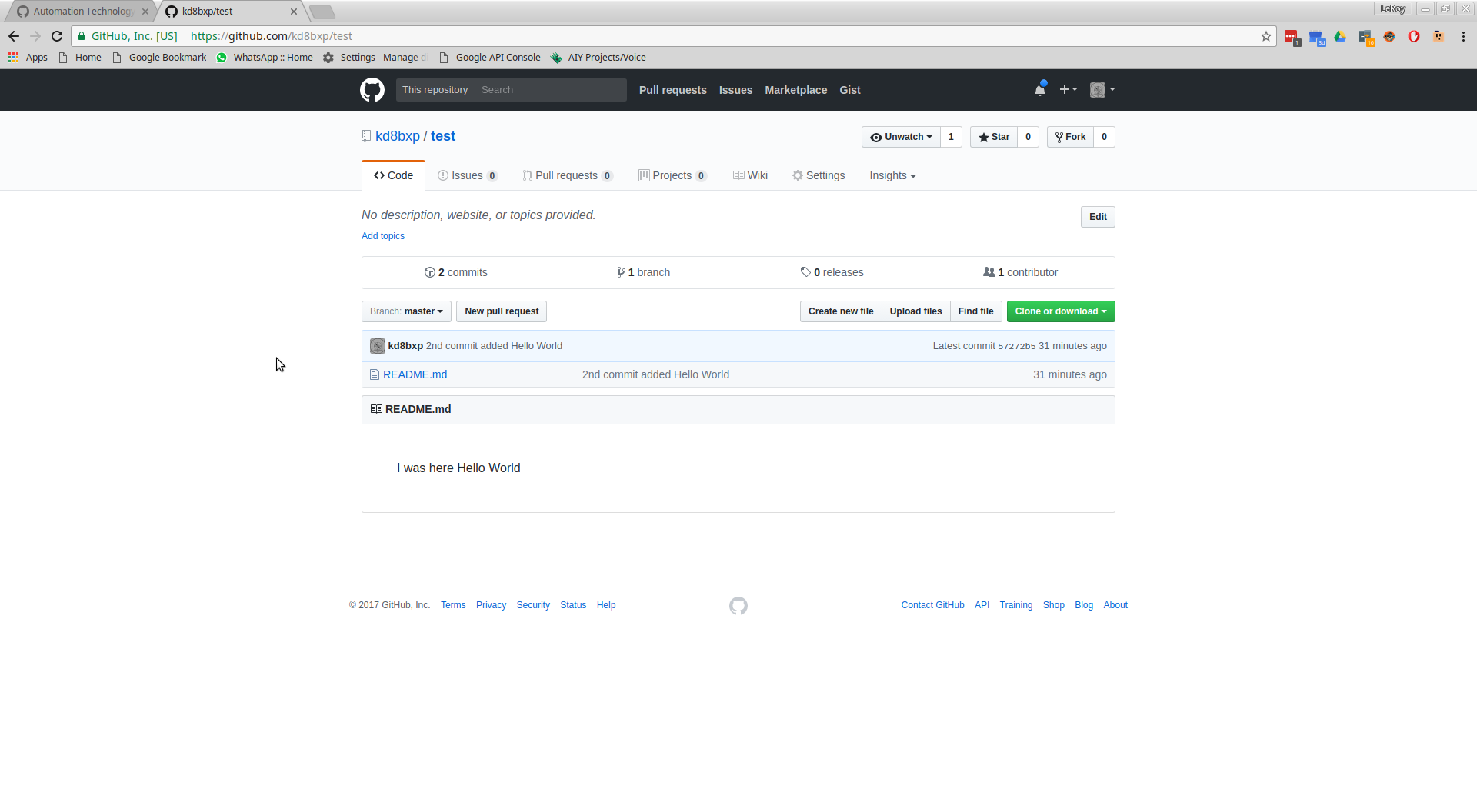
git push -u origin master

You should see a prompt for your username, and password.

And then a bunch of informational hash stuff.

It shouldn’t take very long, so refresh the webpage and you should see your files on github

(There is a number of things you can do here on this page)



You should see on the left side there are 2 commits (if you click on that you can look at the old commit just like we did locally), you should also see there is 1 branch, and other useful information.

You will also notice that “README.md” turns into the information file that is displayed on the opening page.

\*There are a number of additional features here, such as pull requests, issues, and a project wiki\* All of which are beyond this scope.

Learned how to PUSH a repository to Github (BTW: Bitbucket, and other online Git repositories are very similar)

New Commands:

git remote add origin <https://github.com/yourswillbedifferentfrommine.git>

git push -u origin master

\*\* git push -u origin master is only used the first time, each additional time you need to push to your online repository just issue “git push” - git is “smart” and knows to push it to the master branch \*\*

# Part 3: (Cloning a repository)

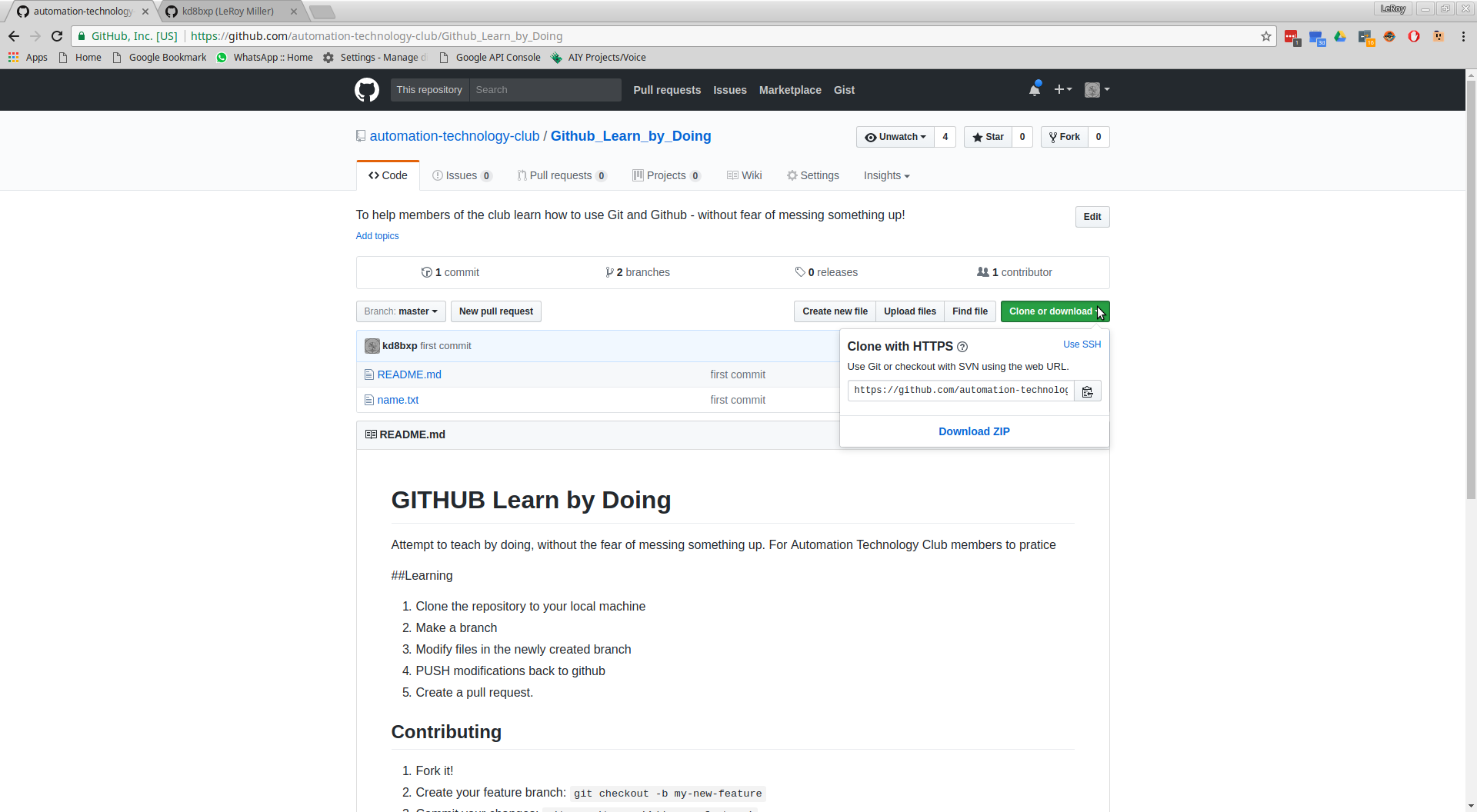
We are going to learn how to CLONE and make a new branch to work on:

For this next step, I will need to add you to the Automation Technology Club (Organization) repository.

For this we are going to clone the <https://github.com/automation-technology-club/Github_Learn_by_Doing> repository.

Goto the above URL, and you should see a green button that says “Clone or download”

Click that button, you should see this screen:



Click the clip-board icon, you’ll need that url to clone this.

Go back to your BASH shell, and type

cd ..

(dir for windows, or ls for linux) You should be in your temporary directory we created earlier

If your not, get there.

Once you verify you are in that directory, type:

git clone <https://github.com/automation-technology-club/Github_Learn_by_Doing.git>

It might take a few minutes (you shoud see a status update, and finally a “done.” message)

Type:

dir or ls (to get a directory now)

You should now see a new directory called

“Github\_Learn\_by\_Doing”

Type:

cd Github\_Learn\_by\_Doing

You now have a working copy of the online repository on your local machine. We want to branch to make changes.

Type:

git checkout -b <newbranchname>

For this lesson, we will call the new branch your name:

IE:

git checkout -b leroy

IF you do another directory, you’ll notice that your new branch also has the same files as the master branch - it’s safe to make changes here thou.

\*git branch\* will tell you the current working branch.

In your file manager, find and open the “name.txt” file - this should be the current working copy on your “name” branch

At the end of the file, add your name.

“I was here Aug 14, 2017 - LeRoy Miller” and save the file.

Go back to the bash prompt and type:

git status

You should see that “name.txt” has changed, go ahead and add it and commit the changes you made.

\*git commit -am ‘I added my name’\* just in case you forgot.

Now we are going to push our branch back to the github repository.

\*It’s going to be similar to what we have already done, this is a new branch and we have to tell the repository about it\*

IF you type: git push now, you’ll get a error message and a way to correct the error.

“Fatal: the current branch leroy2 has no upstream branch.”

“To push the current branch and set the remote as upstream, use

git push --set-upstream origin leroy2”

Type:

git push --set-upstream origin <yourbranchname>

And you should see the hashes, and objects, and the push, it should tell you it created a new branch, and that it finished.

Learned:

How to clone a repository, and push a new branch.

Commands:

git clone <repository url>

git checkout -b <branch>

git push --set-upstream origin <newbranch>

# PART 4: (Using PULL and Create a PULL Request)

DONE: BUT MORE:

\*git pull it is the opposite of push, when working in a cloned repository (or a repository you don’t own) you should issue a git pull before doing more work on it. This will merge any new changes to your local copy.

Now that everyone has pushed the new branch to the online repository, your local copy may be out of date.

First you want to go back to the “master” branch.

Type:

git checkout master

Next to verify:

git branch

You should see a list with your branch, and a \*master (The \* is the working branch)

leroy

leroy2

\*master

IF you type “git status” it will probably say your update to date - which you are on the local repository, you aren’t from the online repository so we type:

git pull

This should pull all the changes just made. And if you type git branch again you should see all of the other branches.

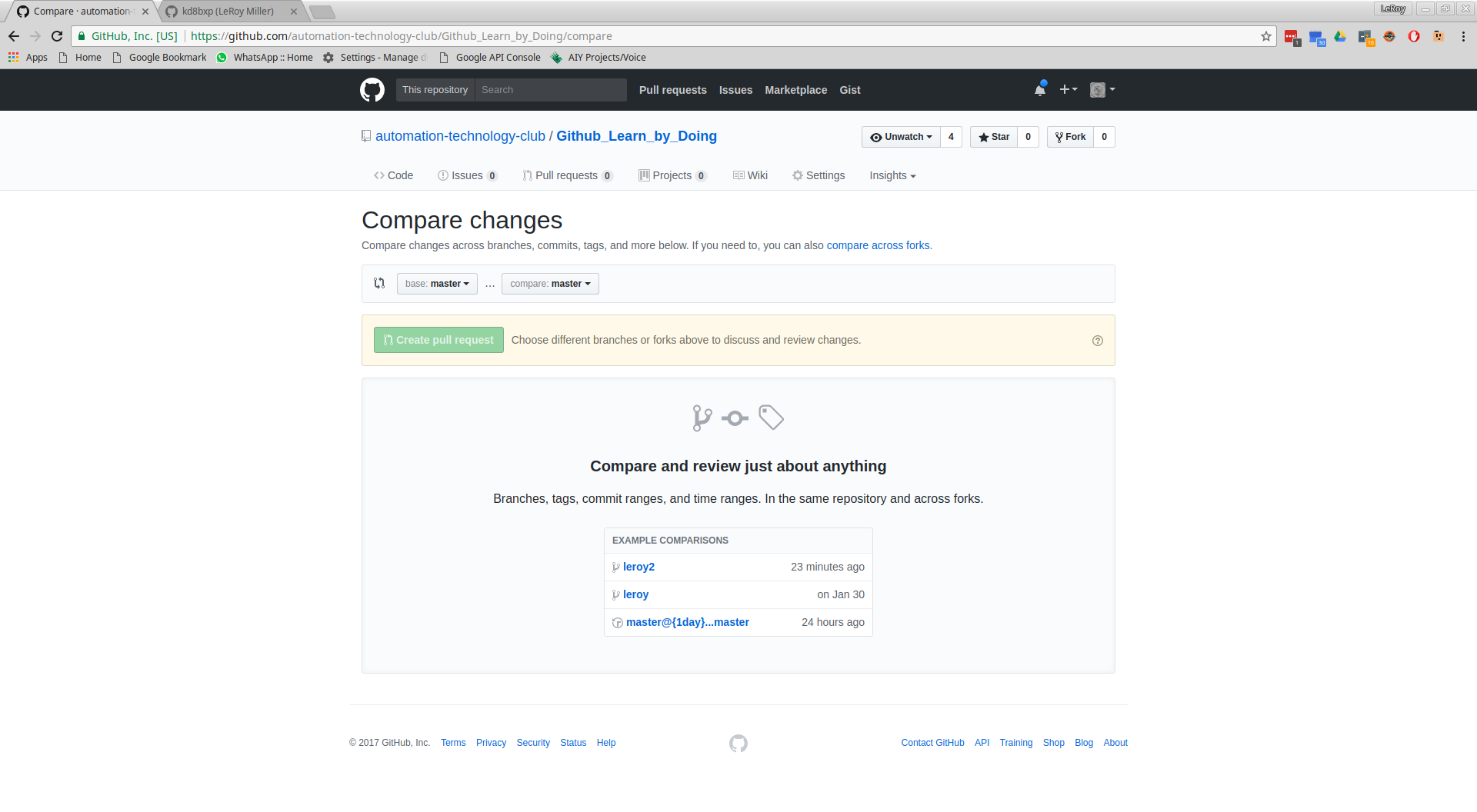
\*git checkout <branchname> will let you switch to a different branch

The last thing I want to show you is how to make a “pull” request. To do this we go back to the website. And you should see “pull requests” just under the name of the repository.

\*A pull request - lets changes from either branches, or other repositorys get pulled into the master branch. It doesn’t make since right now, but hopefully doing it, it will make since.

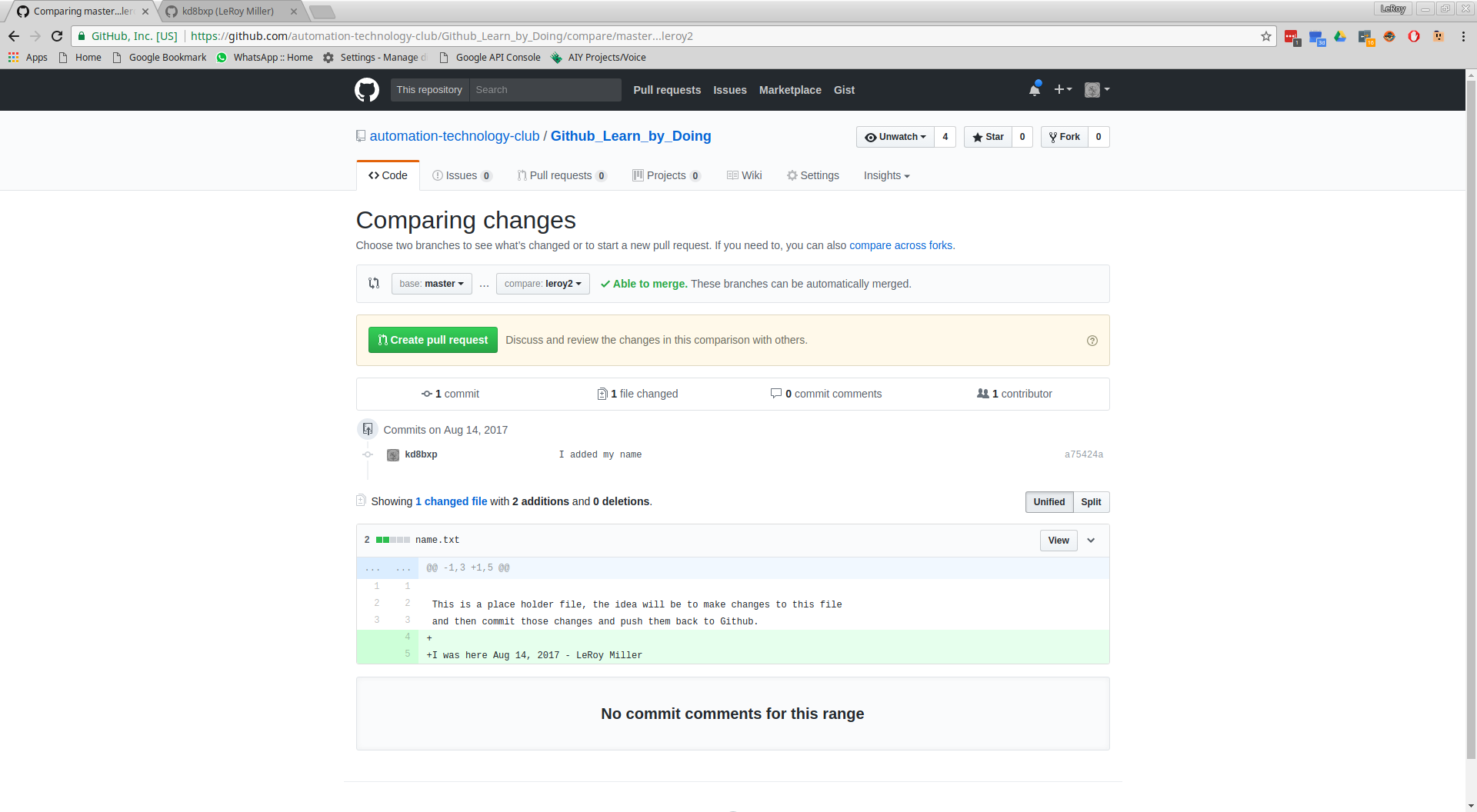
IT lets the admin of the repository know that someone has made a change, and they believe the change is going to be useful to everyone using the repository. Admins make the call on what to include, or reject\*

Click that...and you should see a green tab that says “new pull request” -

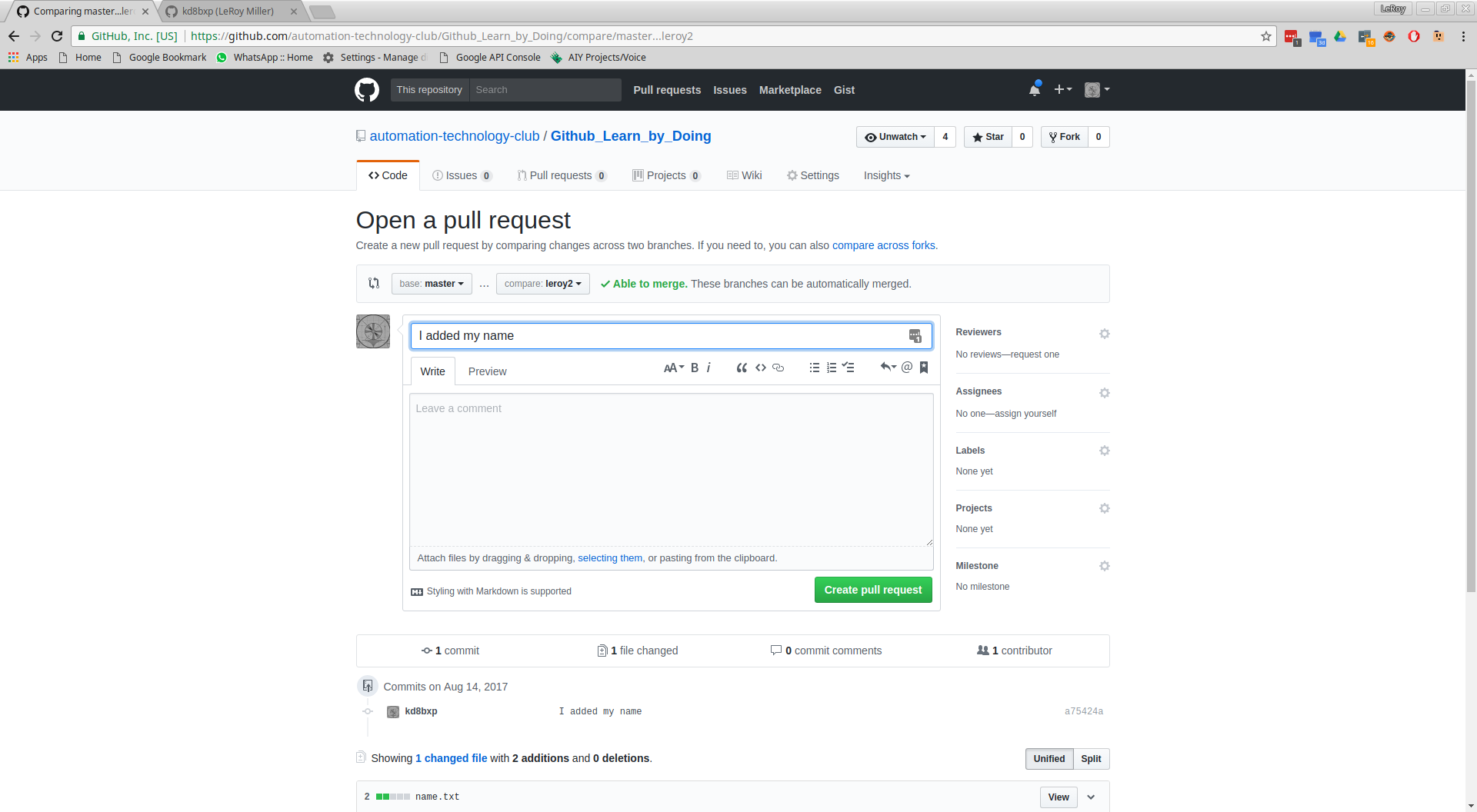


You see where it says “base: master” …. “Compare: master” click the compare, and select your branch.

\*You can see here I selected the “leroy2” branch - and this screen tells me that there will be no problems with merging to the master branch, and what was changed (in green).

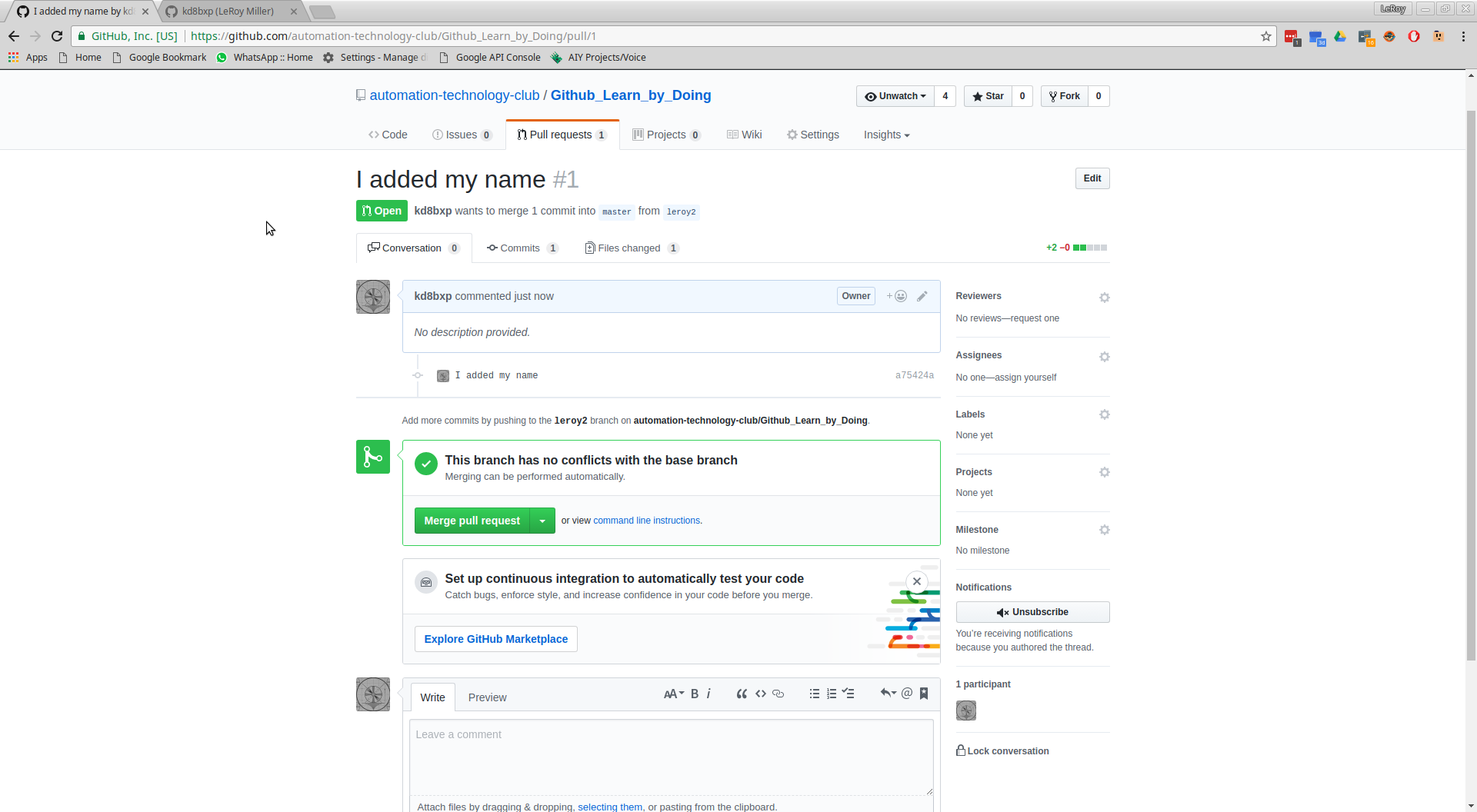


Clicking on the green “create pull request” will bring you to a screen like this:



You can see the commit message is included, and a larger space for a more detailed message is also given. And again one more chance to back out of the request, or not. Click the green “Create pull request” to create it.

You should see something like this:



And at this point, the admins (me) can make a comment on the request, approve the request or reject it.

# GIT commands used in this tutorial:

git init

git status

git add <files>.....

git commit -am ‘message….’

git checkout

git log

git remote add origin <https://github.com/yourswillbedifferentfrommine.git>

git push -u origin master

git clone <repository url>

git checkout -b <branch>

git push --set-upstream origin <newbranch>

git pull

git branch

Git is powerful, and this tutorial just give the very basics of what I believe are needed for Git to be useful. It can do a lot more. But using only 9 of the very basic command you are able to do quite a lot, and using just 6 commands can create and push repositories.

This is not meant to be a complete tutorial, it is meant to get you started.

More Information: Keeping a fork up to date -

So you forked a project, and the original project updated - how do you bring the new updates into your fork? - This is how….

<https://gist.github.com/CristinaSolana/1885435>

Summary:

After you already cloned your fork.

Add remote from original repository in your forked repository:

“Cd into/cloned/fork-repo

Git remote add upstream git://github.com/ORIGINAL-DEV-USERNAME/REPO-YOU-FORKED-FROM.git

Git fetch upstream

“

Git pull upstream master

To update the online repository

Git push