

GITHUB WORKSHOP

INSTALL GIT

- Go to <https://www.git-scm.com>
- Download Git 2.5.0



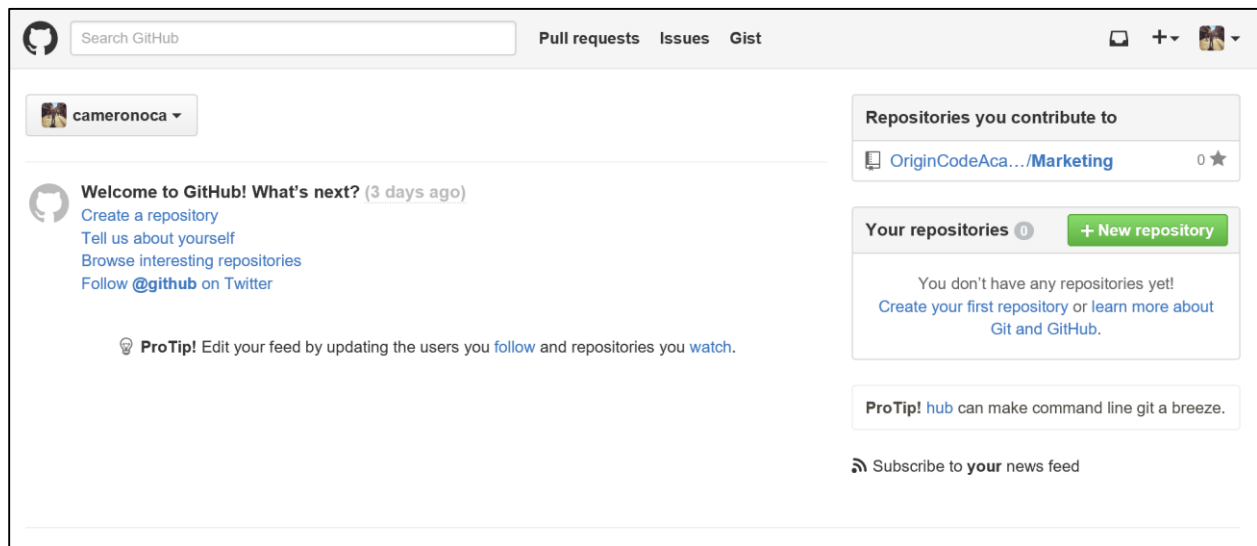
- Click on the downloaded executable to start the installation process.
1. **Welcome Screen** – click Next
 2. **Read the license** – click Next
 3. **Select Destination Folder** – By default this will be installed under C:\Program Files (x86)\Git
 4. **Select Components** – Everything here is fine by default, click Next
 5. **Select Start Menu Folder** – click Next
 6. **Adjusting your PATH environment**
This is very important. Make sure you select “Use Git from the Windows Command Prompt” before clicking Next.
 7. **Configuring line end conventions** – Leave as is (Checkout Windows-style, commit Unix-style endings)
 8. **Installing** – Watch Git install itself.
 9. **Completing the Git Setup Wizard** – Click on finish to exit the setup.
- Open Command Prompt (Windows + R > type “cmd” > hit Enter)
 - Enter **git –version**
 - If you see **git version 1.9.5.msysgit.1**, success! You’ve installed git 😊

- The next thing we need to do is tell Git who you are. In command prompt, enter the following commands:
- **git config --global user.email "enter your email here"**
git config --global user.name "enter your name here"
For example, I would enter
git config --global user.email "cameron@origincodeacademy.com"
git config --global user.name "Cameron Wilby"
- After that, you're done! Let's move on to create a GitHub account and start moving stuff between your local computer and GitHub.

CREATE A GITHUB ACCOUNT

- Navigate to <https://www.github.com>
- Click the green Sign Up button in the top right hand corner
- Follow the instructions on screen to create your account
- Once you've created your account, use the Sign In button to sign in to your account.

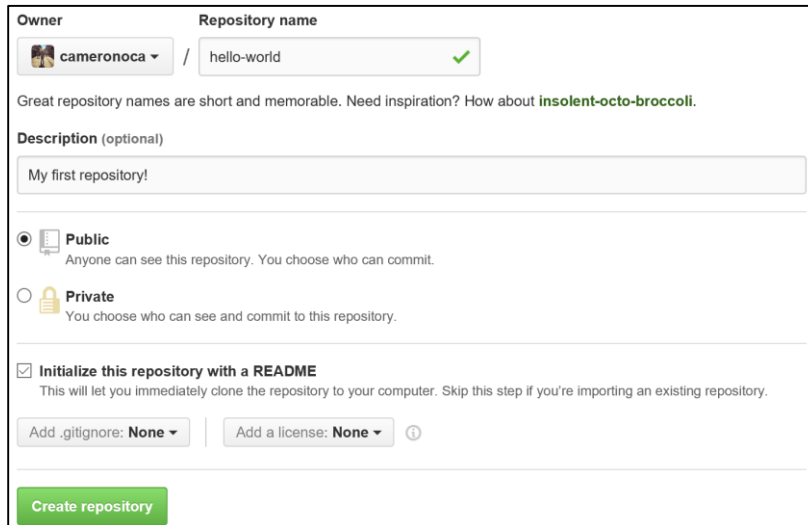
You should be presented with this screen



Success! You've signed up with GitHub. Next, let's create your first repository.

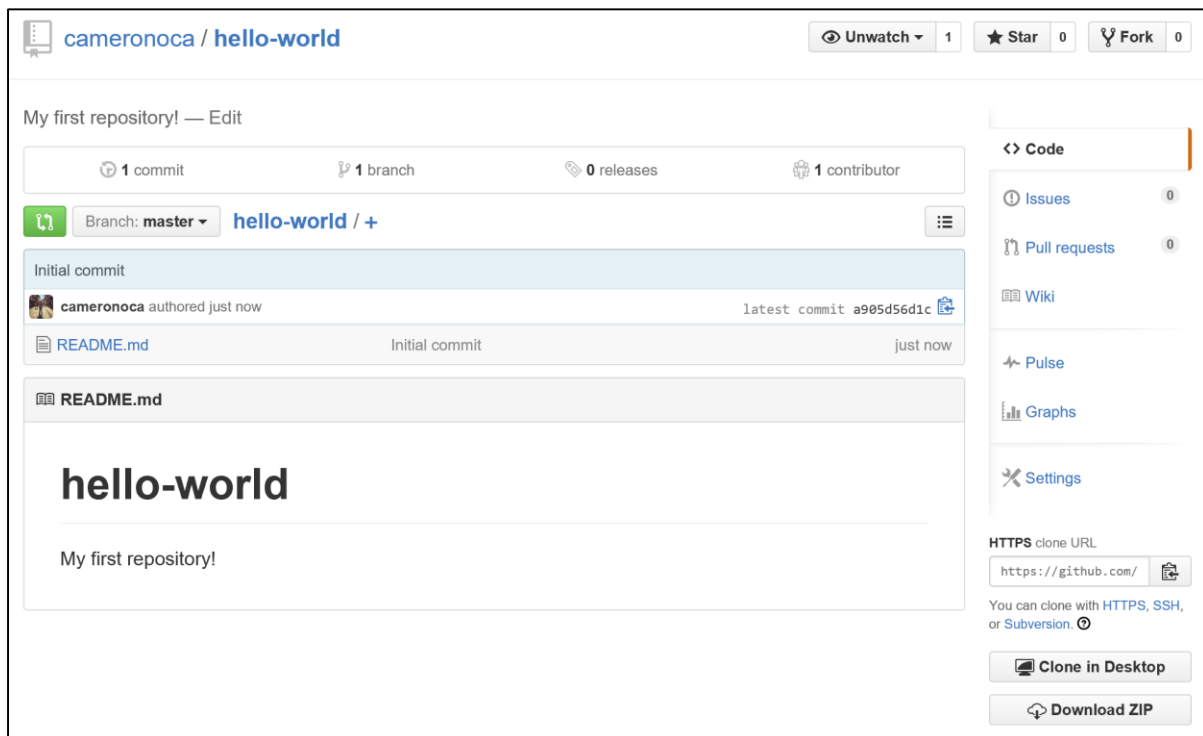
CREATE A REPOSITORY

1. Click the green New Repository button.
2. You will be presented with the following screen. Fill in the fields to make it look this:



The screenshot shows the 'Create a new repository' form on GitHub. At the top, the 'Owner' is 'cameronoca' and the 'Repository name' is 'hello-world', which has a green checkmark. Below this, a hint says 'Great repository names are short and memorable. Need inspiration? How about [insolent-octo-broccoli](#).' The 'Description (optional)' field contains 'My first repository!'. Under 'Visibility', the 'Public' radio button is selected, with the text 'Anyone can see this repository. You choose who can commit.' The 'Private' option is unselected, with the text 'You choose who can see and commit to this repository.' The 'Initialize this repository with a README' checkbox is checked, with the text 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' Below this are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. At the bottom is a green 'Create repository' button.

3. Click the green Create Repository button.
- Once you've created your repository, you're presented with the following screen



The screenshot shows the GitHub repository page for 'cameronoca / hello-world'. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below this is the repository name 'cameronoca / hello-world' and a link to 'Edit'. The main content area shows 'My first repository! — Edit' and a summary of the repository: '1 commit', '1 branch', '0 releases', and '1 contributor'. Below this is a section for the 'Initial commit' by 'cameronoca' just now, with the latest commit hash 'a905d56d1c'. The 'README.md' file is shown, with the text 'hello-world' and 'My first repository!'. On the right side, there is a sidebar with links to 'Code', 'Issues' (0), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. At the bottom of the sidebar, there is a section for 'HTTPS clone URL' with the URL 'https://github.com/' and a 'Clone in Desktop' button, and a 'Download ZIP' button.

CLONE THIS REPOSITORY

1. Let's clone this repository to your local machine. Open up command prompt and enter the following commands

- a. **mkdir C:\dev**

This command uses the **mkdir** command to create a dev directory in your C: drive. You will store all of your projects/repositories in this dev directory.

- b. **cd C:\dev**

This command uses the **cd** command to change the current directory to C:\dev.

- c. **git clone <HTTPS clone URL>**

This command uses the **git** command to clone your GitHub repository to your local hard drive. Replace **<HTTPS clone URL>** with the URL seen on your repository page



- d. **cd hello-world**

This command uses the **cd** command again to move you into the hello-world directory you just created with **git clone**.

- e. **dir**

This command will list the contents of your repository. Success! You've cloned your first repository!

ADD A FILE TO THE REPOSITORY

1. Next, let's create a file, add it to your repository, then commit our change to the repository. In command prompt, enter the following commands.

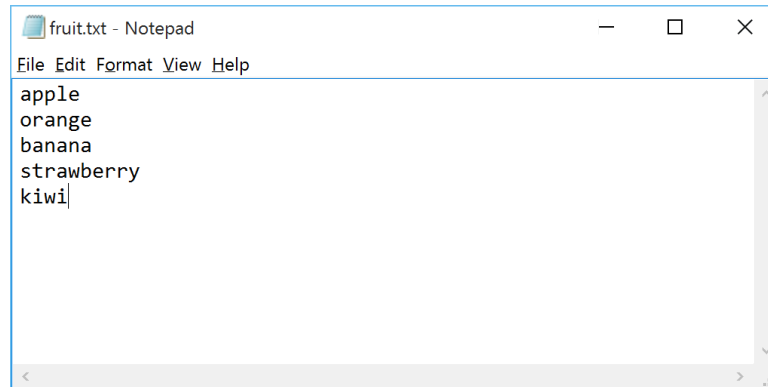
- a. **notepad fruit.txt**

This command tells Windows to open Notepad and create a file called **fruit.txt**.

It will tell you that the file does not exist and ask if you want to create a new file.

Select Yes.

- b. Enter the following into Notepad and click File > Save.



- c. Back in Command Prompt, enter

git status

This command tells git to show you the current status of your repository. It should look like this:

```
C:\dev\hello-world>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)

    fruit.txt

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

- d. Git is telling us in that our fruit.txt file is not currently being tracked in our repository. We need to explicitly tell git to do this by entering **git add fruit.txt** into the command prompt.
- e. Enter **git status** again.
Notice that your file has now been *staged*. For now, all you need to know is that this means that your file is ready to be committed to the repository.
- f. Enter **git commit -m "Added fruit.txt"**

This command tells git to commit your changes with a message of "Added fruit.txt".

g. Enter **git log**

This command uses the **git** command with an argument of **log** to show you the *local* changelog for your repository. You should see something like this.

```
C:\dev\hello-world>git log
commit f62fccd3803abb2361ed5b814adb90b44f187f70
Author: Cameron Wilby <cameron@origincodeacademy.com>
Date:   Sun Aug 9 17:32:12 2015 -0700

    Added fruit.txt

commit a905d56d1c8396c126917ae9cc158d4952ef4b29
Author: Cameron Wilby <cameron@origincodeacademy.com>
Date:   Sun Aug 9 17:06:55 2015 -0700

    Initial commit
```

The log shows you a history of committed changes from new to old. You can see:

- The **commit** id - a very long combination of letters and numbers that uniquely identifies a commit.
- The **Author** – the name of the person who committed these changes (If you are working on a project as part of a team, you would often see other names here)
- The **Date** – The date and time that the commit was made to the repository
- The **Commit Message** – The message that you gave when you entered **git commit -m "Added fruit.txt"**.

You'll notice that if you go look at your repository GitHub right now, your fruit.txt is nowhere to be found. This is because although you've added the file and committed the change, you still need to enter the command that pushes your changes to GitHub.

PUSH YOUR CHANGES TO GITHUB

1. Next, we're going to push the changes we committed up to GitHub, so that our changes are safe and other people can work on them. Open up command prompt and enter the following commands:
 - a. **git push -u origin master**
This tells git to push the commits in your local repository to GitHub.
(Note: The origin argument has nothing to do with Origin Code Academy. It is simply a shorthand for the URL of your repository at GitHub, and it is saved in the repository in which you are working in.)
 - b. You will be prompted to enter your username and password.
 - c. Git will output some messages to the command prompt, once that's done – success! You've pushed your changes to GitHub!
2. To verify that this is the case, head back to GitHub, go to your hello-world repository, and you'll see that your repository now contains the fruit.txt file we entered earlier.
At this point, if you were working with a team of developers, they could now download this fruit.txt file, make changes, commit their changes, and push back to GitHub.

Alright! What you've learned here are the basics of keeping track of files in GitHub. Next, let's fork someone else's repository so we can make changes to it.

FORK A REPOSITORY

So to recap, we've installed Git, created a GitHub account, created a repository, cloned that repository to our hard drive, made changes locally, and pushed our first change back into the repository.

We are now going to fork your instructors repository, clone it to our hard drive, make changes locally, push those changes back to GitHub, and then issue a pull request so that your instructor can pull your changes into his repo.

Let's start!

1. Make sure you are logged in to GitHub
2. Go to <https://github.com/cameronoca/instructor-hello-world>
3. Click on the "Fork" button in the top right of the repository.
4. You will see a loading screen presented by GitHub. GitHub is automatically forking your instructor's repository and copying it to your own account so you can clone it down locally.
5. Open up Command Prompt and enter the following commands
 - a. **cd C:\dev**
This will set your current directory back to our local repository storage.
 - b. **git clone <HTTPS clone URL>**
Same as last time, this command uses the **git** command to clone your forked repository to your local hard drive. Replace **<HTTPS clone URL>** with the URL seen on your forked repository page. Note that you cannot clone your instructor's repository, because you have not been given access to that repository. (This is why forking is a thing.)
 - c. **cd instructor-hello-world**
This will set your current directory to your local copy of the forked repository.
6. Cool! We've now cloned somebody else's repository! We can now make the changes we want. Let's do that now. Open up Command Prompt again and enter the following commands.
 - a. **notepad fruit.txt**
This will open the fruit.txt in notepad. Think of a fruit that isn't already in the list and add your name in brackets. For example:
tomato (John Smith)
Save your changes (File > Save), and close notepad.
 - b. **git add .**
Even though we're updating this file, we still need to add our changes to the staging area. We could use **git add fruit.txt**, but for convenience you can use **git add .** to add all changed files to the staging area, instead of specifying them individually.


- c. **git commit -m "Added my fruit!"**

This commits your contribution to the repository, and you are now ready to push back to GitHub.

- d. **git push -u origin master**

This will push your commit to GitHub.

CREATE A PULL REQUEST

1. **Now we're getting some work done around here!** The final step is to create a pull request on GitHub. If your instructor likes what he sees, he will accept your pull request and you will have successfully contributed to the Origin Fruit List Project™!
 - a. Go to GitHub and find your fork of **instructor-hello-world**.
The URL should be `https://github.com/<your username>/instructor-hello-world`
 - b. Click this green button.

 - c. You will now be presented with the fruits of your labor. What you see here are the changes that you have made to the original repository. The final step is to create a pull request, which can be done by clicking the big green Create Pull Request button.
 - d. Leave a comment, and then click Create pull request once again.
 - e. Your instructor will now be given the opportunity to accept your pull request.

You've completed this
workshop!

Let your instructor know that you have finished.