Introduction to Git and GitHub

Natasha Watkins

January 23, 2018

Version control

git and GitHub are tools used for **version control**

Version control:

- tracks changes made to your work
- prevents you from losing work (for example, accidentally deleting a file)
- allows you to go back to an old version of you work (undo changes)

What is Git and GitHub?

Git is a widely used, open source **version control system**

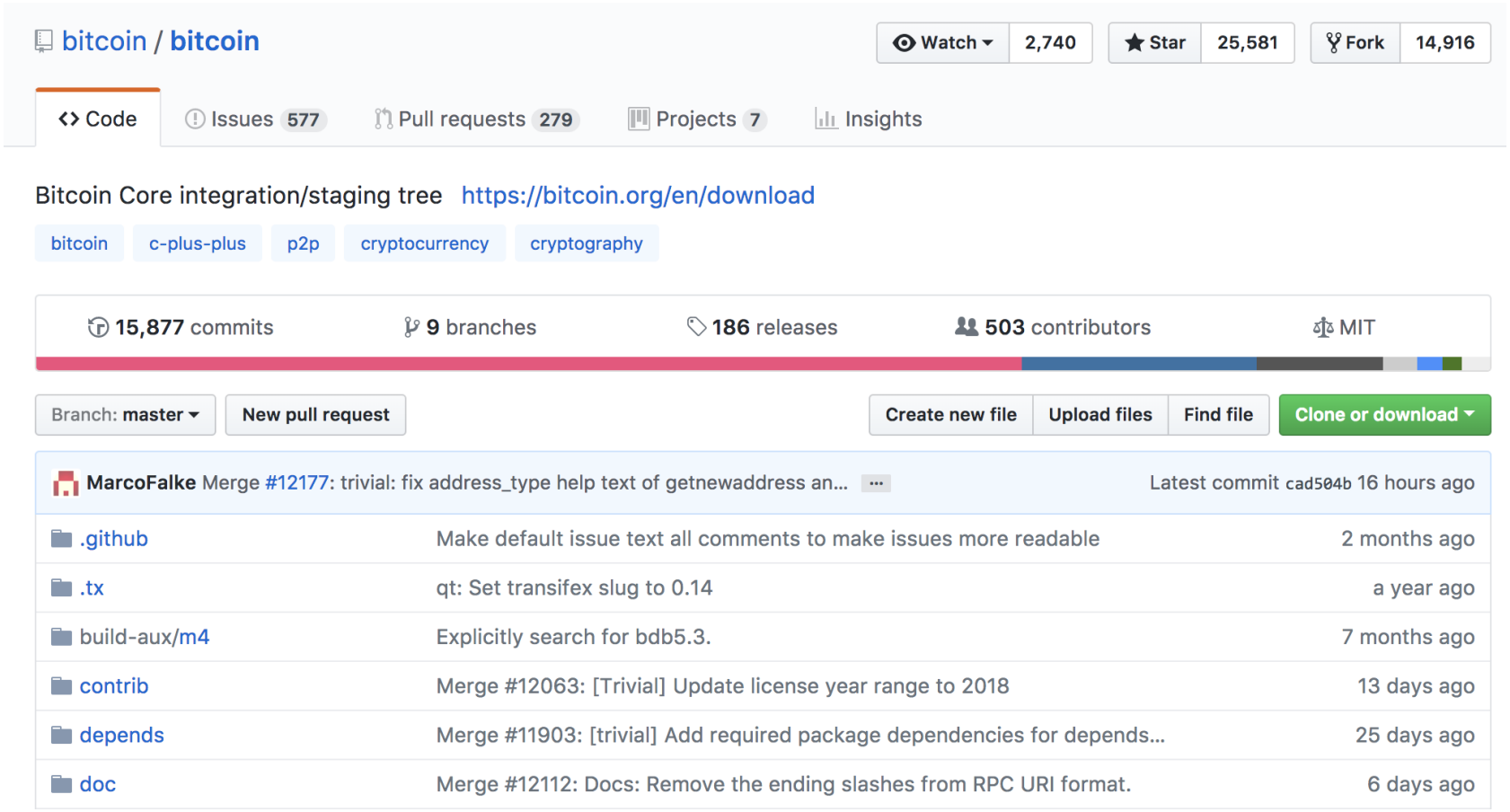
- Does not require an internet connection to use
- Usually used with the **command prompt** - there are also desktop applications (such as GitHub Desktop)

GitHub is a (mostly) free hosting service for your Git repositories

- Makes it simple to collaborate and share code/documents on the internet
- Acts as an online backup of your project

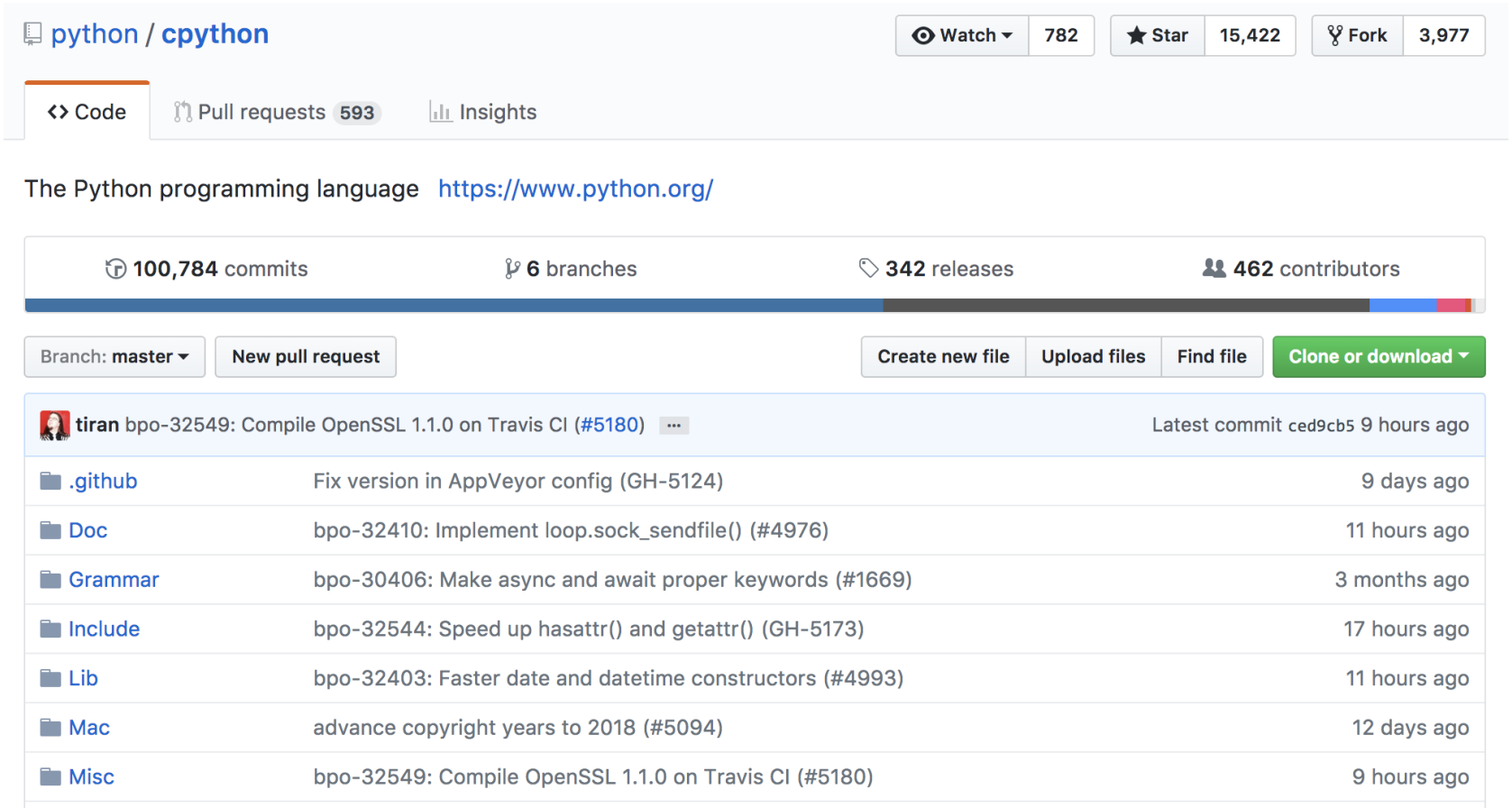
Projects using GitHub

`https://github.com/bitcoin/bitcoin`



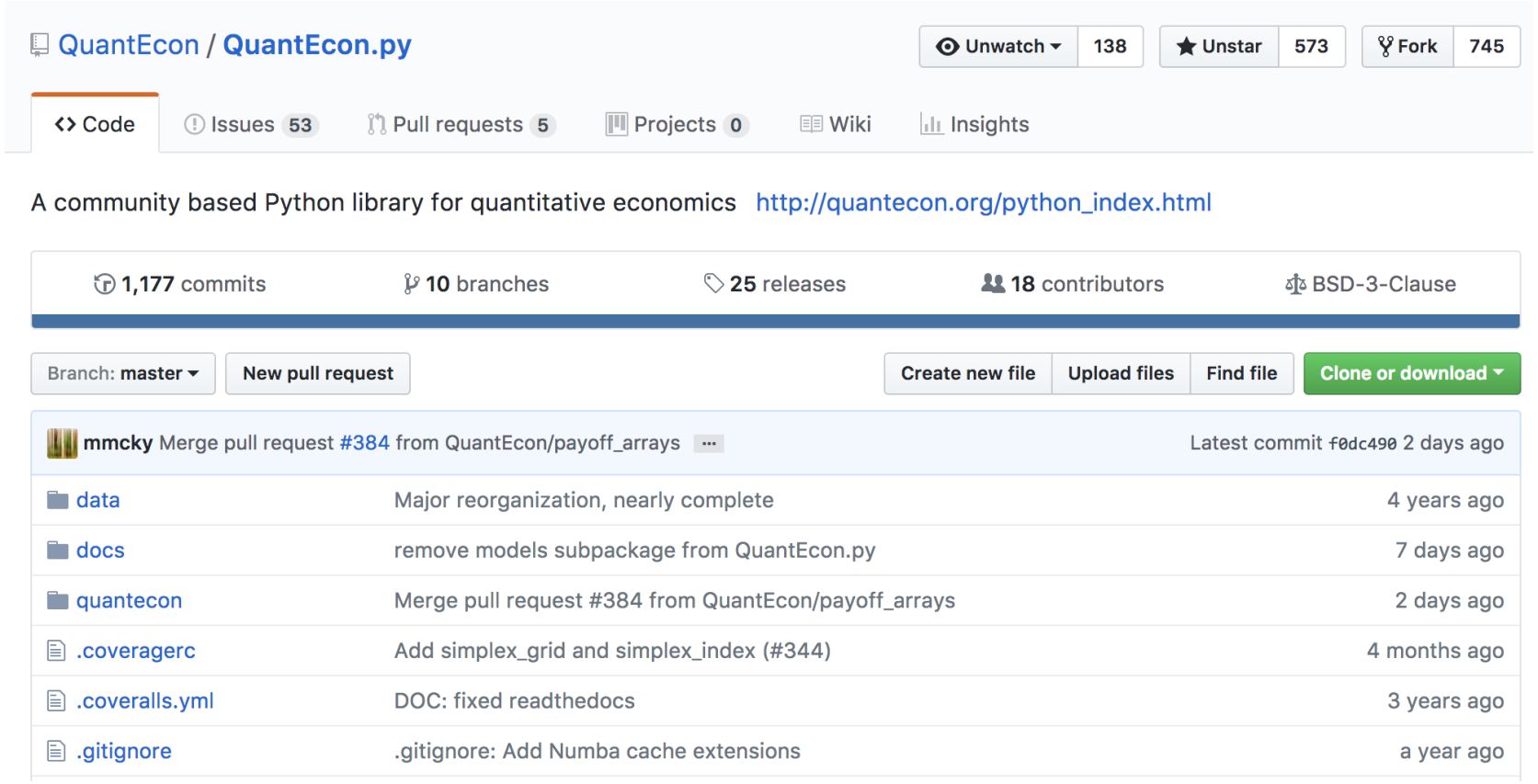
Projects using GitHub

`https://github.com/python/cpython`



Projects using GitHub

`https://github.com/QuantEcon/QuantEcon.py`

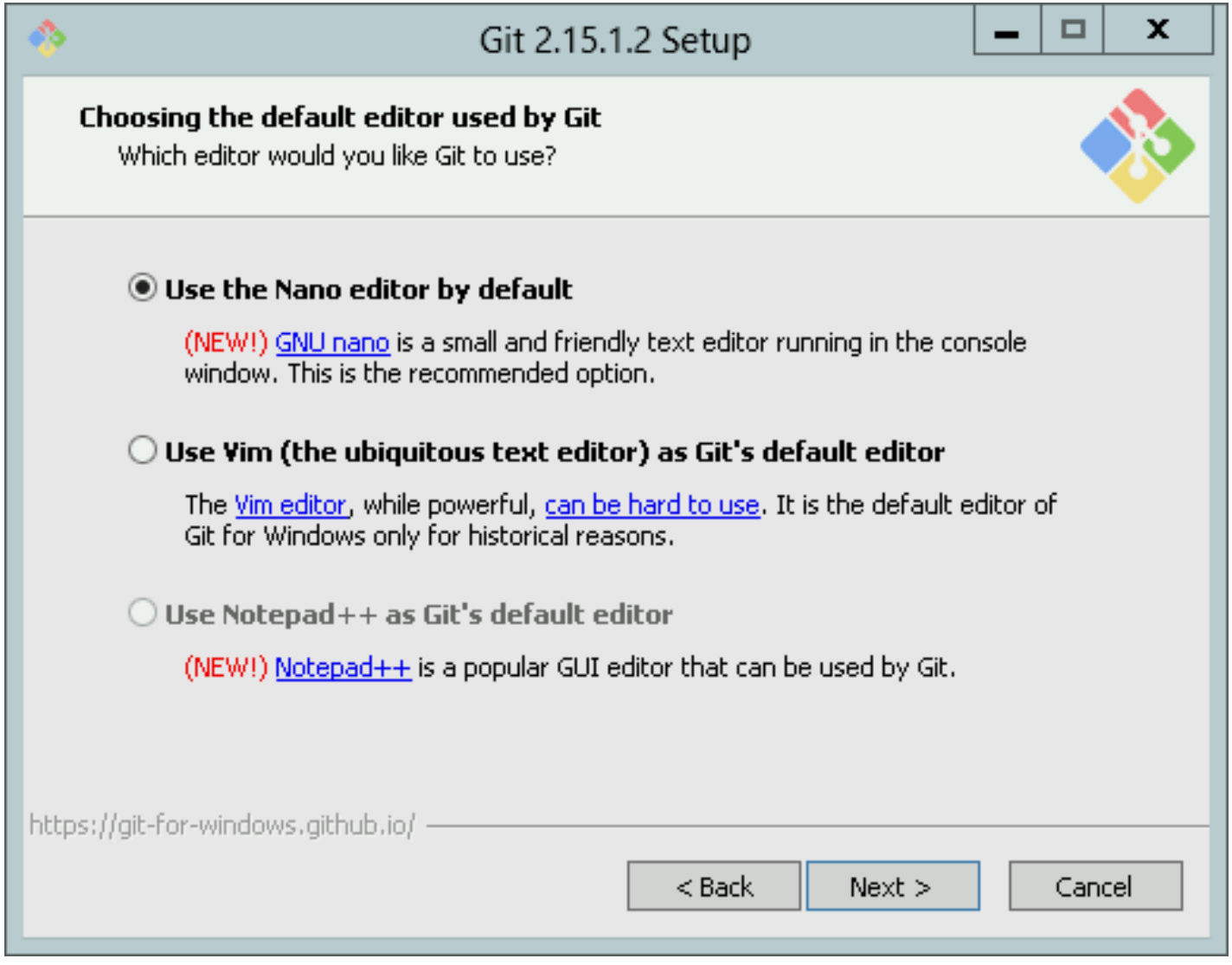


What will we use Git for?

- Git will be used to access course materials and submit assignments during the Shenzhen Winter Camp
- Learning Git and GitHub will also be very useful for your own projects, such as a thesis or group assignment

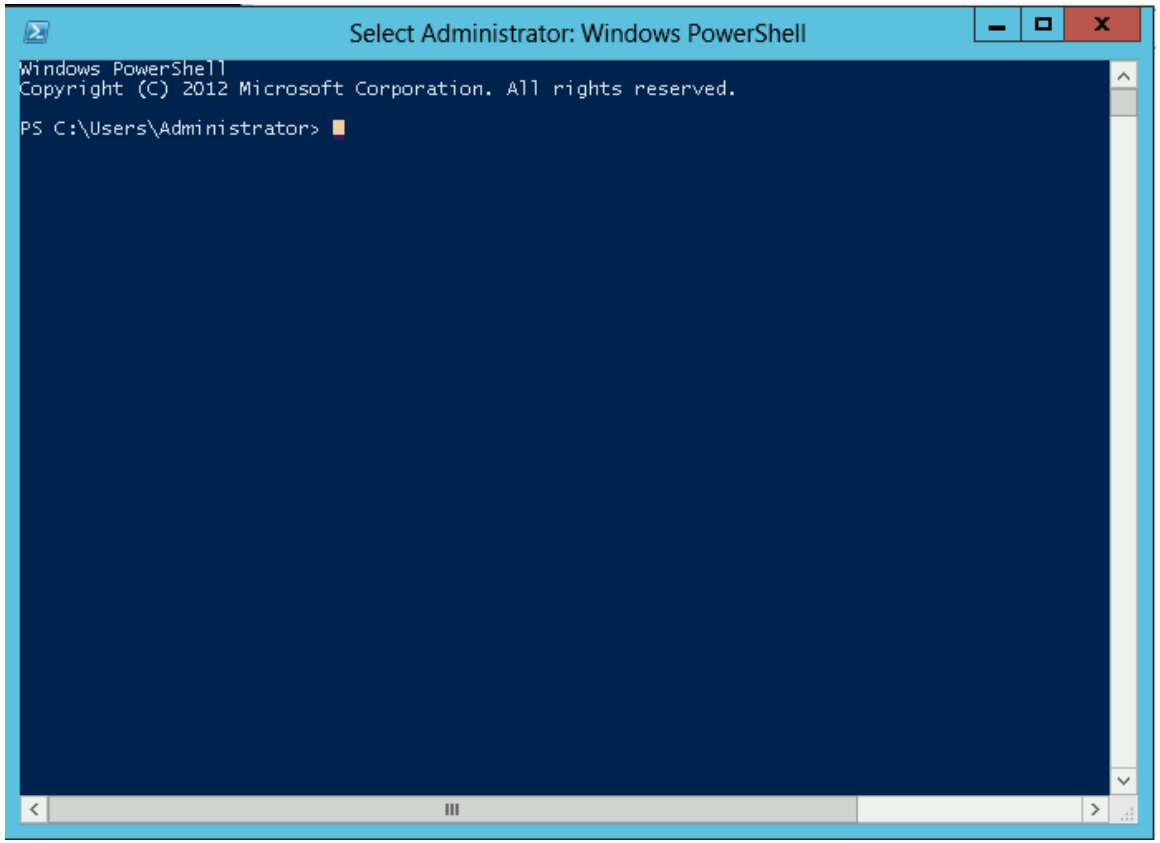
Let's install Git for Windows

1. Go to `https://git-scm.com/download/win`
2. The download should start automatically
3. Open the `.exe` file and start setup
4. Follow the setup instructions and leave the default selections
5. **Recommended:** Change the text editor to **nano** if you are unfamiliar with **vim**


Git 2.15.1.2 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use the Nano editor by default
(NEW!) GNU nano is a small and friendly text editor running in the console window. This is the recommended option.
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. It is the default editor of Git for Windows only for historical reasons.
Use Notepad++ as Git's default editor
(NEW!) Notepad++ is a popular GUI editor that can be used by Git.
https://git-for-windows.github.io/
< Back
Next >
Cancel

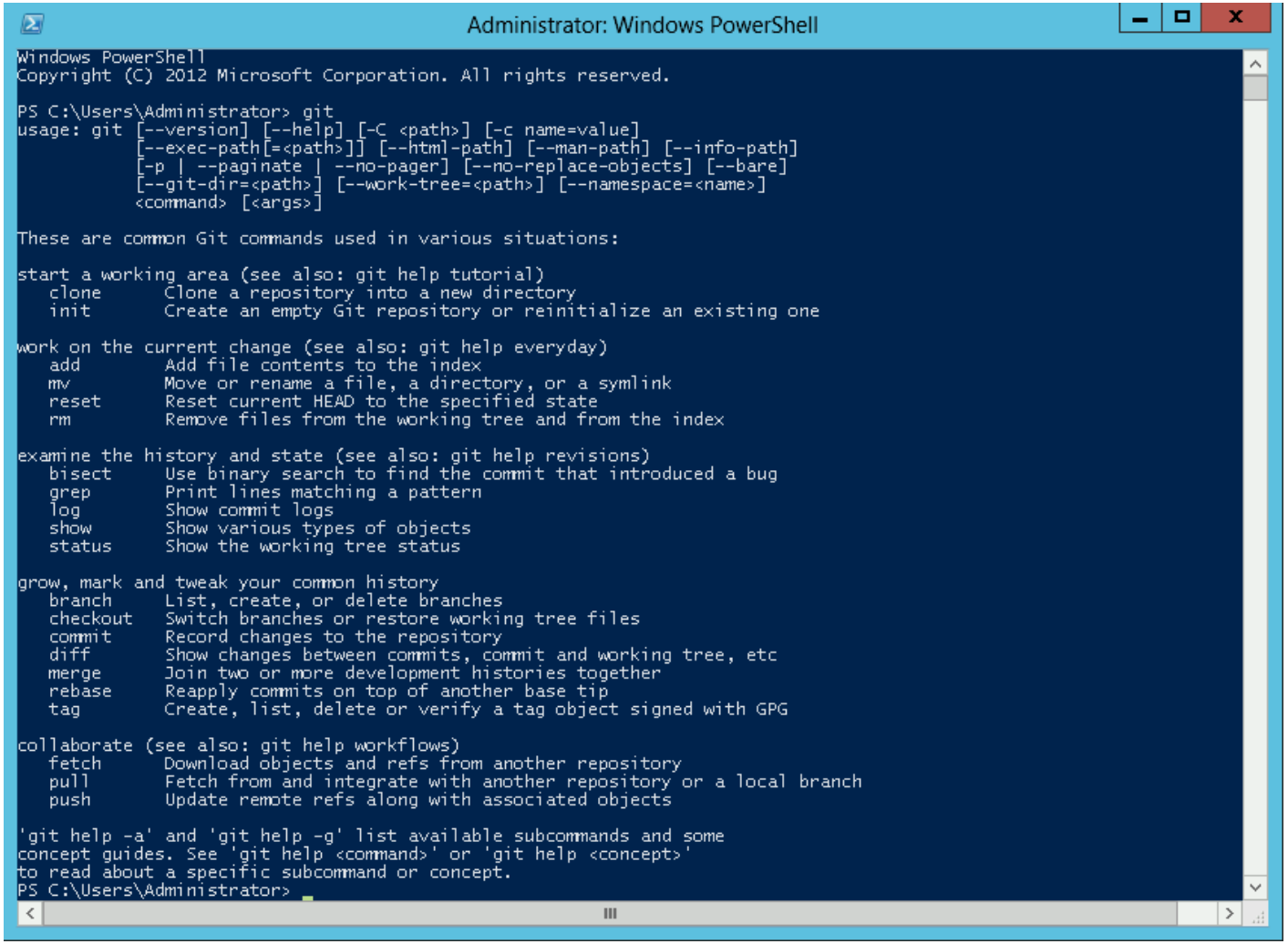
Setting up Git

Now that we have Git installed, open up **Windows PowerShell**



Setting up Git

Check that Git is working by typing `git` and hitting enter



Setting up Git

We need to configure Git to recognize us as the user

In the **prompt**, type:

- `git config --global user.name "Your Name"`
- `git config --global user.email "email@web.com"`

Note: Please make sure you have access to this email!

Creating a GitHub account

We will also need to set up an account on GitHub

1. Open `https://github.com/`
2. Sign up for GitHub using the same email you entered into the prompt

Using a Git repository

A **repository** is like a folder that contains your project's files, as well as the history of changes to the files

Instead of creating a new repository, we will be downloading (or **cloning**) one from QuantEcon

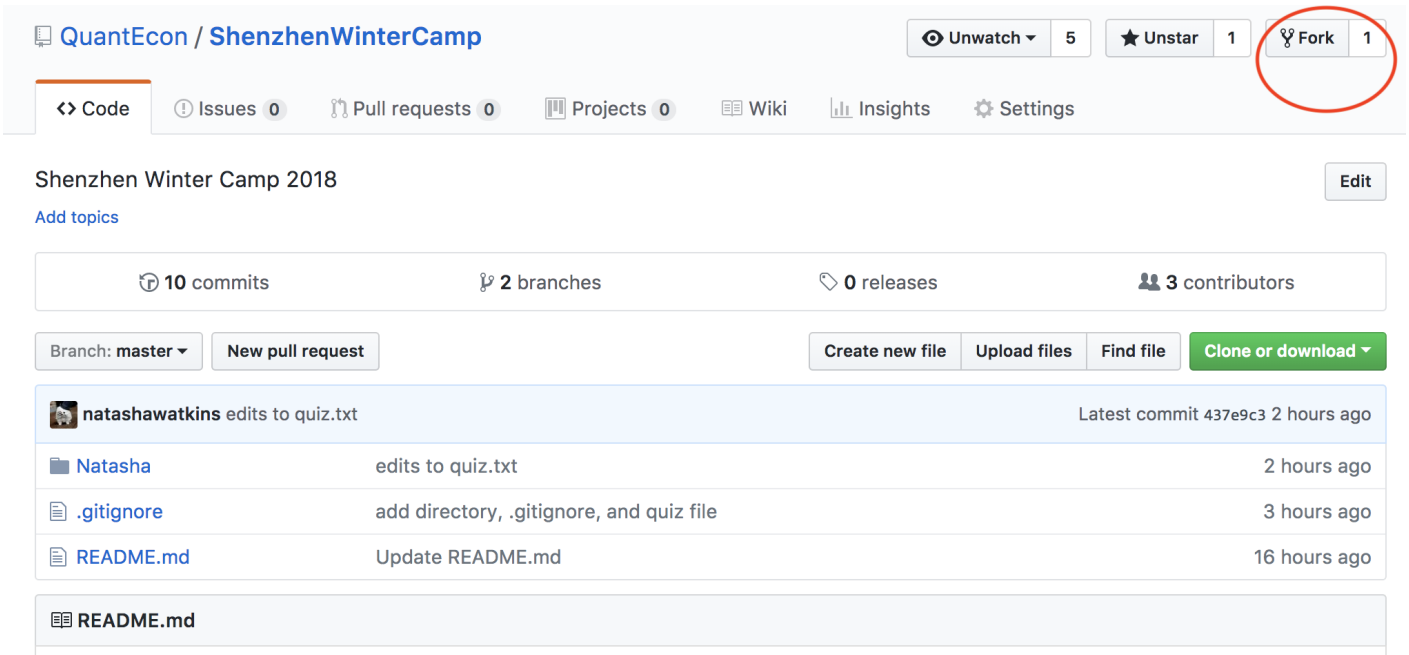
The repository is located at

`https://github.com/QuantEcon/ShenzhenWinterCamp`

Forking a repository

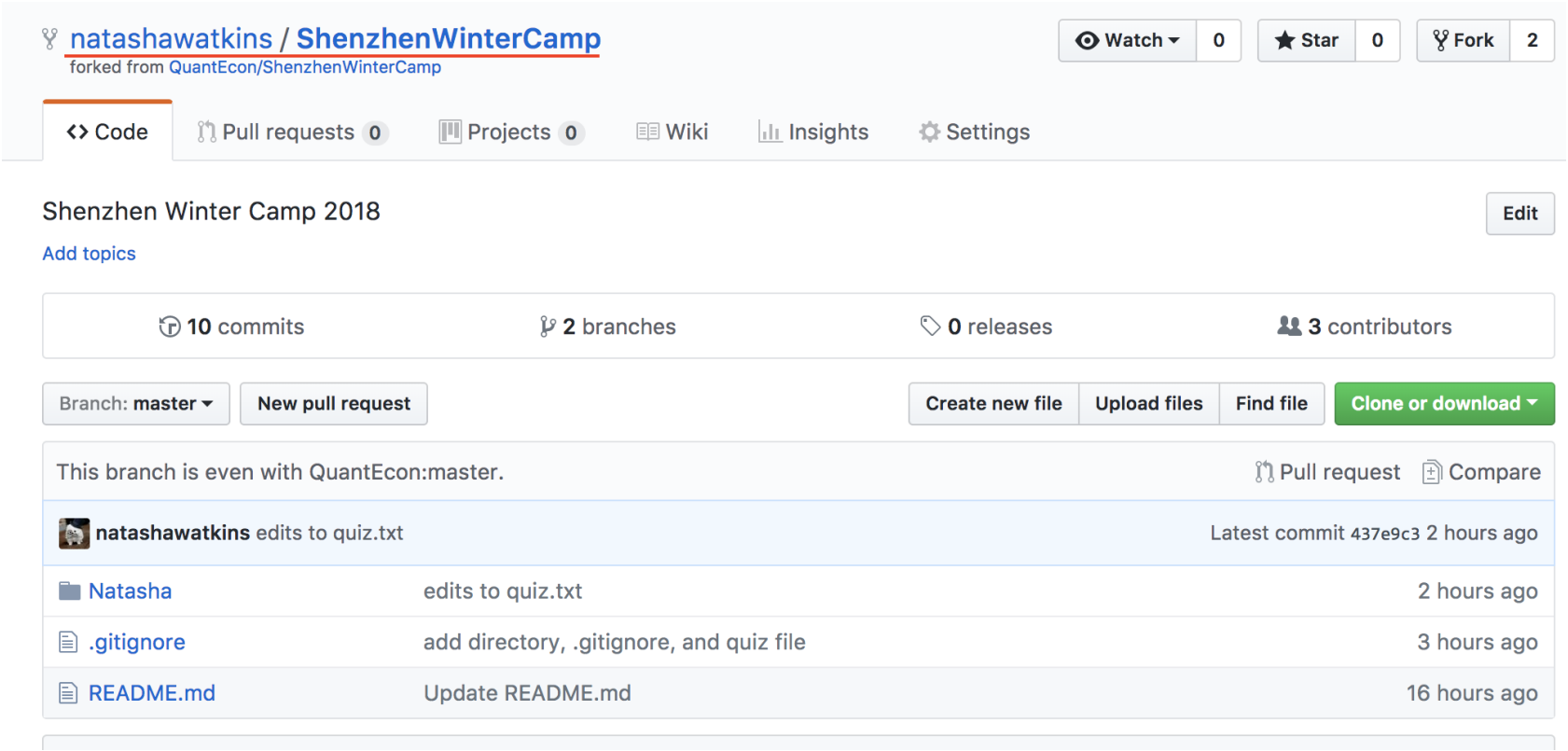
You will first need to **fork** the `ShenzhenWinterCamp` repository

- **Forking** means to copy a repository into your own GitHub account
- Go to the `ShenzhenWinterCamp` repository and click the fork button on the upper right



Forking a repository

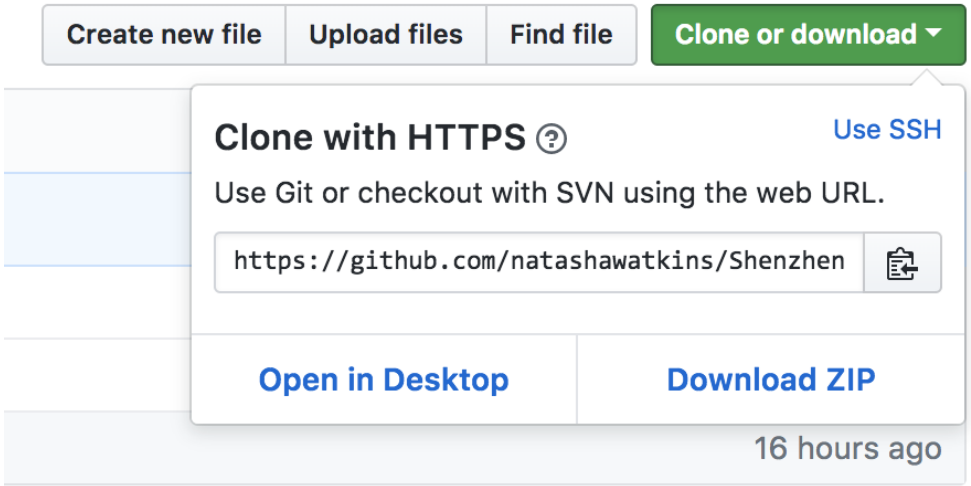
You should now have a repository called `ShenzhenWinterCamp` in your **own** GitHub account



Cloning a Git repository

Next we need to copy (or **clone**) the repository to your local computer

1. On your repository's GitHub page, click the **Clone or download** button and copy the url



2. Navigate to somewhere on your computer (maybe the Desktop) where you would like to save the folder

```
cd ~/Desktop
```

Using a Git repository

3. In the **prompt**, type

   ```
   git clone
   ```

   and paste the url (to paste, right click in the PowerShell window)
4. You should now see a folder called `ShenzhenWinterCamp` on your Desktop

Using a Git repository

5. Navigate to the repository using the **prompt**

```
cd ~/Desktop/ShenzhenWinterCamp
```

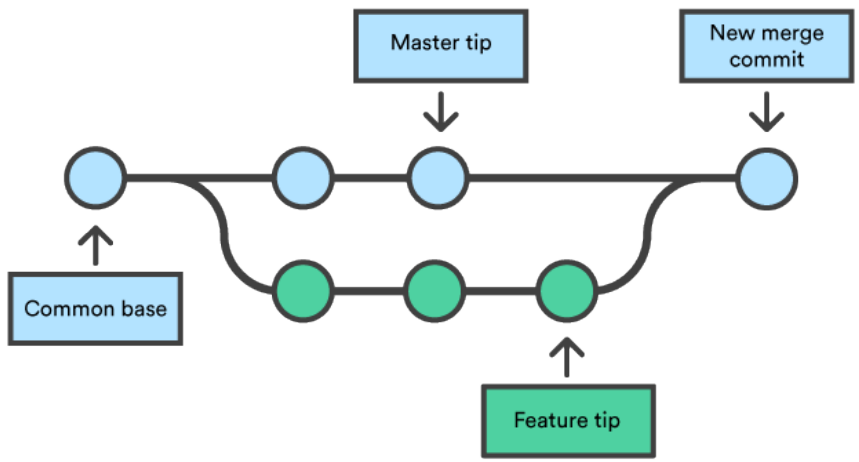
Helpful prompt commands:

- `cd` - change directory
- `cd ..` - move back one level in directory
- `pwd` - print working directory
- `ls` - list files in working directory

Creating a branch

Before we start editing files in the repository, I want you to set up a **branch** in the repository

- A **branch** allows you to make changes to your files without having to worry about the main version
- Used when adding new features to your project
- When you are finished, changes in the branch are **merged** into the main version[1]



[1] Image source: `https://www.atlassian.com/git/tutorials/using-branches/git-merge`

Creating a branch

Create a branch called `your-name` by typing the following command in the prompt

```
git checkout -b "YOUR-NAME"
```

- `-b` is used to create a new branch

Exercise: Quiz!

...and adding/editing files in a Git repository

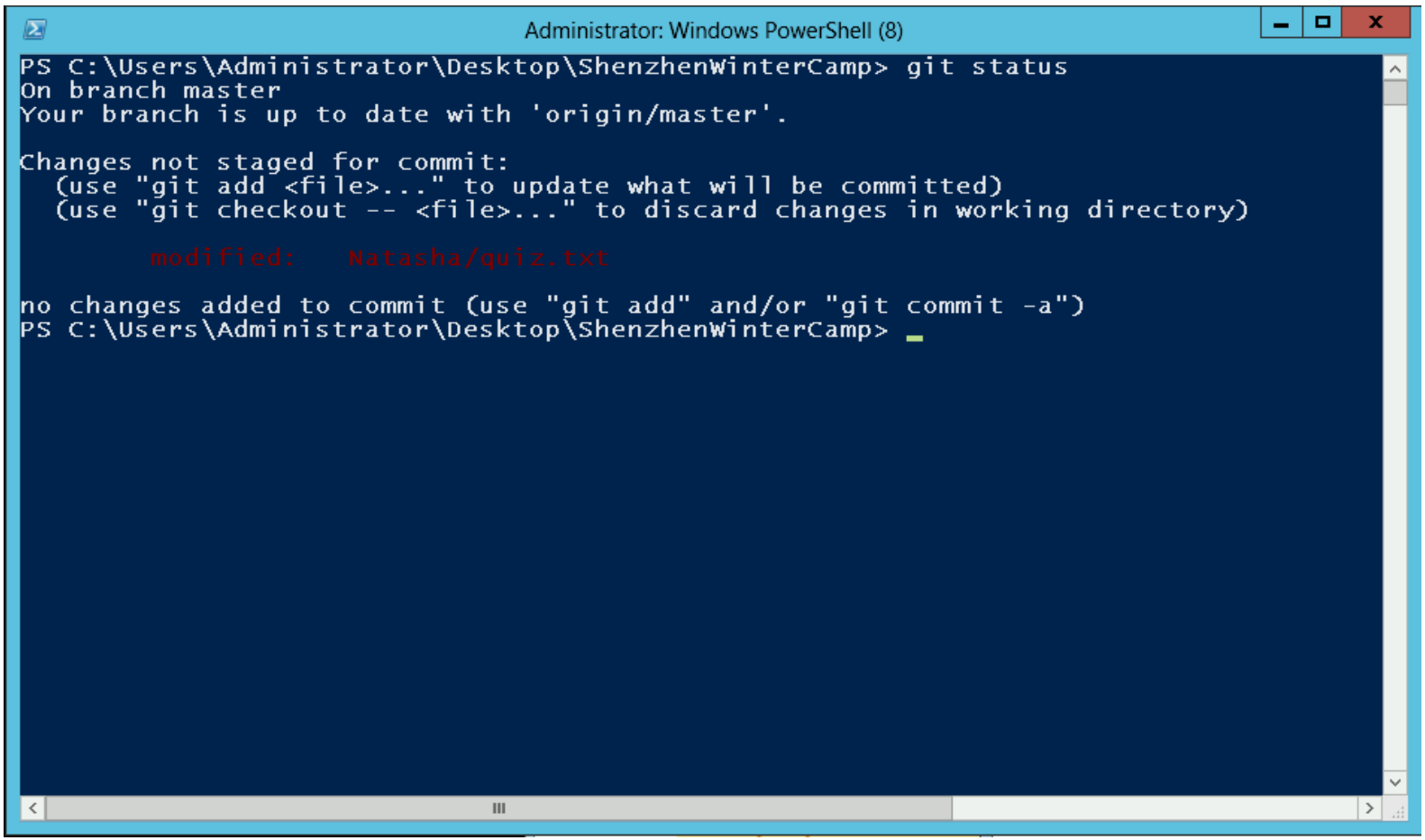
Task: Open `Natasha/quiz.txt` and complete the quiz

- You may use the internet to help you answer the questions
- When you are finished, save the file

Using a Git repository

Git is aware that you have modified a file in the repository

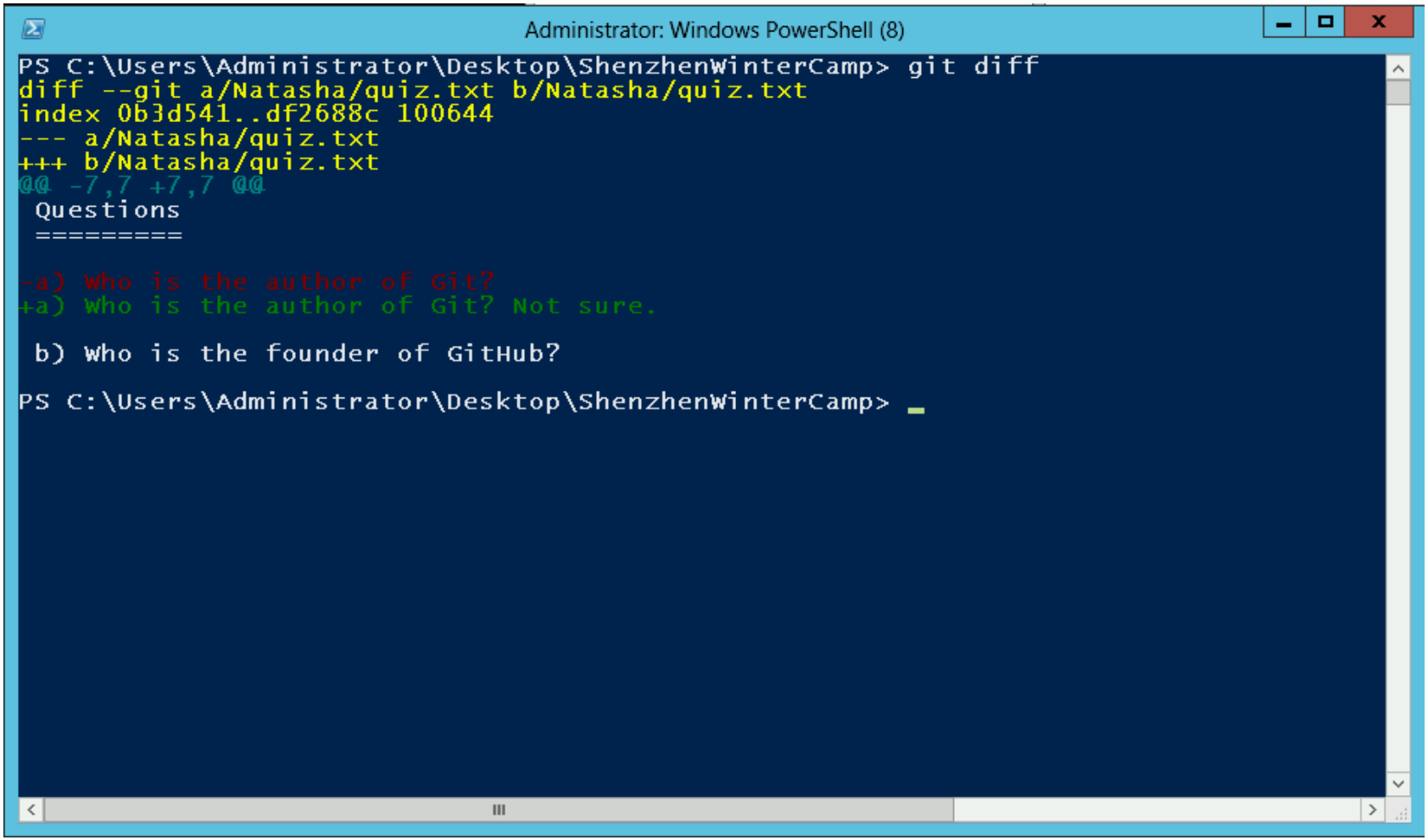
Try typing `git status`



Using a Git repository

You can view the changes you made to `quiz.txt` in the prompt

Try typing `git diff`



Staging changes in the repository

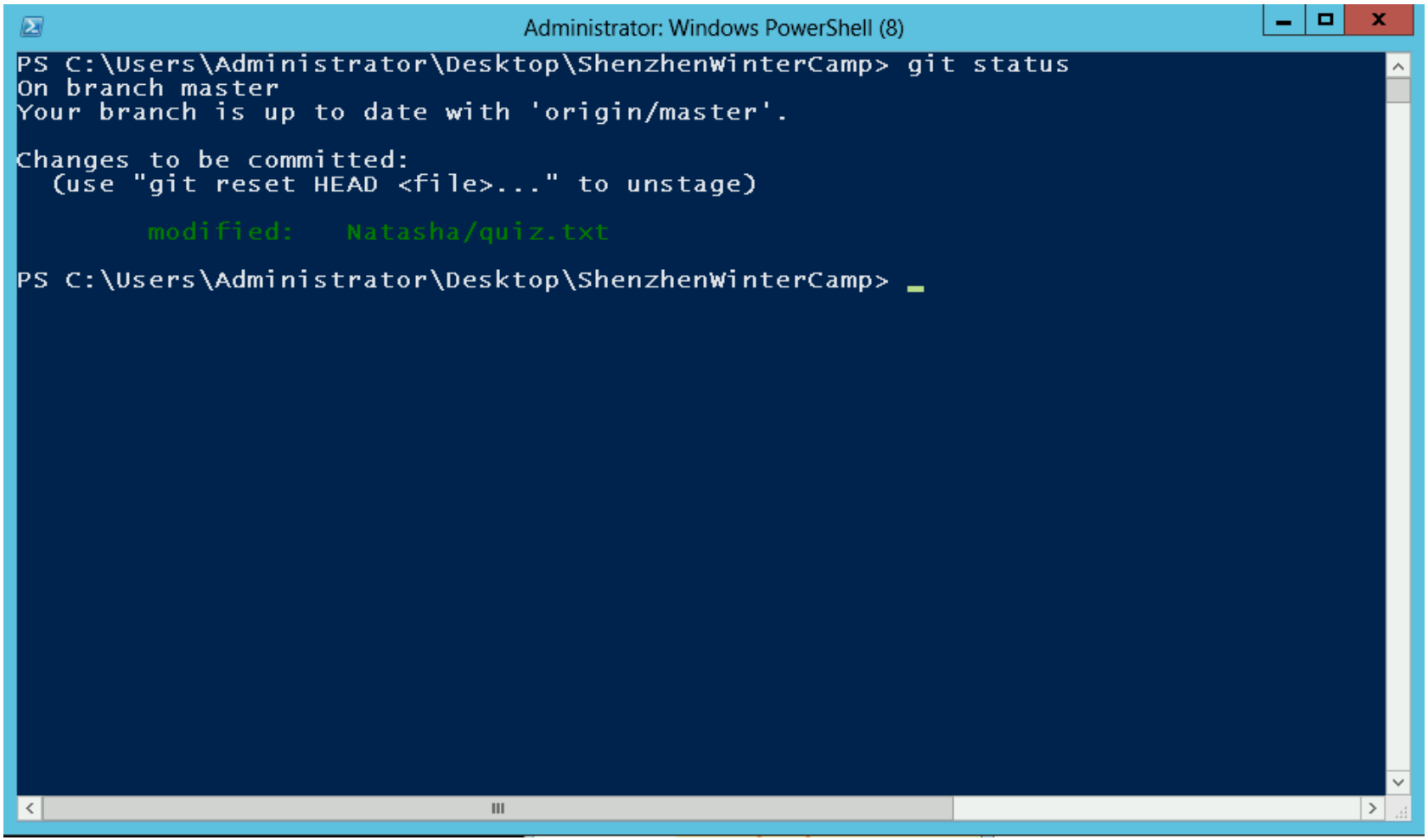
- Once you have finished an edit, you want to update Git's latest "snapshot" of your repository
- First you should move changes into the **staging area**, where edits are organized before they are **committed**
- Type

  ```
  git add Natasha/quiz.txt
  ```

  to add the file to the staging area

Staging changes in the repository

- Type `git status` to check you have added the file you want to the staging area



Committing changes to your repository

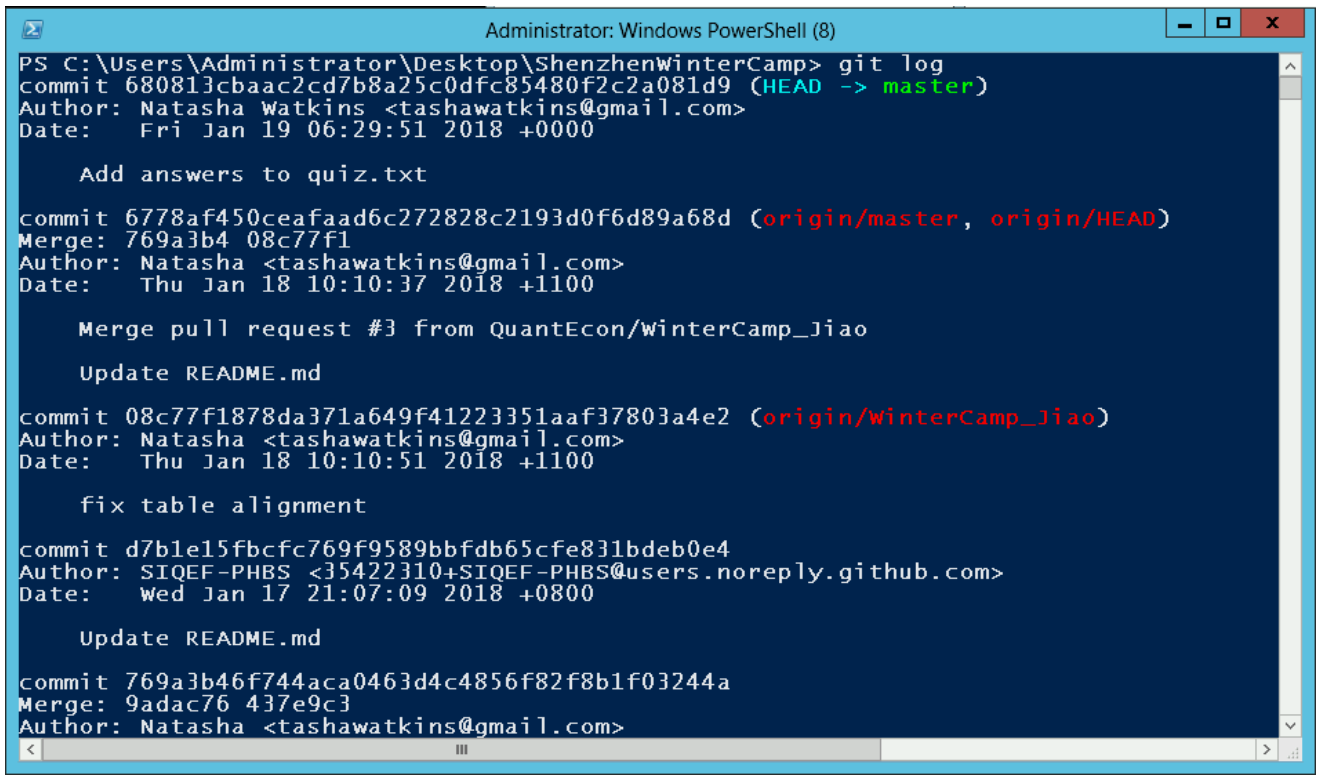
- If you are happy with the changes that are staged, you should now **commit** them to your repository
- **Commits** are permanent snapshots of your repository
- A descriptive message of the changes made should accompany a commit

```
git commit -m "Add answers to quiz.txt"
```

Viewing the commit history

- Commit messages are useful when looking through the history of commits to the repository
- Type the following command to view the log

```
git log
```



Pushing changes to GitHub

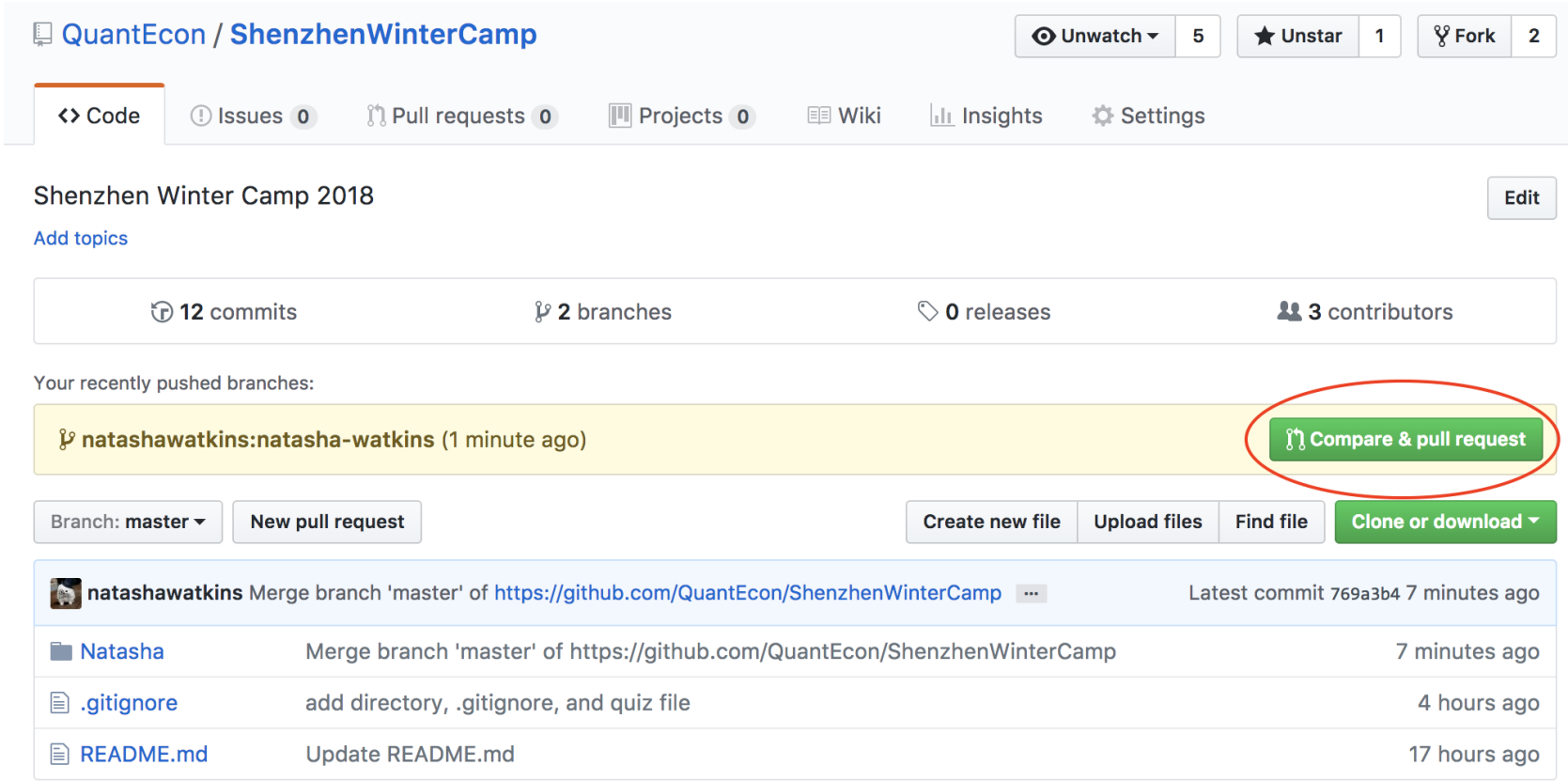
Now we're going to correct each other's quiz answers!

- You will make a **pull request** on GitHub with your quiz answers committed to your branch
- A pull request asks the maintainer of the repository to **merge** the changes into the main version (`master`)
- First we need to **push** (upload) the changes to GitHub

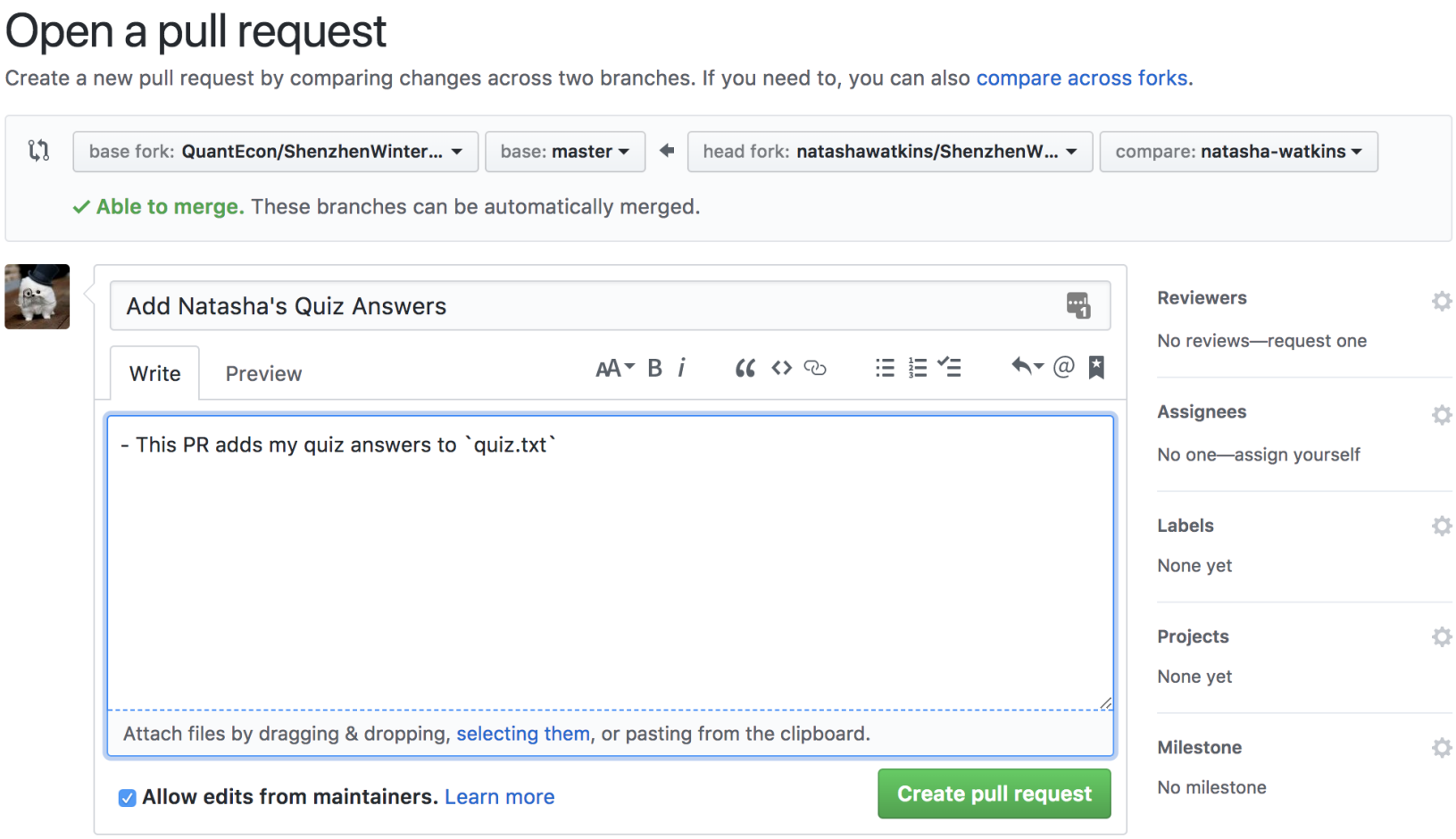
```
git push origin <branch-name>
```

Making a pull request (PR)

- Navigate to the **original** repository: `https://github.com/QuantEcon/ShenzhenWinterCamp`
- Click **Compare & pull request**

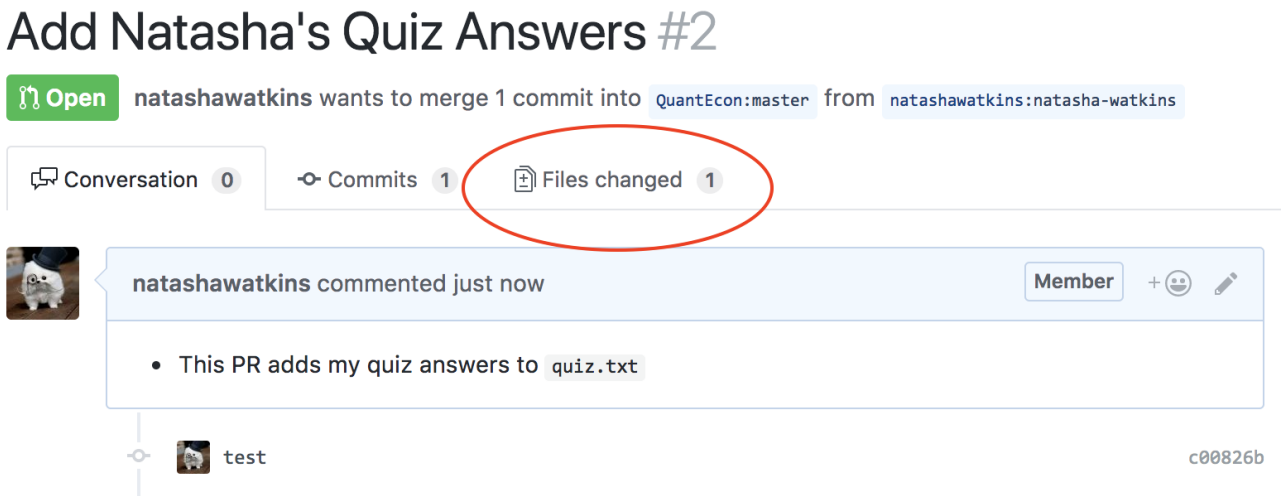


Making a pull request (PR)



Exercise: Make a PR and comment on other PRs

1. Make a pull request to the original repository with your quiz answers
2. Give a score to the quiz of the person sitting next to you - go to their PR and view changes



3. Add the score in a comment on the pull request

More resources

- Version Control with Git (`https://swcarpentry.github.io/git-novice/`)
- Git cheatsheet from Atlassian (`https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet`)