GETTING STARTED

with

GitHub

WHAT IS GIT HUB:

Let's get right into Git Hub! What is it? Basically, in laymen's terms, one could think of Git as a filing service for different drafts of a particular document. Technically speaking, Git is a version control system, or repository hosting service, and an extremely powerful command line tool. However, a web graphical interface for Git is also available, and it can be very useful for those who are not comfortable with working on the command line, but you have to break the comfort of command-line comfortless behavior if you plan on getting around in the cyber world.

Primarily, Git is used as a code repository. And some may ask: just what is a repository in this context? Well, it is simply this: a place where something is stored and managed. In most cases, code. However, Git is not limited to managing just a code repository, but its use extends far and wide to many types of draft files; it can manage Word documents, Excel documents, Adobe Premiere projects, Adobe Photo Shop projects, Final Cut projects, and the list goes on and on ....

WHY USE VERSION CONTROL:

The reason one would use a version control system like GitHub is pretty obvious. It makes it easier for one person or a group of people to keep up with incremental updates or changes made to a draft item, all at neck-breaking speed. The premise behind a version control systems like Git, and one of its flagship functionalities, is forking/branching. This functionality is such that a developer starts a master project (repository) in his/her account, then other users can fork (copy) the master repository to their account. Now, the user can make modifications to the project, locally, without affecting the master/central repository for which they do not have write access, and when the changes have been made, reviewed, and verified by the developer or his/her team, the local changes can be proposed to the owner of the repository, and he/she can decide whether to pull in the changes into the master/central repository. This method promotes a more granular creative noodling around with a project, resulting in a more efficient, or less buggy code base, all while extending a flexibility to the developer or development team that would not have existed were they all logged into a central repository, without the forking/branching functionality, working in the project simultaneously, or even successively.

HOW DO YOU GET AN ACCOUNT:

Now let's get into the driver's seat and take GitHub for a spin. The first thing we want to do is to go to: https://github.com/ and sign up for a free account. Once we create an account, using a valid email. Verify your email account. Now, that your email has been verified, we are ready to rock and roll.

HOW DO YOU USE THE COMMAND LINE WITH GITHUB:

So, the Git shell has about six basic commands, demonstrated below, that will be used over and over ... more often than not. Let's setup our environment to get started using the Git command line:

1. On the main page, click on panel number 1 towards the top of the page where the title says "GitHub Bootcamp".

2. Once the new page opens, scroll down to the section "Setting up Git", and select the hyperlink in the first line that says: "GitHub Desktop ...."

Set Up Git

MAC | WINDOWS | LINUX | ALL

At the heart of GitHub is an open source version control system (VCS) called *Git*. Git is responsible for everything GitHub-related that happens locally on your computer.

If you're not comfortable using the command line right now, GitHub lets you complete many Git-related actions *without* using the command line, including:

- Creating a repository
- Forking a repository
- Being social

However, if you find that you need to use Git, we can help you set it up!

Tips:

- GitHub has a Desktop client! You can use it without ever touching the command line.
- To learn more about Git, see "Getting Started - Git Basics" on the git-scm website.

Input

Setting up Git

1. Download and install the latest version of GitHub Desktop. This will automatically install Git *and* keep it up-to-date for you.
2. On your computer, open the **Git Shell** application.

Article versions

GitHub.com
GitHub Enterprise 2.3
GitHub Enterprise 2.2
GitHub Enterprise 2.1
GitHub Enterprise 2.0

FIGURE 1.0 GitHub Desktop download link

3. Now, choose the "Download GitHub Desktop" link to begin the download (Be mindful of your PC requirements and select the appropriate link).

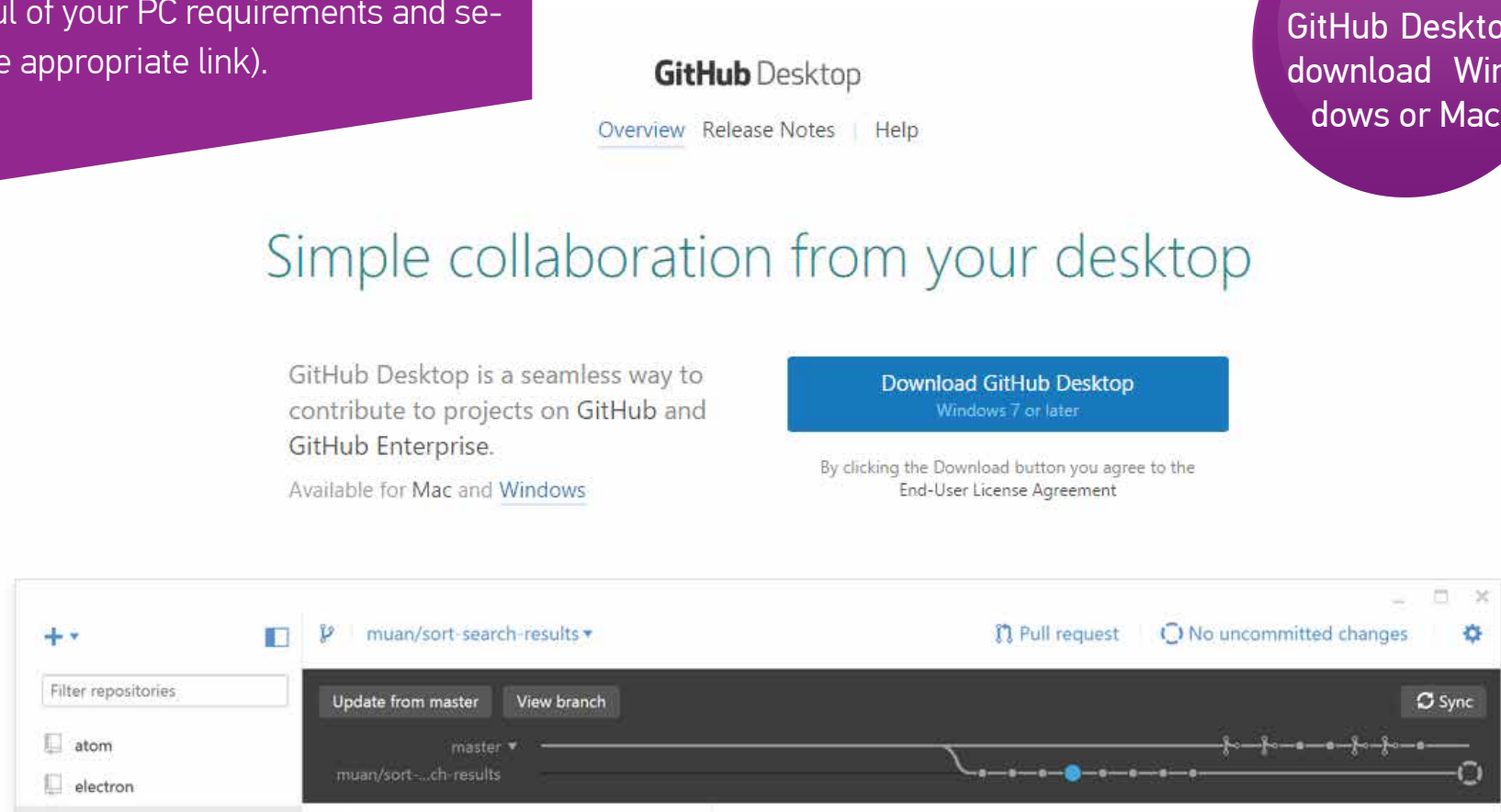


FIGURE 1.1 GitHub Desktop download Windows or Mac

4. When the download completes, double-click the downloaded file to start installation.

5. While the app is installing, go back to you your GitHub login page, and click the green button on the right of the page that's labeled: "+ New repository" (The "+" icon at the top can also create a repository).

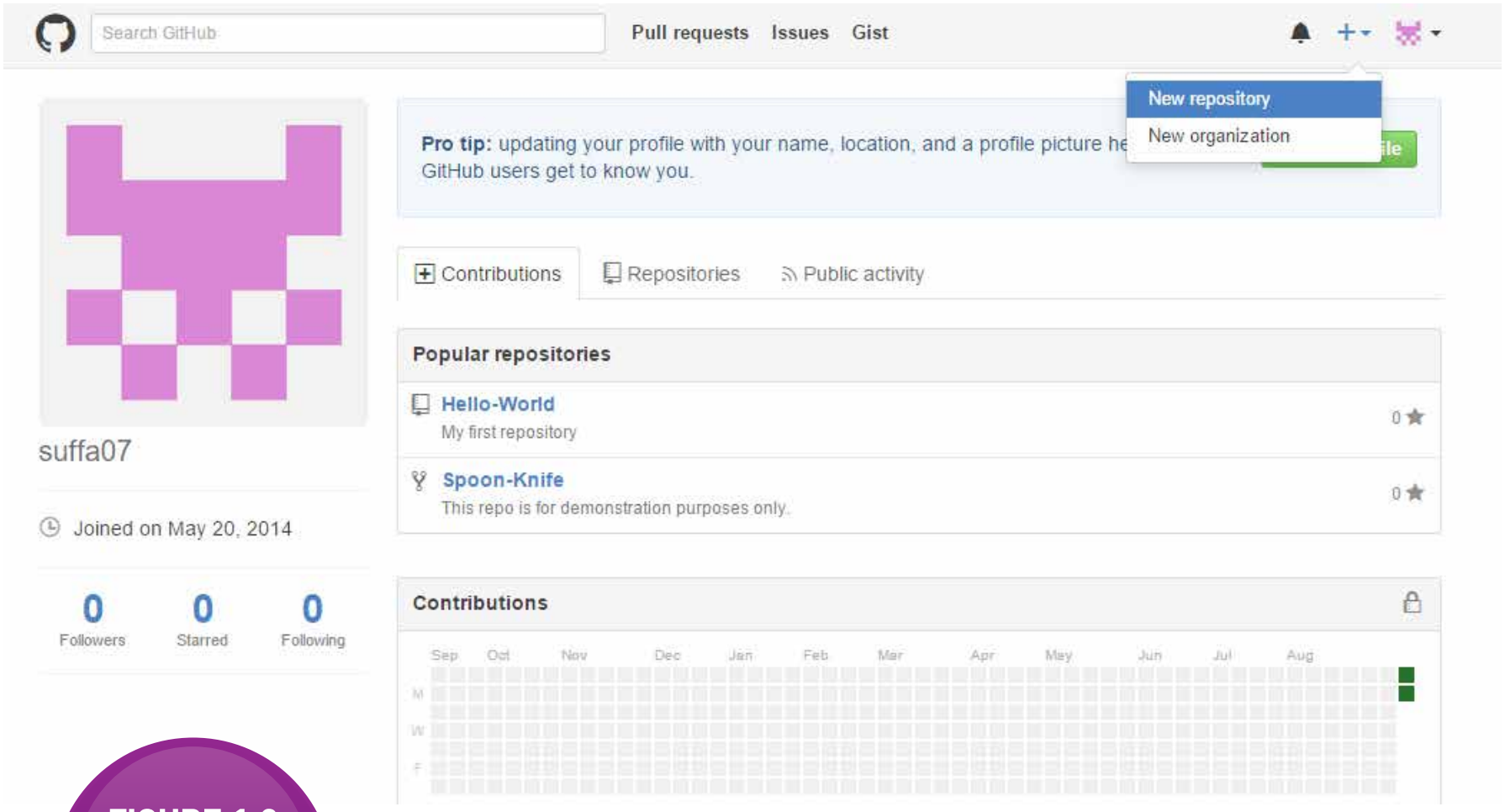


FIGURE 1.2 GitHub.com create new repository

6. When the new page opens, name your repository: "Hello World". You can give your repository a brief description if you so choose (Optional). Select whether you want the repository to Public (visible to all of GitHub) or Private (accessible to you and selected users). For the sake of our example, we will choose Public. And, select "Initialize this repository with a README" (to immediately clone the repository to your pc). Note: we are not using any specific programming language for this tutorial, so we do not have to worry about files being accidentally included, neither do we need a license. But you can choose to ignore files for a particular language if needed by clicking the "Add .gitignore" button and selecting the language to ignore. You can also choose to add a license to your project by clicking the "Add a license" button, and selecting the appropriate license.

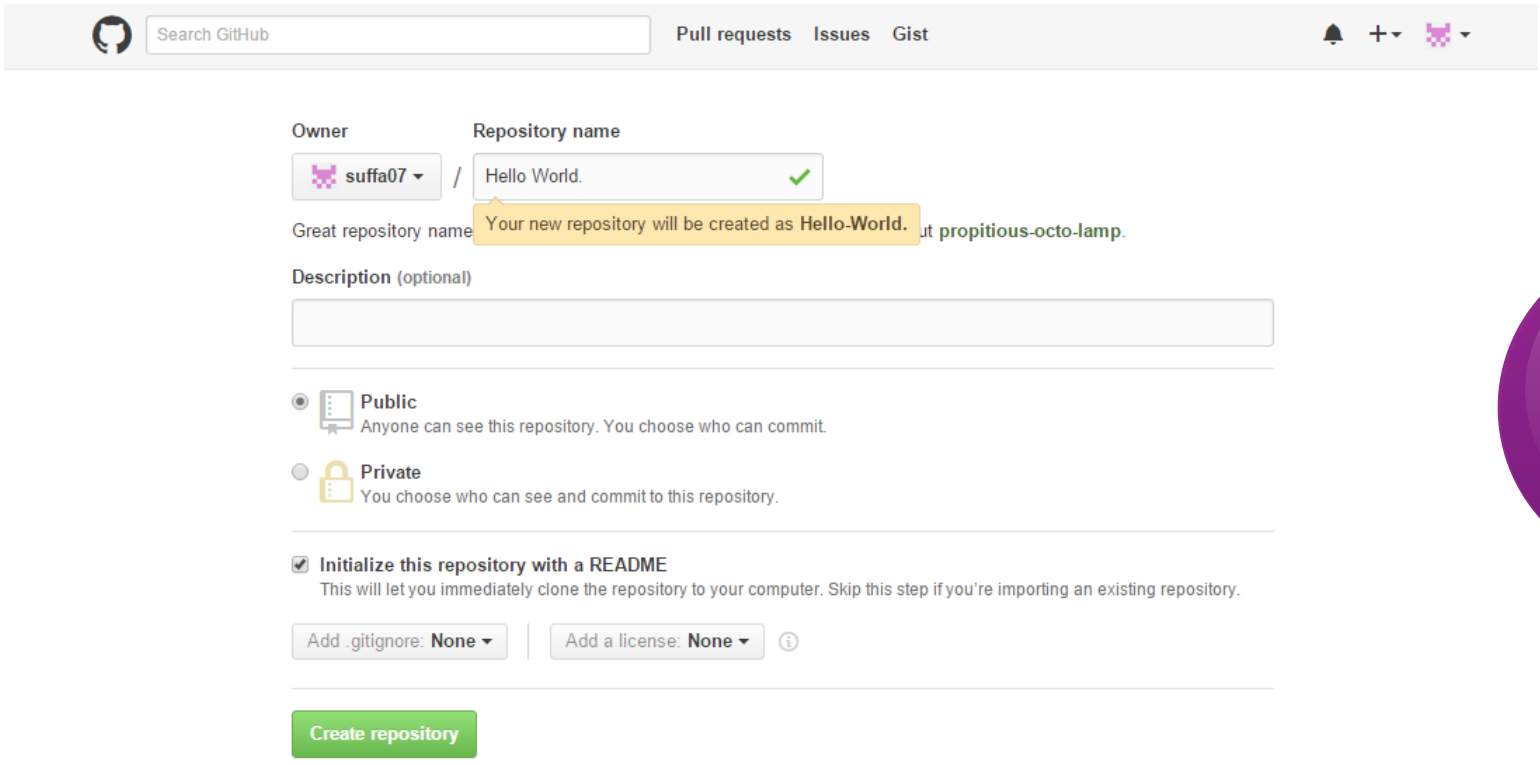


FIGURE 1.3 GitHub.com Name respository

7. Click the green “Create repository” button to create our new repository.

8. We opted to initialize our repository with a README file, which can describe our project in minute details or provide instructions on how our project is to be used.

9. We will quickly commit a change to our project, click the hyperlink of the “README.md” file.

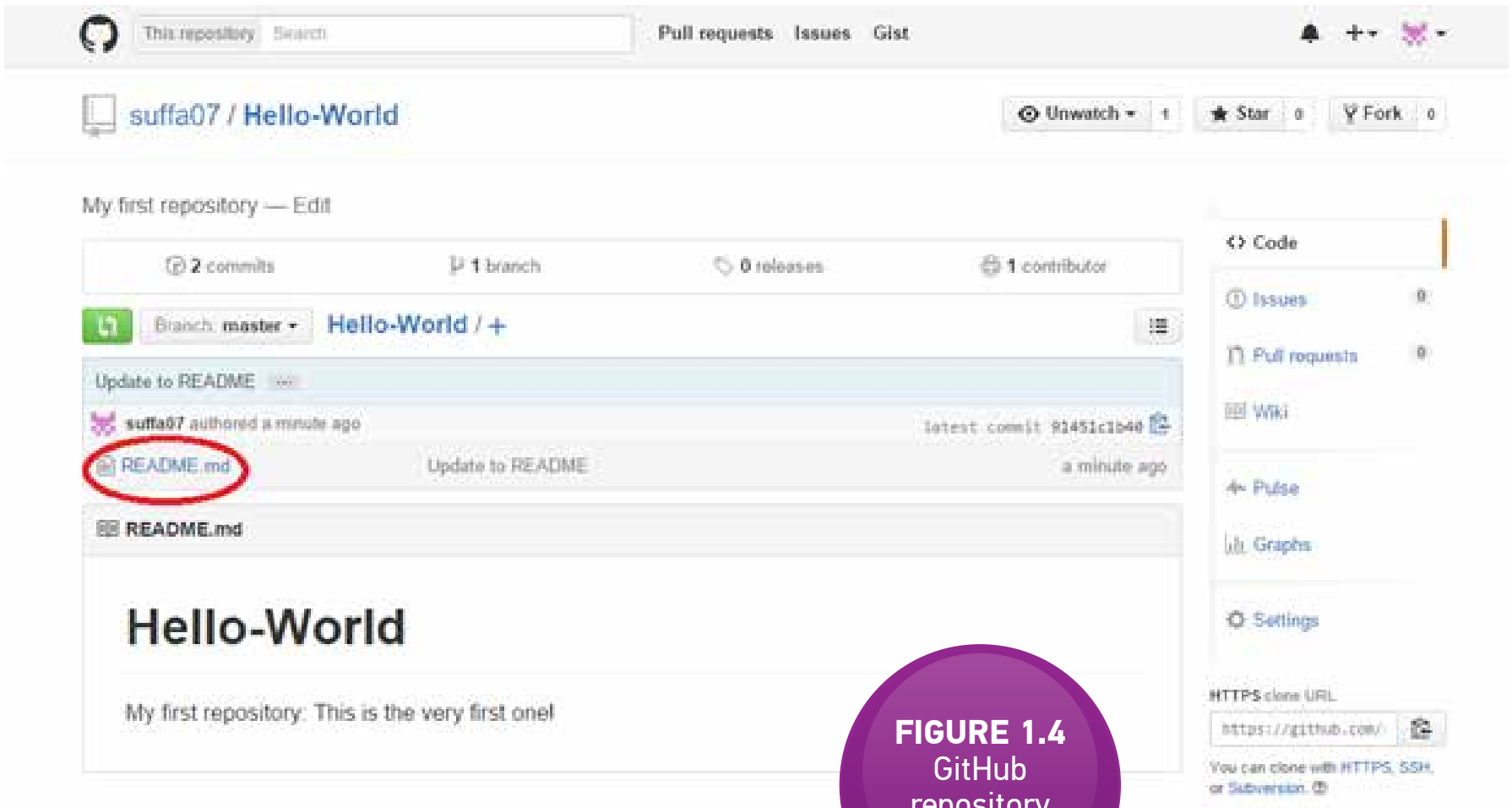


FIGURE 1.4 GitHub repository README.md

10. Now, click the pencil icon to edit, towards the right of the screen, and type some generic information about your project in the editable region.

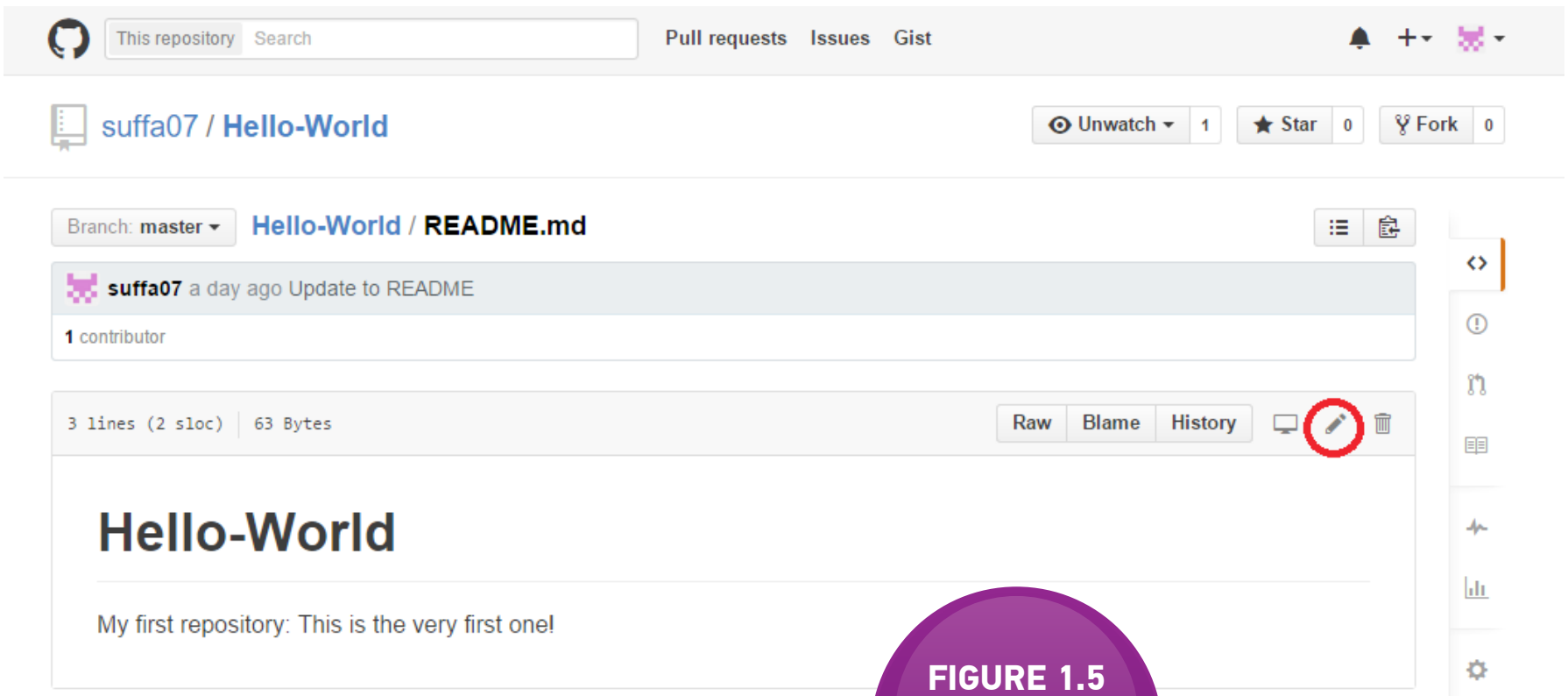


FIGURE 1.5 GitHub Edit the README.md file

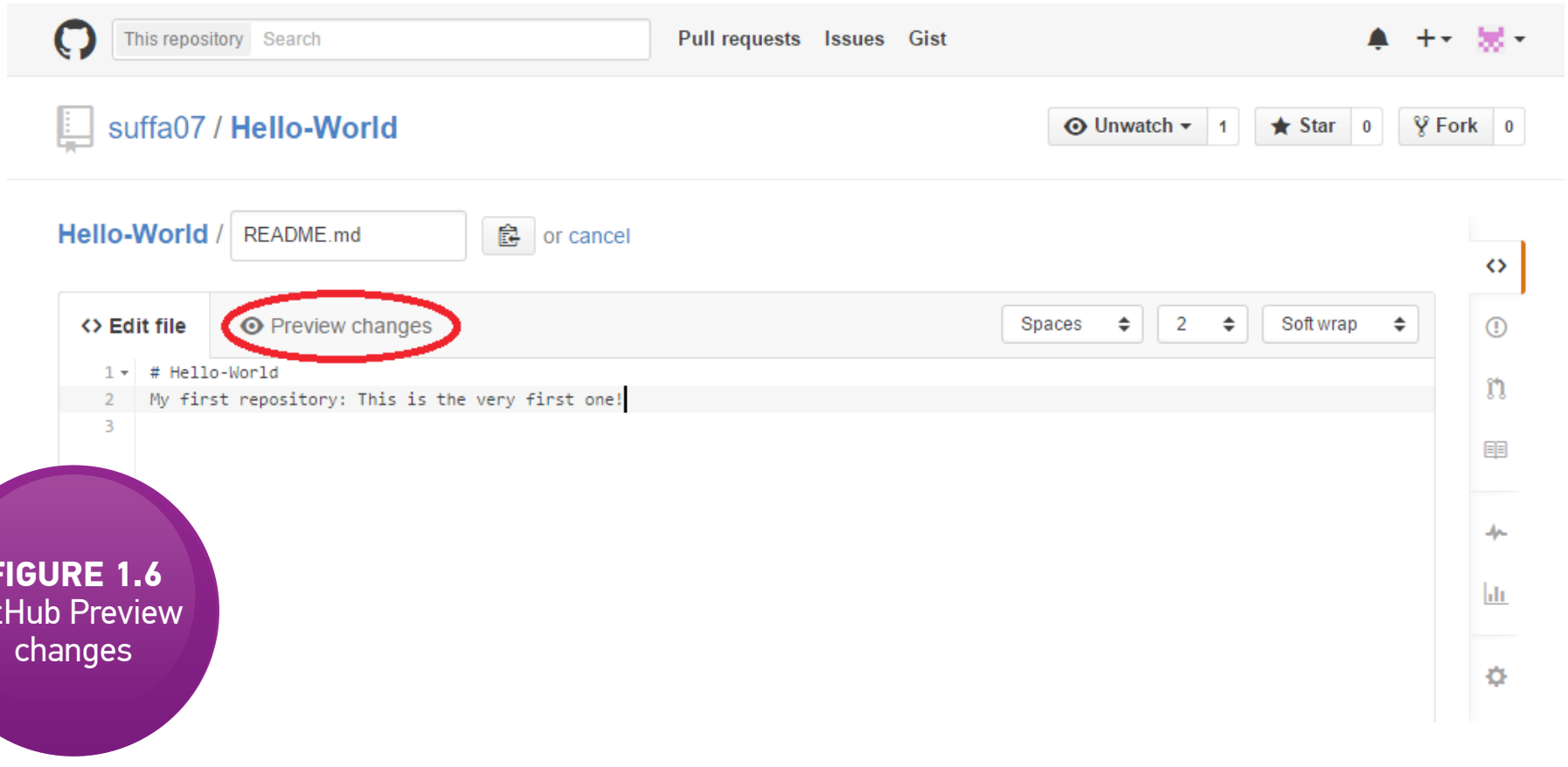


FIGURE 1.6 GitHub Preview changes

11. Above the editable region, click the "Preview changes" tab, and you will see the modifications made to the file next to a green vertical line. The original text is displayed next to a red vertical line.

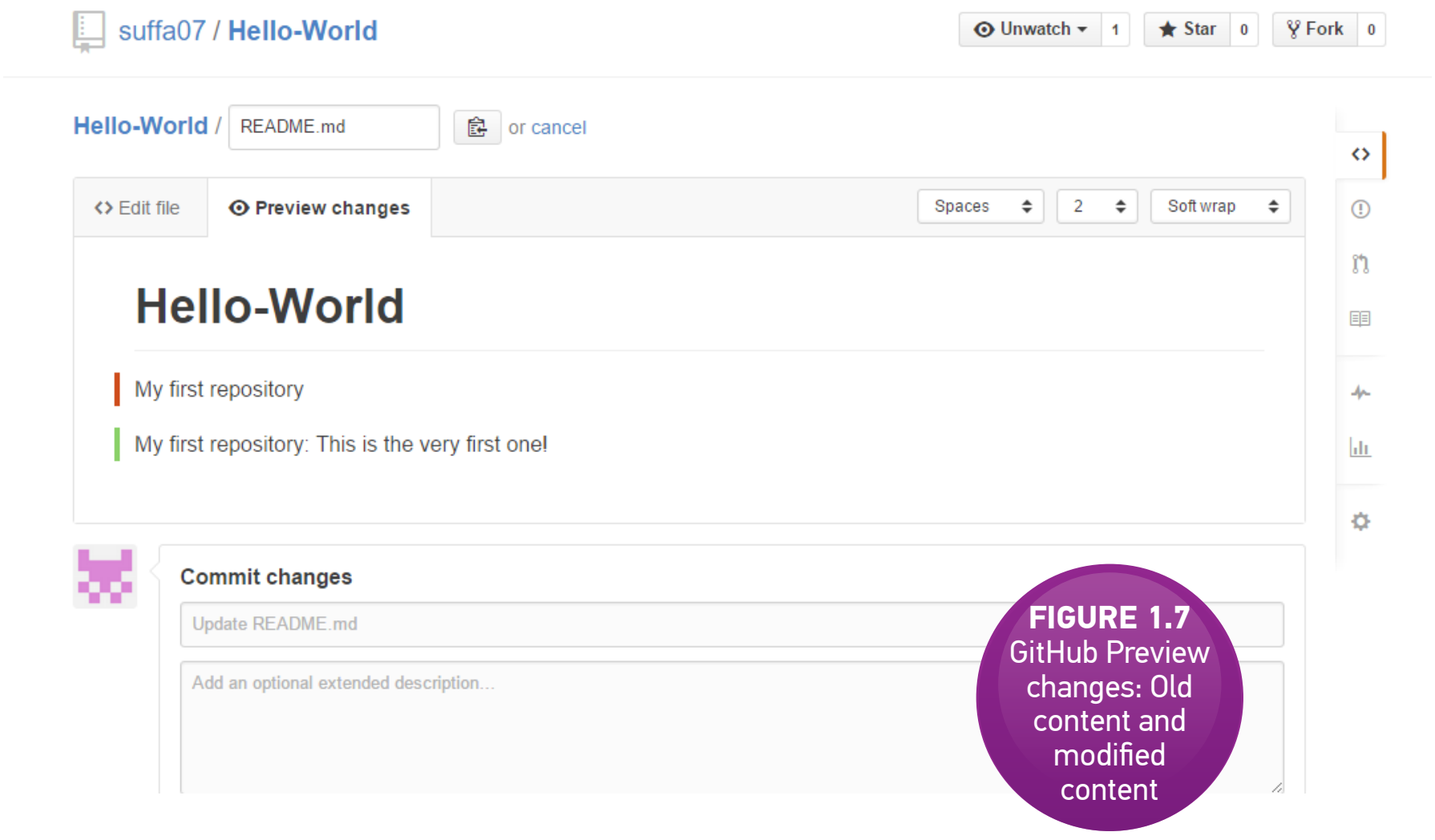


FIGURE 1.7 GitHub Preview changes: Old content and modified content

12. Scroll to the bottom, add notes to "Commit changes", and you will see commit directly to the master branch or create a new branch. For the sake of brevity, we will choose: "Commit directly to the master branch", and click the "Commit changes" button.

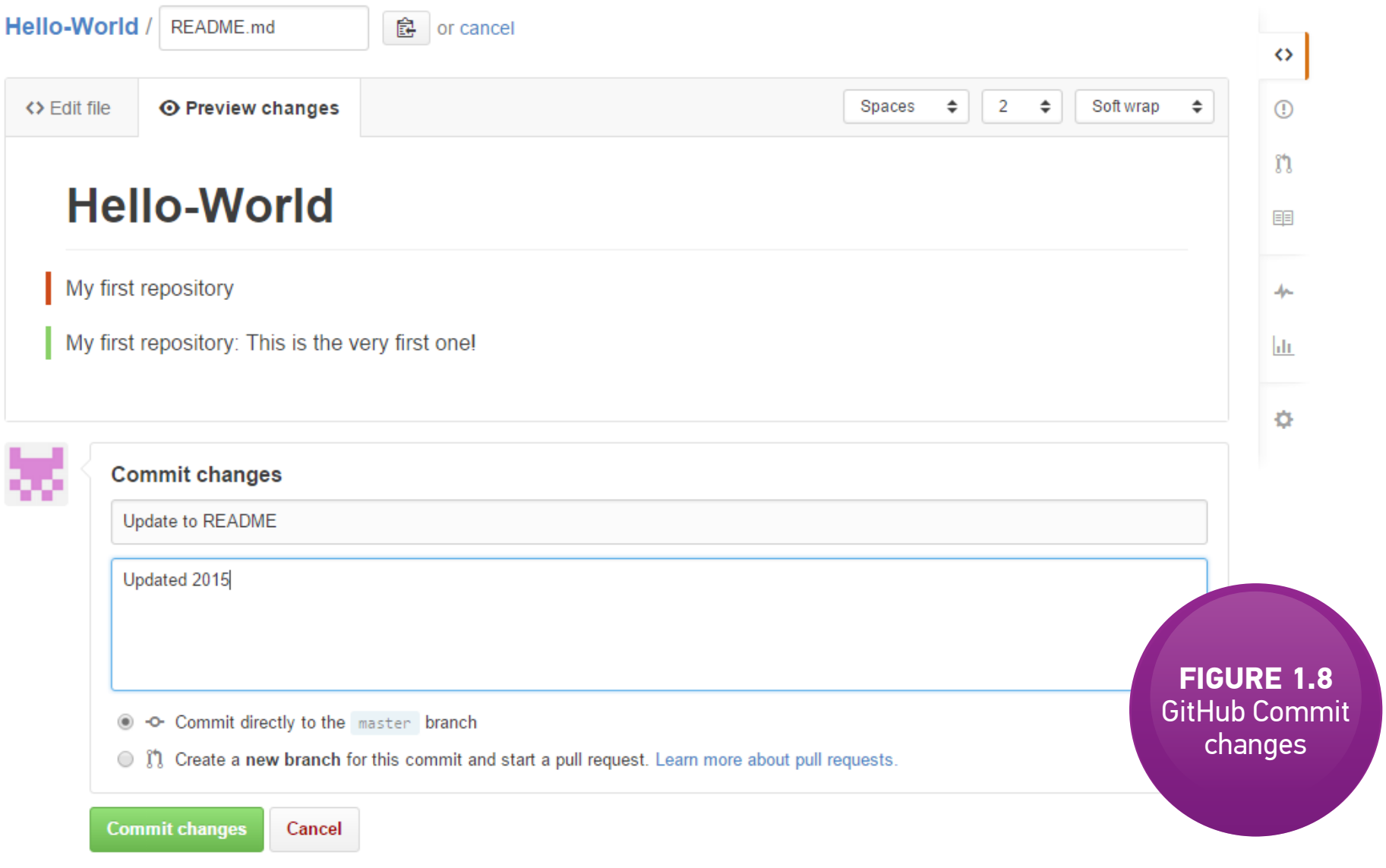


FIGURE 1.8 GitHub Commit changes

13. Now, click on the Hello-World link at the top of the screen to get back to our main repository window, and click on the "Copy clipboard" button towards the bottom-right of the page, just below "HTTPS clone URL".

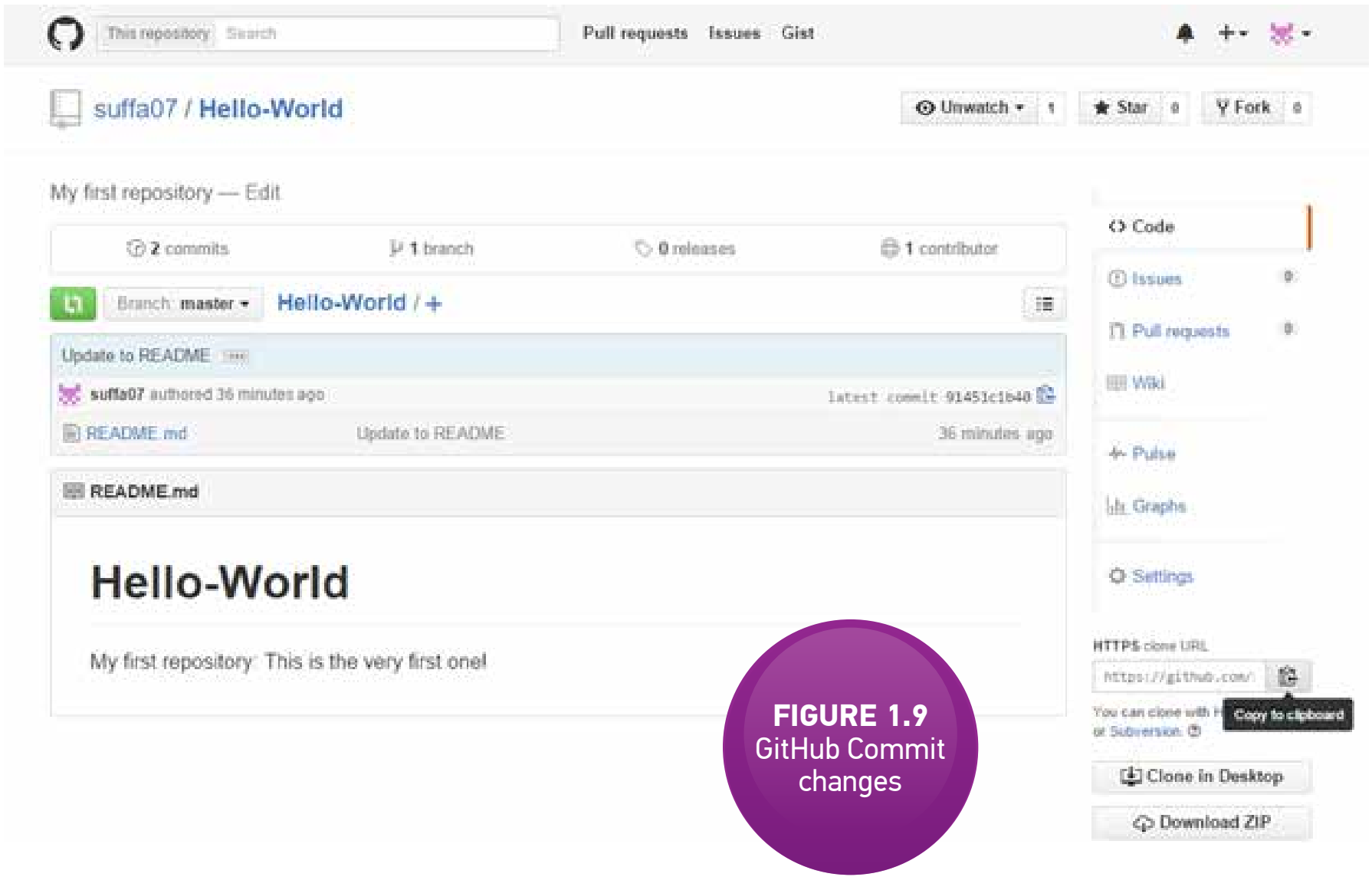


FIGURE 1.9 GitHub Commit changes

14. Back to your pc, once the installation is complete, head over to your programs and find GitHub, and then select GitHub Shell (a shell environment will open at C:\Users\Your User Name\Documents\GitHub> , or some variant depending on your operating system: Windows: dos-cmd or PowerShell, Mac: Terminal, Linux: Terminal).

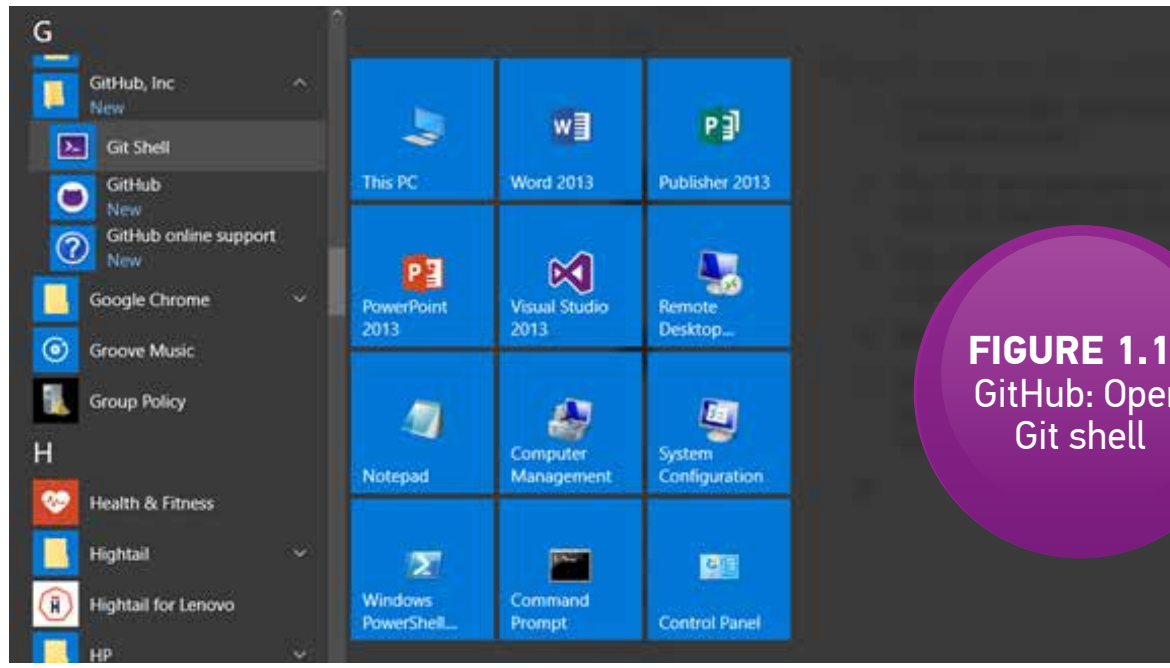


FIGURE 1.10 GitHub: Open Git shell

15. On the command line enter: git (this shows all the git commands)

```
posh~git ~ GitHub [master]
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]> git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -g' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]>
```

FIGURE 2.0 GitHub: git commands

16. The first thing we are going to do is to insert our name, so when a commit is executed it will be associated with our user. Enter the following command in the command shell: git config --global user.name “YOUR NAME”

17. Similarly, we want to enter our email address. This should be the same email associated with your Git account when signing up: git config --global user.email “ YOUR EMAIL”

```
posh~git ~ GitHub [master]
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]> git config --global user.name "Your Name"
C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]> git config --global user.email "Your Email"
C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]>
```

FIGURE 2.1 GitHub: configure name and email via git shell

18. Now we need to authenticate our Git app with GitHub using either HTTPS (recommended) or with SSH. We will use HTTPS, as it is more straightforward. Back in our browser, the Hello-World directory should still be displayed. On the lower-right of the screen, just under the Https to clone URL, select the copy to clipboard icon to copy the url.

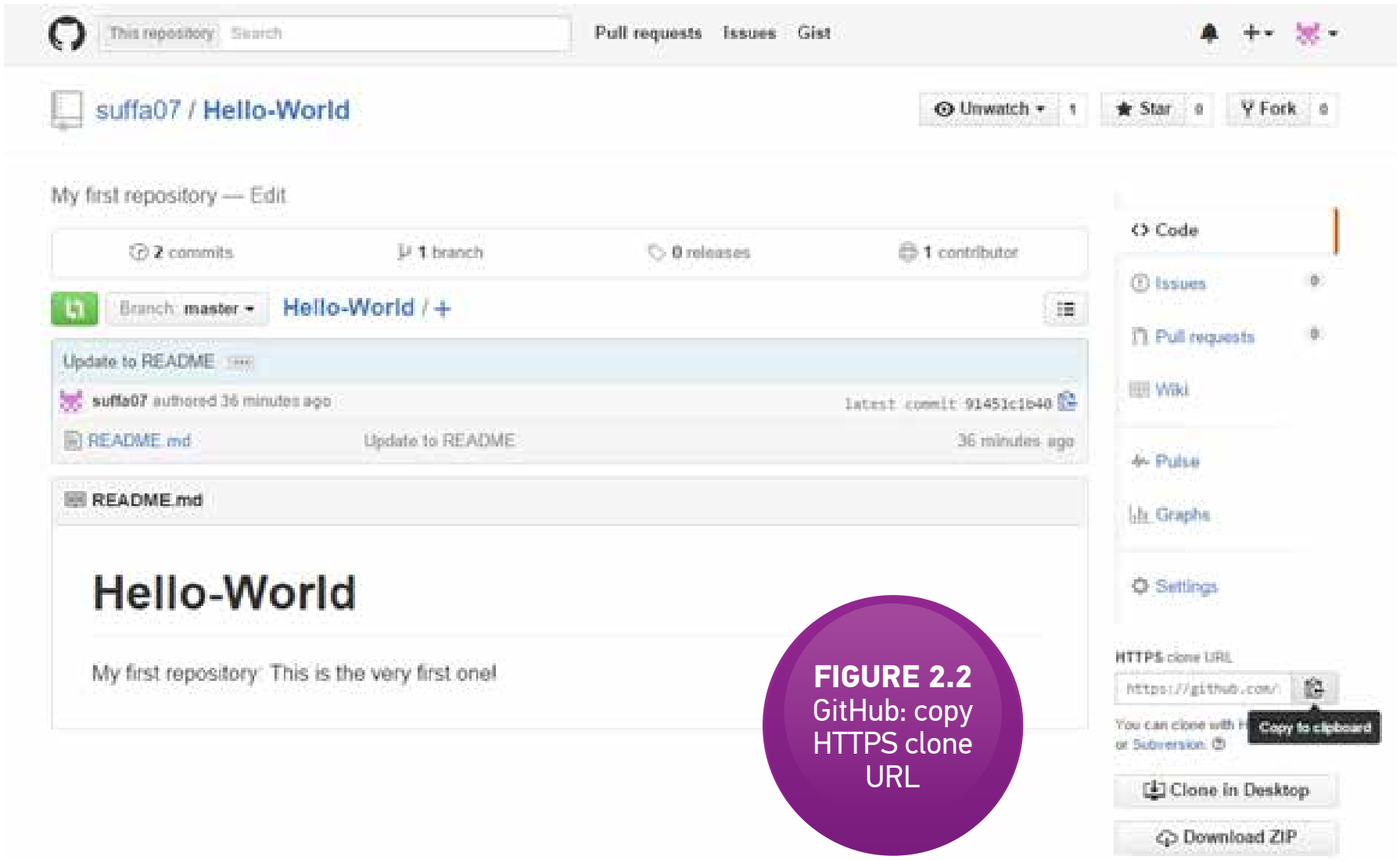


FIGURE 2.2 GitHub: copy HTTPS clone URL

19. On the Git command shell, enter our first git command (downloads a copy of our repository), git clone and paste the clone URL: **git clone** https://github.com/username/Hello-World.git (the local clone will be created on our system). Alternatively, another way is just to open the command prompt and go to the following directory: c:\Users\User Name\Documents\GitHub\Hello-World, then enter the above command + the clone url, and hit "Enter" (if you get the error "git is not recognized as an internal or external command..." you will need to set the environment path variable for the git executable file on your pc).

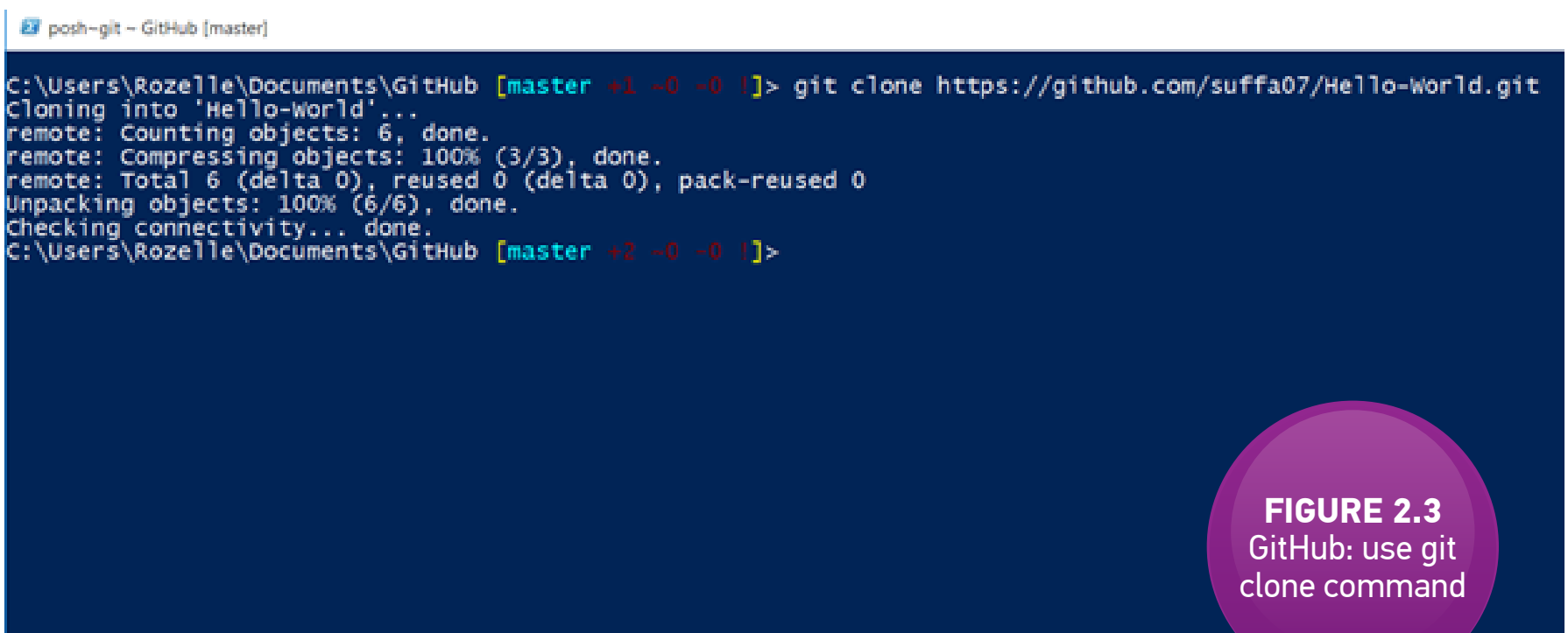


FIGURE 2.3 GitHub: use git clone command

20. Now, on the command line if you enter: dir (ls for Mac or Linux – Terminal) you will see the folder created in GitHub (Hello-World) displayed on your local machine. Enter cd Hello-World to change paths to our repository folder.

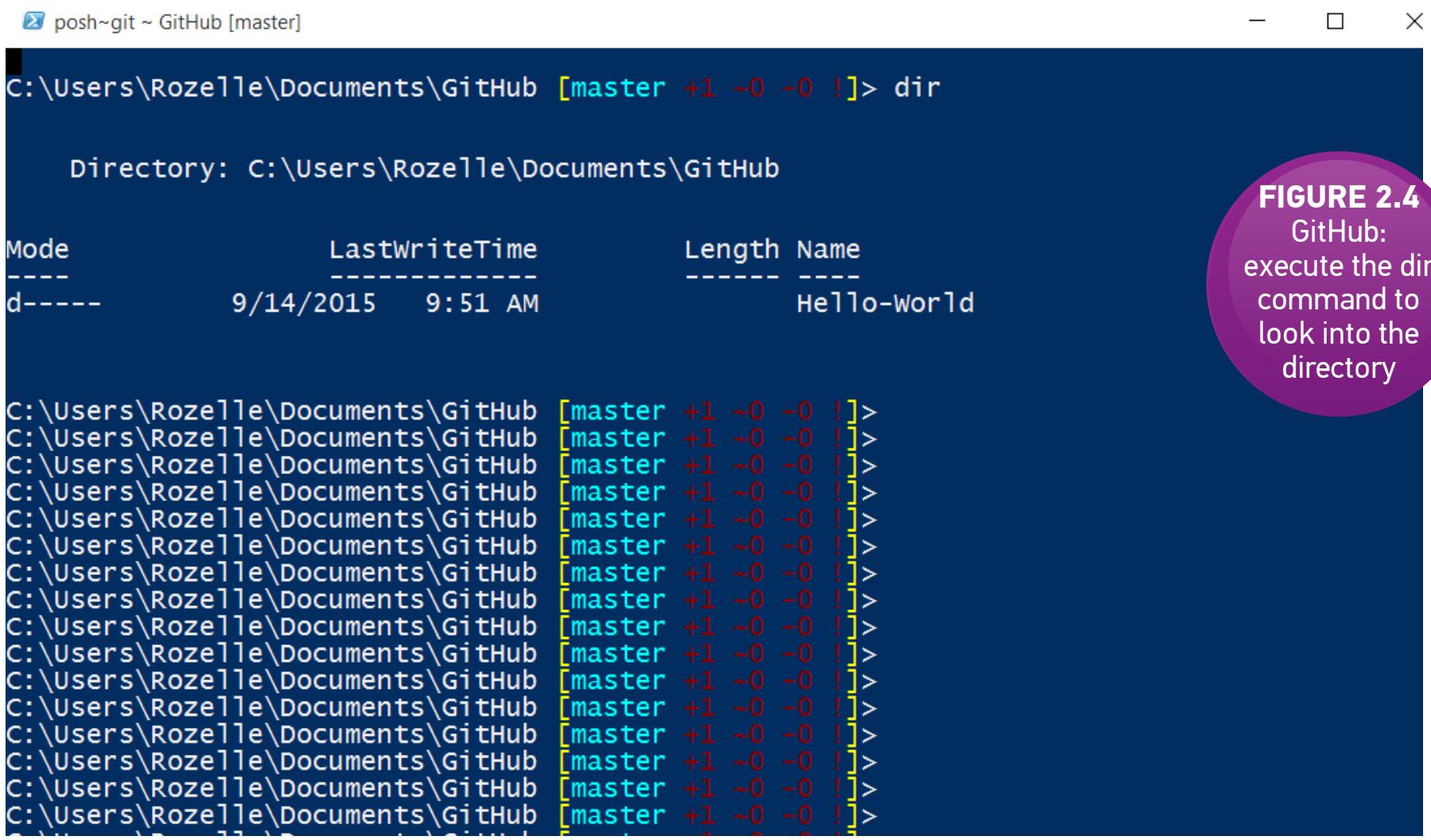


FIGURE 2.4 GitHub: execute the dir command to look into the directory

21. On your pc, open up file explorer and go to directory: c:\Users\User Name\Documents\GitHub\Hello-World.

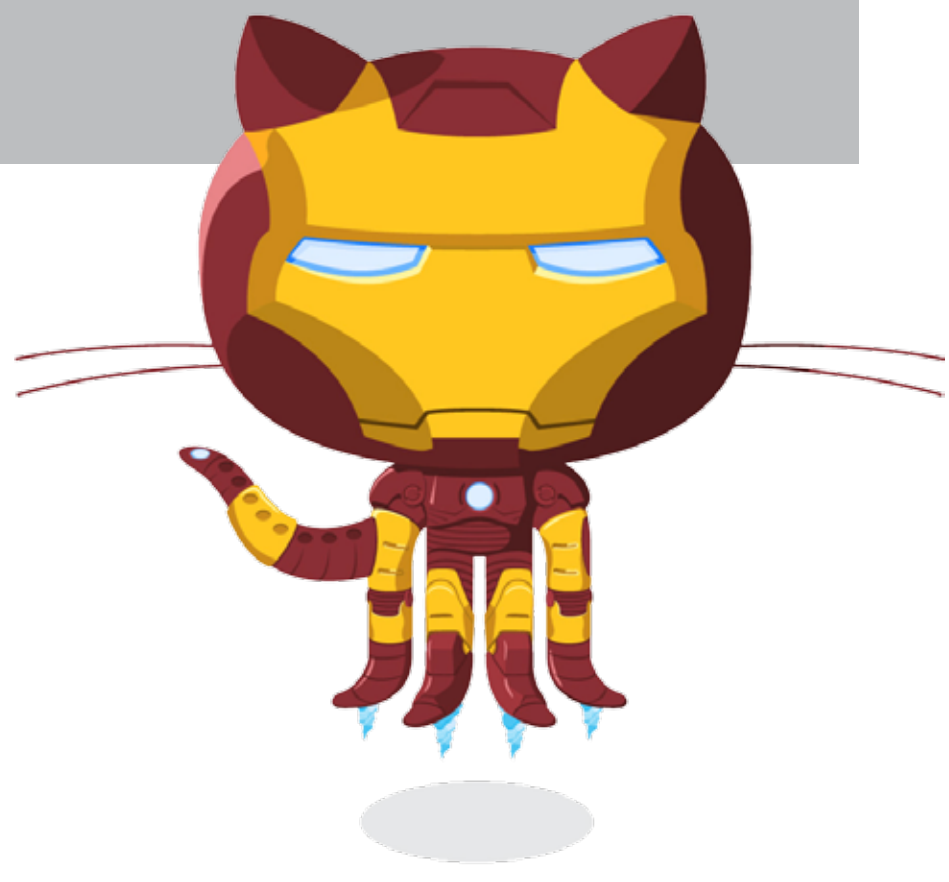
22. Open up notepad and create a file with the following HTML Data:

CODE LISTING: HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>Git Page</title>
   </head>
</html>
```

23. Save and name the page: index.html, and place it in your Hello-World directory opened up with File Explorer

24. Now, on the command line enter our next git command (shows the status of the files in our repository): git status (notice that we are shown our original repository, and our index.html file shows up on our local machine, but it is marked as untracked).



```
C:\Users\Rozelle\Documents\GitHub [master +2 ~0 -0 !]> cd  .\Hello-World
C:\Users\Rozelle\Documents\GitHub\Hello-World [master]> ls


    Directory: C:\Users\Rozelle\Documents\GitHub\Hello-World


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/13/2015   5:50 PM             65 README.md


C:\Users\Rozelle\Documents\GitHub\Hello-World [master]> 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)

        Hello-World/

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0 !]> 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)

        Hello-World/
        index.html

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +2 ~0 -0 !]>
```

FIGURE 2.5
GitHub: git status command is executed on newly added file (index.html)

25. Let's use our next git command: git add index.html (this command adds the index.html file to our project. To add all files in our directory you would enter: git add . or git add -A).

```
-a----        9/13/2015   5:50 PM             65 README.md


C:\Users\Rozelle\Documents\GitHub\Hello-World [master]> 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)

        Hello-World/

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0 !]> 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)

        Hello-World/
        index.html

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +2 ~0 -0 !]> git add index.ht
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0 | +1 ~0 -0 !]> git s
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

FIGURE 2.6
GitHub: git add command executed on index.html file

26. Enter command: git status

27. Now, we see our index.html file is a change to be committed.

28. To commit to our repository, we enter: git commit –m "added index.html" (the –m flag stands for message; if there were more than one file to be committed, this command would add all files.) see Figure 2.7

29. The index.html file is now committed, but, so far, only to the local machine.

30. Enter the following command to sync files from our local machine with GitHub.com: git push (see Figure 2.7)

```
nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +2 ~0 -0 !]> git add index.html
C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0 | +1 ~0 -0 !]> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Hello-World/

C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0 | +1 ~0 -0 !]> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   index.html

C:\Users\Rozelle\Documents\GitHub\Hello-World [master +1 ~0 -0]> git commit -m "add index.html"
[master b86f216] add index.html
 1 file changed, 6 insertions(+)
 create mode 100644 index.html
C:\Users\Rozelle\Documents\GitHub\Hello-World [master]> git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/suffa07/Hello-World.git
   91451c1..b86f216  master -> master
C:\Users\Rozelle\Documents\GitHub\Hello-World [master]>
```

FIGURE 2.7
GitHub: git commit and git push commands executed

31. If you enter: git status, you will now see that "your branch is up-to-date with origin/master".

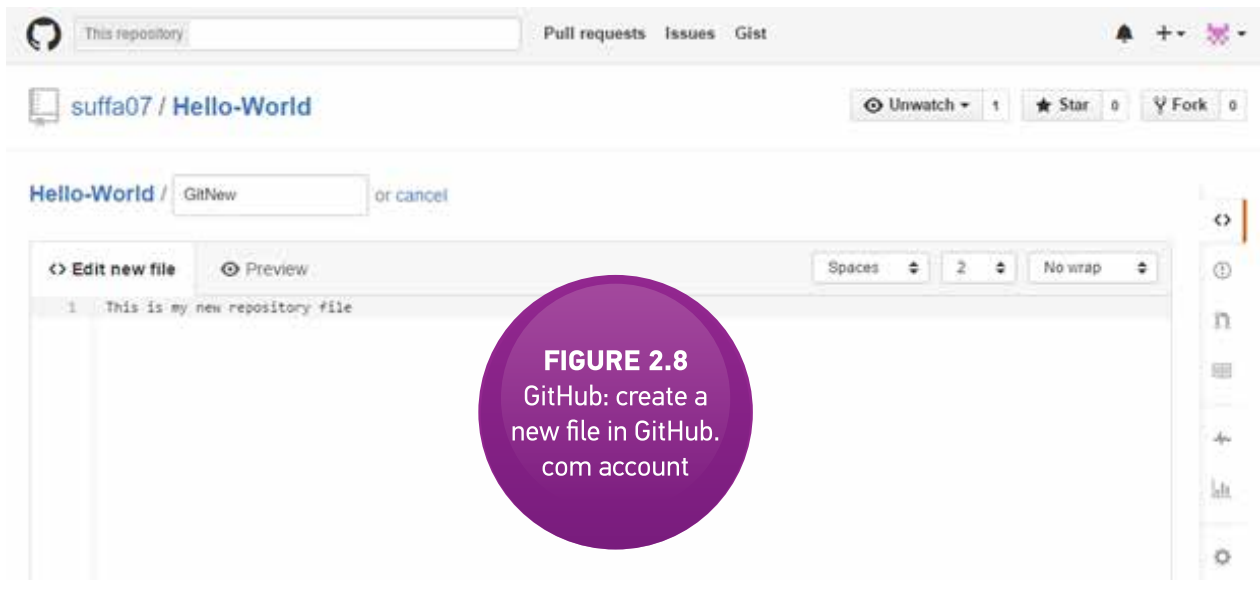


FIGURE 2.8
GitHub: create a new file in GitHub.com account

32. Now, if you go back to your browser (GitHub.com) and refresh your repository page, you will notice that the index.html file is listed there.

33. Let's look at an example for the pull command. Go to your GitHub account and select your Hello-World repository. Select the "+" (create new file) button next to your repository name.

34. Now, give your file a name ("GitNew.txt"), and add a line of text ("This is my new repository file").

35. Click the "Preview" tab, and add a note title ("New Git File") and description ("This will demo the pull command") below.

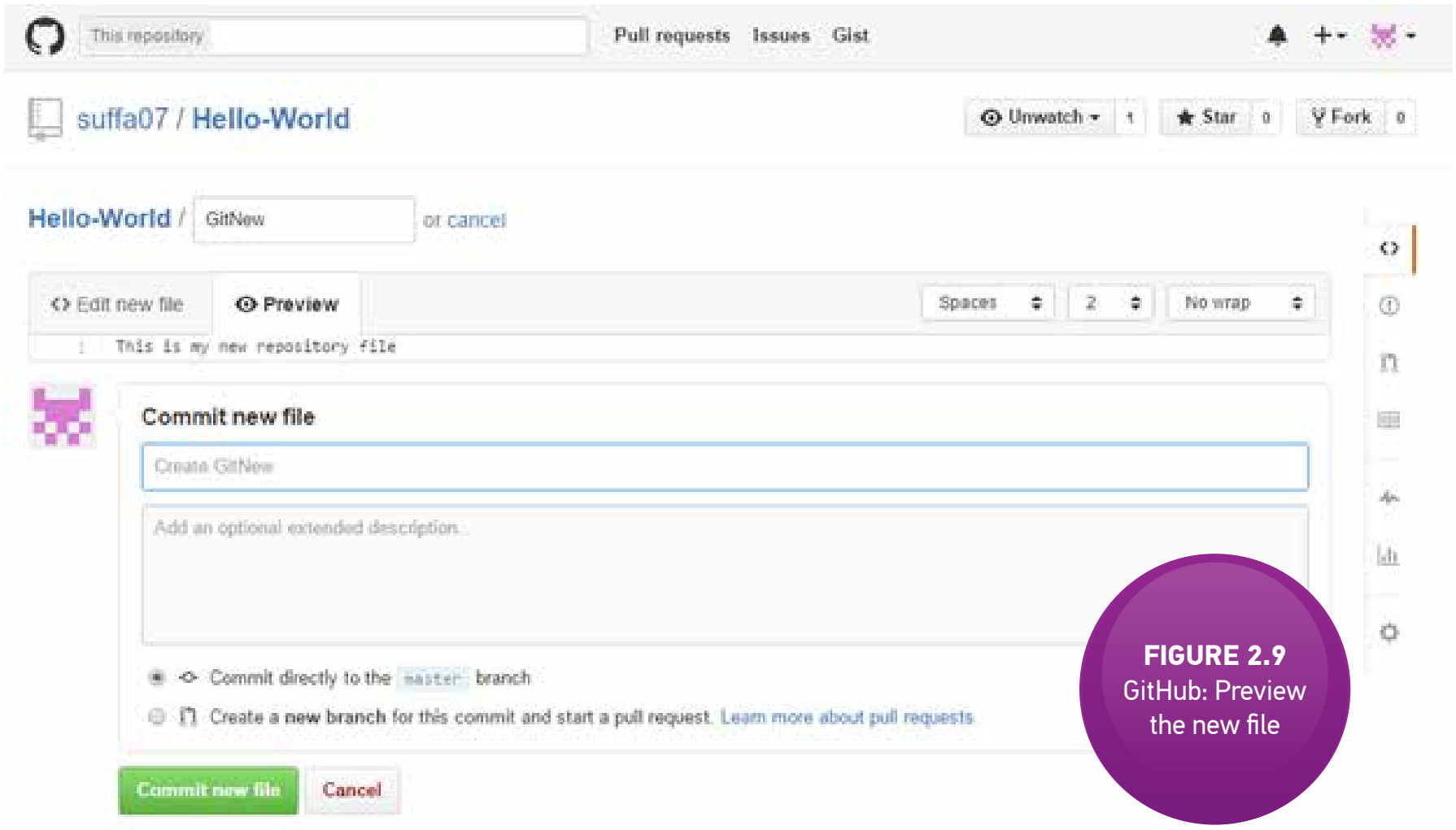


FIGURE 2.9
GitHub: Preview the new file

36. Select the radio button: "Commit directly to the master branch, and click the green "Commit new file" button.

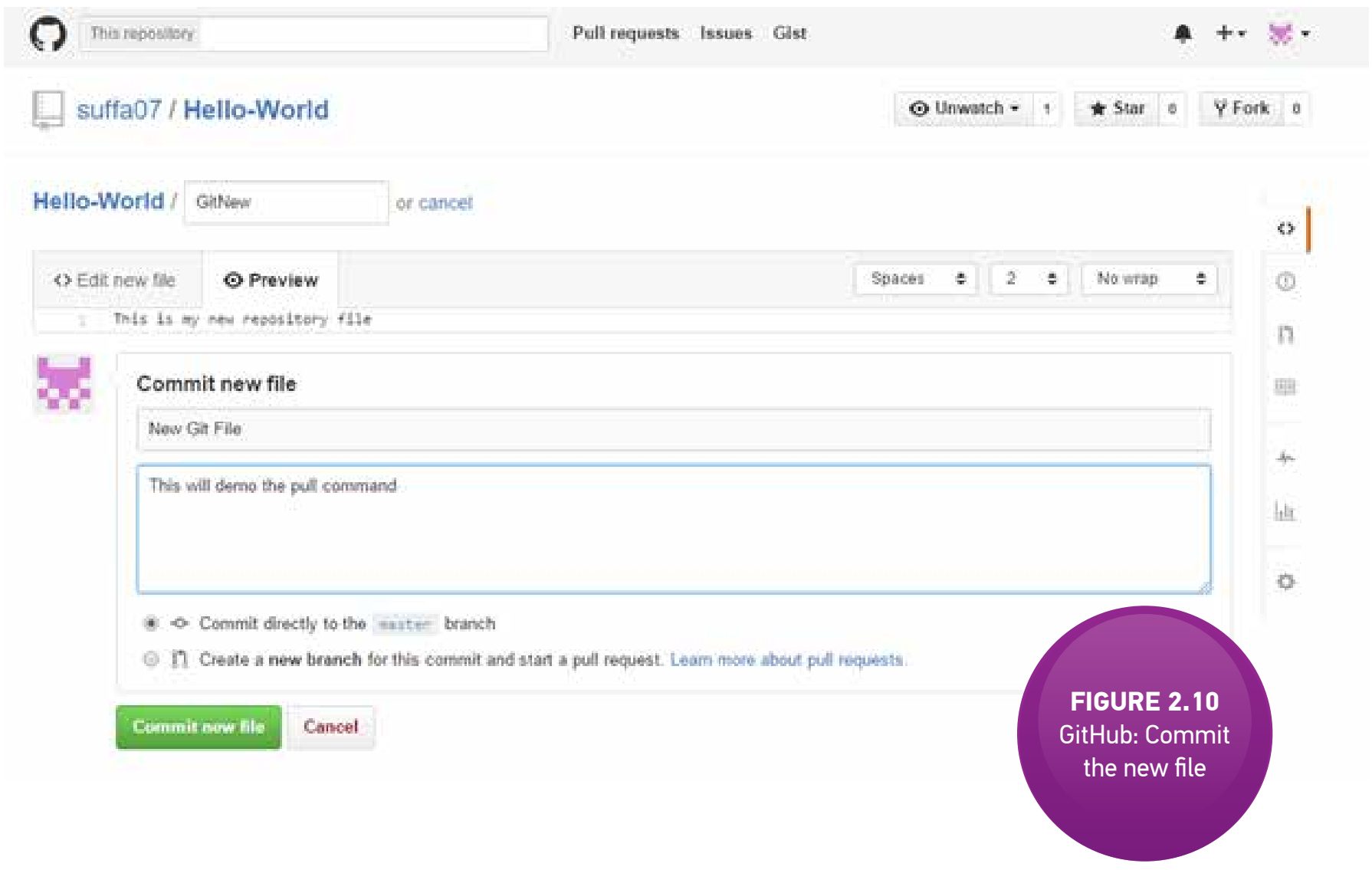


FIGURE 2.10
GitHub: Commit the new file

37. You will now see this new file ("GitNew.txt") in your list of files in your repository on GitHub.

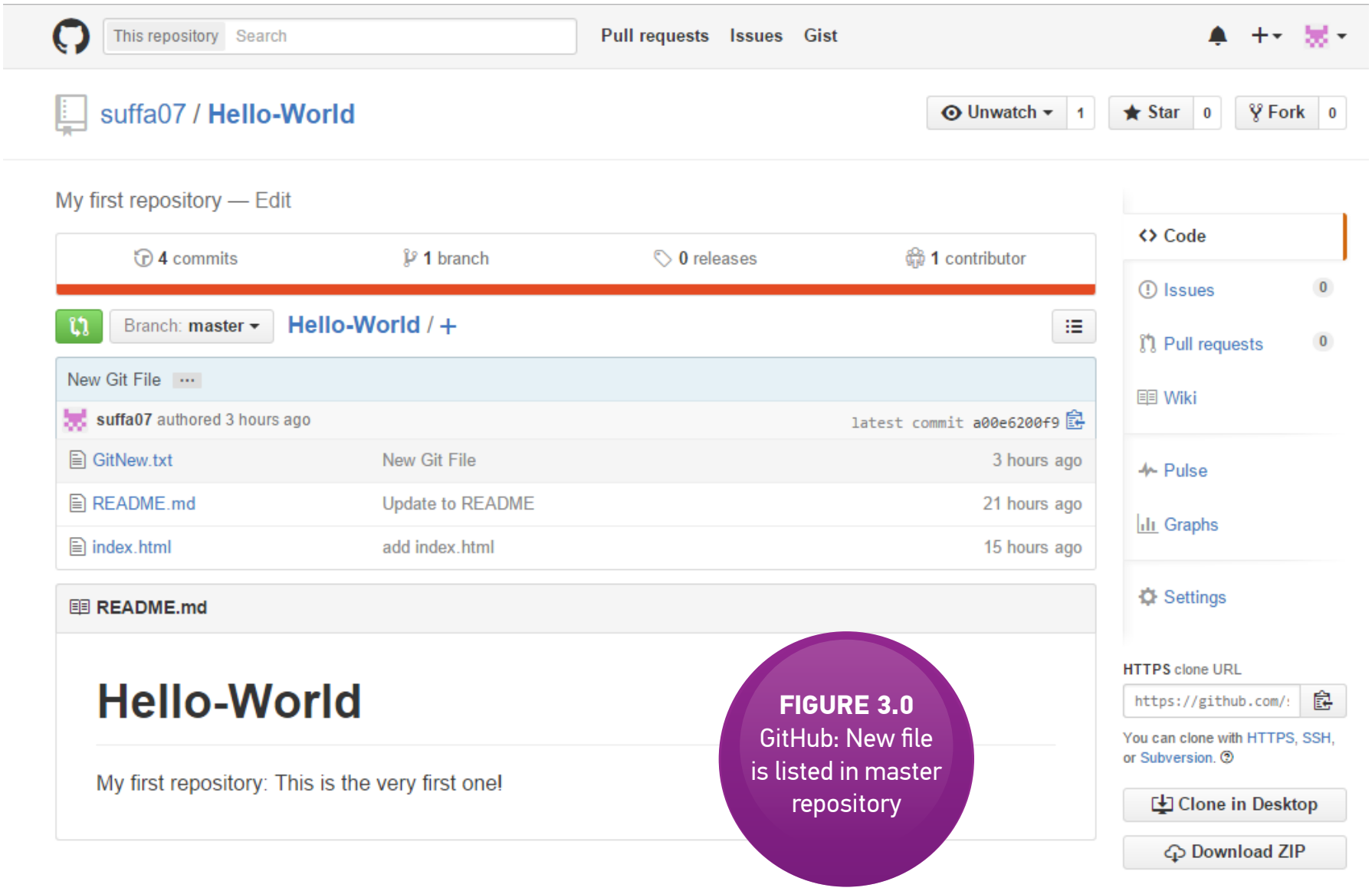


FIGURE 3.0
GitHub: New file is listed in master repository

38. To get this file in your local machine, go back to your git shell, and enter the following command: **git pull**

39. You will see a message in the command shell that the GitNew.txt file was created. And if you enter the command: dir, a list of all files in your local directory will be displayed (the same files in your original repository).

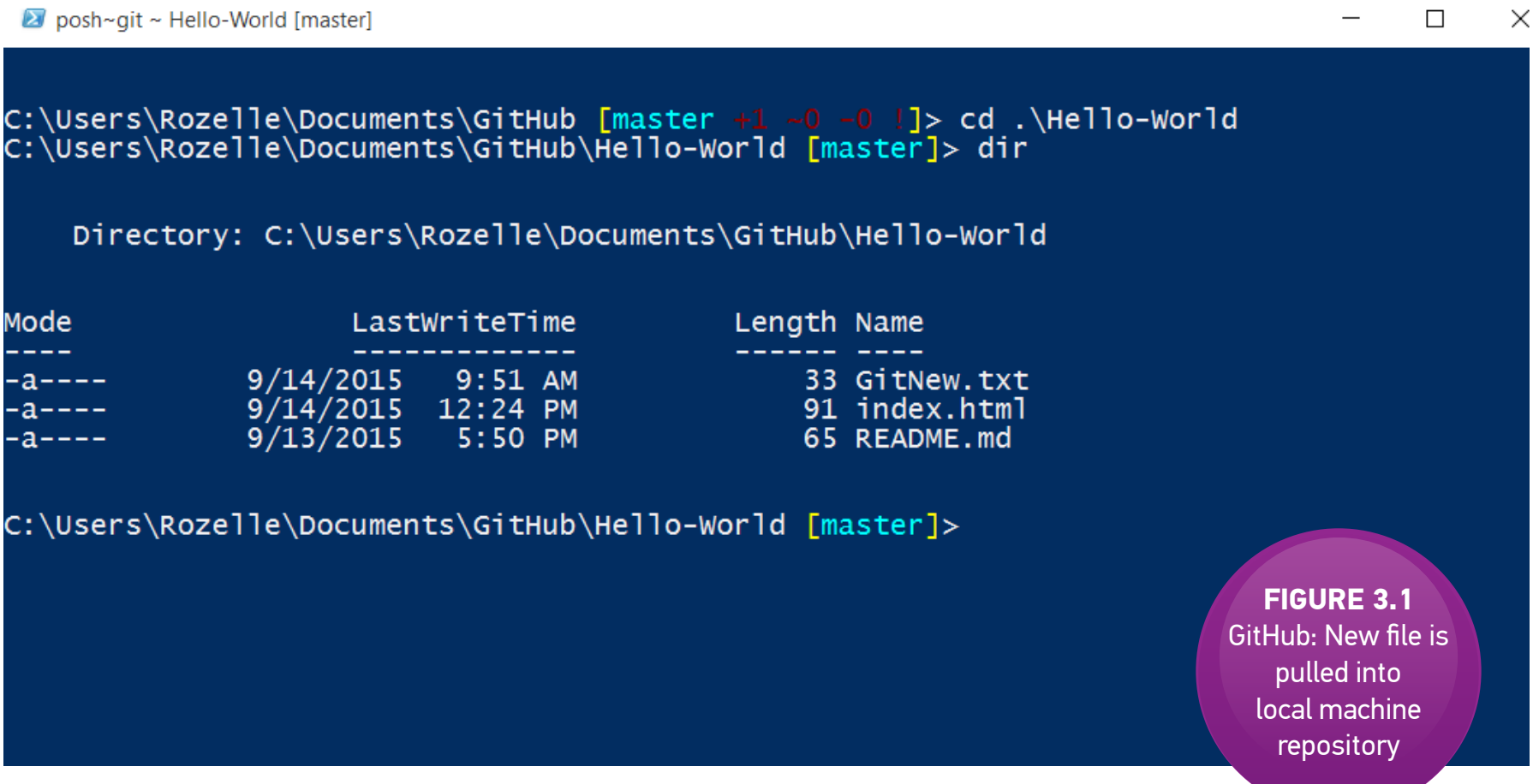


FIGURE 3.1
GitHub: New file is pulled into local machine repository

HOW DO YOU USE A GITHUB CLIENT:

The GitHub client is a graphical interface that is designed to make things easier for those who are intimidated with working on the command line. It's very intuitive and easy to use. However, I would encourage that you really learn to use GitHub with the git command line, as this will give you a better understanding of how and what is happening. Additionally, if you go to work for some company as a developer, you will almost assuredly be required to use GitHub or whatever version control system used via the command line. Let's get into it:

1. Head over to your programs and find GitHub, and then select the GitHub icon.
2. Once the app opens, select to "Skip setup".

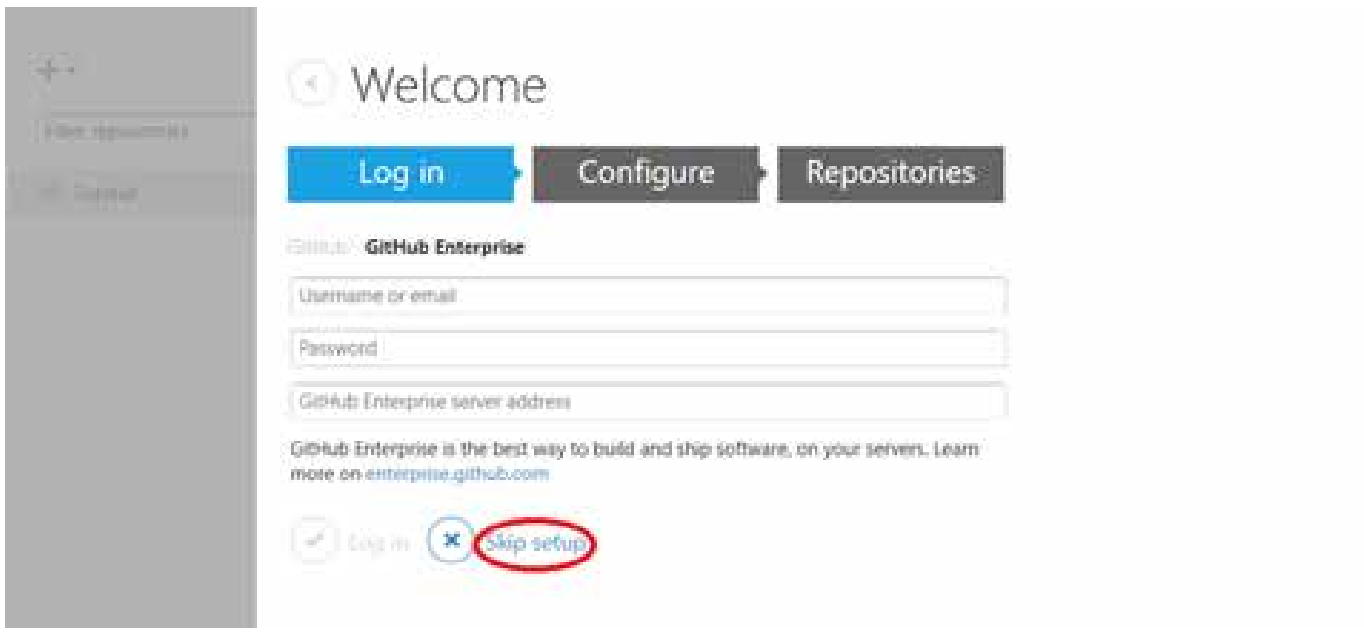


FIGURE 3.2
GitHub: Skip Git GUI setup

3. Select the "+" icon in the upper left corner.

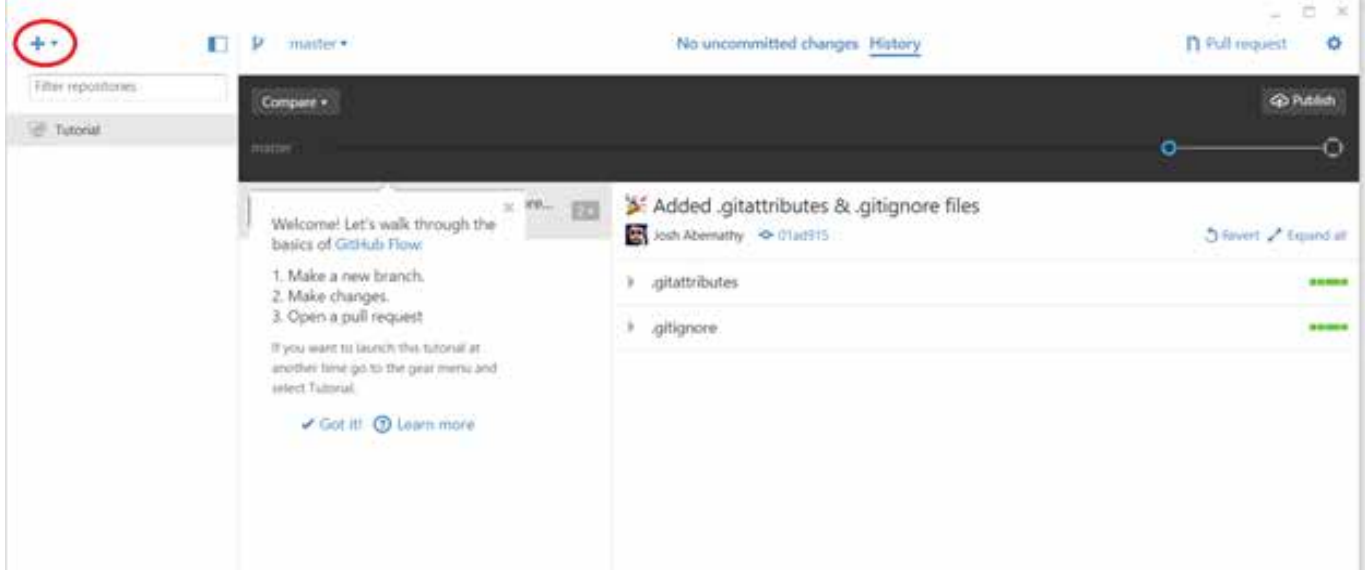


FIGURE 3.2
GitHub: Select drop down

4. Select the "Clone" icon at the top-left of the screen, then select the "Hello-World" repository.

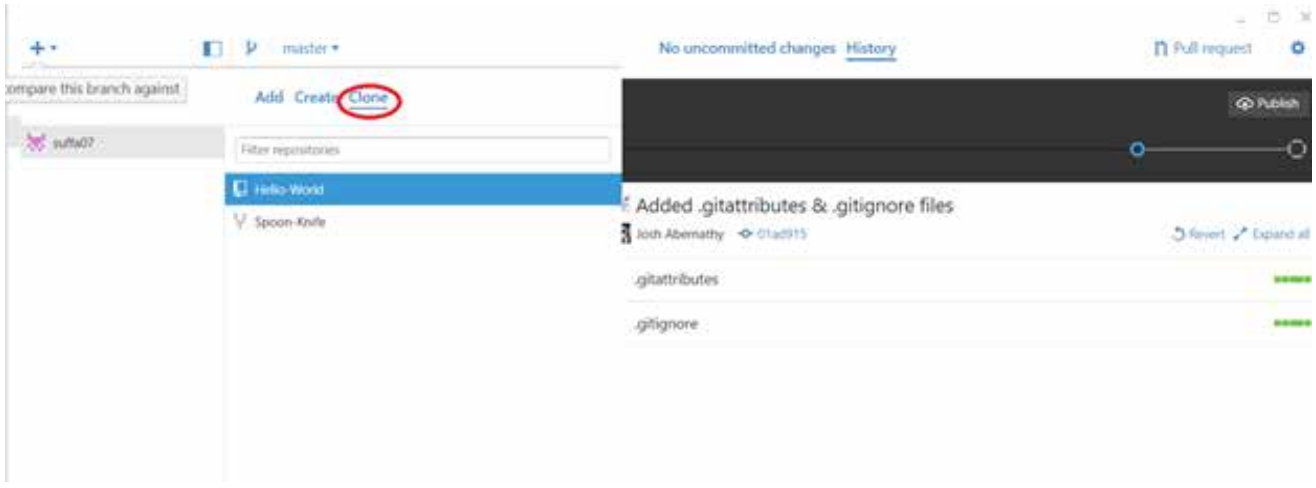


FIGURE 3.3
GitHub: Select clone Hello-World repository

5. Now, scroll down to the bottom and select the “check mark + Clone Hello-World” link (Note: this menu also allows us to add or create repositories).

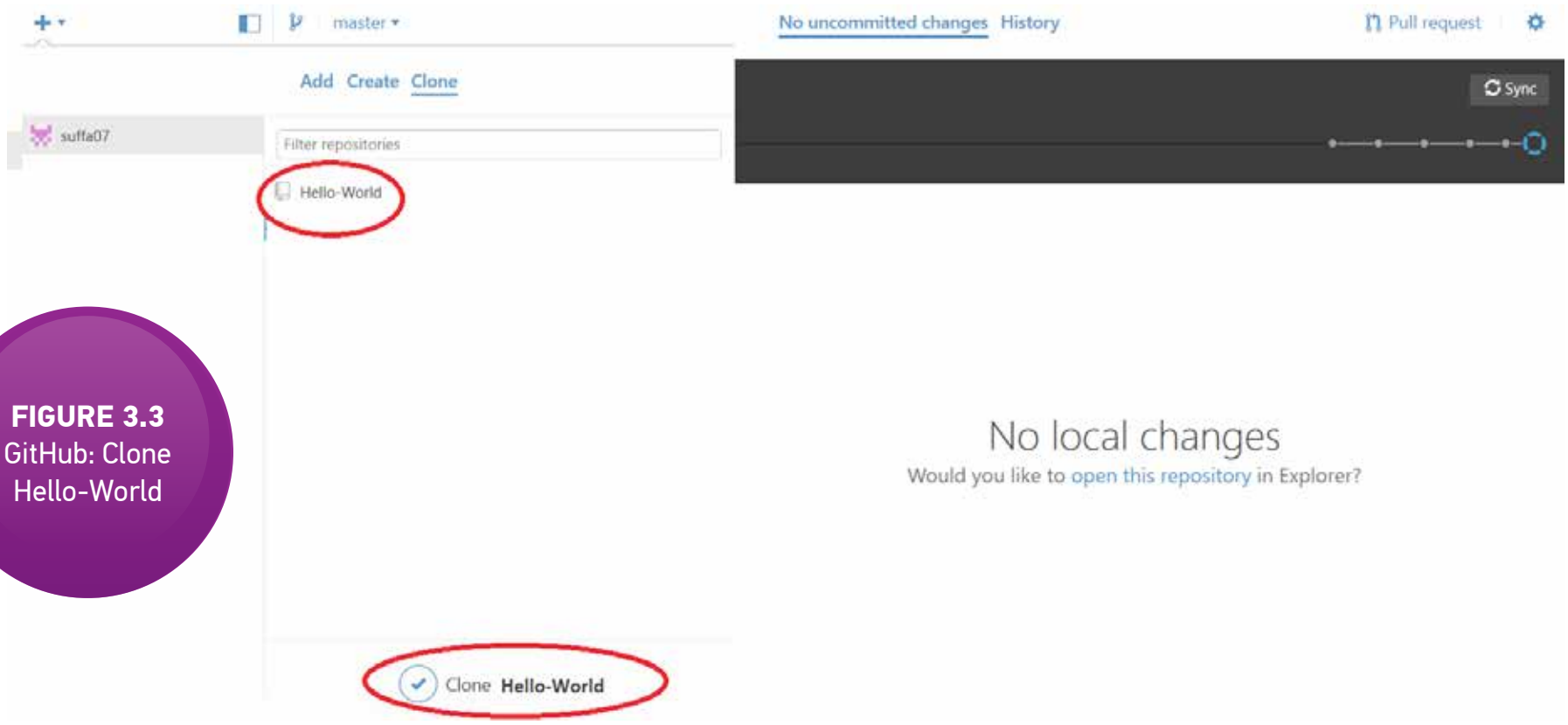


FIGURE 3.3
GitHub: Clone Hello-World

6. Select c:\Users\User Name\Documents\GitHub folder to clone into.

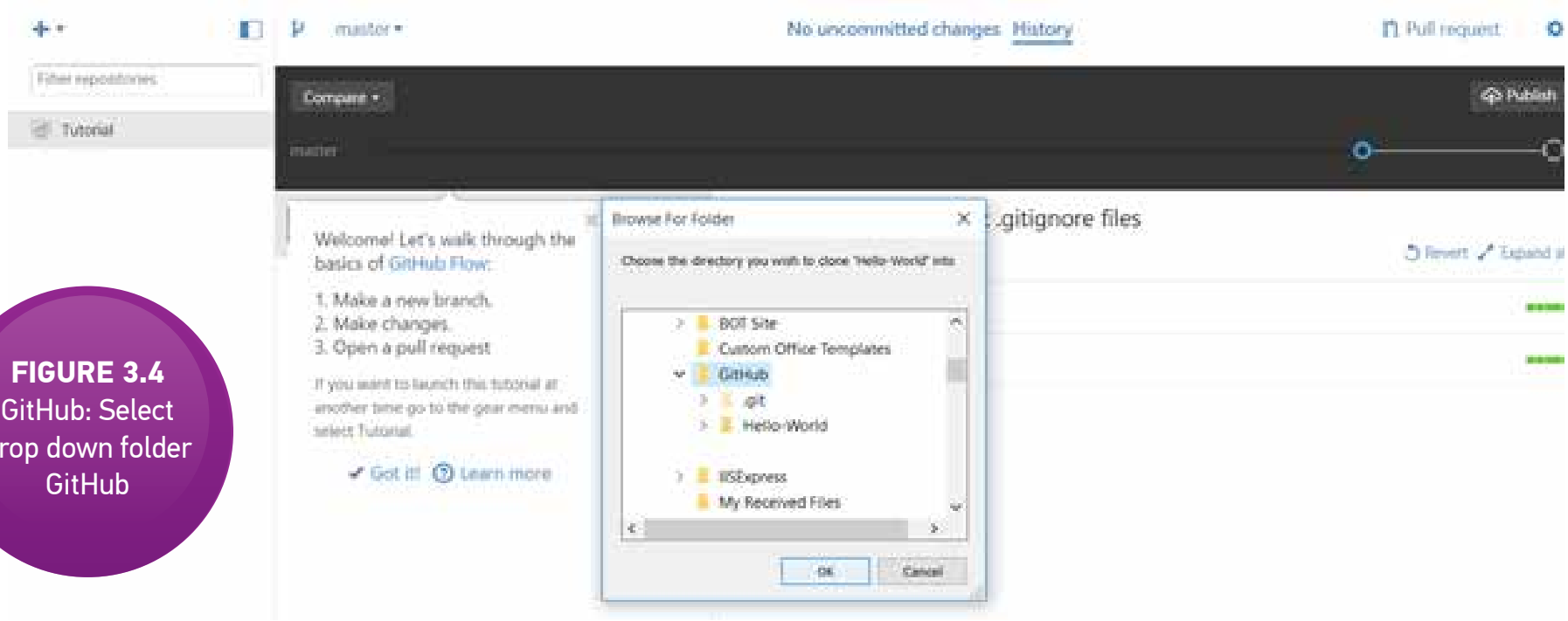


FIGURE 3.4
GitHub: Select drop down folder GitHub

7. Your repository’s contents will be displayed on the GitGub GUI, in History view by default.

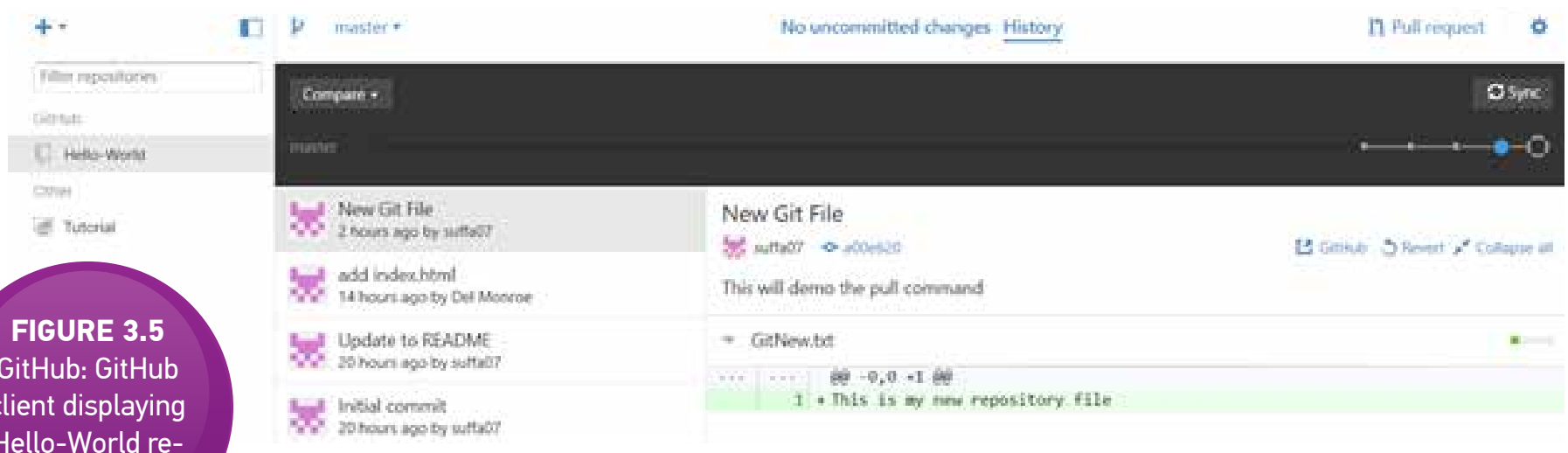


FIGURE 3.5
GitHub: GitHub client displaying Hello-World repository

8. The left of the screen lists your files, and the middle displays the contents of the file selected in the list (see Figure 3.5)

9. At the top of the screen, select No uncommitted changes. We can see that there are no committed changes. Let's make a change to a file.

10.

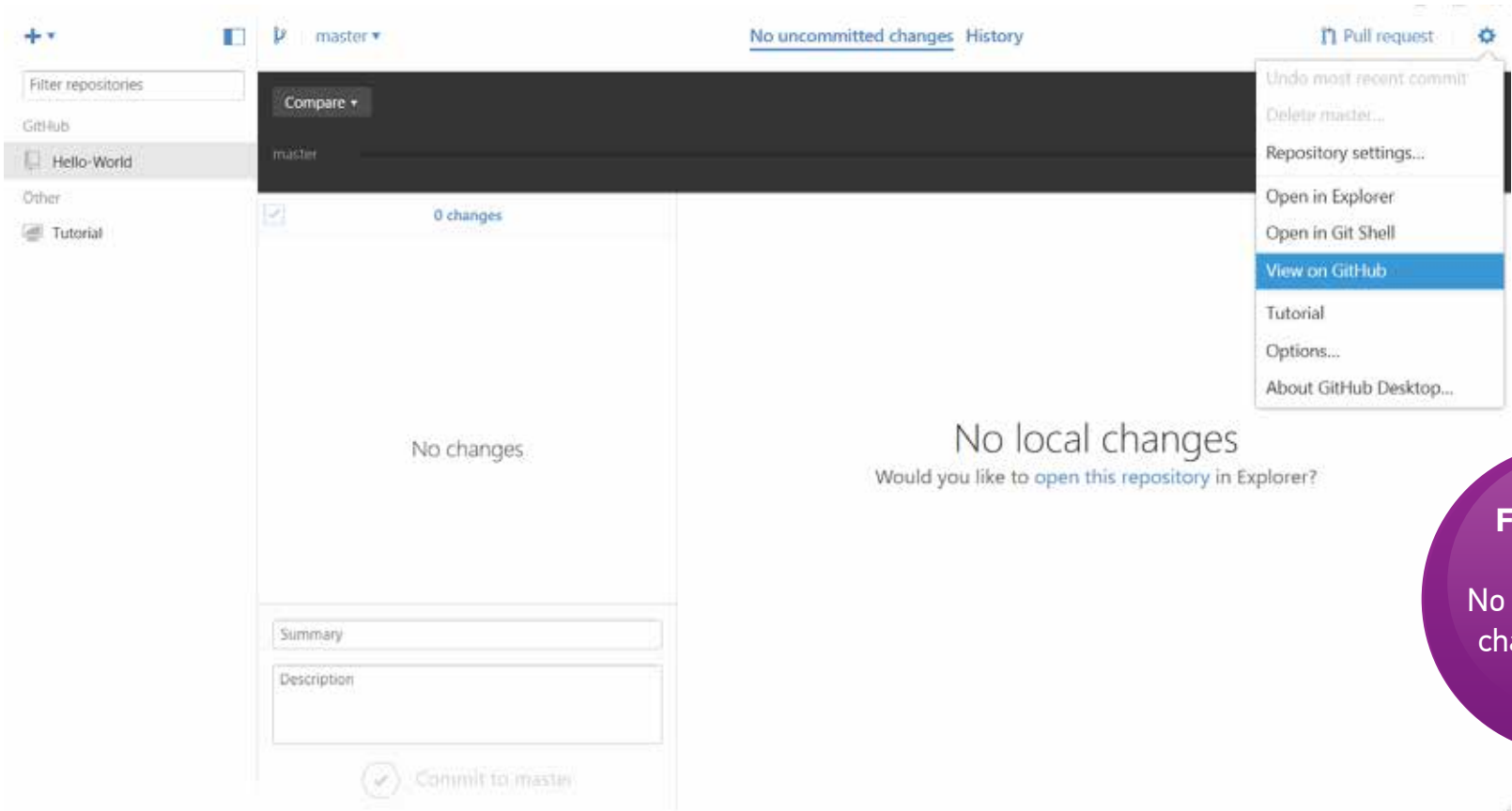


FIGURE 3.6
GitHub:
No uncommitted changes status message

11. Use Windows Explorer to open the index.html file in your c:\Users\User Name\Documents\GitHub\Hello-World repository with notepad

12. Change the <title> tag name to Hello World: <title> Hello World</title>, then save and close the index.html file.

13. Once the focus is back on the GitHub GUI, you will notice that the name at the top of the screen changes to "1 uncommitted change", and the index.html file is listed to the left with a green and red bar beside it.

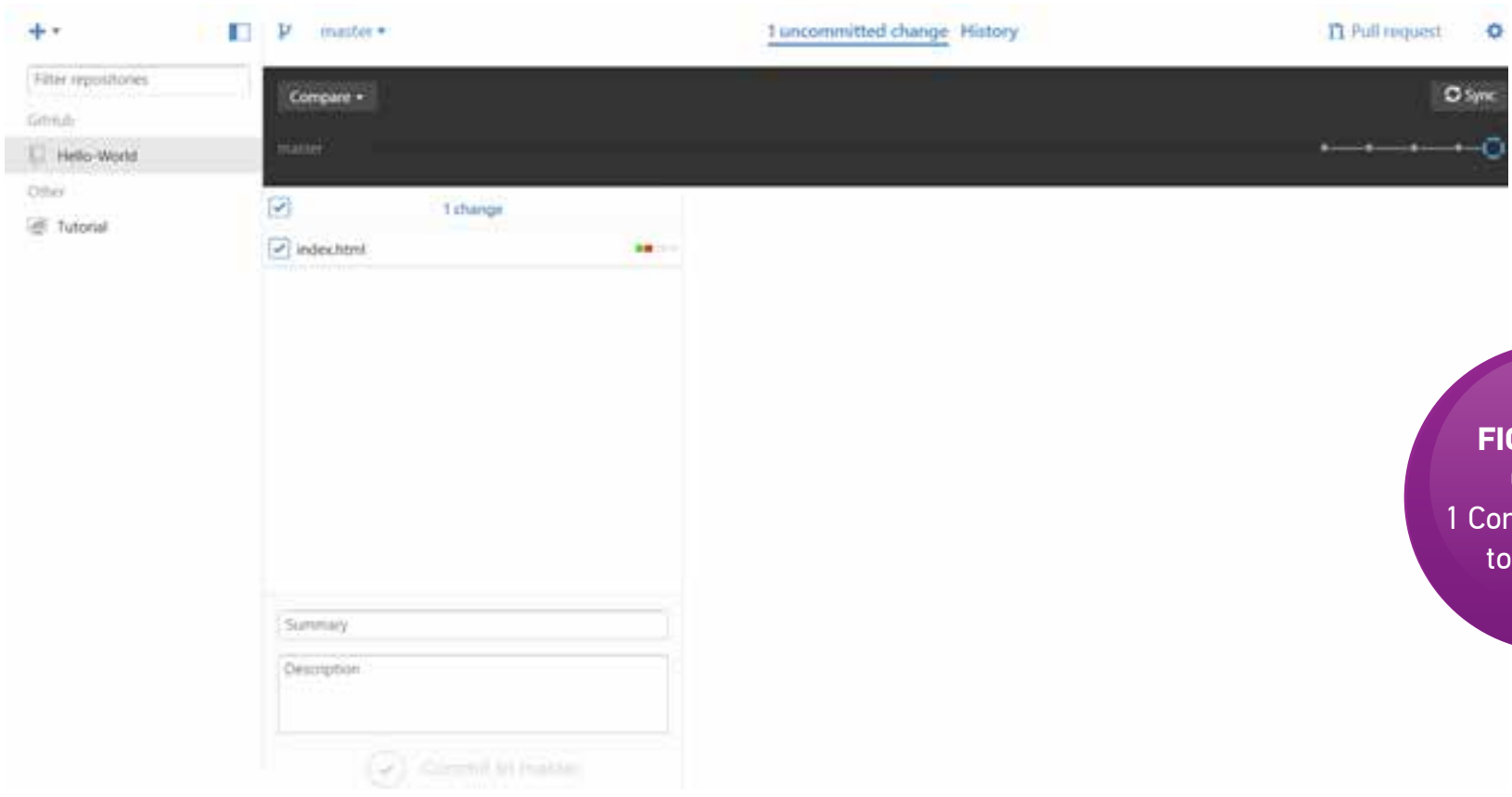


FIGURE 3.7
GitHub:
1 Commit change to be made

14. If we click on this index.html file listed next to the green and red bar, we will see the contents in the middle of the screen that highlights our original content in red, and highlights our modified content in green.

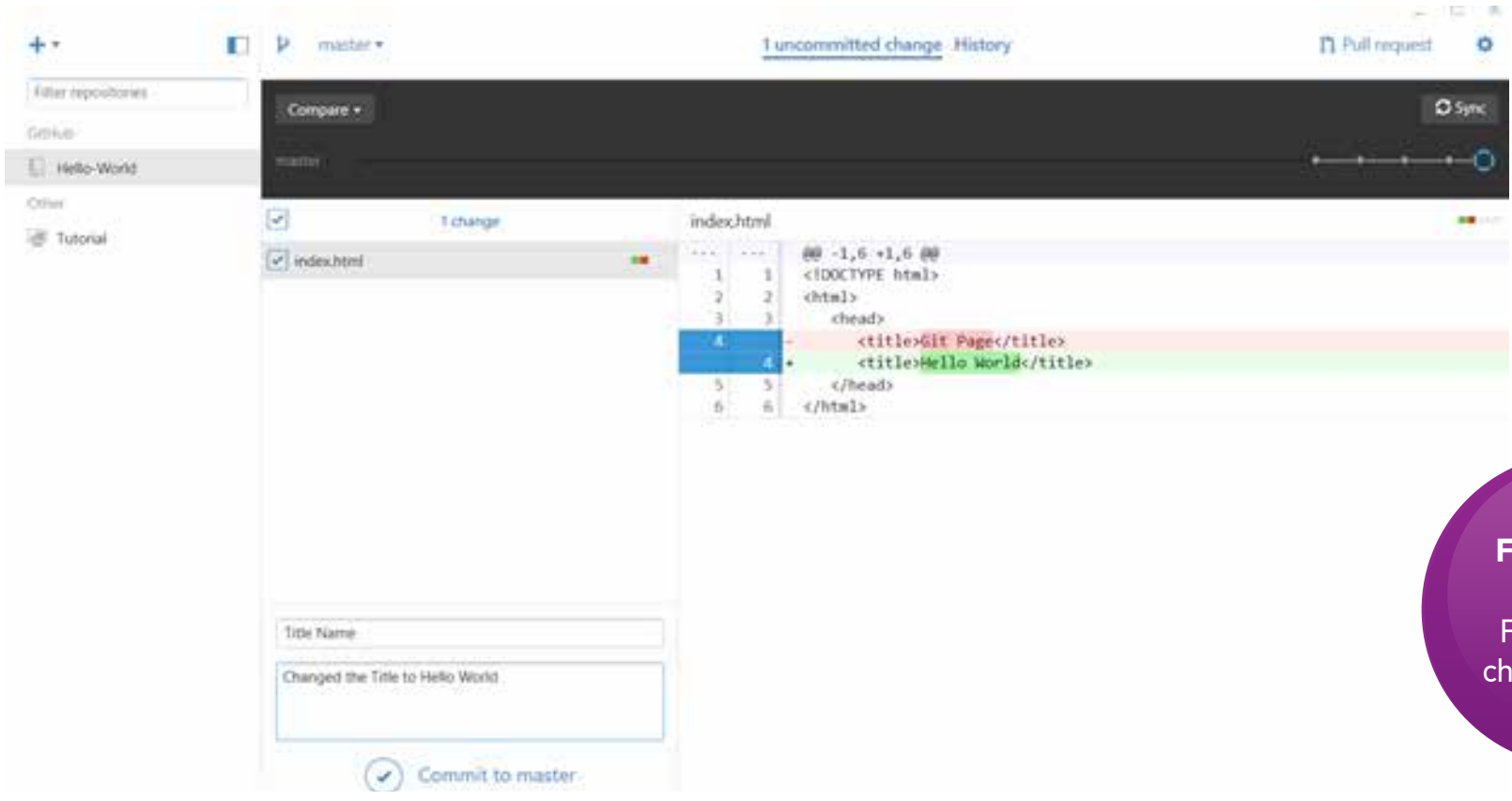


FIGURE 3.8
GitHub: File commit change details

15. To commit this change, we simply fill in a summary and description below the index.html file, and select the check mark next to the "Commit to master" label.

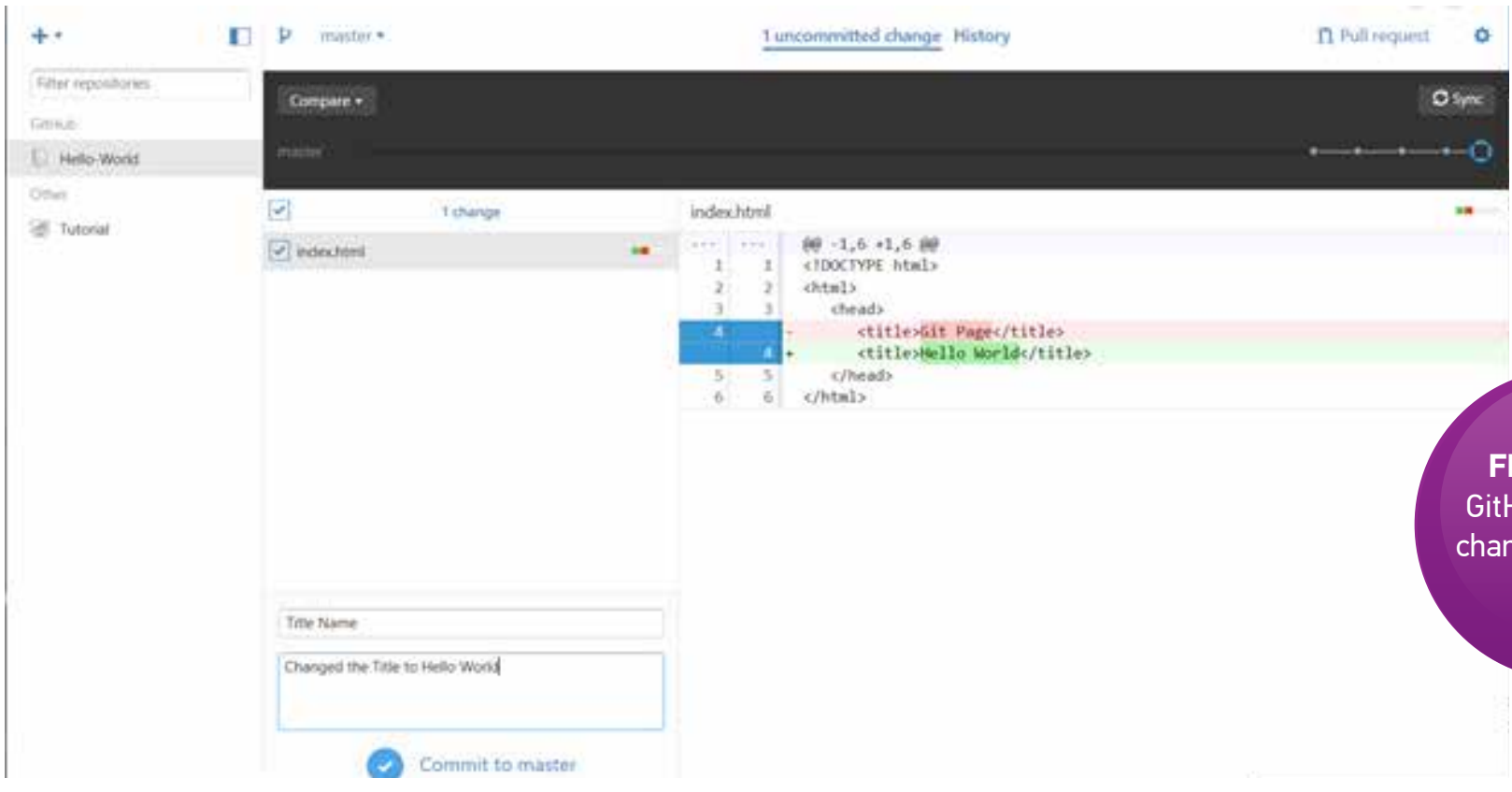


FIGURE 3.9
GitHub: Commit changes to index.html

16. To push this change to our master repository, we simply click the "Sync" button at the top-right of the screen (Note: the Sync button executes to functions, it pushes and pulls files at the same time between the master repository and your local machine cloned repository).

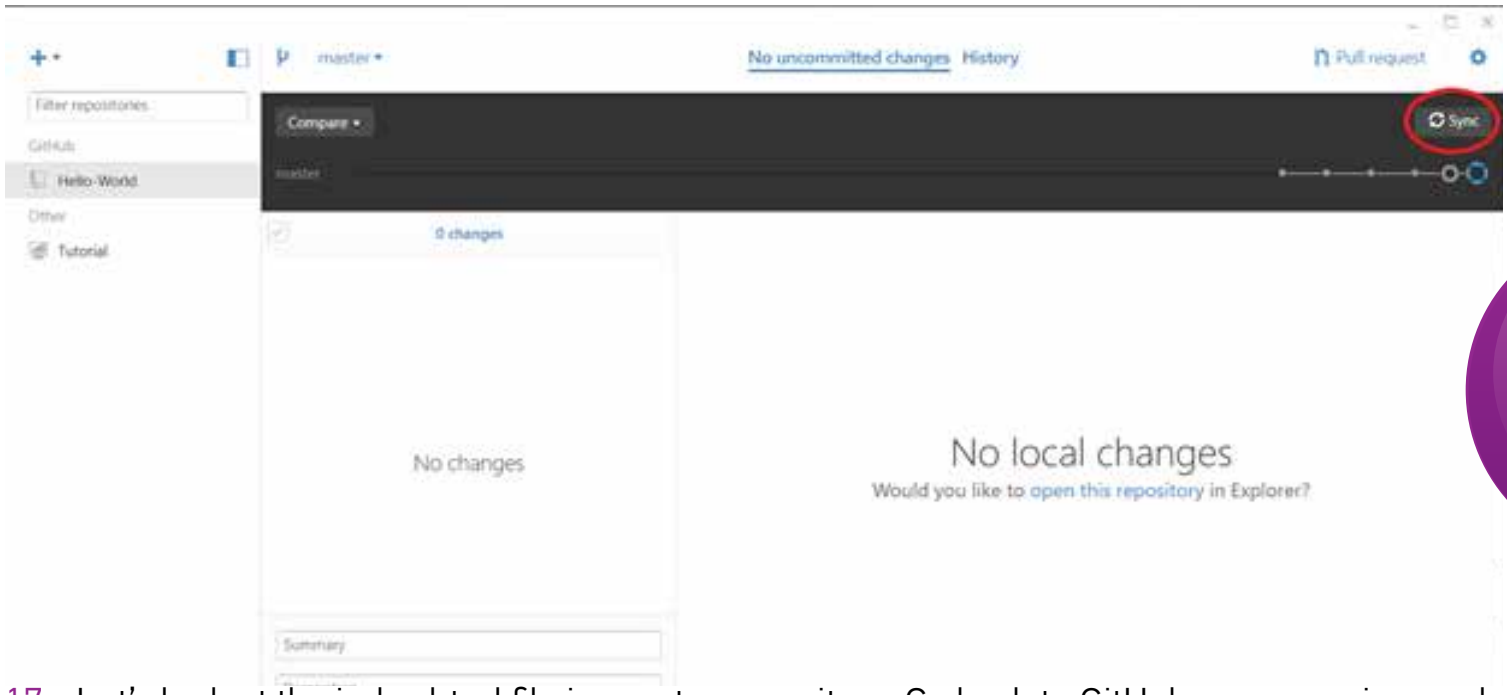


FIGURE 3.10
GitHub: push the index.html modified file to the master repository

17. Let's look at the index.html file in master repository. Go back to GitHub.com page in your browser, and select the Hello-World directory.

18. Click the index.html file. When it opens you will see the modified file just pushed to the master repository.

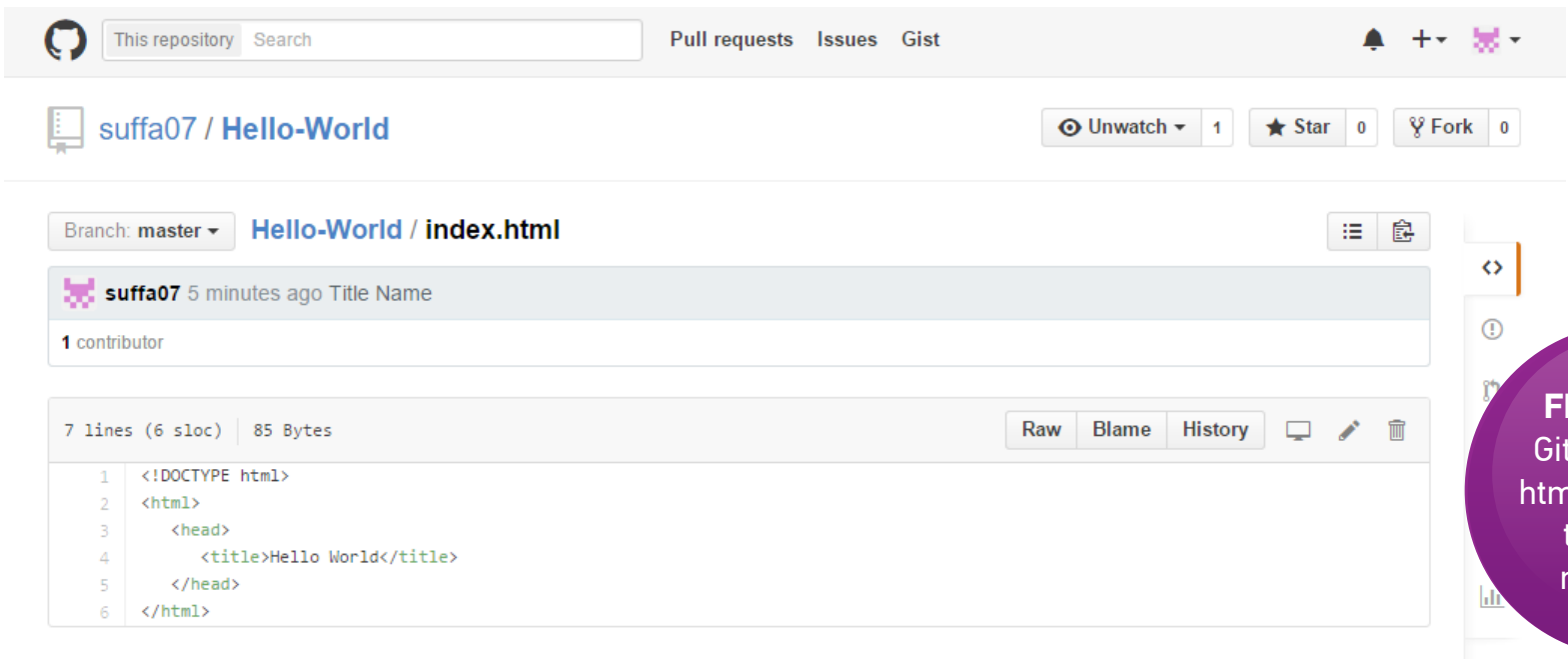


FIGURE 4.0
GitHub: index.html file pushed to master repository

WHAT ARE THE SOCIAL ASPECTS TO GITHUB:

Github is used for code sharing and a publishing service for programmers. It's basically a social networking site for programmers. The Git feature that really sheds light on the social aspect is forking. Forking is simply making a copy/clone of the repository. Say a user decided to make modifications or a bug fix to someone's project. They can fork the project, as done in the above examples. And, with Git, a fork can have multiple copies, entirely independent of one another, giving a user complete control over each particular fork, allowing them to merge, delete, or make modification ridiculously fast. And once the user proposes the changes to the original owner, he/she can now decide whether they want to pull the changes into the original repository (also called the upstream) to apply a modification or fix to the original project.

This is significantly beneficial to open source development, or a development team that is on a tight deadline; it's helpful in that it streamlines the creative process of all involved parties and that it limits the probability of introducing bugs into the master code base. Additionally, despite the physical locations of the members of a particular development team, they can all work on modifying the code base at similar times. They can be in different parts of a city, or in different geographical worldwide locations.

As long as there is an internet connection available, the social networking aspect of people in varying physical locations coming together to exchange ideas is a common reality with GitHub. The ability of multiple users cloning the original repository, and working in tandem, vicariously, through the cloud is what makes GitHub so powerful, and one of the more ubiquitous version control systems out here today. This article just scratches the surface of GitHub, affording you just a taste. I encourage you to sign-up for a GitHub account today to really sink your teeth into the power of Git.