



## Assignment#1

### HTML + VCS (Git, Github) using Command Line

<b>Program Name</b>	BSCS
<b>Course Code</b>	CSSE3143
<b>Course Name</b>	Web Application Development
<b>Course Instructor</b>	Mohammad Talha
<b>Time Allowed</b>	Until 31th Oct, 2018
<b>Assessment</b>	HTML + VCS (Git & Github), CMD
<b>Instructions</b>	<ul style="list-style-type: none"> <li>• Email at <a href="mailto:mohammad.talha@ucp.edu.pk">mohammad.talha@ucp.edu.pk</a> with subject “WAD-H Assignment#1”.</li> <li>• Submit the soft copy of your solution on or before the due date. Late submission will result in some penalty.</li> <li>• Always keep a backup of your solution till it is graded.</li> </ul>
<b>Conditions</b>	This is an individual assignment
<b>Total marks</b>	10 + 20 + 20 = 50
<b>Academic Honesty Policy</b>	Academic dishonesty will not be tolerated. Academic dishonesty includes cheating, plagiarism (copying) or any other attempt to gain an academic advantage in a dishonest or unfair manner.

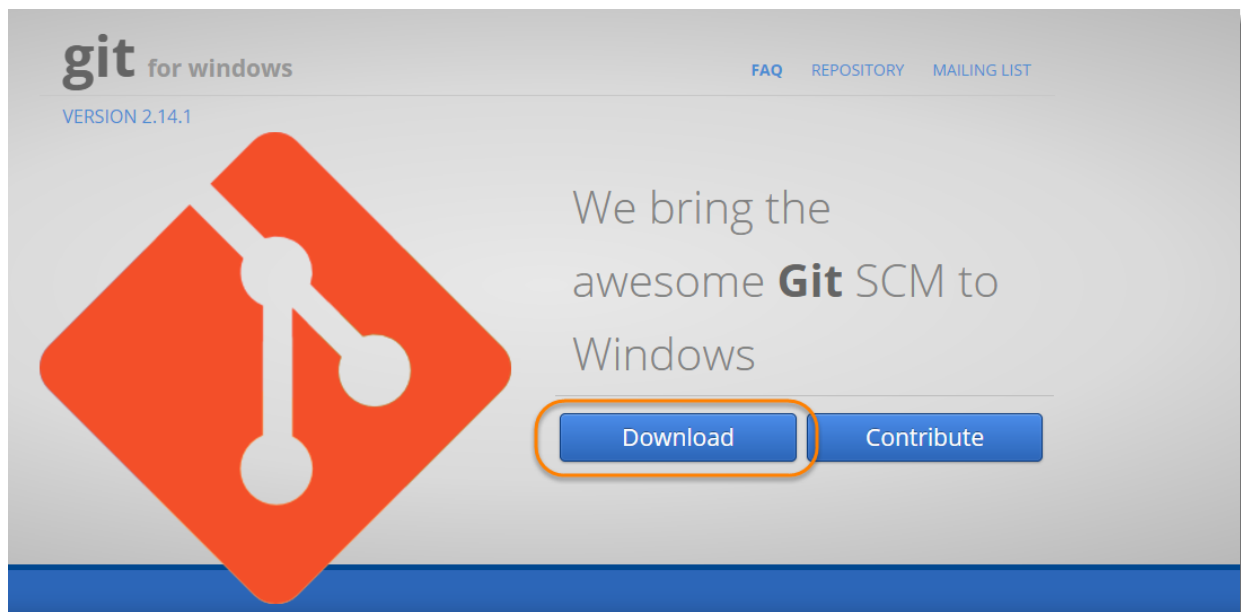
## Perform the following tasks

- Create a folder with HTML file
- Code in HTML Table
- Create a resume design and layout with the help of HTML Table
- HTML filename should be index.html
- Create a images folder if you have used any image in the file
- Setup a Git on your system (laptop/computer)
- Create Github account
- Create a git repo
- Configure your Github account on your system (laptop/computer)
- Clone your git repo on your system
- Commit and push your resume HTML files to your git repo

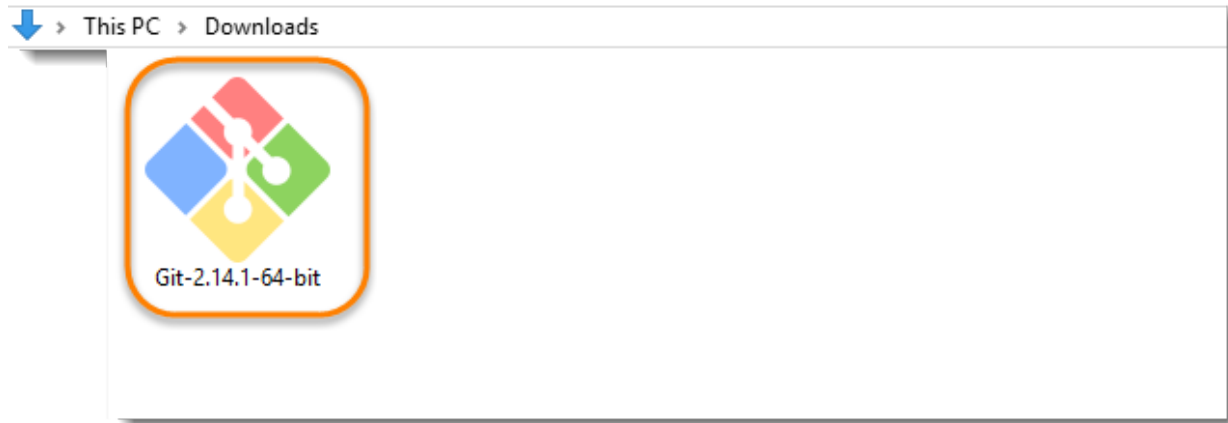
All these tasks instruction are discussed below in details.

## Steps to Install Git on Windows

1) Download the latest [Git for Windows](#).



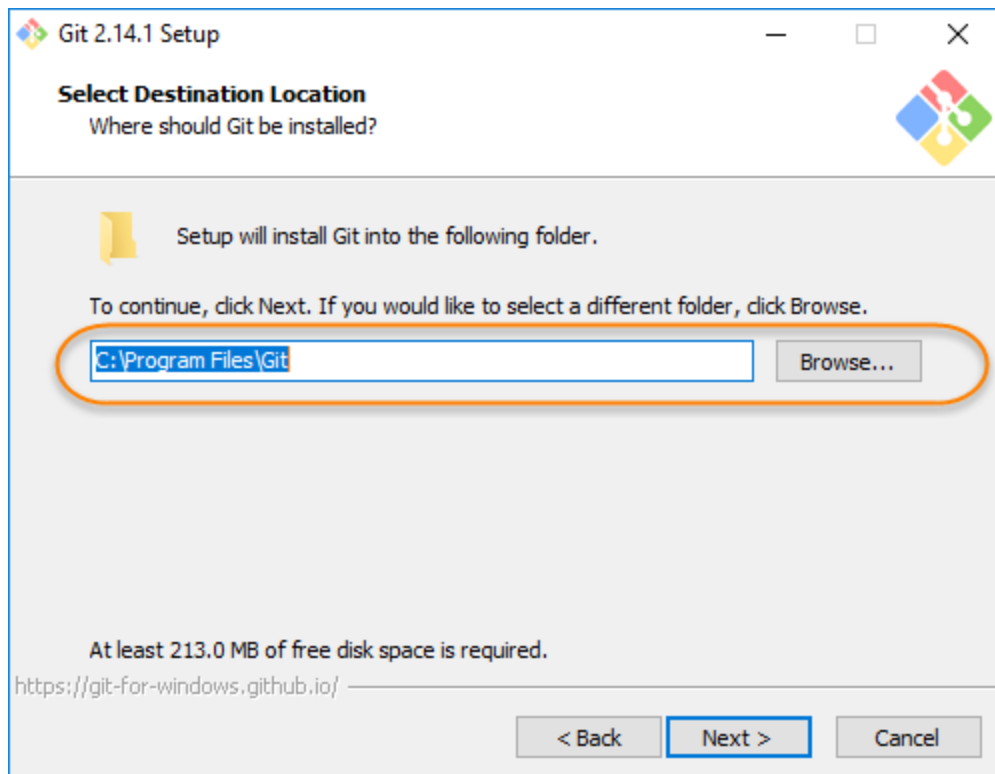
2) Go to the folder where new downloads gets store, at my machine by default folder is **Download** folder. **Double click** on the installer. The installer gets save on the machine as per the Windows OS configuration. My machine is **64 bits**.



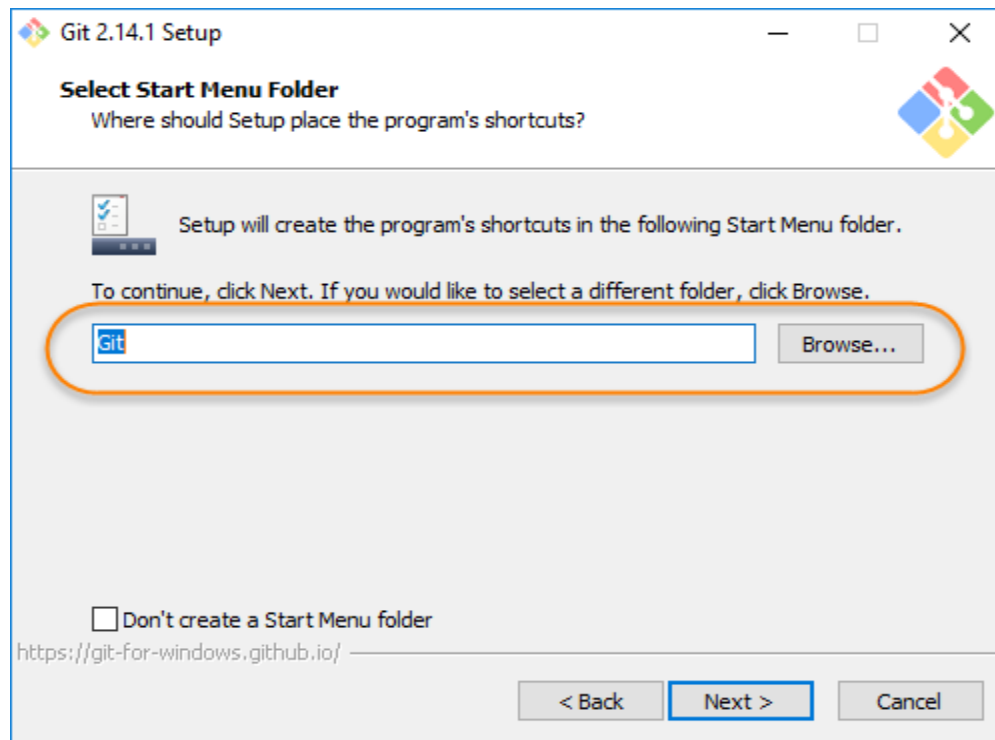
*Note: When you've successfully started the installer, you should see the Git Setup wizard screen. Follow the Next and Finish prompts to complete the installation. The default options are pretty sensible for most users.*

*Note: At the time of writing the tutorial on 9th Sep '17, the latest version is **Git-2.14.1**.*

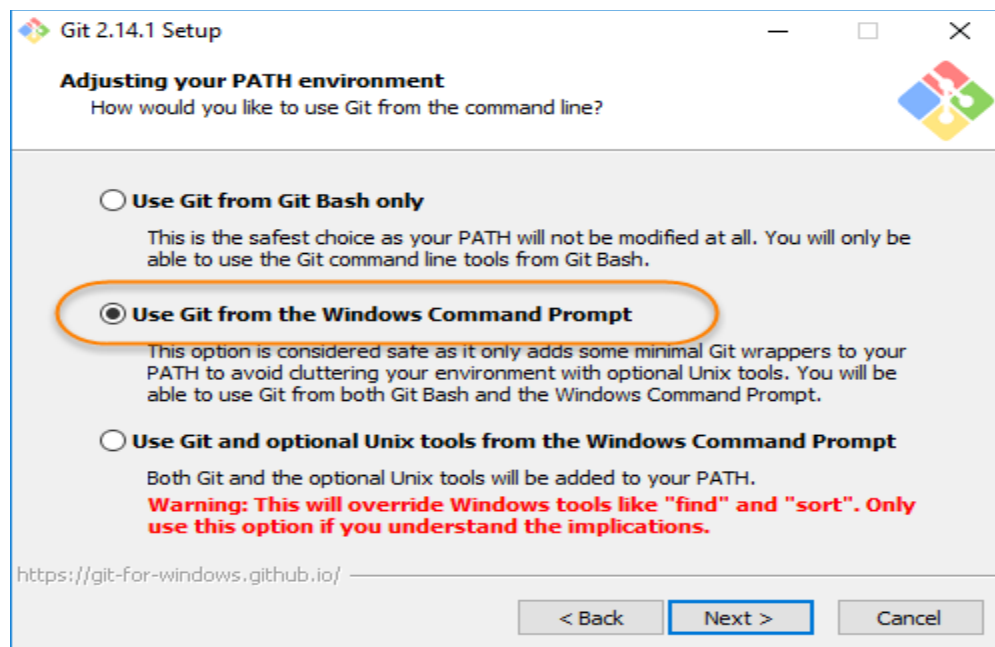
3) You may like to keep the installation to another folder, so here is the chance to do so. I just want to keep it in the suggested default folder in my **Program Files\Git**.



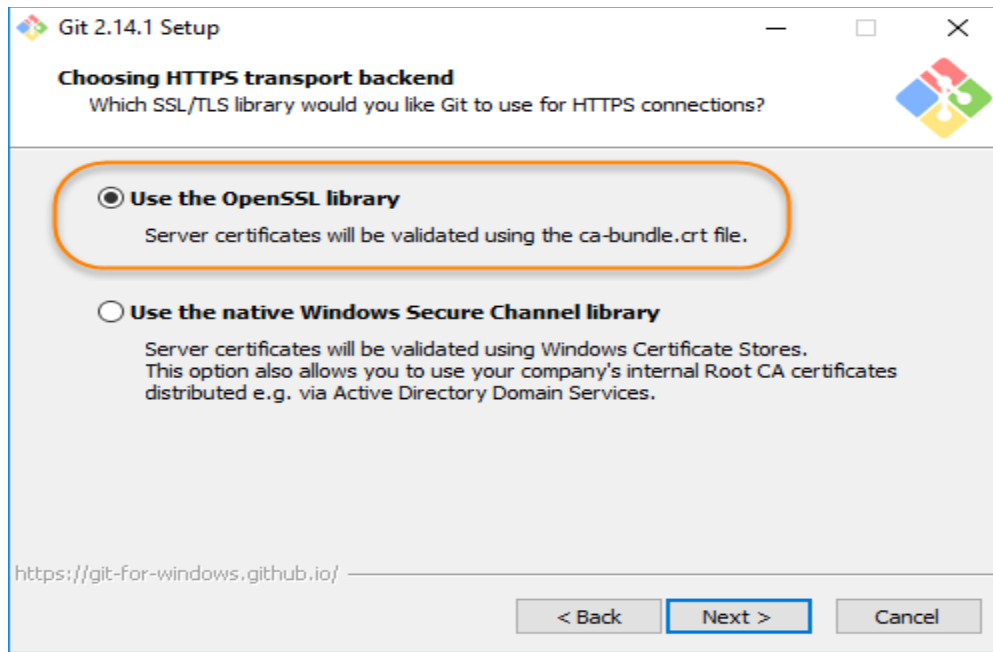
4) This is the option to store the *shortcut of the Git* under the **Program Menu**.



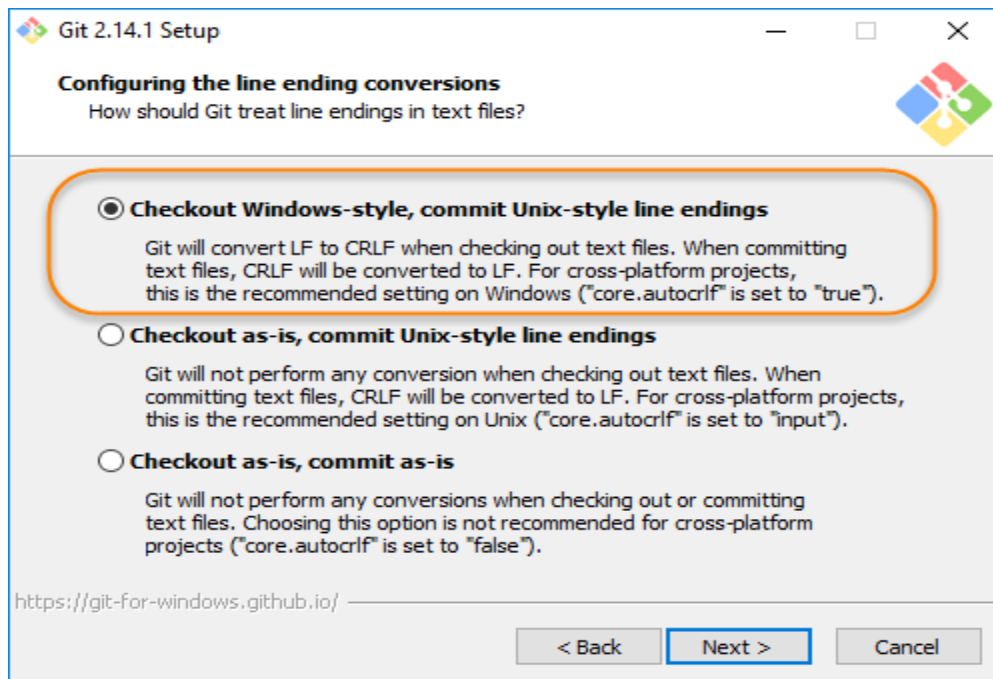
5) This is asking your choice that whether you like to Git from the **Windows Command Prompt** or you like to use some other program like **Git Bash**. As of now just select the *Windows Cmd* for simplicity of the tutorial, later we will cover *Git Bash* and other as well.



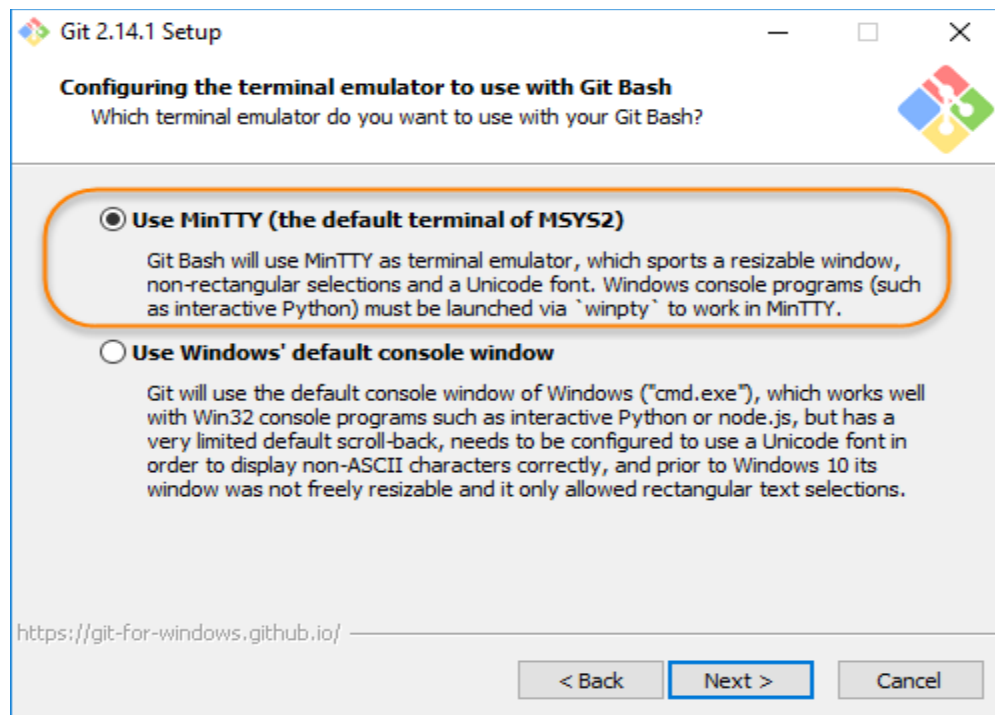
6) If you have *PuTTY/TortoiseSVN* installed, you may see this screen, otherwise just ignore this. Regardless, use *OpenSSL* to make things easy.



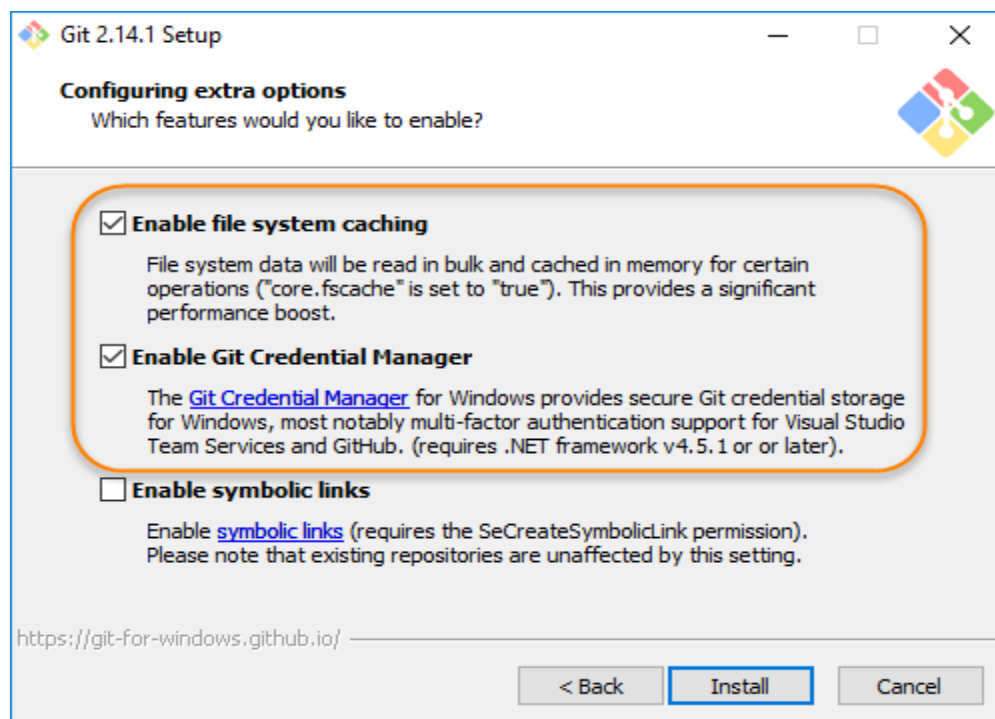
7) Here, we recommend to choose the option of *Checkout Windows-style, commit Unix-style line endings*. Select next once you have done this.



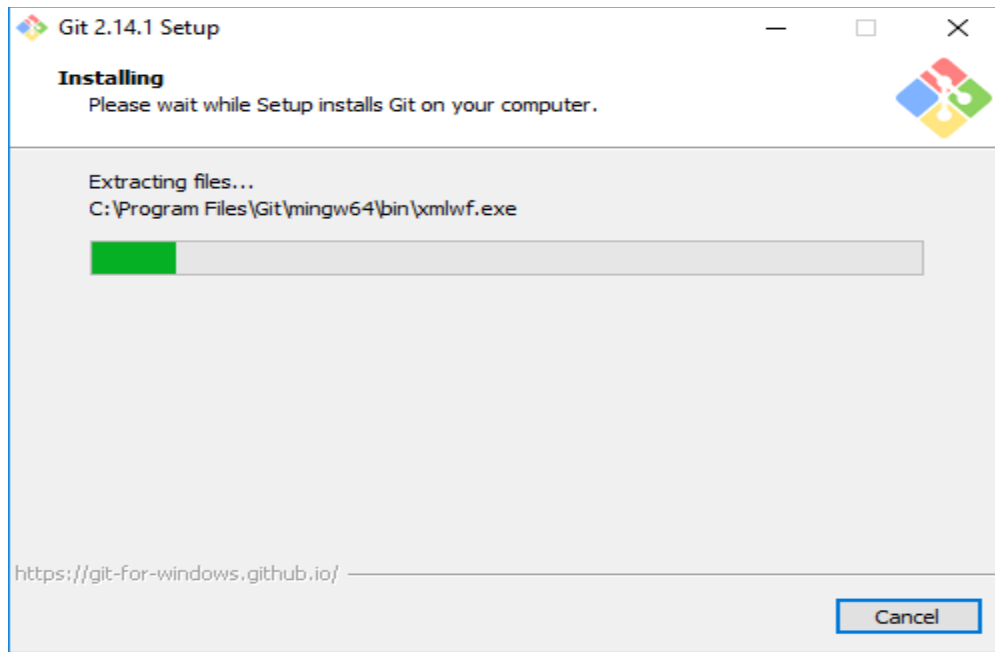
8) Again, just go with default selection and move forward.



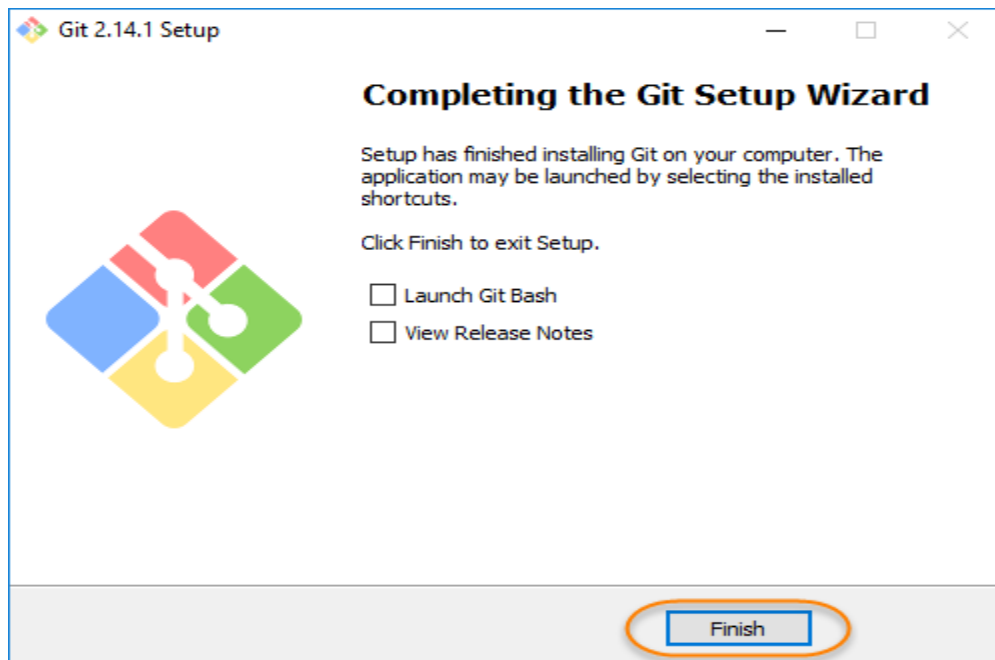
9) Just go with default selections, as we will cover the details in later advance chapter.



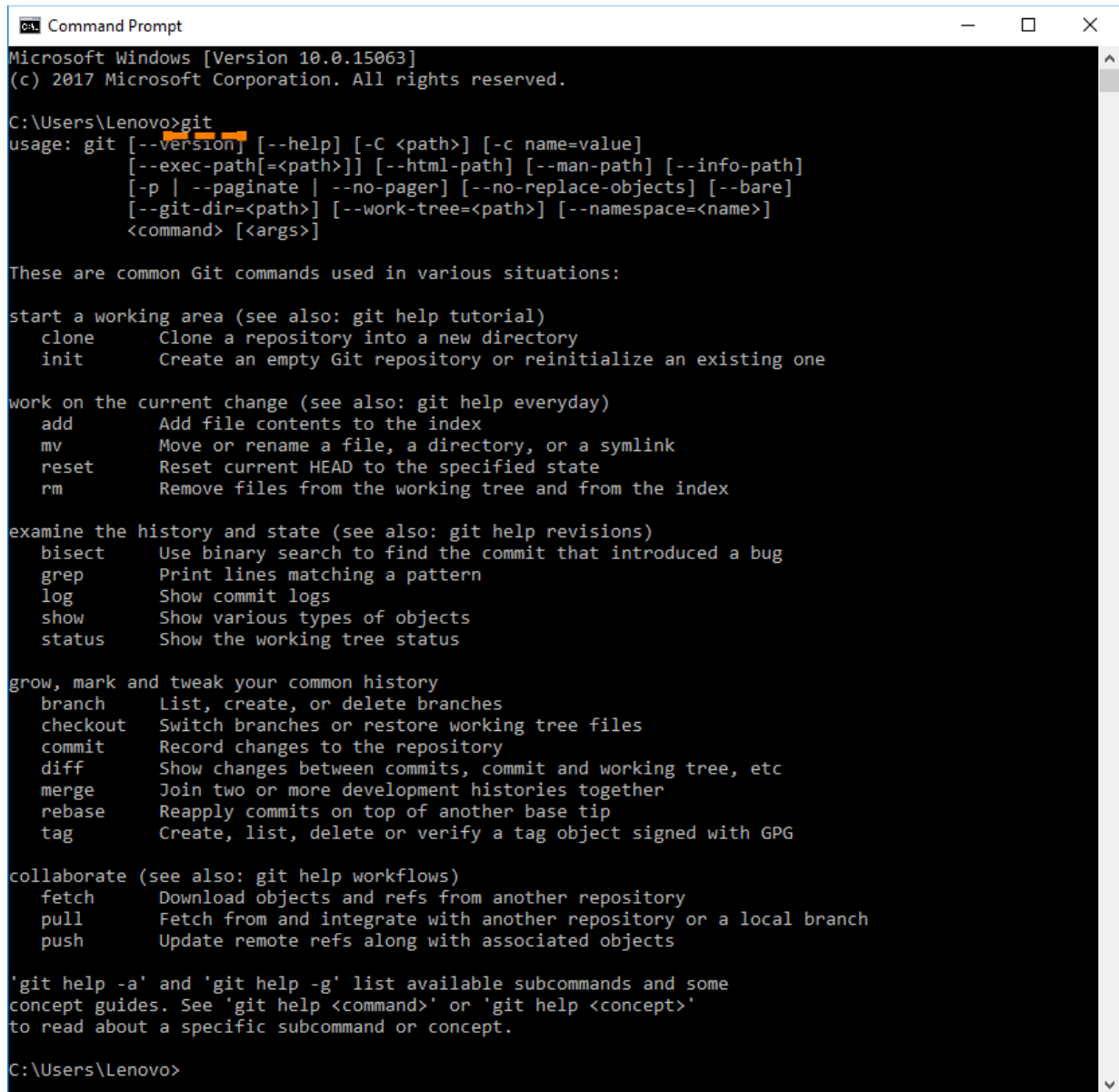
10) Now, it's all done. This will just take few minutes to complete the installation as per your machine speed.



11) Once done, just click on Finish button.



12) Let's just verify if the installation went well for Git. Go to *cmd* and type *git* and press *enter*. you should get the following output on the screen.

A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the output of the 'git' command. The first line is the Microsoft Windows version and copyright information. The second line shows the current directory as 'C:\Users\Lenovo>'. The third line shows the 'git' command being executed. The output displays the usage of 'git' with various options like --version, --help, --exec-path, etc. It then lists common Git commands grouped into categories: starting a working area, working on the current change, examining history and state, growing and tweaking history, and collaborating. At the bottom, it mentions 'git help -a' and 'git help -g' for more information.

```
C:\Users\Lenovo>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>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

C:\Users\Lenovo>
```

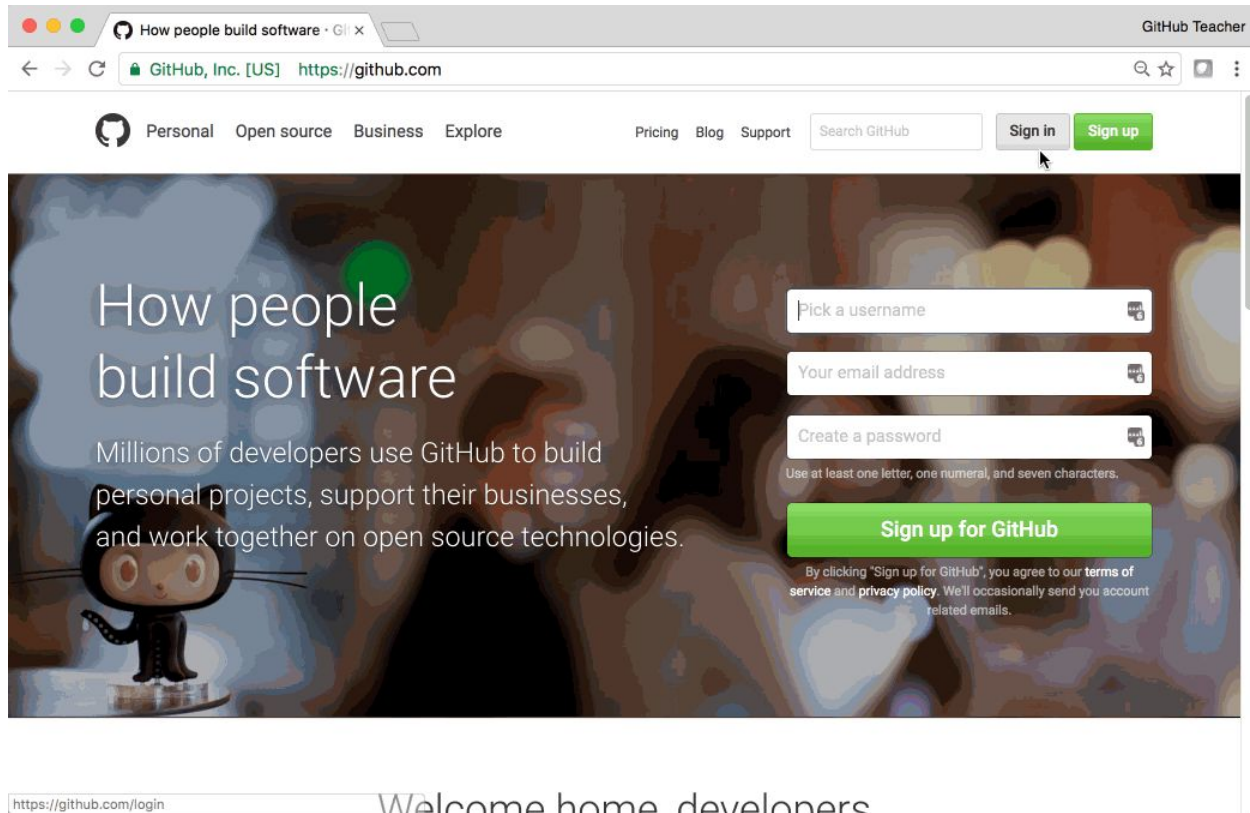
The cmd window will display different options and commands you can try with git. Just get little warm up and use the following commands and observe the outputs for your understanding:

- *git --version*
- *git --help*



## Create an account on GitHub.com

If you already have a Github.com account you are ready to get started. Otherwise, you can set up your free account by following these steps:



The screenshot shows the GitHub homepage in a web browser. The browser's address bar displays "GitHub, Inc. [US] https://github.com". The page features a navigation bar with links for "Personal", "Open source", "Business", "Explore", "Pricing", "Blog", and "Support". A search bar labeled "Search GitHub" is present, along with "Sign in" and "Sign up" buttons. The main content area has a dark background with the text "How people build software" and "Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies." Below this, there is a sign-up form with three input fields: "Pick a username", "Your email address", and "Create a password". A green "Sign up for GitHub" button is positioned below the form. A small note indicates that the password must contain at least one letter, one numeral, and seven characters. At the bottom of the form, a disclaimer states: "By clicking 'Sign up for GitHub', you agree to our terms of service and privacy policy. We'll occasionally send you account related emails." The footer of the page includes the URL "https://github.com/login" and the text "Welcome home developers".

1. Access [GitHub.com](https://github.com) and click and enter a username, email address, and password in the supplied field and click Sign up for GitHub.
2. Select the Unlimited public repositories for free option.
3. You will receive a verification email at the address provided.
4. Click the emailed link to complete the verification process.

## Create a Repository


A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a *README*, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository. *It also offers other common options such as a license file.*

Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.


### To create a new repository

1. In the upper right corner, next to your avatar or identicon, click
1. and then select **New repository**.
2. Name your repository hello-world.
3. Write a short description.
4. Select **Initialize this repository with a README**.

PUBLIC

 hubot

/


hello-world 


Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

**Description** (optional)

Just another repository

---


☒  **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 allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None**

Add a license: **None** 

---

Create repository

Click **Create repository**. 

## Create a Branch

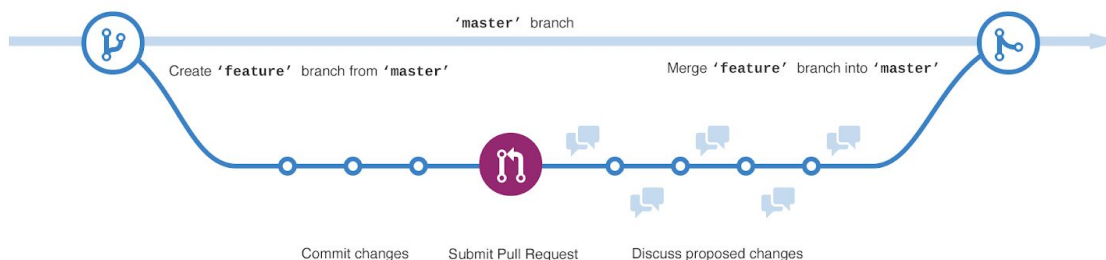
**Branching** is the way to work on different versions of a repository at one time.

By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.

When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

This diagram shows:

- The master branch
- A new branch called feature (because we're doing 'feature work' on this branch)
- The journey that feature takes before it's merged into master



Have you ever saved different versions of a file? Something like:

- story.txt
- story-joe-edit.txt
- story-joe-edit-reviewed.txt

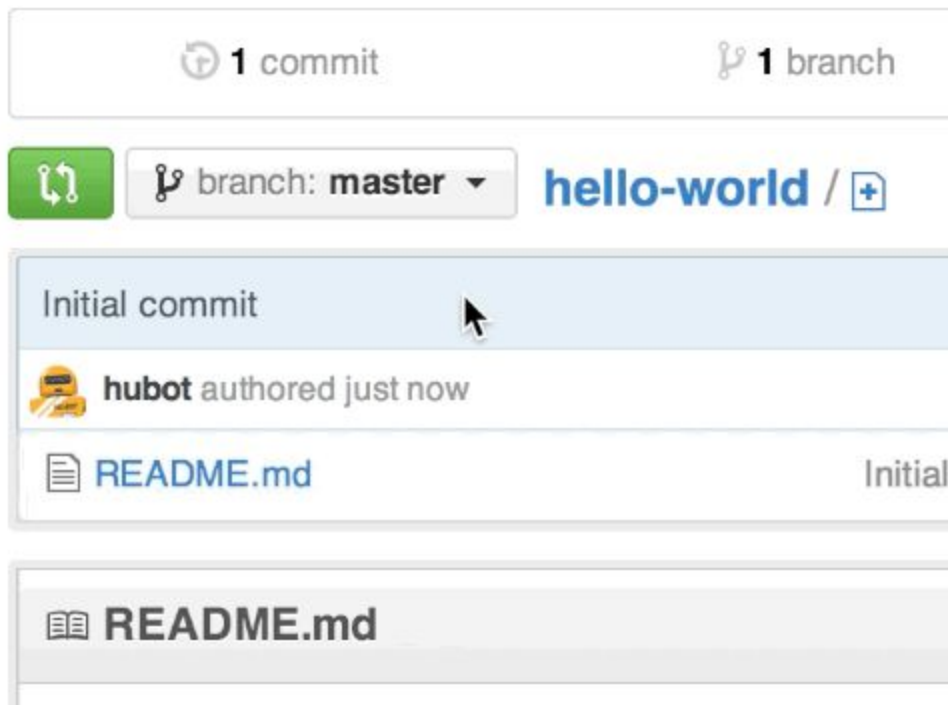
Branches accomplish similar goals in GitHub repositories.

Here at GitHub, our developers, writers, and designers use branches for keeping bug fixes and feature work separate from our master (production) branch. When a change is ready, they merge their branch into master.

### To create a new branch

1. Go to your new repository hello-world.
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, readme-edits, into the new branch text box.
4. Select the blue **Create branch** box or hit "Enter" on your keyboard.

## Just another repository — Edit



Now you have two branches, master and readme-edits. They look exactly the same, but not for long! Next we'll add our changes to the new branch.

## Make and commit changes

Bravo! Now, you're on the code view for your readme-edits branch, which is a copy of master. Let's make some edits.

On GitHub, saved changes are called *commits*. Each commit has an associated *commit message*, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.

### Make and commit changes

1. Click the README.md file.
2. Click the
  1. pencil icon in the upper right corner of the file view to edit.
  2. In the editor, write a bit about yourself.
  3. Write a commit message that describes your changes.
  4. Click **Commit changes** button.

hubot / hello-world

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

hello-world / README.md or cancel

Edit file Preview changes Spaces 2 Soft wrap

```
1 # hello-world
2
3 Hi Humans!
4
5 Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
6 I've had tacos on the moon and find them far superior to Earth tacos.
7
```

Commit changes

Finish README

And mention moon tacos

☒ Commit directly to the `readme-edits` branch

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

These changes will be made to just the README file on your readme-edits branch, so now this branch contains content that's different from master.

## Open a Pull Request

Nice edits! Now that you have changes in a branch off of master, you can open a *pull request*.

Pull Requests are the heart of collaboration on GitHub. When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show *diffs*, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

By using GitHub's [@mention system](#) in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

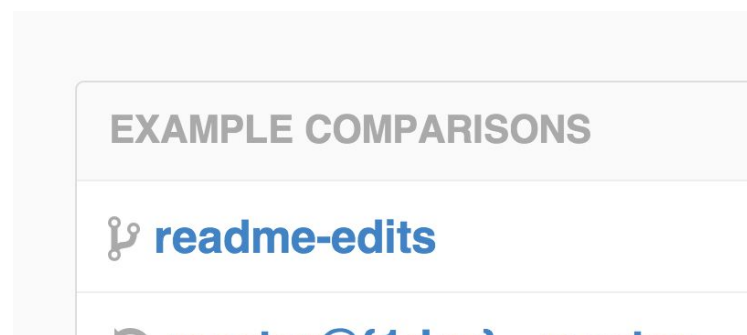
You can even open pull requests in your own repository and merge them yourself. It's a great way to learn the GitHub flow before working on larger projects.

### Open a Pull Request for changes to the README


**Pull Request** tab, then from the Pull Request page, click the green **New pull request** button.






In the **Example Comparisons** box, select the branch you made, readme-edits, to compare with master (the original).




Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.

 **1 commit**

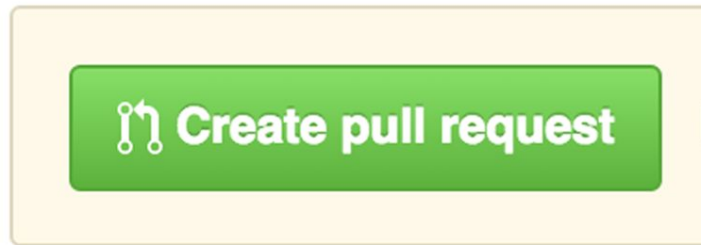
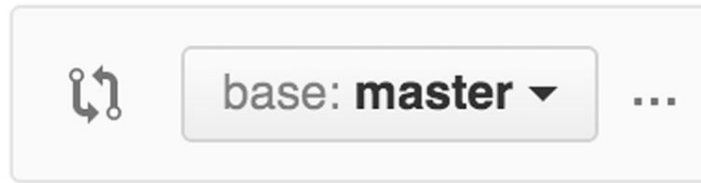
 **Commits on Oct 27, 2014**

  **hubot**

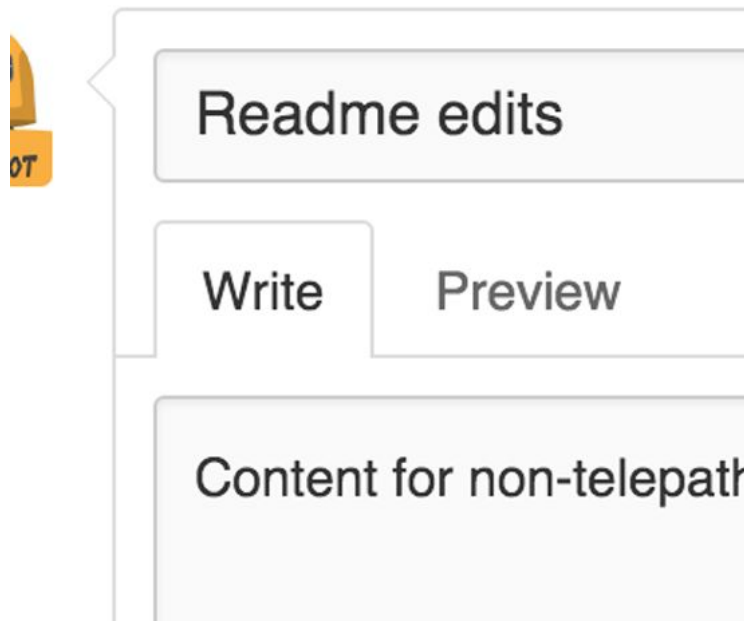
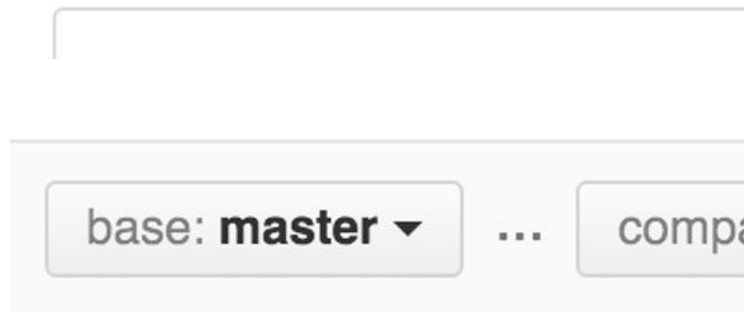
 **Showing 1 changed file with 1 addition**

2 README.md		
...	...	@@ -1,4 +1,4 @@
1	1	hello-world
2	2	=====
3	3	
4		-Just another r
	4	+Hubot here, I
		them far super

When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.



Give your pull request a title and write a brief description of your changes.




When you're done with your message, click **Create pull request**!





## Merge your Pull Request


In this final step, it's time to bring your changes together – merging your `readme-edits` branch into the master branch.

1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.




**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](#).



**Pull request successfully merged and closed**  
You're all set—the `readme-edits` branch can be safely deleted.

 **Delete branch**

### Celebrate!

By completing this tutorial, you've learned to create a project and make a pull request on GitHub! 🎉 🐙 ⚡

Here's what you accomplished in this tutorial:

- Created an open source repository
- Started and managed a new branch
- Changed a file and committed those changes to GitHub
- Opened and merged a Pull Request

# Command Line

## create a new repository

- create a new directory, open it and perform a
- `git init`
- to create a new git repository.

## checkout a repository

- create a working copy of a local repository by running the command
- `git clone /path/to/repository`
- when using a remote server, your command will be
- `git clone username@host:/path/to/repository`

## add & commit

- You can propose changes (add it to the **Index**) using
- `git add <filename>`
- `git add *`
- This is the first step in the basic git workflow. To actually commit these changes use
- `git commit -m "Commit message"`
- Now the file is committed to the **HEAD**, but not in your remote repository yet.

## Pushing changes

- Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute
- `git push origin master`
- Change *master* to whatever branch you want to push your changes to.
- If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with
- `git remote add origin <server>`
- Now you are able to push your changes to the selected remote server