



NYC DATA SCIENCE
ACADEMY

Introduction to Git & Github

Data Science Bootcamp

Outline

- ❖ **Set up Git and GitHub**

- ❖ **Introduction to Git**

- **Creating a Git Repository**
- **Manipulating files**

- ❖ **Introduction to GitHub**

- **Lightning Tour of Github**
- **Create a Remote Repository**

Pre-requisites

- ❖ Patience and willingness to keep learning
- ❖ Have a good text editor for editing plain text files. Good examples are:
 - Notepad++
 - TextWrangler
 - Sublime
 - Emacs
 - Vim
 - LightTable
- ❖ Syntax highlighting and side-by-side editing are also useful features.

Pre-requisites for Windows

- ❖ Notepad++: <https://notepad-plus-plus.org/download/>
 - ❖ The default is plain UTF-8 encoding, without a byte-order mark (BOM).

C:\Users\Andreas\Desktop\variant-duo\variant-duo.css - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

index.html

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
3 <head>
4   <meta http-equiv="content-type" content="text/html; charset=utf-8" />
5   <meta name="description" content="Your description goes here" />
6   <meta name="keywords" content="your,keywords,goes,here" />
7   <meta name="author" content="Your Name" />
8   <link rel="stylesheet" type="text/css" href="variant-duo.css" title="Variant Duo" media="screen,projection" />
9   <title>Variant Duo v1.0</title>
10 </head>
11 <body>
12   <div id="wrap">
13     <h1><a href="index.html">Variant Duo</a></h1>
14     <p class="slogan">Dual-column content template</p>
15
16     <div id="menu">
17       <ul class="menulinks">
18         <strong class="hide">Main menu:</strong>
19         <a class="menulink active" href="index.html">Home</a> <span class="hide">|</span>
20         <a class="menulink" href="index.html">Page 2</a> <span class="hide">|</span>
21         <a class="menulink" href="index.html">Page 3</a> <span class="hide">|</span>
22         <a class="menulink" href="index.html">Page 4</a> <span class="hide">|</span>
23         <a class="menulink" href="index.html">Page 5</a>
24       </ul>
25     </div>
26
27     
28
29     <div id="content">
30       <div class="left">
31         <h2>Introducing Variant Duo</h2>
32         <p>If you are looking for a really simple website template with a basic dual-column layout that is easy to get started with, then Variant Duo may be a good starting point. This template is completely free and may be used without any limitations or obligations. I kindly ask you to leave the design credit link in
33
34 
```

variant-duo.css

```
1 /* Original design: Variant Duo (v1.0 - Oct 08, 2010) - A free xhtml/css website template by Andreas Vikilund. For more information, see http://andreasviklund.com/templates/variant-duo/ */
2
3
4 */
5
6 /* Main containers */
7 body {padding:0; margin:0; font:83% tahoma,verdana,sans-serif; background-color:#e4e4e4; color:#333; text-align:center; line-height:1.6em; }
8 #wrap {width:980px; text-align:left; margin:0 auto; }
9 #header {text-align:right; margin-top:40px; }
10 #content {background-color:#fff; text-align:left; padding:20px 20px 5px; margin:15px 0 15px 0; }
11
12 /* HTML Tags */
13 a {text-decoration:none; font-weight:400; color:#b59e4e; }
14 a:hover {text-decoration:underline; }
15 a img {border:0; }
16 h1 {margin:15px 0 10px auto; padding:0; font-size:2.4em; color:#27a9f; letter-spacing:-1px; line-height:1.2em; }
17 h1 a {color:#27a9f; font-weight:700; text-decoration:none; }
18 h1 a:hover {color:#333; text-decoration:none; }
19 h2 {margin:0 12px 0; padding:10px 0; color:#27a9f; font-size:1.4em; font-weight:700; border-top:1px solid #ddd; border-bottom:1px solid #ddd; }
20 h2 a {font-weight:700; }
21 h3 {font-size:1.4em; font-weight:400; margin:0 10px 0; }
22 p {margin:0 15px 0; }
23 ul {margin:0 15px 20px; padding:0; }
24 li {margin:0; padding:0 0 5px; }
25
26 /* Various classes */
27 .slogan {color:#555; font-size:1.4em; margin:0 0 15px 0; padding:0; }
28 .feature {border-top:2px solid #ff; border-bottom:2px solid #ff; }
29 .menulinks {font-size:1.5em; line-height:1.9em; color:#555; margin:0; }
30 .menulink {padding:0 8px 5px 8px; font-weight:400; margin:0; }
31 .menulink a {color:#27a9f; font-weight:400; }
32 .menulink a:hover {background-color:#27a9f; color:#fff; text-decoration:none; }
33 .active {background-color:#27a9f; color:#fff; }
34 .active a {color:#fff; }
35 .active a:hover {background-color:#27a9f; color:#fff; }
36
37 .footer {font-size:0.8em; clear:both; width:980px; line-height:1; }
```



Pre-requisites for Windows

- ❖ Sublime Text: <http://www.sublimetext.com/3>
 - ❖ Heavy-weight IDE: RStudio, Eclipse, Visual Studio, PyCharm

C:\Sample Files\interactive.py - Sublime Text 2

File Edit Selection View Tools Project Preferences Help

interactive.py

```
1 # coding: utf-8
2 #
3 # Copyright (C) 2006-2007 Alec Thomas <alec@swapoff.org>
4 #
5 # This software is licensed as described in the file COPYING, which
6 # you should have received as part of this distribution.
7 #
8 #
9 """CLY and readline, together at last.
10
11 This module uses readline's line editing and tab completion along w/
12 grammar parser to provide an interactive command line environment.
13
14 It includes support for application specific history files, dynamic
15 customizable completion key, interactive help and more.
16
17 Press ?? at any location to contextual help.
18 """
19
20 import os
21 import sys
22 import readline
23 import cly.readline
24 import cly.context
25 import cly.console as console
26 from cly.exceptions import Error, ParseError
27 from cly.builder import Grammar
28 from cly.parser import Parser
29
30 __all__ = ['Interact', 'Interact']
31 __docformat__ = 'restructuredtext en'
32
33
34 class Interact(object):
35     """CLY interaction through readline. Due to readline limitation,
36     Interact object can be active within an application.
37
38     Constructor arguments:
39
40     *parser*: ``Parser`` or ``Grammar`` object
41         The parser/grammar to use for interaction.
42
43     *application*: string
44         The application name. Used to construct the history file name
45         prompt, if not provided.
46
47     *prompt*: string
48         The prompt.
```

Line 1, Column 1

interactive.py

```
71     ``help_key``?: ``key``
72         Key to use for tab completion.
73
74     """
75     _cls_inject_text = ''
76     _completion_candidates = []
77     _parser = None
78     prompt = None
79     user_context = None
80     history_file = None
81     application = None
82
83     def __init__(self, grammar_or_parser, application='cly', prompt=None,
84                  user_context=None, with_context=None, history_file=None,
85                  history_length=500, completion_key='tab',
86                  completion_delimiters=' \t',
87                  help_key='?', inhibit_exceptions=False,
88                  with_backtrace=False):
89         if prompt is None:
90             prompt = application + '> '
91         if history_file is None:
92             history_file = os.path.expanduser('~/.%s_history' % app)
93         if isinstance(grammar_or_parser, Grammar):
94             self._parser = Parser(grammar_or_parser)
95         else:
96             self._parser = grammar_or_parser
97
98         if with_context is not None:
99             self._parser.with_context = with_context
100        if user_context is not None:
101            self._parser.user_context = user_context
102        Interact._parse = self._parser
103        Interact.prompt = prompt
104        Interact.application = application
105        Interact.with_context = user_context
106        Interact.history_file = history_file
107        Interact.history_length = history_length
108        Interact.completion_delimiters = completion_delimiters
109        Interact.completion_key = completion_key
110
111    try:
112        readline.set_history_length(history_length)
113        readline.read_history_file(history_file)
114    except:
115        pass
116
117    readline.parse_and_bind("%s complete" % completion_key)
118    readline.set_completer_delims(self.completion_delimiters)
```

Python

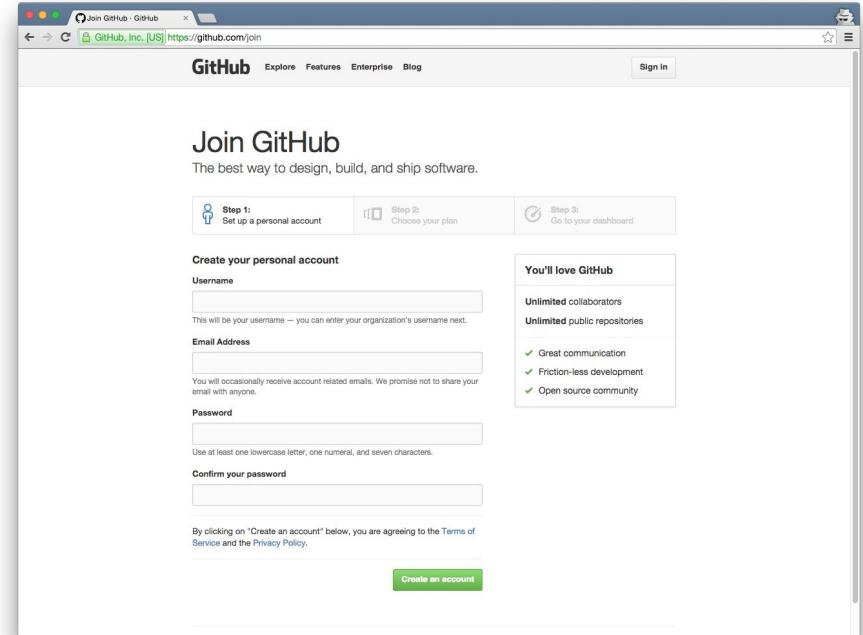
UI

Pre-requisites for Mac

- ❖ Sublime Text: <http://www.sublimetext.com/3>
- ❖ TextMate: <http://macromates.com/download>
- ❖ LightTable: <http://lighttable.com/>
- ❖ MacVim: <https://github.com/macvim-dev/macvim>
- ❖ Many other options ...

Pre-requisites

- ❖ Create a personal GitHub account.
- ❖ Choose the free option: get unlimited open repositories, but no private repositories.
- ❖ Save your credentials in a safe place.



Pre-requisites

- ❖ Bookmark the free Pro Git book <http://git-scm.com/book/en/v2/>

The screenshot shows the official website for the Pro Git book. At the top, there's a navigation bar with links for 'About', 'Documentation' (which is currently selected), 'Reference', 'Book', 'Videos', and 'External Links'. Below this is a 'Blog' section, followed by 'Downloads' and 'Community' sections.

The main content area is titled 'Book'. It contains a brief description of the book: 'The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com.' To the right of this text is the 'Pro Git' book cover for the 2nd Edition (2014). Below the cover, there are links to '2nd Edition (2014)' and 'Switch to 1st Edition'.

Under the 'Downloads' section, there are four icons representing different file formats: PDF (blue), EPUB (green), MOBI (orange), and HTML (purple).

About

Documentation

- Reference
- Book
- Videos
- External Links

Blog

Downloads

Community

Download this book in PDF, mobi, or ePub form for free.

This book is translated into Deutsch, 简体中文, 正體中文, Français, 日本語, Nederlands, Русский, 한국어, Português (Brasil) and Čeština.

Partial translations available in Arabic, Español, Indonesian, Italiano, Svenska, Magyar, and Dansk.

Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).

Pro Git

SECOND EDITION

Scott Chacon and Ben Straub

Apress

2nd Edition (2014)

Switch to 1st Edition

1. Getting Started

- 1.1 About Version Control
- 1.2 A Short History of Git
- 1.3 Git Basics
- 1.4 The Command Line
- 1.5 Installing Git
- 1.6 First-Time Git Setup
- 1.7 Getting Help
- 1.8 Summary

Download Ebook

pdf epub

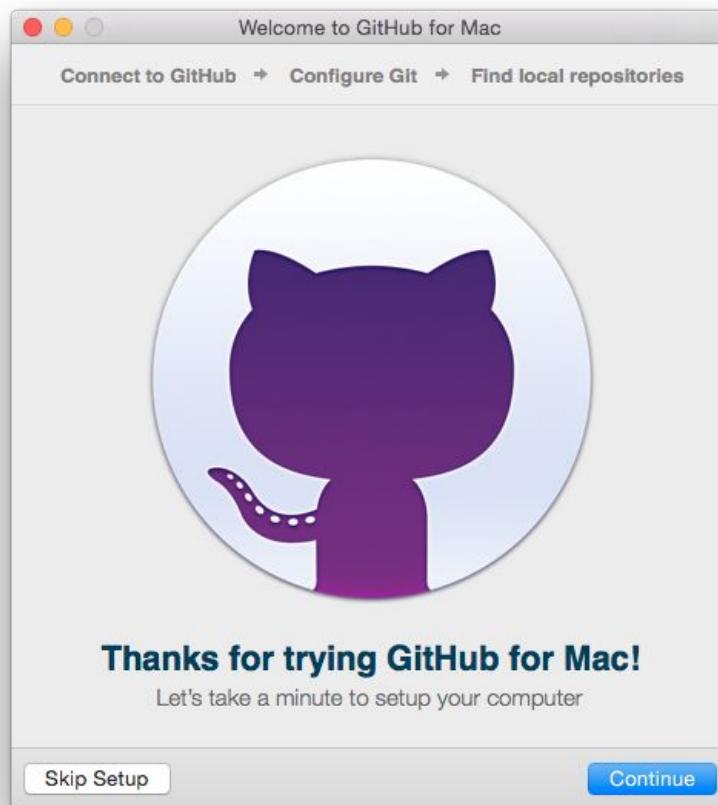
mobi html

Download and Install Github GUI For Your OS (Optional)

- ❖ Easy method: Download and install GitHub's application. May not work for all versions of your OS. If this doesn't work, it's ok, let an instructor know.
- ❖ Windows: <https://windows.github.com/>
- ❖ Mac: <https://mac.github.com/>

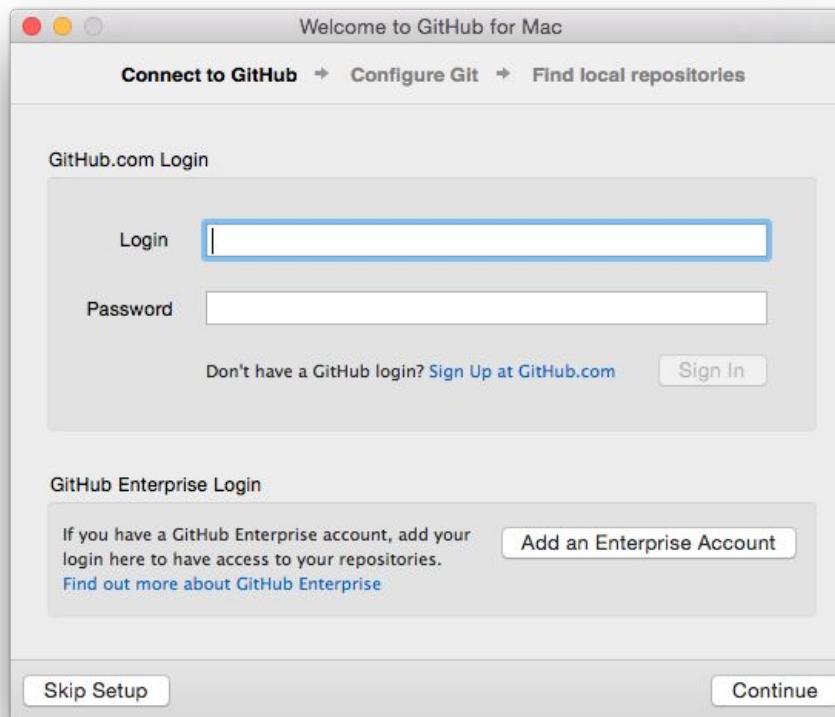
Download and Install Github GUI For Your OS (Optional)

- ❖ After you install the GUI, find the application in the start menu or Applications, and open it.



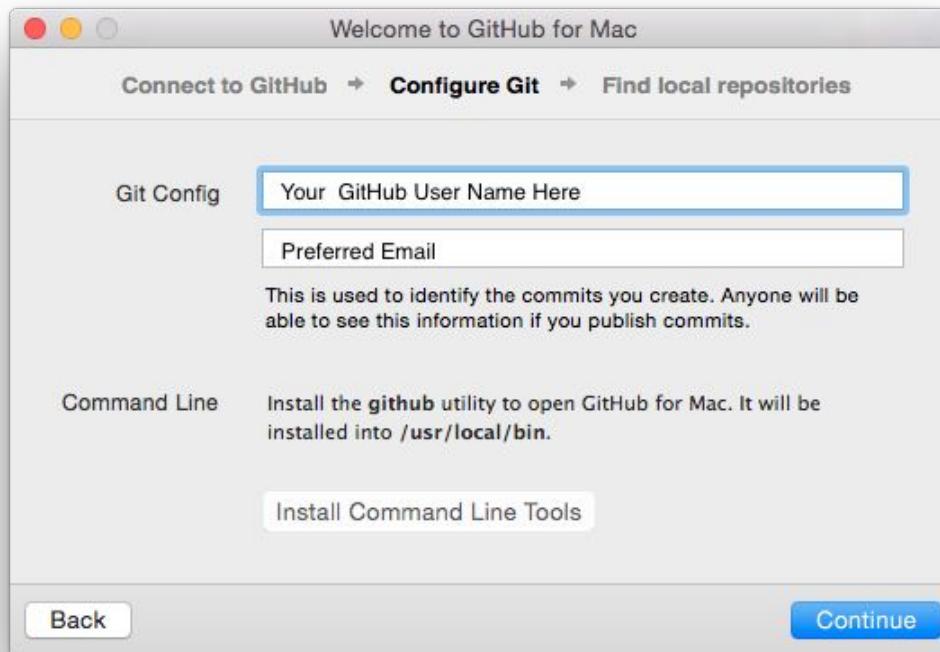
Download and Install Github GUI For Your OS (Optional)

- ❖ Enter your GitHub credentials through the setup wizard. This step should automatically configure your SSH keys for identification.



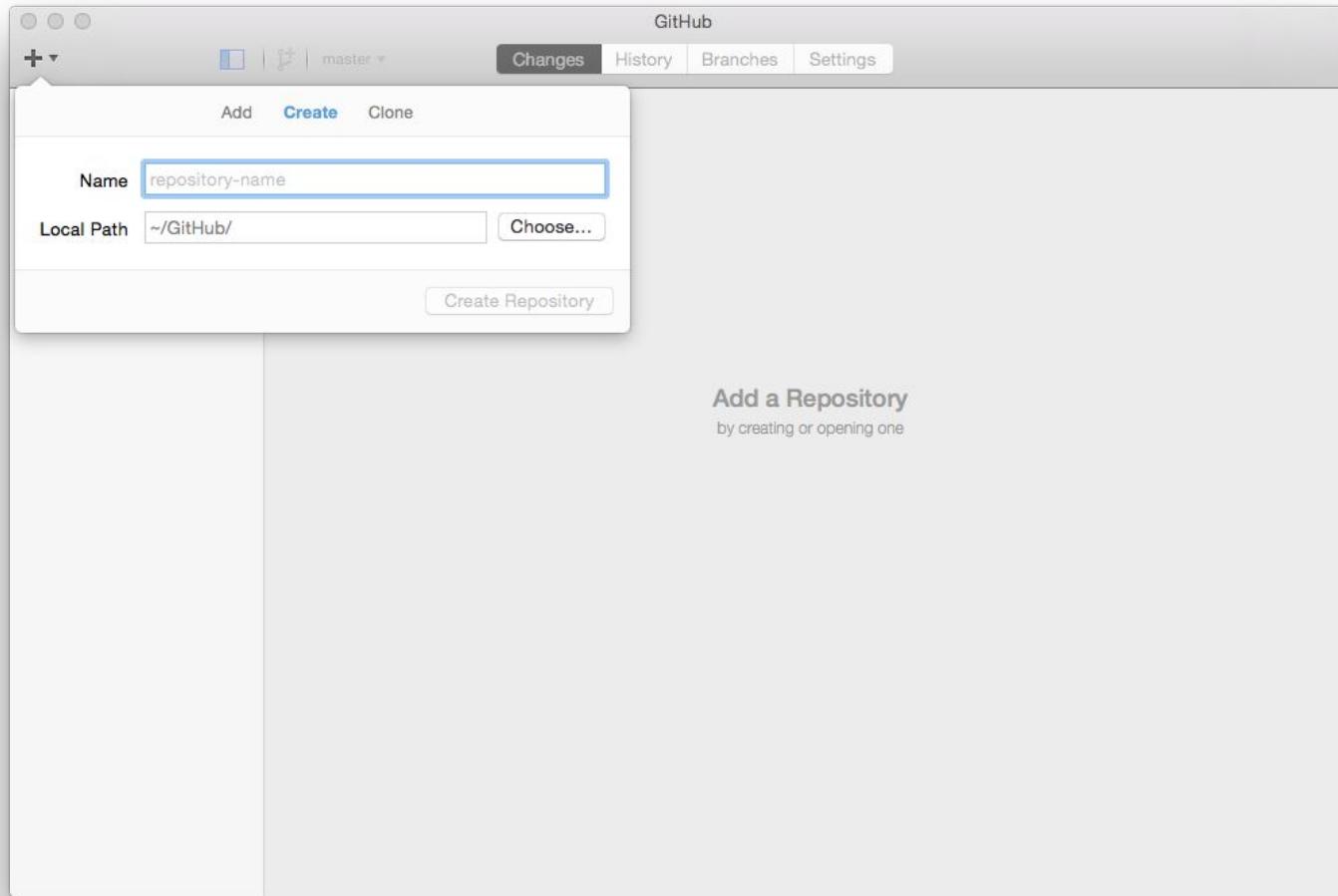
Download and Install Github GUI For Your OS (Optional)

- ❖ Enter the github username you created, and the preferred email you used as well. Your commits will be identified by these details, so make sure it's okay to share.



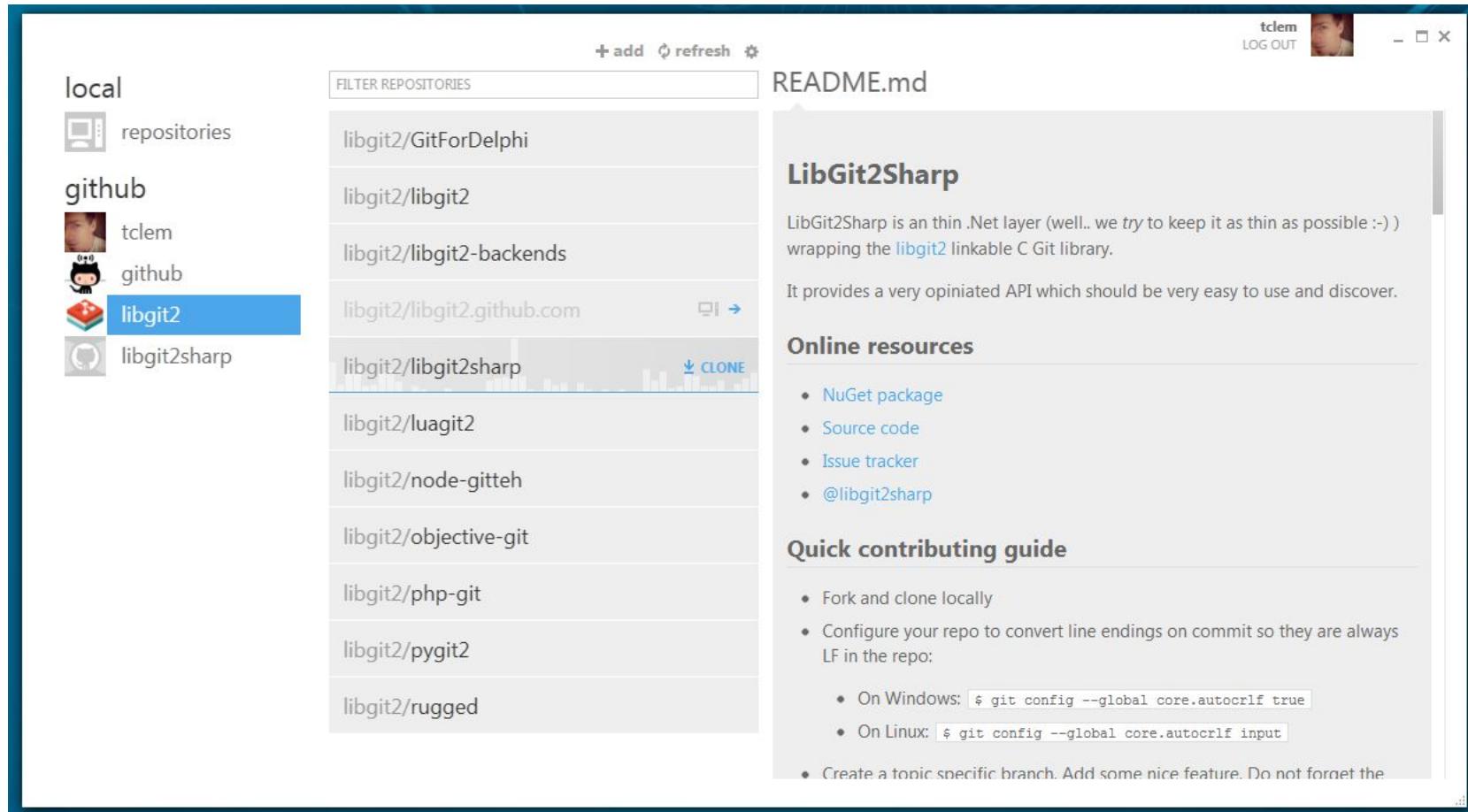
Download and Install Github GUI For Your OS (Optional)

- ❖ Example Application for Mac



Download and Install Github GUI For Your OS (Optional)

❖ Example Application for Windows



Command-Line For Windows

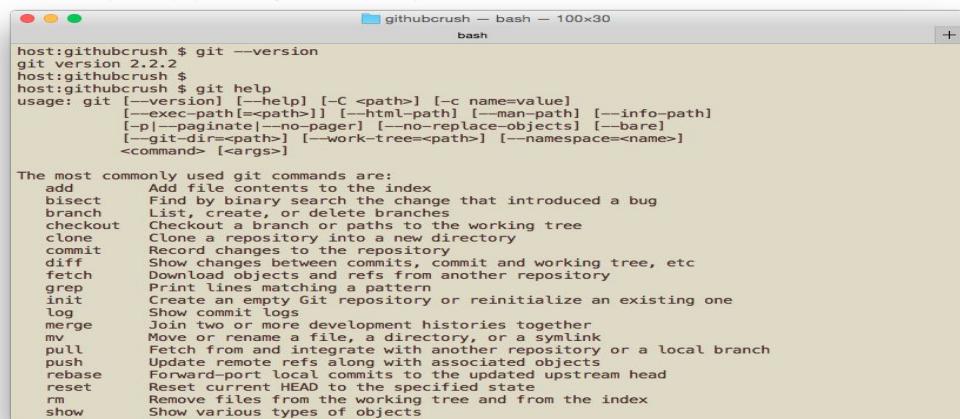
- ❖ Download the package msysgit from <https://git-for-windows.github.io/> and install it. This wizard will take you through the steps to install it.
- ❖ Assuming the default settings, open your 'My Documents' folder to find the root folder of the command-line environment that msysgit created.
- ❖ On your computer, find Git in the start menu, and open GitBash. If you find something like the following window, the installation is successful!



The screenshot shows a terminal window titled "MINGW32:/c/Users/alisa". The title bar also includes the path "MINGW32:/c/Users/alisa". The window contains the following text:
Welcome to Git (version 1.9.4-preview20140815)
Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.
alisa@WWW ~ (master)
\$

Command-Line for Mac

- ❖ Three alternative install methods:
 - Use the pre-packaged version of git provided by OS X
 - Directly install it from a package <http://git-scm.com/download/mac>
 - Install the Homebrew package manager <http://brew.sh/> and then open your terminal type `$ brew install git`
- ❖ Within Applications, find the Utilities folder, and open the Terminal application. If you can successfully type `git help` at the terminal, then installation was successful!



```
githubcrush $ git --version
git version 2.2.2
githubcrush $
githubcrush $ git help
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 files and update local branches, a 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
```

View All Configuration Information

- ❖ Start the terminal program for your OS. Type in this command to list all the known global Git configuration. If you don't see a **user.name** or **user.email** setting in your global config, then it has to be manually configured.

```
$ git config --list --global  
...  
user.name='your username'  
user.email=your email  
...
```

Configuration for Git

- ❖ View the version information and verify that the installation was successful. E.g. on Windows:

```
$ git --version  
git version 1.9.4.msysgit.1
```

- ❖ Set your user information in the command window/terminal

```
$ git config --global user.name "username"  
$ git config --global user.email "your email"
```

- ❖ Note : The parameter --global in the git config command indicates that all repositories on this host will use this configuration. For setting a different configuration on an individual repository, use the --local parameter.

View Specific Configuration Information

- ❖ If you want to view a particular setting, like user.email, specify that parameter after the git config command.

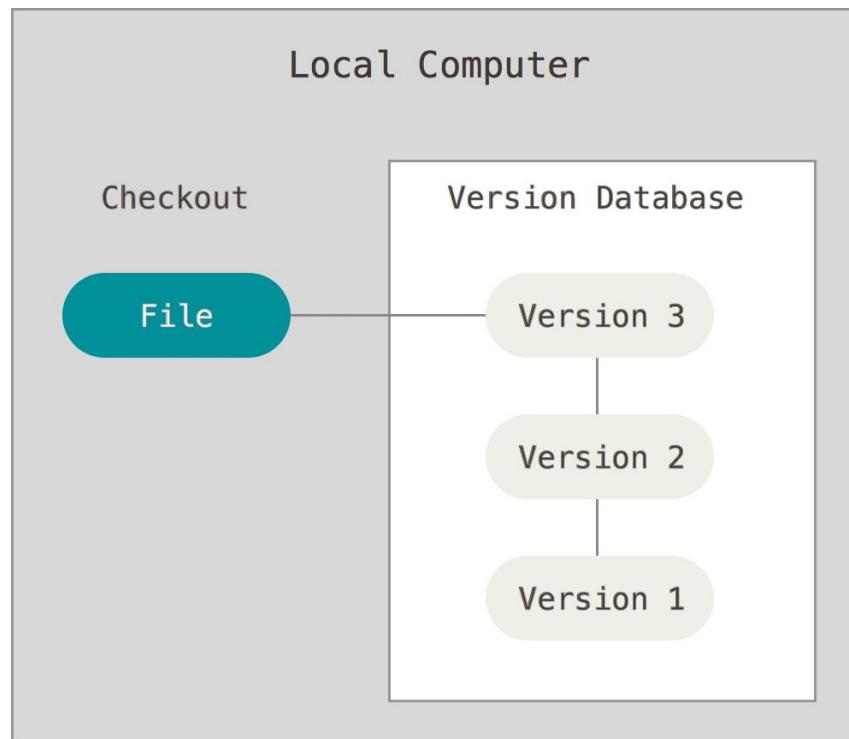
```
$ git config user.email  
my.email@domain.com
```

Outline

- ❖ Set up Git and GitHub
- ❖ Introduction to Git
 - Creating a Git Repository
 - Manipulating files
- ❖ Introduction to GitHub
 - Lightning Tour of Github
 - Create a Remote Repository

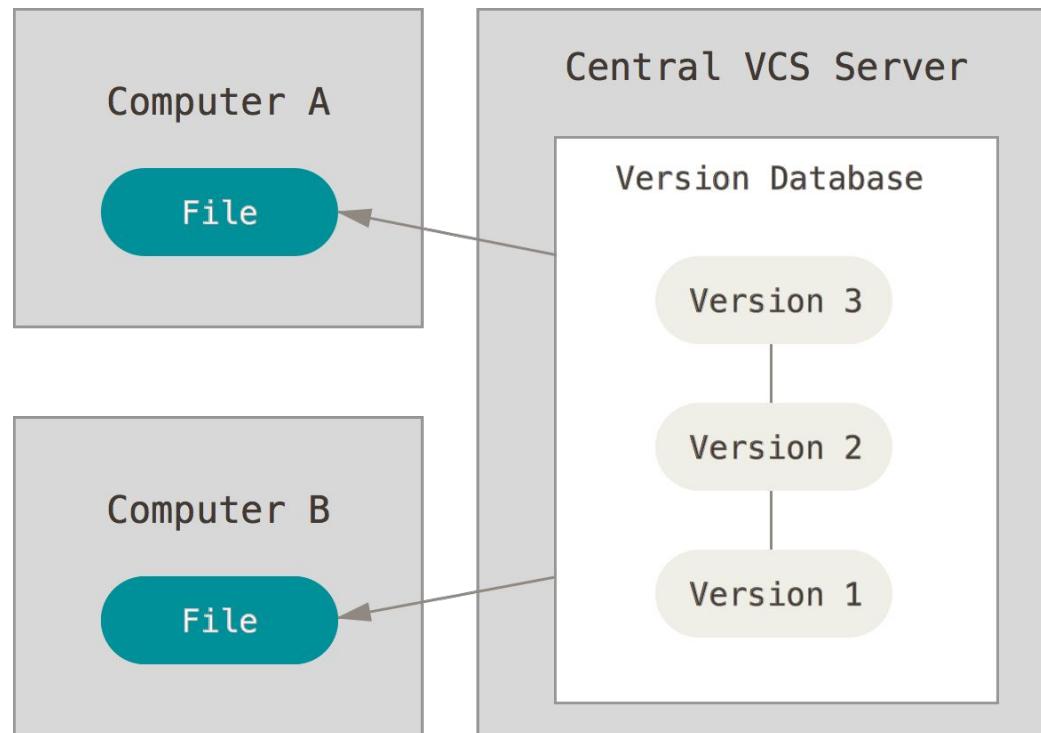
What is Version Control?

- ❖ Version control allows us to capture and refer back to a known state of a file across a series of changes.
- ❖ E.g. Mac OS X File System, MS Office App 'Track Changes' feature



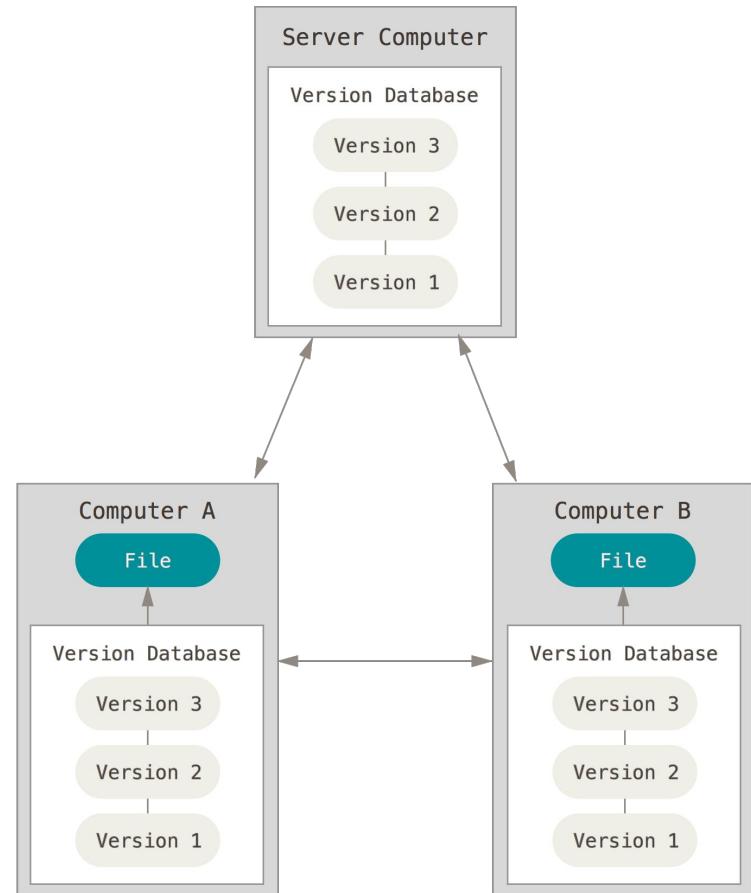
What is Centralized Version Control?

- ❖ Sharing changes between hosts is done through a central authority (the source of truth).
- ❖ E.g. standard Google spreadsheet



What is Distributed Version Control?

- ❖ Sharing changes between hosts is done using a peer-to-peer model (no real source of truth).
- ❖ E.g. Git, offline Google spreadsheet



Introduction to Git

- ❖ Distributed version control system, great for collaboration over source code
- ❖ Linus Torvalds developed Git in 2005 to help manage the Linux kernel development across thousands of volunteers
- ❖ Fast and robust performance
- ❖ Wide community adoption
- ❖ Allows massive, parallel branch development
- ❖ Has the ability to efficiently manage large scale projects

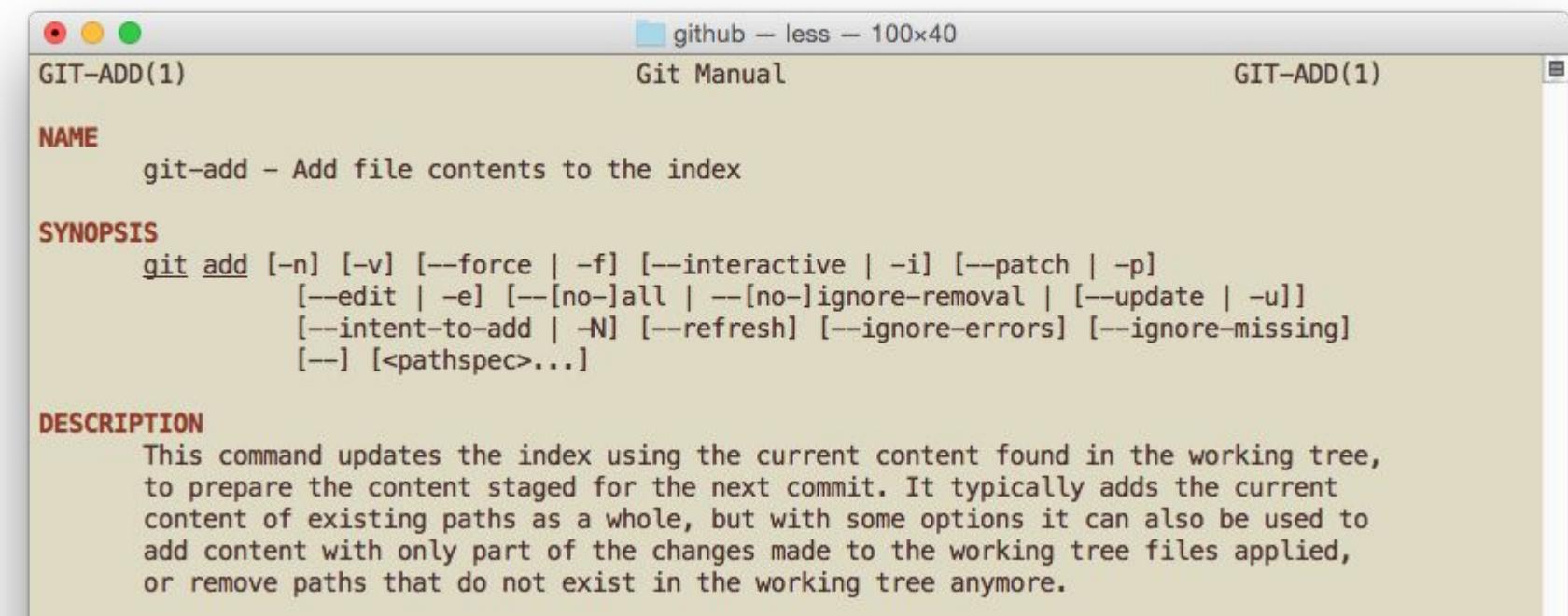
Introduction to Git

- ❖ If you're coming from another Version Control System (VCS), note that Git has some differences:
 - Doesn't really work like CVS, Perforce, i.e. centralized version control.
 - Instead of encoding deltas between one version and the next, Git works with files as a whole.
 - Treats the history as a series of snapshots, which are then efficiently stored in the local database.
 - Many commands in Git work on primarily the local data. No need to talk to a remote server, unless you want to exchange history, branches, and/or tags.

Help with Git Commands

- ❖ For more information about Git's many commands, consult the help command.

```
$ git add --help  
$ git help add
```



The screenshot shows a terminal window titled "github — less — 100x40". The window contains the output of the "git help add" command. The output is organized into sections: NAME, SYNOPSIS, and DESCRIPTION. The NAME section defines "git-add" as "Add file contents to the index". The SYNOPSIS section provides the command-line syntax for "git add". The DESCRIPTION section explains that the command updates the index using current content from the working tree, preparing it for the next commit. It can add content as a whole or part, and handle paths that no longer exist.

```
GIT-ADD(1)                               Git Manual                               GIT-ADD(1)  
  
NAME  
git-add - Add file contents to the index  
  
SYNOPSIS  
git add [-n] [-v] [--force | -f] [--interactive | -i] [--patch | -p]  
        [--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]]  
        [--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing]  
        [--] <pathspec>...  
  
DESCRIPTION  
This command updates the index using the current content found in the working tree,  
to prepare the content staged for the next commit. It typically adds the current  
content of existing paths as a whole, but with some options it can also be used to  
add content with only part of the changes made to the working tree files applied,  
or remove paths that do not exist in the working tree anymore.
```

Help on the Web

- ❖ Many online resources, but here is one: <http://git-scm.com/docs/git-add>

The screenshot shows a web browser displaying the documentation for the `git add` command. The URL is <http://git-scm.com/docs/git-add>. The page is titled "git - local-branching-on-the-cheap". On the left, there's a sidebar with links to "About", "Documentation" (which is currently selected), "Blog", "Downloads", and "Community". The main content area shows the command syntax in a code block:

```
'git add' [-n] [-v] [--force | -f] [--interactive | -i] [--patch | -p]
[--edit | -e] [--no-all | --[no-]ignore-removal | --update | -u
--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-mis
[--] <pathspec>...]
```

Below the syntax, there are sections for "DESCRIPTION" and "OPTIONS". The "DESCRIPTION" section explains that the command updates the index using the current content found in the working tree to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore. The "OPTIONS" section details various flags like `-n` (dry-run), `-v` (verbose), `--force`, `--interactive`, `--patch`, `--edit`, `--no-all`, `--refresh`, `--ignore-errors`, and file glob patterns.

Outline

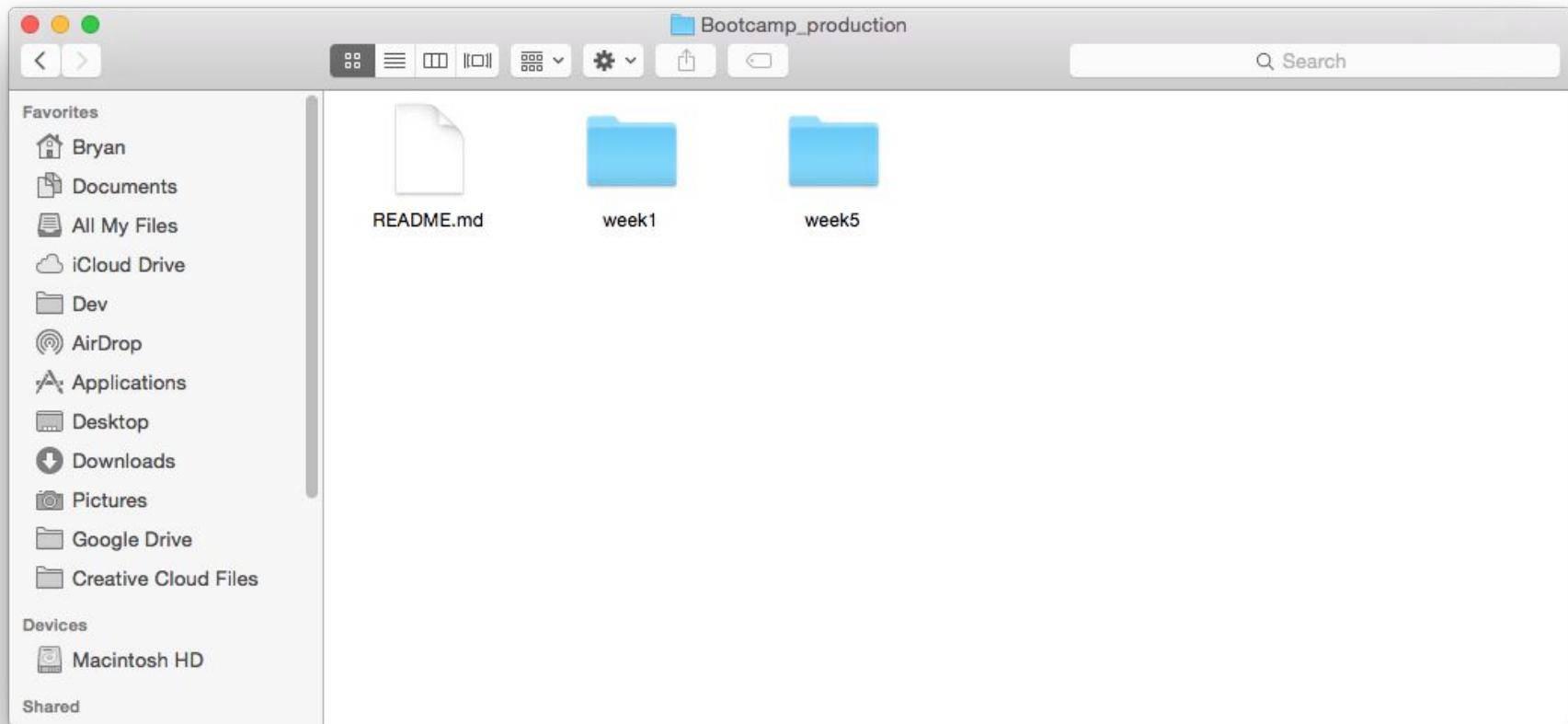
- ❖ Set up Git and GitHub
- ❖ Introduction to Git
 - Creating a Git Repository
 - Manipulating files
- ❖ Introduction to GitHub
 - Lightning Tour of Github
 - Create a Remote Repository

What is a Git Repository?

- ❖ View the repository as a controlled working area (a directory) with a version database.
 - Some or all of the files in this directory can be managed by Git:
 - add and commit files to the repository
 - move or delete files
 - view updated files
 - view history
 - cancel local changes
 - Git can even follow renamings (or moving) of files!

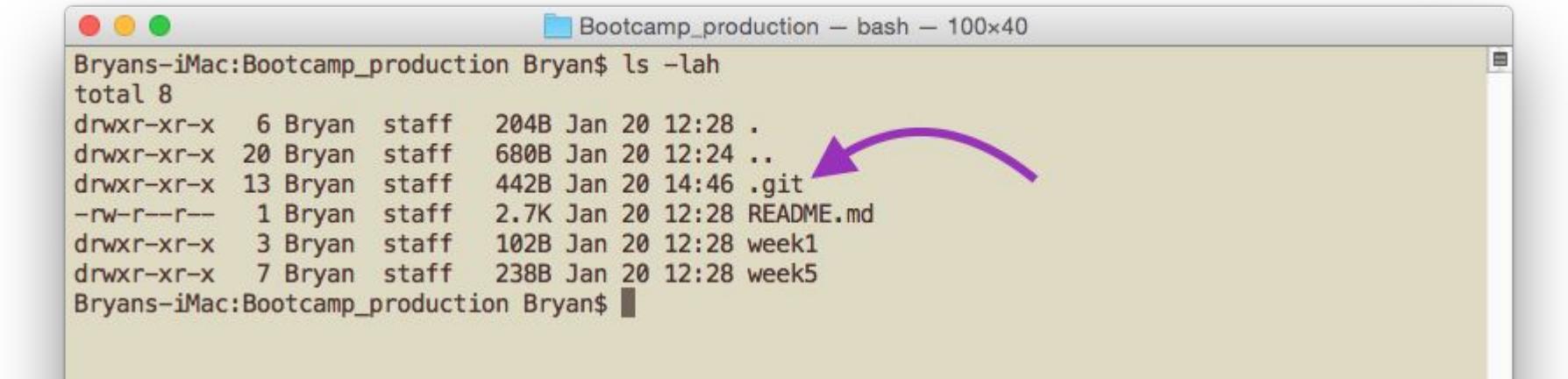
What is a Repository?

- ❖ A repository is a directory that contains other files and subdirectories on some volume.



What is a Repository?

- ❖ Plus the local, sometimes hidden .git database within.



```
Bryans-iMac:Bootcamp_production Bryan$ ls -lah
total 8
drwxr-xr-x  6 Bryan  staff  204B Jan 20 12:28 .
drwxr-xr-x 20 Bryan  staff  680B Jan 20 12:24 ..
drwxr-xr-x 13 Bryan  staff  442B Jan 20 14:46 .git
-rw-r--r--  1 Bryan  staff  2.7K Jan 20 12:28 README.md
drwxr-xr-x  3 Bryan  staff  102B Jan 20 12:28 week1
drwxr-xr-x  7 Bryan  staff  238B Jan 20 12:28 week5
Bryans-iMac:Bootcamp_production Bryan$
```

- ❖ Note: On Windows, the .git folder is usually visible. On Linux and Mac, all '.name' folders are hidden by default.

Create a Repository

- ❖ There are two ways to create a repository:
 - Create a local repository directly:
 - Create a repository locally using the git init command.
 - Later, add a remote repository reference to the configuration, and push data to that remote.
 - Clone a remote repository to your local machine.

Create a Local Repository

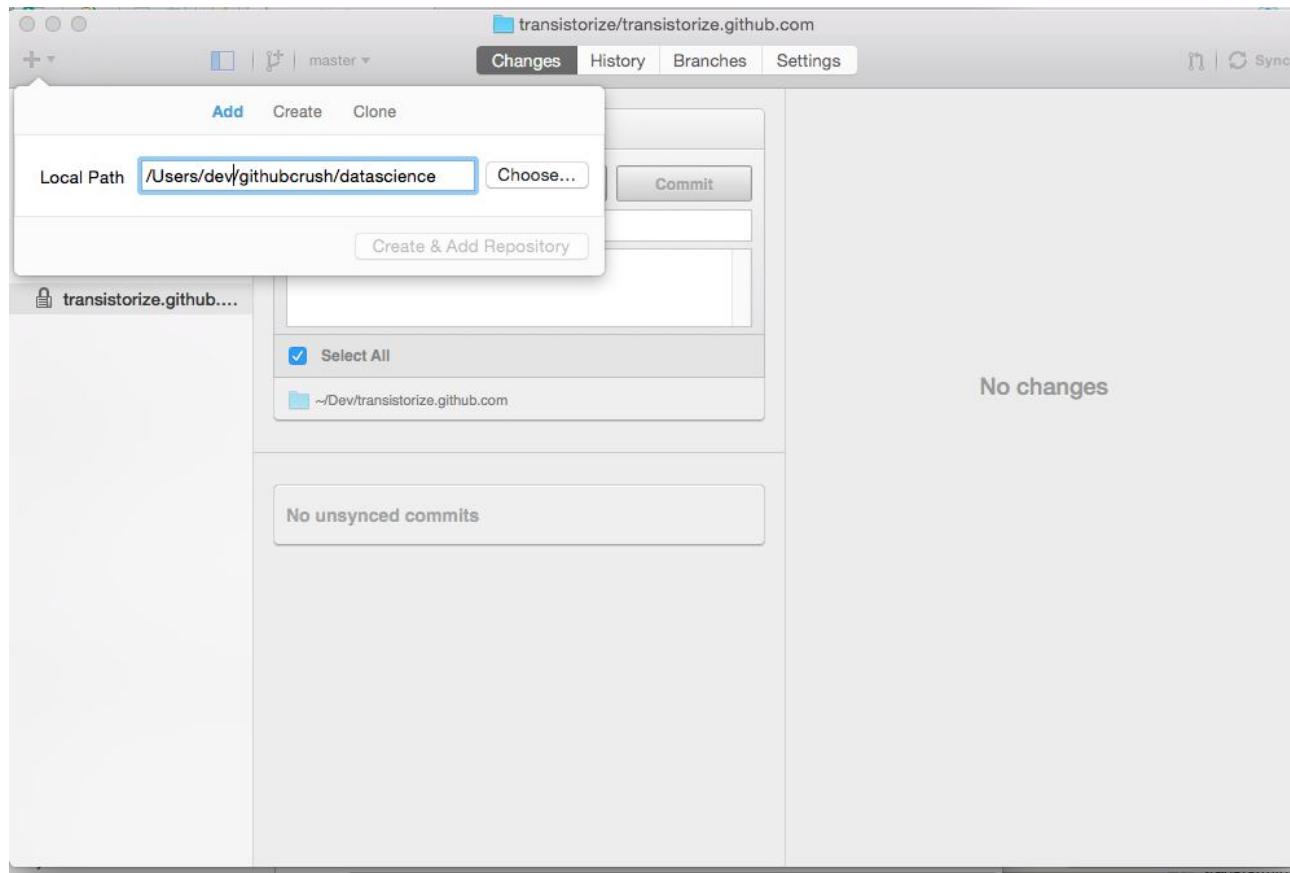
- ❖ **Step 1:** Choose a working directory and initialize an empty directory
 - Type the following commands one at a time. Everything after the double forward slash (//) are just comments.

```
$ pwd                  // show current directory  
/c/Users/dev/githubcrush  
$ mkdir datascience    // create empty directory  
$ cd datascience      // enter the directory  
$ git init             // initial new repository, an empty repository  
Initialized empty Git repository in c:  
/Users/dev/githubcrush/datascience/.git/
```

- ❖ Note: The git init command is used to change a local directory into a repository managed by Git.

Create a Local Repository

- ❖ Optionally, add the repository to the GitHub UI.



Add Content to the Local Repository

- ❖ **Step 2:** Create a new text file using your text editor, create a file named data.txt within the datascience directory you created before. Add the following content to this file:

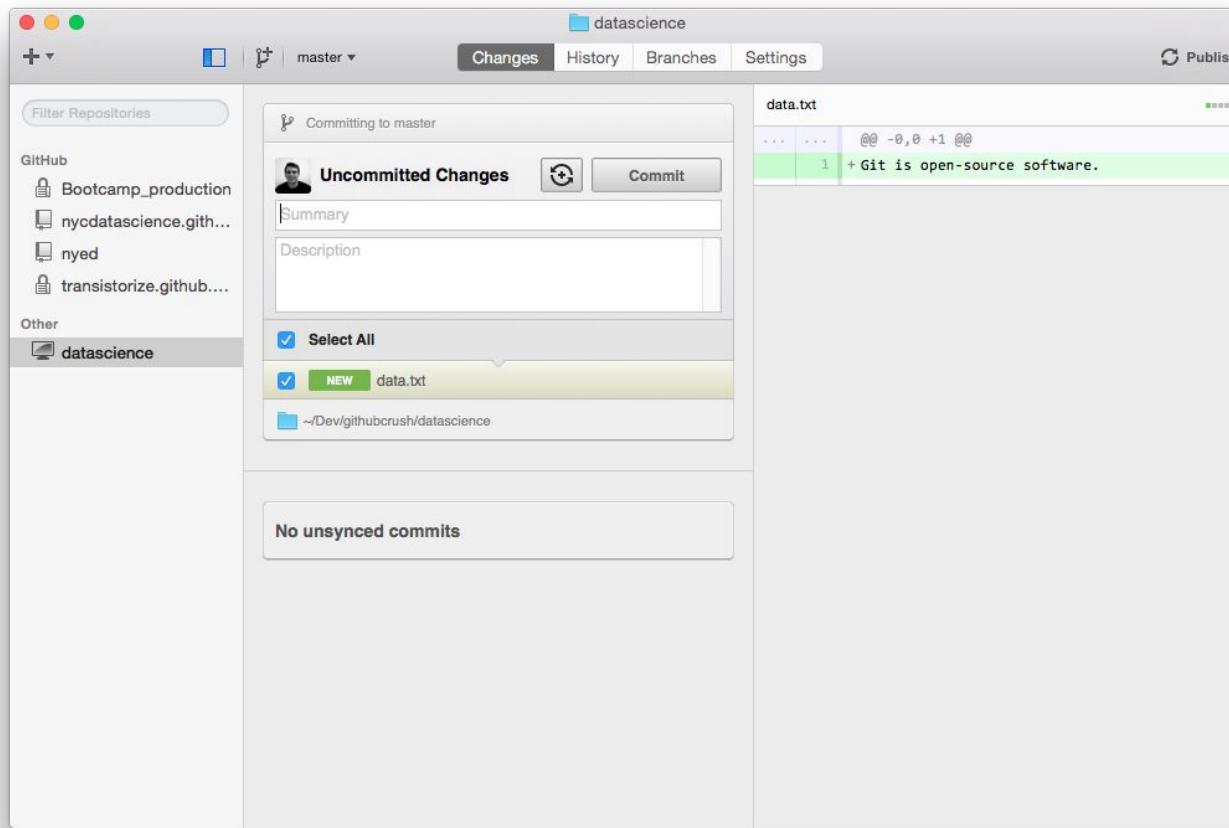
```
Git is open-source software.
```

- ❖ Note: on Windows, make sure Notepad++ is configured with the settings: UTF-8 without BOM.
- ❖ FYI: On Linux and Mac, you can create a file via the command line.

```
$ touch data.txt          // creates an empty file
$ echo "Git is open-source software." > data.txt
$ ls -a                  // lists files in the directory
.   ..      .git           data.txt
```

Status of the Local Repository

- ❖ Verify in the UI that a new file has been created.

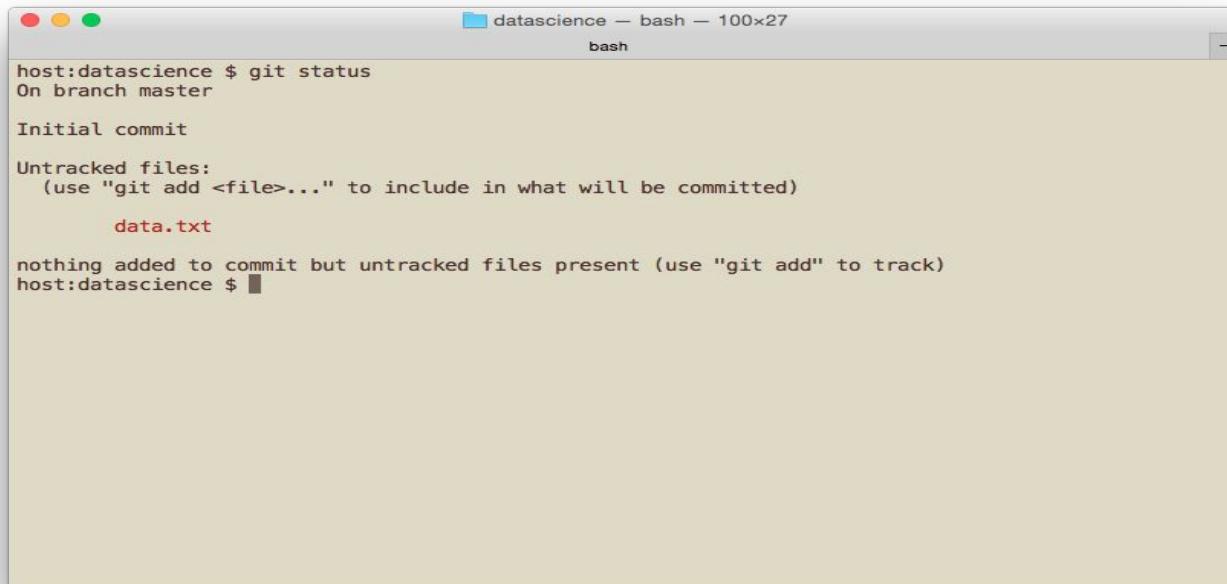


Status of the Local Repository

- ❖ The status command is used to show what changes git has detected in the local working directory.

```
$ git status          //examine git's point of view
```

- ❖ What's going on here? What does it mean to be untracked?



A screenshot of a macOS terminal window titled "datascience — bash — 100x27". The window shows the following text output:

```
host:datascience $ git status
On branch master

Initial commit

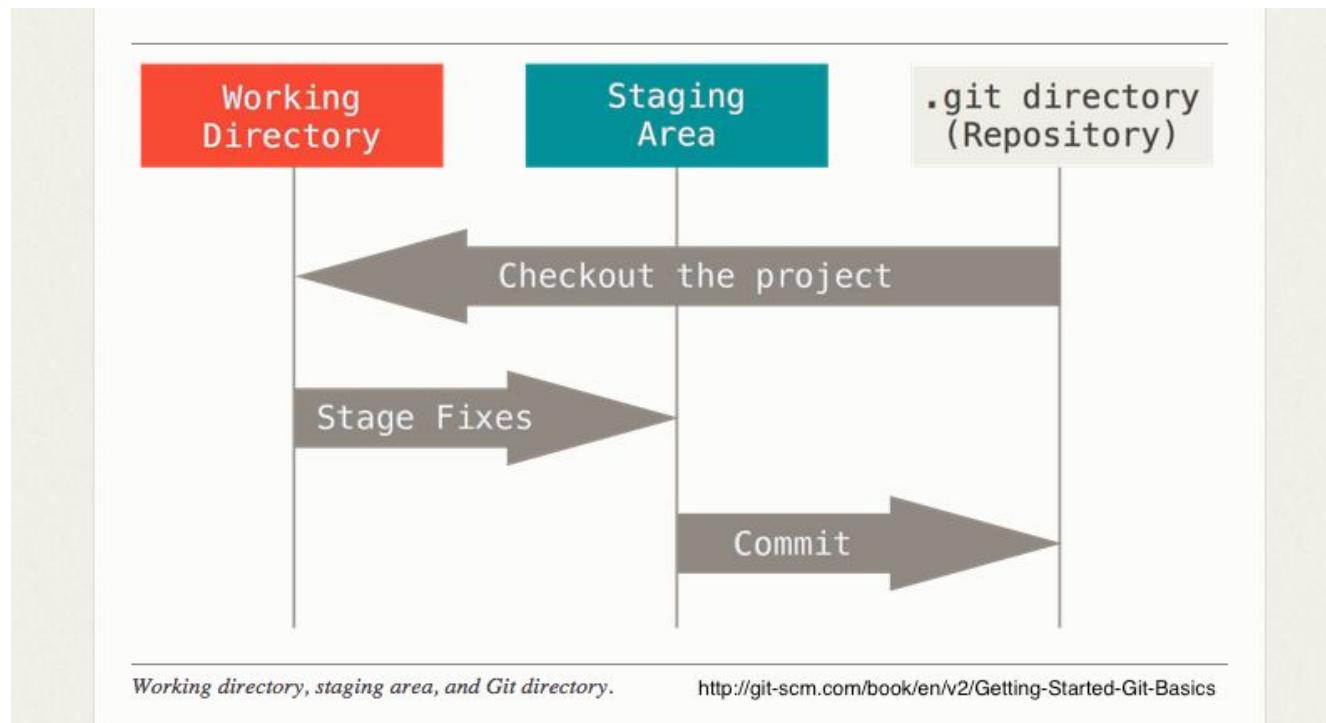
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    data.txt

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

Git Model: Working Directory

- ❖ Untracked or modified items have to be added to the index file in the .git repository, a.k.a. the staging area



- ❖ Note: Later, we will learn how to go straight from creating or modifying a file to commit it.

Add a file to the staging area

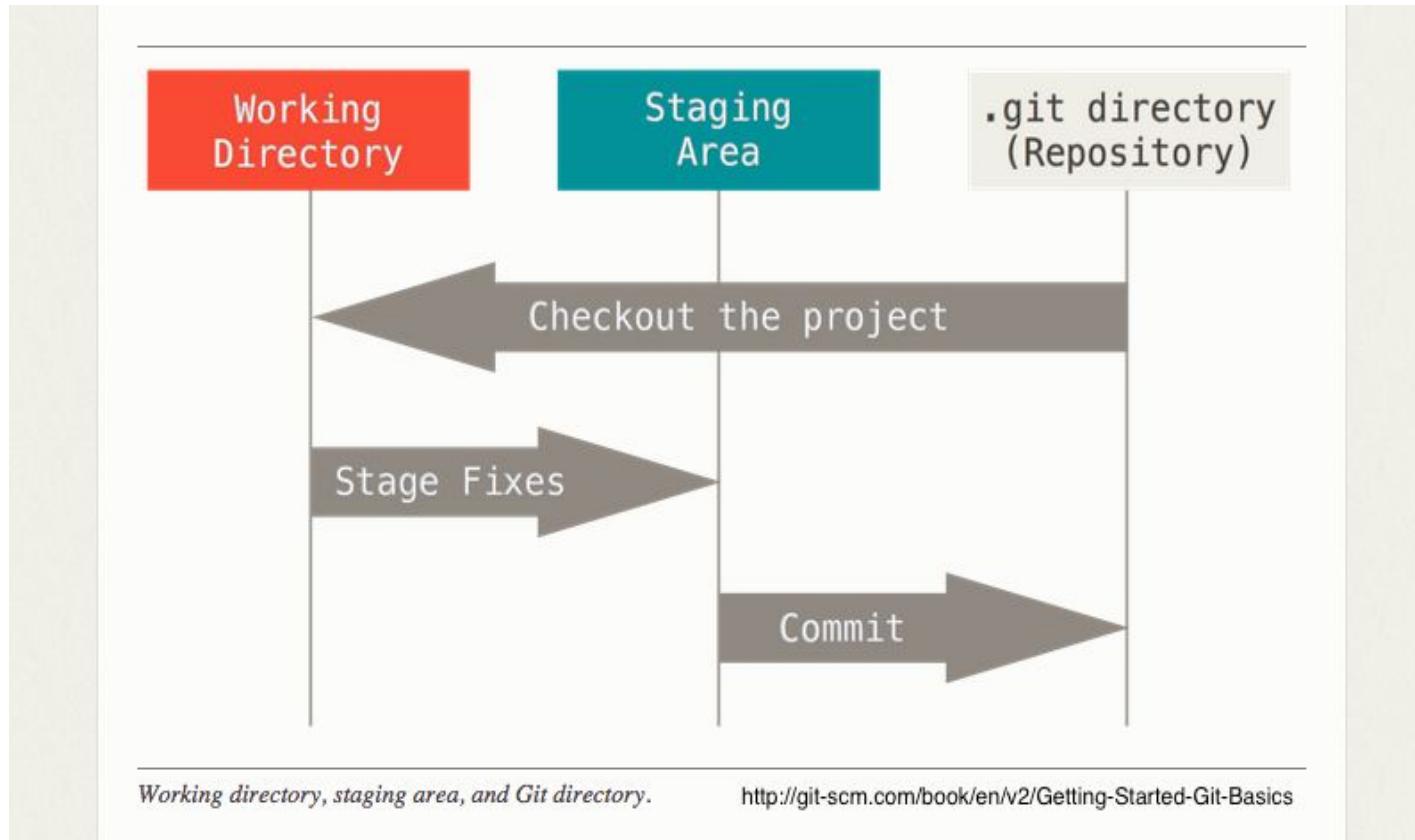
- ❖ **Step 3:** Once the file is created, we must add it to the staging area and then commit the file to the repository.

```
$ git add data.txt      // add file to staging
$ git status            // view status of current working
directory
On branch master        // name of branch currently checked out
Initial commit
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   data.txt
```

- ❖ Note: The command `git status` tells us a new file was added and reminds us to commit it. Also, we can cancel the tracked status of the file by using the command `git rm --cached`

Git Model: Staging Area (Index)

- The staging area (index) tracks all the modified file content.



Commit Content to the Repository

- ❖ **Step 4:** Commit the file to the local repository

```
$ git commit -m "Add new data."  
[master (root-commit) 2a6487f] Add new data to the project.  
 1 file changed, 1 insertion(+)  
   create mode 100644 data.txt  
$ git status  
On branch master  
nothing to commit, working directory clean
```

- ❖ Note : After the parameter -m in git commit, provide a short, clear description of this change. It's makes it easier to find and understand changes in the history log. Also, we know the commit ID of the file on the master branch begins with 2a6487f. The description of this file submitted is 'Add new data to the project'.

Commit History

- ❖ View the commit log to see the recent history

```
$ git log --graph --decorate  
* commit 2a6487f3bd883297451f3c6412b81555ef45942f  
  Author: user.name <my.email@domain.com>
```

Add new data.

- ❖ Note: Try `$ git log --oneline` You can mix flags to find your output preference. See `git help log` for more details.

Exercise 1

- ❖ Create a new directory that is a sibling to the datascience directory. Call it exercise_1. Initialize that directory to be another git repository. If it helps, add that second repository to the UI tool.
- ❖ Create a new empty file in the exercise_1 directory. Add and commit this new file to your second repository.
- ❖ What branch did you commit this change to?
- ❖ Change directories back to the datascience directory. What is the status of the datascience repository?

Change Content in the Repository

- ❖ **Step 1:** Create multiple files using a text editor.
 - Modify data.txt file, and append this line to the file: **R is open-source software.**
 - Create the new file www.txt to have the string 'www'
 - Create the new file test.txt to have the string 'test'
 - Run `$ git status` Are all the files treated the same?

Change Content in the Repository

- ❖ Step 2: Try adding multiple files simultaneously to the staging area
- ❖ Using the --all parameter, add all files to the repository simultaneously.

```
$ git add --all
```

- ❖ Run \$ git status What do you find?

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

modified:   data.txt
new file:   test.txt
new file:   www.txt
```

Reset a File from Staging

- ❖ **Step 3:** Reset the files you just added, out of the staging area.

```
$ git reset HEAD data.txt test.txt www.txt
```

```
$ git status
```

- ❖ Note: In this case, this command does not modify the actual files. For now, ignore the mention of the HEAD reference.

Reset a File from Staging

- ❖ **Step 4:** Re-add all the files you just reset, back into the staging area.

```
$ git add data.txt test.txt www.txt
```

```
$ git status
```

Change Content in the Repository

- ❖ **Step 5:** Commit all the files as one change. Pick an appropriate message.

```
$ git commit -m "..."
```

- ❖ When nothing is modified or untracked, it is said you have a clean working directory.

```
$ git status  
On branch master  
nothing to commit, working directory clean
```

Change Content in the Repository

- ❖ Without looking at the slides, what are the steps to creating or modifying content within a repository?
- ❖ Could I add an image to the datascience repository?
- ❖ When you touch a file, does git consider that a modification? How can you tell?

Commit a Modified File in One Command

- ❖ 'Skipping' the add-to-staging part
- ❖ Adding the -a parameter to git commit will commit all known files to the repository simultaneously. A file is known if it has been added and committed to the git index before. Still, be careful when using this combination.
- ❖ Modify test.txt to read as follows:

```
test! test! test!
```

- ❖ Then execute:

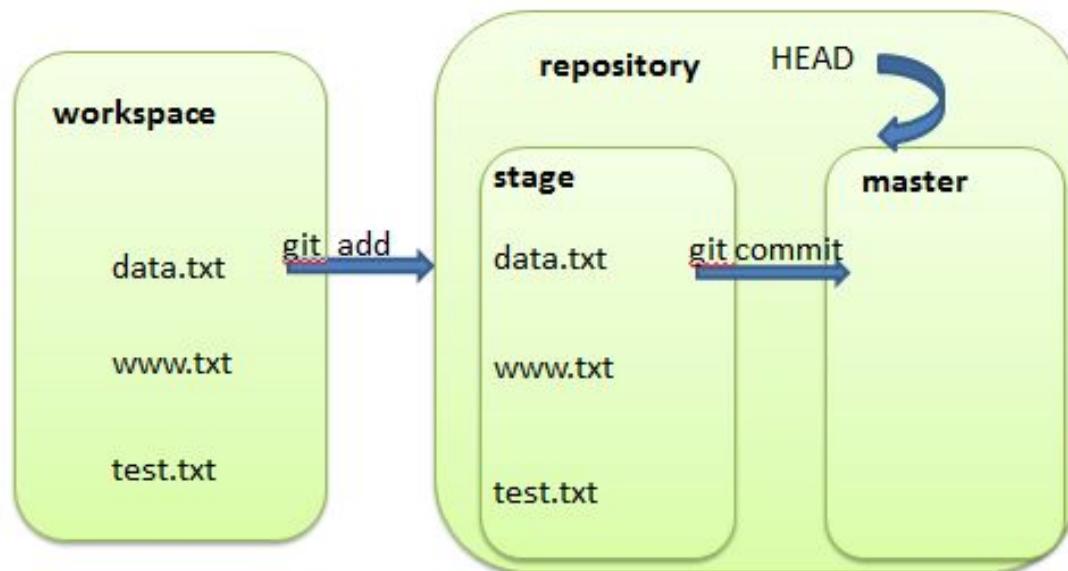
```
$ git commit -a -m "Commit more tests"  
[master ed1d0f8] Commit more tests  
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Quick Review

- ❖ In order to better understand the purpose of Git commands, keep the following model in mind:
 - **Branch:** A stream of file snapshots, like a tree branch. Only one branch is active at a time in a repository, often defaults to the master branch.
 - **Workspace:** What we see in the directory, the datascience folder created before is the working area.
 - **Staging:** Used to prepare a commit of one or more files, and you can review the change list.
 - **Repository:** There is a hidden directory called .git in the workspace, that is the 'real' repository. Keeps track of snapshots, history, the index, the HEAD pointer (references), branches, and more.

Quick Review - Workflow

- ❖ The workflow figure of adding and committing files to repository.
 - Firstly, git add adds files in the workspace to staging.
 - Secondly, git commit commits files in staging to the current branch.



Outline

- ❖ Set up Git and GitHub
- ❖ Introduction to Git
 - Creating a Git Repository
 - Manipulating files
- ❖ Introduction to GitHub
 - Lightning Tour of Github
 - Create a Remote Repository

Change a Local Repository - Delete a file

- ❖ We take the test.txt file, committed under datascience, for example.
 - the delete command is rm [file name]

```
$ rm test.txt

$ ls -a
.  ..  .git  data.txt  www.txt
```

Change a Local Repository - Delete a file

- ❖ When you delete the test.txt file, view the current status of the workspace.

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
    (use "git checkout -- <file>..." to discard changes in working
     directory)

          deleted:    test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Change a Local Repository - Delete a file

- ❖ See the difference between the current workspace and the latest version of the repository.

```
$ git diff HEAD -- test.txt
diff --git a/test.txt b/test.txt
deleted file mode 100644
index e69de29..0000000
```

Change a Local Repository - Delete a file

- ❖ Command rm test.txt just deleted the file in the workspace, but did not remove it from the repository, so:
 - If you want to restore files to the workspace, use the command:

```
$ git checkout -- test.txt
```

- ❖ Note : Command git checkout -- [filename] can restore the file to the workspace, so you can undo accidental, non-commited changes. The --, in this case, is a generic way to refer to the current branch.

Change a Local Repository - Delete a file

- ❖ If you really want to remove the file from the repository, use the git rm command:

```
$ git rm test.txt  
rm 'test.txt'
```

```
$ git commit -m "delete test.txt"  
[master b81549a] delete test.txt  
 1 file changed, 0 insertions(+), 0 deletions(-)  
 delete mode 100644 test.txt
```

- ❖ Note: When you delete a file, commit the updated status of the file. When you do this, you are removing the file from future snapshots, but the previous versions of the file are still in the history where it can be retrieved.

Change a Local Repository - View Changes

- ❖ We completely overwrite the contents of data.txt, by changing its contents to:

```
R is software.
```

```
R is free software.
```

- ❖ Before adding file to the staging area, view its status:

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working  
  directory)
```

```
    modified:   data.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

Change a Local Repository - View Changes

- ❖ After modifying the file within the workspace, view the status of the file.
 - if the file is unstaged, git diff views the difference between the current workspace and last known snapshot:

```
$ git diff -- data.txt
diff --git a/data.txt b/data.txt
index 15b81a8..a5c20b1 100644
--- a/data.txt
+++ b/data.txt
@@ -1,2 +1,2 @@
-Git is open-source software.
-R is open-source software.
+R is software.
+R is free software.
```

Change a Local Repository - View Changes

- ❖ After modifying the file within the workspace, view the status of the file.
 - if the file is staged already, have to add the --cached parameter:

```
$ git add data.txt

$ git diff data.txt

$ git diff --cached data.txt
diff --git a/data.txt b/data.txt
index 15b81a8..a5c20b1 100644
--- a/data.txt
+++ b/data.txt
@@ -1,2 +1,2 @@
-Git is open-source software.
-R is open-source software.
+R is software.
+R is free software.
```

Change a Local Repository - View history

- ❖ Firstly, we add the modified file data.txt and commit it to the repository.

```
$ git add data.txt  
$ git commit -m "Add free to data"
```

- ❖ Note: Under Windows, sometimes adding a file to staging outputs the following warning:

```
$ git add data.txt  
warning: LF will be replaced by CRLF in data.txt.  
The file will have its original line endings in your working  
directory.
```

- ❖ It doesn't matter, just ignore it. You can enforce this behavior by issuing the command:

```
$ git config core.autocrlf true
```

Change a Local Repository - View history

- ❖ If we don't want too much information, output can be controlled by the parameters --oneline, or to see the full SHA-1 commit IDs, use --pretty=oneline.

```
$ git log --pretty=oneline  
96f01db1d6e026b7284a7d1267c1adb70b20aa42 Add free to data  
063624c7165dc5a6c6f5949ab7cc82a7d8a28036 delete test.txt  
ed1d0f8ad270606bda51290a1bfed37f40f15469 Commit more tests  
4d8db1c6129c836e48c36d265bcb938bc07abf21 Create three more  
files.  
a638a8aab1a396c6cbd8ab425e8a8976af498dfb Add new data.
```

- ❖ Note: Output information may not look exactly as above, the important part is that only the commit IDs and description are printed.

Change a Local Repository - Rename a file

- ❖ Git can track file renames and also handle moving ('mv') files and directories about.

```
$ git mv www.txt www2.txt // make sure www.txt was committed
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    www.txt -> www2.txt
...
...
```

- ❖ Note: Note how the rename automatically added itself to the staging area. Now commit it.

Hide files from Git

- ❖ Sometimes, we want to tell Git to ignore certain files. These could be generated files, files with secret keys, or special IDE files that should not be committed. The `.gitignore` exists to hide these kind of files and prevent mistakes.

```
$ echo ".DS_Store" > .gitignore          // Ignore common OSX file.  
$ echo "Thumbs.db" > .gitignore          // Ignore common Windows  
file.  
$ git add .gitignore; git commit -m "add gitignore file"  
$ git status
```

- ❖ Note: `.gitignore` files support a rich set of matching patterns, to match multiple files at any folder depth.

Outline

- ❖ **Set up Git and GitHub**
- ❖ **Introduction to Git**
 - **Creating a Git Repository**
 - **Manipulating files**
- ❖ **Introduction to GitHub**
 - **Lightning Tour of Github**
 - **Create a Remote Repository**

Sharing Secrets

- ❖ If sticking with the command line, need to manually configure SSH keys:
<https://github.com/settings/ssh/audit>
- ❖ Or, click on the gear icon in the upper corner:

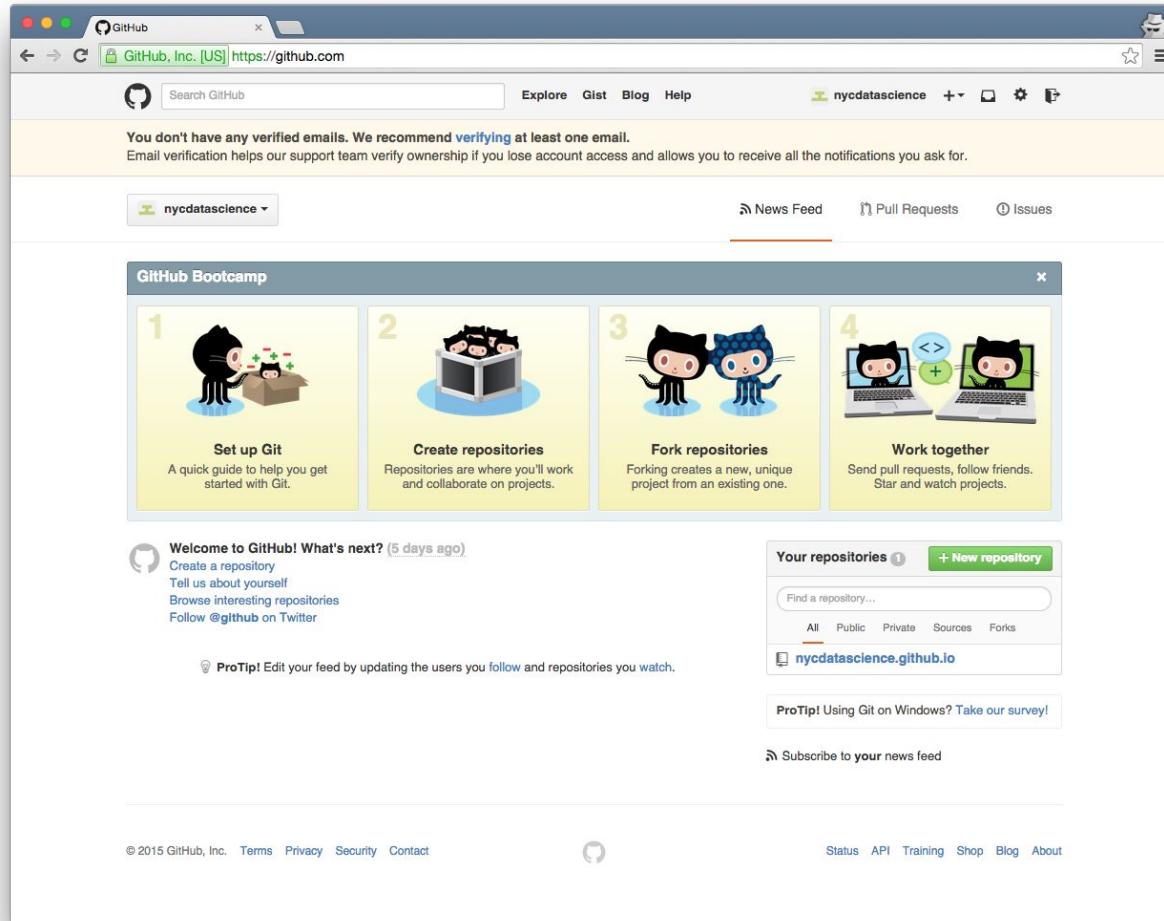


Adding SSH Keys to Github

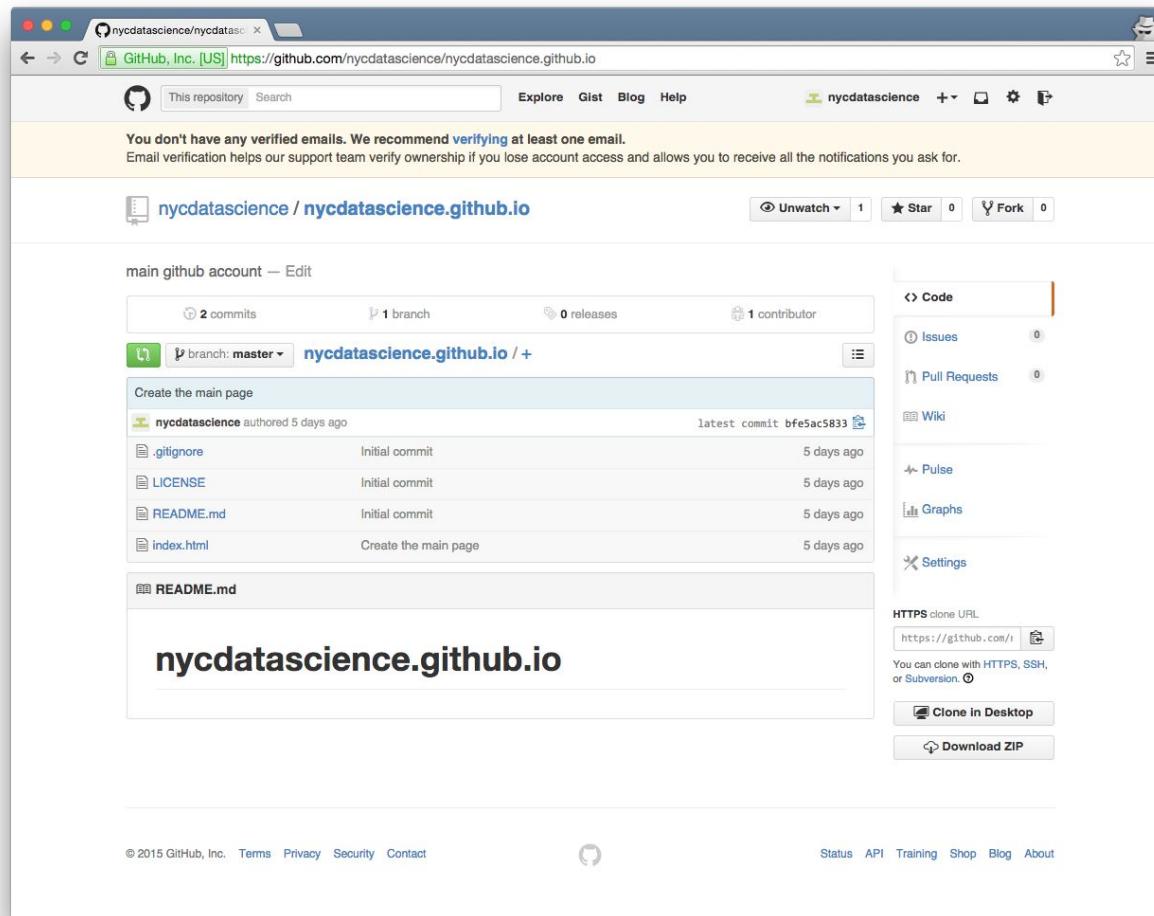
- ❖ Click 'generating SSH keys' link in the settings UI.

The screenshot shows a web browser window for GitHub, Inc. [US] at <https://github.com/settings/ssh>. The page displays a sidebar with links: Personal settings, Profile, Account settings, Emails (with a warning icon), Notification center, Billing, SSH keys (which is the active tab), Security, and Applications. The main content area has a yellow banner stating, "You don't have any verified emails. We recommend [verifying](#) at least one email. Email verification helps our support team verify ownership if you lose account access and allows you to receive all the notifications you ask for." Below the banner, there is a section titled "SSH Keys" with a button labeled "Add SSH key". A message states, "There are no SSH keys with access to your account."

Quick Feature Tour - Main Page



Quick Feature Tour - Personal GitHub Page



Quick Feature Tour - Organization Page

The screenshot shows a GitHub organization page for "The Apache Software Foundation". The page features a header with the Apache logo and navigation links for Explore, Gist, Blog, and Help. A sidebar on the left lists repositories: "infrastructure-puppet", "incubator-nifi", "ambari", and "pdfbox". Each repository entry includes a thumbnail, the name, a star icon, a commit count, and a last updated timestamp. To the right of the repositories is a "People" section displaying a grid of 156 profile pictures. At the bottom of the page is a footer with the URL "https://github.com/apache".

Quick Feature Tour - Forking Projects

Mirror of Apache Hadoop

9,837 commits | 131 branches | 199 releases | 40 contributors

branch: trunk +

HDFS-7603. The background replication queue initialization may not le... · Kihwal Lee · latest commit 89b07490f8

dev-support · HADOOP-11473. test-patch says "-1 overall" even when all checks are +1 · 17 days ago

hadoop-assemblies · HADOOP-10530 Make hadoop build on Java7+ only (stevel) · 2 months ago

hadoop-client · HADOOP-11412 POMs mention "The Apache Software License" rather than "... · a month ago

hadoop-common-project · HADOOP-9907. Webapp http://hostname:port/metrics link is not working.... · 2 hours ago

hadoop-dist · HADOOP-11268. Remove no longer supported activation properties for pa... · 3 months ago

hadoop-hdfs-project · HDFS-7603. The background replication queue initialization may not le... · an hour ago

hadoop-mapreduce-project · HADOOP-9907. Webapp http://hostname:port/metrics link is not working.... · 2 hours ago

hadoop-maven-plugins · HADOOP-11419 improve hadoop-maven-plugins. (Hervé Boute my via stevel) · 4 days ago

hadoop-minicluster · HADOOP-11412 POMs mention "The Apache Software License" rather than "... · a month ago

hadoop-project-dist · Merge from trunk to branch · 6 months ago

hadoop-project · HADOOP-10574. Bump the maven plugin versions too -moving the numbers ... · 3 hours ago

hadoop-tools · HDFS-7566. Remove obsolete entries from hdfs-default.xml (Ray Chiang ...) · 2 days ago

hadoop-yarn-project · HADOOP-9907. Webapp http://hostname:port/metrics link is not working.... · 2 hours ago

.gitattributes · HADOOP-10040. svn propset to native line endings on Windows files. Co... · a year ago

.gitignore · HADOOP-10714. AmazonS3Client.deleteObjects() need to be limited to 10... · 3 months ago

BUILDING.txt · HADOOP-11428. Remove obsolete reference to Cygwin in BUILDING.txt. Co... · a month ago

LICENSE.txt · HADOOP-11184. Update Hadoop's lz4 to r123 (cmccabe) · 4 months ago

Quick Feature Tour - Forking Projects

The screenshot shows a GitHub fork page for the repository `nycdatascience/hadoop`. The repository is a mirror of Apache Hadoop. The main statistics are 9,837 commits, 131 branches, 199 releases, and 40 contributors. The current branch is `trunk`. A list of recent commits is displayed, all of which are merged from the `apache/trunk` branch. The commits are as follows:

Author	Commit Message	Time Ago
Kihwal Lee	HADOOP-11473. test-patch says "-1 overall" even when all checks are +1	17 days ago
dev-support	HADOOP-10530 Make hadoop build on Java7+ only (stevel)	2 months ago
hadoop-assemblies	HADOOP-11412 POMs mention "The Apache Software License" rather than "..."	a month ago
hadoop-client	HADOOP-9907. Webapp http://hostname:port/metrics link is not working....	2 hours ago
hadoop-common-project	HADOOP-11266. Remove no longer supported activation properties for pa...	3 months ago
hadoop-dist	HADOOP-7603. The background replication queue initialization may not lo...	an hour ago
hadoop-hdfs-project	HDFS-7603. The background replication queue initialization may not lo...	2 hours ago
hadoop-mapreduce-project	HADOOP-9907. Webapp http://hostname:port/metrics link is not working....	4 days ago
hadoop-maven-plugins	HADOOP-11419 improve hadoop-maven-plugins. (Hervé Bouteemy via stevel)	4 days ago
hadoop-minicluster	HADOOP-11412 POMs mention "The Apache Software License" rather than "..."	a month ago
hadoop-project-dist	Merge from trunk to branch	6 months ago
hadoop-project	HADOOP-10574. Bump the maven plugin versions too -moving the numbers ...	3 hours ago
hadoop-tools	HDFS-7566. Remove obsolete entries from hdfs-default.xml (Ray Chiang ...)	2 days ago
hadoop-yarn-project	HADOOP-9907. Webapp http://hostname:port/metrics link is not working....	2 hours ago
.gitattributes	HADOOP-10040. svn propset to native line endings on Windows files. Co...	a year ago
.gitignore	HADOOP-10714. AmazonS3Client.deleteObjects() need to be limited to 10...	3 months ago
.github/workflows	HADOOP-1128. Remove obsolete reference to Cygwin in BUILDING.txt. Co...	a month ago

Quick Feature Tour - Pull Requests

The screenshot shows the GitHub interface for the `apache/hadoop` repository. The title bar indicates the current view is "Pull Requests". The main header includes the repository name "apache / hadoop", a star count of 328, and a fork count of 399. Below the header, there are tabs for "Pull requests" (which is selected), "Labels", and "Milestones". A search bar allows filtering by "is:pr is:open". A green button labeled "New pull request" is visible. The main content area displays a list of 8 open pull requests:

- #13 Merging from apache trunk (opened on Dec 22, 2014)
- #12 Hdfs ec (opened on Dec 3, 2014)
- #8 Branch 2 (opened on Nov 6, 2014)
- #7 YARN-1964 Launching containers from docker (opened on Oct 6, 2014)
- #6 YARN-1964 Launching containers from docker (opened on Sep 28, 2014)
- #4 Update TaskInputOutputContext.java javadoc (opened on Sep 25, 2014)
- #3 [HADOOP-10724] better interoperation with `sort -h` (opened on Sep 23, 2014)
- #1 MAPREDUCE-6096.SummarizedJob Class Improvement (opened on Sep 18, 2014)

A tooltip at the bottom left of the list area says: "ProTip! Adding no:label will show everything without a label."

At the bottom of the page, there are links for "Status", "API", "Training", "Shop", "Blog", and "About".

Quick Feature Tour - Example Branching Structure

The screenshot shows a GitHub page for the Apache Cassandra repository. The URL is <https://github.com/apache/cassandra/blob/trunk/README.asci>. The page displays the README file content, which includes a sidebar for switching branches and tags. The 'trunk' branch is selected. The content of the README file discusses partitioning and row store, and provides links to requirements and getting started guides.

Switch branches/tags

Branches

- cassandra-1.0
- cassandra-1.1
- cassandra-1.2
- cassandra-2.0
- cassandra-2.1
- trunk** (selected)

Partitioning means that Cassandra can distribute your data across multiple machines in an application-transparent matter. Cassandra will automatically repartition as machines are added and removed from the cluster.

Row store means that like relational databases, Cassandra organizes data by rows and columns. The Cassandra Query Language (CQL) is a close relative of SQL.

For more information, see the [Apache Cassandra web site](#).

Requirements

1. Java >= 1.7 (OpenJDK and Oracle JVMS have been tested)
2. Python 2.7 (for cqlsh)

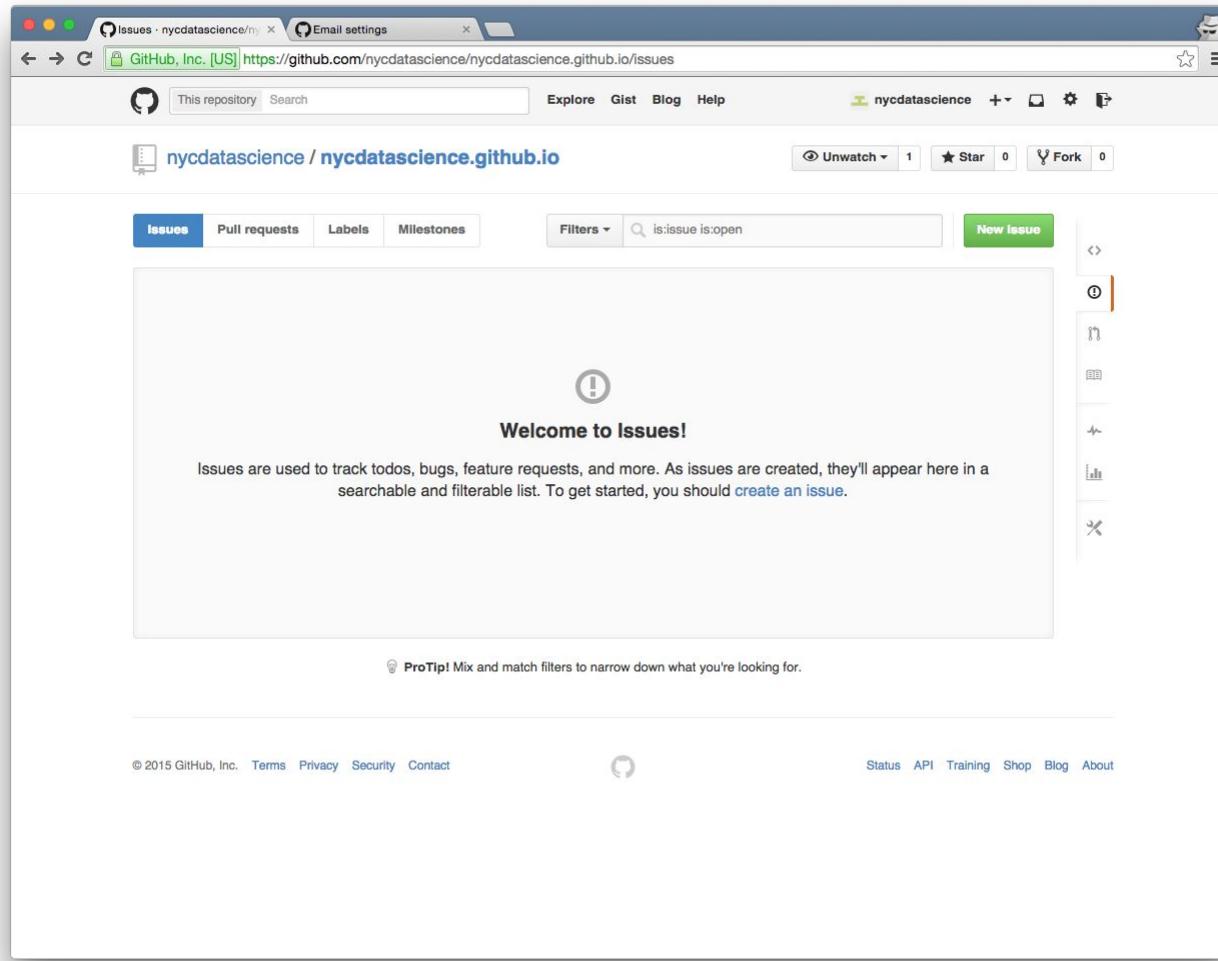
Getting started

This short guide will walk you through getting a basic one node cluster up and running, and demonstrate some simple

Quick Feature Tour - Example Tagging Structure

The screenshot shows a GitHub repository page for `cassandra`. The URL is <https://github.com/apache/cassandra/blob/trunk/README.asc>. The page title is `cassandra / README.asc`. On the left, there is a sidebar with a dropdown menu for "branch: trunk" and a list of branches and tags. The "Tags" tab is selected, showing several entries: `cassandra-1.2.2`, `cassandra-1.2.1`, `cassandra-1.2.0-rc2`, `cassandra-1.2.0-rc1`, `cassandra-1.2.0-beta3`, `cassandra-1.2.0-beta2` (which is highlighted with a blue background), `cassandra-1.2.0-beta1`, `cassandra-1.1.12`, `cassandra-1.1.11`, `cassandra-1.1.10`, `cassandra-1.1.9`, and `cassandra-1.1.8`. The main content area displays the `README.asc` file's text content, which includes information about primary keys, data distribution, and the Cassandra Query Language (CQL). Below the file content, there is a "Getting started" section with two numbered steps: "1. Java >= 1.7 (OpenJDK and Oracle JVMs have been tested)" and "2. Python 2.7 (for cqlsh)".

Quick Feature Tour - Issue Tracking



Quick Feature Tour - Issue Tracking

The screenshot shows the GitHub issue creation interface for the repository `nycdatascience/nycdatascience.github.io`. The page title is "New Issue · nycdatascience". The main content area displays a single issue titled "Missing student pages for G001". The issue details include:

- Assignee: No one is assigned
- Milestone: No milestone
- Description:

Missing Student Profiles

This issue will be solved in class as part of the tutorial.
- Labels: enhancement (selected), bug, duplicate, help wanted, invalid, question, wontfix
- Buttons: Write, Preview, Markdown supported, Edit in fullscreen, Attach images, Submit new issue

At the bottom of the page, there are links for GitHub terms and conditions, and navigation links for Status, API, Training, Shop, Blog, and About. The URL in the address bar is <https://github.com/nycdatascience/nycdatascience.github.io/milestones>.

Quick Feature Tour - Issue Tracking

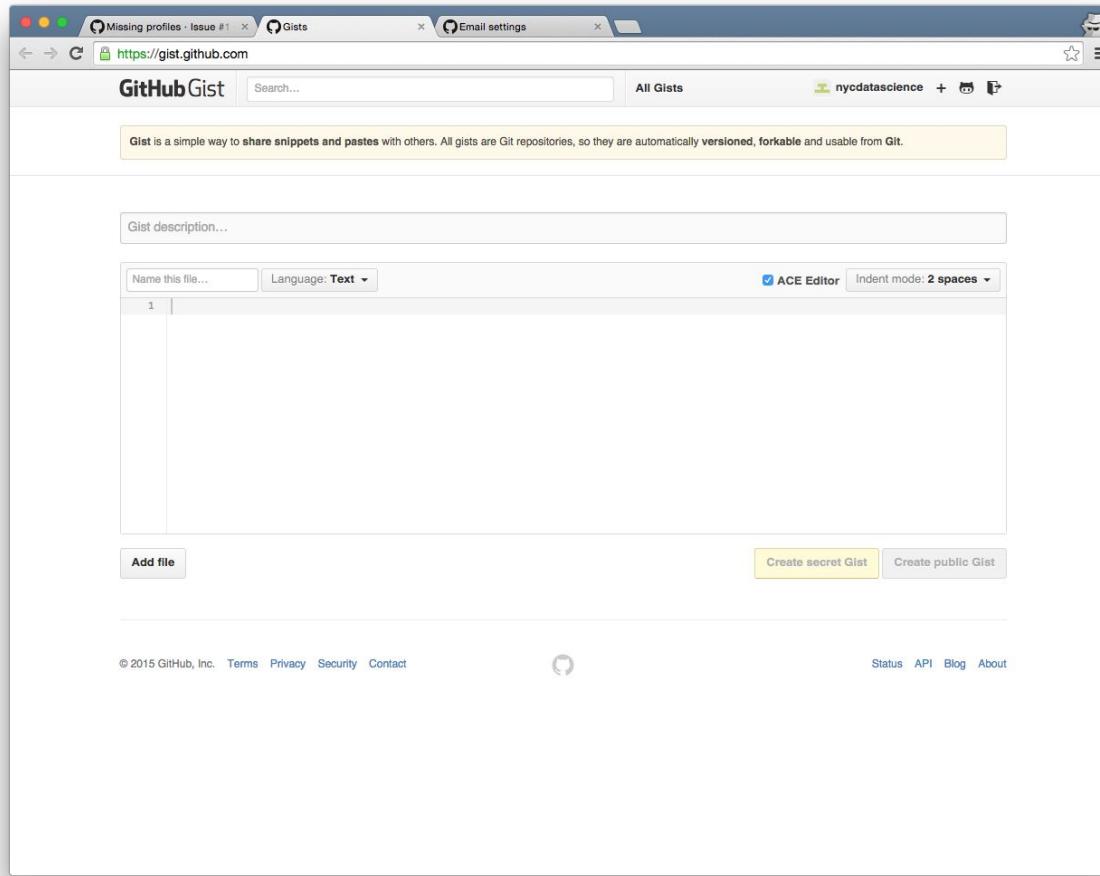
The screenshot shows a GitHub issue page for a repository named `nycdatascience / nycdatascience.github.io`. The issue is titled "Missing profiles #1". A green button indicates it was opened by `nycdatascience` a minute ago with 0 comments. The main content of the issue is a comment from `nycdatascience` stating "Missing Student Profiles" and noting that the issue will be solved in class as part of the tutorial. This comment was added a minute ago. The issue has been labeled "enhancement" and assigned to the "M1" milestone. The sidebar on the right provides options to edit the issue, change labels, set milestones, assignees, and notifications. It also shows that there is 1 participant and a link to unsubscribe from notifications. At the bottom, there is a comment input field with "Write" and "Preview" tabs, a note about Markdown support, and buttons for "Close issue" and "Comment".

Quick Feature Tour - Wiki Pages

The screenshot shows a GitHub repository page for `thinkaurelius/titan`. The main content area displays the `Home` page of the wiki. It features a large image of the Greek titan Atlas holding up the celestial spheres, with the word "TITAN" written in large, bold, black letters below it. A text block describes Titan as a distributed graph database optimized for storing and querying graphs over a cluster of machines, mentioning its support for various database technologies like Apache Cassandra, HBase, and Oracle BerkeleyDB. Below the text is a red "DOWNLOAD" button with a white "TINKERPOP" logo. On the right side, there is a sidebar titled "Pages" containing a list of wiki pages such as Acknowledgements, Advanced Blueprints, and Data Caching. At the bottom of the sidebar, there is a link to "Show 35 more pages...".

Quick Feature Tour - Gist

- ❖ Gist: Sharing quick snippets of code - <https://gist.github.com/>



Disk Quotas

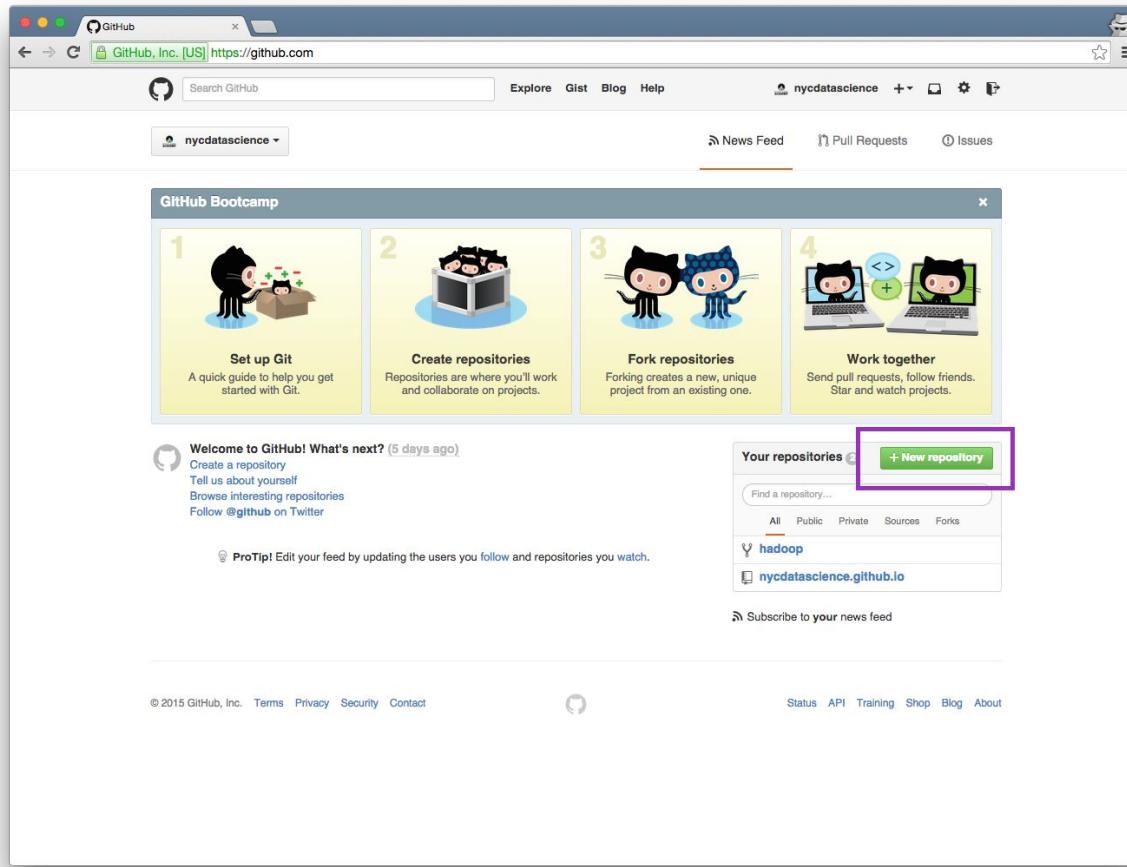
- ❖ How much information can you store on GitHub?
- ❖ Try to keep large datasets out of Git, and try to keep repositories below 100 MB. Instead, link to where a person can find a secure copy. See this help article.
- ❖ Also, for really large data sets, consider using BitTorrent, GoogleDrive, or DropBox with SHA-256 checksums for security.

Outline

- ❖ **Set up Git and GitHub**
- ❖ **Introduction to Git**
 - **Creating a Git Repository**
 - **Manipulating files**
- ❖ **Introduction to GitHub**
 - **Lightning Tour of Github**
 - **Create a Remote Repository**

Create a Remote Repository

- ❖ Look for the **+ New repository** button.



Create a Remote Repository

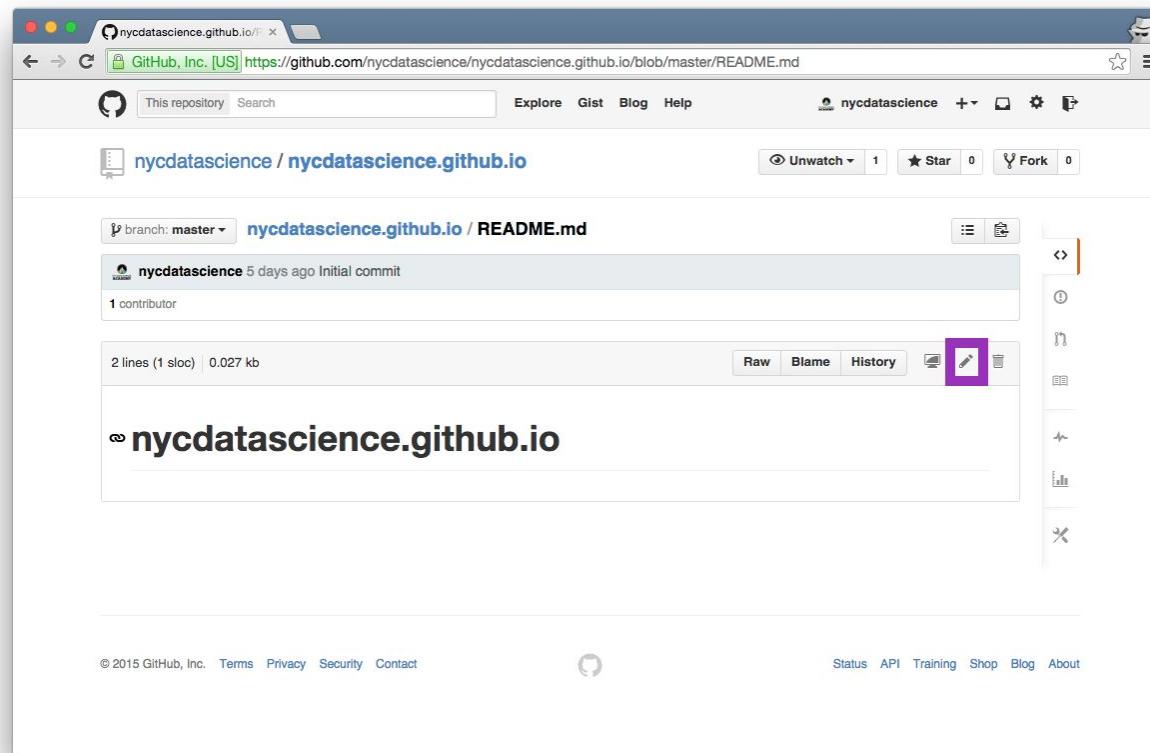
- ❖ For the repository name, create the domain name for your personal page, using your personal github username, in the following format
your_username.github.io
- ❖ username: nycdatascience

```
repository name = nycdatascience.github.io
```

- ❖ Click on the checkbox Initialize this repository with a README. The license choice is up to you.
- ❖ Click the Create repository button.

Edit From the Site

- ❖ Once the new repository has been created, click on your repository's README.md file. For example, this was the NYC Data Science README:



Edit From the Site

- ❖ Markdown <https://help.github.com/articles/github-flavored-markdown/>
- ❖ Change the contents of the README to have a good description, optionally using markdown like in the following:

```
# my_user_name.github.io
```

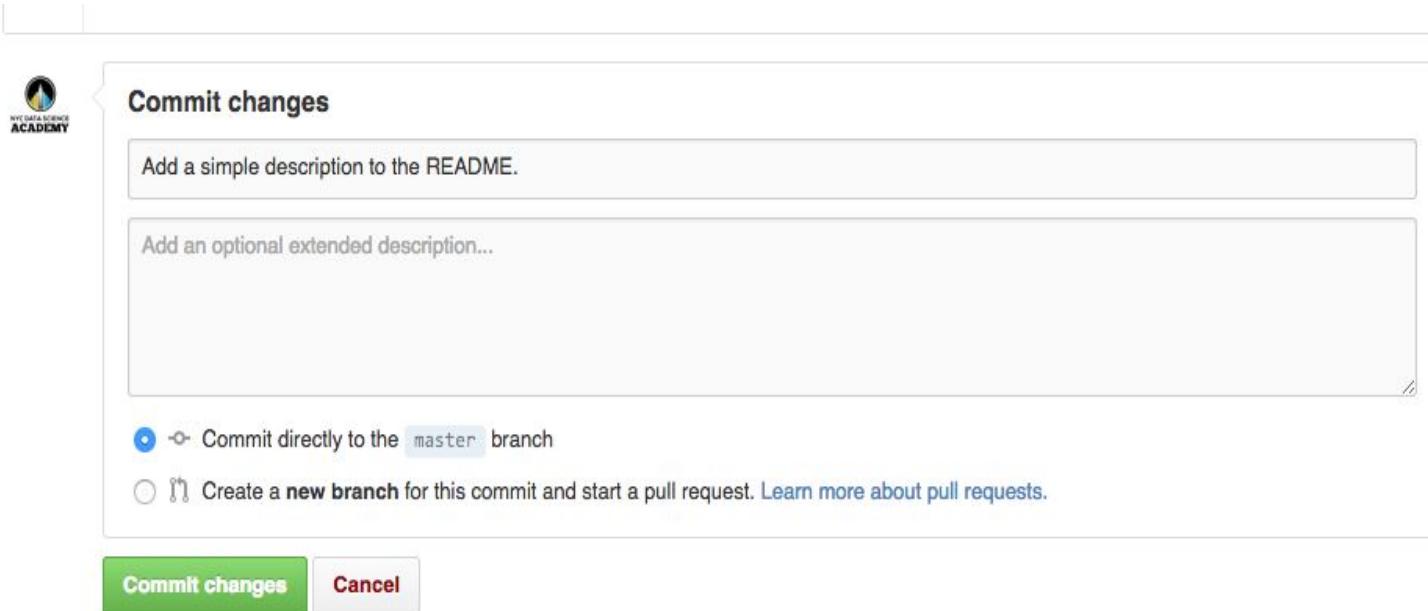
```
Copyright @ My Name
```

```
## Description
```

```
This will be the main portfolio page for the My Name. I am  
currently located in New York City.
```

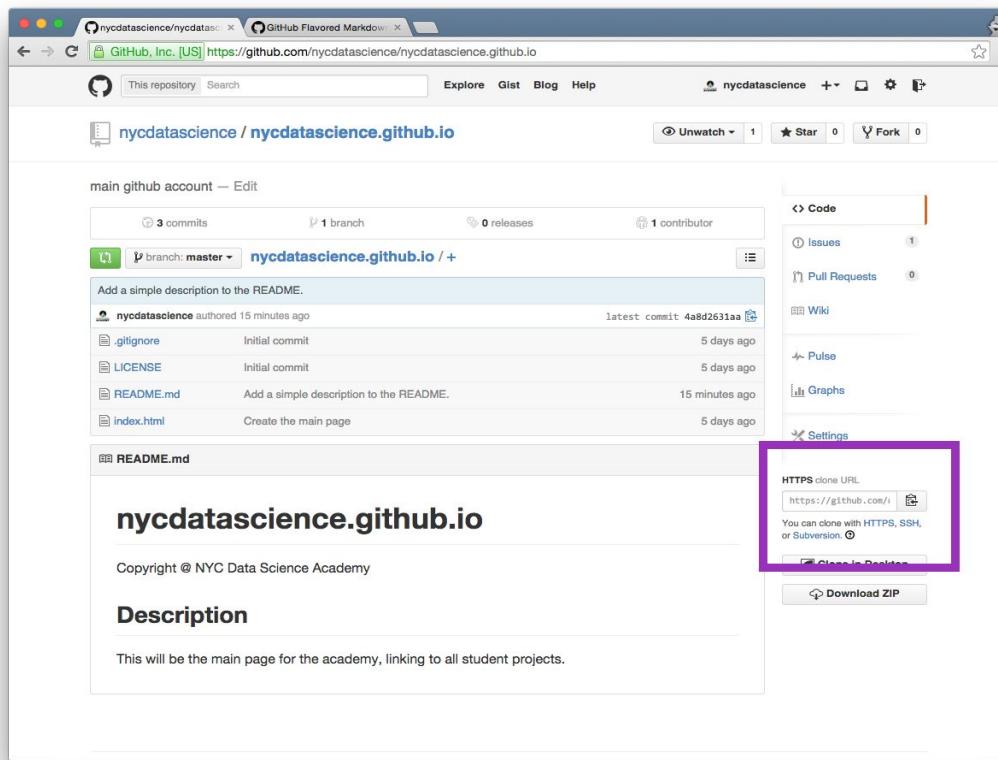
Edit From the Site

- ❖ Scroll down, add a commit message, and hit the **Commit changes** button. Leave the change to be made against the master branch.



Cloning

- ❖ Go back to the main page for the repository, and look for the remote HTTPS/SSH address of your repository:



Cloning

- ❖ We want to clone the repository using one of the following addresses, but your repository.
- ❖ HTTPS:

```
https://github.com/nycdatascience/nycdatascience.github.io.git
```

- ❖ Alternatives:

```
SSH: git@github.com:nycdatascience/nycdatascience.github.io.git
```

```
Subversion: https://github.com/nycdatascience/nycdatascience.github.io
```

Cloning

- ❖ Copy the HTTPS address (the default), and open the local command line.

```
$ git clone https://github.com/<username>/<username>.github.io.  
git  
Cloning into 'nycdatascience.github.io'...  
remote: Counting objects: 11, done.  
remote: Compressing objects: 100% (9/9), done.  
remote: Total 11 (delta 2), reused 0 (delta 0)  
Unpacking objects: 100% (11/11), done.  
Checking connectivity... done.
```

New Local Repository

- ❖ Change directories into the folder that was cloned locally. Be sure to substitute your username in the command below.

```
$ cd <username>.github.io  
$ ls  
README.md
```

Remotes

- ❖ Repositories cloned from GitHub automatically have the 'remote' setting put into the local configuration.

```
$ git remote -v
origin  https://github.com/nycdatascience/nycdatascience.github.
io.git (fetch)
origin  https://github.com/nycdatascience/nycdatascience.github.
io.git (push)
```

- ❖ Note: Run a git status to see what the set branch is.

Remotes

- ❖ When the local repository and remote repository are both created independently, how can you associate them?
- ❖ Associate the local and remote database versions, by specifying a simple name instead of the full remote address. Use the command
 - `$ git remote add [simple name] [url]`
- ❖ In the previous example, this was not necessary as the origin was set already.

```
// https protocol  
$ git remote add origin https://github.  
com/nycdatascience/nycdatascience.github.io.git  
// or ssh protocol  
$ git remote add origin git@github.com:  
nycdatascience/nycdatascience.github.io.git
```

Pull

- ❖ Say you want check for the latest changes on the remote server. First, go back to GitHub, and modify the README.md file again. Then type one of the following.

```
$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/nycdatascience/nycdatascience.github.io
  4a8d263..f527c76  master      -> origin/master
Updating 4a8d263..f527c76
Fast-forward
 README.md | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Push

- ❖ Now we want to publish a local edit back to the remote GitHub repository. Create a new file in the local repository called `index.html`, with some sample content.

```
Hello, World!
```

Push

- ❖ Commit the new content to your local repository.

```
$ git add index.html  
$ git commit -m "Add the primary webpage for the portfolio."
```

Push

- ❖ Commit the new content to your remote personal repository's master branch.

```
$ git push origin master
```

- ❖ If there are errors, then there is a problem with your setup. If not, wait a few seconds, and then refresh your GitHub page. Visit your actual portfolio page at <https://username.github.io/> where username is your Github username.