Ekaba Bisong
Programming in C++
University of Calabar

Tutorial Notes #3
May 08, 2015

Setting-up Git and GitHub

What is Version Control
Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

What is Git
Git is version control software, which means it manages changes to a project without overwriting any part of that project.

What is GitHub
GitHub makes Git easier to use in two ways. First, if you download the GitHub software to your computer, it provides a visual interface to help you manage your version-controlled projects locally. Second, creating an account on GitHub.com brings your version-controlled projects to the Web, and ties in social network features for good measure.

Download and Install Git on Windows
   Download Git
   - Go to the following website and click on the Windows download link:
       o   http://git-scm.com/downloads
   Install Git
   - Once the file is done downloading, open it up to begin the Git installation
   - Go with the default options at each step of the installation
   - Once the install is complete, hit the "Finish" button (you may want to uncheck the box next to "Review ReleaseNotes.rtf")
   Open Git Bash
   - Find a program called Git Bash, which is the command line environment for interacting with Git
   - It should be located in the Git directory within your Start Menu (or in the directory into which Git was installed)
   - Once Git Bash opens, you'll see a short welcome message followed by the name of your computer and a dollar sign on the next line
   - The dollar sign means that it's your turn to type a command
   Configure Username and Email
   - Each commit to a Git repository will be "tagged" with the username of the person who made the commit
   - Enter the following commands in Git Bash, one at a time, to set your username and email:

- o $ git config --global user.name "Your Name Here"
- o $ git config --global user.email "your_email@example.com"
- Make sure there are 2 dashes side-by-side before the word "global"
- You'll only have to do this once, but you can always change these down the road using the same commands
- Now type the following to confirm your changes (they may be listed toward the bottom):
  - o $ git config --list
- Make sure there are 2 separate dashes (side-by-side) before the word "global"
- Now that Git is set up on your computer, we're ready to move on to GitHub, which is a web-based platform that lets you do some pretty cool stuff
- Once GitHub is up and running, we'll show you how to start using these tools to your benefit

GitHub
What is GitHub?
- "GitHub is a web-based hosting service for software development projects that use the Git revision control system."
- Allows users to "push" and "pull" their local repositories to and from remote repositories on the web
- Provides users with a homepage that displays their public repositories
- Users' repositories are backed up on the GitHub server in case something happens to the local copies
- Social aspect allows users to follow one another and share projects

Set Up a GitHub Account
- Go to the GitHub homepage at https://github.com/
- Enter a username, email, and password and click "Sign up for GitHub"
- ***IMPORTANT: Use the same email address that you used when setting up Git***

Recap: Git vs. GitHub
- You don't need GitHub to use Git
- Git = Local (on your computer); GitHub = Remote (on the web)
- GitHub allows you to:
  - o Share your repositories with others
  - o Access other users' repositories
  - o Store remote copies of your repositories (on GitHub's server) in case something happens to your local copies (on your computer)

Creating a GitHub Repository
- Two methods of creating a GitHub repository:
  - o Start a repository from scratch
  - o "Fork" another user's repository
- ***NOTE: A repository is often referred to as a "repo"***

Start a Repository From Scratch
- Go to your profile page (https://github.com/your-user-name-goes-here/) and click on "Create a new repo" in the upper righthand corner of the page
- Create a name for your repo and type a brief description of it
- Select "Public" (Private repos require a paid [or education] account)
- Check the box next to "Initialize this repository with a README"
- Click the "Create repository" button
- Congratulations! You've created a GitHub repository.

Creating a Local Copy
Now you need to create a copy of this repo on your computer so that you can make changes to it.
- Open Git Bash
- Create a directory on your computer where you will store your copy of the repo:
  - $ mkdir ~/test-repo
- *Note:* The tilde (~) symbol refers to your "home" directory, so this will create a directory called test-repo in your home directory
- Navigate to this new directory using the following command (cd stands for "change directory"):
  - $ cd ~/test-repo
- Initialize a local Git repository in this directory
  - $ git init
- Point your local repository at the remote repository you just created on the GitHub server
  - $ git remote add origin https://github.com/your-user-name-goes-here/test-repo.git
- ***Tip: You can easily copy the URL of your Github repository with the copy to clipboard button***

Fork a Another User's Repository
The second method of creating a respository is to make a copy of someone else's This process is called "forking" and is an important aspect of open-source software development
- Begin by navigating to the desired repository on the GitHub website and click the "Fork" button
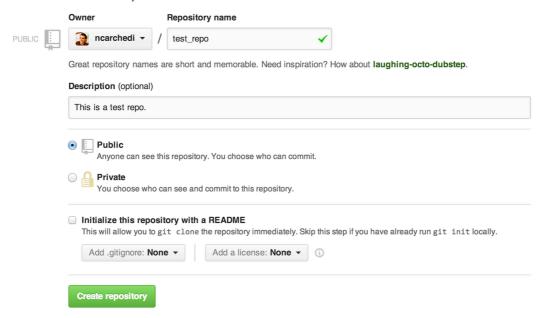
Clone the Repo
- You now have a copy of the desired respository on your GitHub account
- Need to make a local copy of the repo on your computer
- This process is called "cloning" and can be done using the following command from either Git Bash
  - $ git clone https://github.com/your-user-name-goes-here/repo-name-here.git
- ***NOTE:*** This will clone the repository into your current directory. You can always tell what directory you are in by typing pwd (for "print working directory") from the command line.

An example workflow
- Feel free to create a test repo of your own (on your GitHub account) and follow along
- This is not meant to be an exhaustive introduction to all Git and GitHub features
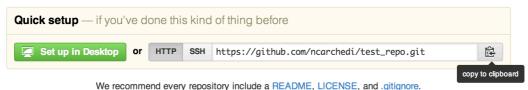- Best way to learn Git and GitHub are by using them

Create a new repo on GitHub
- Leave the box next to "Initialize this repository with a README" **unchecked**, since we will add one later

Copy repo URL to clipboard
- After clicking "create repository", you will get a screen with some basic instructions for getting started
- They are a good starting point, but we won't follow them exactly here
- Copy the URL displayed below to your clipboard (yours will be slightly different since it depends on your username and repo name)

Create a new local directory
- Open Git Bash and create a new directory on your computer
- "Step inside" of this new directory with the cd command
- mkdir stands for "make directory", cd stands for "change directory", and ls stands for "list"
- Since we get no output after typing ls, we can see that the directory we created is still empty

```
~ $ mkdir ~/test_repo
~ $ cd ~/test_repo/
~/test_repo $ ls
~/test_repo $
```

Create a new file in directory
- Open your favorite text editor and create a new text file
- Make sure to save it in the directory you just created

```
untitled
1    This is a new text file that I'm creating to show you how Git works....
```

Create a new repo locally
- We already created a **GitHub** repository, but we still need to create a **Git** repository locally on your computer
- We can see that our new file is now in our chosen directory with ls
- *git init* initializes a Git repo in our current directory
- *git status* is a helpful command that we'll make frequent use of

```
~/test_repo $ ls
test_file.txt
~/test_repo $ git init
Initialized empty Git repository in /Users/nicholasacarchedi/test_repo/.git/
~/test_repo $ git status
On branch master

Initial commit

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

        test_file.txt

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

Stage file for commit
- Notice that our newly created file falls under "untracked files" when we look at *git status*
- Use *git add* to tell Git that we want it to start "paying attention" to this file
  - Could have used *git add test_file.txt* for the same result, but *git add .* is often easier if you are okay tracking all currently untracked files

```
~/test_repo $ git add .
~/test_repo $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test_file.txt

~/test_repo $ 
```

Commit changes

- Using *git commit* with a *-m* after it tells Git that whatever follows in double quotes is the message that we want to attach to this round of changes
- Another call to *git status* confirms that there is nothing new to commit (since our first commit)

```
~/test_repo $ git commit -m "first commit, which includes a test text file"
[master (root-commit) d7126e2] first commit, which includes a test text file
 1 file changed, 1 insertion(+)
 create mode 100644 test_file.txt
~/test_repo $ git status
On branch master
nothing to commit, working directory clean
~/test_repo $ 
```

Check log

- *git log* shows us a history of all commits
- So far, there's only one

```
~/test_repo $ git log
commit d7126e2ab3ec1d6df4cac1c40d0f9cc239888017
Author: Nick Carchedi <nick.carchedi@gmail.com>
Date:   Mon Jan 20 16:58:21 2014 -0500

    first commit, which includes a test text file
~/test_repo $ 
```

Add link to remote repo

- We now have a remote repo on GitHub's servers and a local repo on our computer, but they still don't know about each other
- To establish a link between the two, we paste the URL copied earlier from

GitHub as follows

```
~/test_repo $ git remote add origin https://github.com/ncarchedi/test_repo.git
~/test_repo $ git remote -v
origin  https://github.com/ncarchedi/test_repo.git (fetch)
origin  https://github.com/ncarchedi/test_repo.git (push)
~/test_repo $
```

- **git remote -v** shows us that our GitHub repo is now set up as a "remote" repository for our local repo, which allows the two repos to communicate

Push changes to GitHub
- We want our GitHub repo to reflect the changes we've made locally (i.e. to contain our new text file)
- **git push -u** origin master tells Git to push our changes to the "master" (or main) branch of the "origin" (or primary) remote
- You only need to include the **-u** origin master once, as Git will remember this configuration for future pushes
  - **git push** then becomes sufficient, assuming you don't want to do anything fancy

```
~/test_repo $ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ncarchedi/test_repo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
~/test_repo $
```
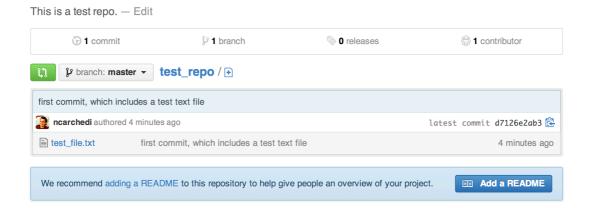
Check status
- Check status again for piece of mind
- Notice that it confirms *Your branch is up-to-date with 'origin/master'*

```
~/test_repo $ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
~/test_repo $
```

Check GitHub
- We want to make sure that our changes made it to GitHub safely and indeed they did
- Our text file shows up in the file list

This is a test repo. — Edit

⊙ **1** commit	⑂ **1** branch	◌ **0** releases	⑂ **1** contributor

↕   ⑂ branch: **master** ▾   **test_repo** / ⊞

first commit, which includes a test text file

ncarchedi authored 4 minutes ago   latest commit `d7126e2ab3` 📋

📄 test_file.txt   first commit, which includes a test text file   4 minutes ago

We recommend adding a README to this repository to help give people an overview of your project.   📖 **Add a README**
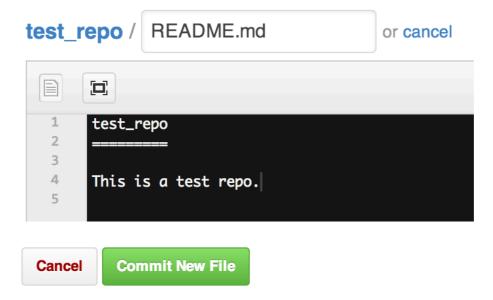
Add README file from GitHub
- How can we can "pull" changes from a remote repository to our local repository?
- To illustrate, we'll add and edit a README file directly from the GitHub website
- A more common scenario would be that a collaborator makes changes to a shared repository and you want to incorporate those changes into your local repo
- Click on the "Add a README" button on your GitHub repo page, under the list of files

We recommend adding a README to this repository to help give people an overview of your project.   📖 **Add a README**

Edit README file from GitHub
- Put anything you want in the README, then press "Commit New File" to commit the file to your GitHub repo
- **Note:** README files written in Markdown will render in HTML on your repository's homepage.

test_repo /   README.md   or cancel

```
1   test_repo
2   =========
3
4   This is a test repo.
5
```

Cancel   **Commit New File**

Fetch changes from GitHub

- Now that README.md exists on GitHub, we want to "pull" it down to our local repo
- There is a *git pull* command that allows you to do this, but it's recommended that you instead use a combination of *git fetch* and *git merge*
- *git fetch* origin tells Git to fetch all changes from the "origin" (primary) remote repo, which we set up earlier with git remote add origin ...

```
~/test_repo $ git fetch origin
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/ncarchedi/test_repo
   d7126e2..d0e46d4  master     -> origin/master
~/test_repo $
```

Merge changes into local repo

- Git is now aware of all changes that have been made to the remote (i.e. GitHub) repo
- Still need to incorporate these changes into our local repo
- *git branch -a* shows us that we now have two "branches" stored on our computer: master and remotes/origin/master
- master represents the files on our local repo and remotes/origin/master represents the files we pulled from our remote repo
- Use *git merge origin/master* to incorporate the changes from our remote repo

```
~/test_repo $ git branch -a
* master
  remotes/origin/master
~/test_repo $ git merge origin/master
Updating d7126e2..d0e46d4
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 README.md
~/test_repo $
```

Check status

- A quick call to ls confirms that the README.md file is now in our local directory
- git status tells us that we have no new changes and are up-to-date with our remote repo

9

```
~/test_repo $ ls
README.md        test_file.txt
~/test_repo $ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
~/test_repo $ 
```