

Software Fundamentals Series

Workshop #1: Source Control Management Systems - Git

IEEE Computer Society Ryerson Chapter
September 18th, 2018

What is Source Control?

- Also known as revision control, source code management or version control
- The practice of managing and tracking changes to code
- A source control management system (scm)/version control system (VCS) such as git is used to manage and track changes & resolve conflicts when merging contributions made from multiple sources

Why Use Source Control?

Scenario:

- Large programming project with multiple files and many team members
- Everyone may be working on the same files at different points in time
- Main code has been delivered to production
- Team member later finds an improvement that can be made to one specific area of the code

Why Use Source Control?

- We want to be able to simply add team member changes to the main code in production while preserving everything that was untouched by each team member in this instance
- It is also useful to track who made the changes and when just in case something goes wrong
- A source control management system allows us to do this seamlessly

How?

- In general:
 - One main production version of code
 - Team member makes personal local copy of the code
 - Changes applied to local copy
 - Modified local copy submitted to main code
 - Source control system adds the changes that team member made to the main code
 - Everything else that was not changed in the main code remains the same

Why Learn SCM Systems?

- EVERY company that writes software has some form of source control
- It is an expected mandatory skill for everyone working in software
- Learning source control systems (especially Git) makes you more EMPLOYABLE

Common SCM Systems

- Git
- Subversion (SVN)
- Mercurial
- CVS
- ClearCass
- RTC

Git & GitHub

- Git:
 - Most popular open source version control system today
 - Originally created for Linux systems
 - Features easy commands and structure for source control management
- Github:
 - Web-based service for version control using Git
 - Largest host of source code in the world
 - Over 28 million users, over 57 million repositories

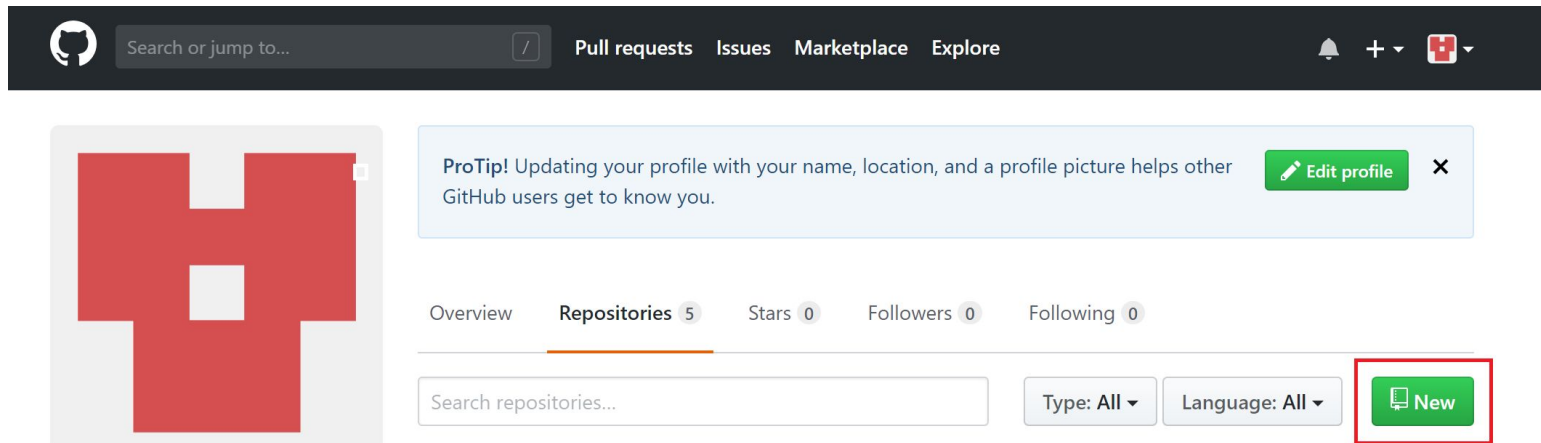
Git & GitHub Tutorial

General Workflow

*Fork (Clone) > Branch > Edit > Stage (Add) > Commit > Pull
Request > Merge*

Creating a Repository

- Start off by creating a repository inside GitHub



- Begin by pressing the “New” button highlighted in red

Creating a Repository (con't)

- Enter an appropriate name for your repo
- Ensure the repository initialized with a README
- Press "Create repository" when finished

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



ieee-ryerson-cs ▾

Repository name

softwarefundamentals ✓

Great repository names are short and memorable. Need inspiration? How about **bug-free-invention**.

Description (optional)

This is a repository for practicing!



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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

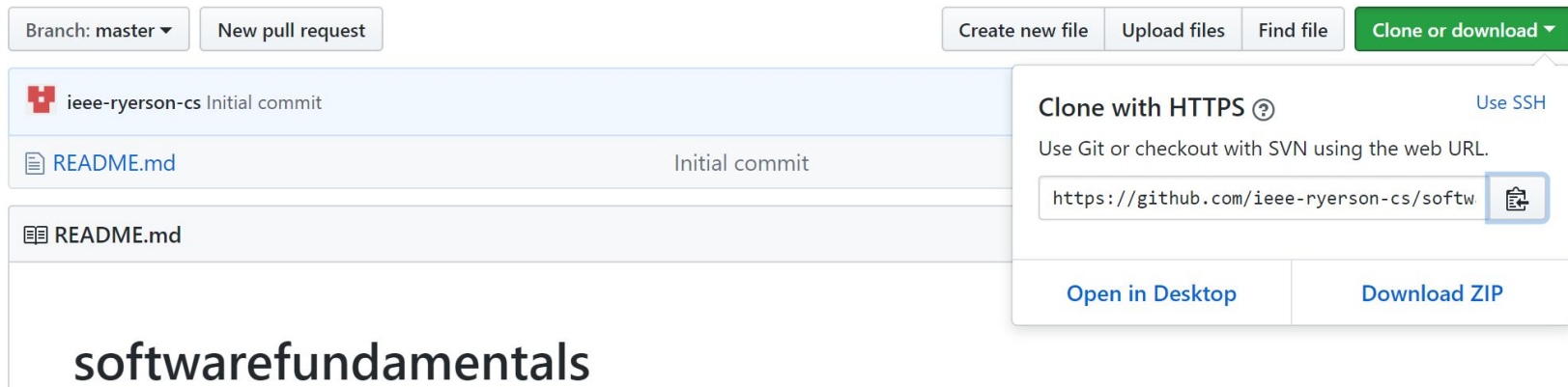
Add a license: None ▾



Create repository

Cloning the Repository

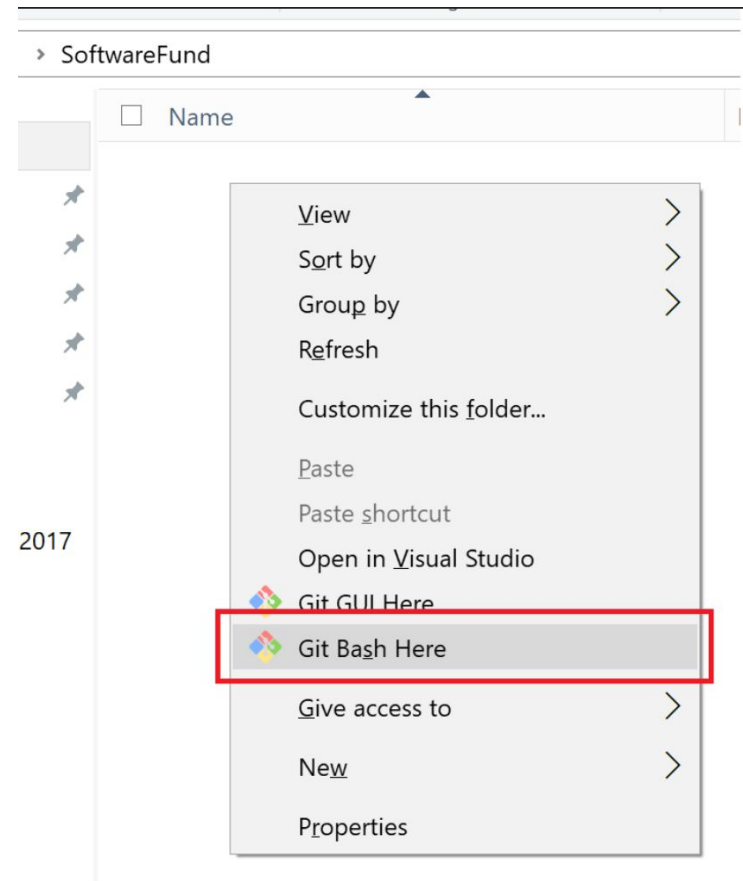
- To work on the files inside the repo, they need to be first cloned to your local machine



- Copy the HTTPS url found in your repo

Cloning the Repository (con't)

- Create an empty folder on your computer
- Open Git Bash and navigate to the new folder you just created
- Alternatively, open the folder you created and right click and press "Git Bash Here"



Cloning the Repository (con't)

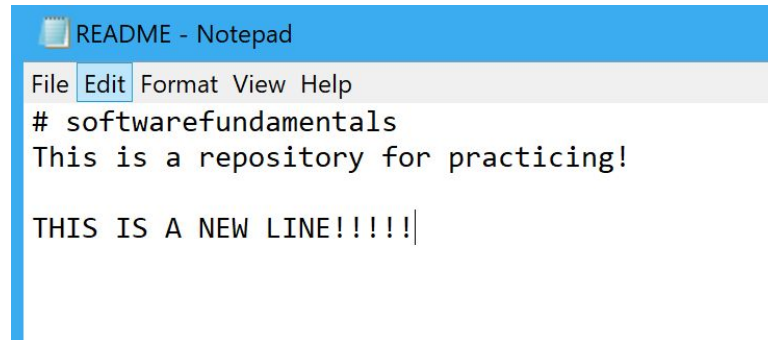
- Use the command below to clone
 - "git clone YOUR-HTTPS-URL-HERE"
 - Replacing "YOUR-HTTPS-URL-HERE" with the URL you copied earlier

```
danie@DANIEL-SURFACE MINGW64 ~/Desktop/SoftwareFund (master)  
$ git clone https://github.com/ieee-ryerson-cs/softwarefundamentals.git
```

- The README.md file from your repo should now be inside the folder

Modify the README.md

- Modify the README.md file using any text editor of your choice (i.e., Notepad)



```
README - Notepad
File Edit Format View Help
# softwarefundamentals
This is a repository for practicing!

THIS IS A NEW LINE!!!!!!|
```

- Now that, the file has been modified, the changes must be added to your GitHub repo

Status of your Local Repository

- To see what files have been modified, use the following GIT command
 - "git status"
- Since we modified that README.md file, the output should look similar to this

```
danie@DANIEL-SURFACE MINGW64 ~/Desktop/SoftwareFund/softwarefundamentals (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/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:   README.md

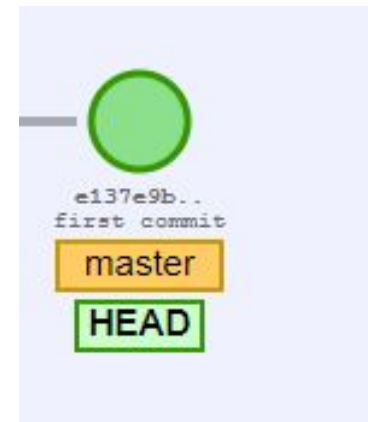
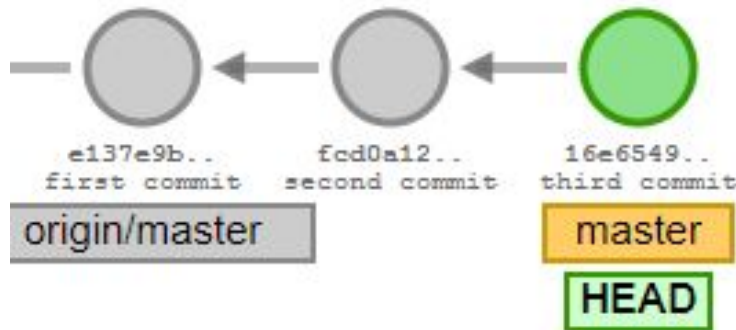
no changes added to commit (use "git add" and/or "git commit -a")
```

Adding Files to be Committed

- In the previous slide, the README.md file was shown as red with the "modified" tag
- This indicates that there was a change within the file
- To add these changes to the GitHub repo, it must first be added to the "Staging Area"
 - "git add FILE-ADDED"
 - Replace "FILE-ADDED" with the file you wish to add

Committing

- Once your modified files have been added, you can commit them
- Committing your changes creates a local “commit”
 - Commits are simply different versions of your code
 - These commits are not added to the GitHub repo until they have been pushed
- Local Repo (Left) | GitHub Repo (Right)



Committing (con't)

- Committing can be done after you have added a file to the Staging Area using the GIT command below
 - `git commit -m "YOUR-MESSAGE"`
 - Replace YOUR-MESSAGE with a short line describing the changes you made to your code

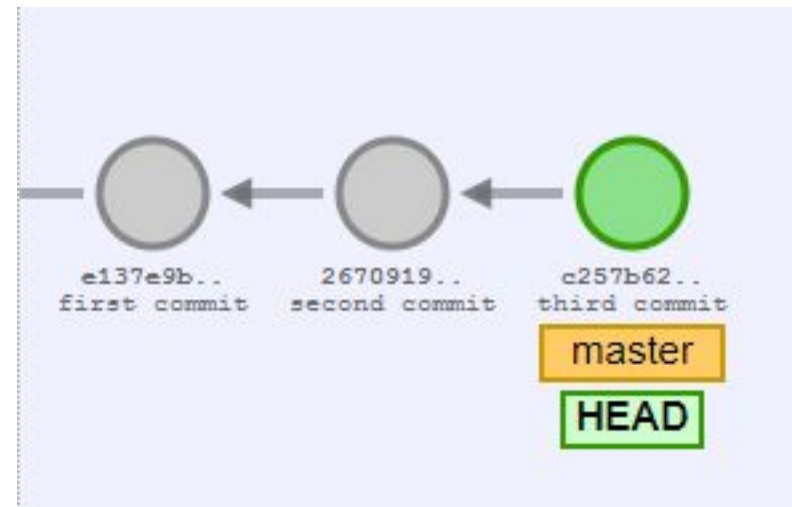
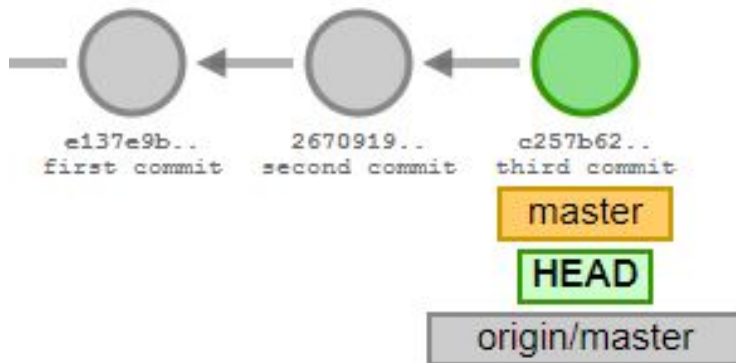
```
danie@DANIEL-SURFACE MINGW64 ~/Desktop/SoftwareFund/softwarefundamentals (master)
$ git commit -m "changed README"
[master 4921721] changed README
1 file changed, 2 insertions(+)
```

Pushing to GitHub Repository

- Once you have added all your modified files to the Staging Area, then committed those changes to a local commit, they can then be pushed to your GitHub repo
- This is done simply by using the GIT command below
 - "git push"
- You may get a window popup if you have not setup an SSH-key
 - This is fine, just enter your GitHub User and Password, and the push should continue

Pushing to GitHub Repository (con't)

- If done successfully, your Local and GitHub repo should now be identical
- Local Repo (Left) | GitHub Repo (Right)



Authentication (OPTIONAL)

- To avoid typing in your credentials every single time, you can follow the steps shown in the following link.
 - [SSH-key Guide](#)
- This will give your computer access to your entire GitHub account

Closing Remarks

By no means is this a complete guide to Git. It is simply meant to get you started by showing you what a typical workflow might look like if you worked in industry.

Clone > Modify File > Add to Staging Area > Commit > Push to GitHub

There is much, much more you can do with Git. Refer to the included cheat sheet for more information. Thank you to those who attended our workshop and showed your support.

- IEEE Computer Society Ryerson Team