

Git In YU Computer Science

Avraham Leff

COM2545: 2017

Yeshiva University

avraham.leff@yu.edu

Perspective On This “Git” Lecture



- We'll discuss what you need to get by
- And a bit more to whet your appetite

Why Version Control?

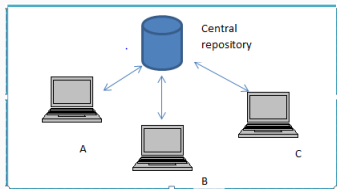
- A system that keeps track of your code changes
 - “code” == all of your project’s artifacts
- Who changed the code? When?
- Revert any changes, return to known state

Why Git?

- Git provides **distributed** version control
- Users keep entire code base (and *history*) on their own machines
- Users can work locally
 - Without network access
 - Blazingly fast
- Until you're ready to **push** or **pull**

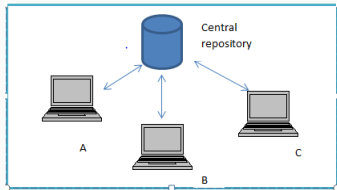
A Very Small Subset of Git

- You'll be using git to as a place to make your code visible to **me**
- And, optionally, for me to give **you** feedback



A Very Small Subset of Git

- You'll be using git to as a place to make your code visible to **me**
- And, optionally, for me to give **you** feedback
- You also benefit from having your code backed up to github



Git Features We're **Not** Using

- You will **not really** be using git's powerful *distributed version control* features

Git Features We're **Not** Using

- You will **not really** be using git's powerful *distributed version control* features
- Or, *peer-to-peer* development features

Workflow Sketch

1. You work on your local machine

Workflow Sketch

1. You work on your local machine
2. You **commit** your work **locally**

Workflow Sketch

1. You work on your local machine
2. You **commit** your work **locally**
3. You're ready to show me your work
 - Homework is due in five minutes 😊

Workflow Sketch

1. You work on your local machine
2. You **commit** your work **locally**
3. You're ready to show me your work
 - Homework is due in five minutes 😊
4. You **push** your work to the remote repository

Workflow Sketch

1. You work on your local machine
2. You **commit** your work **locally**
3. You're ready to show me your work
 - Homework is due in five minutes 😊
4. You **push** your work to the remote repository
5. Did I make changes to your code (or add a file etc)?

Workflow Sketch

1. You work on your local machine
2. You **commit** your work **locally**
3. You're ready to show me your work
 - Homework is due in five minutes 😊
4. You **push** your work to the remote repository
5. Did I make changes to your code (or add a file etc)?
6. You **pull** any changes from the remote repository to your local machine

Install Git On Your Laptop: Many Options

- Command-line
- Other command-line installs
- Altassian SourceTree, visual interface to git
- Other GUI clients
- Github Desktop
- Versions available for both Mac and Windows
 - Some are free, some are not.

Install Git On Your Laptop: Many Options

- Command-line
- Other command-line installs
- Altassian SourceTree, visual interface to git
- Other GUI clients
- Github Desktop
- Versions available for both Mac and Windows
 - Some are free, some are not.
- Or use your favorite search engine

Minimal Git Configuration

You'll only need to do this once!

- *git config --global user.name "John Doe"*
- *git config --global user.email johndoe@example.com*
- If you want to specify the editor to use when interacting with git

You **Already Have** a Git Repository!

```
avraham@leff-3:tmp$ pwd
/tmp
avraham@leff-3:tmp$ mkdir YU_Git
avraham@leff-3:tmp$ cd YU_Git/
avraham@leff-3:YU_Git$ git clone https://github.com/Yeshiva-University-CS/testforleff.git
Cloning into 'testforleff'...
remote: Counting objects: 121, done.
remote: Total 121 (delta 0), reused 0 (delta 0), pack-reused 121
Receiving objects: 100% (121/121), 35.91 KiB | 5.13 MiB/s, done.
Resolving deltas: 100% (7/7), done.
avraham@leff-3:YU_Git$ ls
testforleff
avraham@leff-3:YU_Git$ ls testforleff/
IntroToAlgorithms IntroToCompSci  README.md      test.md
avraham@leff-3:YU_Git$
```

- git clone **URL**
 - You received that **URL** in your Git *invitation*
- You'll **continue to push work** to that repo

Connect File System To Remote Git Repo

1. `git clone` your private YU repository
2. Create a `ROOT` directory for all your COM 2545 work
 - As a *subdirectory* of your cloned repo

Connect File System To Remote Git Repo

1. `git clone` your private YU repository
2. Create a **ROOT** directory for all your COM 2545 work
 - As a *subdirectory* of your cloned repo
 - e.g.,
`../YUGit/IntroToAlgorithms/homework`

Connect File System To Remote Git Repo

1. `git clone` your private YU repository
2. Create a **ROOT** directory for all your COM 2545 work
 - As a *subdirectory* of your cloned repo
 - e.g.,
`../YUGit/IntroToAlgorithms/homework`
 - **Your** choice on **YUGit**

Connect File System To Remote Git Repo

1. `git clone` your private YU repository
2. Create a **ROOT** directory for all your COM 2545 work
 - As a *subdirectory* of your cloned repo
 - e.g.,
`../YUGit/IntroToAlgorithms/homework`
 - **Your** choice on YUGit
 - **My choice** on
`IntroToAlgorithms/homework`

Connect File System To Remote Git Repo

1. `git clone` your private YU repository
2. Create a **ROOT** directory for all your COM 2545 work
 - As a *subdirectory* of your cloned repo
 - e.g.,
`../YUGit/IntroToAlgorithms/homework`
 - **Your** choice on YUGit
 - **My choice** on
`IntroToAlgorithms/homework`
 - “IntroToAlgorithms/homework” is your **ROOT** directory

Arrange File System To Taste

- You can clone the **URL** anywhere in your file system
 - Multiple times

Arrange File System To Taste

- You can clone the **URL** anywhere in your file system
 - Multiple times
 - Multiple locations
- Create sub-directories as needed

```
ROOT/RationalNumbers/src/main/java/edu/yc/oats/  
algs
```

Work Locally

```
leff@leff-3:~$ cd testforleff/IntroToCompSci/homework/
[avraham@leff-3:YU_Git$ cd testforleff/IntroToCompSci/homework/
[avraham@leff-3:homework$ ls
arrays          conditional_logic exceptions      objects
[avraham@leff-3:homework$ mkdir showme
[avraham@leff-3:homework$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
[avraham@leff-3:homework$ cd showme/
[avraham@leff-3:showme$ touch App.java
[avraham@leff-3:showme$ more App.java
[avraham@leff-3:showme$ git add App.java
[avraham@leff-3:showme$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

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

    new file:   App.java
```

Create, edit, remove files in blissful ignorance
of git

You Made Changes To Your Code

- Created, edited, removed files

You Made Changes To Your Code

- Created, edited, removed files
- You're ready to *bundle these changes together* in a **snapshot**
 - Can be a *subset* of these changes: **you decide**

You Made Changes To Your Code

- Created, edited, removed files
- You're ready to *bundle these changes together* in a **snapshot**
 - Can be a *subset* of these changes: **you decide**
- Invoke *git add file*
 - This adds **file** to the **snapshot** you're preparing

Commit **When Appropriate** Locally

```
[avraham@leff-3:showme$ git commit -m 'Started empty homework assignment.' App.java  
[master 99279fe] Started empty homework assignment.  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 IntroToCompSci/homework/showme/App.java
```

- *Ready?*

Commit **When Appropriate** Locally

```
avraham@leff-3:showme$ git commit -m 'Started empty homework assignment.' App.java
[master 99279fe] Started empty homework assignment.
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 IntroToCompSci/homework/showme/App.java
```

- *Ready?*
- Invoke *git commit*
 - You've just created a snapshot!
- Use *-m* option to specify a message that will be associated with this commit
 - Or git will demand one from you 😊

Some Useful Commands

- *git status*: What is git's view of your repository's status?
- *git diff*: What changes have been made since your last commit?
- *git log*: What's the **history** of commit messages
- You can supply a specific file name to these commands
 - To see the status of, or changes to, a specific file

Push When Ready

```
avraham@leff-3:showme$ git push
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 430 bytes | 430.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Yeshiva-University-CS/testforleff.git
   0ae10ff..99279fe master -> master
avraham@leff-3:showme$
```

- **git push** takes your *local* commit(s)
 - And pushes those changes to your remote repository
- Now other people (with access to your repository) can see your commit(s)

Pull When Available

```
0dc10111392731c master -> master  
[avraham@leff-3:showme$ git pull  
Already up-to-date.  
avraham@leff-3:showme$ ]
```

- **git pull** takes the changes made to your *remote* repository
 - Any *commits* made by other people
 - Pulls them to your local computer
 - Updates your file system with these changes

(Re)Moving Files

- If you've made git aware of a file

(Re)Moving Files

- If you've made git aware of a file
- What do you think will happen if you yank it away?
 - *rm file* or *mv file*

(Re)Moving Files

- If you've made git aware of a file
- What do you think will happen if you yank it away?
 - *rm file* or *mv file*
- Trust me: you don't want to confuse git

(Re)Moving Files

- If you've made git aware of a file
- What do you think will happen if you yank it away?
 - *rm file* or *mv file*
- Trust me: you don't want to confuse git
- Instead
 - *git rm file* removes file from working tree
 - *git mv file* move or rename a file or a directory
- You *commit* those changes like any other change set

A Graphical View: SourceTree

The screenshot displays the SourceTree application interface for a repository named "testforleff (Git)". The top toolbar includes buttons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. The left sidebar shows the workspace structure with sections for File status (14), History, Search, BRANCHES (master), TAGS, REMOTES (origin), STASHES, SUBMODULES, and SUBTREES.

The main area is divided into three panes:

- Left Pane (Graph):** Shows a commit history graph with a vertical line of blue dots representing commits. The current commit is highlighted.
- Middle Pane (Description):** Displays the commit history table with columns for Description, Commit, Author, and Date. The table shows several commits, including "Started empty homework assignment.", "My code", "Files from my implementation of homework", "Copied from my code.", "Copied from my directory.", "Re-rooted 'conditional_logic' directory off 'homework'.", "Experimenting with ability to pull code & compile created sample test file", and "first commit".
- Right Pane (Staged files):** Shows the "Staged files" section with a list of files. The selected file is "IntroToAlgorithms/homework/BinarySearchST/BinarySearchST.md". The file content is displayed in the bottom right pane, showing a class definition for "BinarySearchST" and a "Grading" section.

Description	Commit	Author	Date
Uncommitted changes	-	-	Today, 7:46 PM
Started empty homework assignment.	99279fe	avraham <avraha...	Today, 7:41 PM
My code	0ae10ff	avraham <avraha...	Aug 8, 2017, 2:20 PM
Files from my implementation of homework	148b160	avraham <avraha...	Aug 7, 2017, 10:56 AM
Copied from my code.	743c3d	avraham <avraha...	Aug 3, 2017, 1:36 PM
Copied from my directory.	68bb6f9	avraham <avraha...	Aug 2, 2017, 6:33 PM
Re-rooted "conditional_logic" directory off "homework".	026db87	avraham <avraha...	Jul 25, 2017, 5:56 PM
Experimenting with ability to pull code & compile created sample test file	feb91c2	avraham <avraha...	Jul 25, 2017, 5:47 PM
first commit	b57313a	avraham <avraha...	Jul 25, 2017, 1:46 PM
	8ab45ed	avraham <avraha...	May 15, 2017, 8:54 AM

The "Staged files" section shows the file "IntroToAlgorithms/homework/BinarySearchST/BinarySearchST.md" with a diff view. The diff shows changes to the file, including a class definition and a "Grading" section.

```
41 41  // Your class must be named __BinarySearchST.java__, and it
42 42  must be reside 'DIR/src/main/java/edu/yc/oats/oLgs'.
43 43
44 44  // Your assignment may not use any external libraries: just the SDK and your code in this package.
45 45  // Grading
46 46  // If your code cannot be compiled & run (either because it doesn't follow the packaging conventions
```

Gazillion Resources

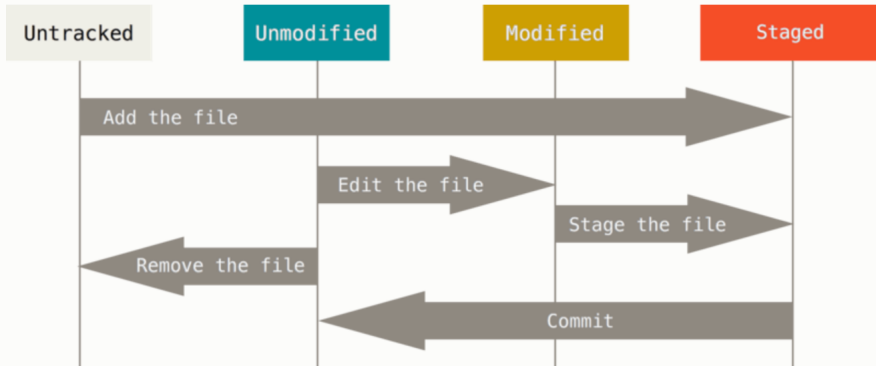
- I strongly urge you to keep it simple!
 - Not the focus of this course
 - And you don't need more to get your work done
- git - the simple guide
- Using Git (Sections A through E)
- OK: you asked for it ...

The Three Git States

Your files can be in one of three states:

- **Committed**: data are safely stored in your local git database
- **Modified**: file has been changed, but not yet **committed**
- **Staged**: you asked that current version of **modified** file go into your next commit snapshot

Git File Lifecycle



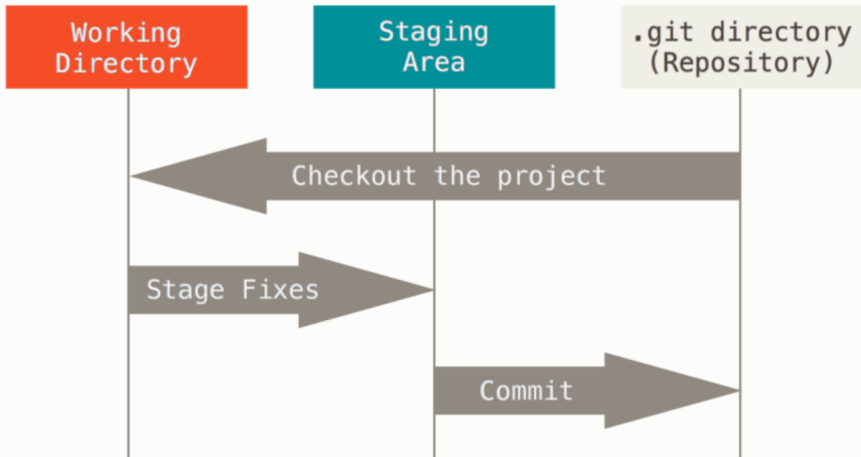
Git Directories: .git & Working Tree

- **.git** directory: where Git stores the metadata and object database for your project
 - Most important part of Git: what's copied when you clone a repository from another computer
- **Working tree**: a single checkout of one version of the project
 - Files are pulled out of the compressed **.git database**

Git Directories: Staging Area

- **Staging area**: a file in **.git directory** that stores information about what will go into your next commit
 - You modify that file, e.g., with **git add** and **git commit**

Interaction Between Git Directories



Tracked versus Untracked

- Each file in your working directory can be in one of two states
 - *Tracked*: files in last snapshot
 - *Untracked*: everything else – any files in your working directory that were
 - not in your last snapshot
 - not in your staging area
- Tracked files can be unmodified, modified, or staged
- As you edit files, Git sees them as modified
 - You changed them since last commit

Commits Create a Snapshot

- Records entire state of your file system at that point
- Efficient
 - Delta relative to previous commit
 - Reference to parent (previous) commit
 - SHA-1 hash (40-character string), calculated based on the contents of a file and directory structures

Git Is Additive

- Interactions (nearly all) with Git only add data to git database
 - Data are not erased
- Once you commit

Git Is Additive

- Interactions (nearly all) with Git only add data to git database
 - Data are not erased
- Once you commit
 - Especially if you **push**
- Hard to lose data
- This make it easy to undo things
 - Which we won't be discussing

What Else **Aren't** We Discussing?

- Concept of **branch**
- Concept of **merge**
- Concept of **rebase**

What Else **Aren't** We Discussing?

- Concept of **branch**
- Concept of **merge**
- Concept of **rebase**
- These give you tremendous freedom to **experiment** and thus be more *agile*