CMSC388T: Introduction to Git, Github and Project Management Tools

Winter 2021

Dr. Anwar Mamat

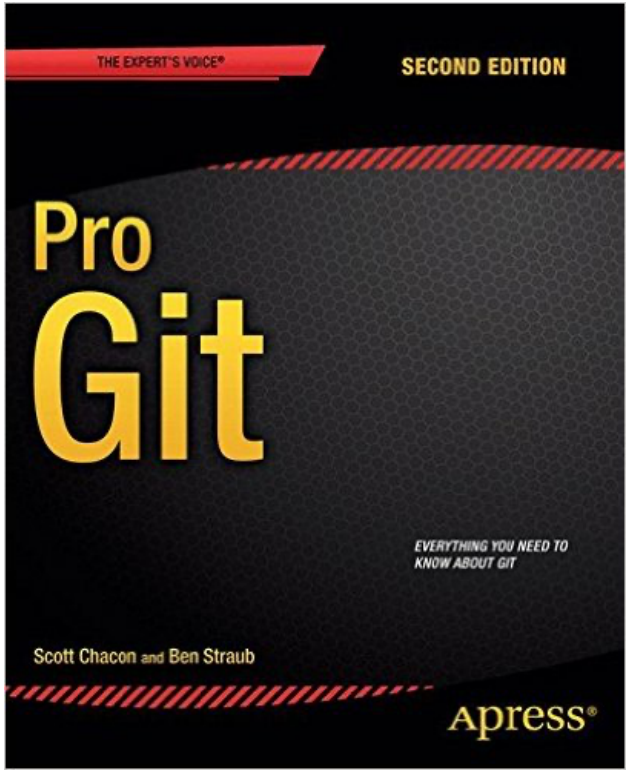

2020
a year to Forget!
TOILET PAPER SHORTAGE
HAND SANITIZER
CURBSIDE PICK UP
DRIVE BY PARTIES
TRAVEL BAN
WORK FROM HOME
WORLD WIDE PANDEMIC
QUARANTINE
SOCIAL DISTANCING
ONLINE SCHOOL
WEARING MASKS
SPORTS CANCELLED

Course Home Page

http://www.cs.umd.edu/class/winter2021/cmsc388T/

- Syllabus
- Course schedule
- Contact info
- TAs: Nandhini, Sagar, Sanjay
- Office hours schedule
- Lecture slides
- Projects
- Lecture videos will be posted on ELMS.

Recommended Textbook



Download the PDF and videos
https://git-scm.com/book/en/v2

Schedule

- Lectures:
  - M/W/F 9:30am-10:50am
  - Synchronous, online
  - Zoom link is posted on ELMS.

- Group Projects:
  - Tu/Th 9:30am-10:50am

Grading

- 3 Group Projects
  - 10 groups, 4 students in each group
  - Git/Github, and other topics
- No Exams ;)

Getting Started With Git

What is Git?

- A version-control system for tracking changes in your code
- Used for coordinating work on files among multiple people.
  - Who wrote this module?
  - When was this function edited? By whom? Why was it edited?
  - Over the last 1000 revisions, when/why did a particular unit test stop working?

- Git is a ‘Distributed Version Control System'.
  - Git does not rely on a central storage unit for a user’s code history
  - Users clone (or download) repositories and maintain a history of changes to your version of a project

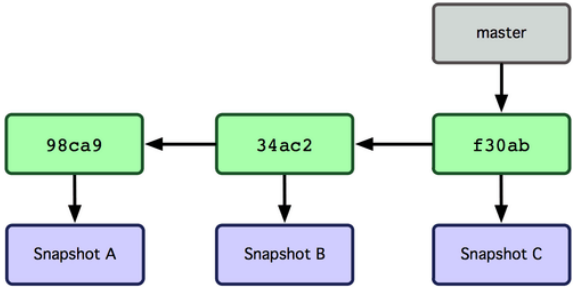
Why use Git?

- Great for coordinating changes on a project among multiple contributors
- Great for debugging purposes
- Extremely fast version control
- Cloud storage of your code.
- Show off your code!

How Does Git Work?

Before we learn Git command line interface, let us understand the underlying design ideas of Git.

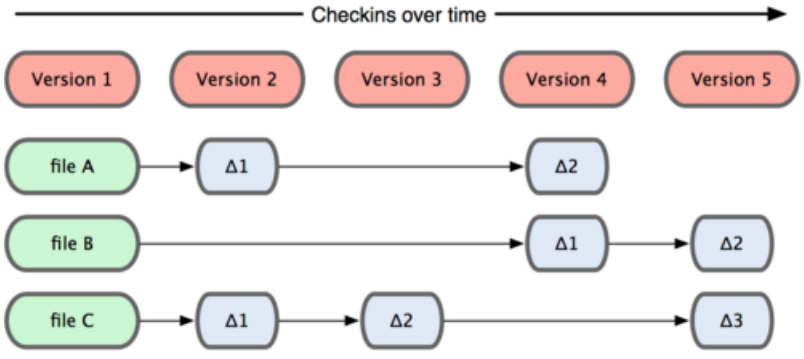
Snapshots



- Git thinks of its data more like a set of snapshots of a miniature filesystem.
- Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
- To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a **stream of snapshots**.

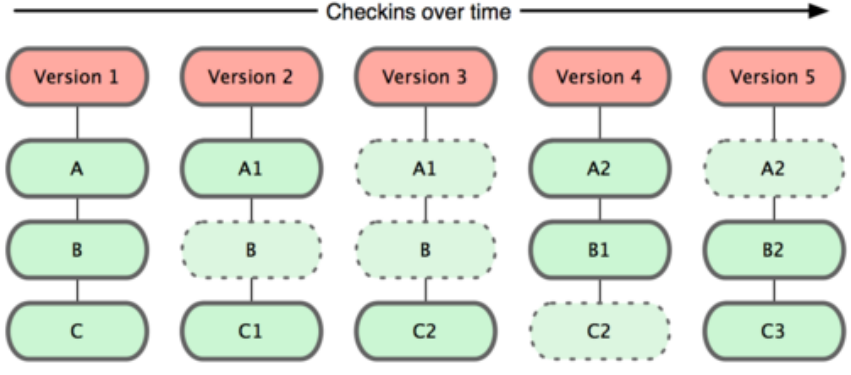
Snapshot

Delta Storage:
CVS, Subversio, or other version control systems



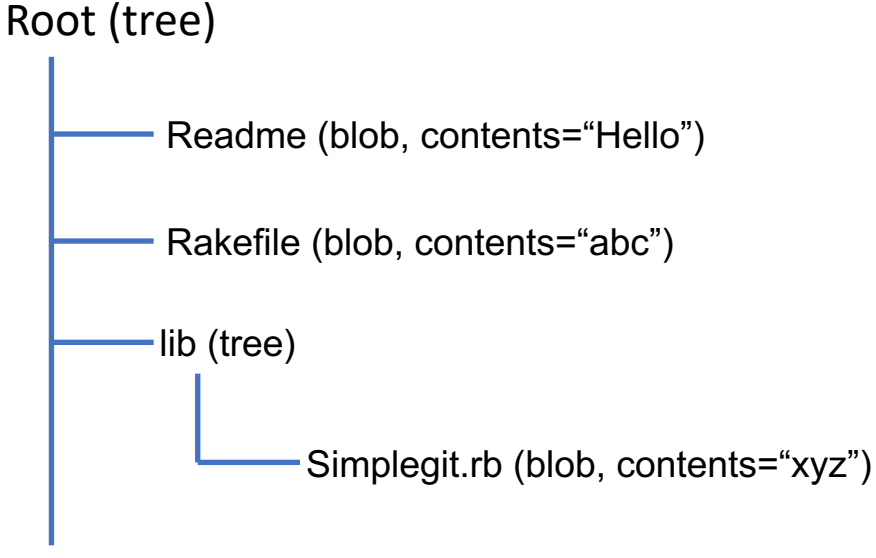
Snapshot Storage:

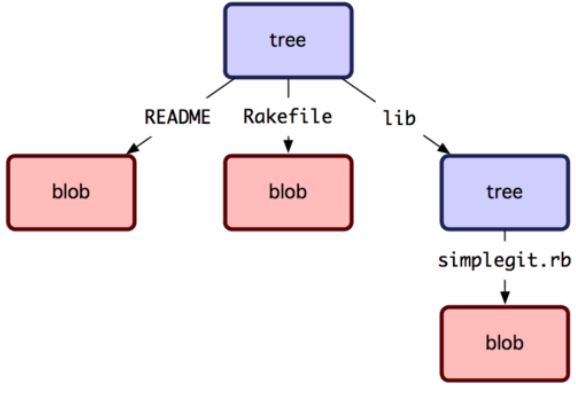
Git



Snapshot

In Git terminology, a file is called a “blob”, and it’s just a bunch of bytes. A directory is called a “tree”, and it maps names to blobs or trees (so directories can contain other directories). A snapshot is the top-level tree that is being tracked.

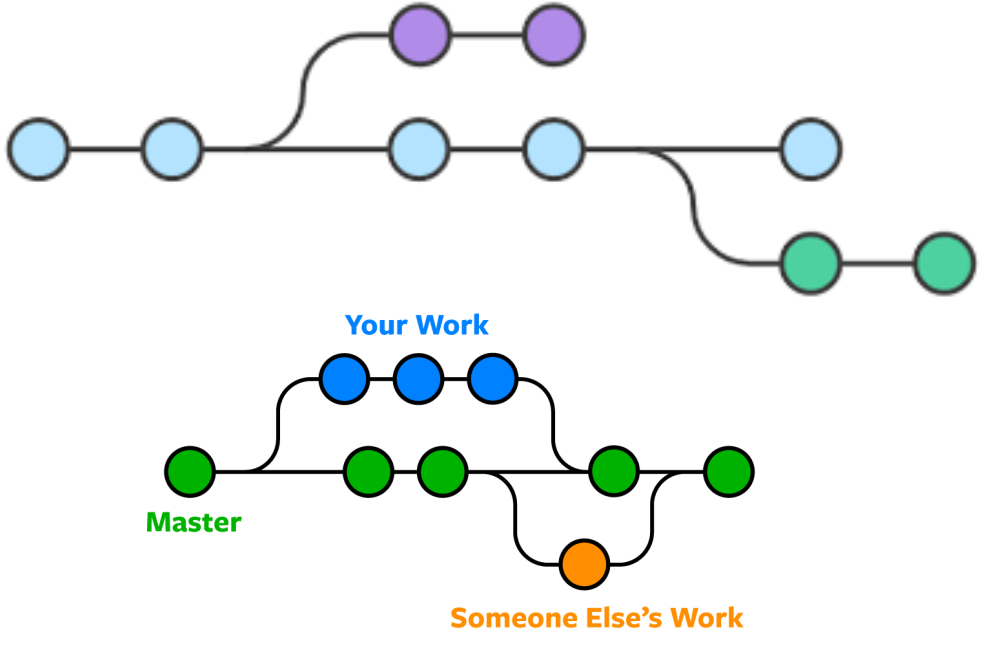




Modeling History

Git models the history as a directed acyclic graph (DAG) of snapshots.

- Each snapshot in Git refers to a set of “parents”, the snapshots that preceded it.
- Git calls these snapshots “commit”s



Objects and Hash

Objects : Blobs, trees, and commits are called objects.

Hash: In Git data store, all objects are content-addressed by their SHA-1 hash.

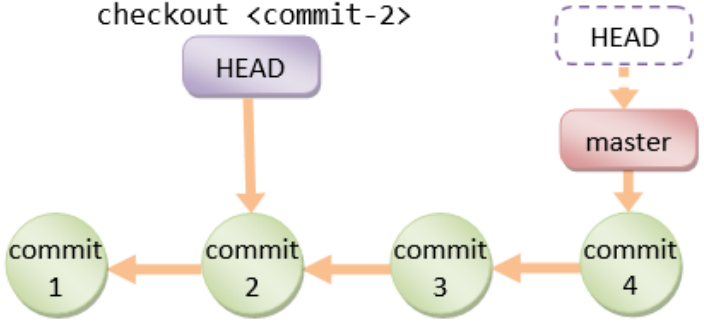
Objects and Hashes are immutable.

```
commit 61f57e85d04682467cde2436247ee80b6efdf1ed
Author: Anwar Mamat <anwarmamat@gmail.com>
Date:   Mon Dec 7 21:36:53 2020 -0500

    adds garbage collection slides
```

References

- All snapshots can be identified by their SHA-1 hash. It is hard to remember 4- hexadecimal characters.
- Git assigns human readable names to for the hashes, called “references”.
- References are pointers to the commits.
- For example,
  - “master” (or main) reference usually points to the latest commit in the main branch of development.
  - “HEAD” is reference to “where we currently are”



Repositories

- Git repository: the data objects and references.
- All git commands map to some manipulation of the commit DAG by adding objects and adding/updating references.
- Whenever you're typing in any command, think about what manipulation the command is making to the underlying graph data structure.

Example:

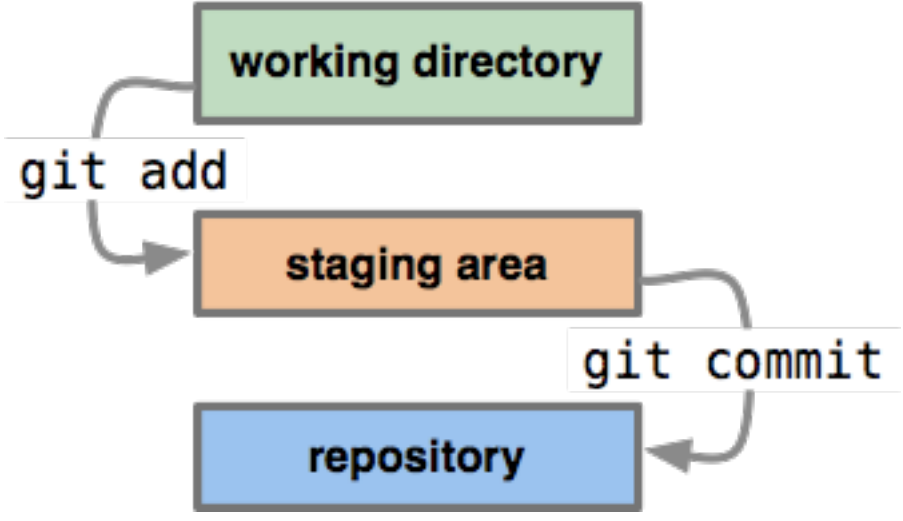
Discard uncommitted changes and make the 'master' ref point to commit 5d83f9e

```
git checkout master
git reset --hard 5d83f9e
```

Staging Area

We want clean snapshots, and it might not always be ideal to make a snapshot from the current state.

The **staging area** is a place to record things before committing.



Git command-line interface

- `git init`: creates a new git repo, with data stored in the .git directory
- `git status`: Gives you a current overview of your repository. It telling you which files have or haven't been saved and what changes are in staging.
- `git add`: Adds any changes made to your project to the staging area. This does NOT affect your repository until your change is committed.
- `git commit`: Essentially saves any added changes to your local repository

Git command-line interface

- `git help <command>`: get help for a git command
- `git log`: shows a flattened log of history
- `git log --all --graph --decorate`: visualizes history as a DAG
- `git diff <filename>`: show changes you made relative to the staging area
- `git diff <revision> <filename>`: shows differences in a file between snapshots
- `git checkout <revision>`: updates HEAD and current branch

Putting it All Together