CS3300 Introduction to Software Engineering

Lecture 04: Tools of the Trade #2

Version Control Systems, GIT

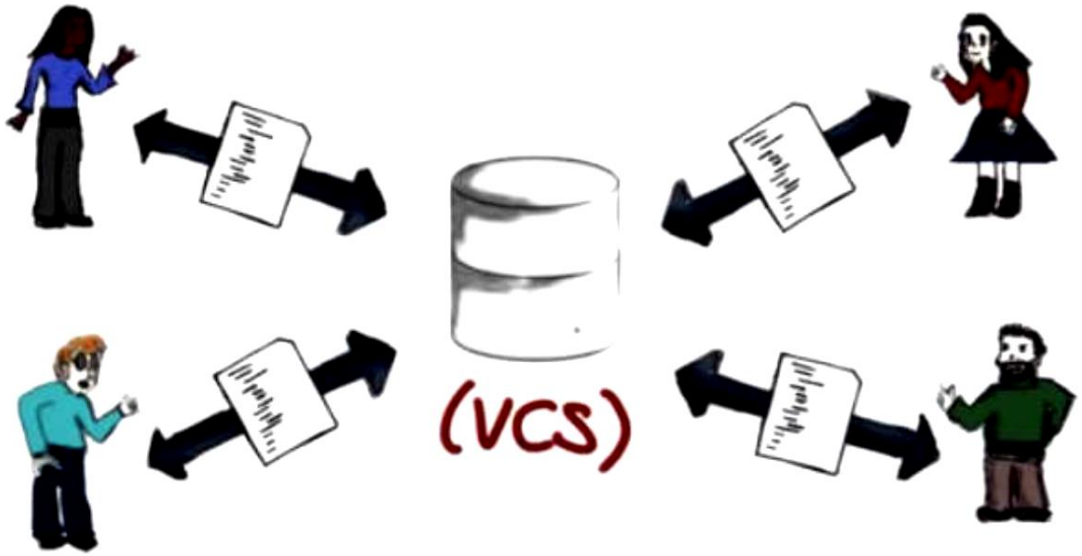
Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

Contents

- What are Version Control Systems?
- Importance/Benefits
- Essential Actions
- Example Workflow
- Don’ts in VCS
- Types of VCS
- GIT – Workflow/ Demo
- EGit
- GitHub

What are Version Control Systems?



- A tool that software developers use for keeping track of revisions of their project
  - snapshots of your project over time.
  - Documents, source files etc.
- Most obvious benefits:
  - Option to go back and revisit
  - Collaborate with multiple people

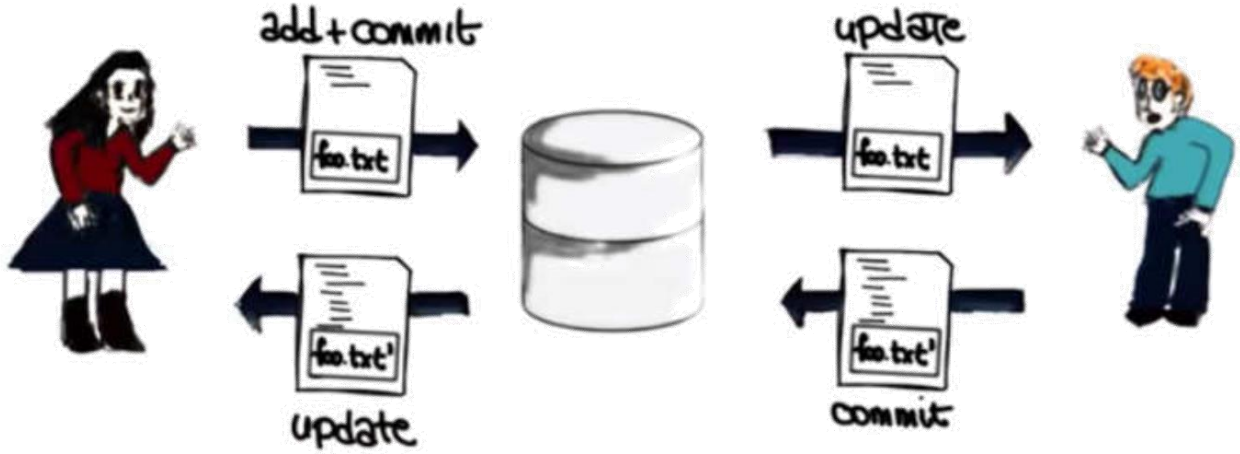
Importance

- **Enforce Discipline**: Manages process by which control of items passes from one person to another
- **Archive versions**: store subsequent versions of source-controlled items
- **Maintain Historical Information**: Author of a specific version; date and time of a specific version; etc. Retrieve and compare.
- **Enables Collaboration**: share data, files and documents
- **Recover from accidental edits/revisions**
- **Conserve Disk Space**: centralizing the management of the version.
  - Instead of having many copies spread around, one or a few central points where these copies are stored
  - efficient algorithms to store changes, so keep many versions without taking up too much space.

Essential Actions

- **Add**:
  - Add file to repository
  - Accessible to other people who have access to repository
- **Commit**: make local changes to file already present in the repository, commit changes to the central repository so that others can see the difference
- **Update**: Someone else can modify files in the repository, get changes made to the files in your local repository

Example Workflow



- Sarah creates a file called foo.txt and puts some information. She wants to **add** the file to the repository and to **commit** it so that her changes and the file get to the central repository.
- When she adds and commit, foo.txt will become available and will be accessible to the other users, in this case, Peter.
- Peter runs an **update** command to copy foo.txt onto his local workspace. He now has access to the file.
- Peter now wants to add something to the file, he edits foo.txt in his local repo and **commits** the edited version to the central repository.
- The result will be the same as when Sarah committed her file—the updated file will be sent to the repository and the repository will store that information and make it available for other users.
- Now if Sarah performs an **update**, she will get the new version of foo.txt with the additional information that was added by Peter.

Don’ts in VCS

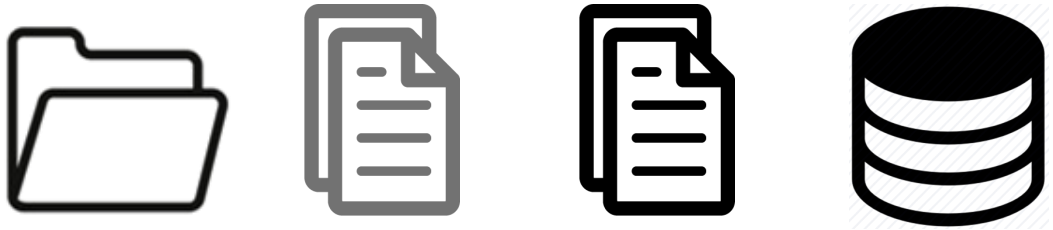
- Adding Derived Files
  - E.g., executable file derived from compiling a set of source codes
  - No reason to add it
- Adding bulky binary files
  - Try to keep them local
- Creating a local copy of files/tree of files
  - Don’t do this!!
  - Useless, risky, confusing
  - Trust the version control system

Types of VCS

- Centralized VCS
  - Single centralized repository on which you are committing files
- Decentralized/ Distributive VCS
  - Sort of local repository on which you can commit changes
  - Commit without other users of VCS being able to see the changes
  - When you are happy with the version, push to central repository. Now available to other users of VCS
  - Advantages:
    - Having a local version
    - Can use multiple remote repositories

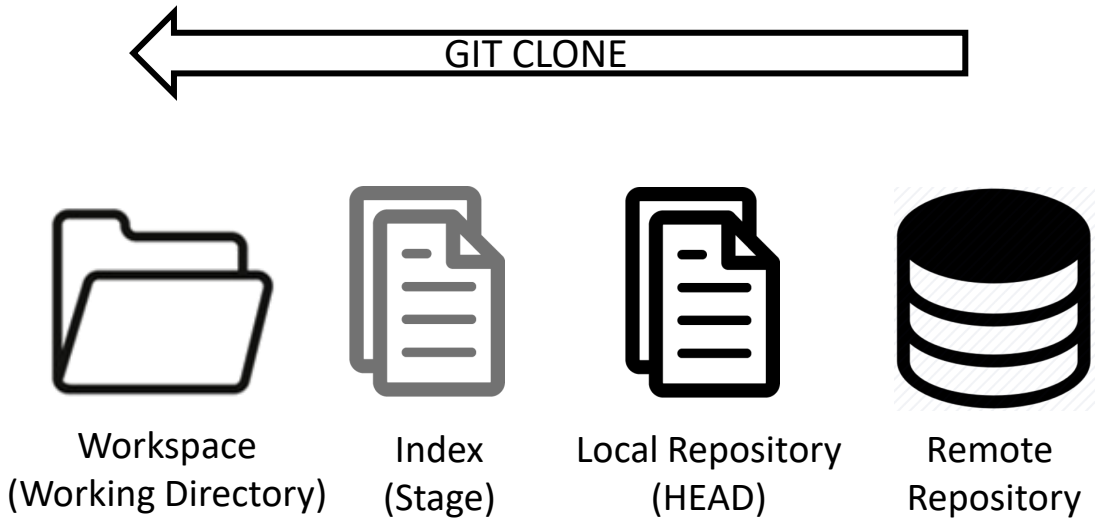
GIT

- One good representative of distributed version control systems, is GIT.
- Designed and developed by Linus Torvalds, creator of the Linux operating system.
- Linus was unhappy with the existing version control systems and wanted a different one for maintaining the Linux kernel, with some key characteristics.
  - distributed.
  - fast.
  - simple design
  - strong support for parallel branches
  - able to handle large projects.
- GIT workflow



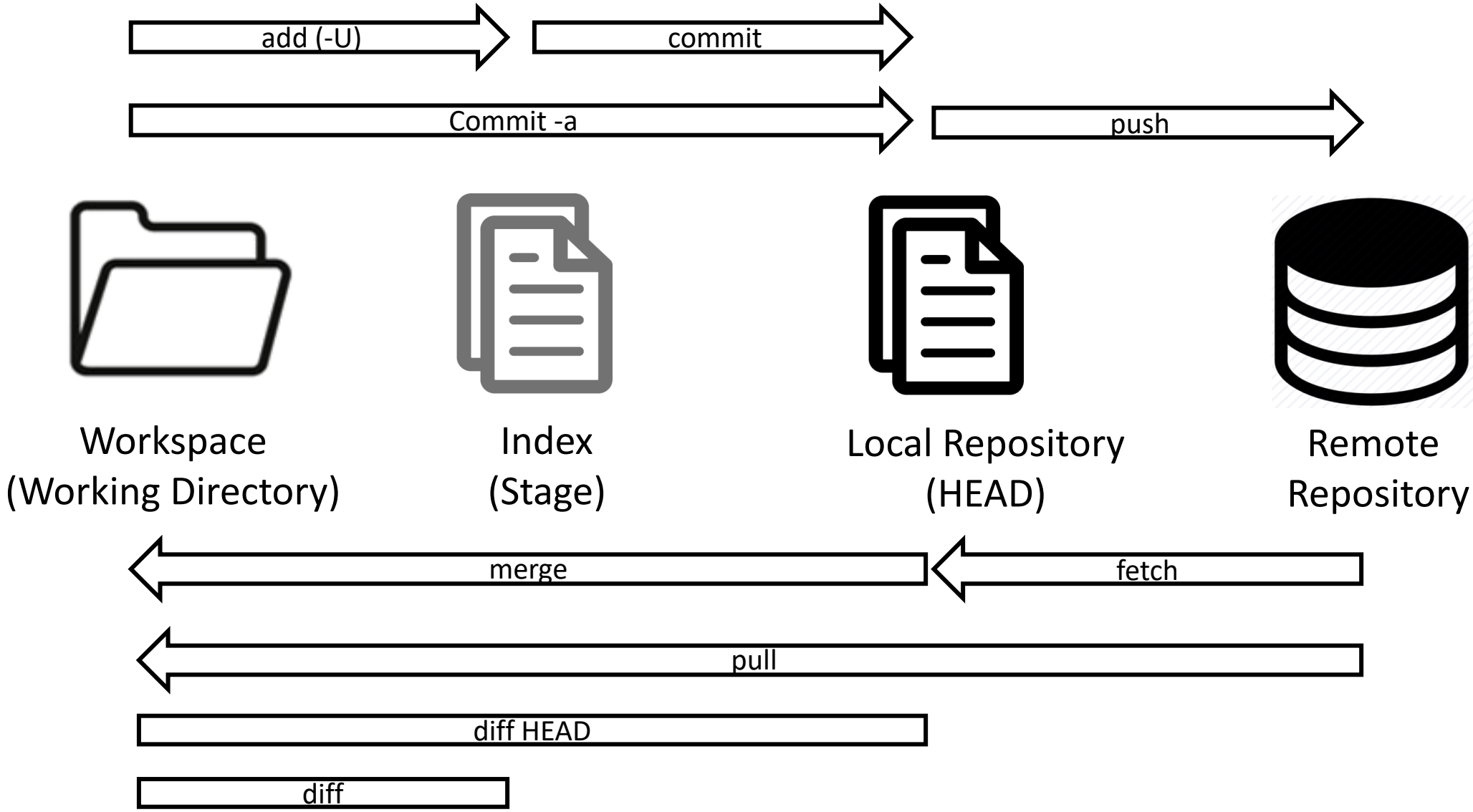
GIT Workflow

At any point, your files can be **modified** (locally) , **staged** or **committed.**



The git clone, followed by the url for that repository, will create a local copy of the repository in your workspace. You don't have to do this step if you're creating the repository yourself.

GIT Workflow



GIT Demo

Install GIT: Follow instructions on https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

EGit

- GIT Plugin available for Eclipse
- The plugin can be downloaded at www.eclipse.github.com and can be installed in Eclipse

GIT Recap

LOCAL REPOSITORIES

CREATE

mkdir ProjectName
cd ProjectName
git init

MODIFY

git add foo.txt
git commit –m “message”

INSPECT

git log
git status
git diff
git diff HEAD
git show

REMOTE REPOSITORIES

COPY REPOSITORY

git clone <repo url>

- Creates a completely local copy of repository
- links it to remote repository (origin)

RECEIVE CHANGES

git pull

SEND CHANGES

git push

GitHub

- GIT hosting website. Get an account and create your remote repositories
- GitHub repository for your projects
- Provides easy-to-use FREE desktop clients for Mac and Windows (https://desktop.github.com)

- **GitHub Pages**:
  - One click to enable for your GitHub repo.
  - Hosted directly from your GitHub repository.
  - Just edit, push, and your changes are live.
  - This course's website is a GitHub Page.

- ALWAYS SET YOUR GITHUB REPOSITORY TO BE PRIVATE, UNLESS YOU ARE ABSOLUTELY SURE YOU WANT IT PUBLIC !!!

Next Class

- TTOT#3
- HTML, CSS, JS
- Download and Install Visual Studio Code: https://code.visualstudio.com/
- Bring your Laptops !!