Tutorial 1: Cluster Access and CLI Basics

Informatik elective: GPU Computing

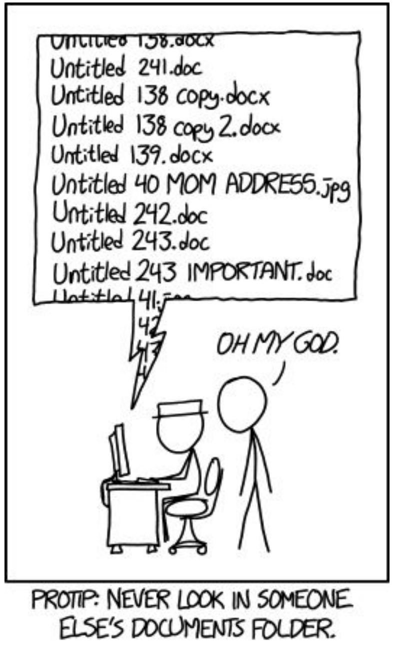
Pratik Nayak

In this session

- Version control: principles, git, using github.
- Command line basics in Linux.
- Compiling and running a basic C/C++ program.

Version control

- How do you record changes to a file/project over time ?
    - Copy and create new files for each version.



[xkcd: #1459]

Version control

- How do you record changes to a file/project over time ?
    - Copy and create new files for each version.
    - Use version control

Why Version control ?

	Advantages
Shareability	Easily share state with others
Trackability	Track changes across time
Collaborate	Easily collaborate with people
Workflow	Greatly simplifies workflows

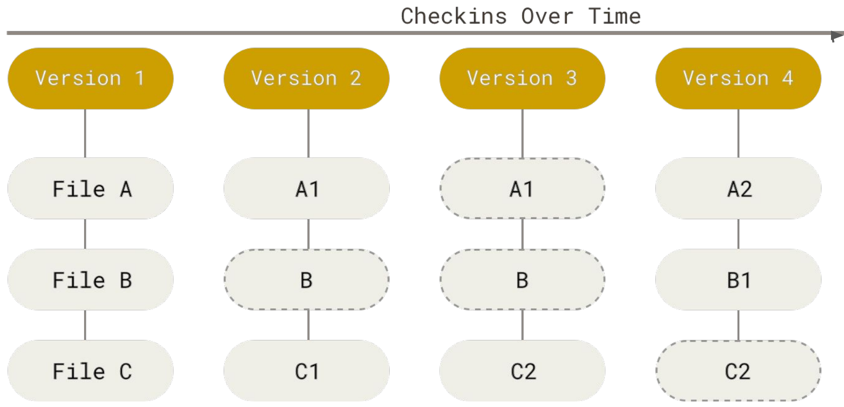
Git: A distributed version control system

```
GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

 - random three-letter combination that is pronounceable, and not
   actually used by any common UNIX command.  The fact that it is a
   mispronunciation of "get" may or may not be relevant.
 - stupid. contemptible and despicable. simple. Take your pick from the
   dictionary of slang.
 - "global information tracker": you're in a good mood, and it actually
   works for you. Angels sing, and a light suddenly fills the room.
 - "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager.  It
doesn't do a whole lot, but what it _does_ do is track directory
contents efficiently.
```

[Linus Torvalds on git, 2005]

Git: A distributed version control system

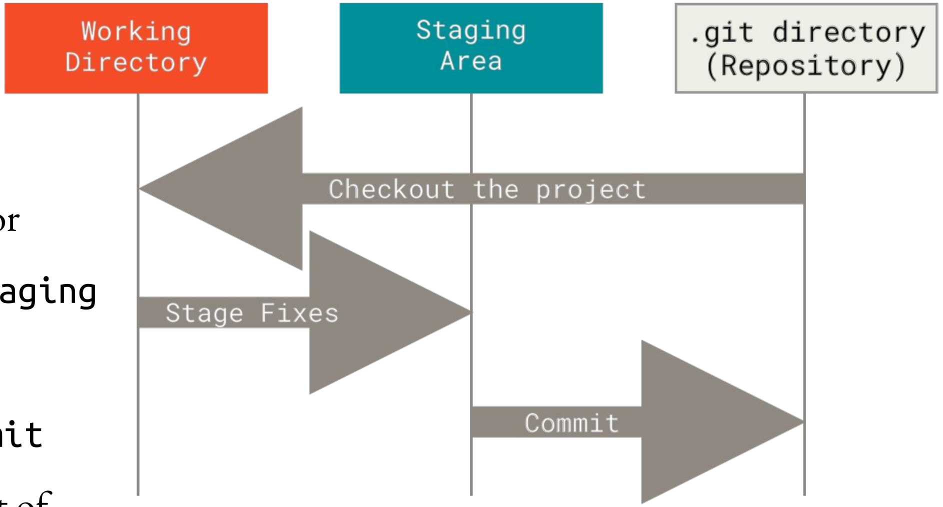
- A stream of snapshots of the entire repository.
  - Store only references to unchanged files.
- Almost every operation is local.
  - Don’t need to fetch data from remote server.
- It only adds data.
  - Almost everything is undoable.



[git-scm.com/book]

Git: How does it work ?

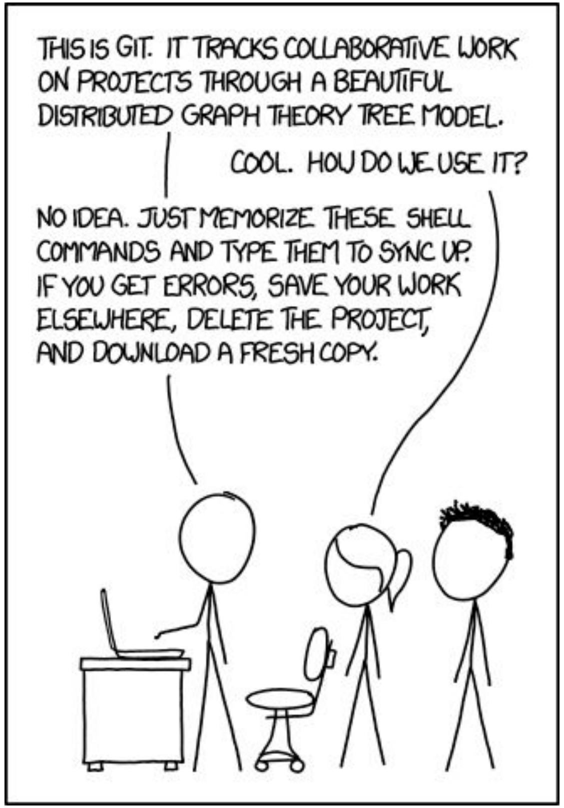
- 3 main states: `modified`, `staged` and `committed`
- The workflow therefore, consists for 3 main steps:
  - Changed files are in the `modified` state
  - You decide which files you want to `commit`, or store a snapshot of, and move them to the `staging` area
  - Once added to the staging area, you can `commit` them, which "permanently" stores a snapshot of the repository at that point in time.



[git-scm.com/book]

[HANDS-ON] Git: Demo

- Initialise a repository with git init
- Create a file, add some text
- Move file to staging area
- Commit the changes
- Look at the repository history
- Clone some public repository and look at its history.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

[xkcd: #1597]

`git` good practices

- Commit files frequently, and in a modular fashion.
- If there is a remote, keep it updated.
- Commit messages are very helpful, be descriptive, you'll thank yourself in the future.
- Optimize/personalize your configuration with `git config`
- `git bisect` helps with debugging

COMMENT
DATE
CREATED MAIN LOOP & TIMING CONTROL
14 HOURS AGO
ENABLED CONFIG FILE PARSING
9 HOURS AGO
MISC BUGFIXES
5 HOURS AGO
CODE ADDITIONS/EDITS
4 HOURS AGO
MORE CODE
4 HOURS AGO
HERE HAVE CODE
4 HOURS AGO
AAAAAAAA
3 HOURS AGO
ADKFJSLKDFJSDKLFJ
3 HOURS AGO
MY HANDS ARE TYPING WORDS
2 HOURS AGO
HAAAAAAAANDS
2 HOURS AGO
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

[xkcd: #1296]

Github, Gitlab?

- How do we collaborate with others ?
  - Host the repository on the cloud, and all collaborators synchronize through it.
- They also provide additional functionalities:
  - Pull/Merge requests: Collaborative editing
  - Continuous Integration: Testing, benchmarking etc

[HANDS-ON] Github: Demo

- Create a github account
- Push your repository to github.
- Look at the commit history.
- Compare history of some public project.

[HANDS-ON] Command line basics

- Look up documentation:

```
$ man command
```

- Which directory/path are you currently in ?
  - Print working directory (pwd)

```
$ pwd
```

- List current files

```
$ ls
```

- Change directories (cd)

```
$ cd new_directory_path
```

- Copy files

```
$ cp current_file_name new_file_name
```

- Move/rename file

```
$ mv current_file_name new_file_name
```

Platform-independent builds

- Building from source can be frustrating and annoying
- Use platform-independent build tools like CMake
- Provide necessary configuration, dependencies

  without worrying about platform-specifics.
- Widely used for C/C++ projects.

[HANDS-ON] Build a demo C++ project and push it to github

- Create a file test.cpp
- Add some test function.
- Create a CMakeLists.txt file, and build it.
- Check if it runs as expected.
- Commit the changes and push them to github