Git Version Control

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra
mbouguerra@langara.ca

Langara College

May, 2022

Lesson's Outline

Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will learn:

- ❒ What is a version control (VCS) system?
- ❒ How to use Git VSC system?
  - ➡ to track files of a local project?
  - ➡ to collaborate on a remote project?

Version Control Systems (VCS)s

VCS

- Version Control System (VCS) keeps tracks of changes made to
  - a set of files over time
- Version Control System (VCS) records the state of
  - a file as a new version after any changes

Version Control Systems (VCS)s

VCS

- Three types of Version Control Systems (VCSs)
  - Local VSCs
  - Centralized VSCs
  - Distributed VSCs

Version Control Systems (VCS)s
- Local Version Control Systems (VCS)s
- Centralized Version Control Systems (VCS)s
- Distributed Version Control Systems (VCS)s

Version Control Systems (VCS)s

Local VCS

Local VCS manages multiple versions of files in a local computer. The Revision Control System (RCS) is the most commonly used local VCS.

Centralized VCS

Centralized VCS keeps of the versions of the files in a single server, and the clients has to check out files from the server. All the changes has to be made on the server. Examples of Centralized VSC are Subversion and Perforce.

Distributed VCS

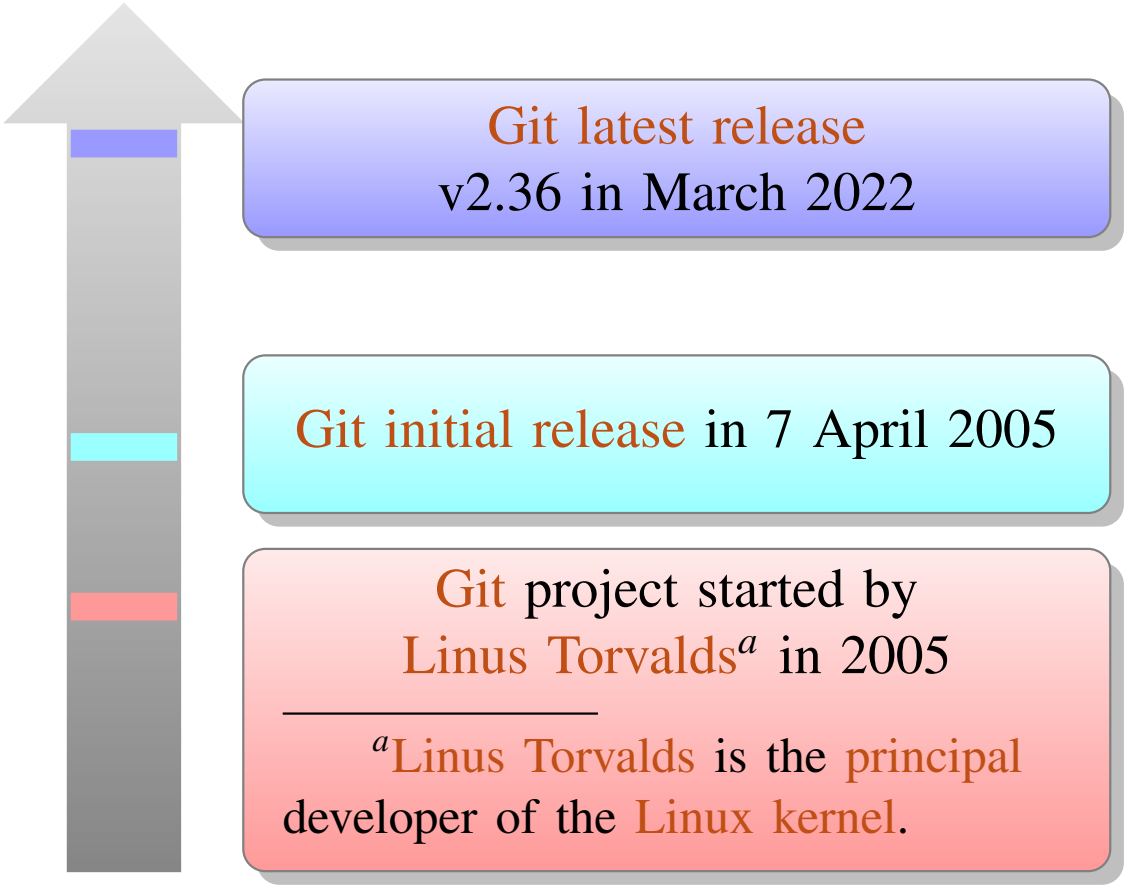
Distributed VCS allows clients to have copies of the versions of the files and be able to update the versions on the server. The most commonly used distributed VSC is Git.

Version Control Systems (VCS)s

VCS

- Git is a distributed VCS for tracking changes to a set of files
  - accessed & edited by multiple users
- Git
  - records who made what changes and when
  - allows to revert back to any version at any time
  - coordinates working on the same files by multiple users

Cryptanalytic Attacks



Version Control Systems (VCS)s

GitHub

- ❒ GitHub is a cloud platform for hosting
  - ➡ software development and version control using
    - ✓ Git distributed VCS

Git Installation

- Go to Git homepage https://git-scm.com/
- Download the latest Git version for your operation systems (OS)
- To check your installation, open your terminal[a]

```
git version
# or
git --version
```

[a] A CMD or PowerShell in Windows.

Git Basic Commands

Git Commands

- Git cammand has the format

```
git <subcommand> --<option>
```

  - ➡ option is an optional argument for the
    - ✓ the git subcommand

- To display the most commonly used Git commands

```
git help
```

- To display all git commands

```
git help --all
# OR
git help -a
```

Using git help

- To get help about a Git subcommand

```
git help <subcommand>
```

- To get help about the version git subcommand

```
git help version
```

Class Activity

- Using git help subcommand, display the documentation of the following git subcommands

  - init
  - add
  - commit
  - status
  - log
  - clone
  - pull
  - push

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Git Setup Commands

git config

- git config subcommand allows to configure

```
git config --<option>
```

  - Most commonly used options are

```
git config --global
git config --local
git config --list
```

    - ✓ The global option saves the settings to a global configuration file
    - ✓ The local option saves the settings to a local configuration file
    - ✓ The list option displays the git configuration file content

Using git config

- To display the current settings of your git configuration file

```
git config --list
```

- To add/update your name and email

```
git config --global user.name 'Mourad Bouguerra'
git config --global user.email 'mbouguerra@langara.ca'
```

- To check the updates to your git configuration file

```
git config --list
```

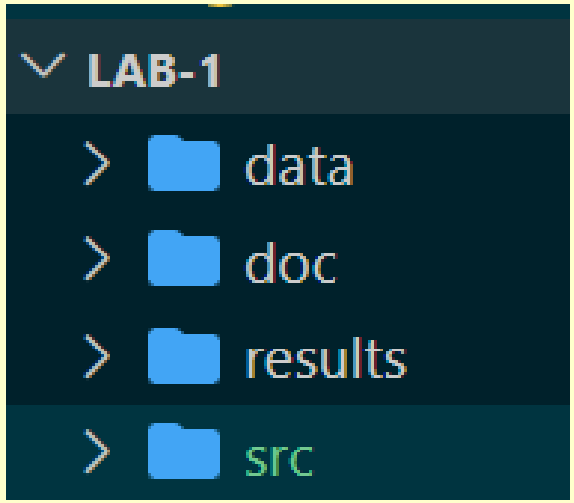
Git Applications

Git Application

- Two Git applications
  - Tracking a local data analytic project
  - Collaboring on a remote data analytic project

Tacking a Local Project

- Setup the following folder structure for data analytic project



Tacking a Local Project

- To start tracking your data analytic project
  - Open your terminal
  - Navigate to Lab-1 directory
  - Using ls to list the content of Lab-1 directory

```
ls -la
```

  - Use the git init command

```
git init
```

  - Using ls to list the content of Lab-1 directory

```
ls -la
```

Class Activity

- After running the git init command, list the newly created files and folders

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Tacking a Local Project

- After running git init
  - a new .git hidden directory is created
    - ✓ .git stores tracking information
  - To stop tracking your project

```
rm -rf .git
```

Tacking a Local Project

- To check the status of your project use git status

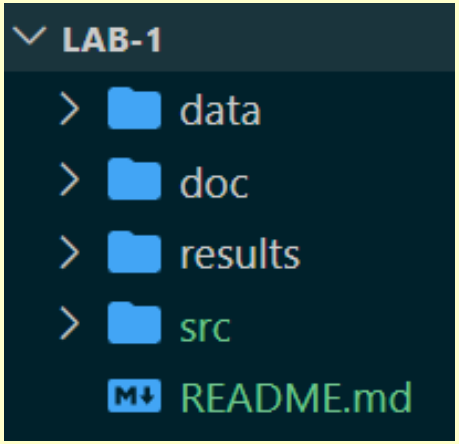
```
git status
```

- Add README.me

```
# lab-1
CPSC 4800 Lab 1
Your Name
ID# 45678909
```

- To check the status of your project

```
git status
```



Tacking a Local Project

- Add hello_world.py to your source code directory

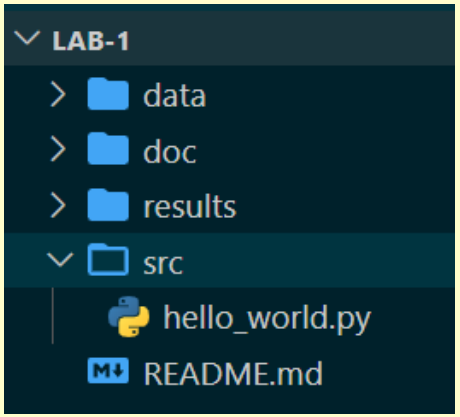
```
print(f'Hello, World!')
```

- To check any new file changes

```
git diff
```

- To check the status of your project

```
git status
```



Tacking a Local Project

- To track a file in your project

```
git add README.md
```

- To track all files in your project

```
git add --all
```

Tacking a Local Project

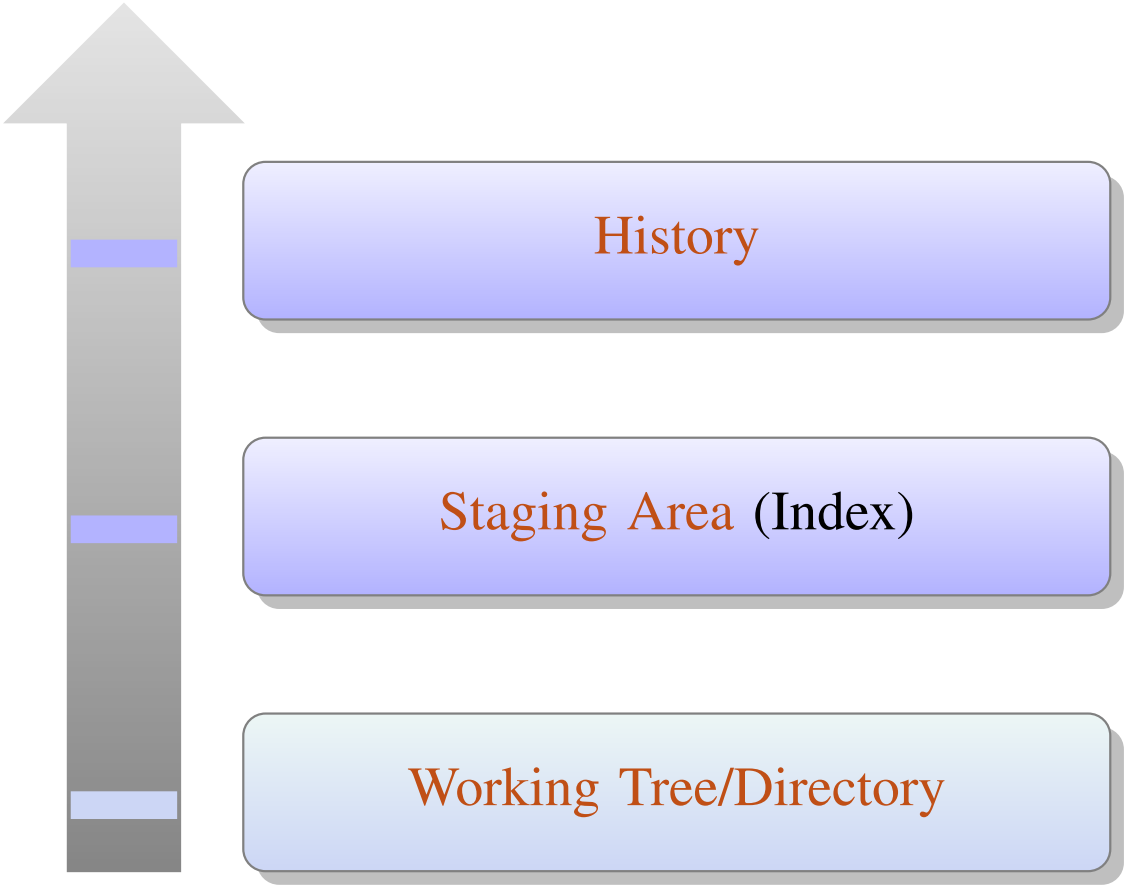
- To save a version of the tracked file(s)

```
git commit -m 'My first commit'
```

- To check all versions of the tracked file(s)

```
git log
```

Git Conceptual Areas



Git Terminology

Working Tree

The working tree is what see on the file system

Staging Area

The staging area stores the files that are tracked and ready to be committed. The git add moves the files from working tree to the staging area.

History

A history stores all changes we made in the .git hidden directory. After each git commit the history is updated.

Tacking a Local Project

- To track specific file in your project
- You have to add .gitignore file to your project

```
# .gitignore
data/*
doc/*
results/*
```

- To remove a file from a staging area

```
git reset hello_world.py
# remove everything from the staging area
git reset
```

Tacking a Local Project

- To revert back previous version

```
git checkout <<previous-hash-value>>
git checkout 8ddbe4ec42415c142abf1a29410a50b53115b824
```

Tracking Local Project Summary

```
$ git status
-----------------------------------------------------
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        src/
-----------------------------------------------------
$ git add --all
$ git status
-----------------------------------------------------
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   src/hello_world.py
-------------------------------------------------------
$ git commit -m 'My first git commit'
-------------------------------------------------------
[master (root-commit) ab3815d] My first git commit
 2 files changed, 5 insertions(+)
 create mode 100644 README.md
 create mode 100644 src/hello_world.py
-----------------------------------------------------
$ git log
commit ab3815d34c23d3c05439724b005018d1eadb8cf1 (HEAD -> master)
Author: Mourad Bouguerra <mbouguerra@langara.ca>
Date:   Sun May 15 15:16:36 2022 -0700

    My first git commit
-----------------------------------------------------
$ git checkout 8ddbe4ec42415c142abf1a29410a50b53115b824
```

Collaboration on a Remote Project

- ❒ Sign in to your GitHub account `https://github.com/`
- ❒ Create a Git repository with a name lab-2
  - ➡ Select the README.md checkbox
- ❒ Copy the url of your remote repo
- ❒ Open a terminal in you computer
- ❒ To make a local copy of your remote repo

```
git clone <<remote-repo-url>> <<local-destination>>
```

Collaboration on a Remote Project

- To make a local copy of your remote repo

```
git clone <<remote-repo-url>> <<local-destination>>
```

- Navigate to lab-2 directory
- Check the status of this Git repo
- To view remote repo information

```
git remote --verbose
# OR
git remote -v
```

Collaboration on a Remote Project

- Add the four subdirectories
  - data
  - doc
  - results
  - src
- Add hello_world.py file to the src folder
- Check the status of this Git repo

Class Activity

- Untracked files refer to which Git conceptual area?
  - Working tree
  - Staging Area
  - History

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Collaboration on a Remote Project

- Add .gitignore that instructs Git NOT to track the files in the following directories
  - data
  - doc
  - results

```
# .gitignore
data/*
doc/*
results/*
```

- Check the status of this Git repo

Collaboration on a Remote Project

- To update the remote repo with the local changes

```
git pull
git push
```

➡ It import to run git pull before git push

Tracking Remote Project Summary

```
$ git clone <<remote-repo-url>> .
-----------------------------------------------------------------
Cloning into 'lab-2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
-----------------------------------------------------------------
$ cd lab-2
-----------------------------------------------------------------
$ git status
-----------------------------------------------------------------
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean
-----------------------------------------------------------------
$ git remote --verbose
-----------------------------------------------------------------
origin  <<remote-repo-url>> (fetch)
origin   <<remote-repo-url>> (push)
-----------------------------------------------------------------
$ git status
On branch main
Your branch is up to date with 'origin/main'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        src/
-----------------------------------------------------------------
git add --all
```

Tracking Remote Project Summary

```
-----------------------------------------------------------------
$ git commit -m 'My initial remote repo changes'
-----------------------------------------------------------------
[main fd0fa94] My initial remote repo changes
 2 files changed, 5 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 src/hello_worl.py
-----------------------------------------------------------------
$ git log
-----------------------------------------------------------------
commit fd0fa94fb00e443e2628fc920908c46c16dbabeb (HEAD -> main)
Author: Mourad Bouguerra <mbouguerra@langara.ca>
Date:   Sun May 15 17:51:02 2022 -0700

    My initial remote repo changes
-----------------------------------------------------------------
$ git pull
-----------------------------------------------------------------
Already up to date.
-----------------------------------------------------------------
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 430 bytes | 430.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mbougerra/lab-2.git
   7f2138d..fd0fa94  main -> main
-----------------------------------------------------------------
```