### LEARNING OBJECTIVES

\_After this lesson, you will be able to:\_

- Explain basic git commands like init, add, commit, push, pull and clone

- Distinguish between local and remote repositories

- Create, copy, and delete repositories locally, or on GitHub

- Fork and clone remote repositories

### STUDENT PRE-WORK

\_Before this lesson, you should already be able to:\_

- Use the command line

- Use a text editor

## Introduction: Git vs GitHub and version control (10 mins)

#### What is Git?

- A program you run from the command line

- A distributed version control system

Programmers use Git so that they can keep the history of all the changes to their code. This means that they can rollback changes (or switch to older versions) as far back in time as they started using Git on their project.

You know how Google Docs allow you to have a "version history" and move between different versions whenever you want? Git enables you to do that with any folder - and its contents - on your computer!

A codebase in Git is referred to as a \*\*repository\*\*, or \*\*repo\*\*, for short.

\*\*\_Knowledge Check\_\*\* Who invented Git?

Git was created by [Linus Torvalds](https://en.wikipedia.org/wiki/Linus\_Torvalds), the principal developer of Linux.

#### What is GitHub?

- A hosting service for Git repositories

- A web interface to explore Git repositories

- A social network of programmers

- We all have individual accounts and put our codebases on our GitHub account

- You can follow users and star your favorite projects

- A place where developers can access public codebases

#### Can you use git without GitHub?

> Instructor Note: Explain the differences between 'remote' and 'local'.

Git is a software that enables version control on local folders on your machine. GitHub is a place to host your Git repositories, remotely. You can certainly have local files, that are using Git, that are not sent to or stored on GitHub.

Read [this](http://stackoverflow.com/questions/11816424/understanding-the-basics-of-git-and-github) when you have time!

> Check: Ask students to describe how remote files compare to local files. Describe how Git compares to GitHub.

## Demo: Why is Git tricky to understand? (15 mins)

Git is tricky to understand because describing 'how' it works would require the use of strange and technical-sounding words like:

- [Directed acyclic graph](https://en.wikipedia.org/wiki/Directed\_acyclic\_graph)

- [SHA-1](https://en.wikipedia.org/wiki/SHA-1)

- blob

- tree

However, you don't actually need to know how it works under the hood in order to use it.

#### Trees?!

Even though you don't need to know how they work, it is useful to know that your local repository consists of three "trees" maintained by Git.

- \*\*Working Directory\*\*: like any other folder on your machine; just holds the actual files

- \*\*Index\*\*: acts as a staging area (an area that holds files "ready" to be versioned)

- \*\*HEAD\*\*: points to the last commit you've made (the last "version" of the code you've stored using git)

![workflow](https://cloud.githubusercontent.com/assets/40461/8221736/f1f7e972-1559-11e5-9dcb-66b44139ee6f.png)

#### So many commands?!

There are also a lot of commands you can use in Git. You can take a look at a list of the available commands by running:

```bash

$ git help -a

```

Even though there are lots of commands, on the course we will really only need about 10.

#### Git File Lifecycle

To understand how Git works, we need to talk about the lifecycle of a Git-tracked file.

![lifecycle](https://cloud.githubusercontent.com/assets/40461/8226866/62730b4c-159a-11e5-89cd-20b72ed1de45.png)

Schema From [git-scm.com](https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository)

There are 4 main stages of Git version controlled file:

1. \*\*Untracked\*\*: When a file is "untracked," Git is not "watching" this file; the file will not be added in the next commit

2. \*\*Staged\*\*: Staged files have not yet been committed to memory but they are "on deck", so to speak, for your next commit

3. \*\*Unmodified\*\*: The file has already been committed and has not changed since the last commit

4. \*\*Modified\*\*: You have changes in the file since it was last committed, you will need to stage them again for the changes to be added in the next commit

Once you have committed a file and it becomes "unmodified" then it's contents are saved in Git's memory.

- \*\*Not saved in git memory\*\*: Your file is not saved until you commit the file to Git's memory

- \*\*Saved in git memory\*\*: Only once you have committed a file, it becomes saved in Git's memory

> Check: Work with the person next to you, get on the web, and take 3 minutes to come up with a one-sentence explanation of the difference between untracked and stage files. Get ready to share out.