Introduction to Version Control with Git

Alastair Droop, 2023-09-01

What is version control?

Version control allows you to:

- Keep *incremental backups* of your work
- Keep a record of *who modified what, when and why*
- *Work collaboratively* on a single body of work without introducing conflicts
- *Publish your work simply* with confidence that you're uploading the correct version

Incremental backups allow you to view (and recover) your work at a specific time in the past

If you don't already have a robust version control system in place, you probably should

What is Git?

Git is a source code manager (SCM) program that allows you to use version control

Git was created by Linus Torvalds in April 2005

Git is published under the GNU Public License version 2.0, so it is free and open source

See Git's website at: https://git-scm.com/

How does Git work?

Git tracks all the changes that occur in a *workspace*

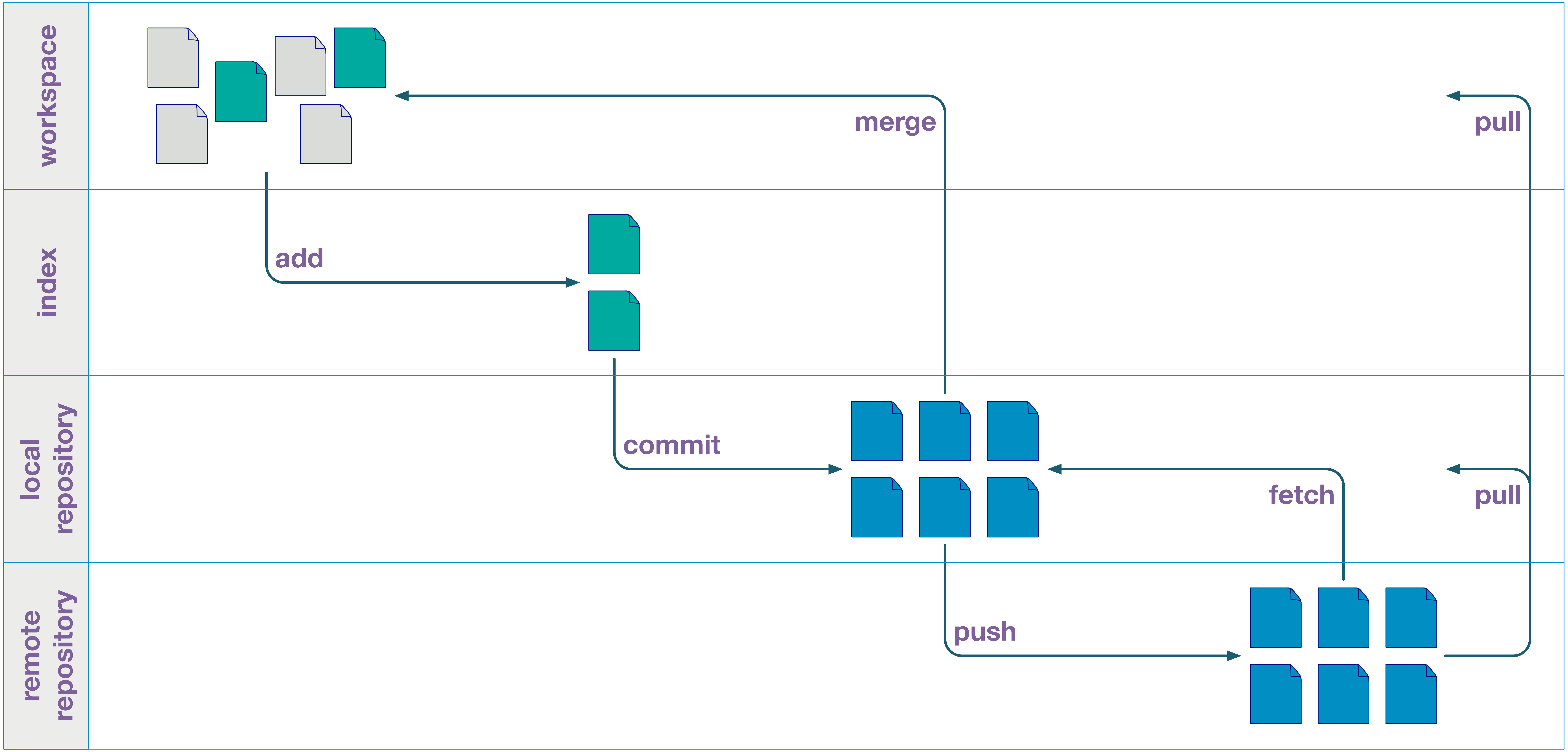
A set of changes to (one or more) files is collated into an *index* (staging area)

Once ready, all changes in the index are *committed* to the *local repository*

The local repository can be uploaded (*pushed*) to an *remote repository*

A repository can be *branched* to create a separate line of development in a workspace

Branches can be *merged* back into the main codebase when required



https://cloudstudio.com.au/2021/06/26/git-command/

The Workspace

The *workspace* is simply a directory on your computer containing a *local repository*

You modify files in the workspace as normal

The only thing that makes it a workspace is that it contains a local repository (a `.git` subfolder)

Command	Description
`git init`	Create a new (empty) local repository within the current directory
`git status`	Get the status of the local repository

The Index

The *index* a staging area for files that will be committed to the local repository

You add modified files to the index and when ready *commit* them to the local repository

Command	Description
`git add`	Add modified files from the workspace to the index
`git rm`	Remove a file from both the workspace and the index
`git status`	Show the status of the index (things that have been changed, etc)
`git diff`	Shows modified files not yet in the index
`git commit`	Stores the current contents of the index into the repository with a message

The Local Repository

A *local repository* is a folder that the git software controls containing the history of a workspace

A local repository resides in a folder called `.git` within a workspace

Command	Description
`git log`	Displays a log of the recent commits and their messages
`git branch`	Lists & creates branches in the local repository
`git checkout`	Check out a specified branch from the local repository

The Remote Repository

A *remote repository* is a repository on a remote server

The most common remote repository site is GitHub (https://github.com/)

Command	Description
`git push`	Update a remote repository with the contents of the local repository
`git fetch`	Update the local repository with the contents of a remote repository

Stashes

As well as branches, git allows you to *stash* modifications away whilst you work on something else

These are stored separately to commits, so they don't appear in the index

Stashes are useful when you want to make a quick change but don't want to record it

Command	Description
`git stash push`	Save the current modifications to a new stash then remove them for the workspace
`git stash pop`	Applies the changes in the latest stash to the workspace and removes the stash
`git stash apply`	Applies the changes in the latest stash to the workplace
`git stash list`	Lists the stashes you currently have

Commits

A set of modifications to the workspace is called a *commit*

The act of recording these changes into the local repository is *committing*

When committing a set of changes, you should specify a *commit message* that describes the changes

Good messages consist of

- A short (less than 50 characters) title; and
- A longer description if necessary

Decide on a standard format for your commit messages, and stick to it

Each commit gets a UUID (eg. 603a39fd53d31c01db003e9948ddf0c7c136e8d2 shortened to 603a39f)

Branches

Quite often you’ll want to work on a specific part of your workspace without wanting those changes to become part of your main repository.

For example

- Working on a modification to a thesis chapter
- Working on a new feature for a piece of software

In these cases, you can make a *branch* of a local repository

You can have multiple branches in a repository at one time

Each time you swap to a branch, git will update the workspace to reflect that branch

Once you’re ready, you can *merge* branches together to incorporate your work into the main repository

HEAD
feature 1
main
feature 2
feature 3

- feature 1 was branched off from main but then merged back later
- feature 2 was branched off main, but was never merged back
- feature 3 branched off feature 2 and was then merged back into main after feature 1

HEAD is a reference that tells git what the current workspace is at in the branch tree

Forks

If you want to modify code in a remote repository that you don't have access to edit:

- You can't make a new branch, as you don't have permission to edit the repository
- You can simply *fork* the repository instead

A *fork* is a clone of an existing remote repository that you control

Think of a *fork* like a branch, but not as closely integrated into the original

Integration

Most modern text editors have *git integration*

This makes the process of creating commits and pushing to remote repositories simple

If your favourite editor has git integration, take the time to install it and learn how to use it

VSCode has builtin git integration

See https://code.visualstudio.com/docs/sourcecontrol/overview#_git-support

Installing Git

Git is sometimes difficult to install, especially on manages PCs

If possible, get your local IT to install on manages computers

On a Mac, the easiest way is to install git as part of Xcode from the App Store

On Windows, you can use git for Windows from https://gitforwindows.org/

Online Help & Resources

The Git website	https://git-scm.com
The GitHub website	https://github.com
Git for Windows	https://gitforwindows.org/
Git command reference	https://git-scm.com/docs
The Git Book	https://git-scm.com/book/en/v2
Useful introduction videos	https://git-scm.com/videos
Git visual cheatsheet	https://ndpsoftware.com/git-cheatsheet.html
VSCode git integration	https://code.visualstudio.com/docs/sourcecontrol/overview#_git-support