UNIVERSITY OF LIVERPOOL

LIFE748

INTRODUCTION TO GIT

Applied version control with Git

DR TARANG MEHTA – Tarang.Mehta@liverpool.ac.uk

LEARNING OUTCOMES (LO)

By successfully engaging in this lecture and workshop, you will be able to:

LO1: Basic workflow of Git, including initialising a repository, making commits, and pushing changes to a remote repository

LO2: Use Git for version control, including creating branches and merging changes.

LO3: Gain knowledge of GitHub as a platform for hosting and collaborating on Git repositories.

SKILLS (S)

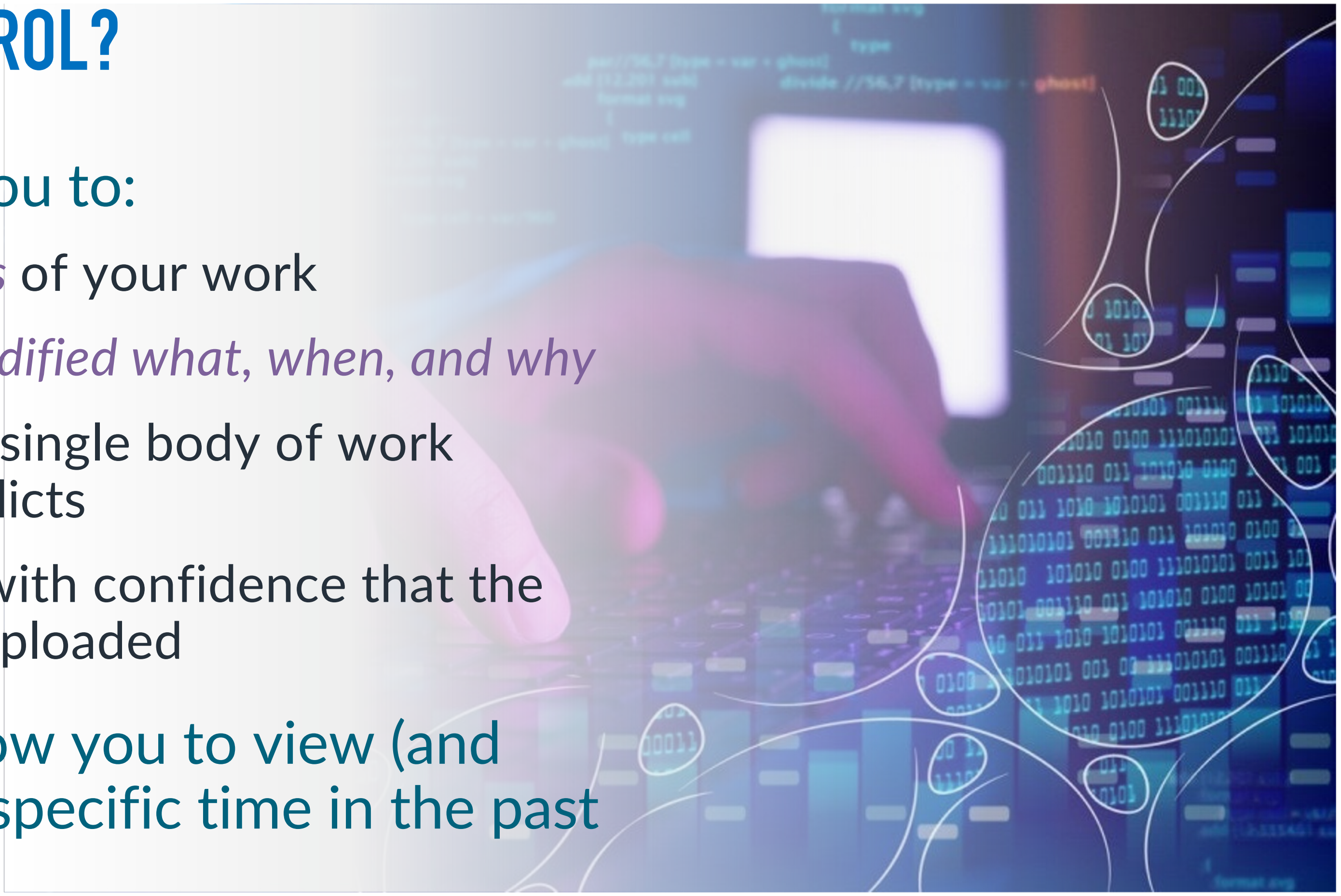
By successfully engaging in this lecture and workshop, you will gain skills in:

S1: Command-line proficiency - Practice using terminal commands for Git operations and file management

S2: Code versioning - How to track changes in code and revert to previous versions if needed.

S3: Markdown usage - Gain experience in writing documentation using Markdown syntax.

WHAT IS VERSION CONTROL?

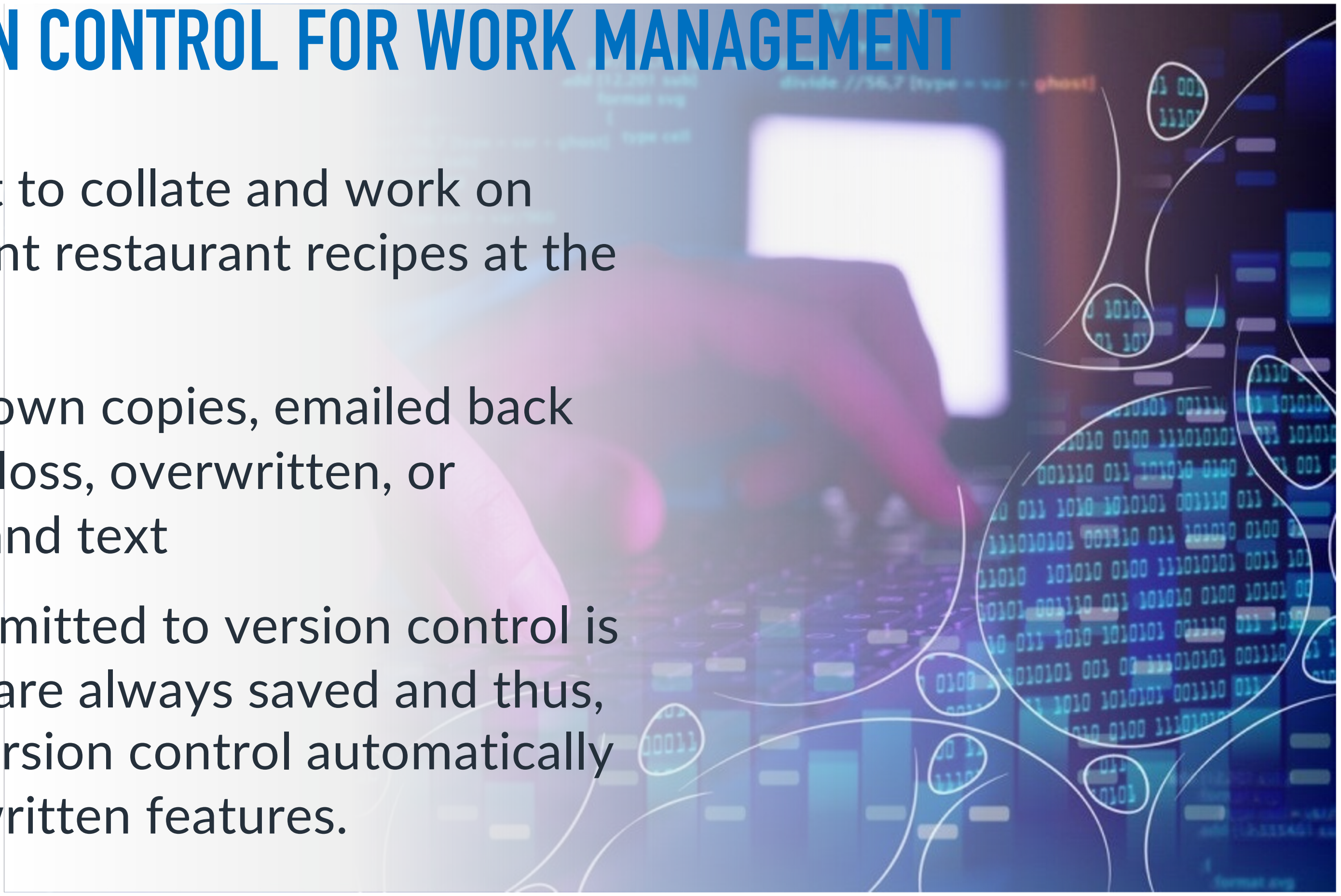


Version control allows you to:

- 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 the correct version is being uploaded

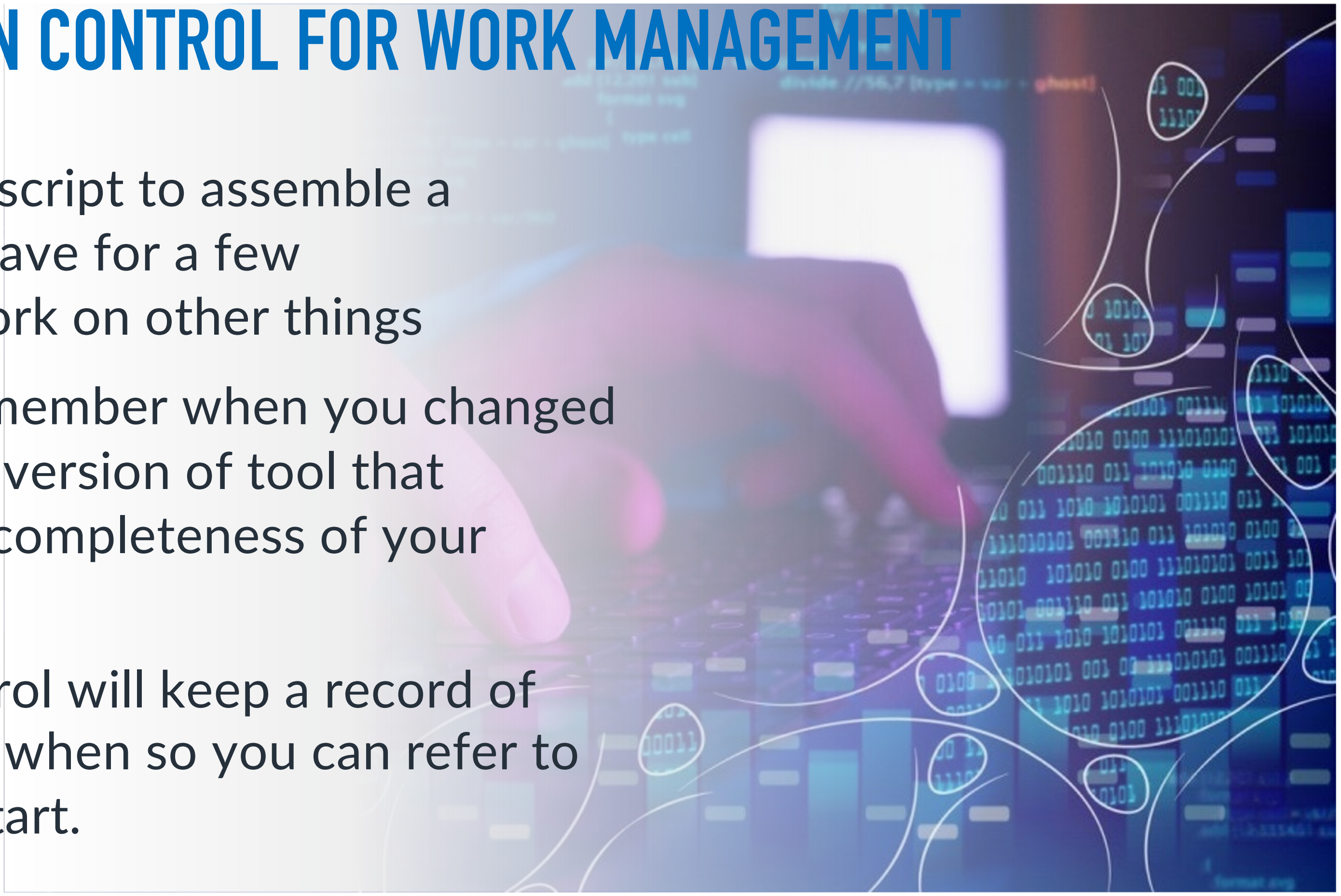
Incremental backups allow you to view (and recover) your work at a specific time in the past

EXAMPLE #1 OF VERSION CONTROL FOR WORK MANAGEMENT



- **Teams:** Two people want to collate and work on price indexing for different restaurant recipes at the same time.
  - **Problem:** Working on own copies, emailed back and forth will result in loss, overwritten, or duplication of figures and text
  - **Solution:** Nothing committed to version control is ever lost, old versions are always saved and thus, tracking is possible. Version control automatically notifies users of overwritten features.

EXAMPLE #2 OF VERSION CONTROL FOR WORK MANAGEMENT



- **Single users:** You code a script to assemble a genome, that you then leave for a few weeks/months as you work on other things
  - **Problem:** You can't remember when you changed a certain parameter or version of tool that resulted in a different completeness of your genome
  - **Solution:** Version control will keep a record of what has changed and when so you can refer to all changes since the start.

MOVING FORWARD WITH VERSION CONTROL

- If you don’t have a robust version control system in place for your project, consider how you can implement from now
- We will want to see this as part of your Assessment 3 project poster assessment!

WHAT IS GIT?

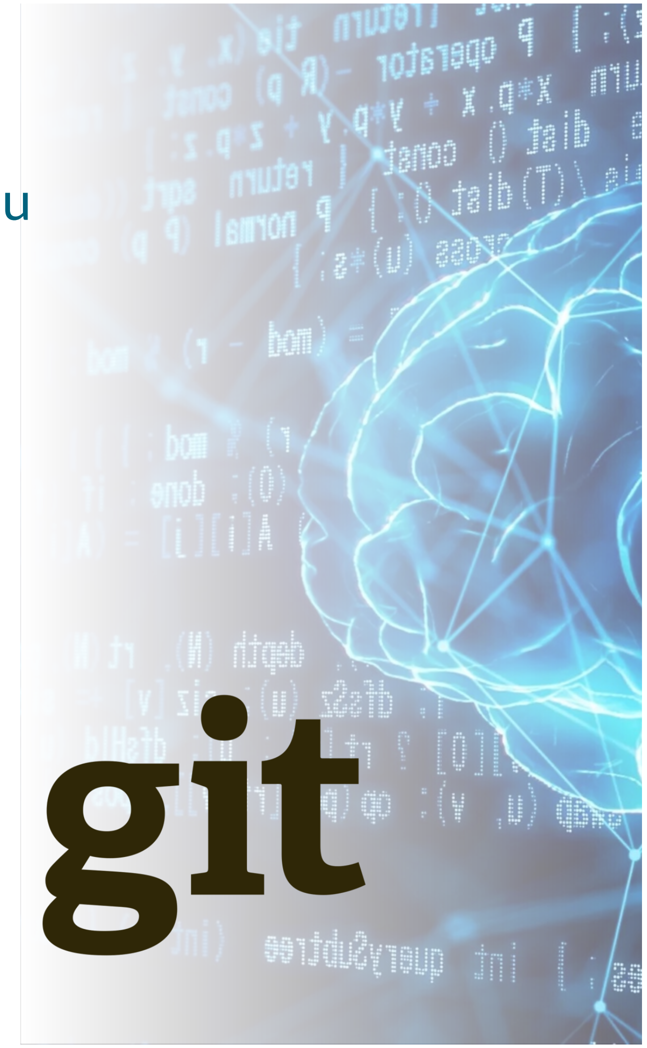
- Git is a source code manager (SCM) program that allows you to use version control

- 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 to use

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

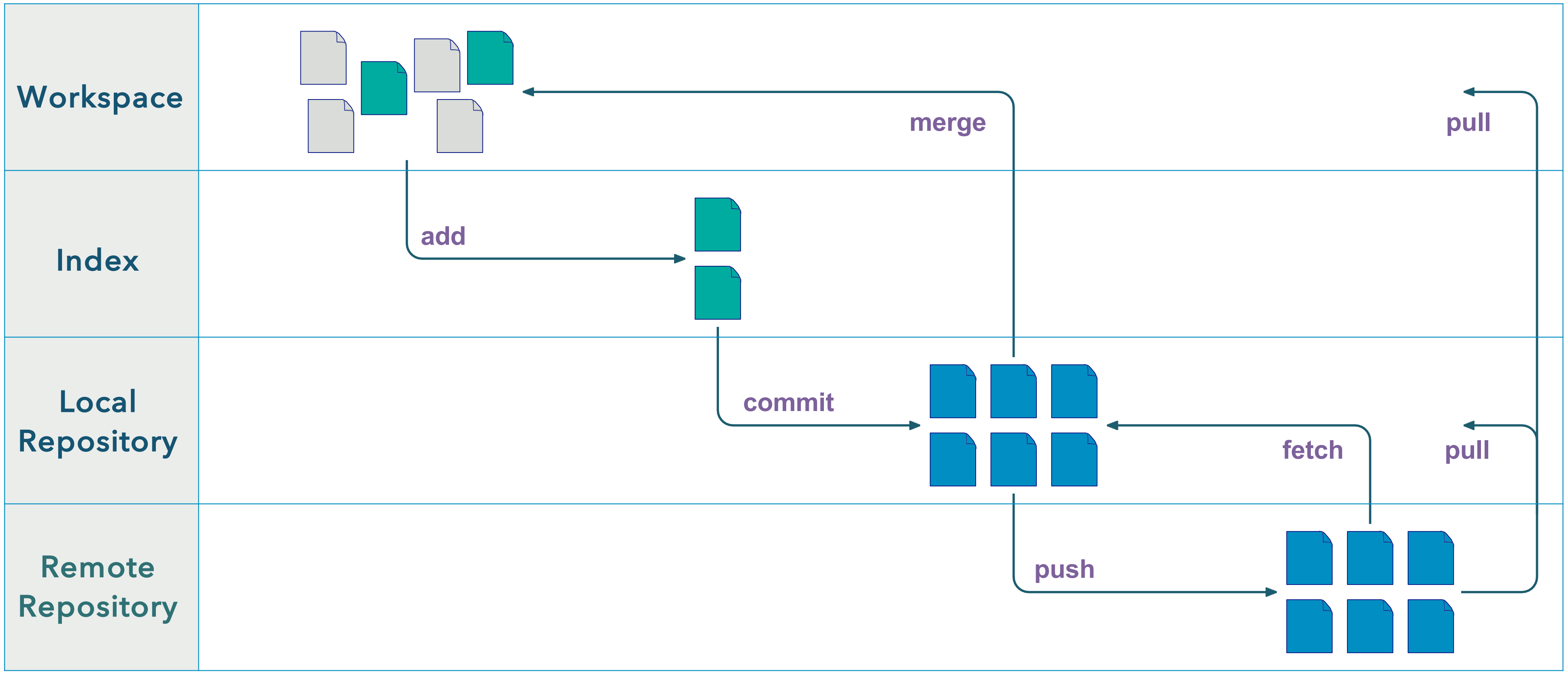


HOW DOES GIT WORK?

- 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 a *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

UNIVERSITY OF LIVERPOOL

WORKFLOW OF GIT COMMANDS



Adapted from 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 .gitwithin 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

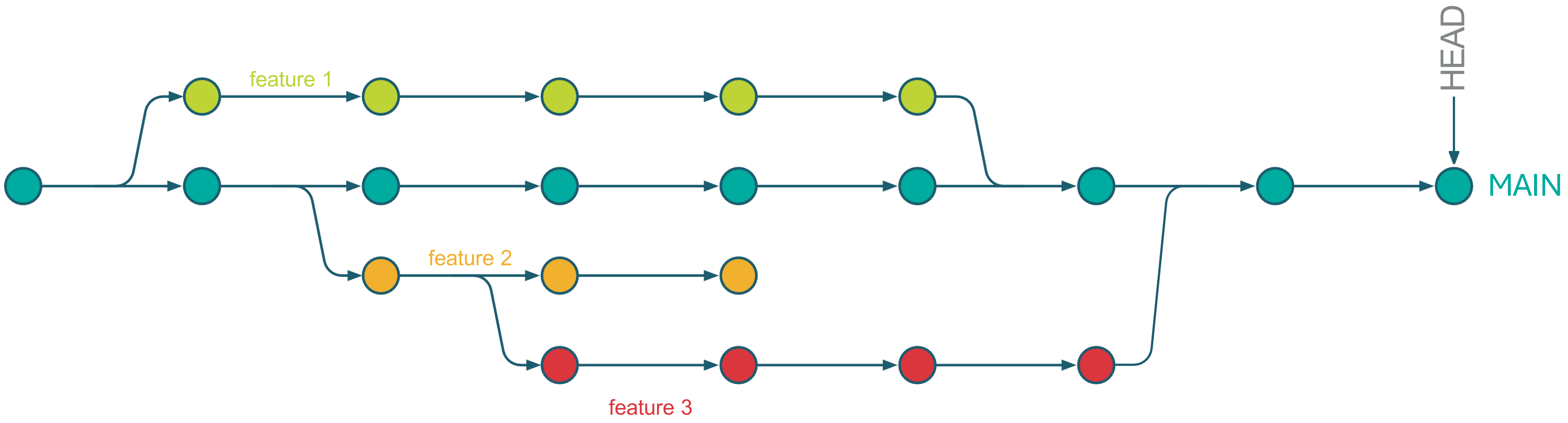
- A set of modifications to the workspace is called a *commit*
- The act of recording these changes into the local repository is *committing*
- When committing a set of changes, you should specify a *commit message* that describes the changes Good messages consists of:
    - A short (less than 50 characters) 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. 504a42dd53d31c01db003e9948ddf0c7c136e8d2 shortened to 504a42d)

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 e.g., 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

FOR EXAMPLE . . .



- 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

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 e.g., VSCode has built-in git integration and is a great code/text editor!

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

INSTALLING GIT

- Git is sometimes difficult to install, especially on managed PCs
- Please follow the workshop guidelines for installation

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

PLEASE COMPLETE THE SHORT 5 QUESTION POLL PRIOR TO THE WORKSHOP

https://pollev.com/tarangmehta193