1. **Video 25: Module Introduction**

* Module Content



1. **Video 26: How Git works**

* It is a version management tool
* Turns a folder into a working directory by creating commits
* Commits are updated versions of your working folder
* Track changes through commits
* Git stores changes in the Master/Main branch

1. **Video 27: Theory Working Directory vs Repository**

* Working directory: Project folder managed by Git
* .git folder: Git repository where version management takes place
  + Staging area: Stores files to get them ready for the next commit
  + Commits (Object folder): hosts updated files
* NB!: git track changes, it does not store files

1. **Video 28: Theory - Understanding Branches**

* Branches: sub-folders under the master branch
* Master branch: where all commits are stored (different versions)
* Branches allow you to create copies of repositories for you to work on and later be able to merge the copy with the master branch.

1. **Video 30: Installing Git on Windows**

* Go to <https://git-scm.com>
* Click on download for Windows
* Open the downloaded zip file
* Click next on the Installer pop-up screen
* Click next on Select components screen (unselect windows explorer integration)
* Click Next on the Choosing the default editor used by Git (Select default option)
* Click next on Adjust the name of initial branch in new repository (Select default option)
* Click next on Adjust your path environment (Select default option)
* Click next on Choosing HTTPS transport backend (Select default option)
* Click next on Configuring the line ending conversions (Select default option)
* Click next on Configuring the terminal emulator to use with Git Bash (Select default option)
* Click next on Choose the default behavior of “git pull” (Select default option)
* Click next on Choose a credential helper (Select default option)
* Click next on Configuring extra options (Select default option)
* Configuring experimental options : Keep boxes unticked
* Click on Install
* Untick Launch Git Bash and View Release Notes
* Then click on finish
* To if Git was installed, go to command prompt:
  + git –version: shows you the git version installed

1. **Video 32: Installing Visual Studio Code**

* Got <Https://Code.Visualstudio.com>
* Click on Download for Windows
* Open the downloaded file
* Accept the agreement
* Select destination location (Browse) then click next
* Select Start Menu Folder then click next
* Select additional tasks then click next
* Click Install
* Click Finish
* VSC will open automatically

1. **Video 33: Initializing the Repository & Creating the First Commit (git init & git commit)**

* **git status:** Gives you a status on your working directory
* **git init:** initialises a repository
* **git add .** : Stages your changes
* **git** commit -m “commit message”: Commits your changes

1. **Video 35: Diving Deeper Into Commits with git log**

* **git log:** shows you all information on branches and files and commits (Including commit ID’s) that exist under them
* **git checkout [commit ID]:** commits a specific commit/change

1. **Video 36: Understanding & Creating Branches**

* **git branch:** shows you all your branches
* **git branch [branch name]:** creates a new branch
* **git checkout [branch name]:** switches to a specific branch
* **git checkout -b [branch name]:** creates and switches to a branch in one step
* **git add [file name]:** stages a specific file

1. **Video 37: Merging Branches - The Basics**

* **git merge [branch name]:** merges branches

1. **Video 38: Understanding the HEAD**

* The head is the latest/last commit in a branch after a branch is checked-out

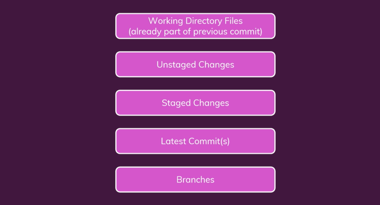
1. **Video 39: Detached Head**

* A checked out commit does into detached head state is not part of a specific branch.

1. **Video 40: Branches & git switch (Git 2.23)**

* **git switch [branch name]:** Creates a new branch
* **git switch -c [branch name]:** creates a new branch and switches to it

1. **Video 41: Deleting Data - An Overview**



1. **Video 42: Deleting Working Directory Files**

* **git ls-files:** shows which files are part of our staging area
* **git rm [file name]:** removes a file in the staged area, after this command the change needs to be committed.

1. **Video 43: Undoing Unstaged Changes**

* git checkout [file name]: revert Unstaged changes
* git checkout .: revert all Unstaged changes (not specific)
* git restore [file name]:: revert Unstaged changes
* git restore .: revert all Unstaged changes (not specific)
* git clean -dn: lists untracked files
* git clean -df: deletes untracked file

1. **Video 44: Undoing Staged Changes**

* **Git reset [file name]:** copies the latest commit stage of a file into the staging area, then **git checkout [file name]:** undo staged change

**Alternatively:**

* **git restore - - staged [file name]:** upstages a change

1. **Video 45: Deleting Commits with git reset**

* **git reset - -soft HEAD~1:** Takes you back to the status before 1 (last commit) commit
* **git ls-files:** shows which files are part of our staging area (Tracked files)
* then stage and commit your changes

**Alternatively:**

* **git reset HEAD~1**: resets your latest commit
* then stage and commit your changes

**Alternatively:**

* **git reset - -hard HEAD~1:** removing the change from working directory, removing commit from staging area as well as in your working area.

1. **Video 46: Deleting Branches**

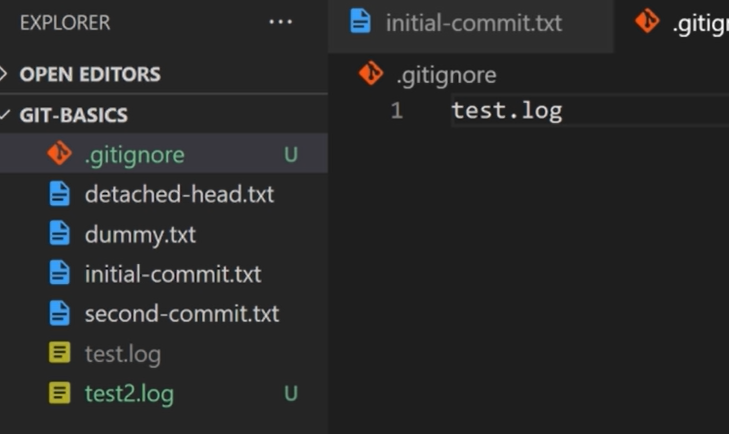
* **git branch -D [branch name]:** Forces branch deletion, merged or not merged
* **git branch -d [branch name:** deletes merged branches
* **git branch -D [branch-1 branch-2]:** deletes multiple branches

1. **Video 47: Committing detached HEAD Changes**

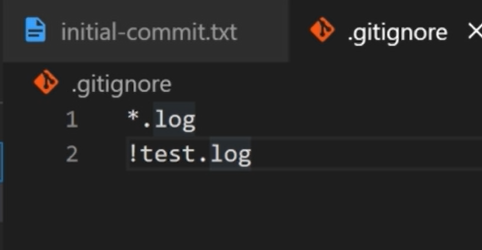
* Directly checking out a branch or commit
* **git checkout [commit ID]:** detach a specific commit
* make changes
* Create new branch
* stage the changes and commit them to the new branch
* Switch to master
* Merge new branch into master branch

1. **Video 48. Understanding .gitignore**

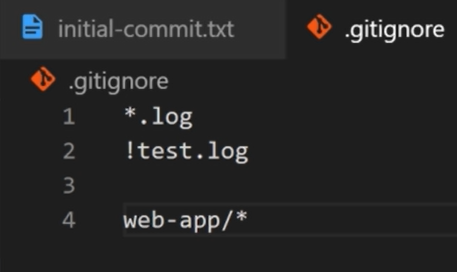
* **File should be created manually on VSC (.gitignore)**
* **Add file names in this file that git needs to ignore:**



* \*.log: tells git to ignore all files ending with .log
* ! file name and file type: tells git to ignore all .log files excluding this one



* Folder name/\*: tells git to ignore a folder name including its contents



* **git clean -df:** deletes all file and folders created and not staged or committed

1. **Video 49: Wrap Up & Basic Commands Overview**

* **Basic commands Summary:**



* **Commit creation and access**



* **Branch creation and access**



* **Deleting Data**

