General

* git log --oneline
  + More readable version of logs
* git status
  + Always check when in a folder
* git init
  + Creates a git repository
* git branch
  + Displays all branches
* .git/HEAD
  + Used in CLI to see where head is pointed at.
* git branch -M main
  + Changes from "master" to "main"
  + Can be whatever name you want

Branches

* Alternate timelines for a project
* Enable us to create separate contexts where we can try new things or work in parallel
  + Changes on one branch do not impact other branches unless we merge
* Each branch references a previous commit
  + Head
    - Points to a particular branch (current location)
* <git branch>
  + Shows all active branches
  + (\*) denotes active branch
  + git branch <branch name> creates new branch
* Git branch -r

* <git switch>
  + Switches branches
  + <git checkout> is the same as git switch
  + Git switch -c <branch name> creates and switches branch
  + Git switch -
    - Takes you back to previous branch
* Delete / rename
  + Git branch -d <branch name>
    - 'D' will force delete branch regardless of merge
  + To rename, you have to be in specified branch
    - Git branch -m <new name>
* Merging
  + Move to receiving branch then use 'git merge <file name>'

Commit

* git commit -m "<message>"
* Git commit -am "<message>" commits and adds files in one

Conflicts

Graphical user interface, application

Description automatically generated

* Remove markers and decide what to do with conflicts

Git Diff

* git diff
  + Compares staging area and working directory
* git diff HEAD
  + Shows changes that are staged and unstaged
* git diff --staged / git diff --cached
  + Only shows staged changes
* git diff HEAD <filename> <filename>
  + Narrows search
* git diff <branch>..<second branch>
  + Compare changes between branches
    - Can be a space instead of ".."
    - ORDER MATTERS!

Stashing

Used when needing to switch to a different branch but not wanting to commit or bring changes with you to another branch.

* git stash / git stash save
  + Commits are considered "official" points in a project. This does not show on logs.
  + Can have multiple stashes in the order they were created.
* git stash pop
  + Removes most recent stash and re-apply them to your working copy
  + Used once you return to directory you were working on
* git stash apply
  + Allows you to apply the stash changes in multiple branches without completely removing the stash
* Git stash list
  + Display all current stashes
    - git stash apply stash@{2}
      * Selects a particular stash instead of last one.
* git stash drop <stashid>
  + Removes a particular stash
* git stash clear
  + Removes every stash.

Checkout

Allows you to go back to different points in time in a repo. In a detached state.

* Head points at a commit rather than at the branch pointer.

* git checkout <commit-hash>
* git checkout HEAD~1
  + Goes back one commit
    - Change number to however many commits you want to step back.
* git checkout HEAD <filename> git checkout -- <filename>
  + Reverts back to previous state
  + Used if you want to remove a lot of changes that were made before a commit

Restore

* git restore <filename>
  + Takes some of the burden from checkout
  + Goes back to head
* git restore --source HEAD~2 <filename>
  + There is no going back. Files are deleted permanently
* git restore --staged <filename>
  + Allows you to remove files from staging

Reset

Resets repository to a specific time

Regular and hard resets

* git reset <commit-hash>
  + Changes are kept. Commit hashes are removed.
* Git reset --hard <commit-hash>
  + Undoes commits and changes

Revert

Creates a new commit when it undoes previous changes. Will be prompted to enter a commit message

Allows users to keep a commit but not its changes. Useful for projects with others

* Git revert <commit-hash>

Github Cloning

* Git clone <url>
  + Downloads all of the files and history of selected repository
  + Creates repo in CWD
* SSH Config
  + Follow steps on github website. <https://docs.github.com/en/authentication/connecting-to-github-with-ssh>
  + Wont be able to push commits without this

Git Remote

* Git remote add <name> <url>
  + "origin" is a common name to use
* Git remote -v
  + Displays origin url

Push

* Git push <remote> <branch>
* Git push <remote> <local-branch>:<remote-branch>
  + Allows you to oush to a different branch if yours are different
* Git push -u origin master
  + Used when creating a repo. Allows you to use the shortcut "git push". Stands for upstream

Fetch

Allows the user to bring changes to computer but not putting it in your current working files

* Git fetch <remote> / git fetch <remote> <branch>

Pull

Gets changes from remote branches and updates current branch with new changes merging them in

\*\*not recommended if you have uncommitted changes\*\*

* Git pull <remote> <branch> / git pull (automatically defaults to "origin")

Pull Requests

Allows developers to alert team members and submit code for approval