Git Cheat Sheet

Git is a version control system.

The essentials: Using Git

`git clone`	Clone a Git repository to your local computer
`git fetch`	Fetch changes from a remote repository
`git pull`	Fetch and merge changes from a remote repository
`git status`	See a summary of local changes, remote commits, and untracked files.
`git diff`	See specific local changes. Use **--name-only** to see filenames.
`git add`	Stage changes to tracked and untracked files.
`git commit`	Create a new commit with changes previously added.
`git push`	Send changes to your configured remote repository (like GitLab or GitHub).

Important options: Keeping things organized

`git reset HEAD --`	Get back to the last known commit and unstage files.
`git add -u`	Add only updated, previously committed files.
`git log --graph --oneline`	See a pretty branch history. Create an alias (**git lg**) for easy access.

Basic branching: Branches represent a series of commits

`git branch --all`	List all local and remote branches
`git checkout bugfix`	Change to an existing branch called **bugfix**
`git checkout -b dev main`	Make and checkout a branch called **dev** based on **main**

`git checkout main`
`git merge dev`
	Merge branch changes from **dev** into **main**

Pushing changes: Sending data from your local repository to a remote repository

`git remote -v`	View all configured remotes
`git push origin HEAD`	Push commits located at the HEAD of your repo to the **origin** repo
`git push origin +HEAD`	Push commits, forcing remote to adopt local changes
`git push origin -d dev`	Delete **dev** branch from remote after pushing changes

Git Cheat Sheet

Basic flow: Daily usage of Git, including important options	

`git init demo && cd demo`
`cp ~/Code/mycode.py mycode.py`
`git add mycode.py`
`git commit -m 'My first commit'`
`git show`
	Initialize a local Git repository, creating the directory if it doesn't exist. Change directory to the repo, add files, and commit.

`git diff`
`git commit --all -m 'Another commit'`
	As you begin to hack on local files, you commit them at regular intervals. The **--all** option commits changes to existing files (use **git add** to add new files).

`git log --graph --abbrev-commit`
`git reset --soft HEAD~3`
`git diff --cached`
`git commit -am 'Message for 3 commits'`
	After a while, you have 3 commits that are meaningful as a single commit.

`git push origin HEAD`	Lastly, you push your local changes to a remote repository, designated as **origin**.

Working with a remote repository: Contributing to public repositories	
`git fetch --all`	Download all commits and references
`git pull --rebase <remote> <branch>`	Merge all commits since your last common commit from the remote branch without a merge commit
`git stash`	Save uncommitted changes
`git stash pop`	Restore saved changes
`git add <file>`	Add a file to the staging area, to be committed
`git commit -m 'commit message'`	Most projects have a format for commit messages.
`git checkout -b <new_branch>`	Create and checkout a branch
`git checkout main && git pull --rebase`	Checkout and update the main branch
`git reset head --hard origin/main`	WARNING: Erase all local changes
`git push -u origin HEAD`	Push your changes and the current branch to the **origin** repository
`git push origin HEAD`	Push your changes to the **origin** repository