Commands

Git Setup

Git Init
Create an empty Git repository or reinitialize an existing one.

Git config --global user.name <name>
Define the author name to be used for all commits by the current user.

Git config --global user.email <email>
Define the author email to be used for all commits by the current user.

Git clone <url to remote repository>
Clone a remote repository into a local directory.

Git Workflow

Git Status
List which files are staged, unstaged, and untracked.

Git Diff
Show unstaged changes between your index and working directory.

Git Diff –staged
Show what is staged but not yet committed to local repository

Git Add < . > / <filename> / <directory>
Stage all changes < . > / in <directory> / or < filename > for the next commit.

Git Commit -m “<commit message>”
Commit your staged content to the local repository

Git Push <url to remote repository>
Push the branch to <remote>. Creates named branch in the remote repository if it doesn’t exist.

Git Notes add -m “<note message>” <object-id>
Add or remove notes (attached) to objects (commits or patches).

Git Branches

Git Branch <new branch name>
List all of the branches in your repository. Use **< new branch name >** to create a new branch.

Git Switch <branch name>
Switch to the branch **< branch name>.**

Git Merge <branch name>
Merge <branch> into the current branch.

Git Checkout <branch name>
Switch branches or restore working tree files.
Use the **-b flag** to checkout an existing branch.

Git Log
Display the entire commit history for the current branch.
Additional options:

–oneline	:	show commit properties in 1 line
–all	:	show commit history for all branches
–graph	:	text-based graphical representation of the commit history

Git Update

Git Remote add <name> <url>
Create a new connection to a remote repo.
After adding a remote, you can use **<name>** as a shortcut for **<url>** in other commands.

Git Fetch <branch>
Fetches a specific **<branch>**, from the repo. Leave off **<branch>** to fetch all remote refs.

Git Pull <remote>
Fetch the remote's copy of its current branch and immediately merge it into the local copy.

Git Merge
Join two or more development histories together.

Git Rewrite history

Git Rebase <base>
Rebase the current branch onto **<base>**.
The **<base>** can be a commit ID, branch name, a tag, or a relative reference to HEAD.

Git Commit –amend
Replace the last commit with the staged changes and last commit combined.
Use with nothing staged to edit the last commit's message.

Git Clean -n / -f
Shows which files would be removed from the working directory.
Use the **-f** flag in place of the **-n** flag to execute the clean.

Git Cherry pick <name>
Apply the changes introduced by some existing commits.

Git Stash
Stash the current state of the working directory so you can switch to another branch and come back later to re-stash your work and continue. All without committing or adding your work.

Git Revert <commit>
Create a new commit that undo all of the changes made in **<commit>**, then apply it to the current branch.

Git Restore
Replace the last commit with the staged changes and last commit combined.
Use with nothing staged to edit the last commit's message.
–staged <file name> : restore index to previous save state.
--source <branch name>~<commit index><file name> : on branch restore to commit index.

Git Reset
Clears the staging area and undo commit, merge or pull.

Git rm <file name>
Remove files from the working tree and from the index.