Git Cheatsheet

Adding changes

`git add -u <path>`	Add all tracked files to the **staging area**.
`git add -p <path>`	Interactively pick which files to **stage**

Storing changes

`git stash [push] [path]`	Put current changes in the **working tree** into **stash** for later use.
`git stash pop`	Apply stored **stash** content into **working tree**, and clear **stash**.
`git stash drop`	Delete a specific **stash** from all the previous **stashes**.

Inspecting diffs

`git diff [path]`	Show changes between **working tree** and **staging area**.
`git diff --cached/--staged [path]`	Show any changes between the **staging area** and the **repository**.
`git diff > file.patch`	Generate a patch file for current changes

Reverting changes

`git rebase`	Rebase the current branch on top of another specified branch.
`git rebase -i [commit sha]`	Start an interactive rebase.
`git revert [commit sha]`	Create a new commit, reverting changes from the specified commit. It generates an **inversion** of changes.
`git checkout <path>`	Discard changes in the **working tree**.
`git restore [-W/--worktree] <path>`	Discard changes in the **working tree**.
`git restore -S/--staged <path>`	Remove a file from a **staging area**.
`git restore -SW <path>`	Discard changes in the **working tree** and to the **staged** files
`git reset <path>`	Remove a file from the **staging area**.
`git reset [mode] HEAD^`	Remove the latest **commit** from the current branch and: • `--soft` - keep file changes in the **working tree** and **stage** them; • `--mixed` - keep file changes; • `--keep` - reset only files which are different between current `HEAD` and the last commit • `--hard` - do **not** keep file changes.

Tagging commits

`git tag`	List all tags.
`git tag <name> [commit sha]`	Create a tag reference named `name` for the current or specific commit.
`git tag -a <name> -m <message>`	Create an annotated tag with the given message.
`git tag -d <name>`	Delete the tag with the given name.

Synchronizing repositories

`git fetch [remote]`	Fetch changes from the **remote**, but not update tracking branches.
`git fetch --prune [remote]`	Delete remote refs that were removed from the **remote** repository.
`git pull [remote]`	Fetch changes from the **remote** and **merge** current branch with its upstream.
`git pull -r/--rebase [remote]`	Fetch changes from the **remote** and **rebase** current branch on top of the upstream
`git push -u [remote] [branch]`	Push local branch to **remote** repository. Set its copy as an upstream.