# GIT CHEAT-SHEET

**$ git config --global user.name "Your Name"**
Set the name that will be attached to your commits and tags.

**$ git config --global user.email "you@example.com"**
Set the e-mail address that will be attached to your commits and tags.

**$ git config --global color.ui auto**
Enable some colorization of Git output.

**$ git init [project name]**
Create a new local repository. If [project name] is provided, Git will
create a new directory name [project name] and will initialize a
repository inside it. If [project name] is not provided, then a new
repository is initialized in the current directory.

**$ git clone [project url]**
Downloads a project with the entire history from the remote repository.

**$ git status**
Displays the status of your working directory. Options include new, staged, and modified files. It will retrieve branch name, current commit identifier, and changes pending commit.

**$ git add [file]**
Add a file to the staging area. Use in place of the full file path to add all changed files from the current directory down into the directory tree.

**$ git add .**
Add all the files in the directory to staging area.

**$ git diff [file]**
Show changes between working directory and staging area.

**$ git diff --staged [file]**
Shows any changes between the staging area and the repository.

**$ git commit**
Create a new commit from changes added to the staging area. The commit must have a message!

**$ git commit -m "[descriptive message]"**
commit your staged content as a new commit snapshot

**$ git rm [file]**
Remove file from working directory and staging area.

**$ git stash**
Put current changes in your working directory into stash for later use.

**$ git stash pop**
Apply stored stash content into working directory, and clear stash.

**$ git stash drop**
Delete a specific stash from all your previous stashes.

**$ git branch**
List all local branches in repository.

**$ git branch [branch_name]**
Create new branch, referencing the current HEAD.

**$ git checkout**
switch to another branch and check it out into your working directory

**$ git checkout [-b][branch_name]**
Switch working directory to the specified branch. With -b: Git will create the specified branch if it does not exist.

**$ git merge [from name]**
Join specified [from name] branch into your current branch (the one you are on currently).

**$ git branch -d [name]**
Remove selected branch, if it is already merged into any other. -D instead of -d forces deletion.

**$ git log**
List commit history of current branch.

**$ git log --oneline**
An overview with reference labels and history of commits. One commit per line.

**$ git reset [file]**
Revert your repository to a previous known working state.

**$ git reset [--hard] [target reference]**
Switches the current branch to the target reference, leaving a difference as an uncommitted change. When --hard is used, all changes are discarded.

**$ git revert [commit sha]**
Create a new commit, reverting changes from the specified commit. It generates an inversion of changes.

**$ git fetch [remote]**
Fetch changes from the remote, but not update tracking branches.

**$ git pull [remote]**
Fetch changes from the remote and merge current branch with its upstream.

**$ git push [--tags] [remote]**
Push local changes to the remote.
Use --tags to push tags.

**$ git push -u [remote] [branch]**
Push local branch to remote repository. Set its copy as an upstream.

**git rebase [branch]**
apply any commits of current branch ahead of specified one.