Cheatography

# GIT - Everyday use Cheat Sheet

by TME520 (TME520) via cheatography.com/20978/cs/22547/

## Key concepts

| | |
|---|---|
| **HEAD** | Head is your current branch. You can see what HEAD points to by typing `cat .git/HEAD`, |
| **Remote** | Remotes are non-local repositories you can interact with (push/pull). The default remote is *origin* (you can see that using `git remote -v`). |
| **Branch** | Branches are a way of safely work on new features without messing other peoples work (one feature, one branch). |
| **Commit** | A commit is a change or a set of changes you wish to register and save. |

## Repositories

| | |
|---|---|
| **Create a local repository** | mkdir ./my_repo && cd ./my_repo && git init |
| **Clone a repository from GitHub** | git clone https://github.com/TME520/etm.git |
| **Clone a specific branch from a repo** | git clone -b development https://github.com/TME520/etm.git |

## Branches

| | | |
|---|---|---|
| **List branches** | git branch | git branch --list |
| **Clone a specific branch** | git clone -b <branch> <remote_repo> | git clone -b development git@github.com:user/myproject.git |
| **Switch to an existing branch** | git checkout <branch> | git checkout feat/robert/add-pub-holidays-2020 |

## Branches (cont)

| | | |
|---|---|---|
| **Switch to a new branch** | git checkout -b <new_branch> | git checkout -b conf/marcel/use-EditorConfig |
| **Push changes to a remote branch** | git push origin <branch> | git push origin fix/patrick/remove-parasite-chars-from-config-file |
| **Delete a branch** | git branch -d <branch> | git branch -d feat/raja/count-api-failures |
| **Integrate your feature branch to the main** | git merge <branch> | git merge main |

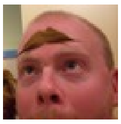
Using branches, several developers are able to work together on the same code base, the same project.
**git merge** is not usually done manually, but is managed by your pull request system.

## Getting out of (mild) troubles

| | |
|---|---|
| **Cancel untracked uncommited local changes** | git reset --hard && git pull remote <remote_branch> |

## Basic configuration

| | |
|---|---|
| **Set your name** | git config --global user.name "John Doe" |
| **Set your email address** | git config --global user.email johndoe@example.com |
| **Set your default editor** | git config --global core.editor emacs |



## Dot files

| | |
|---|---|
| *.gitignore* | Specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected. |
| *.gitattributes* | Gives attributes (end of line type, diff type, merge type...) to certain files. |

Examples of *.gitignore* files: click here
Examples of *.gitattributes* files: click here

## Basic workflow

| | |
|---|---|
| **Clone a repo** | git clone https://github.com/TME520/etm.git |
| **Create a new branch** | git checkout -b conf/yu/add-api-endpoints-to-monitoring |
| **Check repo status and current branch** | git status |
| **Add changes to next commit (track files)** | git add -A |
| **Commit with a message** | git commit -m "Sometimes dogs are grey" |
| **Refresh current local branch with remote branch** | git pull origin development |
| **Push changes to remote** | git push origin conf/yu/add-api-endpoints-to-monitoring |

**git push** uploads your commits to the remote repository.
**git pull** is a combination of git fetch and git merge. It gets the updates from remote repository and applies the latest changes to your local.

## Commits

| | | |
|---|---|---|
| **Add one file to a future commit** | git add <file> | git add benchmark.c |
| **Add all files to a future commit** | git add -A | |
| **Commit with a message** | git commit -m "<message>" | git commit -m "Initial commit" |
| **Change the message of the latest commit** | git commit --amend -m "<new_message>" | git commit --amend -m "Beautiful commit" |
| **Cancel a commit** | git log --oneline | |
| | git revert <commit_id> | |

Git commits are checkpoints in the development process which you can go back to later if needed.
**Git commit saves your changes only locally.**
**git revert** won't delete the commit, it will instead create a new one cancelling the other.