Git Cheat Sheet

Create & Clone

What	How
create new repository	git init
clone local repository	git clone /path_repository
clone remote repository on computer	git clone username@host:/path_repository

Add & remove

What	How
add changes to INDEX	git add *filename*
add all changes to INDEX	git add *
remove/delete	git rm *filename*

Commit & synchronize

What	How
commit changes	git commit -m “Commit message”
push changes to remote repository	git push origin master
connect local repository	git remote add origin *server*
update local repository with remote changes	git pull
know who made a change and create the file	git blame *filename*

Branches

What	How
see which branches exist and we are on	git branch
create new branch	git checkout -b *branch*
switch to master branch	git checkout master
delete branch	git branch -d *branch*
push branch to remote repository	git push origin *branch*

Merge

What	How
merge changes from another branch	git merge *branch*
view changes between two branches	git diff *sourcebranch targetbranch*

Conflicts happen when changes are made on a file on exactly the same lines. We need to open the document, where we can see which line come from which branch and then delete those elements and keep what we want.

Tagging

What	How
create tag	git tag *tag commitID*
get commit IDs	git log

Restore

What	How
replace working copy with latest from HEAD	git checkout –*filename*

Ignore file

It's really important to ignore files in Git sometimes, for example: Configuration files (config.xml, databases.yml, .env...), Temporary files (tmp, temp/...), Less-than-useful files (.DS_Store, .project...). The most important thing to know is that you should NEVER commit sensitive information like access keys, passwords, and so forth. These items should especially never be put on GitHub.

What	How
create a special file	touch .gitignore

Open this .gitignore file and write in there the *filename* that we want to ignore.

Example

What	How
create a folder	mkdir *MyProject*
change directory	cd *MyProject*
create a new repository (only once per folder)	git init
look at the status of folder	git status
create a file	touch *Document.txt*
modify the file	*Document.txt*
file in red (not tracked)	git status
add changes	git add *Document.txt*
file in green (ready for commit)	git status
commit changes	git commit -m "Add a file"
nothing to commit	git status
modify *Document.txt* (red "modified")	git status
add specific lines modified	git add -p
see whole history with commit code	git log
go back to older version	git checkout *CommitCode*

On GitHub, the button *Pull Request* allows to send a message to the owner of a code/file that we cloned and made some modifications. The owner will decide to merge our branch to his master branch.