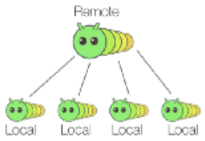
Git Cheat Sheet

Good things to know



Git is a **distributed** version control system

Everyone that has a local copy of the repository at all times. Cannot be partially checked out, it's all or nothing.

Git ≠ GitHub

Git **is not** Github

Git is a repository, where files are stored and managed. Github is a hosting service that keeps your Git repo in the cloud.

```
> git
```

Git CLI is **very powerful**.

OK, that's not a "fact", but once you get used to it, you'll be able to use the commands anywhere. You can also use CLI and GUIs interchangeably on the same Git repository.

There are **many ways** to get there

People use Git in many different ways, and projects may follow different Git flows. Don't be shy, ask around.

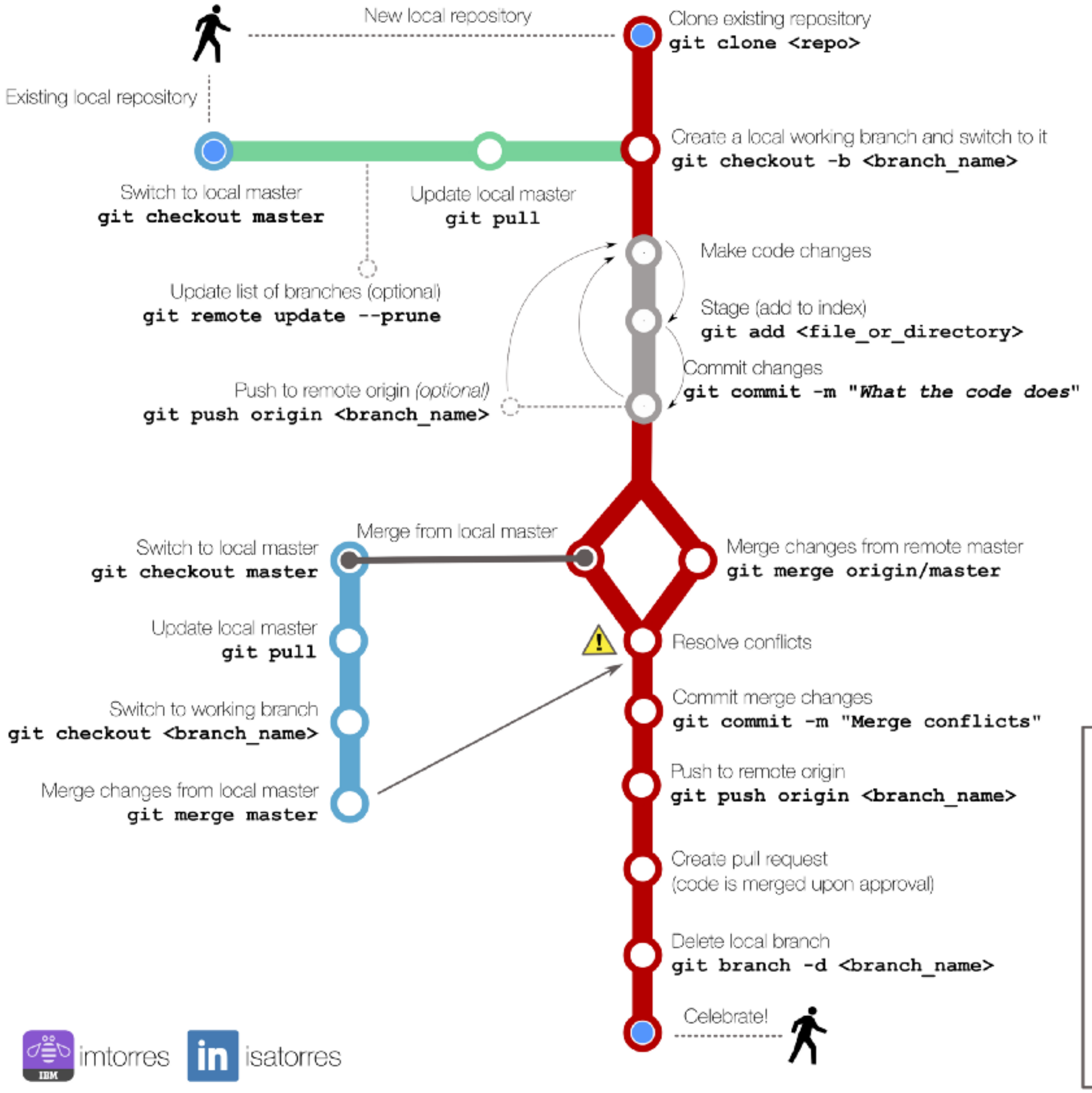
Terminology

repository - *where files are stored, can be remote or local*
remote repository - repository in hosting server, also referred to as origin
local repository - repository in local development machine
branch - *a stream of work where commits are kept, can be remote or local*
remote branch - branch with published commits (commits that have been pushed)
local branch - branch with unpublished commits, only developer can see, not shared
commit - *a change unit, it is a scope of changes that are kept in sequential order*

master branch - *main stream of work, must always be stable*
working branch - *developer's stream of work, sandbox*
staging area/index - *keeps files to include in next commit*
push - *publish changes to remote*
pull - *merge remote changes into local changes*
pull request - *code review process to allow a change be integrated to the master branch*

Simple flow of getting things done with Git

Scenario: developer working in individual branch, team code (stable) lives in the master branch



New local repository

Existing local repository

Clone existing repository
`git clone <repo>`

Create a local working branch and switch to it
`git checkout -b <branch_name>`

Switch to local master
`git checkout master`

Update local master
`git pull`

Update list of branches (optional)
`git remote update --prune`

Make code changes

Stage (add to index)
`git add <file_or_directory>`

Commit changes
`git commit -m "What the code does"`

Push to remote origin *(optional)*
`git push origin <branch_name>`

Merge from local master

Switch to local master
`git checkout master`

Update local master
`git pull`

Switch to working branch
`git checkout <branch_name>`

Merge changes from local master
`git merge master`

Merge changes from remote master
`git merge origin/master`

Resolve conflicts

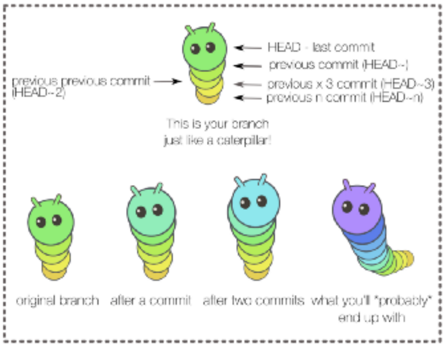
Commit merge changes
`git commit -m "Merge conflicts"`

Push to remote origin
`git push origin <branch_name>`

Create pull request
(code is merged upon approval)

Delete local branch
`git branch -d <branch_name>`

Celebrate!


HEAD - last commit
previous commit (HEAD~)
previous previous commit (HEAD~2)
previous x 3 commit (HEAD~3)
previous n commit (HEAD~n)
This is your branch just like a caterpillar!
original branch
after a commit
after two commits
what you'll "probably" end up with

FREE!

Commands that you can use anytime

See the status of the working branch
`git status`

See the commit history
`git log`

See all branches (remote - since last remote update)
`git branch -a`

⚠ Resolve conflicts

No need to panic! Use a merge tool, or...

```
(remove)  <<<<<<< HEAD
(merge)    your local code
(remove)  =======
(merge)    code in master
(remove)  >>>>>>> master
```

Mark the conflict resolved
`git add <file>`

Want to start over? Roll back all the merged changes
`git reset --hard HEAD`

imtorres isatorres