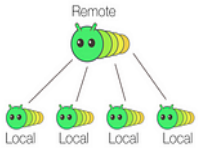
Git Cheat Sheet

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 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

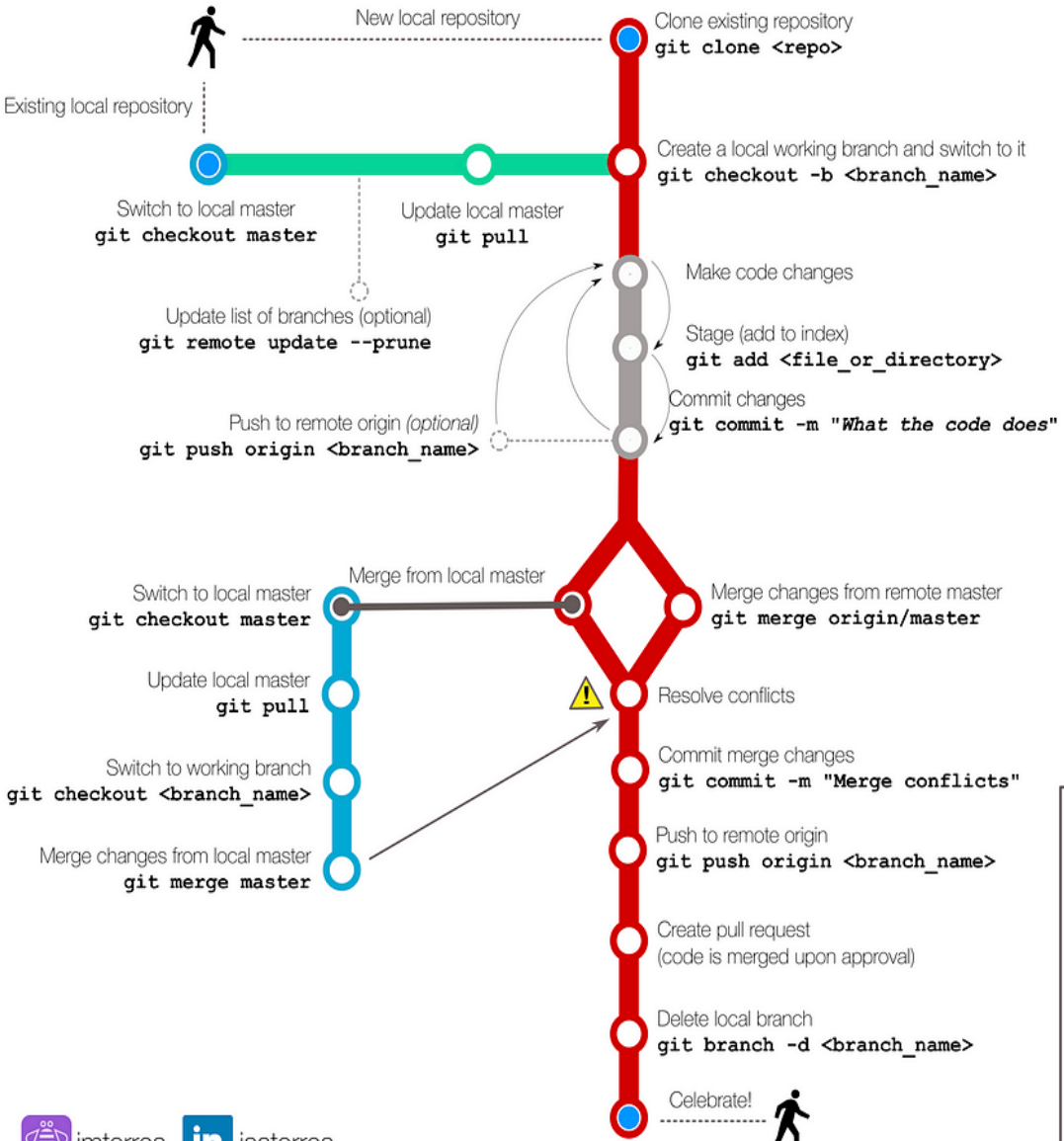
branch - *a stream of work where commits are kept, can be remote or local*
- remote branch - branch with published commits (commits that have been pushed)
- local branch - branch with unpublished commits, only developer can see, not shared

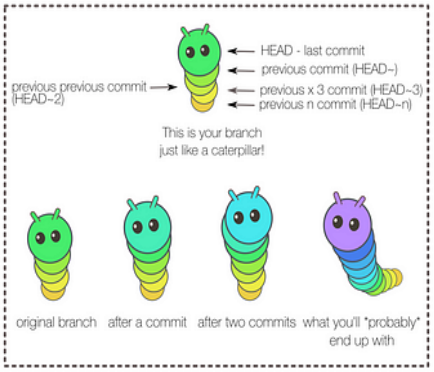
commit - *a change unit, it is a scope of changes that are kept in sequential order*

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





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

Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.