git

By: Megan Avery
Updated July 2018

Before we start

- Make sure you have git on your computer
  - You can check by doing **`git --version`** in your terminal
- Create an account on github.com (or have another git account you want to use) and log in to that account
- Run **git config --global github.user <github user name>**
- Run **git config --global user.name <your name in quotes>**
- Run **git config --global user.email <email in quotes>**

Note: all git configurations can be found in ~/.gitconfig

What is git?

A version control system that lets you keep track of file histories and backup work online.

Vocab Alert!

repo: short for repository, basically a project folder within git

local repo: the local version of a repo, specific to your computer

remote repo: the online version of a repo, shared between all contributors

README.md: file that contains a description of a repo

clone: to put a copy of a remote repo onto your machine

Creating a New Repo

Clone Existing Repo to Computer

Command: `git clone <.git git repo URL>`

Can copy the URL directly from the repos home page on github.com

Vocab Alert!

commit: apply changes you've made on computer to your local repo

commit hash: unique letter/number combination associated with each commit

add: mark file/files as ready to commit, aka move to stage

push: apply changes you've made to local repo to the remote repo

pull: apply changes from the remote repo to your computer's files

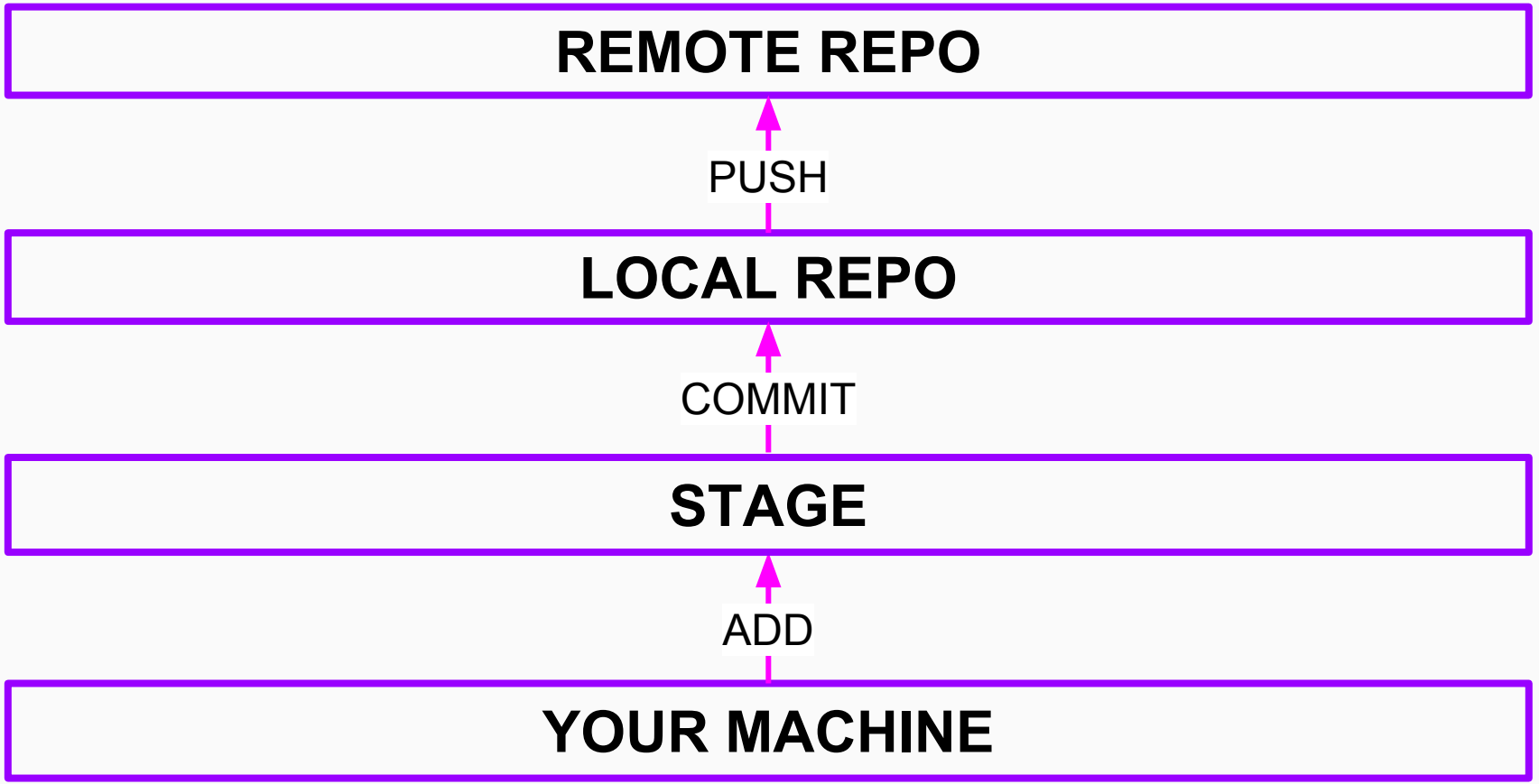
.git

Folder that is present in every git repo that allows it to actually be a repo. If you were to delete this folder it would turn your repo into just another folder on your computer.

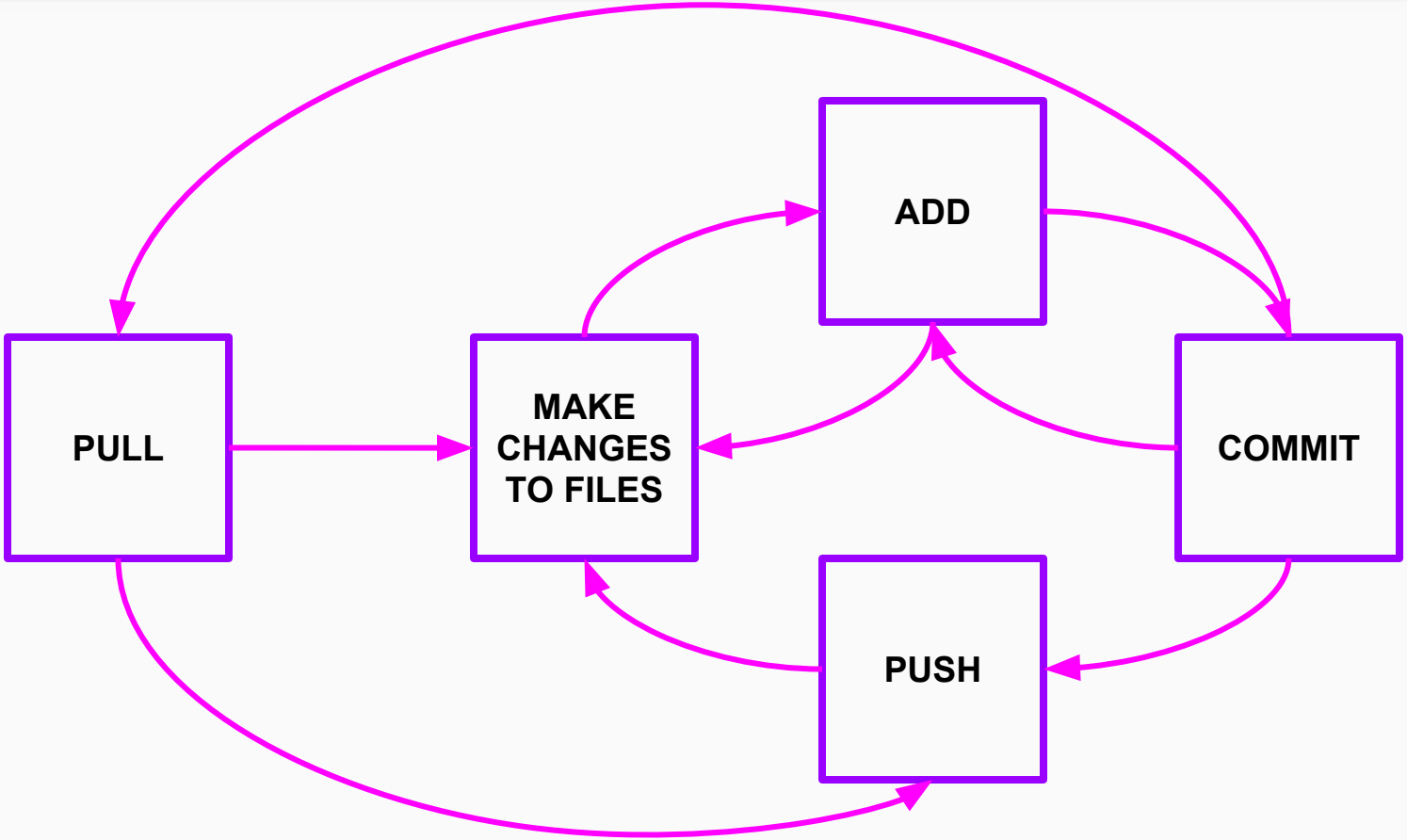
.gitignore

A file in very git repo that configures some files that should not be committed to the git repo. Good for things like notes on the repo you only want to keep locally or files associated with an IDE that are unneeded in the repo. Can involve wildcards (*).

Different Levels of a Github Repo



Basic Workflow



Walkthrough of Workflow

Basic Commands 1

git status - list which files are staged, unstaged, and untracked

git diff - see what changed since last add

- Get out of this view with :q
- Can also do git diff <file> to see differences for single file

git add <file> - stage a file for the next commit

git add <directory> - stage a directory and all files in it for next commit

git add -p - start interactive session that lets you choose portions to add

Basic Commands 2

git commit - commit staged files, opens vim to enter message

git commit -m “<message>” - commit staged files with commit message

git push - send changed to remote git repo

git pull - get latest change from remote repo onto machine

Branches

A branch is an offshoot of a particular repo that is used for making changes without messing up the base files. The default branch for all repos is called “master” and any other branches you create can be named whatever you want.

Branch Commands

git checkout -b <name> - create a new branch from the current branch

git checkout -b <name> <existing branch> - create a new branch off of a given existing branch

git checkout <name> - got to a branch that already exists

git diff <branch> - see differences between current branch and given branch

git push -u origin <branch> - push to the new branch for the first time to start tracking

Merging Branches

Whenever you can to put changes from one branch onto another it’s called merging.

Command: **git merge <branch>** - merge given branch into current one

Most often want to merge your branches back into master to get all the file changes back into one place or merge master into your offshoot branch to get the common changes back into your work

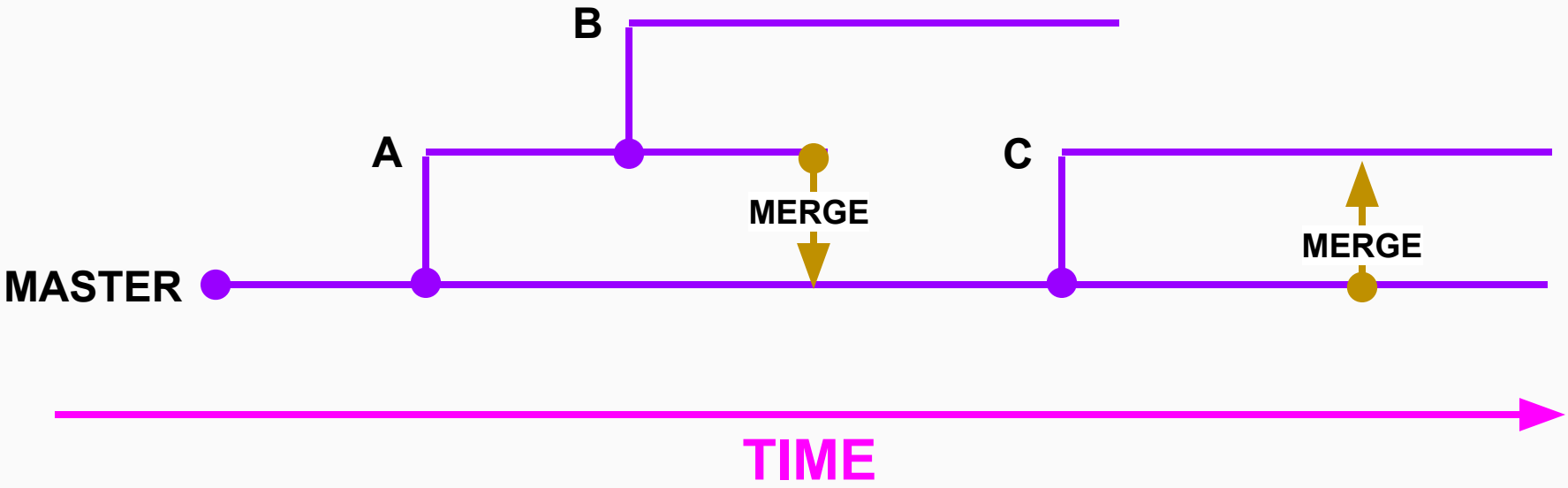
Dealing with Merge Conflicts

merge conflict - when a merge cannot be fully completely because parts of files interfere with each other

If a merge conflict occurs you'll get a message on the command line and a list of files that have conflicts.

You can fix the issues, add, commit, and push manually or use a tool like mergetool to resolve the conflicts

Visual Representation of Branches



Advanced Commands

git stash - save off all change and reset files

git stash pop - re-apply changes that had been saved with stash previously

git checkout <filename> - undo changes since last add

git checkout <commit hash> - checkout a particular commit

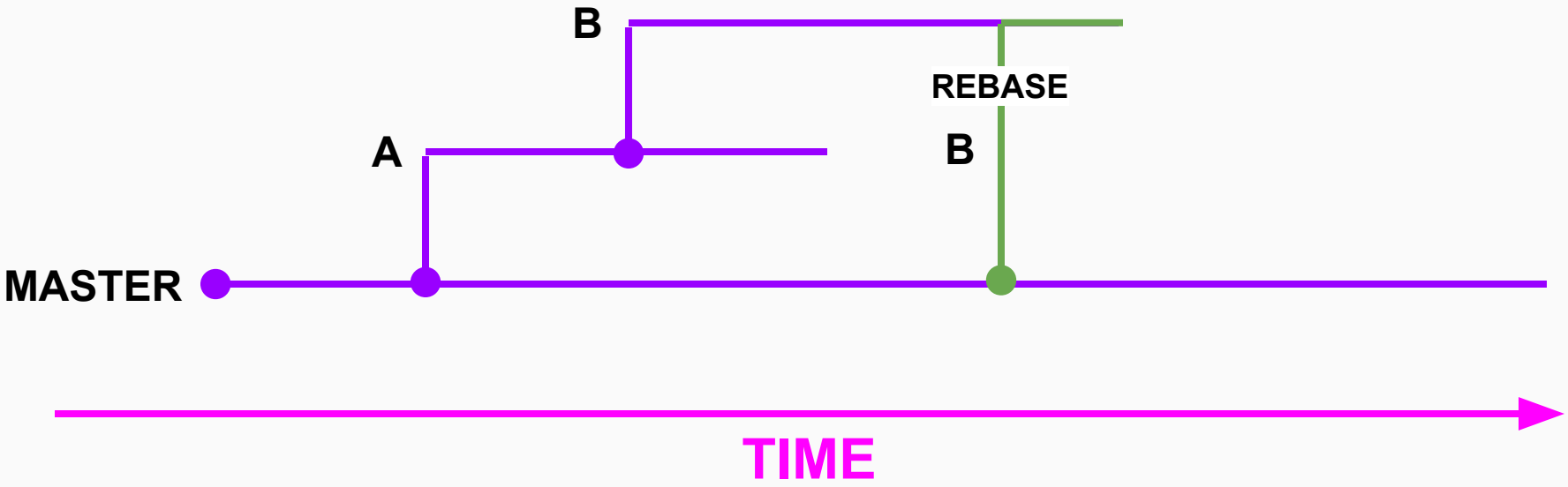
git reset <filename> - unstage a file

git reset HEAD^ - undo a commit and unstage files

Rebasing a Branch

Rebasing changes the branch that the current branch is based off of. It changes the structure of the branches within a repo.

Visual Representation of a Rebase



Forking

A fork is a copy of a repo, it creates a totally new master to work off of. This allows you total independence from the original project with no worries that you are going to mess it up.

Visual Representation of a Fork

Helpful Links

https://github.com/ - github website

https://try.github.io - interactive tutorial for git

https://git-scm.com/downloads - git GUI download

The End