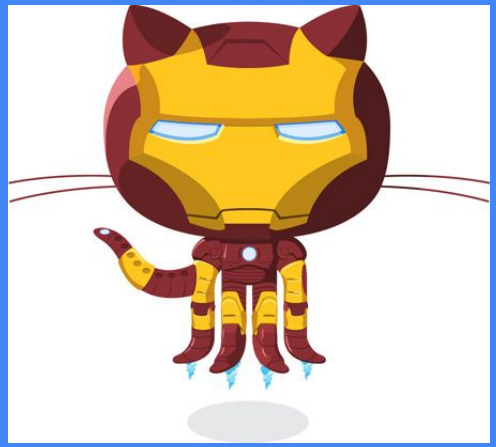
GITHUB - Basics & Version Control



Github New Project Commands:

A. Go to your project folder in the terminal

B. Create a repo in your git hub account

1. *git init*

2. *git add .*

3. *git commit -am "first commit"*

4. *git remote add origin https://github.com/gitaccountnamhere/reponamehere.git*

5. *git push -u origin master*

6:49 PM

Git Pushing New Updates Commands*:*

1. *git add .*

2. *git commit -am"your message here"*

3. *git push*

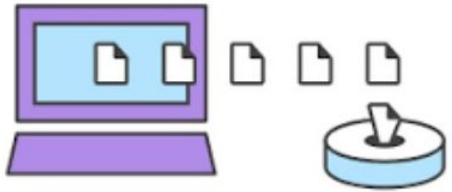
What is Version Control:

But what is versioning? The name says it all: it's a way of keeping versions of files stored in a repository. In our case these files happen to be text representing the source code of a program, nonetheless a VCS (Version Control System) can be used for all sorts of files. That is, imagine that a set of files and directory represents a state in time. Without a VCS you end up overwriting those files and, therefore, rewriting the state. With a VCS, you keep the states as a stack. At any point in time you can retrieve an older state or add a new one on top of the stack.

https://blog.hipolabs.com/how-to-work-in-a-team-version-control-and-git-923dfec2ac3b

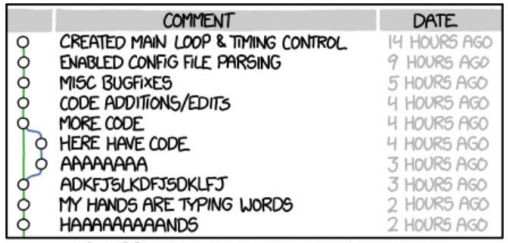
git add:

The **git add** command adds a change in the working directory to the staging area. It tells **Git** that you want to include updates to a particular file in the next commit. However, **git add** doesn't really affect the repository in any significant way—changes are not actually recorded until you run **git** commit .



git commit:

git commit. The "**commit**" command is used to save your changes to the local repository. ... Using the "**git commit**" command only saves a new **commit** object in the local **Git** repository. Exchanging **commits** has to be performed manually and explicitly (with the "**git** fetch", "**git** pull", and "**git** push" commands).

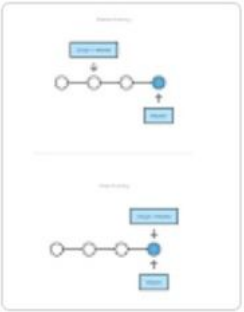


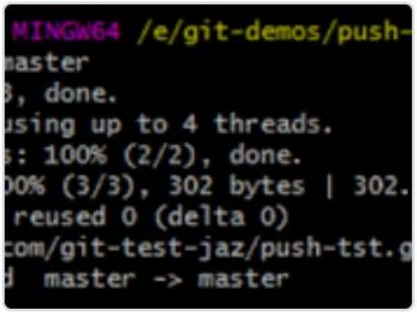
Examples of *good* commits:

- "Scaffolding for project XYZ"
- "Wrote module for communicating with API ABC"
- "Refactored module for API ABC"

https://blog.hipolabs.com/how-to-work-in-a-team-version-control-and-git-923dfec2ac3b

git push:

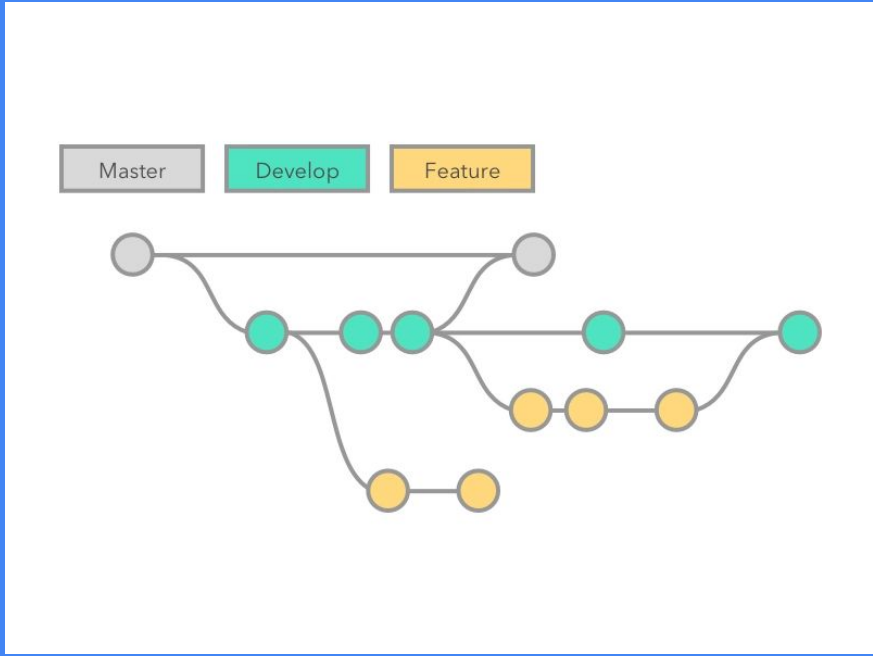


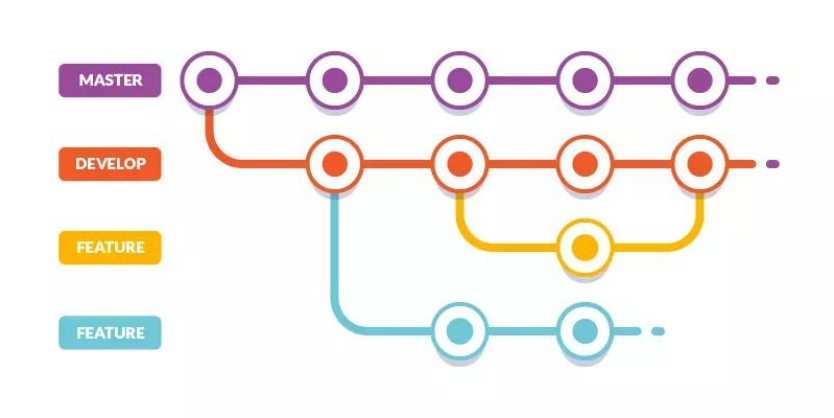


The **git push** command is used to upload local repository content to a remote repository. **Pushing** is how you transfer commits from your local repository to a remote repo. It's the counterpart to **git** fetch , but whereas fetching imports commits to local branches, **pushing** exports commits to remote branches.

git branch:

A **branch** represents an independent line of development. ... The **git branch** command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, **git branch** is tightly integrated with the **git** checkout and **git** merge commands.

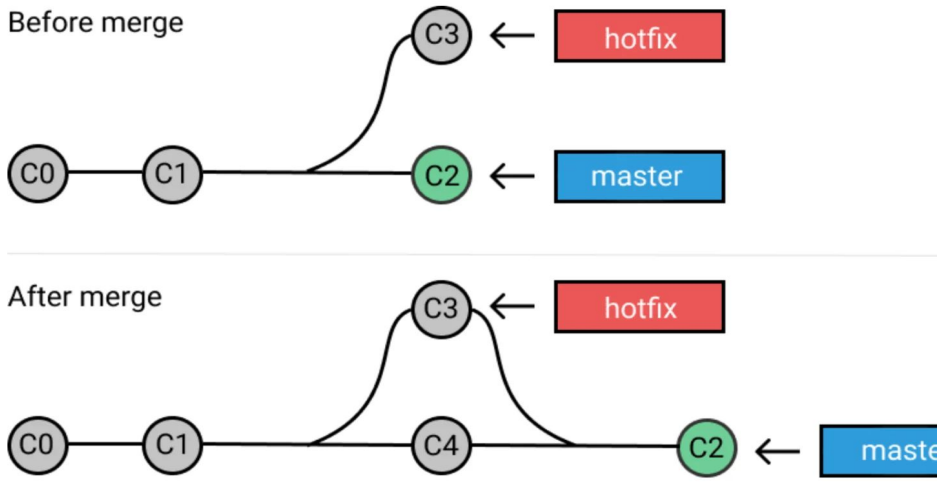




git merge:

Merging

Merging, as the word says, simply “merges” the commits of the two branches into the new one. To understand what happens let’s imagine we have the following situation:



git checkout:

The **git checkout** command lets you navigate between the branches created by **git** branch . Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells **Git** to record all new commits on that branch.