MAKE IT REAL CAMP

TOP - Preparation

Shymmy W. Garcia

Contents

The Terminal

The terminal

What is Terminal?

Terminal is an application that gives us a command line interface (or CLI) to interact with the computer.
Everything you can do in Finder you can do in Terminal.

Developers use Terminal because, more often than not, it is much faster to use Terminal than a graphical user interface (GUI) such as Finder.

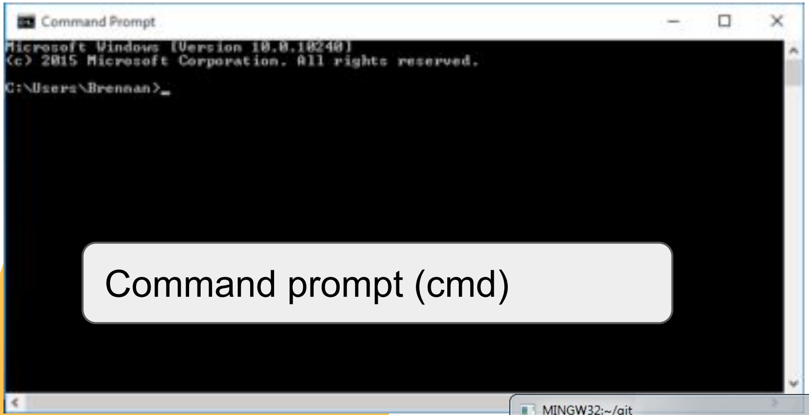
The terminal

What is a shell? Bash/ZSH

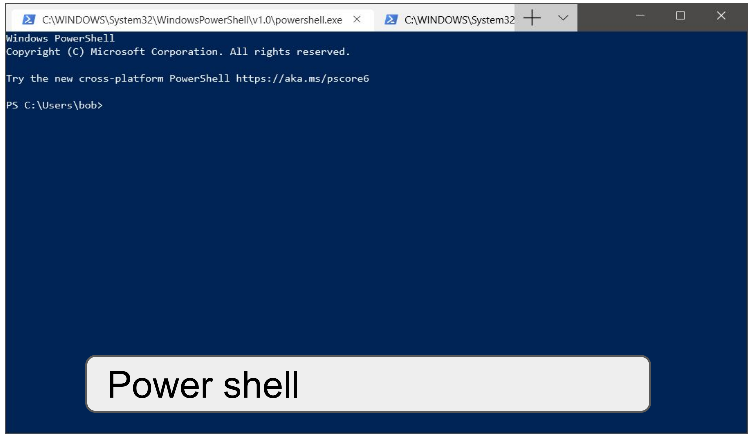
The **shell** is the program which actually processes commands and returns output. Most shells also manage foreground and background processes, command history and command line editing. These features (and many more) are standard in bash, the most common shell in modern linux systems. (We are using zsh).

A **terminal** refers to a wrapper program which runs a shell. Decades ago, this was a physical device consisting of little more than a monitor and keyboard. As unix/linux systems added better multiprocessing and windowing systems, this terminal concept was abstracted into software.

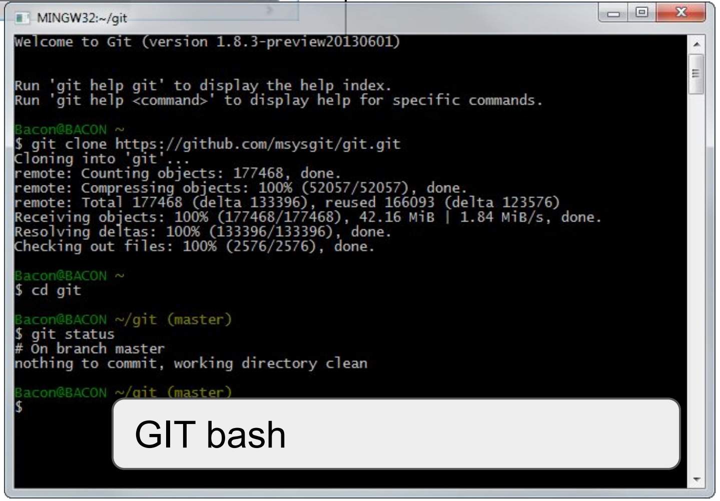
Windows



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Brennan>_
```

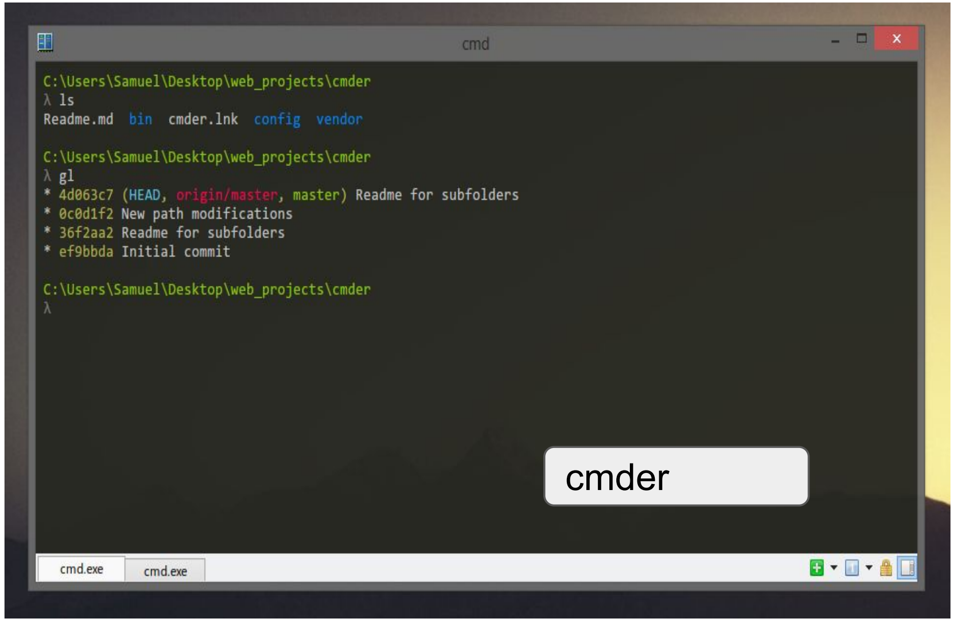


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\bob>
```



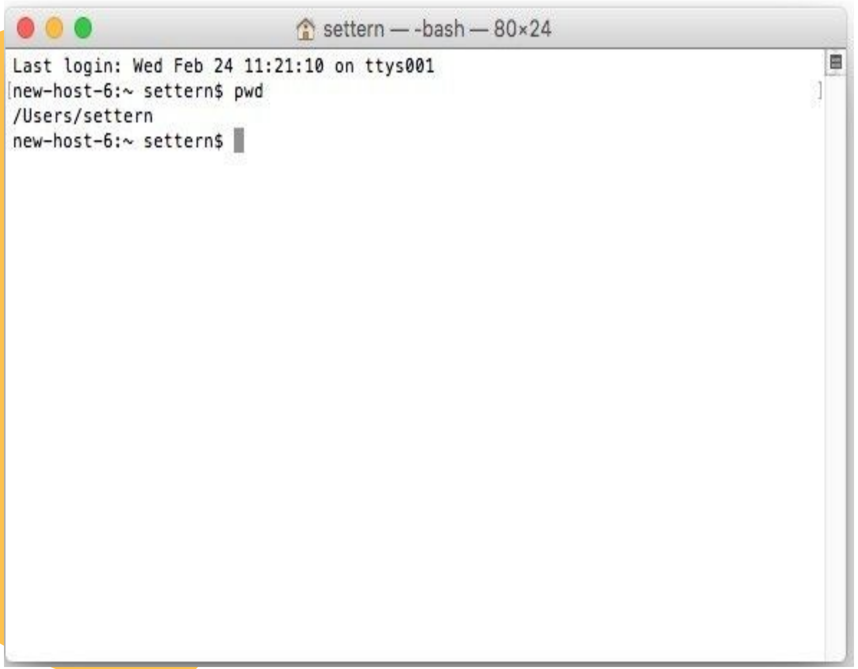
```
Welcome to Git (version 1.8.3-preview20130601)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Bacon@BACON ~
$ git clone https://github.com/msysgit/git.git
Cloning into 'git'...
remote: Counting objects: 177468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 177468 (delta 133396), reused 166093 (delta 123576)
Receiving objects: 100% (177468/177468), 42.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (133396/133396), done.
Checking out files: 100% (2576/2576), done.

Bacon@BACON ~
$ cd git

Bacon@BACON ~/git (master)
$ git status
# On branch master
nothing to commit, working directory clean

Bacon@BACON ~/git (master)
$
```

Windows



```
C:\Users\Samuel\Desktop\web_projects\cmder
λ ls
Readme.md  bin  cmder.lnk  config  vendor

C:\Users\Samuel\Desktop\web_projects\cmder
λ gl
* 4d063c7 (HEAD, origin/master, master) Readme for subfolders
* 0c0d1f2 New path modifications
* 36f2aa2 Readme for subfolders
* ef9bbda Initial commit

C:\Users\Samuel\Desktop\web_projects\cmder
λ
```

MAC



```
Last login: Wed Feb 24 11:21:10 on ttys001
new-host-6:~ settern$ pwd
/Users/settern
new-host-6:~ settern$
```

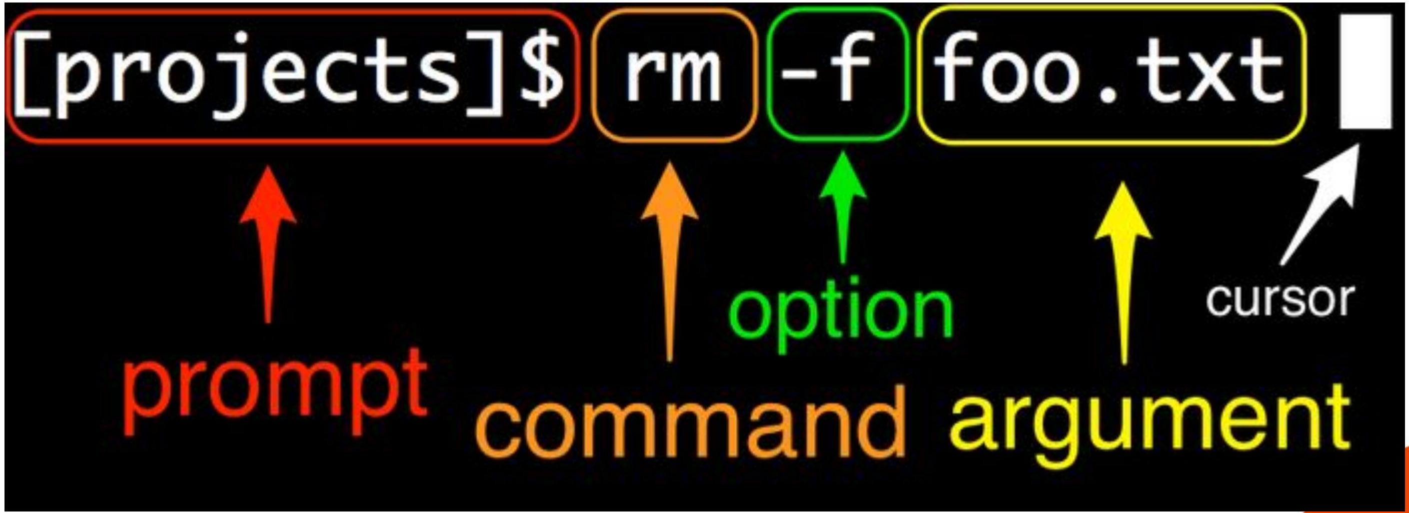
Bash

Zsh → Z-shell

The terminal

The typical elements of a command, are:

- The prompt (to “prompt” the user to do something)
- a command (as in “give the computer a command”),
- an option (as in “choose a different option”),3 and
- an argument (as in the “argument of a function” in mathematics).



```
[projects]$ rm -f foo.txt
```

Mac: Terminal (z-Shell) Basic Commands

Basic commands

echo

pwd
(print working directory)

ls
(list items)

Navigation

cd (..)
(change directory)

cd /
(root directory)

cd /Users
(users directory)

cd or cd ~
(home directory)

Files / folders

touch
(create/ "touch" file)

mkdir
(create directory)

rm
(delete file)

rmdirr
(delete empty folder)

cp
(copy file/folder)

MAKE IT REAL CAMP

Let’s practice commands

MAKE IT REAL CAMP

GIT

git

What is GIT?

Git is an Open Source Distributed (Decentralized) Version Control System (VCS) designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams.

MAKE IT REAL CAMP

Version Control System (VCS)

What is Version Control?

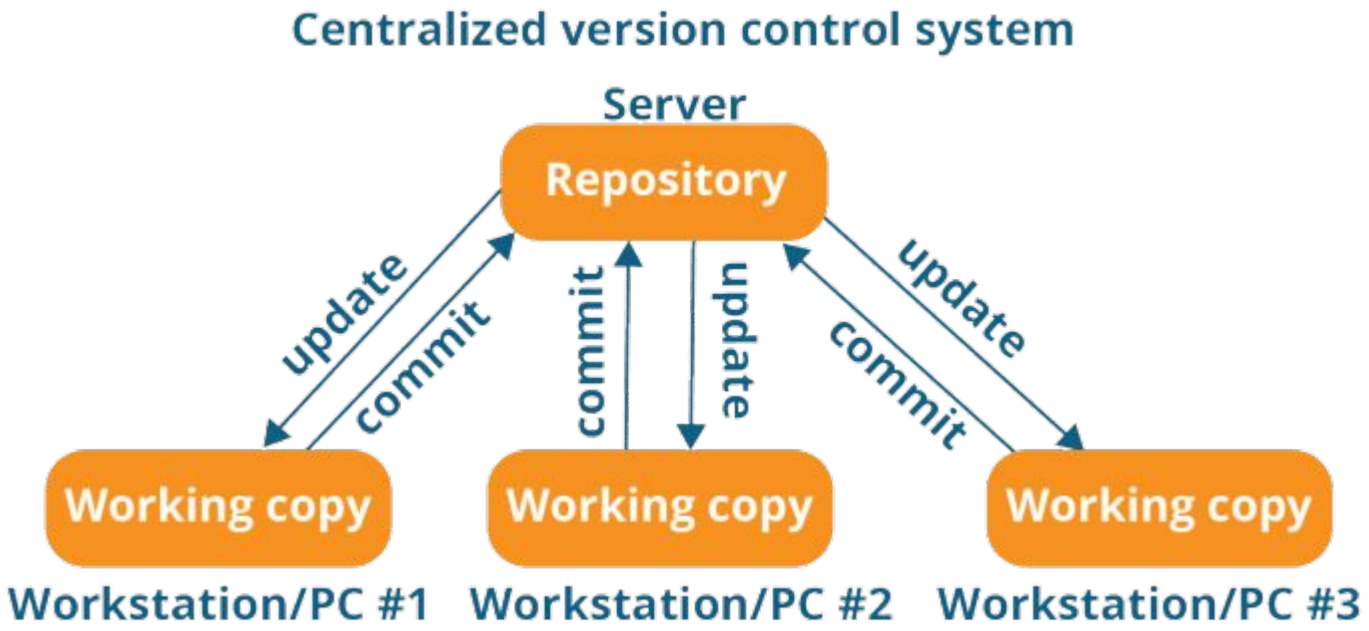
Version control, also known as source control, is the practice of tracking and managing changes to software code.

What is a Version Control System?

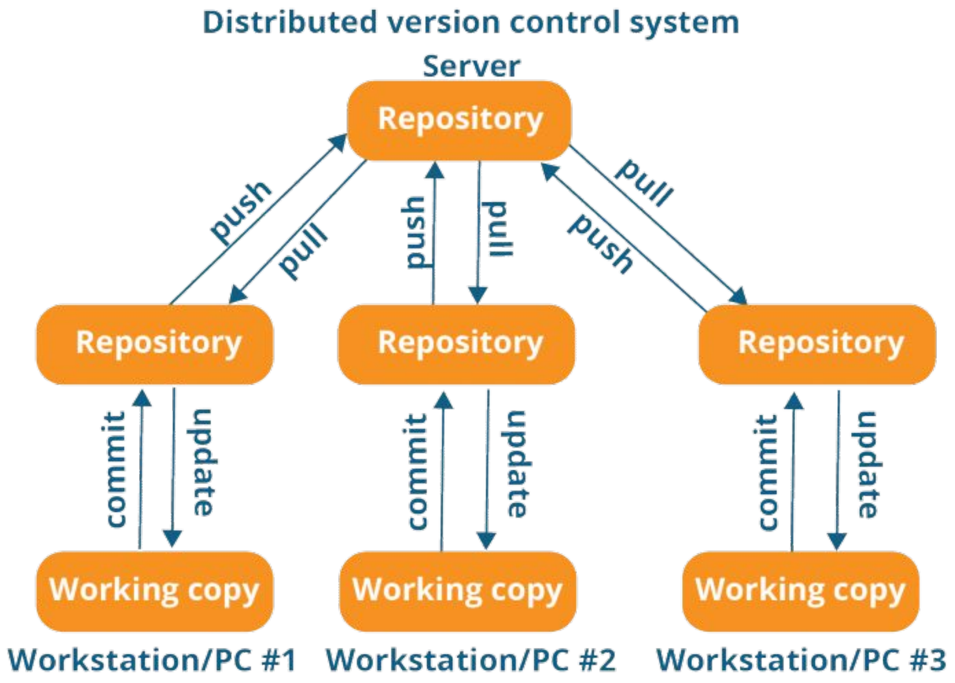
Version control systems are software tools that help software teams manage changes to source code over time.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

What is Distributed?



What is Distributed?



Be Aware!

GIT IS NOT EQUAL TO GITHUB!

GIT: Setup and basics

Installation and setup

- **Download and Install GIT:**
- **Verify GIT version:**
  git --version.
- **Configure GIT:**

  git config --global user.name <name> : Define the author name to be used for all commits by the current user.

  git config --global user.email <email>: Define the author email to be used for all commits by the current user.it.

GIT

Basic commands

git init

git add

git status

git commit

git diff

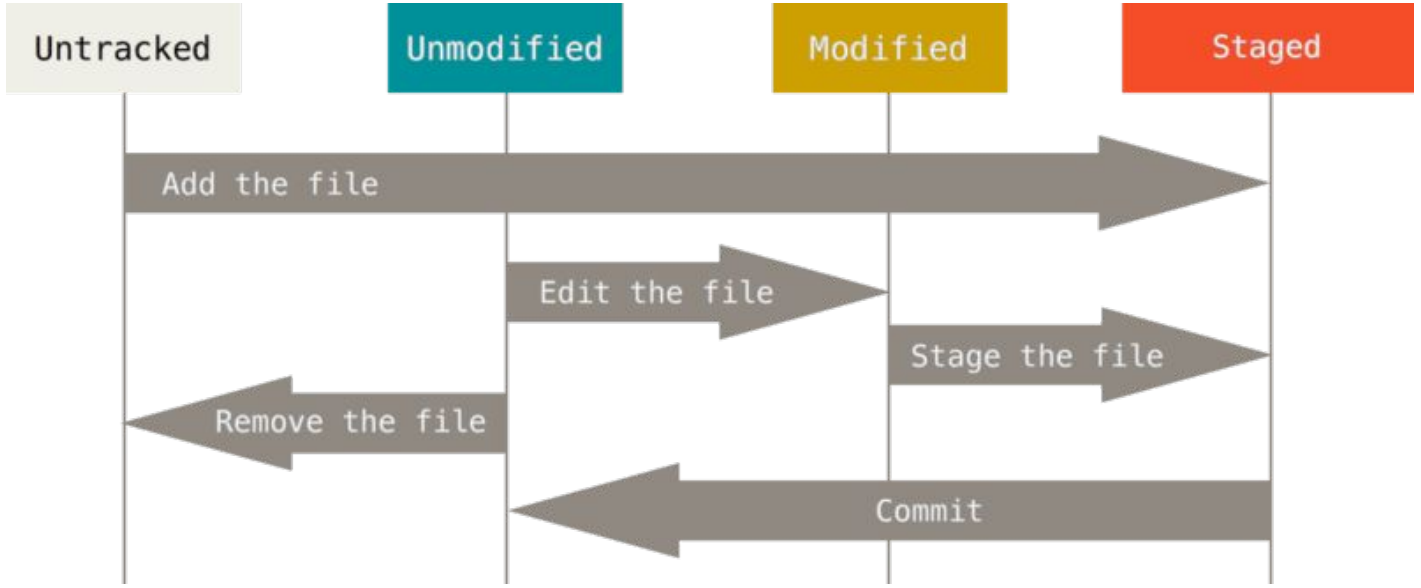
git rm --cached

git log

MAKE IT REAL CAMP

How Does Git Works?

The lifecycle of the status of your files



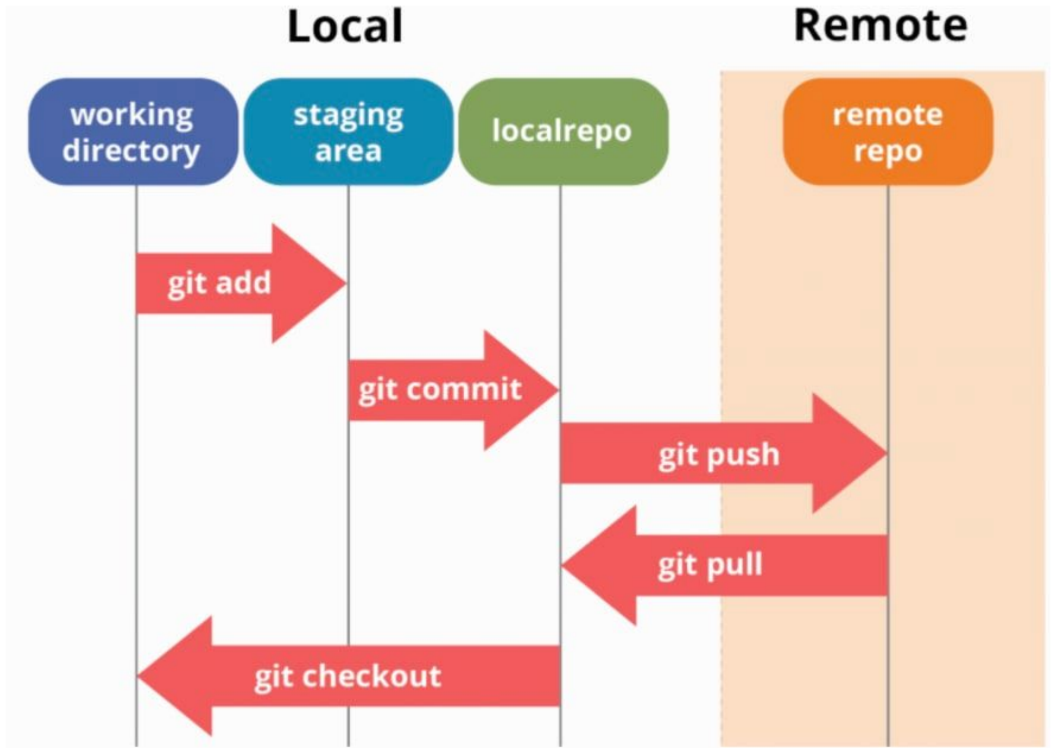
GIT File Status

Each file in your working directory can be in one of the following two states: **Tracked or Untracked**

File status in Life Cycle :

- Untracked
- Unmodified
- Modified
- Staged

Basic GIT WorkFlow



Basic GIT WorkFlow

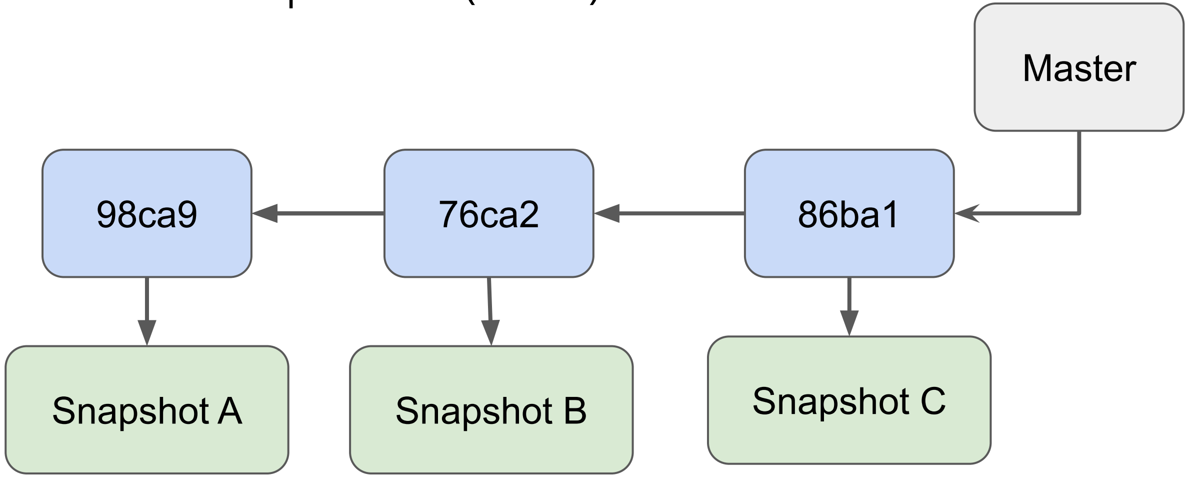
Git directory - Stores the metadata and object database for your project (while cloning the repository) .

Working directory - Single checkout of one version of the project. The files in the directory are pulled out of the compressed database in the Git directory and placed on disk for you to edit and use.

Staging area - It is a simple file, generally present in your Git directory, that stores information about what will go into your next commit.

What is a commit, anyway?

- In Git, a commit is a snapshot of your repo at a specific point in time.
- Keeps references to its parents
- Has a unique hash (SHA-1) over the content



MAKE IT REAL CAMP

Let's practice GIT commands

Let's do it

- Create a new repo
- Create a file
- Check the Status of the file
- Tracking New Files
- Staging Modified Files
- Short Status -s
- Commit
- Commit no Staging area -a -m
- Untrack Files

Let's practice GIT commands

GIT: Branches - Branching

Recap

- `Git -> software - tool`

Recap

- Git -> software-tool
- Repository -> folder Project/.git

```
$ git init
```

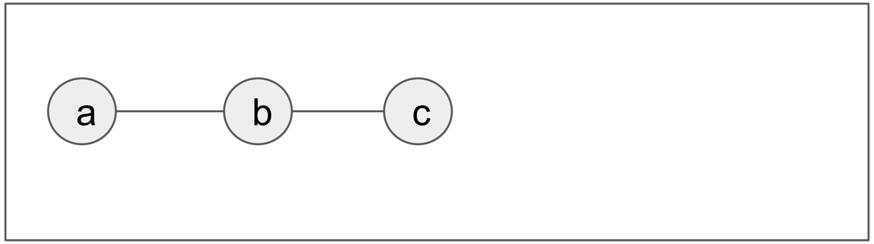
Recap

- Git -> software-tool
- Repository -> folder Project/.git
- Commit -> Project/.git/objects

  object (changes, author, ID ...)

Recap

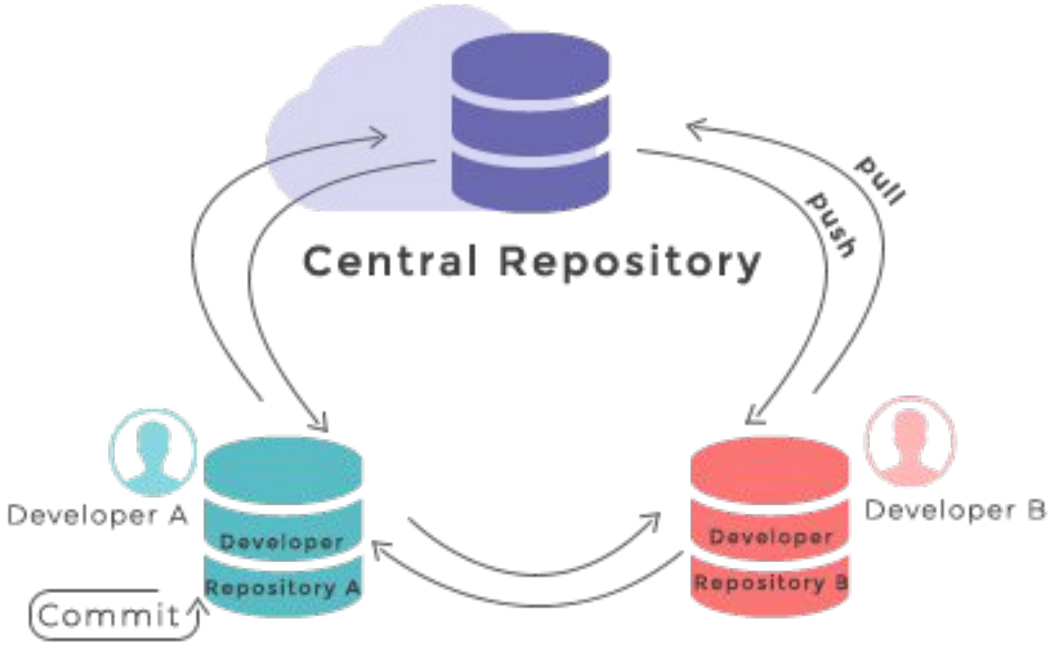
Project/.git



What if ...

Experiment

Collaborate

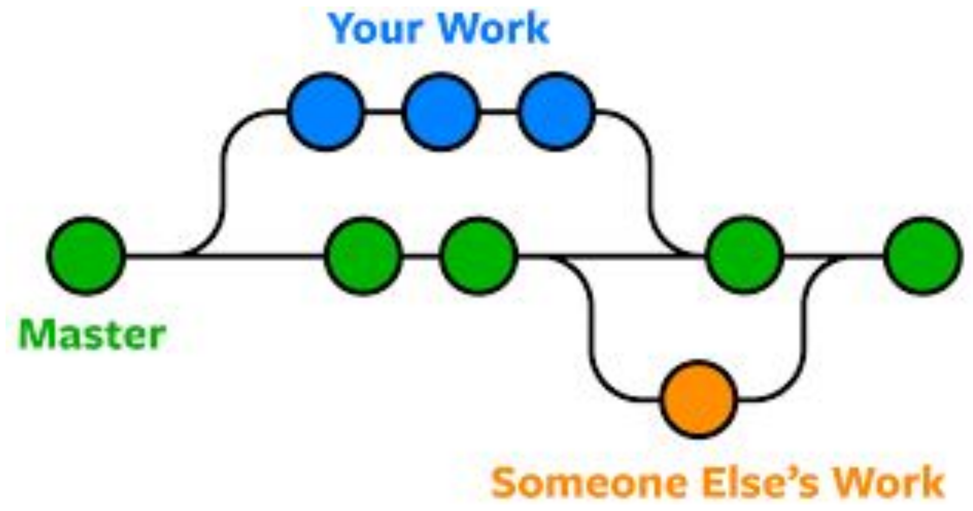


Central Repository
pull
push
Developer A
Developer Repository A
Developer B
Developer Repository B
Commit

How could you experiment/collaborate with your code safely

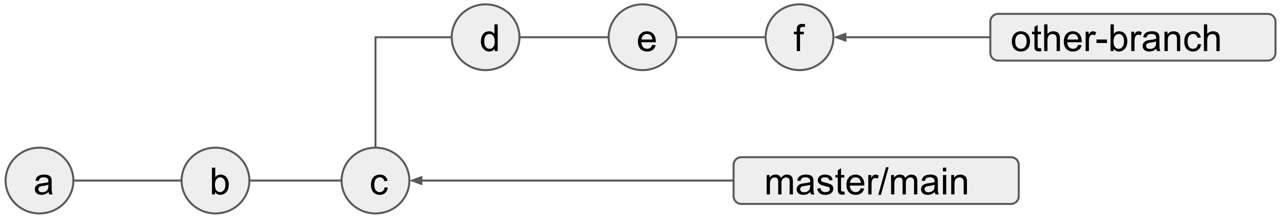
Branching

Branching means you diverge from the main line of development and continue to do work without messing with that main line.



What is a branch

Technically it is a reference to a commit.



Git branch Commands

```
$ git branch <your branch name> // creates
```

```
$ git checkout <your branch name>  // switches
```

```
$ git checkout  -b  <your branch name> // crates and switches
```

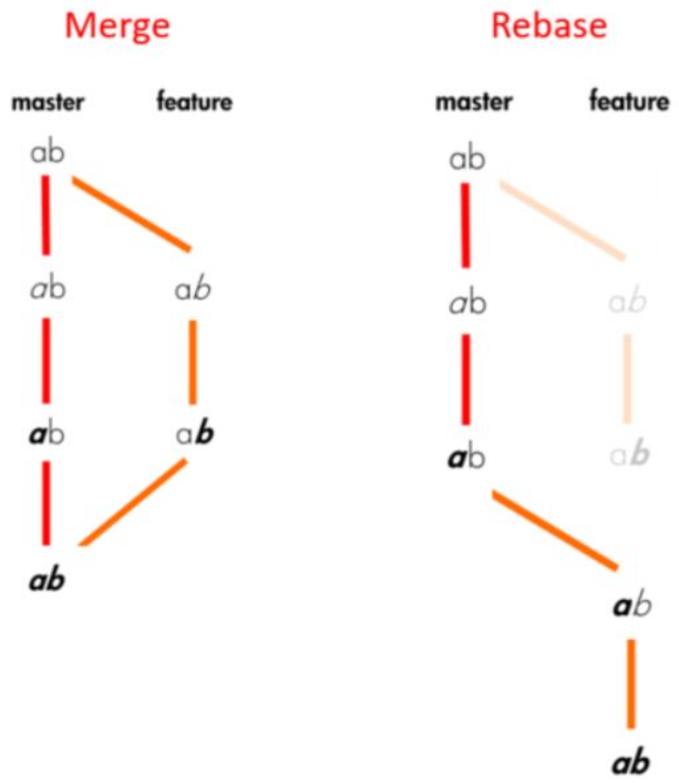
```
$ git branch // lists
```

```
$ git branch -d <your branch name>  // deletes
```

```
$ git branch -m <your branch name>  // renames
```

How could we integrate changes from one branch to another

Merging vs Rebasing



git merge:

```
git checkout master
git merge my-branch
```

Fastforward

When there are no new commits in master

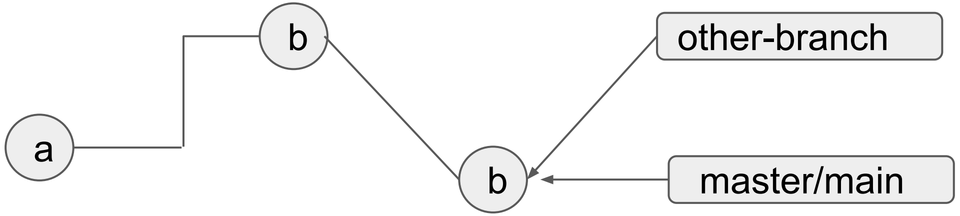
Merge commit

When there are also new commits in master as well as in ny-branch

Solving conflicts

When code was erase or modify in the same lines

git merge:

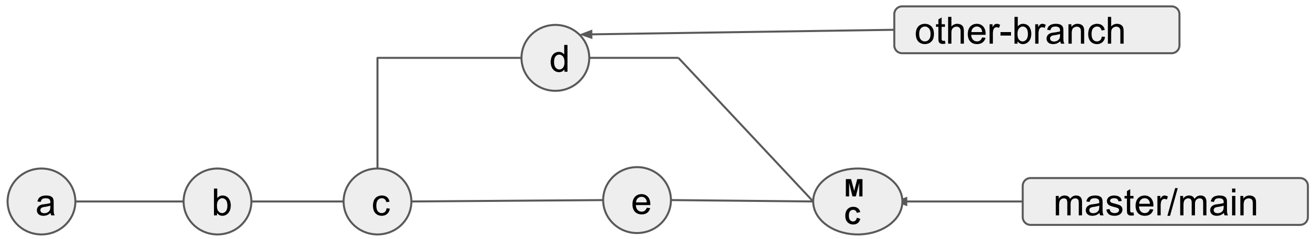


Create a new branch and one new commit from it.
Merge new branch into master

Fastforward

When there are no new commits in master

git merge:

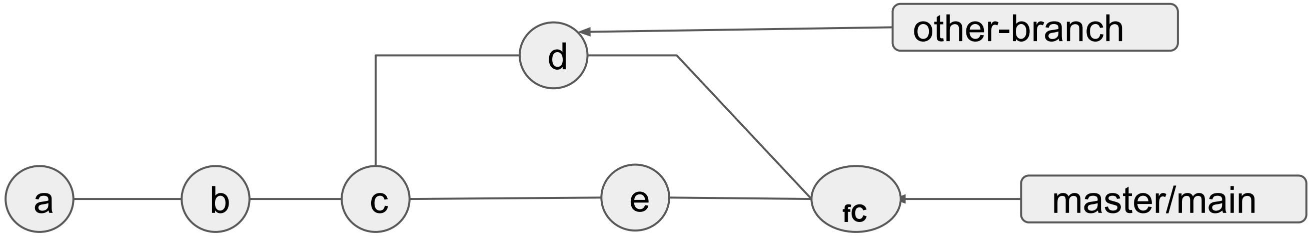


Merge commit

When there are also new commits in master as well as in ny-branch

Create a new file in branch 2
Update file1 in master
Merge branch-2 to master

git merge:



Update a file in branch 2 and commit
Update same file in the same lines in master and commit
Merge branch-2 into master
Solve conflicts
git add .
git commit

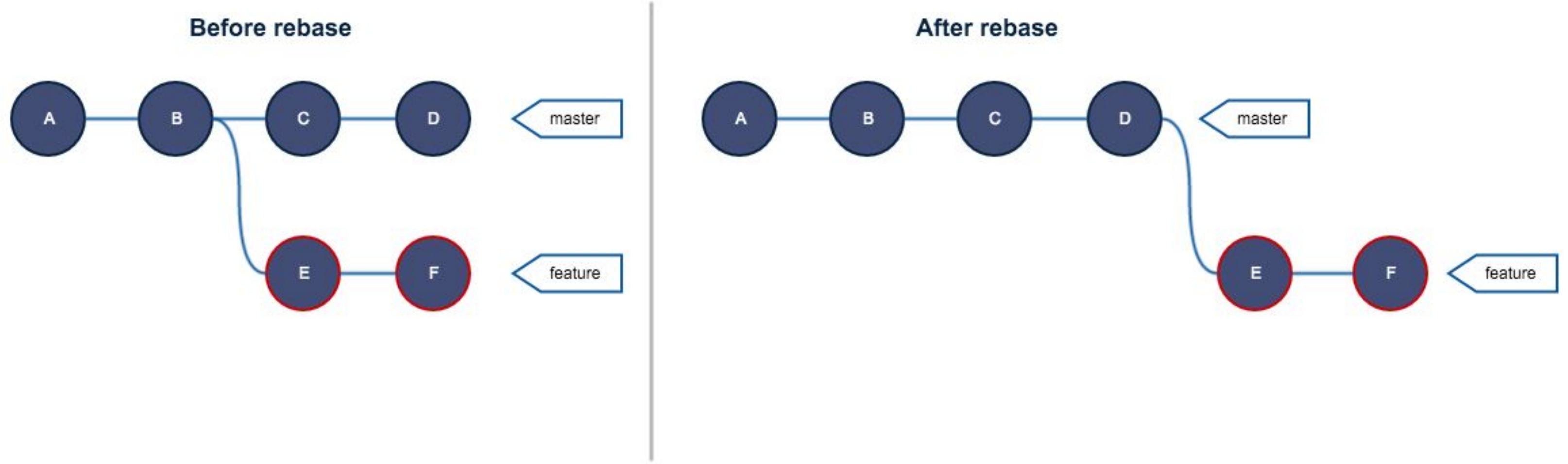
Solving Conflicts

When there are also new commits in master as well as in my-branch and they modified the sames lines in a file or created files with the same name

Git rebase

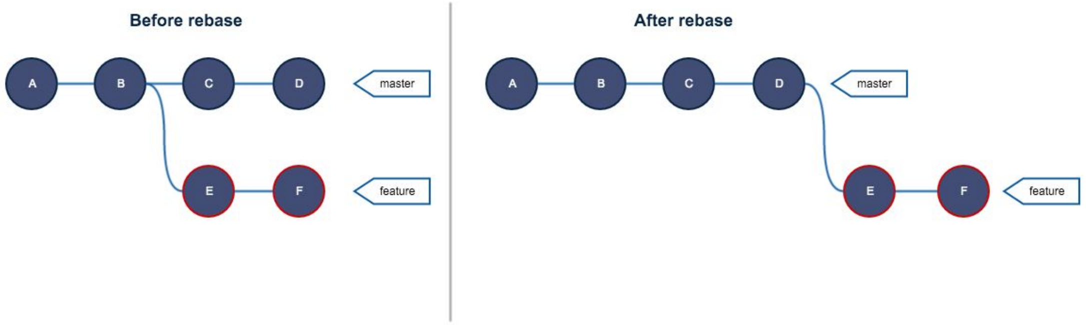
Imagine you need to bring some new commits from master to your branch but you do not one to create a merge commit

git rebase: workflow



git rebase: workflow

```
git checkout my-branch
git rebase master
```



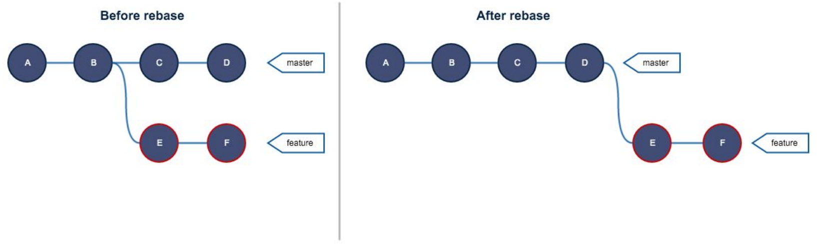
rewinding head to replay your work on top of it

git rebase: workflow

Example of rebasing master into my-branch

1. Move the commits you have made to my-branch to a temporary place.
2. Update my-branch with the new commits that have been created in master.
3. Apply the new commits you have made to my-branch, which had now been updated with respect to master.

The result is that my-branch will now have the same commits as master, in addition to the new commits you had in my-branch.



Local vs Remote repos

Let’s practice GIT branching

Merging vs Rebasing

From a conceptual standpoint, git merge and git rebase are used to achieve the same ultimate goal: to integrate changes from one branch into another branch.

There are, however, distinct mechanics to both methods.

Git merge

- Git merge will combine multiple sequences of commits into one unified history.
- In the most frequent use cases, git merge is used to combine two branches.
- git merge takes two commit pointers, usually the branch tips, and will find a common base commit between them.
- Once Git finds a common base commit it will create a new "merge commit" that combines the changes of each queued merge commit sequence.

git merge: pros

- Generally the easiest option to merge your master branch into your current working feature branch.
- You can `git checkout feature` and then `git merge master` or you could just do it with one command: git merge feature master
- This create a new ‘merge commit’ in your feature branch, which is a non-destructive operation that ties the histories of both branches.
- This preserves the exact history of your project

git merge: cons

- The branch that you merge will always have an extraneous merge commit that will be tracked every time you need to incorporate upstream states.
- In other words, it essentially creates a forked history at the point where you merge.
- This can lead to muddling the history of your branch, thereby making it more difficult for yourself or other developers to track the history of changes using `git log` and/or roll back to previous states

git rebase: pros

- To rebase, you would `git checkout feature` and then `git rebase master`.
- Instead of creating a merge commit, rebase will move the entire feature branch to start from the tip of the master branch by rewriting the project history and creating brand new commits for each commit in the original branch.
- The result is a singular history with no forking of the commit history.

git rebase: cons

- Because rebase rewrites project history, you lose the context provided by a merge commit, i.e. you won't be able to see when upstream changes were actually integrated into the feature branch.
- More importantly, you could potentially cause extreme difficulty by rebasing master to the tip of your feature branch, leading git to think that your master branch's history has diverged from the rest
- In doing so, everyone else would still be working from the original master branch, and both masters would need to be merged together.

GIT: Remote Repos

Remote repositories

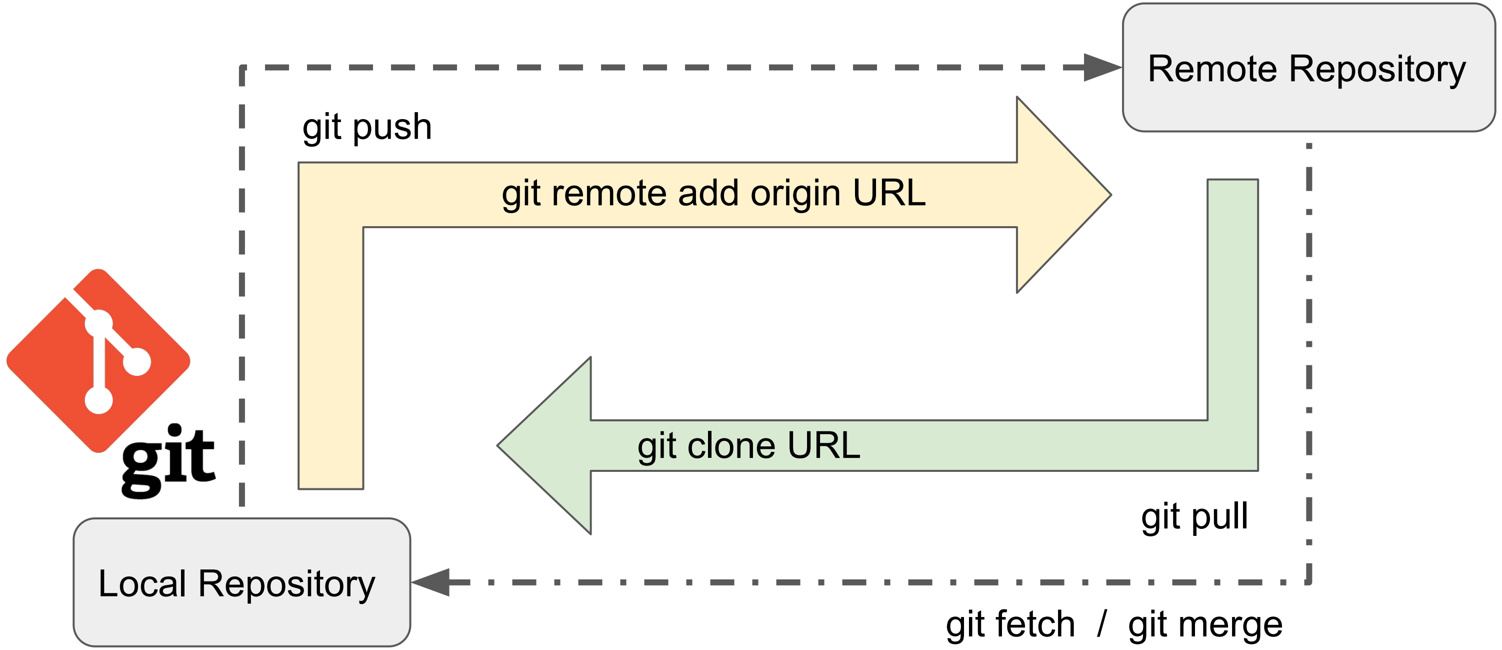
- A remote repository in Git, also called a remote, is a Git repository that’s hosted on the Internet or another network.

GitHub

- GitHub is **a Git repository hosting service**
- GitHub provides a Web-based graphical interface.
- It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

Let’s go to github

Git & GitHub



GitHub

GIT
Remote Commands

Commands

git remote add origin URL

git fetch -> git merge remote/Branch

git pull origin branch

git push origin branch

git clone URL

Remote Branches

git branch -a

git branch -r

git remote show origin

git branch –vv