Git 101

WHERE HAVE YOU BEEN ALL MY LIFE?

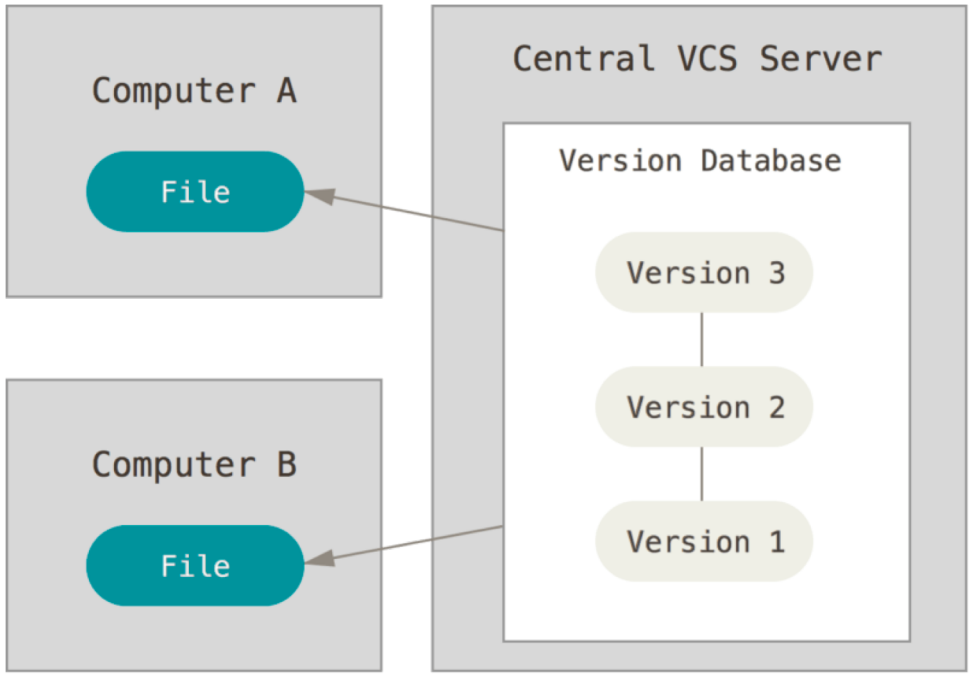
Me, Myself and I

- João Vale
- Automation Engineer at Blip (part of PaddyPower Betfair)
- Painfully aware of the minefield surrounding the term “DevOps”
- You can follow my yearly gems here: @jvale

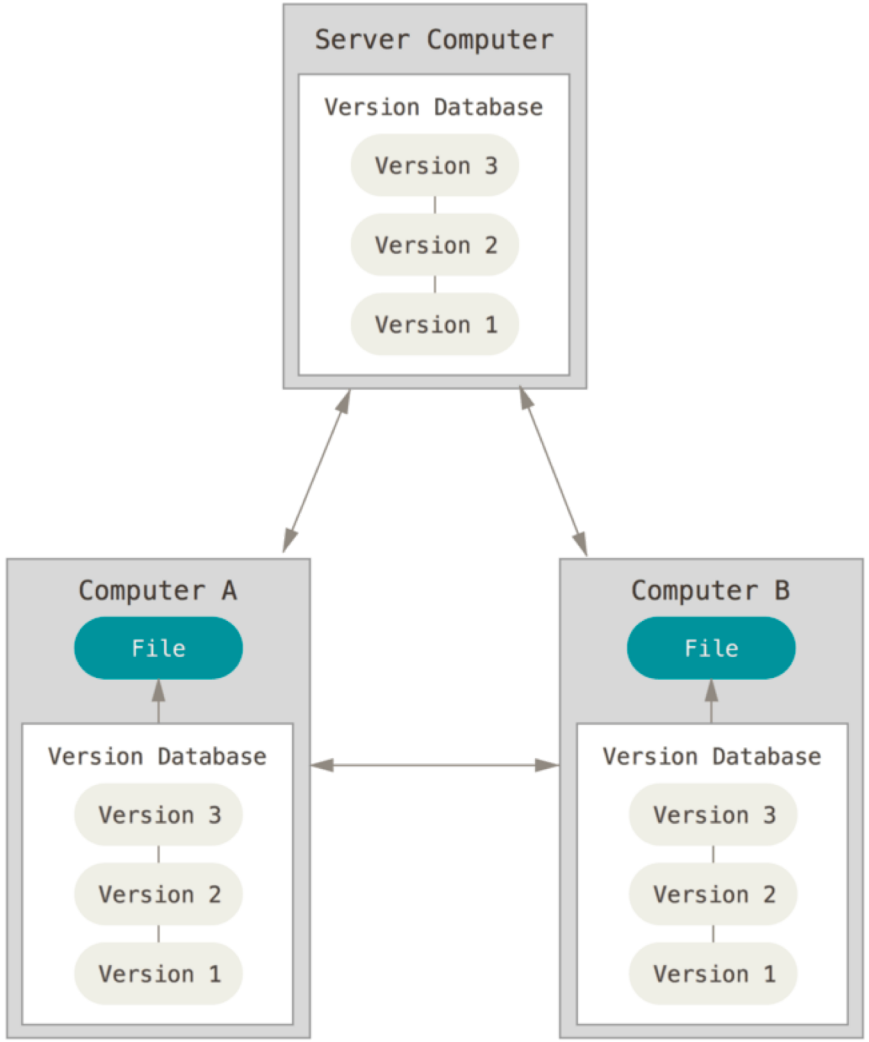
What is a Version Control tool?

- System that records changes to files over time
  - Who? What? When?
- Ability to go back in time
  - No silly stuff like `code.py`, `code_final.py`, `code_final_v3.py`, `code_final_v4.py.bak`
- Implicit backup of all files

Centralized vs decentralized



VS



Enter Git

PADDYPOWER betfair

- Created by Linus Torvalds for the Linux kernel
- Fully distributed
- Built for non-linear development
- Currently the *de facto* standard (87%, StackOverflow Developer Survey 2018)

Git base concepts

- Base unit of information is the ***commit***
- Each commit is a snapshot, not a set of differences
- Contains a reference to a ***parent*** commit

Getting started!


```
$ git init
Initialized empty Git repository in /Users/joaovale/Projects/mob_workshop/.git/
```

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

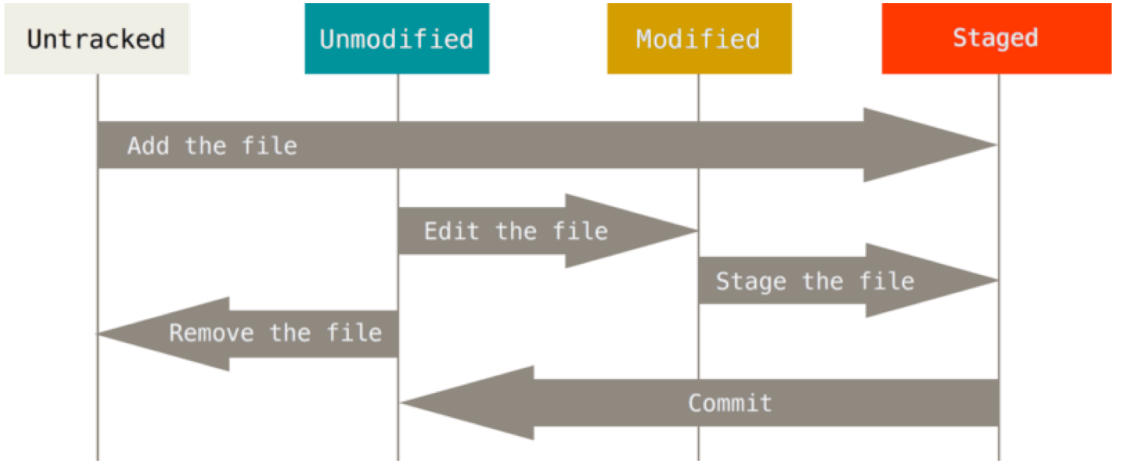
Getting started!


```
$ echo "Hello World" > file.txt
```

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    file.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Index

- File states: untracked, unmodified, modified, staged
- What is the ***index***?
  - a.k.a. *staging area / cache*
  - changes that will be added to the next commit
- Allows you to prepare all the changes for a commit
  - or multiple commits!
- Easier to deal with big merges (with potentially a lot of conflicts)



Index


```
$ git add file.txt
```

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   file.txt
```

Getting started!


```
$ git commit file.txt -m "First commit"
[master (root-commit) e69c102] First commit
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
```

```
$ git status
On branch master
nothing to commit, working tree clean
```

```
$ git log
commit e69c102b0370cb26a9df8303296508e53ec1acd6 (HEAD -> master)
Author: Joao Vale <joao.vale@blip.pt>
Date:   Wed Apr 11 16:27:38 2018 +0100

    First commit
```

Getting started!


```
$ echo "Hello again" >> file.txt
```

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
$ git diff
diff --git a/file.txt b/file.txt
index 557db03..49e9db5 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1,2 @@
 Hello World
+Hello again
```

Getting started


```
$ git add file.txt
```

```
$ git diff
```

- Oops, added the wrong change! How do I undo it?

```
$ git reset file.txt
Unstaged changes after reset:
M   file.txt
```

```
$ git diff
diff --git a/file.txt b/file.txt
index 557db03..49e9db5 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1,2 @@
 Hello World
+Hello again
```

Getting started!


```
$ git add file.txt
$ git commit file.txt -m "Update example"
[master 7455996] Update example
 1 file changed, 1 insertion(+)
```

```
$ git log --graph --decorate --pretty=format:'%h%d %s %ar %cn' --all
* 7455996 (HEAD -> master) Update example 3 minutes ago Joao Vale
* e69c102 First commit 55 minutes ago Joao Vale
```

```
$ git blame file.txt
^e69c102 (Joao Vale 2018-04-11 16:27:38 +0100 1) Hello World
74559960 (Joao Vale 2018-04-11 17:20:24 +0100 2) Hello again
```

Moving around


```
$ cat file.txt
Hello World
Hello again
```

```
$ git checkout e69c102
Note: checking out 'e69c102'.

[...]

HEAD is now at e69c102... First commit
```

```
$ cat file.txt
Hello World
```

```
$ git log --graph --decorate --pretty=format:'%h%d %s %ar %cn' --all
* 7455996 (master) Update example 3 minutes ago Joao Vale
* e69c102 (HEAD) First commit 55 minutes ago Joao Vale
```

Tags

- Useful to mark specific points in history, like releases
- Tags are just pointers
  - You can `git checkout` to them like you would a commit

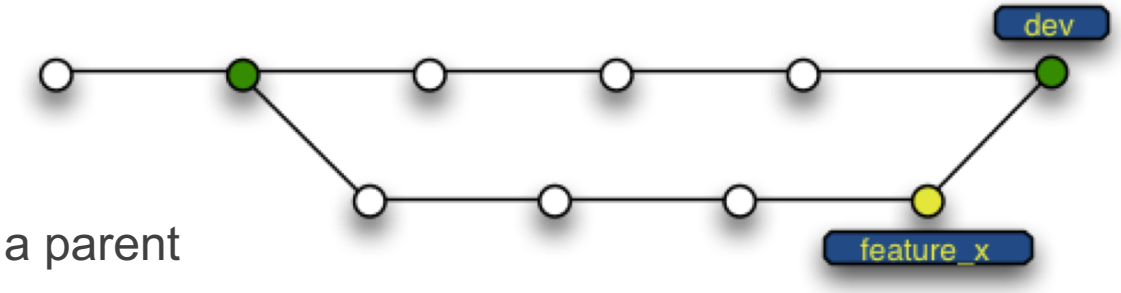
```
$ git tag v1.0
```

```
$ git tag --list
v1.0
```

```
$ git log --graph --decorate --pretty=format:'%h%d %s %ar %cn' --all
* 7455996 (HEAD -> master, tag: v1.0) Update example 3 minutes ago Joao Vale
* e69c102 First commit 55 minutes ago Joao Vale
```

Branching

- Diverge from the main line to work separately
- Branches in git are very lightweight
  - Do it often!
- Start of a branch is a regular commit that shares a parent
- A branch itself is just another pointer to a specific commit
- Merge is a regular commit with 2 parents



Branching


```
$ git branch
* master
```

```
$ git branch new_feature
$ git checkout new_feature
Switched to branch 'new_feature'
```

… make some changes …

```
$ git checkout master
Switched to branch 'master'
$ git merge new_feature
Updating 7455996..61c8437
Fast-forward
 new_file.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new_file.txt
```

Playing with others

- You can share your changes to other ***remote*** repositories

```
$ git remote add origin https://github.com/jvale/mob_git_workshop.git
$ git remote
origin
```

- ***origin*** is the standard name for default remote repository

Playing with others

- Once you have a remote configured, you can ***push*** your changes

```
$ git push --set-upstream origin master
Counting objects: 9, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 697 bytes | 697.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/jvale/mob_git_workshop.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

- You can also ***pull*** changes from remote repositories

```
$ git pull
Already up to date.
```

Contributing

PADDYPOWER betfair

- If you have access, you can ***clone*** the repository directly:

```
$ git clone https://github.com/jvale/mob_git_workshop.git
Cloning into 'mob_git_workshop'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
```

- Usually you don't have permissions to ***push*** to public repositories
  - You have to create your own copy to create changes (in GitHub, a ***fork***)

- After you've done your amazing stuff, create (in GitHub) a ***Pull Request***
  - Project maintainers will review it and, if it's good, merge it
  - If it's not good enough, they'll point you in the right direction

PR etiquette

- Check documentation for instructions
  - *README*, *CONTRIBUTING*, etc
- Name your Pull Request clearly
- Describe your Pull Request
- Make changes manageable

`git init`

Enough talk, let’s do stuff!

Getting your hands dirty

- Install git
  - OS X: `brew install git` / install Xcode Command Line Tools
  - Linux: `apt-get/yum install git`
  - Windows: http://git-scm.com/download/win
- Configure it:
  - `git config --global user.name "John Doe"`
  - `git config --global user.email johndoe@example.com`
- Fork this repo: http://github.com/jvale/mob_git_workshop
- Follow the instructions!

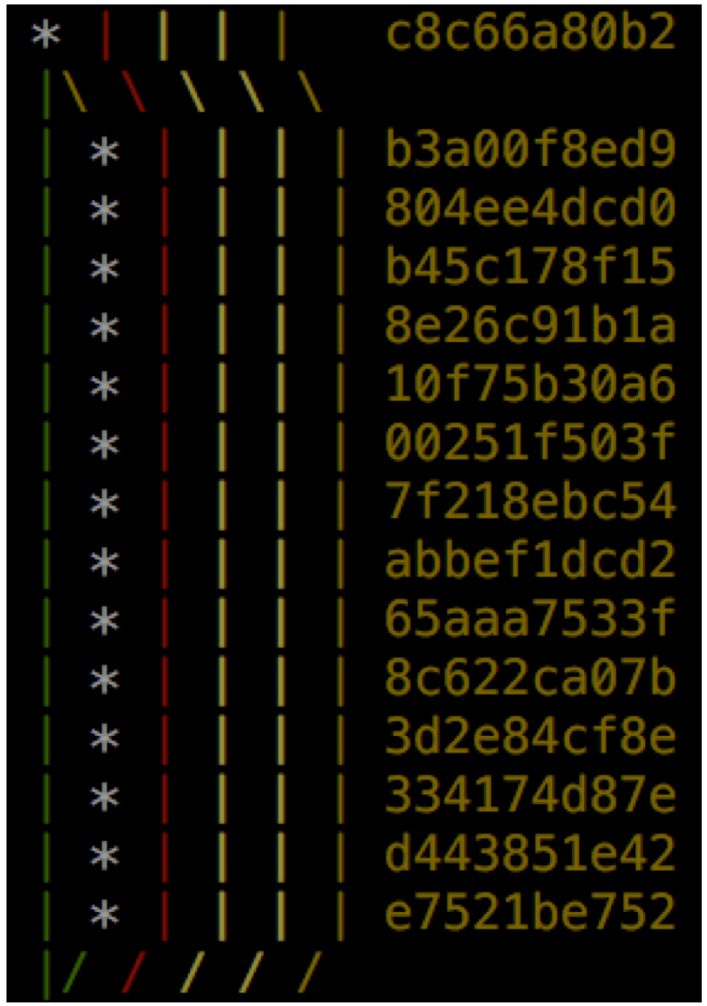
Above and beyond

Some more advanced topics

Nifty tricks

PADDYPOWER. betfair

- git add --interactive
- git commit --amend
- git pull --rebase



Stash it away

- Sometimes you may need to move through the repo and have uncommitted work
  - Or even just put it aside to fix some other issue

```
$ git stash
Saved working directory and index state WIP on master: b62a77e New layout
```

```
$ git stash list
stash@{0}: WIP on master: b62a77e New layout
```

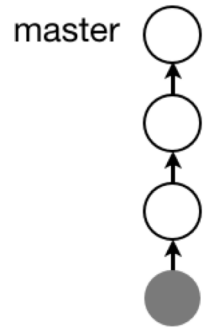
```
$ git stash pop
On branch master
Your branch is up to date with 'origin/master'.

[...]

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (cb29b81fb6e5e28dbec6a3a2e8baab481cba7a64)
```

Playing with others

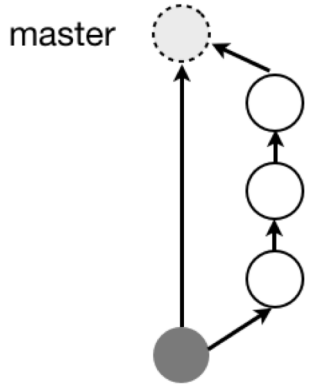
- Remember that “decentralized” part? A ***remote*** can be anywhere! You can:
  - git push/pull to/from another laptop
  - git push/pull to/from another directory

Merge types

- fast-forward
  - Default behavior; just updates the branch pointer
  - Common case: branch based in ***master*** tip will be collapsed when merged



- no fast-forward
  - Always create a merge commit
  - An explicit branch and merge will be recorded in repository tree



Source: Ariya Hidayat, Fast-Forward Git Merge

Rebase

- Operation to change the base of your branch to another place in the tree
  - Can rewrite history! With great power comes great responsibility.
- Simplest application: `git pull --rebase`
- Common application: `git rebase master`
- Advanced application: `git rebase --interactive`
  - Here be dragons! Code may disappear.

Bissect

- Want to find when a certain change was introduced? Git provides!

```
$ git bisect start
$ git bisect good abc1234
$ git bisect bad HEAD
$ git bisect run <command>

[...]
```

- Specify a last known good revision, a bad revision, and a command to test if it's good or bad
  - Git checks out revisions in between the range and drills down to the guilty change

Bissect

- Too much automation for you?

```
$ git bisect start
$ git bisect good abc1234
$ git bisect bad HEAD
```

- Inspect the codebase and tell Git if that revision is good or bad, and it will move on:

```
$ git bisect good
```

References

- THE book: “Pro Git” (available online: https://git-scm.com/book)

- As expected, Google and Stack Overflow have answers for the most common issues

…and we're done!

Questions?