git / GitHub workshop

Mon 09th Oct. 2023 – Ian Dirk Fichtner

Workshop outline

DAY 1: git

1) Concept
2) Main tools

→ Practical part 1

3) Terminal vs GUI
4) Git merge
5) Example workflows
6) Tips ‘n tricks

→ Practical part 2

DAY 2: GitHub

1) Github tour
2) Git-Github syncing

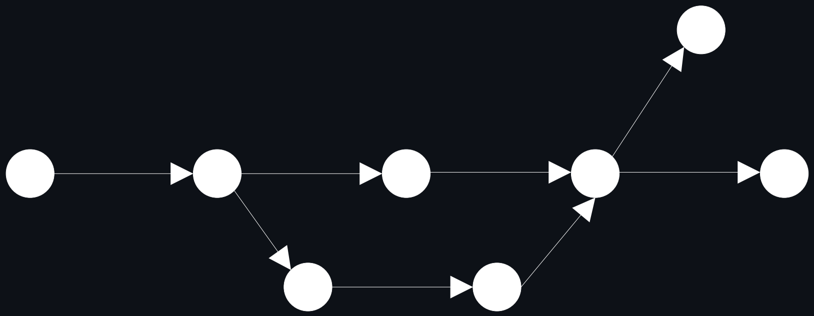
→ Practical part 1

3) Github collaboration
4) Git: down the rabbit hole

→ Practical part 2

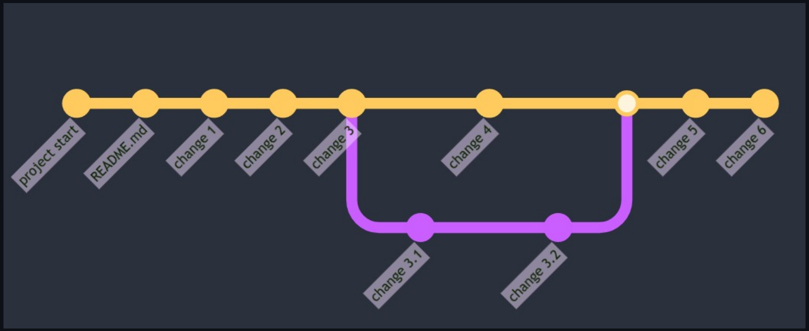
What is Git?

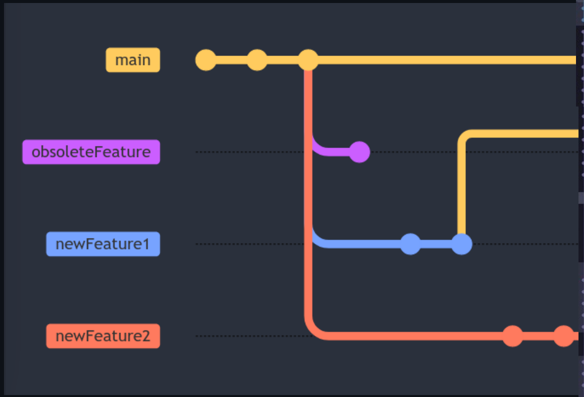
- Version control system
- Store snapshots of your project and its files
- Directed acyclic graph
- Linux, MacOS, Windows
- Usually used in combination with Github, Gitlab etc

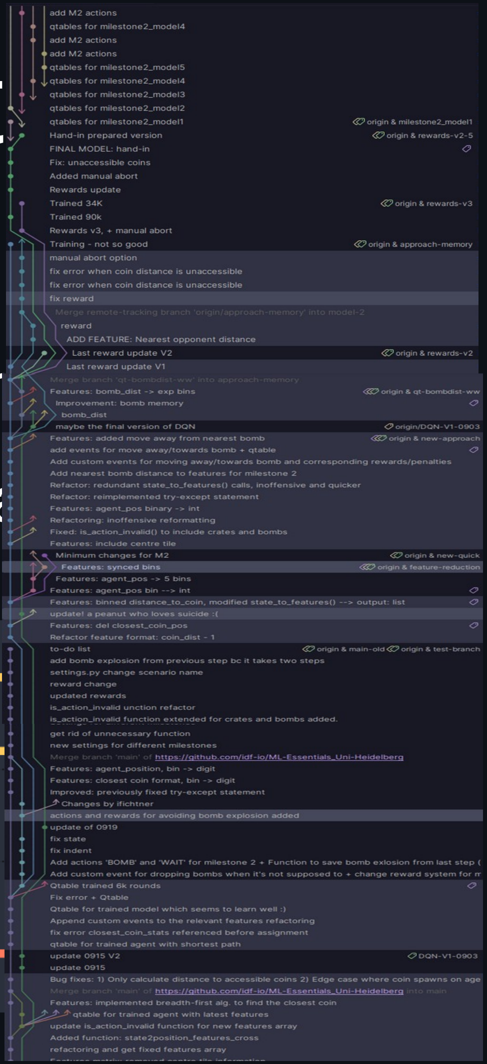


What can I do with git

1) Keep a record of file history
2) Manage and organize changes (features) of a pr
3) Play with different versions of a file e.g. model pa



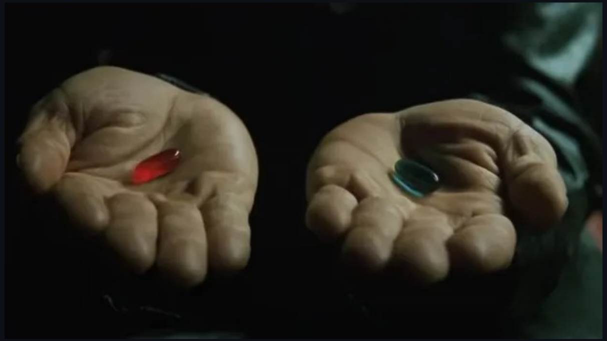



```
add M2 actions
qtables for milestone2_model4
add M2 actions
add M2 actions
qtables for milestone2_model5
qtables for milestone2_model4
qtables for milestone2_model3
qtables for milestone2_model2
qtables for milestone2_model1                                   origin & milestone2_model1
Hand-in prepared version                                        origin & rewards-v2-5
FINAL MODEL: hand-in
Fix: unaccessible coins
Added manual abort
Rewards update
Trained 34K                                                     origin & rewards-v3
Trained 90k
Rewards v3, + manual abort
Training - not so good                                          origin & approach-memory
manual abort option
fix error when coin distance is unaccessible
fix error when coin distance is unaccessible
fix reward
 Merge remote-tracking branch 'origin/approach-memory' into model-2
  reward
   ADD FEATURE: Nearest opponent distance
    Last reward update V2                                       origin & rewards-v2
   Last reward update V1
Merge branch 'qt-bombdist-ww' into approach-memory
Features: bomb_dist -> exp bins                                 origin & qt-bombdist-ww
 Improvement: bomb memory
  bomb_dist
 maybe the final version of DQN                                 origin/DQN-V1-0903
Features: added move away from nearest bomb                     origin & new-approach
add events for move away/towards bomb + qtable
Add custom events for moving away/towards bomb and corresponding rewards/penalties
Add nearest bomb distance to features for milestone 2
Refactor: redundant state_to_features() calls, inoffensive and quicker
Refactor: reimplemented try-except statement
Features: agent_pos binary -> int
Refactoring: inoffensive reformatting
Fixed: is_action_invalid() to include crates and bombs
Features: include centre tile
 Minimum changes for M2                                         origin & new-quick
  Features: synced bins                                         origin & feature-reduction
 Features: agent_pos -> 5 bins
Features: agent_pos bin --> int
Features: binned distance_to_coin, modified state_to_features() --> output: list
update! a peanut who loves suicide :(
Features: del closest_coin_pos
Refactor feature format: coin_dist - 1
to-do list                                     origin & main-old origin & test-branch
add bomb explosion from previous step bc it takes two steps
settings.py change scenario name
reward change
updated rewards
is_action_invalid unction refactor
is_action_invalid function extended for crates and bombs added.
get rid of unnecessary function
new settings for different milestones
Merge branch 'main' of https://github.com/idf-io/ML-Essentials_Uni-Heidelberg
Features: agent_position, bin -> digit
Features: closest coin format, bin -> digit
Improved: previously fixed try-except statement
  Changes by ifichtner
actions and rewards for avoiding bomb explosion added
update of 0919
fix state
fix indent
Add actions 'BOMB' and 'WAIT' for milestone 2 + Function to save bomb exlosion from last step (
Add custom event for dropping bombs when it's not supposed to + change reward system for m
Qtable trained 6k rounds
Fix error + Qtable
Qtable for trained model which seems to learn well :)
Append custom events to the relevant features refactoring
fix error closest_coin_stats referenced before assignment
qtable for trained agent with shortest path
update 0915 V2                                                  DQN-V1-0903
update 0915
Bug fixes: 1) Only calculate distance to accessible coins 2) Edge case where coin spawns on age
Merge branch 'main' of https://github.com/idf-io/ML-Essentials_Uni-Heidelberg into main
Features: implemented breadth-first alg. to find the closest coin
    qtable for trained agent with latest features
update is_action_invalid function for new features array
Added function: state2position_features_cross
refactoring and get fixed features array
```

2 ways of perceiving Git

1) Simplistic: Branching versions of your work
2) Complete: DAG of commits and references

→ Choose your view depending on intent of usage



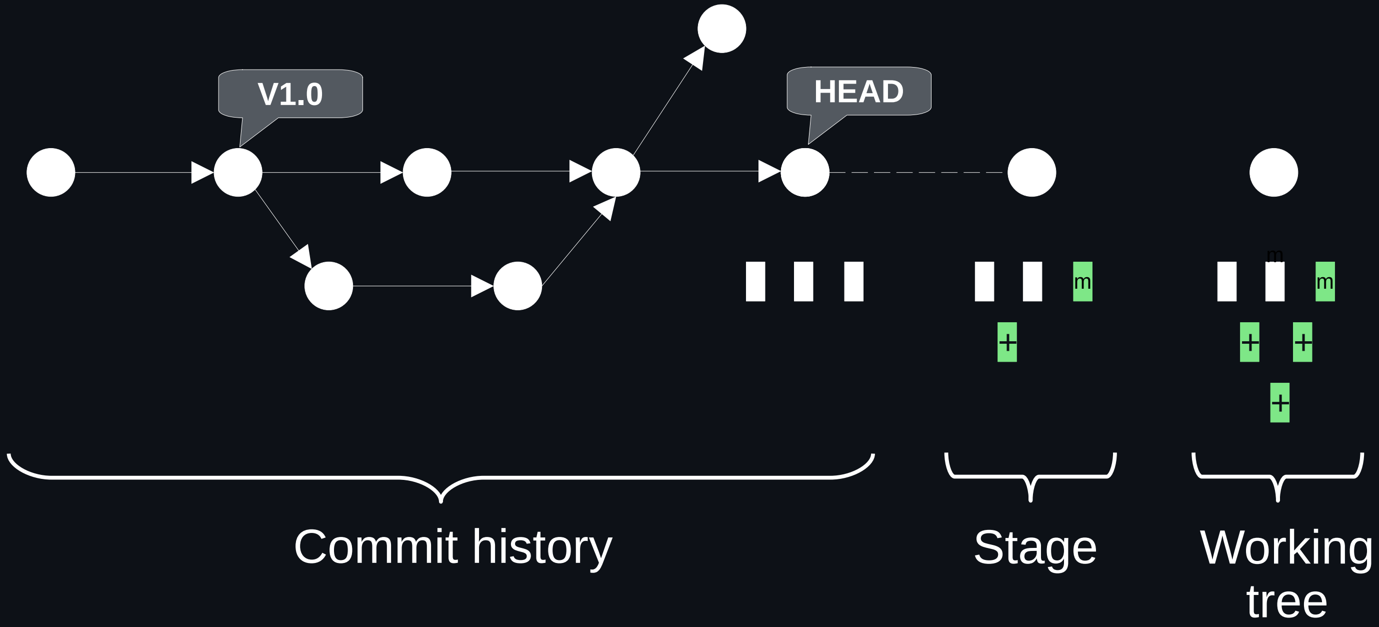
https://www.hitc.com/static/uploads/2021/09/Screenshot-2021-09-08-2.02.40-PM.png

Behind the scenes

- **Commit history DAG** (Immutable, append-only)
- **Mutable stage**
- **Working directory**

- **References**
  - HEAD (where you are)
  - Branches (heads)
    - → dynamic,
      current branch follows HEAD
  - Tags
    - → static

- Stored in .git/
- Commits are identified via SHA
  and contain metdata (author, date...)



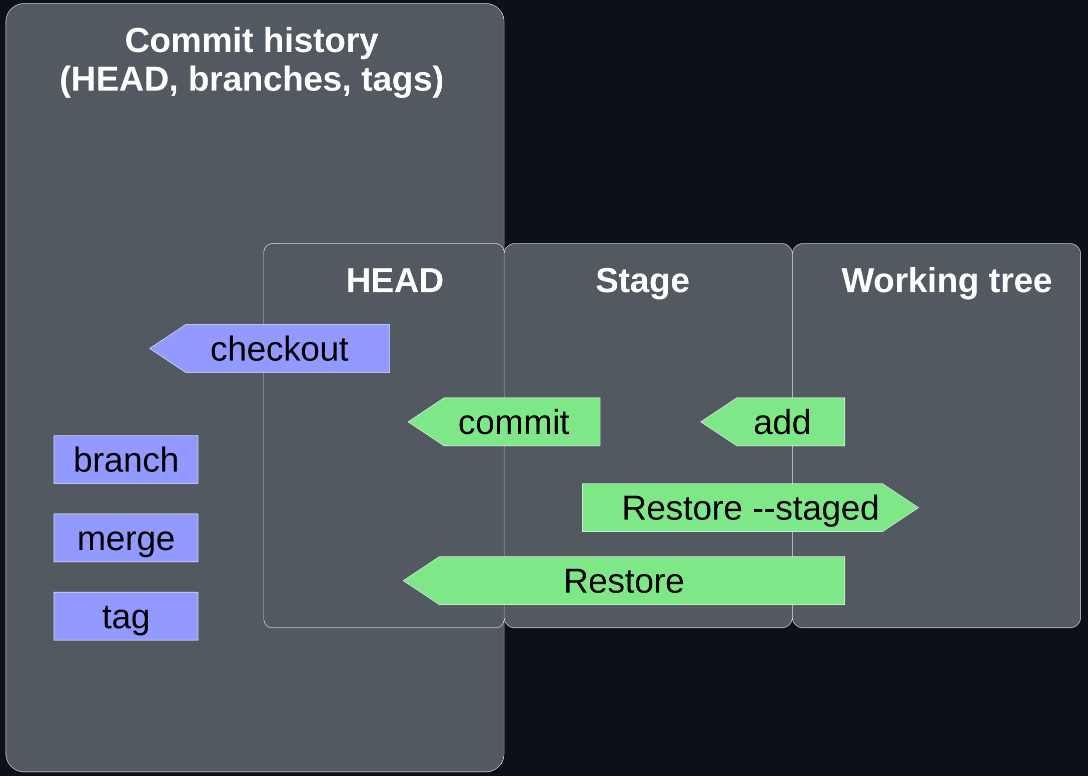
Terminologies

- Commit
- Commit history
- Working tree
- Stage (=index,cache)
- Reference (branch, HEAD, tags)
- Directed acyclic graph (DAG)

Your tool-set

Tool-set

- Init/Clone
- Commit (+ add)
- Status
- Log
- Checkout
- Branch
- Merge
- Tag



Check your working tree's status

```
> git status
```

- Find most relevant info
  - Where am I (HEAD)?
  - Which files have been modified with respect to the parent commit?
  - Which files are not being tracked?
  - Which files have been staged?
  - Are there any merge conflicts? → How to proceed explained
  - Are there any errors? → Solution 99.9% of cases indicated

View your commit history

```
> git log
> git log [file]
```

- Commit hash, message, author + email, date, local and remote references (branches, tags, HEAD)
- Only current branch parent commits

```
> git log --all --oneline –decorate --graph
```

- Only view commit has, message and references
- View all commits (also other branches

Getting started: 2 options

```
> git init
```

- Initiates repository in current folder
- In other words:
  - Creates the .git directory
  - No root commit yet!

```
> git clone [URL]
```

- Download the exact state of an online (remote) repo to your computer (local)

Make your commit history

```
> git add [file1] [fileN]
```

- Add files to the index/staging area

```
> git add .
> git add [--all, -A]
```

Make your commit history

```
> git commit -m “Commit message”
```

- Make a snapshot of the current state of your working tree
- You “commit” to a certain state → Irreversible (as of first)
- Conventions:
  - Make many short commits → history is easier to track (tags for bigger milestones)
  - Keep the message short and identifieable
    - ~~“After pre-processing the data files x and y performed UMAP with n_epochs=1000”~~
    - “Added UMAP to analysis X”

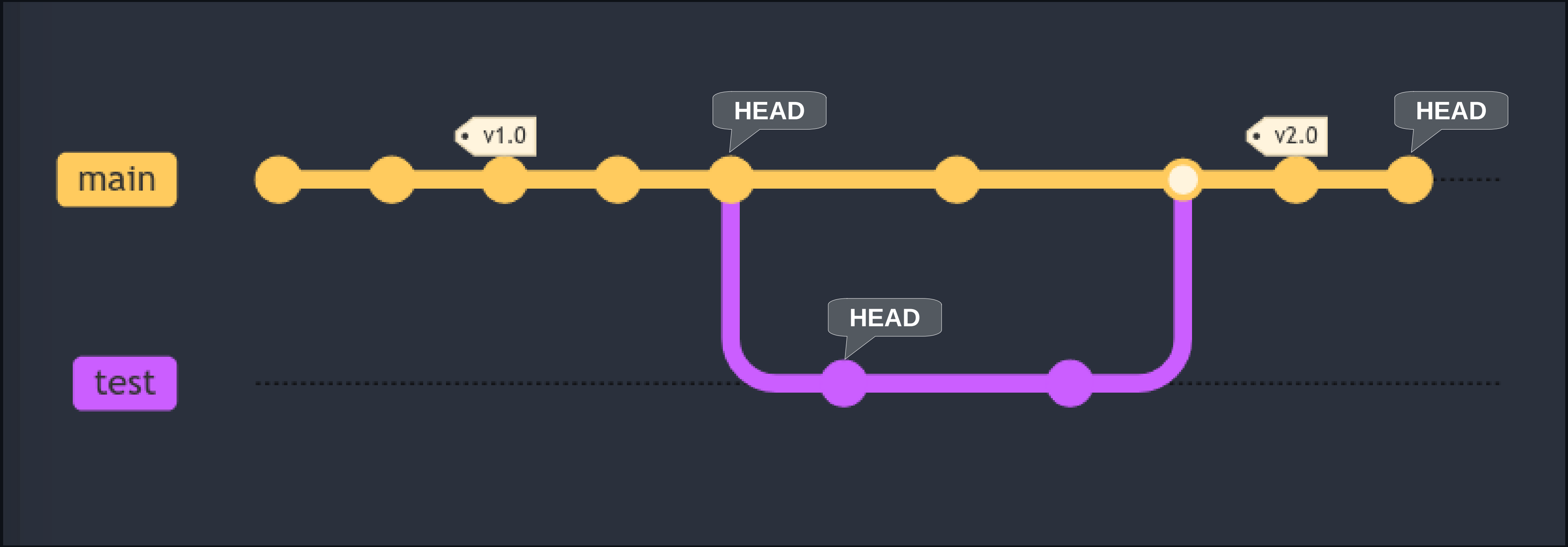
Make your commit history

```
Edit files

> git add [newFile] [modifiedFile] [deletedFile]
> git commit -m “Commit message”
```

- Why does `git add` exist?
- Lets you selectively commit work to the history

Time travel

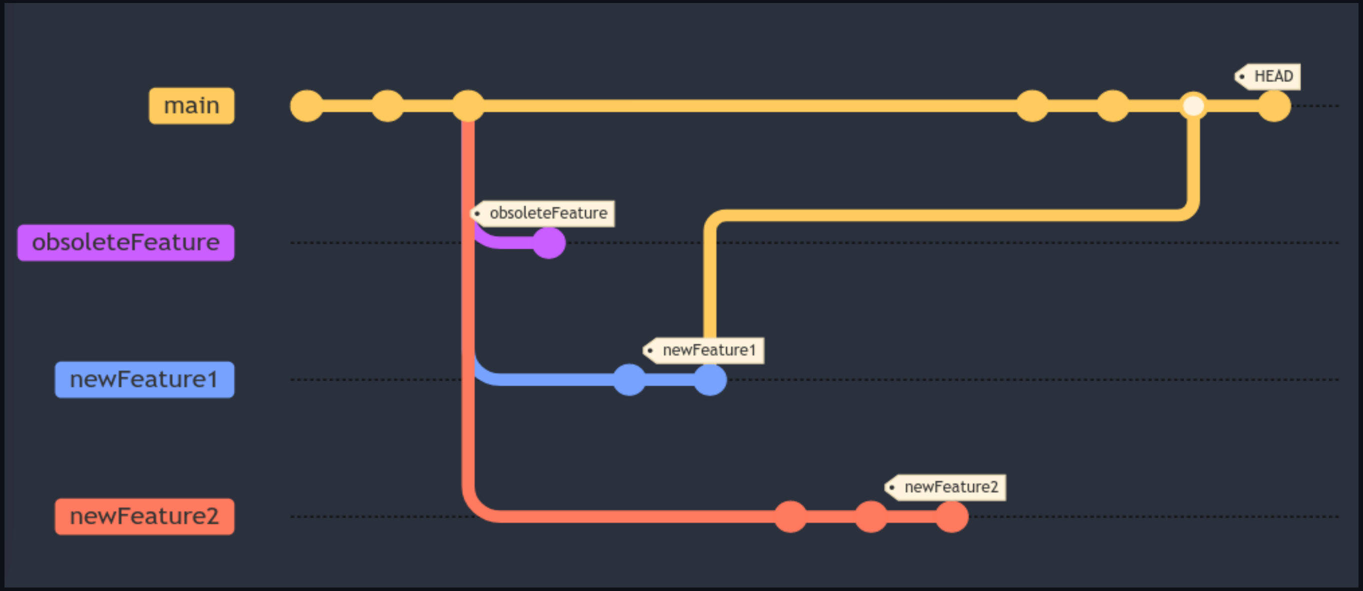
```
> git checkout [branch, tag, commit]
```



Branch out

```
> git branch [newBranchName]
```

- Make a new travelling reference



Branch out

```
> git branch -m [newBranchName]
```

- Rename the current branch

```
> git branch -d [BranchName]
```

- Delete branch

```
> git branch -D [BranchName]
```

- Delete branch with unmerged changes

Tag your commits

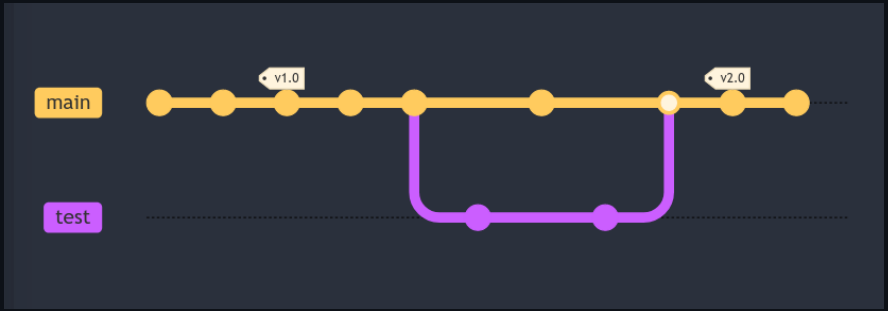
- **Tag = static reference**

```
> git tag [newTagName]
```

- Make a new tag in the current commit (HEAD)

```
> git tag [newTagName] [commitSHA]
```

- Make a new tag in the indicated commit



Tag your commits

```
> git tag
```

- Show all local tags

```
> git tag -d [tagName]
```

- Delete tag

Fun fact time!

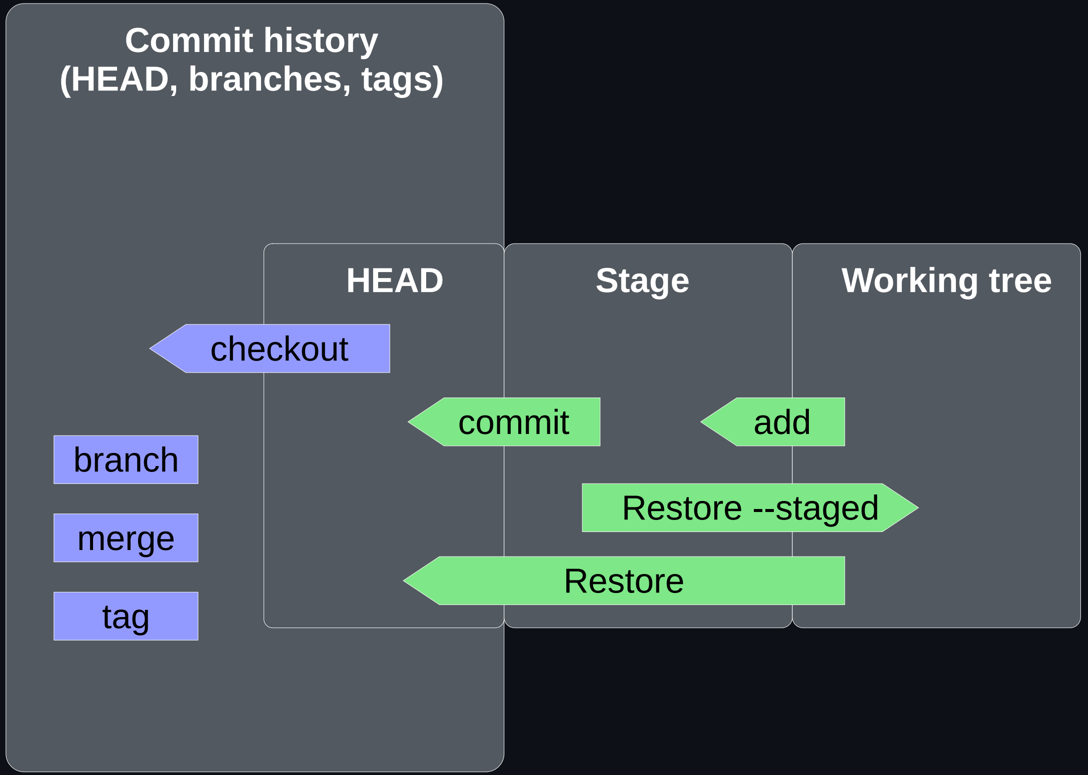
- Git was originally created by Linus Torwalds, the creator of the Linux kernel

#GOAT

https://en.wikipedia.org/wiki/Linus_Torvalds#/media/File:Lc3_2018_(263682303)_(cropped).jpeg

Tool-set

- Init/Clone
- Commit (+ add)
- Status
- Log
- Checkout
- Branch
- Merge
- Tag



Practical session part 1:
Try git out

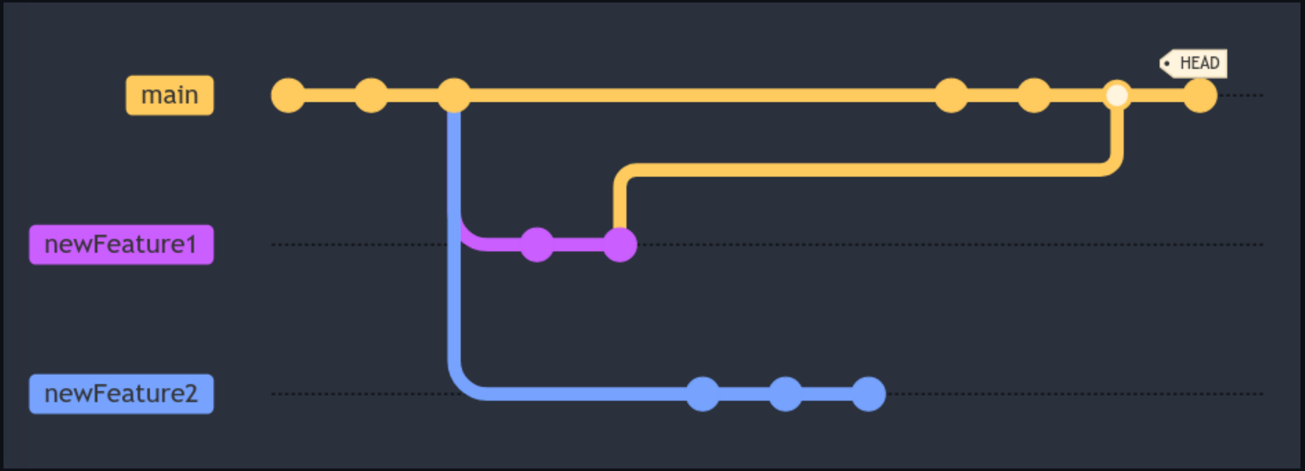
Fuse two branches

- Situation:
  - You started making changes in a new branch
  - You finished those changes
  - You liked them and want to update the code in your main branch (or any other arbitrary branch) with these changes

```
> git merge [branchWithChanges]
```

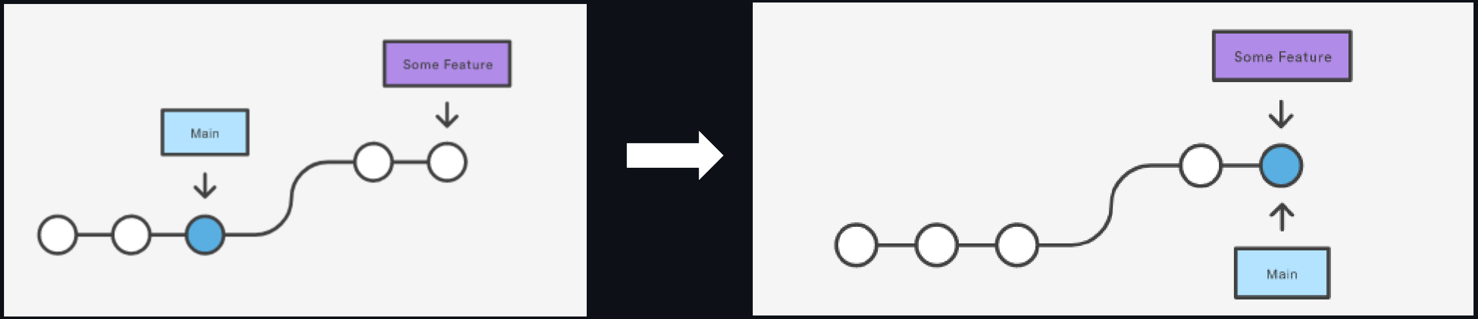
Fuse two branches

```
> git checkout [branch to update]

> git merge [branch with changes]

> git branch -d [obsolete branch]
```

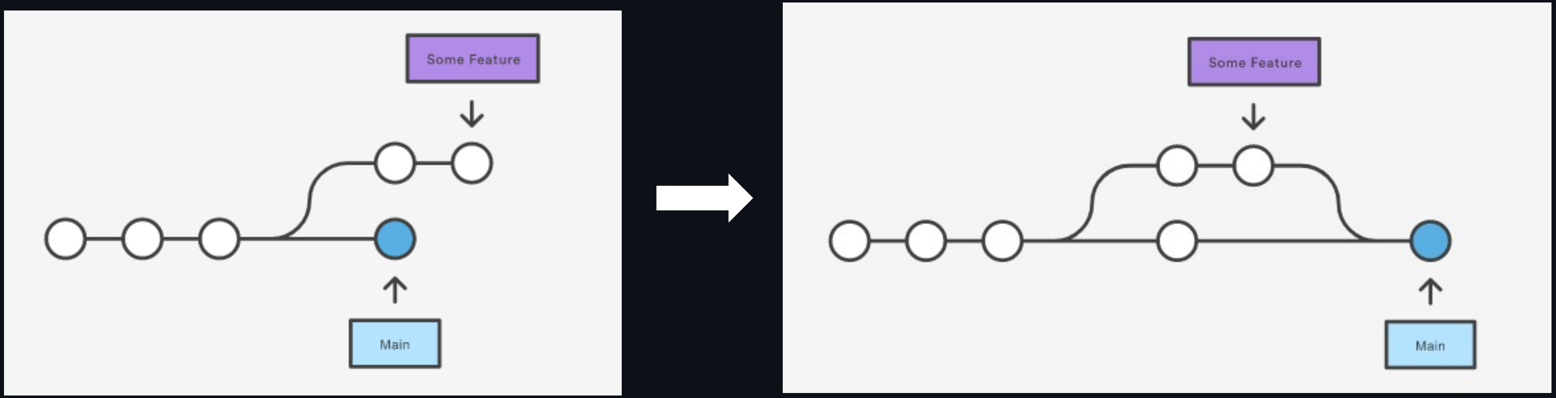


Merge types

1) **Fast forward merge**



2) **3-way merge**

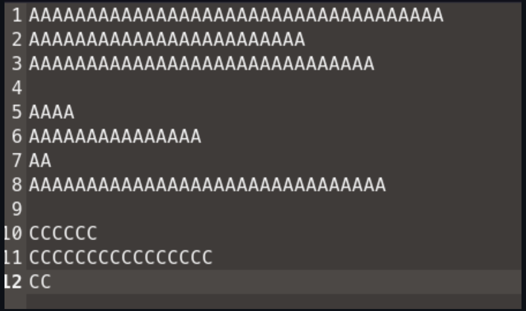


https://www.atlassian.com/git/tutorials/using-branches/git-merge

Git knows what’s best

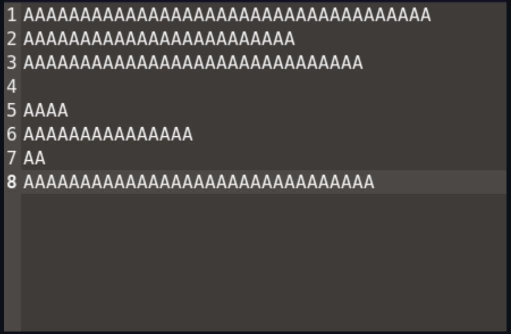
- **Git is very powerful**
- **Git will solve most code merging problems for you!**

MAIN



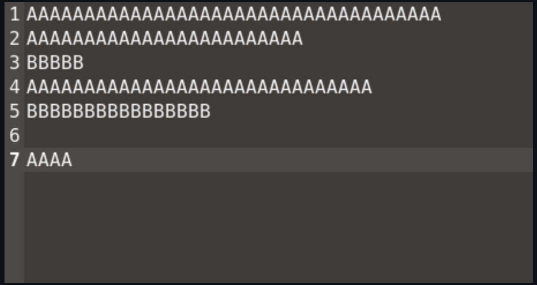
```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAA
AAAAAAAAAAAAAAA
AA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

CCCCCC
CCCCCCCCCCCCCCCC
CC
```

Common ancestor commit



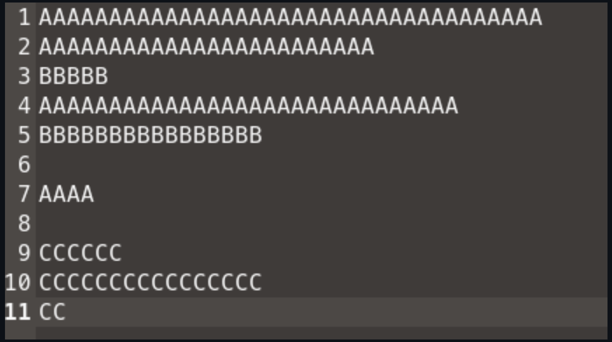
```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAA
AAAAAAAAAAAAAAA
AA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

New branch



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
BBBBB
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBB

AAAA
```

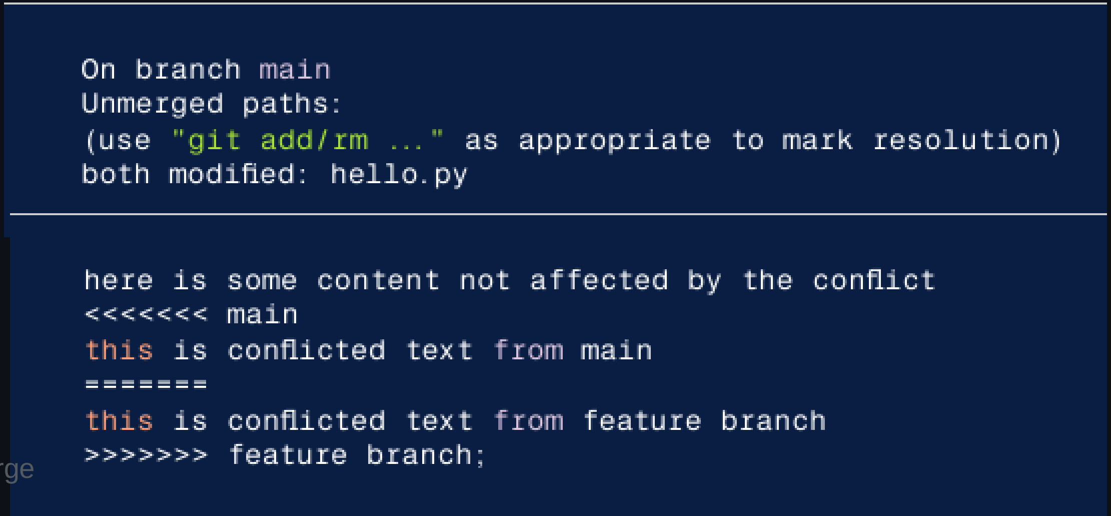
Merge



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
BBBBB
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBB

AAAA

CCCCCC
CCCCCCCCCCCCCCCC
CC
```

Merge conflicts

- Git detects when a merging branch has changed the same file in the same place as the current branch
- Enter merge conflict mode
  - → Uses familiar add + commit workflow
- Visual markers in conflicted files: <<<<<<<, =======, and >>>>>>>
- Fix conflicted sections manually
- Git add
- Git commit



```
On branch main
Unmerged paths:
(use "git add/rm ..." as appropriate to mark resolution)
both modified: hello.py
```

```
here is some content not affected by the conflict
<<<<<<< main
this is conflicted text from main
=======
this is conflicted text from feature branch
>>>>>>> feature branch;
```

https://www.atlassian.com/git/tutorials/using-branches/git-merge

terminal vs GUI

GUI clients

- GitHub desktop (Windows, macOS)
- GitKraken ($$$)
- Many IDEs (integrated development environment) offer Git and Github integration:
  - VS Code
  - Pycharm
  - None for RStudio

Terminal	GUI
Know/Look up commands → Granular control	Given command operations → Limited control
Terminal takes getting used to	Pretty interface
Oftentimes slower navigation	Oftentimes faster navigation
Forces you to learn Git	You can survive by knowing the basics

Example workflows

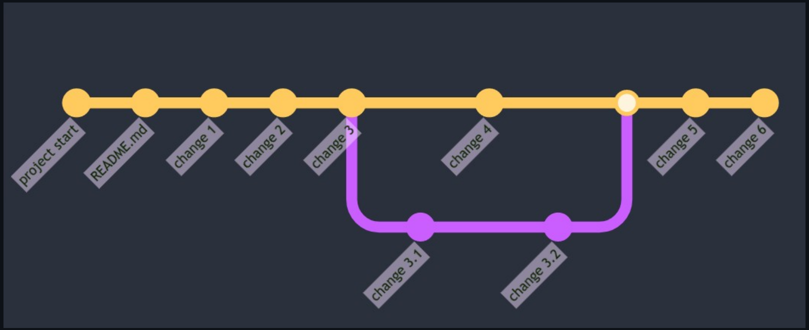
Example workflows

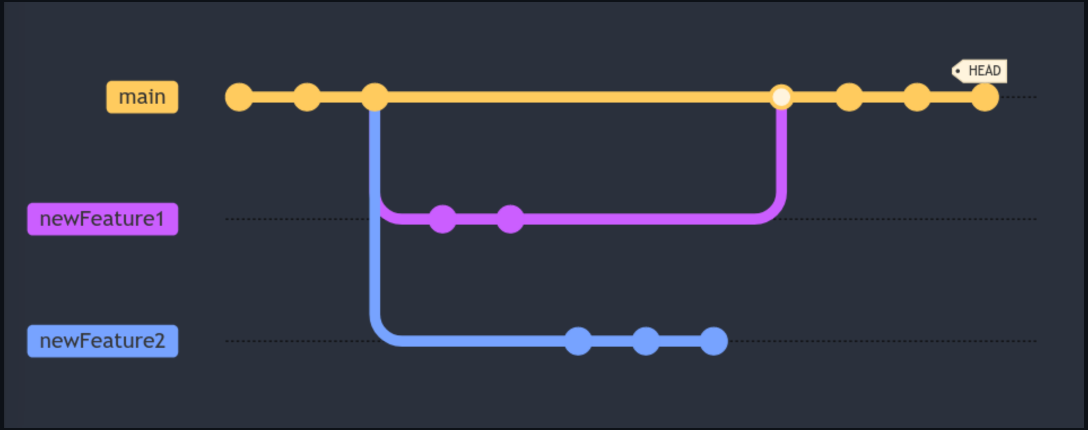
1) Work on MAIN

→ Test new changes in branches and then merge

2) Keep MAIN untouched

→ Work always on branches until ripe then merge





Tips ‘n tricks

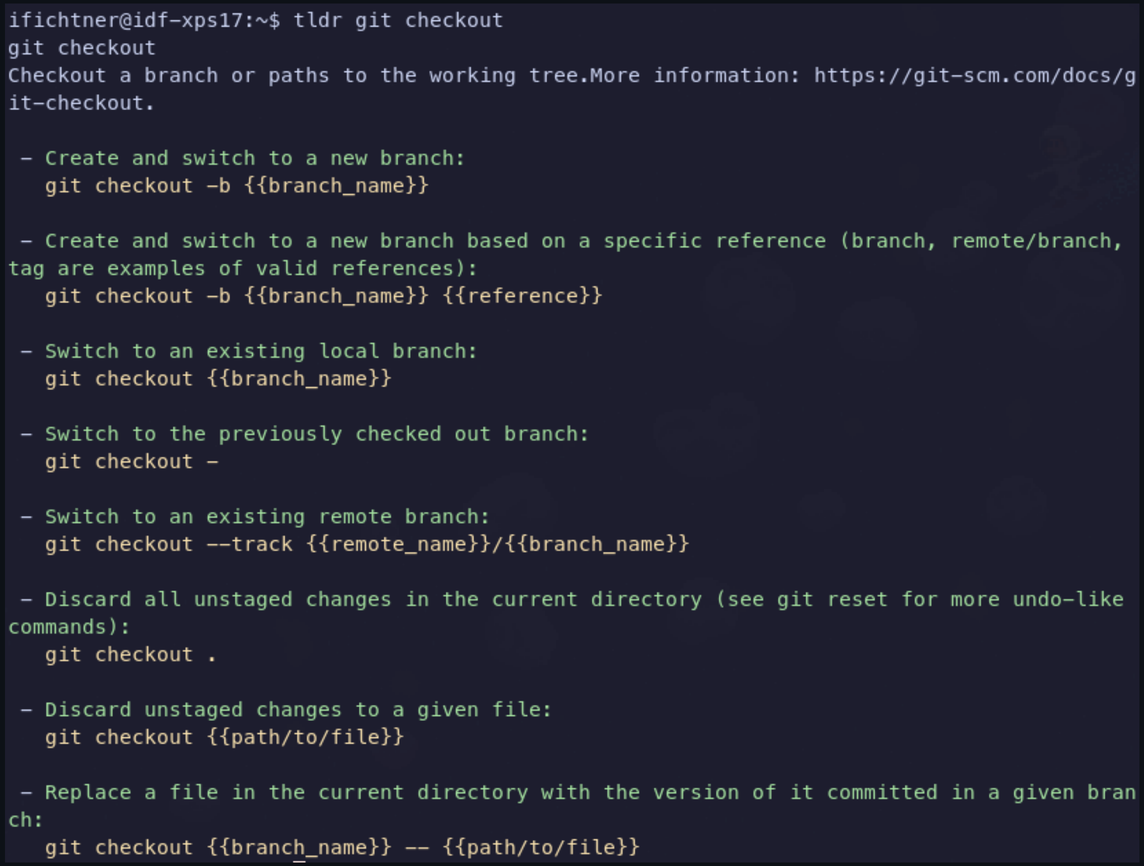
Time travel safely

1) Rest assured: git is very clever and won’t let you be reckless
2) Save/Backup your data
    - git commit → git reset -i
    - git stash
    - git branch
    - Duplicate entire repository → before major actions
3) Never work in a detached HEAD state

Install `tldr package`

on UNIX-based OS

for quick help



```
ifichtner@idf-xps17:~$ tldr git checkout
git checkout
Checkout a branch or paths to the working tree.More information: https://git-scm.com/docs/g
it-checkout.

 - Create and switch to a new branch:
   git checkout -b {{branch_name}}

 - Create and switch to a new branch based on a specific reference (branch, remote/branch,
tag are examples of valid references):
   git checkout -b {{branch_name}} {{reference}}

 - Switch to an existing local branch:
   git checkout {{branch_name}}

 - Switch to the previously checked out branch:
   git checkout -

 - Switch to an existing remote branch:
   git checkout --track {{remote_name}}/{{branch_name}}

 - Discard all unstaged changes in the current directory (see git reset for more undo-like
commands):
   git checkout .

 - Discard unstaged changes to a given file:
   git checkout {{path/to/file}}

 - Replace a file in the current directory with the version of it committed in a given bran
ch:
   git checkout {{branch_name}} -- {{path/to/file}}
```

Time travelling

- Untracked files will follow you around to anywhere

```
> git checkout [branch] – [file]
```

- Bring individual files from other commits to your working tree

Miscellaneous

```
> git add [--interactive | -i]
```

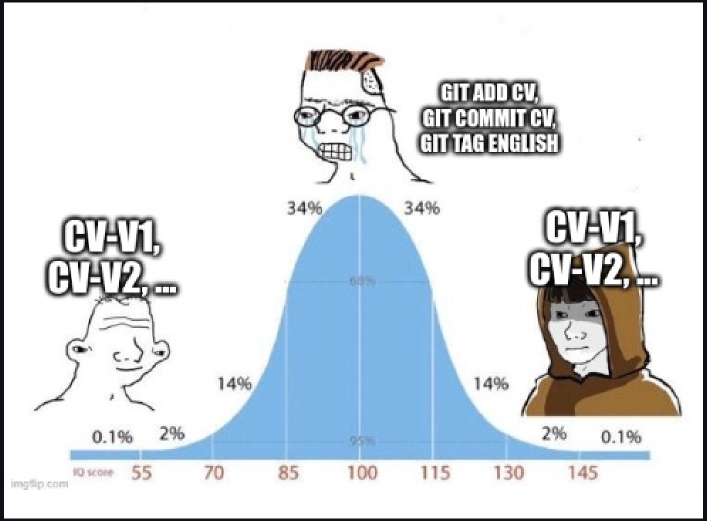
- Select what to do with individual files

```
> git checkout -b [newBranchName]
```

- Create a new branch and move to it

Git is a state of mind

- Start to rely less on where files are and more on keywords that you define every time you make a new branch or tag
  - Easier navigation
  - Focus on the work and less on `Where is this one file or code that I wrote two days ago?`
- Only use it where it makes sense



Practical session part 2:
Handle some merge conflicts