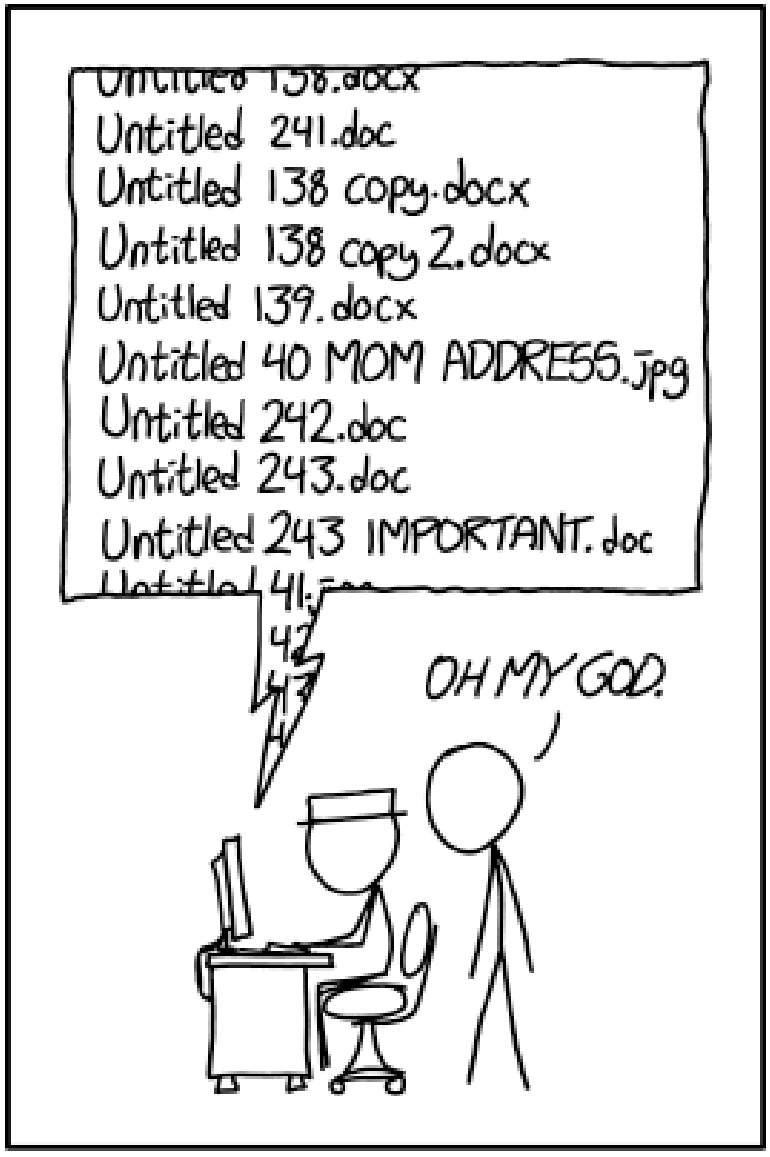
Using Git and GitHub

Videogames Technology
Asignatura transversal

Departamento de Automática

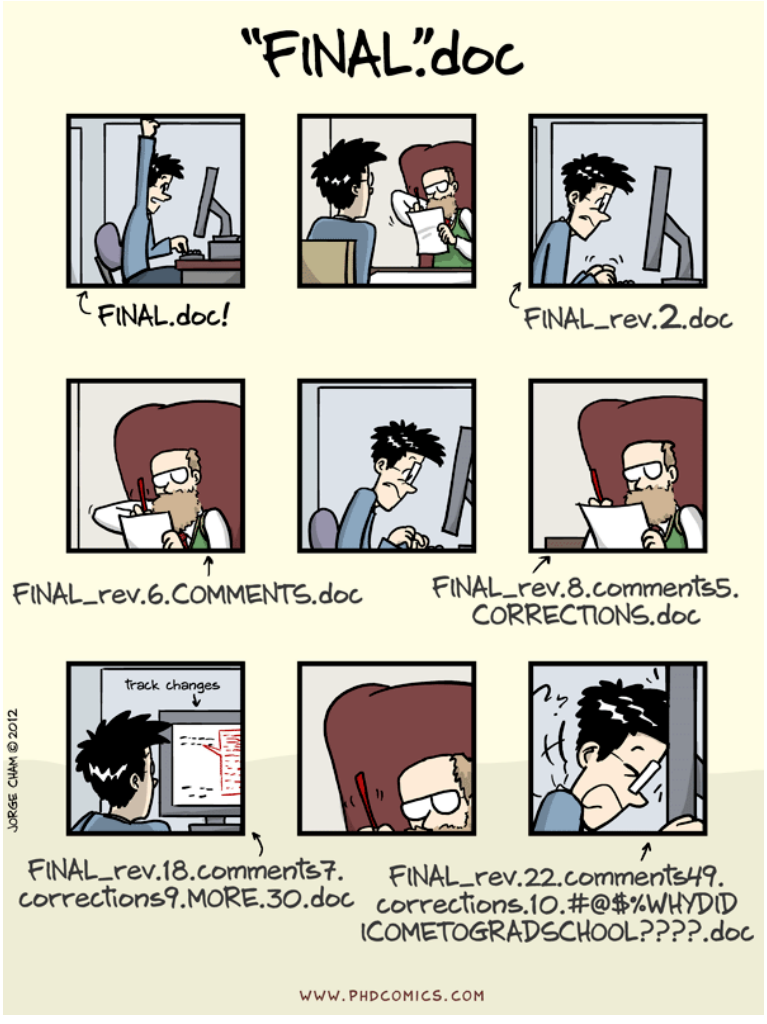
Universidad de Alcalá

Table of Contents



PROTIP: NEVER LOOK IN SOMEONE ELSE'S DOCUMENTS FOLDER.

(Source)



(Source)

Version control

Introduction

Version control systems

Version control systems (VCS) keep track of changes to source code. Allows multiple people to edit a project in a predictable manner.

Main open source VCS

- 1982 RCS
- 1990 CVS
- 2000 Subversion
- 2005 Git/Mercurial

There are many proprietary ones but `Git` is now the most popular one by far.
All software should be under a version control system, if not, it ain't software!

Git

What is Git?

Git is an open source distributed version control system, created by Linus Torvald.

`https://git-scm.com/`

(Interactive tutorial)

Git

Git sites

It is easier to start with free hosting sites instead of maintaining your own server.

- GitHub: public repositories (as many as you want), but private ones are not free (except for academia). It is now part of Microsoft
- Bitbucket: allow us to keep private repositories limiting the number of collaborators.
- GitLab: both public and private without limitations. It is becoming more popular.
- Others ...

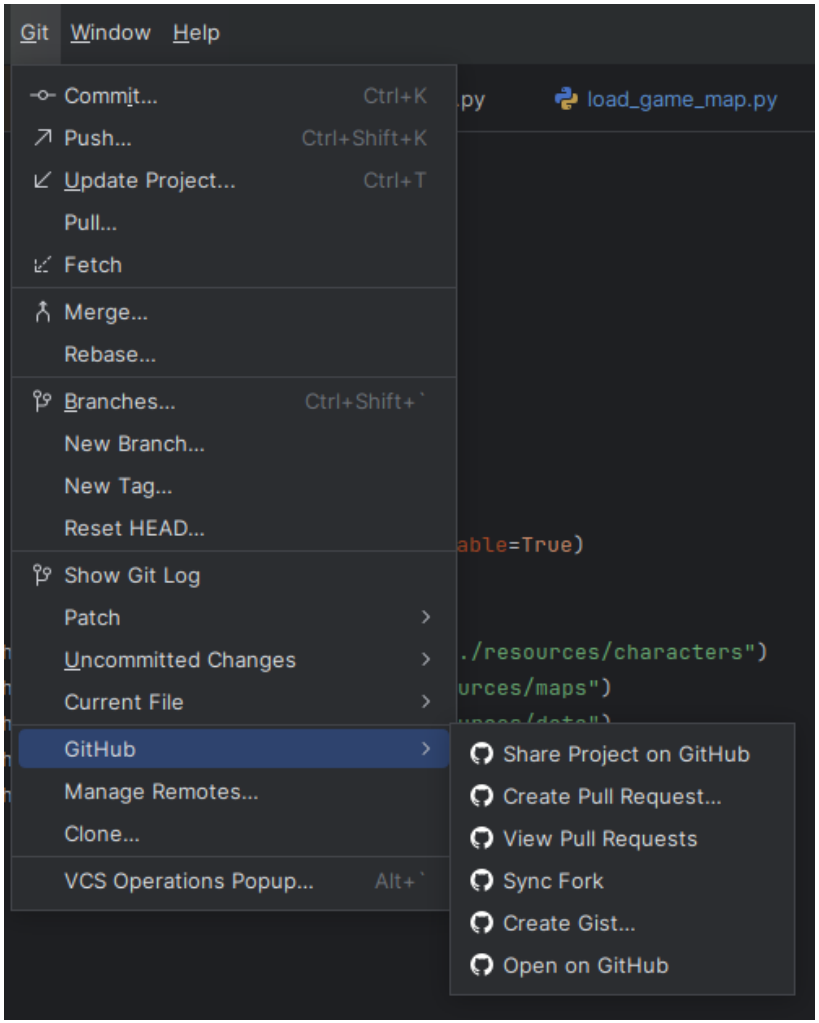
It is typically used as central repository:

- from which everyone pulls other people's changes
- to which everyone pushes changes they have made

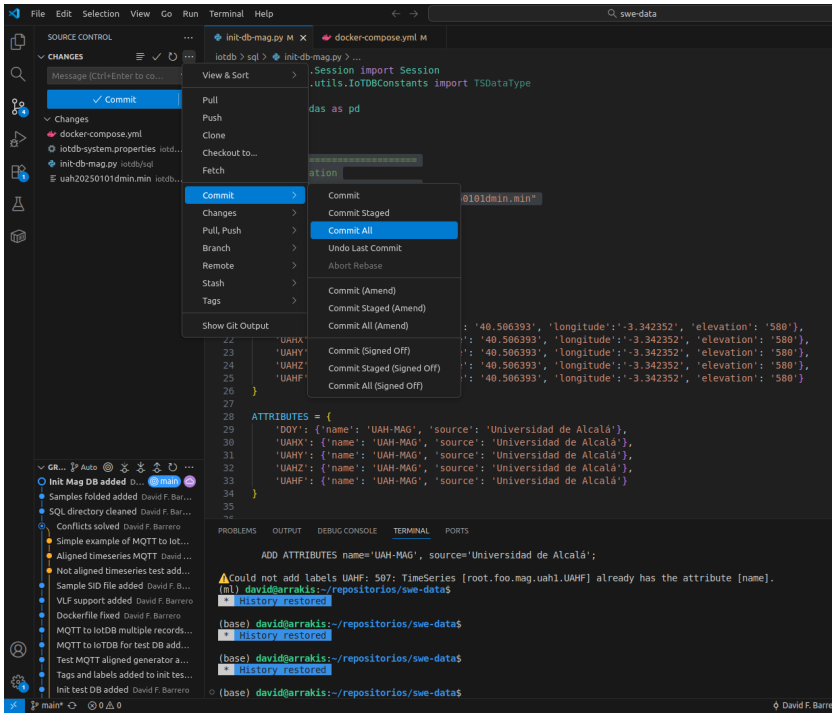
Git

Git in IDEs

PyCharm



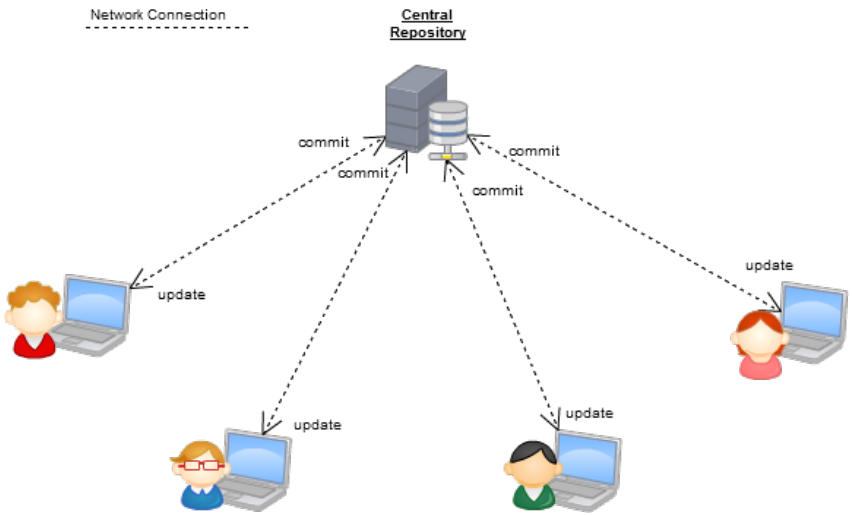
VSCode



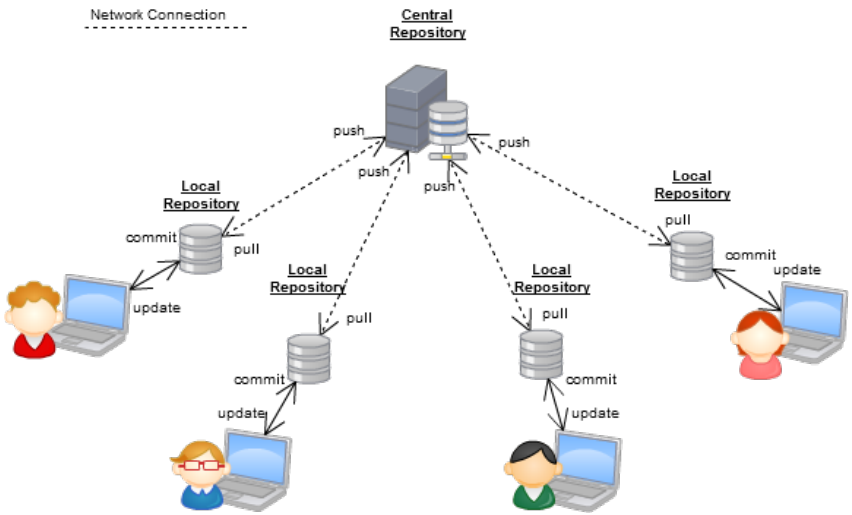
Git

Git vs. SVN (I)

Centralized (SVN)



Distributed (Git)



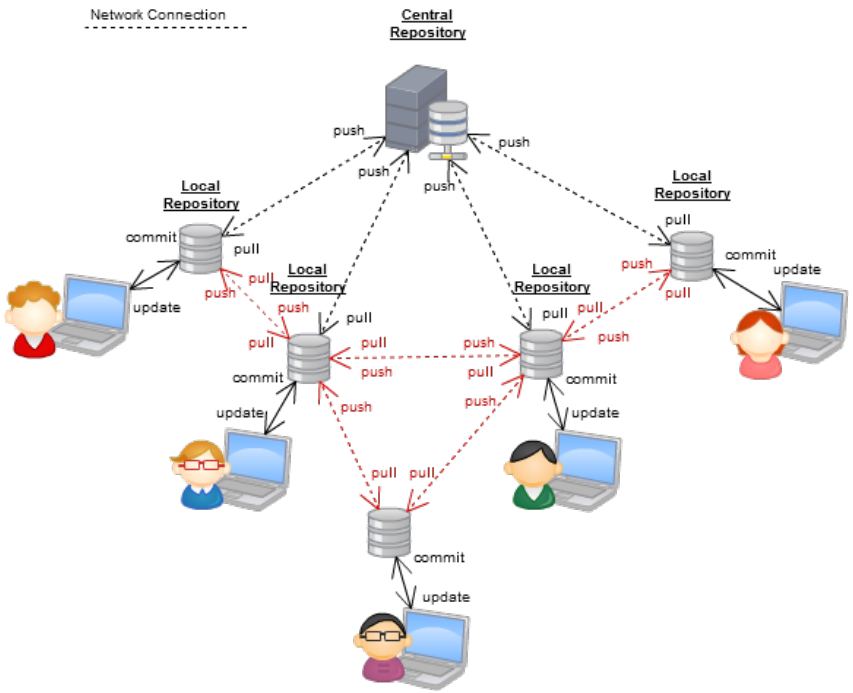
(Source)

Disclaimer: Do not pay attention to the labels of these diagrams

Git

Git vs. SVN (II)

Fully distributed (Git)



(Source)

Git

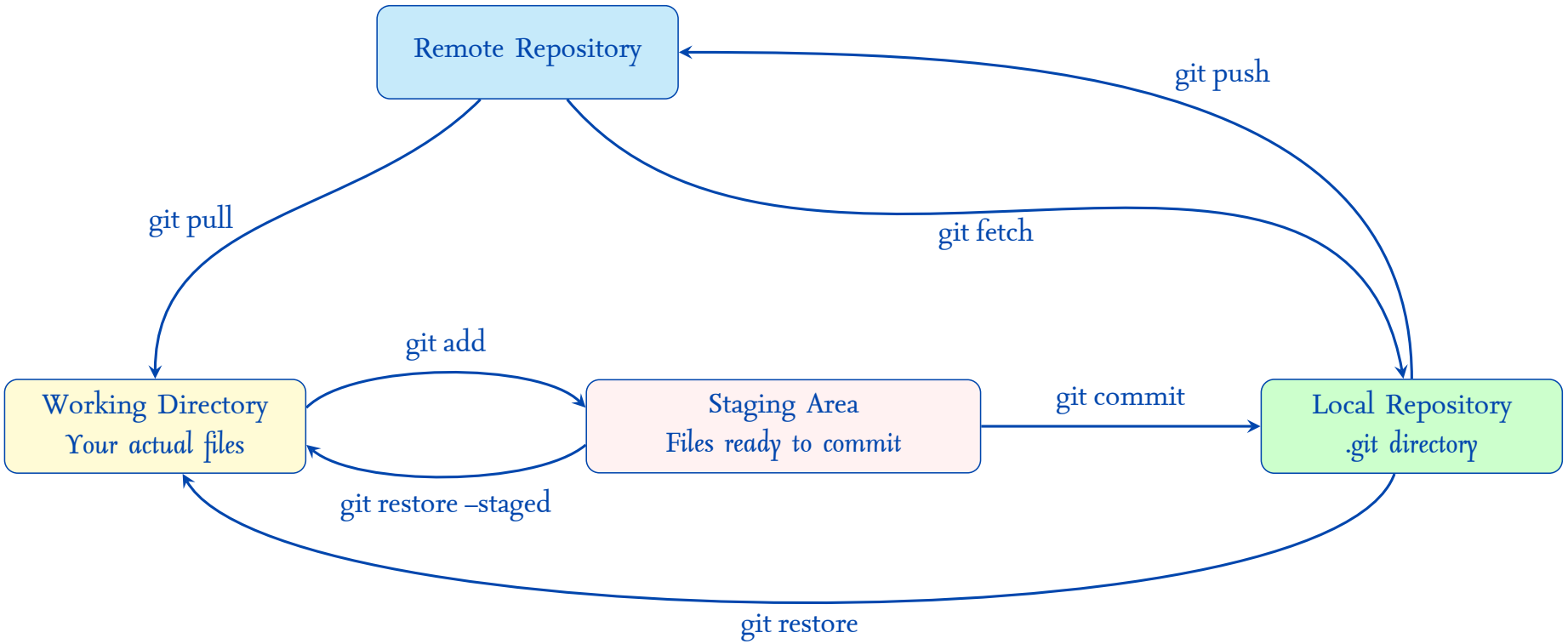
Local and remote repositories

Git concepts to know

- `Working directory`: The files we work on
- `Repository`: Project files + a database with history of all changes (or commits)
  - `Local repository`: A repository located in our computer
  - `Remote repository`: A repository located in a server
- `Staging area`: Intermediary space to prepare changes

Git

Git operations



Git

Git overview (III)

Key Git concepts to know

- `clone`
- `commit`, `push`
- `pull`, `fetch`
- `remote`, `origin`
- `merge`

Using Git

Git basic workflow

Given initialized local and remote repositories

Basic Git workflow

1. Pull changes from the remote
2. Edit your files
3. Add changes to staging area
4. Commit the staged changes
5. Push commits to the remote

Using Git

Initializing a repository

Using plain Git: `Git init`

- Creates a `.git` hidden in the working directory
- Safe operation: All Git data is contained in `.git`
- By default, there are no remote

Using GitHub

1. We create a new repository in GitHub (which will be our remote)
2. Clone the remote repository

Given a GitHub repository

- Clone the remote repository

Using Git

Commits

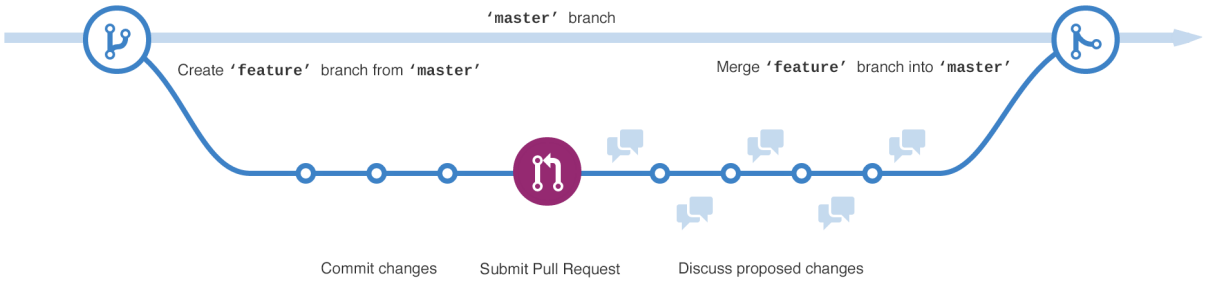
Each commit has ...

- ... an author
- ... a comment: "Fix deprecated py36 black option"
- ... a date
- ... an ID (or *hash*)

```
commit 44161dde6ea234f8cb997644f8e187123c3cc4af
Author: David <foo@foo.com>
Date:   Fri Mar 9 14:57:32 2018 +0100

    Issue with syntax highlighting solved
```

Using Git

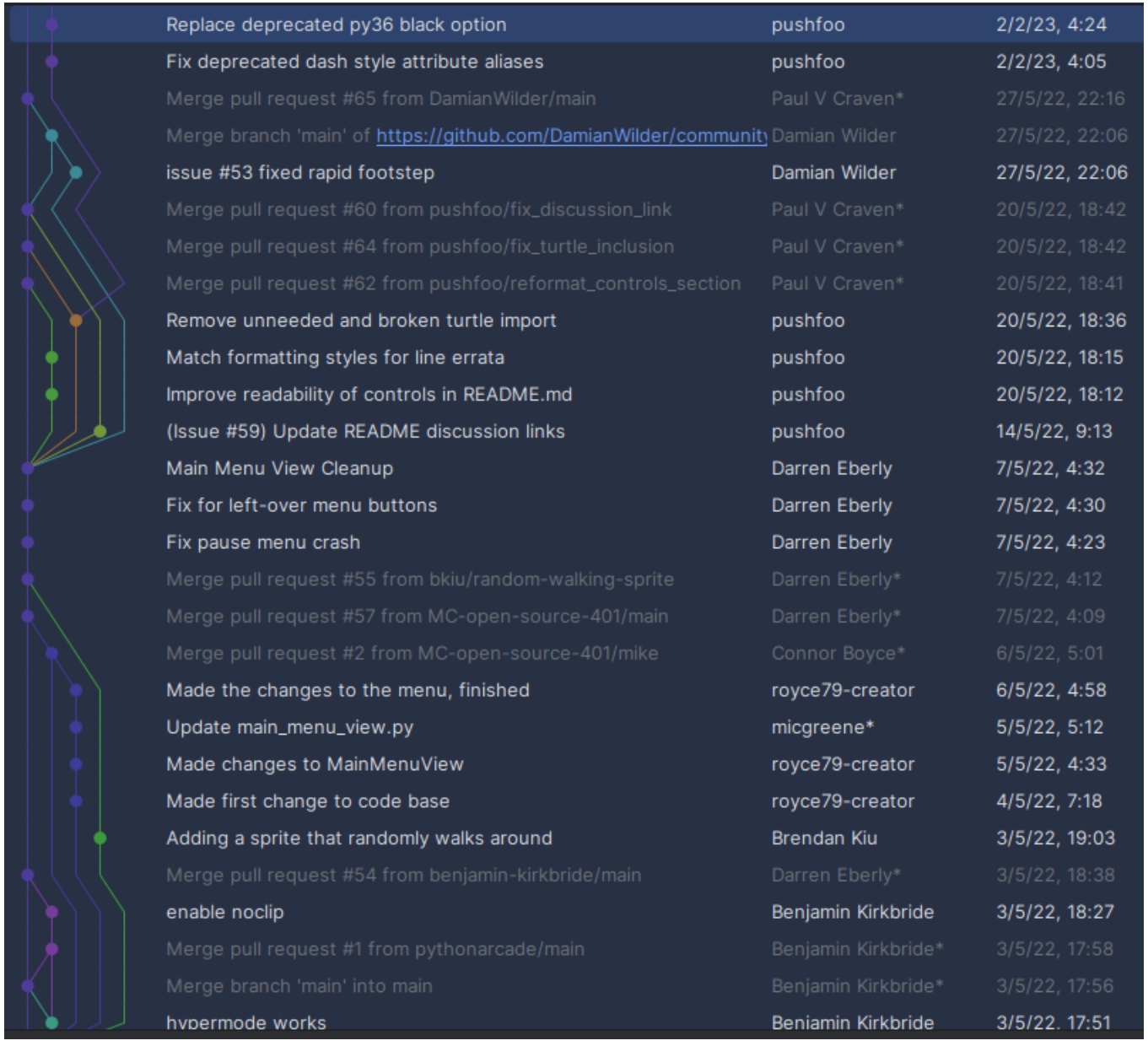
Branches



Branches are used extensively (e.g. some like feature branches).

- A repository (local and remote) can have explicit branches
- The default branch is called master or main
- A merge is a fusion between two branches
- There is a branch with name: HEAD
  - Pointer to the active branch
  - ... sometimes, it may point to a commit, but do not worry about it

Do no use branches in the project!

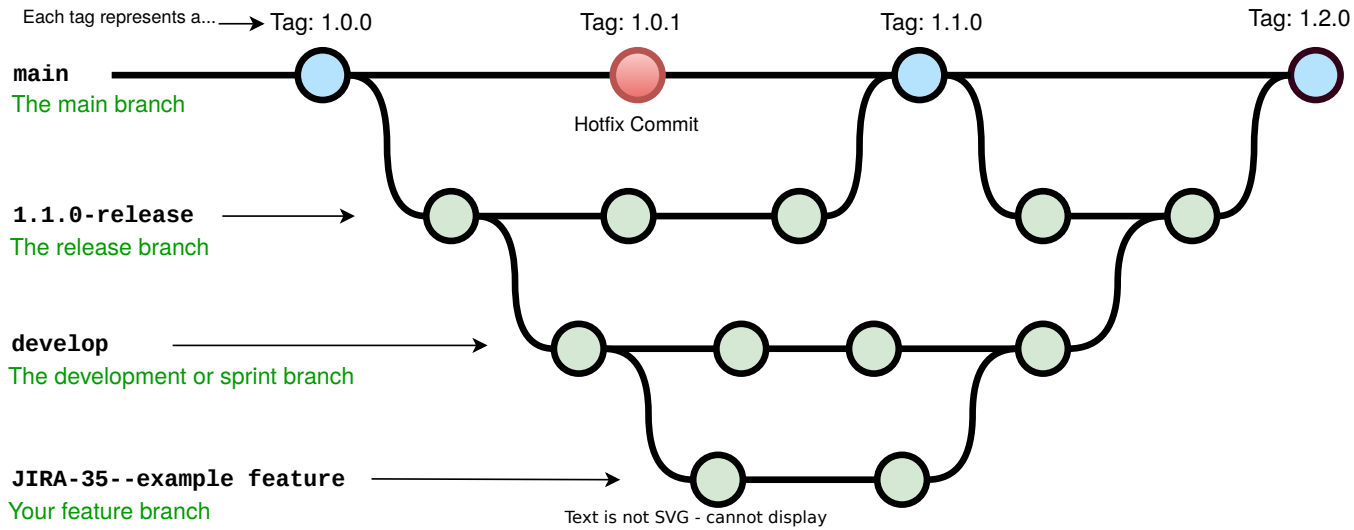

Replace deprecated py36 black option pushfoo 2/2/23, 4:24
Fix deprecated dash style attribute aliases pushfoo 2/2/23, 4:05
Merge pull request #65 from DamianWilder/main Paul V Craven* 27/5/22, 22:16
Merge branch 'main' of https://github.com/DamianWilder/communit Damian Wilder 27/5/22, 22:06
issue #53 fixed rapid footstep Damian Wilder 27/5/22, 22:06
Merge pull request #60 from pushfoo/fix_discussion_link Paul V Craven* 20/5/22, 18:42
Merge pull request #64 from pushfoo/fix_turtle_inclusion Paul V Craven* 20/5/22, 18:42
Merge pull request #62 from pushfoo/reformat_controls_section Paul V Craven* 20/5/22, 18:41
Remove unneeded and broken turtle import pushfoo 20/5/22, 18:36
Match formatting styles for line errata pushfoo 20/5/22, 18:15
Improve readability of controls in README.md pushfoo 20/5/22, 18:12
(Issue #59) Update README discussion links pushfoo 14/5/22, 9:13
Main Menu View Cleanup Darren Eberly 7/5/22, 4:32
Fix for left-over menu buttons Darren Eberly 7/5/22, 4:30
Fix pause menu crash Darren Eberly 7/5/22, 4:23
Merge pull request #55 from bkiu/random-walking-sprite Darren Eberly* 7/5/22, 4:12
Merge pull request #57 from MC-open-source-401/main Darren Eberly* 7/5/22, 4:09
Merge pull request #2 from MC-open-source-401/mike Connor Boyce* 6/5/22, 5:01
Made the changes to the menu, finished royce79-creator 6/5/22, 4:58
Update main_menu_view.py micgreene* 5/5/22, 5:12
Made changes to MainMenuView royce79-creator 5/5/22, 4:33
Made first change to code base royce79-creator 4/5/22, 7:18
Adding a sprite that randomly walks around Brendan Kiu 3/5/22, 19:03
Merge pull request #54 from benjamin-kirkbride/main Darren Eberly* 3/5/22, 18:38
enable noclip Benjamin Kirkbride 3/5/22, 18:27
Merge pull request #1 from pythonarcade/main Benjamin Kirkbride* 3/5/22, 17:58
Merge branch 'main' into main Benjamin Kirkbride* 3/5/22, 17:56
hypermode works Benjamin Kirkbride 3/5/22, 17:51

Using Git

Tags

Example diagram for a GIT workflow:

See: https://nvie.com/posts/a-successful-git-branching-model/



A tag is a pointer to a specific point in the repository history

- Tags usually have names (e.g. “v1.1”)
- Widely used to keep and publish software releases

Using Git

Conflicts

Merging is quite a common operation in Git

- Changes in different parts of a file are automatically merged

Merging changes in the same part of a file cause conflicts

- Git is no longer able to automatically merge
- Human intervention is required

Conflict resolution

1. Identify conflicted files
2. Open files and choose changes to keep
3. Remove markers
4. Stage resolved files and commit

Merging HEAD and feature

```
def greet():
<<<<<<< HEAD
    return "Hello"
=======
    return "Hola"
>>>>>>> feature
```

Using Git

Good practices

Learn on the job: the best way to learn it is by using it.

Best practices

- Regularly push and pull (at least daily, in general)
- ⇒ Test before pushing! ⇐
- Don't push half-baked changes
- Don't pull if you're in the middle of a task
- Never commit temporal/intermediate files
- Keep commit descriptions short and informative
- *The master must be a clean and functional version of the project*

Remember: Git never overwrites local changes without an explicit order

- ... even with a `git pull`

GitHub

Features

Free Git hosting provider

- Free public repositories

Added value features

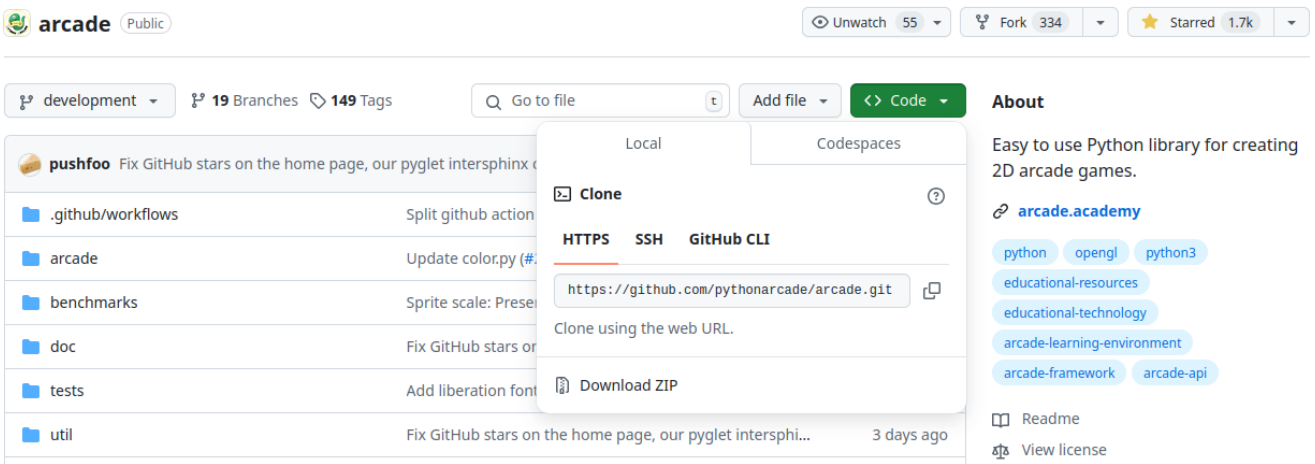
- Social network
- Collaborative tools
- Repository browser
- Pull requests
- Issue tracking
- Web hosting
- Markdown integration
- Organizations

GitHub

Key concepts

Key GitHub concepts to know

- `Pull request`
- `Fork`



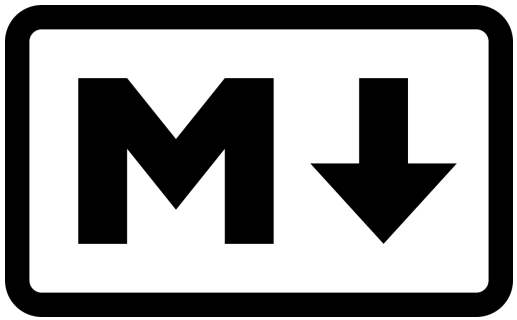
GitHub

README

Special file: README.md

- Contains information about the project
- Automatically visualized
- md means Markdown

Markdown (I)



Markdown: Trivial markup

- Simple
- Very simple
- Extremely simple
- Did I say it's simple?

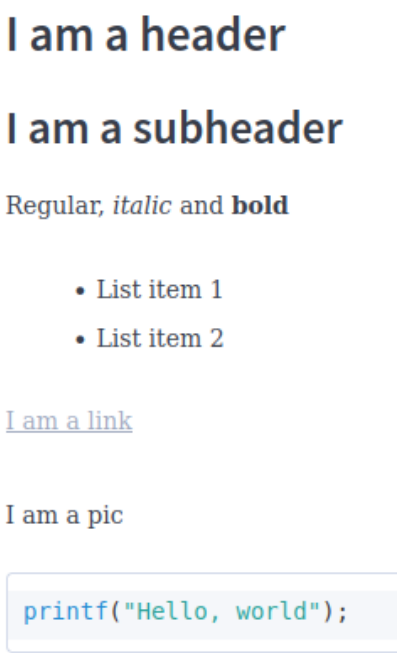
VERY powerful

- Several outputs
- Professional quality
- ... and simple!

Markdown (II)

Markdown example

```
# I am a header
## I am a subheader

Regular, *italic* and **bold**

- List item 1
- List item 2

[I am a link](http://foo.com)

![I am a pic](markdown.png)

~~~C
printf("Hello, world");
~~~
```



I am a header

I am a subheader

Regular, *italic* and **bold**

- List item 1
- List item 2

I am a link

I am a pic

```
printf("Hello, world");
```