Versioning Systems & Git

Eduard-Gabriel Poesina
Senior Consultant
eduard.poesina@thoughtworks.com

thoughtworks

© 2023 Thoughtworks | Confidential

Versioning Systems

The reasoning behind them

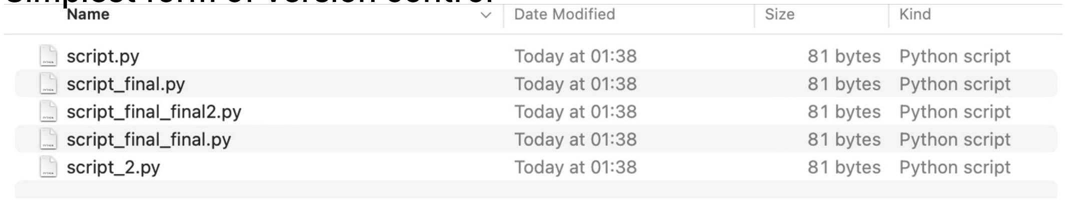
Versioning control systems are software tools that enable developers to track changes to a set of files and folders so that any modification may be recalled;

- Enable collaboration
- Allow Traceability
- Offer backup and recovery
- Branching development

Versioning Systems

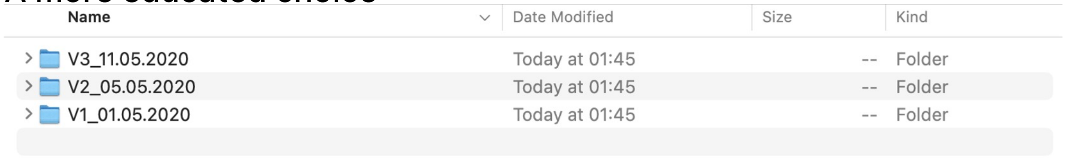
Local Version Control System

Simplest form of version control



Name	Date Modified	Size	Kind
script.py	Today at 01:38	81 bytes	Python script
script_final.py	Today at 01:38	81 bytes	Python script
script_final_final2.py	Today at 01:38	81 bytes	Python script
script_final_final.py	Today at 01:38	81 bytes	Python script
script_2.py	Today at 01:38	81 bytes	Python script

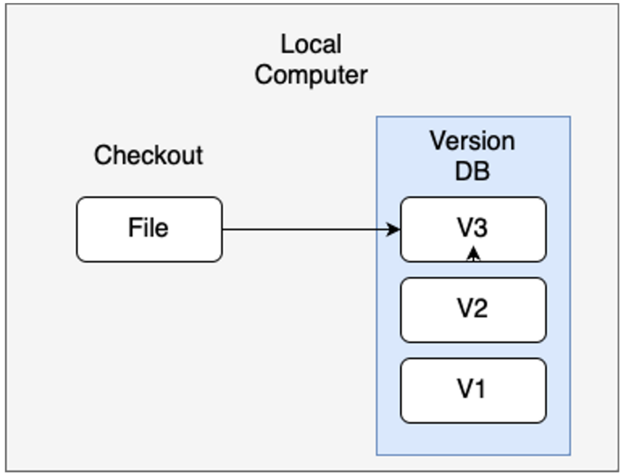
A more educated choice



Name	Date Modified	Size	Kind
V3_11.05.2020	Today at 01:45	--	Folder
V2_05.05.2020	Today at 01:45	--	Folder
V1_01.05.2020	Today at 01:45	--	Folder

Versioning Systems

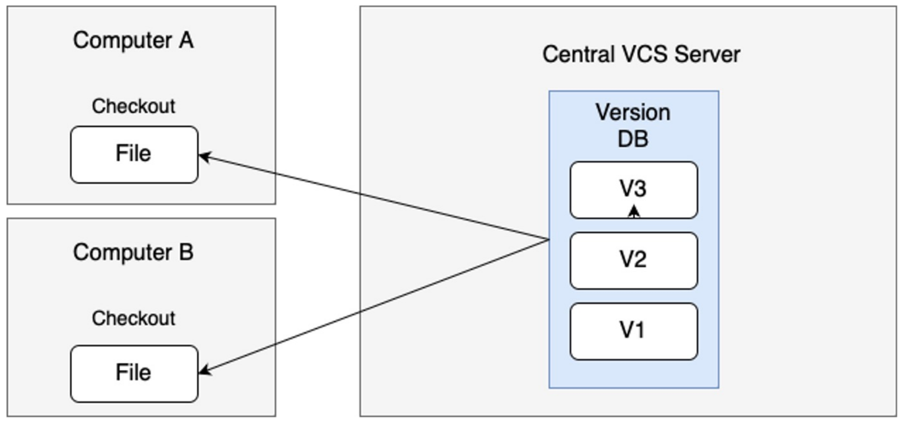
Local Version Control System



RCS (Revision Control System)
https://www.gnu.org/software/rcs

Versioning Systems

Centralised VCS

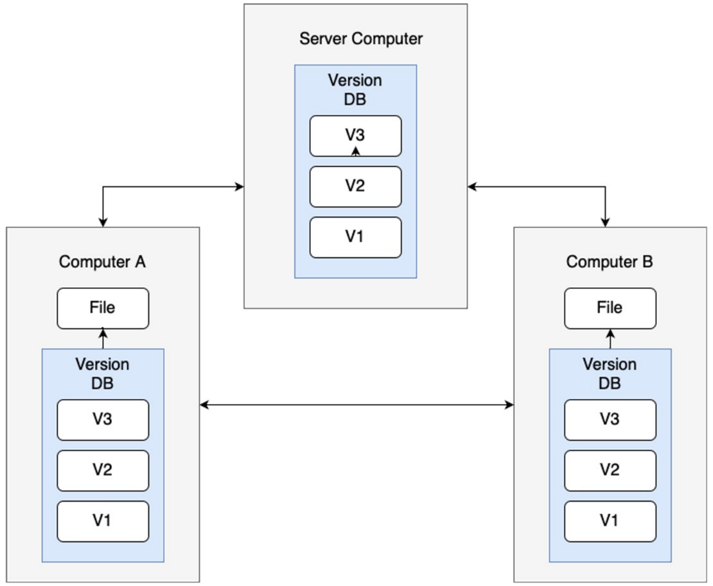


Subversion
https://subversion.apache.com

Perforce
https://www.perforce.com

Versioning Systems

Decentralised VCS



Git
https://git-scm.com

Mercurial
https://www.mercurial-scm.org

© 2023 Thoughtworks | Confidential

Short History of Git

thoughtworks

Git

Started from Linux kernel

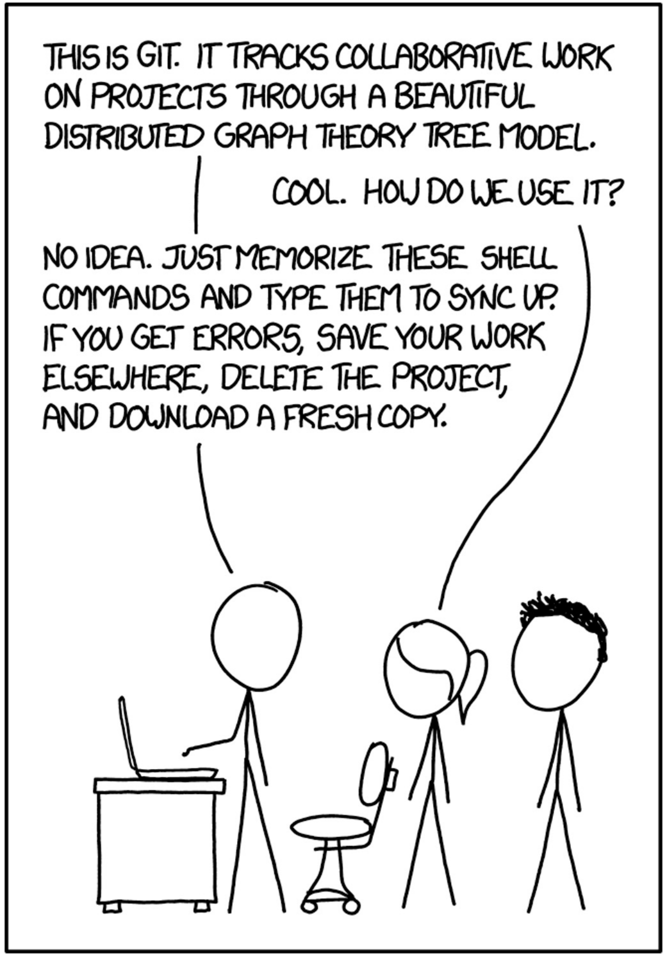
Linux kernel initially used BitKeeper, a proprietary VCS solution. But once the community decided it's time to depart from it, they started to build Git.

- Incredible speed
- Simple Design
- Fully Distributed, Non Linear Development
- Ability to store large projects efficiently

Git

Started from Linux kernel

Linux kernel initially used BitKeeper, a proprietary VCS solution. But once the community decided it's time to depart from it, they started to build Git.

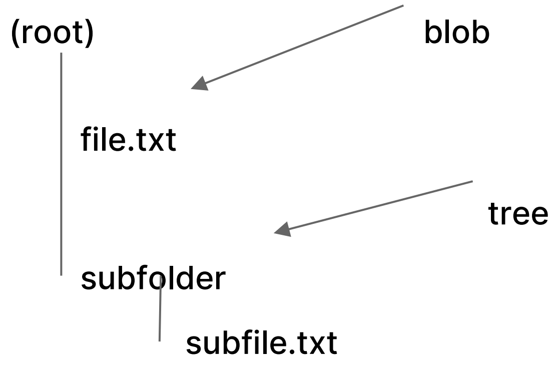

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

https://xkcd.com/1597/

The Basics of Git

Git

Git data structure



File/folder naming

Git

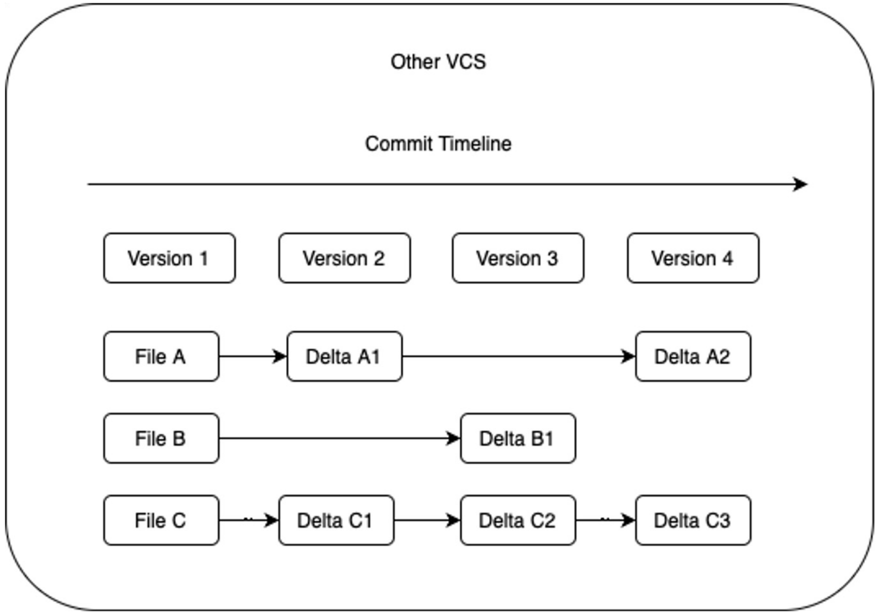
Git data structure

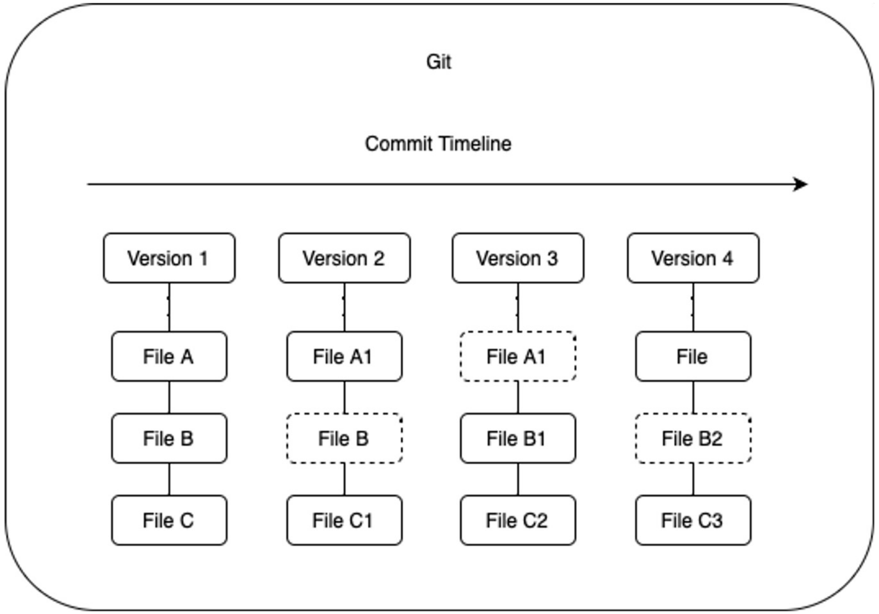
```
type blob = array<byte>

type tree = map<string, tree | blob >

type commit = struct {
    parents: array<commit>,
    author: string,
    message: string,
    snapshot: tree
}
```

```
type object = blob | tree | commit

objects = map<string,object>

def store(o):
    id = sha1(o)
    objects[id] = o

def load(id):
    return objects[id]
```

Git

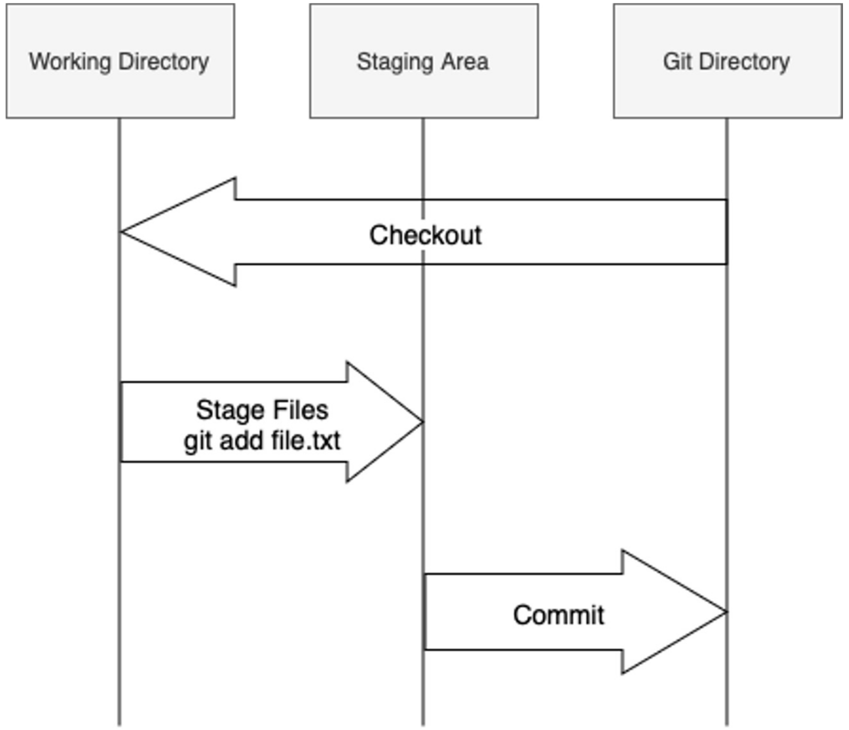
Snapshots rather than Deltas





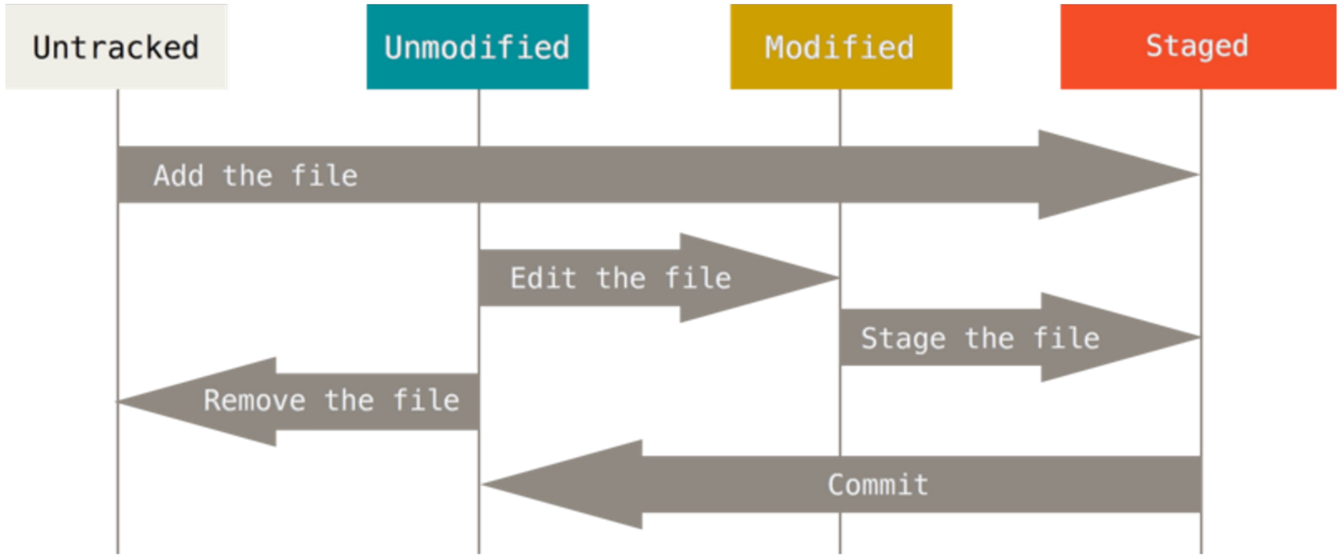
Git

Three main states



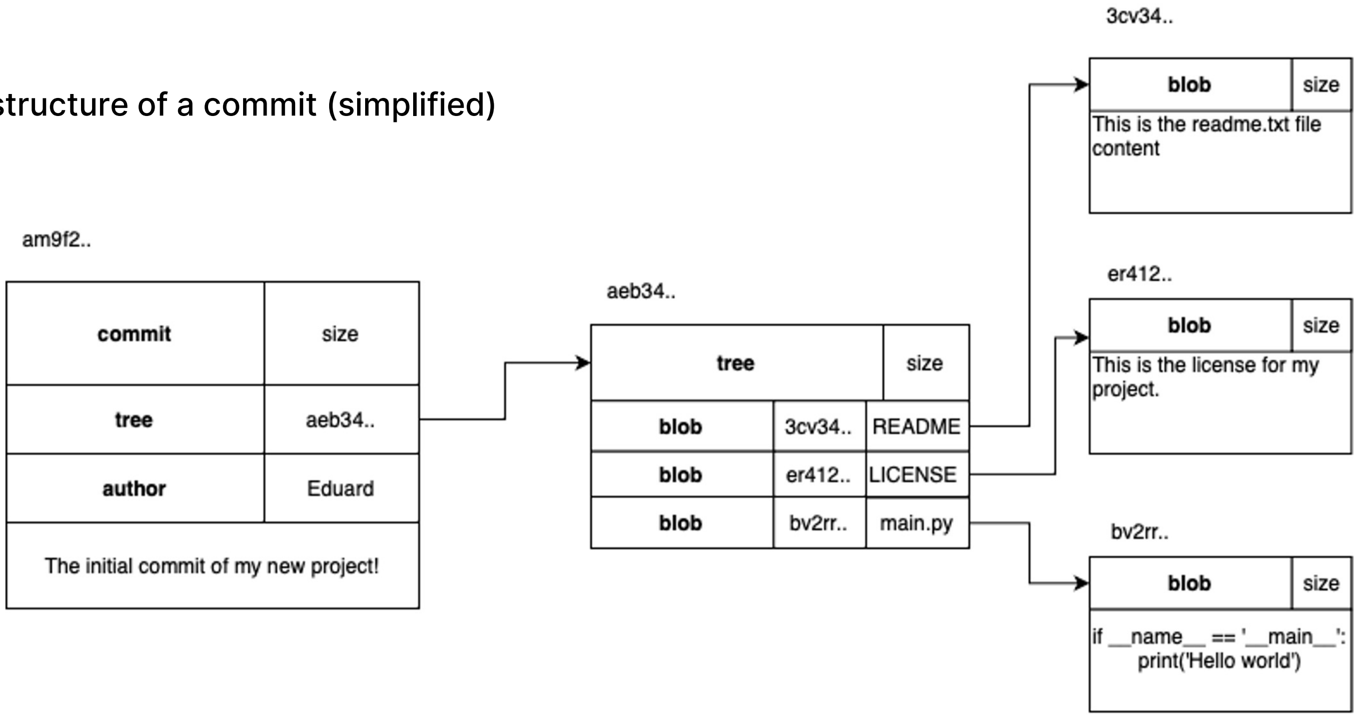
Git

File lifecycle



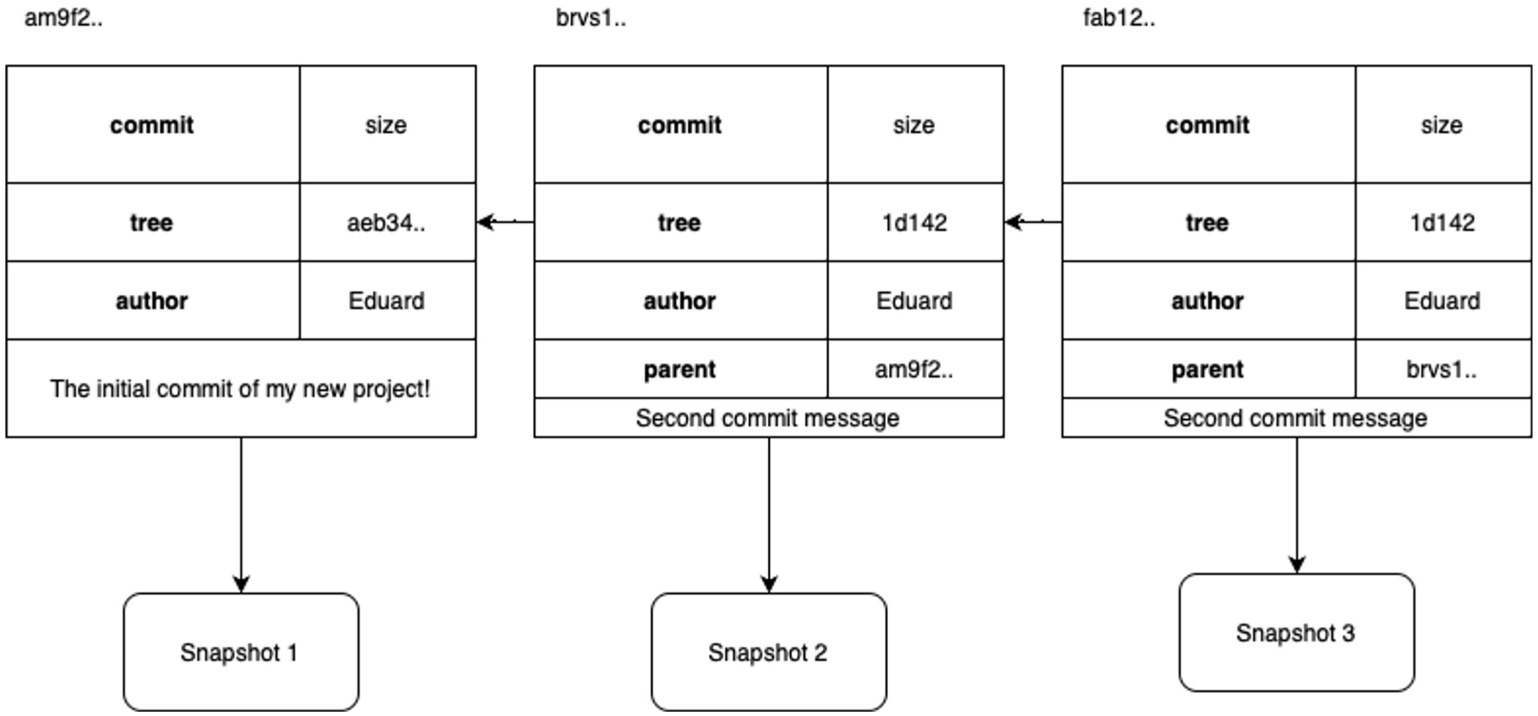
Git

Internal structure of a commit (simplified)



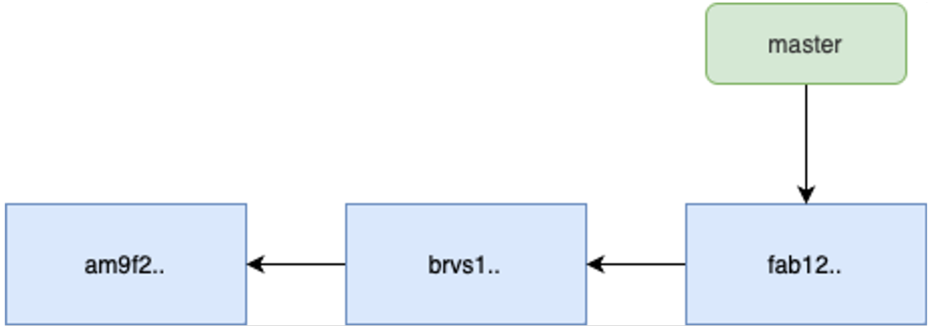
Git

Git DAG


am9f2..
commit
size
tree
aeb34..
author
Eduard
The initial commit of my new project!
brvs1..
commit
size
tree
1d142
author
Eduard
parent
am9f2..
Second commit message
fab12..
commit
size
tree
1d142
author
Eduard
parent
brvs1..
Second commit message
Snapshot 1
Snapshot 2
Snapshot 3

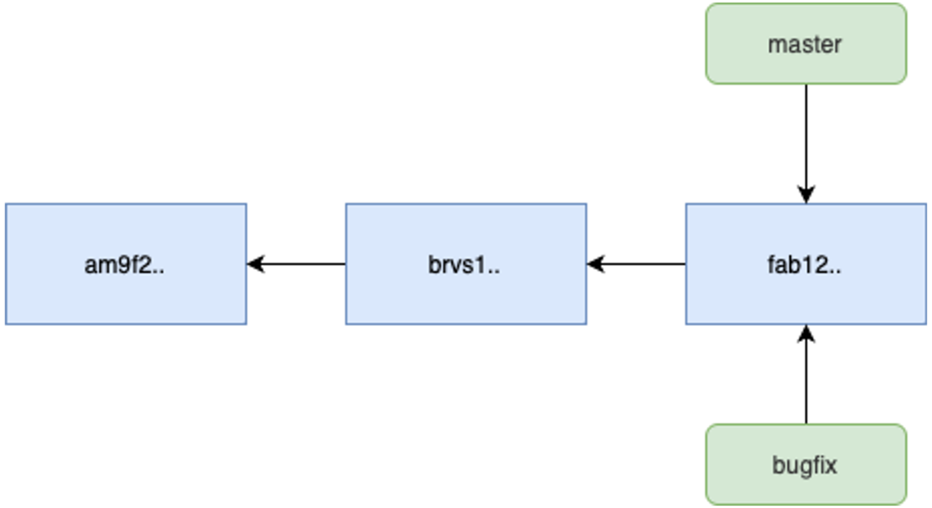
Git

Git Branch



Git

Git Branch

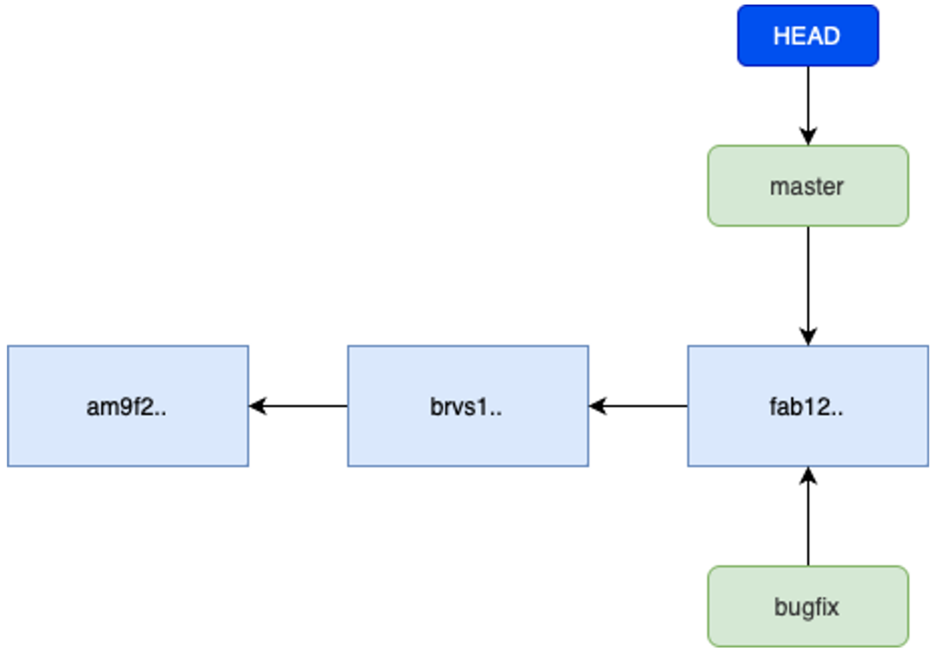


© 2023 Thoughtworks | Confidential

Git

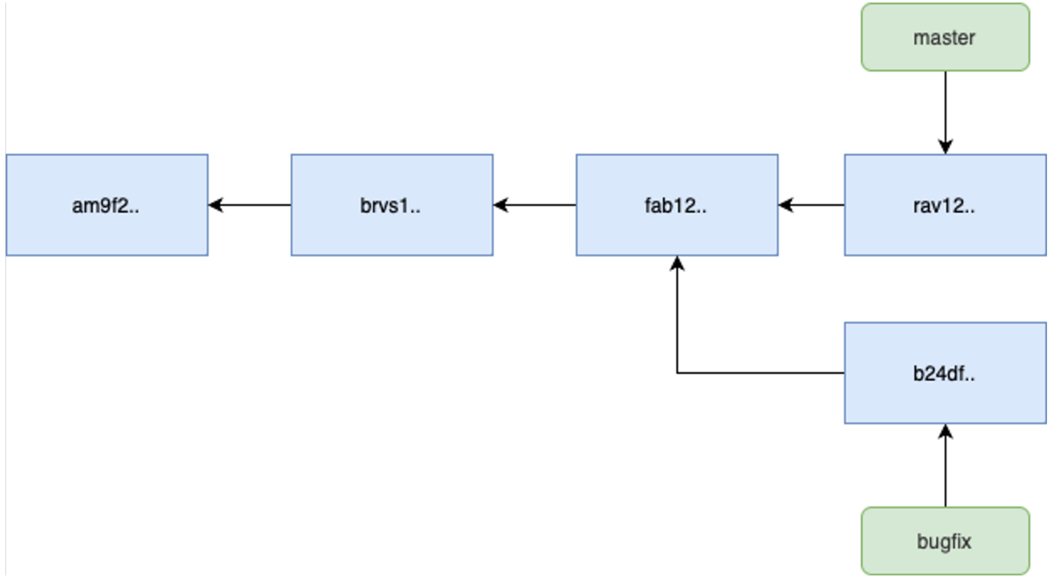
Git Branch

Git uses a HEAD pointer to point to the current branch



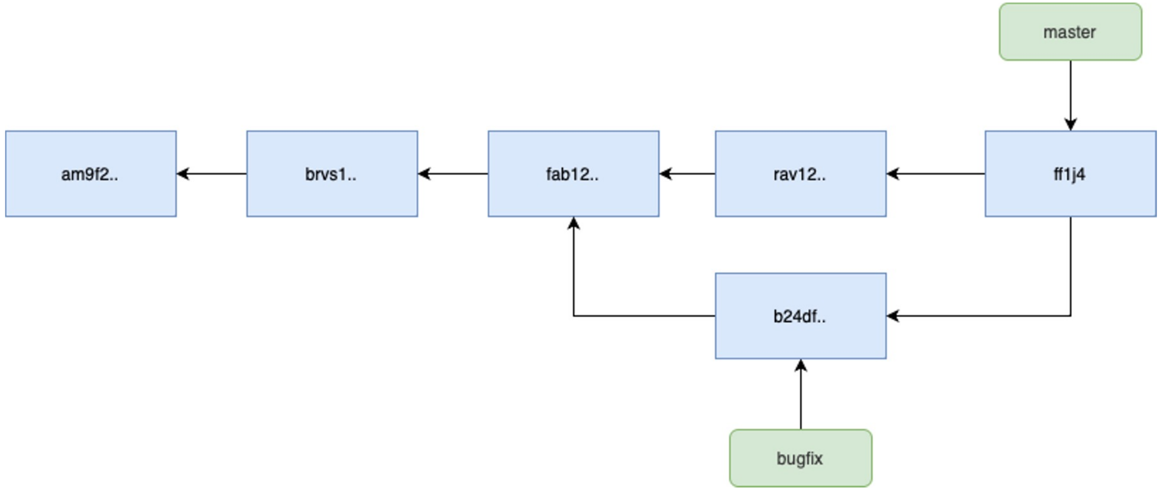
Git

Git Branch



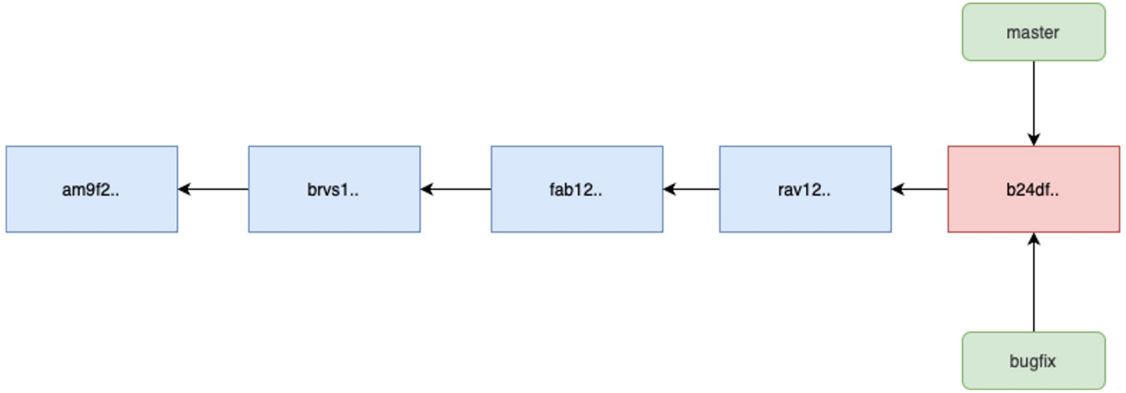
Git

Git Merge



Git

Git Rebase



Github
LIVE DEMO

thoughtworks

© 2023 Thoughtworks | Confidential

Thank you!

Eduard-Gabriel Poesina
Senior Consultant
eduard.poesina@thoughtworks.com

/thoughtworks