alpha

Git: Let's f* up history, and then restore it

Or: How to get rid of my_git_project_bkp (copy1-10)

A small introduction into some more advanced features of Git

Why use version control as a hobbyist?

- Manage progress
  - Revert changes
  - Test new stuff in separate branches (Organise branches)
- Project history
  - Understand why I changed something some years ago
  - Backtrack when some functionality broke

GIT - the stupid content tracker

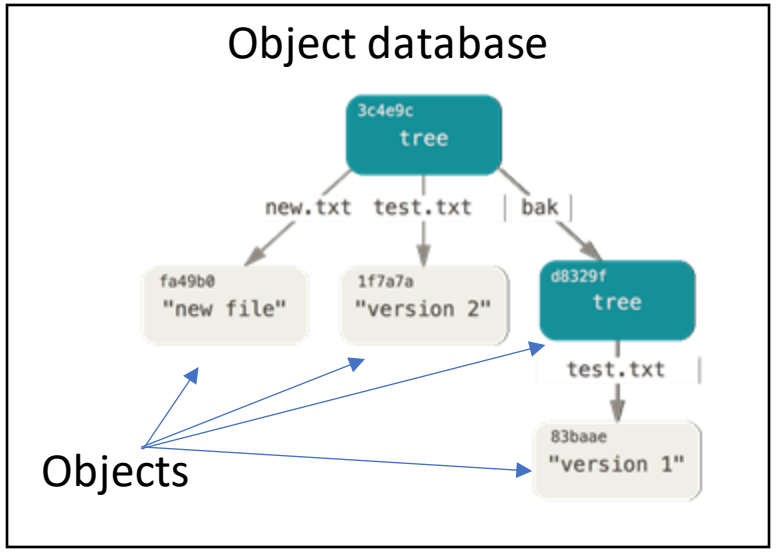
According to its README

> "git" can mean anything, depending on your mood.
>
> - random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronounciation of "get" may or may not be relevant.
> - stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
> - "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
> - "goddamn idiotic truckload of sh*t": when it breaks

GIT – Under the hood

According to its README

> This is a stupid (but extremely fast) directory content manager. It doesn't do a whole lot, but what it **does** do is track directory contents efficiently.
>
> There are two object abstractions: the "object database", and the "current directory cache".



GIT command overview

There are continuously more options and commands added so best check *git command –help from time to time*.

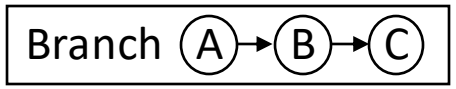
```
NAME
       git-cat-file - Provide content or type and size information for repository objects

SYNOPSIS
       git cat-file <type> <object>
       git cat-file (-e | -p) <object>
       git cat-file (-t | -s) [--allow-unknown-type] <object>
       git cat-file (--batch | --batch-check | --batch-command) [--batch-all-objects]
                    [--buffer] [--follow-symlinks] [--unordered]
                    [--textconv | --filters] [-z]
       git cat-file (--textconv | --filters)
                    [<rev>:<path|tree-ish> | --path=<path|tree-ish> <rev>]
```

GIT command overview

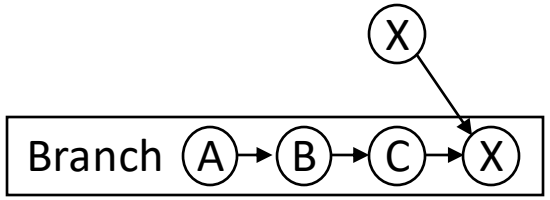
- Basic commands: git ..
  - *add*
    Stage files for commission
  - *commit*
    ommit staged files
  - *push*
    Upload commits to a remote repository
  - *pull*
    Pull all commits of a remote repository into the current branch and merge according to git configuration
  - *checkout*
    Has multiple functions. Create new branches, switch to a branch, revert all unstaged files, etc.

GIT command overview

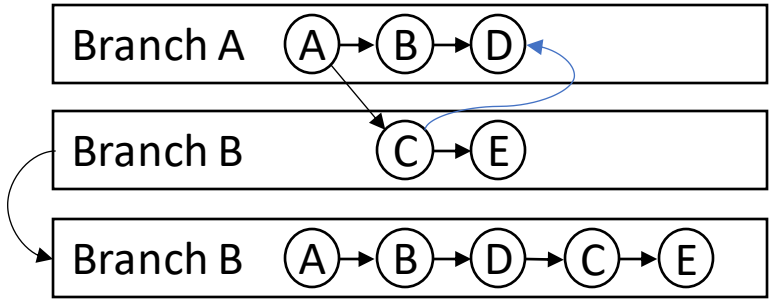
- Working with the commit history: git ..
  - *log*
    List commit history of the currently checked out branch



  - *cherry-pick*
    Pull single commits into the current branch



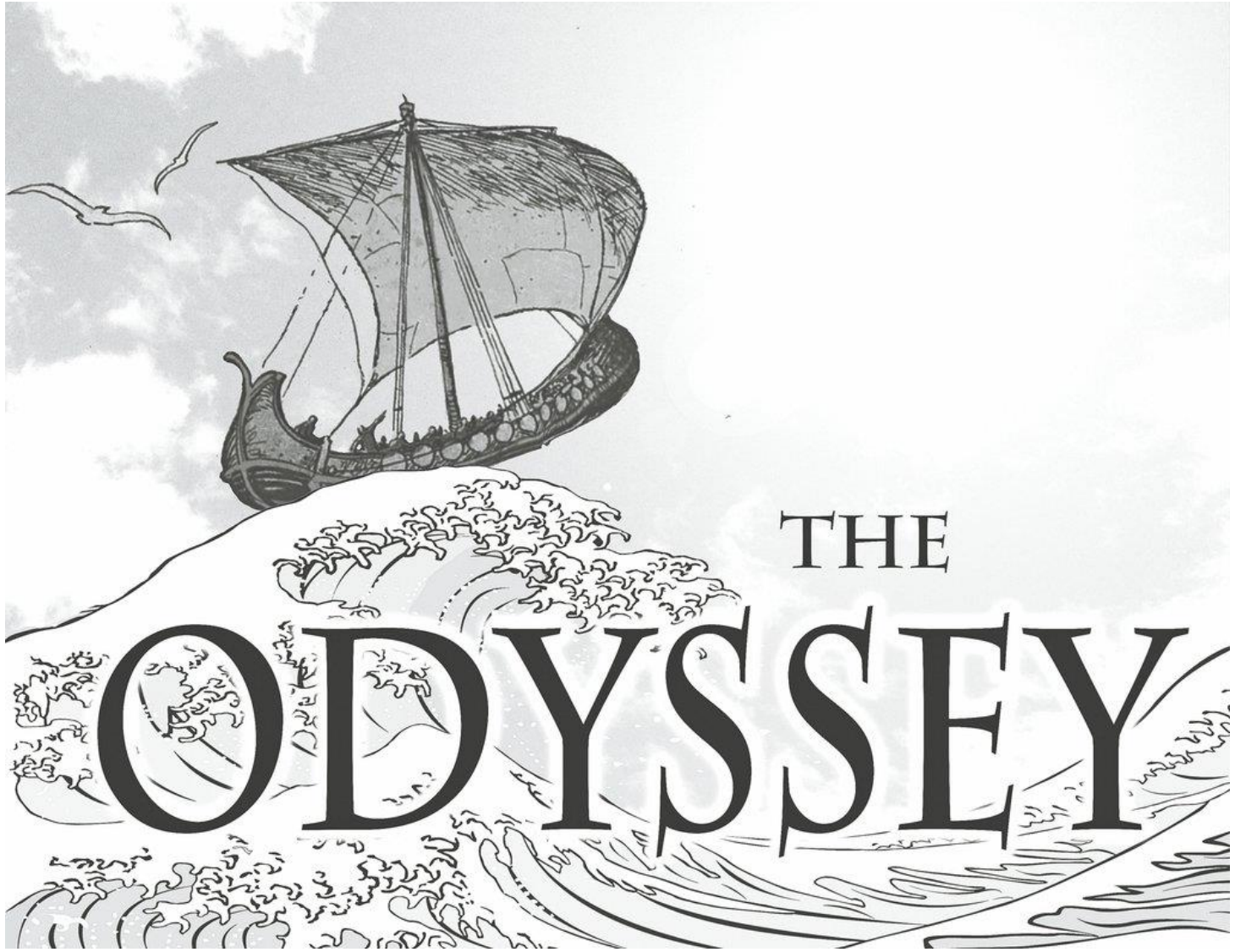
  - *rebase*
    Apply all commits introduced by the current branch on top of another branch, then replace the current branch with the result



GIT command overview

- Lower level commands: git ..
  - *reflog*
    List local "action log" (Current directory cache)
  - *cat-file*
    Cat objects
  - *fsck*
    Inspect the object trees (eg. list dangling (unused) objects)
- Garbage collecting: git ..
  - *prune*
    Deletes dangling object
  - *gc* (garbage collector)
    Simplifies cleanup in general.
- And many, many more

A week with GIT



Goto https://github.com/h2obrain/about/blob/main/odyssey.md

That’s all folks – Thanks for listening

Take aways

- Data can be restored, once it is committed and pushed
  - Do not check in sensitive data
- There are dozens of commands, each having their use case
  - Most of the time, Google what you want to do and you will find it already implemented in one way or another

The presentation can be found here soonish..

https://github.com/h2obrain/about