How git works

Kris.Hermans@pxl.be

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

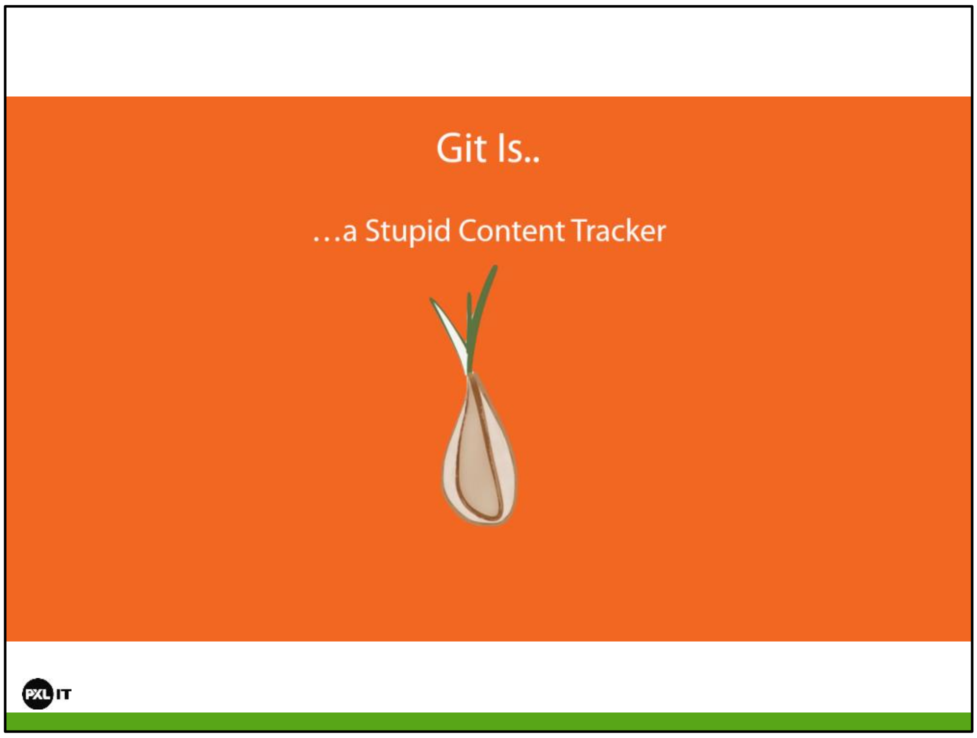
Git Is..

…a Distributed Revision Control System

Git Is..

…a Revision Control System

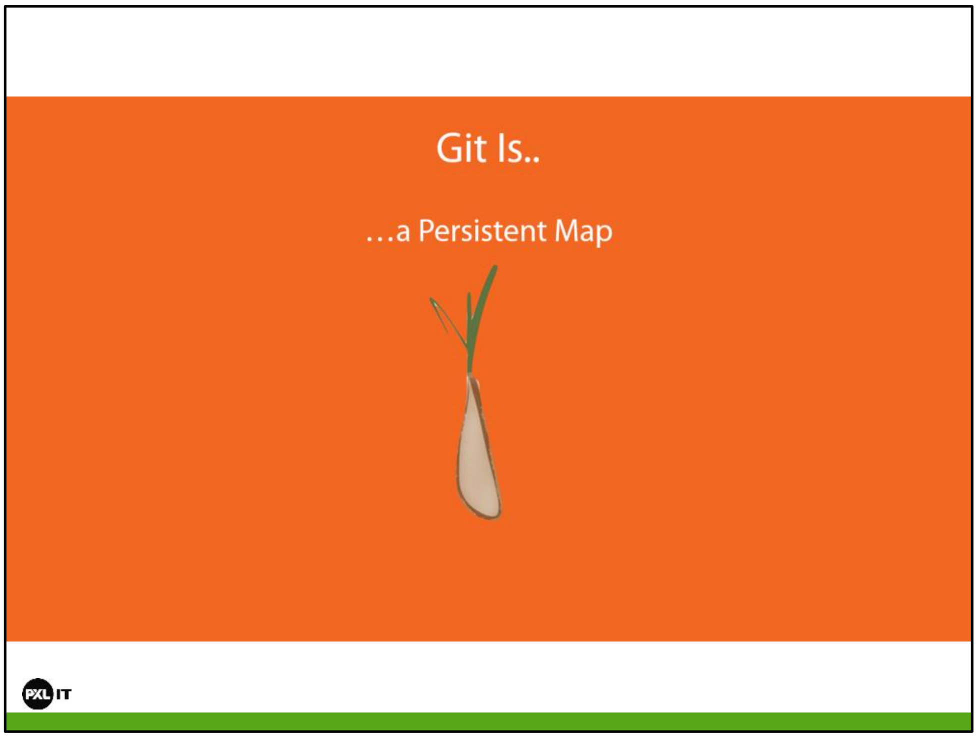
PXL IT



Git Is..

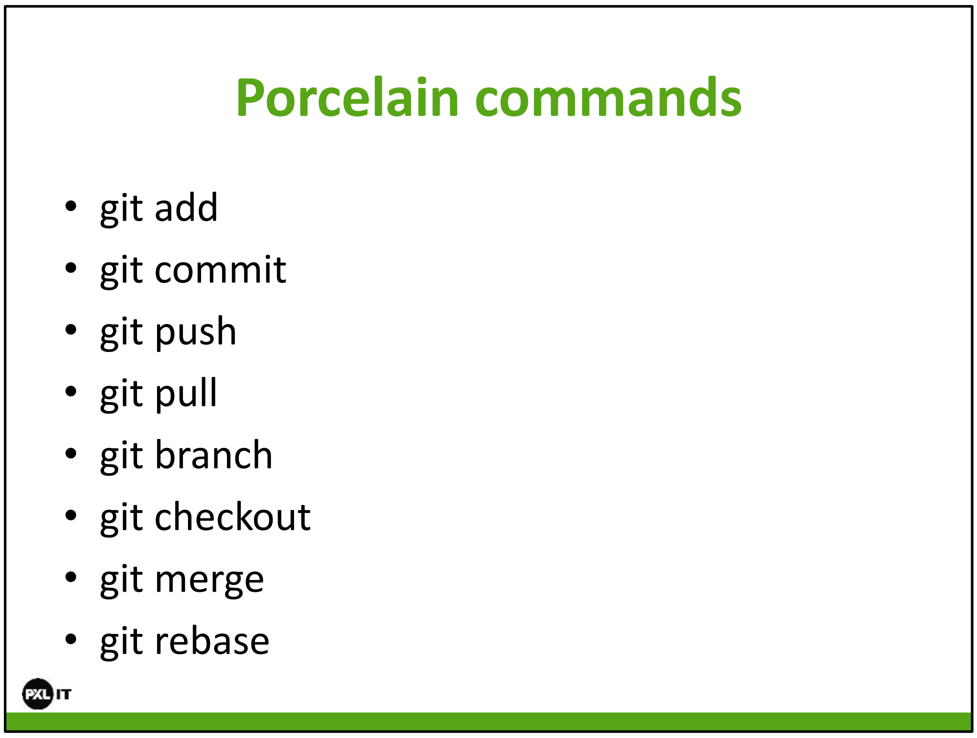
…a Stupid Content Tracker

PXL IT



Git Is..

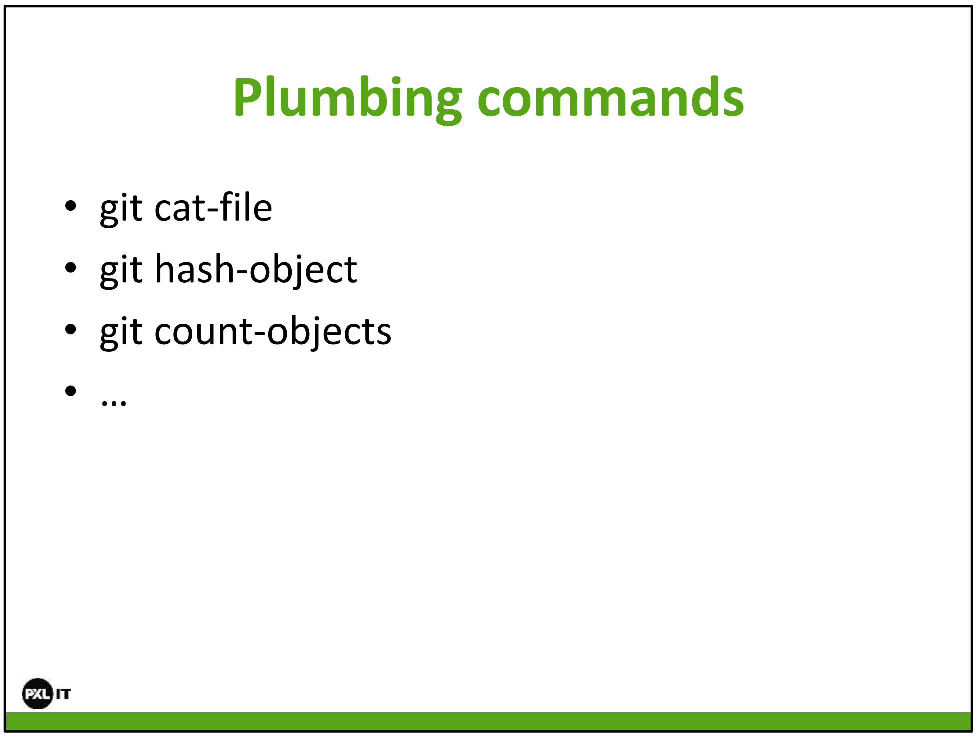
…a Persistent Map



Porcelain commands

- git add
- git commit
- git push
- git pull
- git branch
- git checkout
- git merge
- git rebase

The porcelain commands are the more “user-friendly” command. These commands you use everyday as a developer.

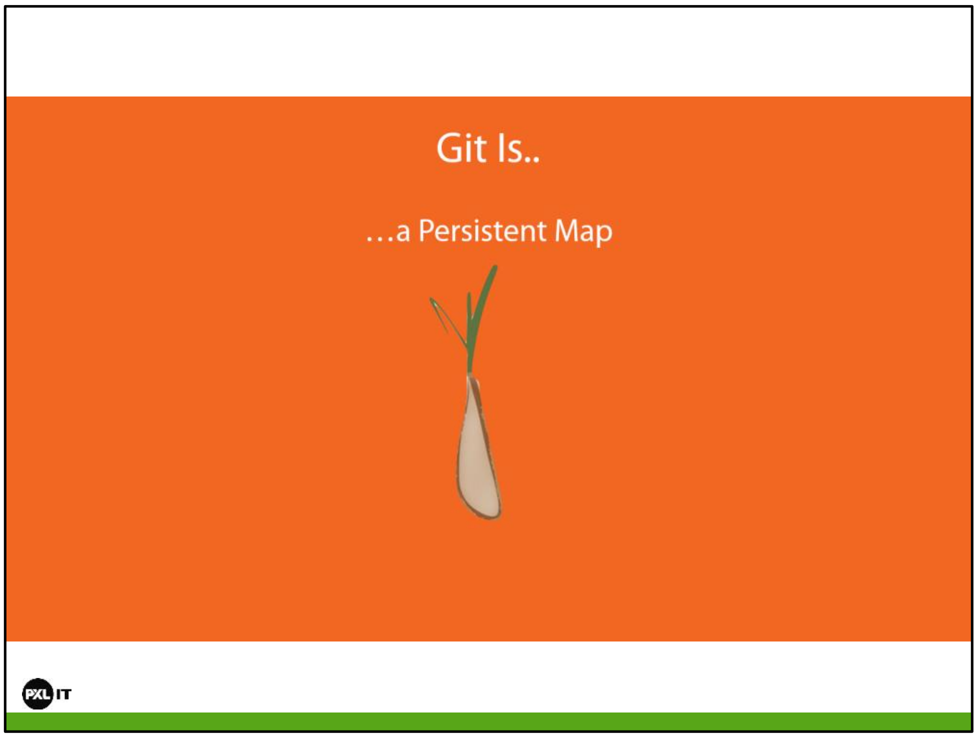


Plumbing commands

- git cat-file
- git hash-object
- git count-objects
- …

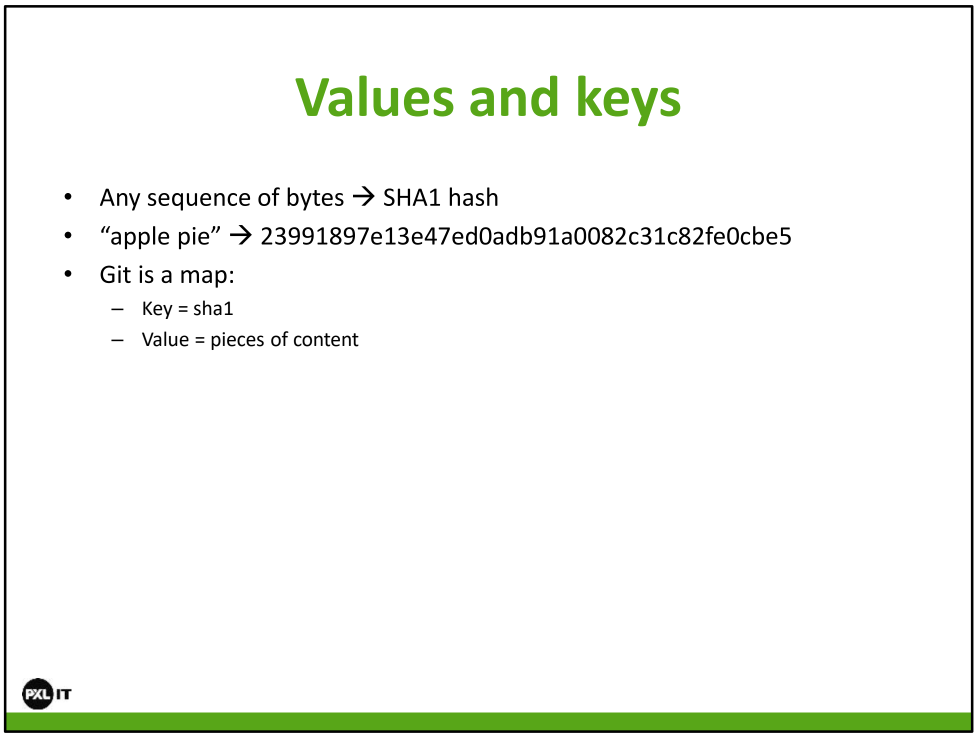
These commands you normally don't use, but the porcelain commands are built from them.

GIT: LEARN ABOUT THE UNDERLYING MODEL



Git Is..

…a Persistent Map



Values and keys

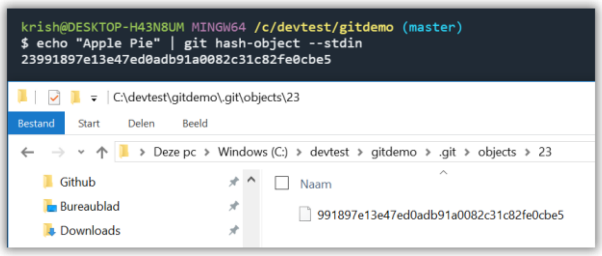
- Any sequence of bytes → SHA1 hash
- “apple pie” → 23991897e13e47ed0adb91a0082c31c82fe0cbe5
- Git is a map:
  - Key = sha1
  - Value = pieces of content

Try this out in a git bash terminal:

```
$ echo "Apple Pie" | git hash-object --stdin
23991897e13e47ed0adb91a0082c31c82fe0cbe5
```

Persistence

- git hash-object -w → stores into a blob
  - A blob is a piece of content
- You must have a repository (.git) → git init



```
krish@DESKTOP-H43N8UM MINGW64 /c/devtest/gitdemo (master)
$ echo "Apple Pie" | git hash-object --stdin
23991897e13e47ed0adb91a0082c31c82fe0cbe5
```

C:\devtest\gitdemo\.git\objects\23

Bestand
Start
Delen
Beeld
Deze pc › Windows (C:) › devtest › gitdemo › .git › objects › 23
Github
Bureaublad
Downloads
Naam

991897e13e47ed0adb91a0082c31c82fe0cbe5

- git cat-file → peek inside an object

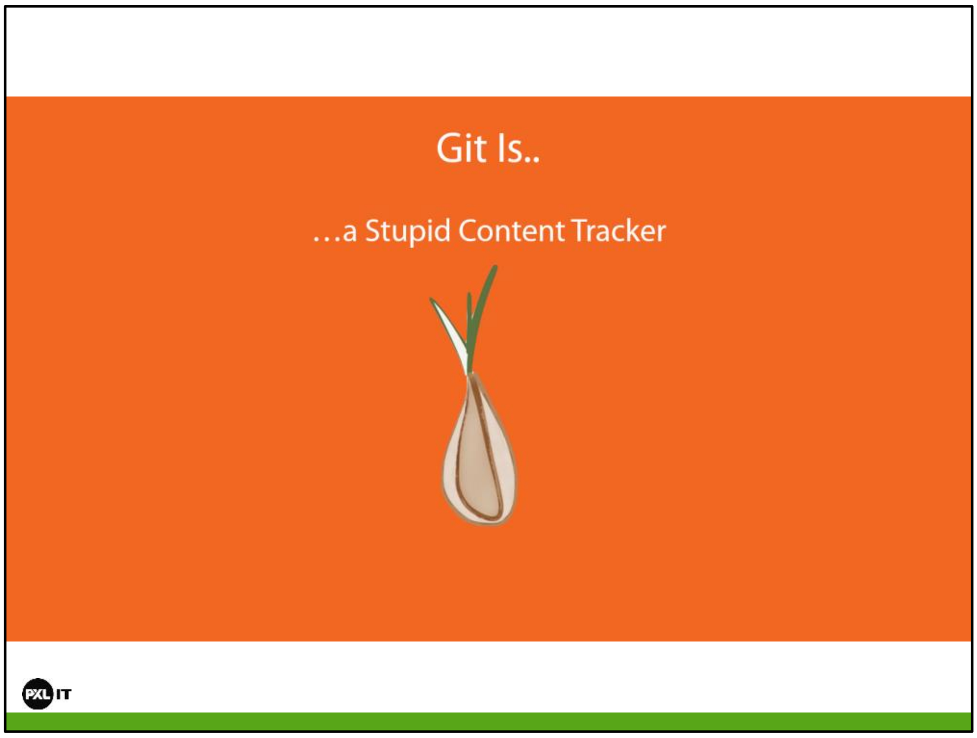
Try this out in a git bash terminal:

$ echo "Apple Pie" | git hash-object --stdin **-w**
23991897e13e47ed0adb91a0082c31c82fe0cbe5

Then explore the .git folder

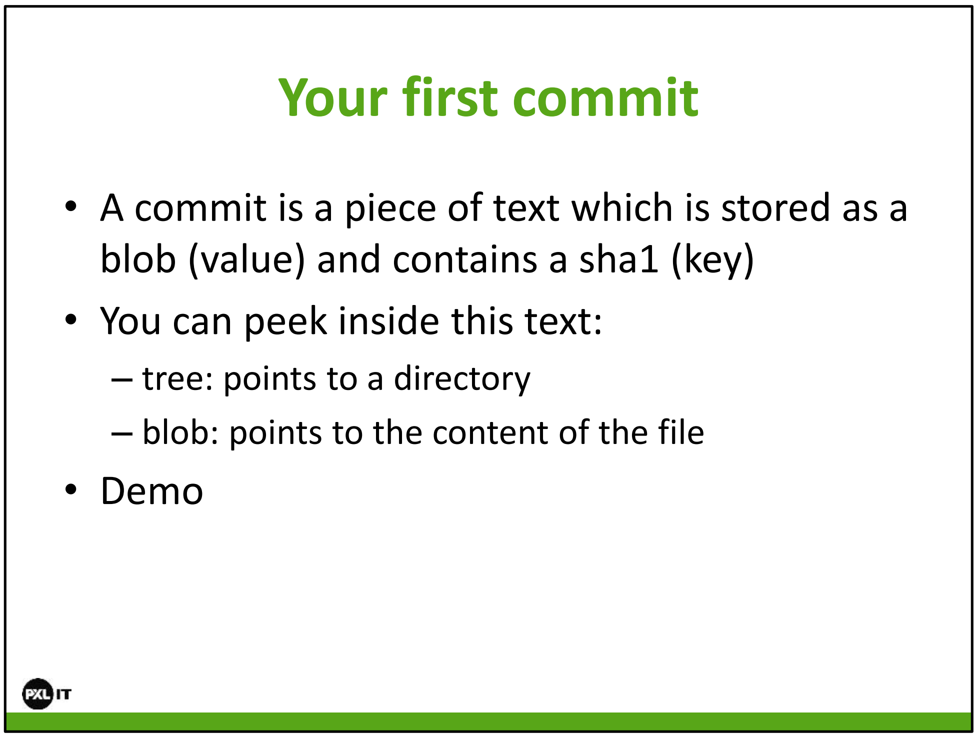
$ git cat-file 23991897e13e47ed0adb91a0082c31c82fe0cbe5 –t
blob

$ git cat-file 23991897e13e47ed0adb91a0082c31c82fe0cbe5 –p
Apple Pie



Git Is..

…a Stupid Content Tracker



Your first commit

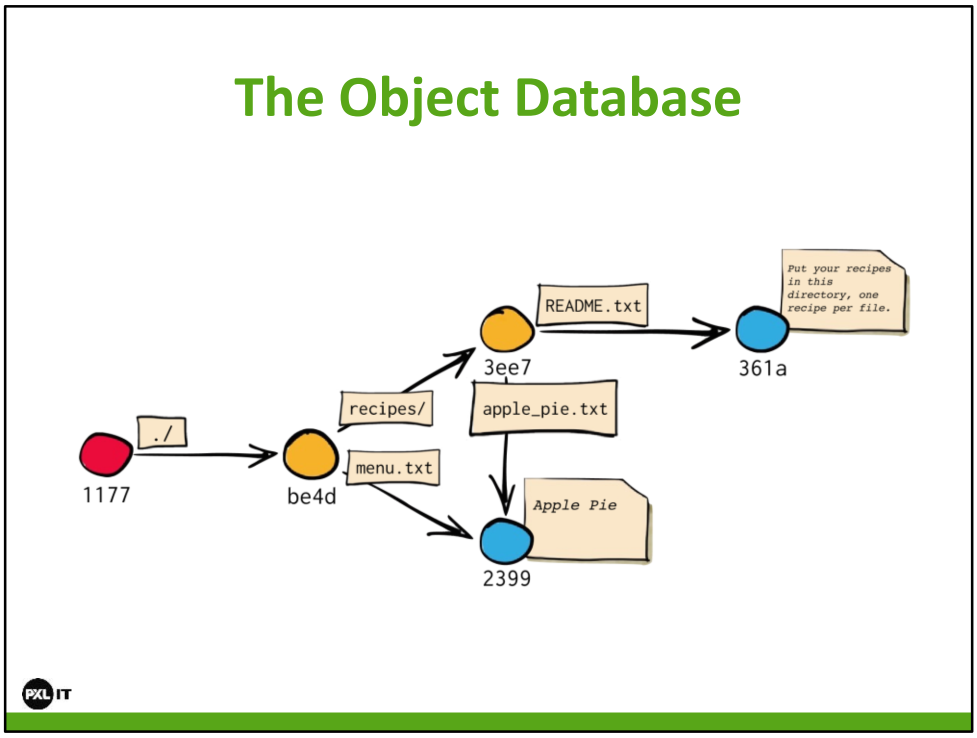
- A commit is a piece of text which is stored as a blob (value) and contains a sha1 (key)
- You can peek inside this text:
  - tree: points to a directory
  - blob: points to the content of the file
- Demo

PXL IT

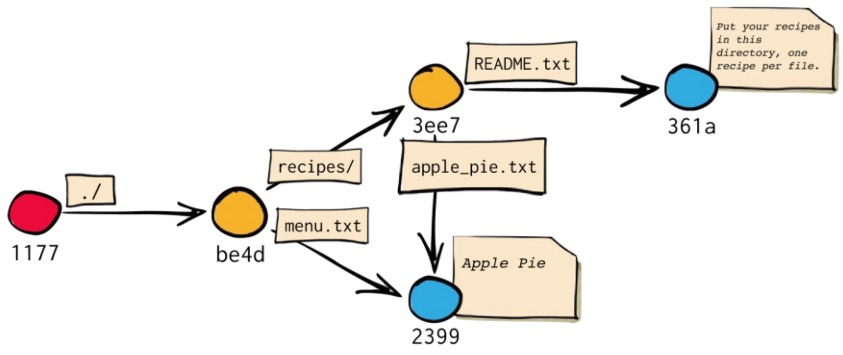
Note: a blob does not point to the file itself! File permissions etc are stored in the tree.

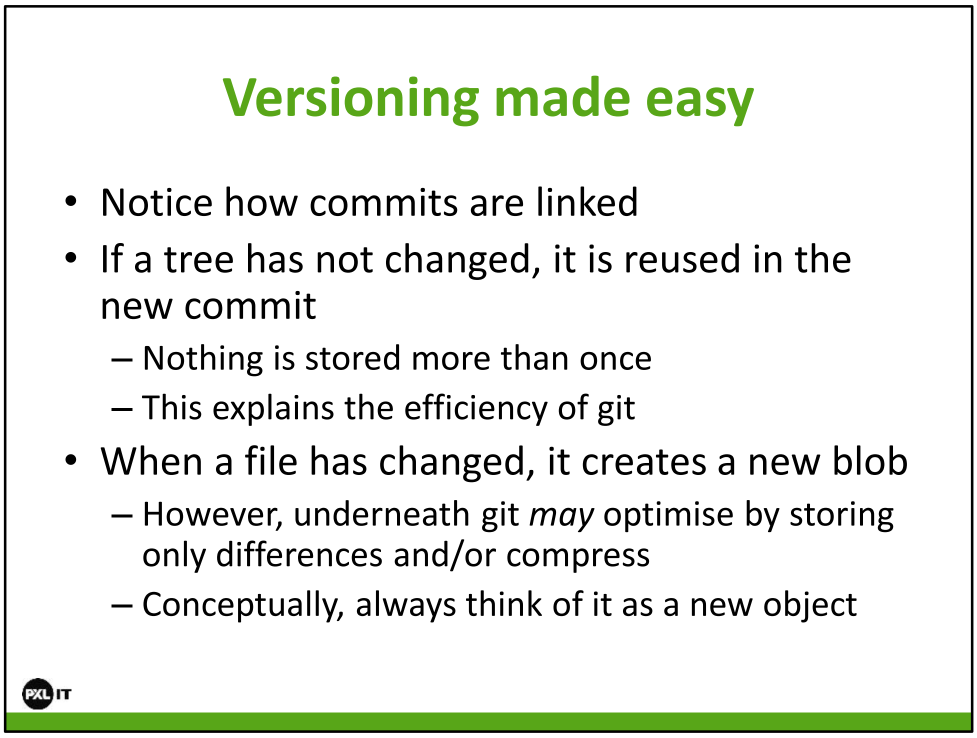
Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m1&clip=4&mode=live

Steps:
- Create an empty folder cookbook and do git init
- Add all files from the demo
- Inspect .git/objects folder



The Object Database





Versioning made easy

- Notice how commits are linked
- If a tree has not changed, it is reused in the new commit
  - Nothing is stored more than once
  - This explains the efficiency of git
- When a file has changed, it creates a new blob
  - However, underneath git *may* optimise by storing only differences and/or compress
  - Conceptually, always think of it as a new object

Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m1&clip=5&mode=live

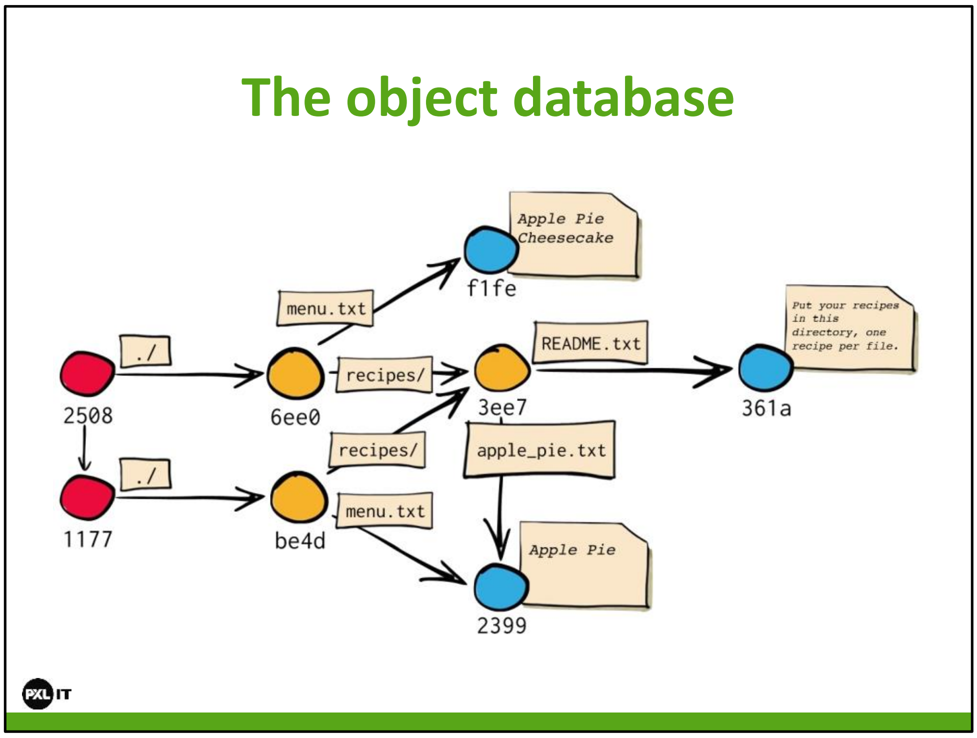
Steps:

- Add a line (Cheesecake) to menu.txt
- Commit this change ("Add cake")
- git cat-file to watch the second commit: it is linked to its parent (the first commit)
- Draw the object model and notice how nothing is stored more than once

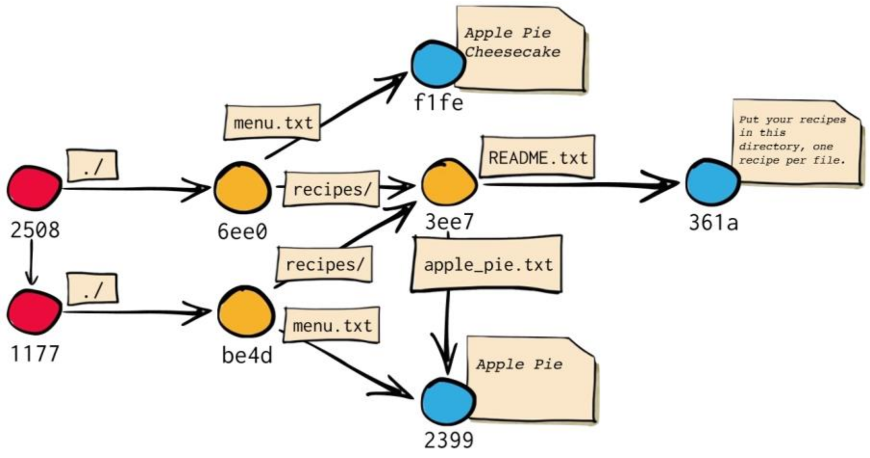
git count-objects -H to count the number of objects

Question: does git always create a new object, even when you change a single line from a (large) file?

No, but from a logical point of view (the plumbing commands), it is a new object with a new SHA1. Underneath is a seperate layer that stores only differences to the file when needed (this is the purpose of the pack and info folders).

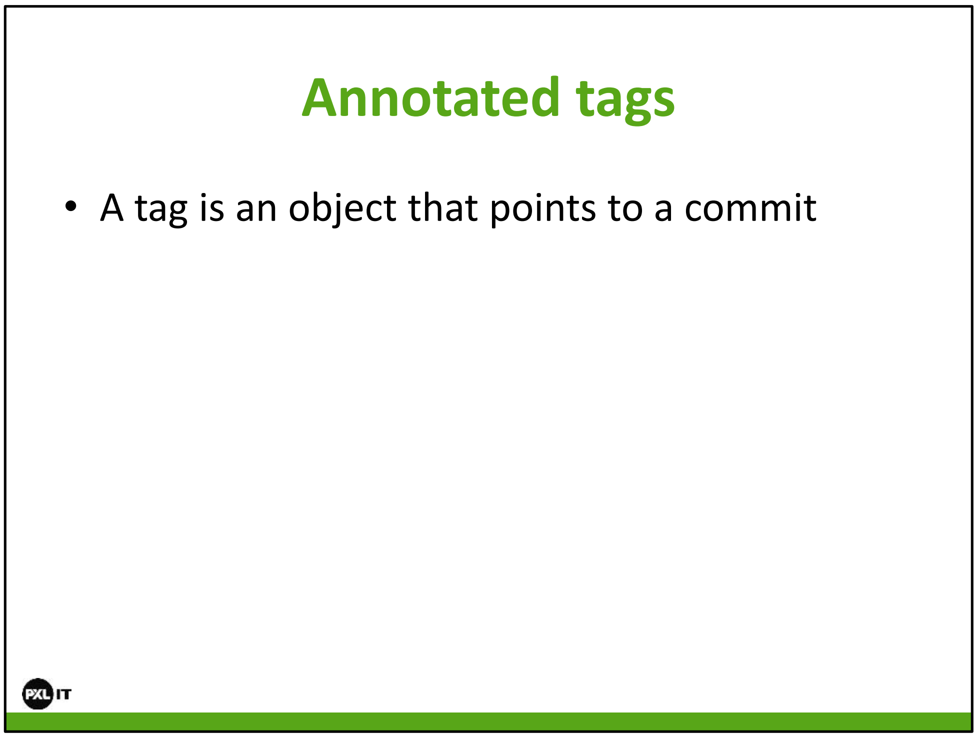


The object database



A new commit points to it parent

Unchainged blobs are reused



Annotated tags

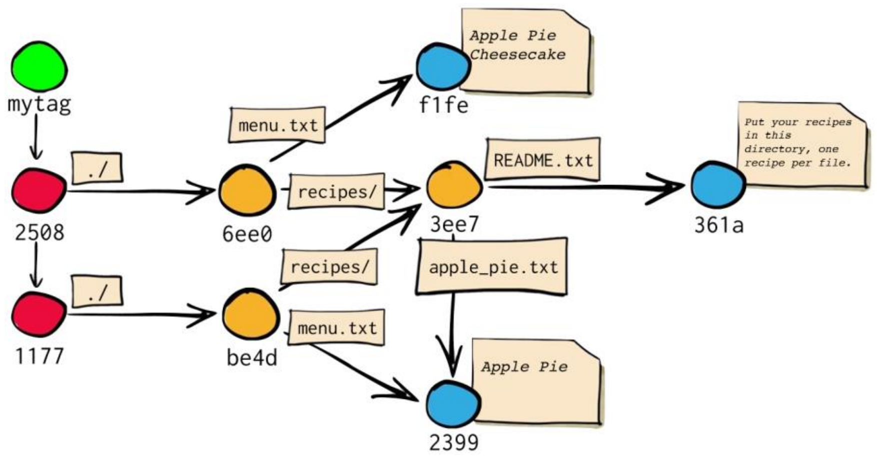
- A tag is an object that points to a commit

Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m1&clip=6&mode=live

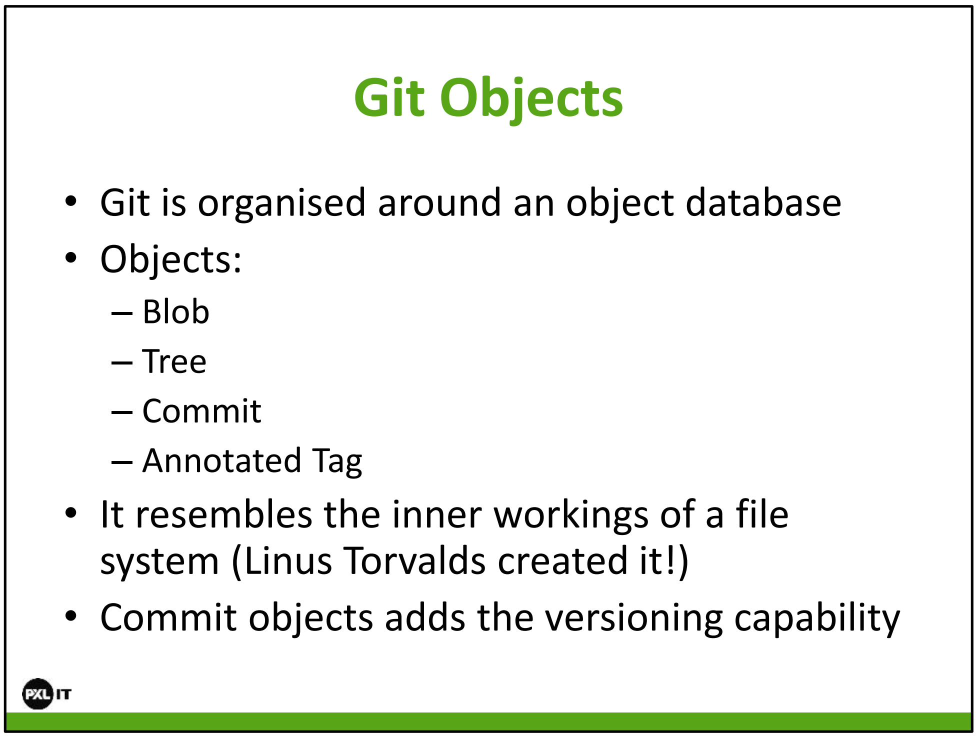
Steps:

- git tag –a mytag –m “I love cheesecake”
- git tag → shows the name of the tags
- git cat-file –p mytag

The Object Database



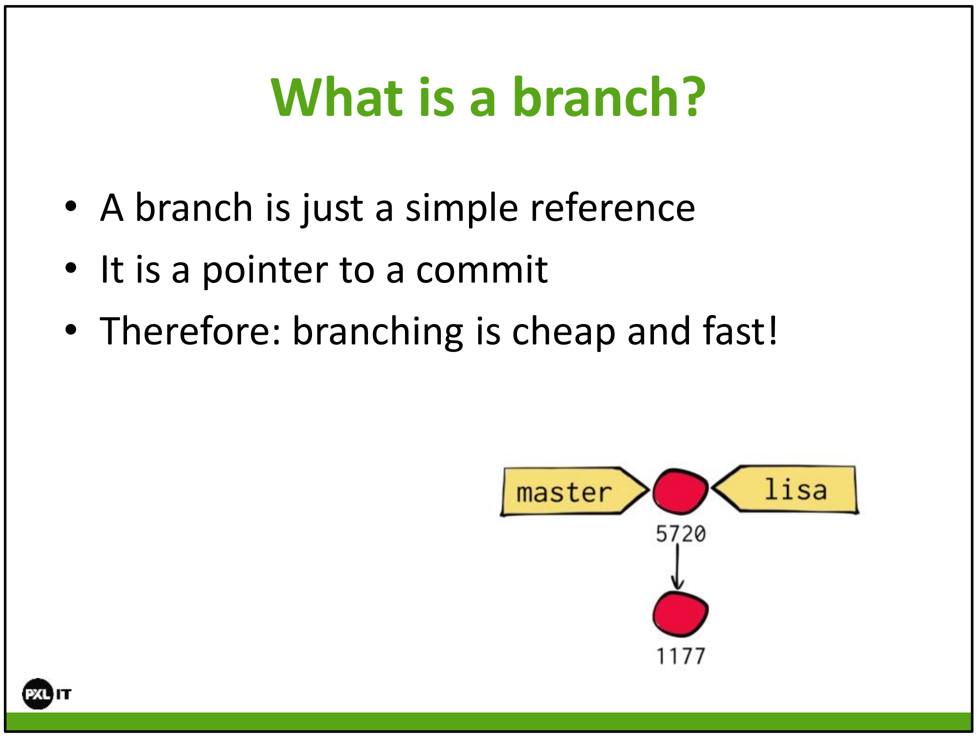
The tag <mytag> points to a commit



Git Objects

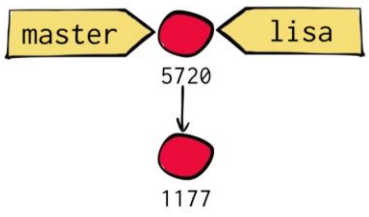
- Git is organised around an object database
- Objects:
  - Blob
  - Tree
  - Commit
  - Annotated Tag
- It resembles the inner workings of a file system (Linus Torvalds created it!)
- Commit objects adds the versioning capability

BRANCHING DEMYSTIFIED



What is a branch?

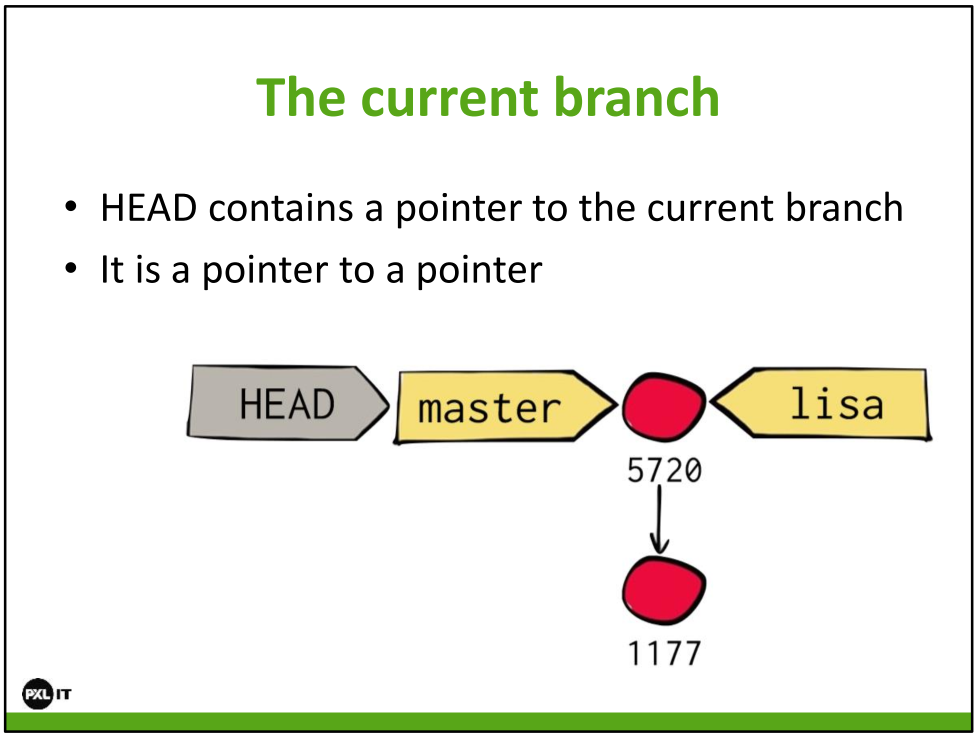
- A branch is just a simple reference
- It is a pointer to a commit
- Therefore: branching is cheap and fast!



Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m2&clip=1&mode=live

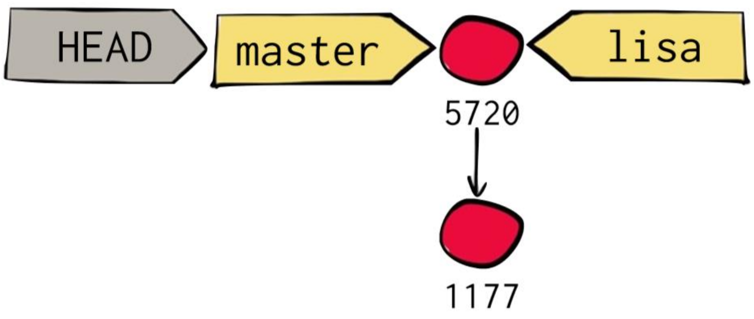
Steps:

- git branch → shows only one branch, the “master” branch
- Look in .git → directory refs/heads/master → just a text file
- Create new branch: git branch lisa → new file in refs/heads



The current branch

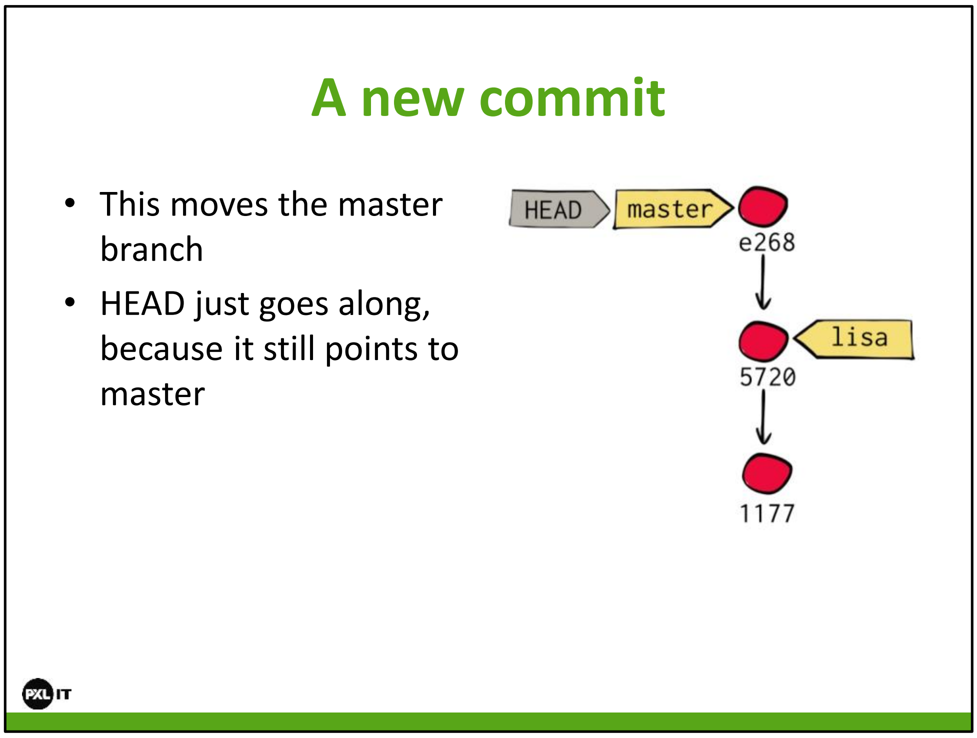
- HEAD contains a pointer to the current branch
- It is a pointer to a pointer



Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m2&clip=2&mode=live

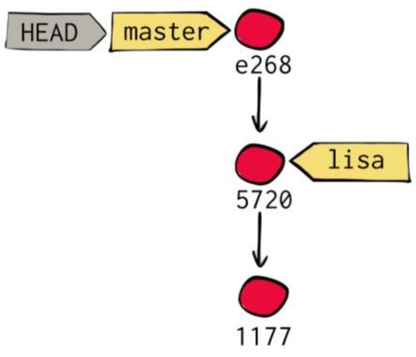
Steps:

- git branch → * denotes “current” branch
- HEAD contains a reference a file refs/head/master



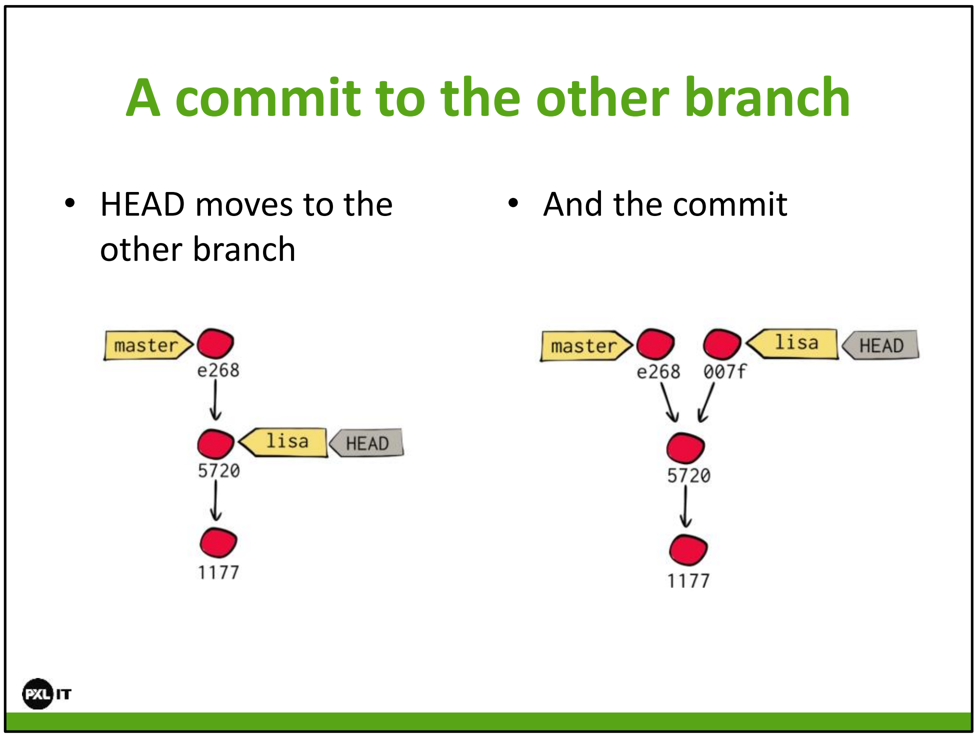
A new commit

- This moves the master branch
- HEAD just goes along, because it still points to master



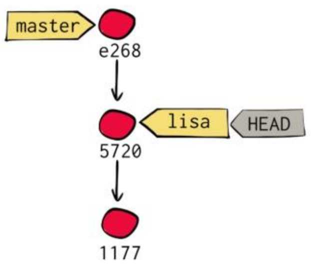
Steps:

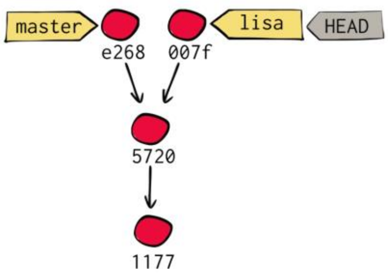
- Make a change to apple_pie.txt and commit



A commit to the other branch

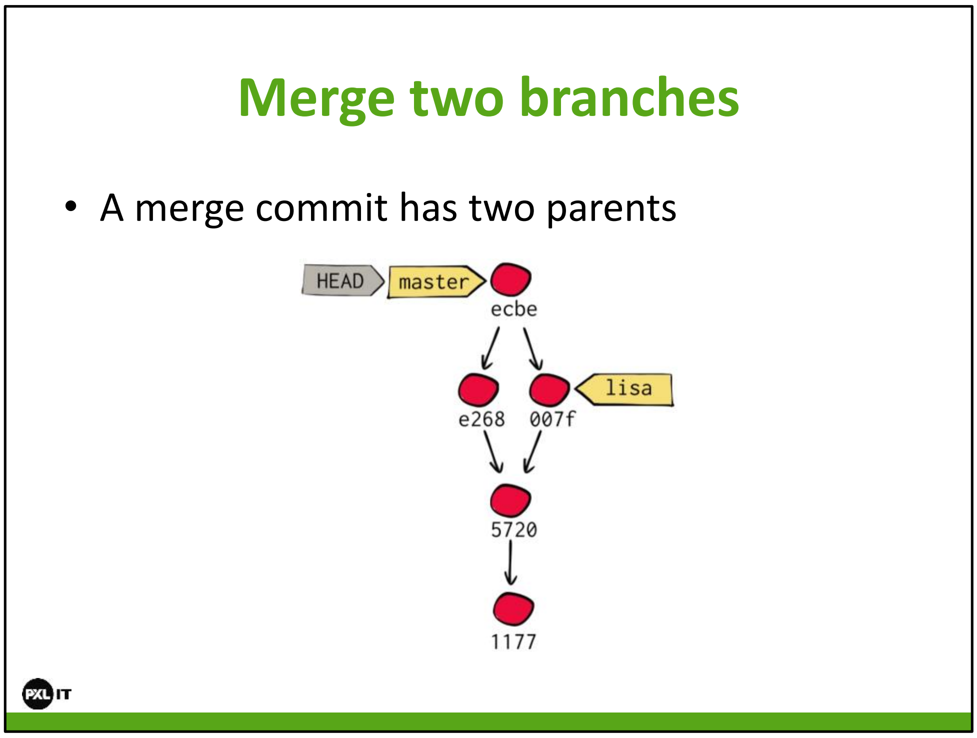
- HEAD moves to the other branch
- And the commit





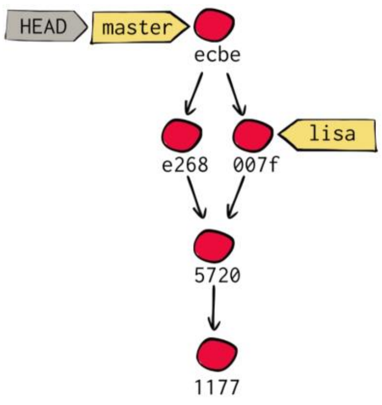
A checkout to another branch does 2 things:

- HEAD moves to the other branch
- The working area gets restored based on the current objects in the database.



Merge two branches

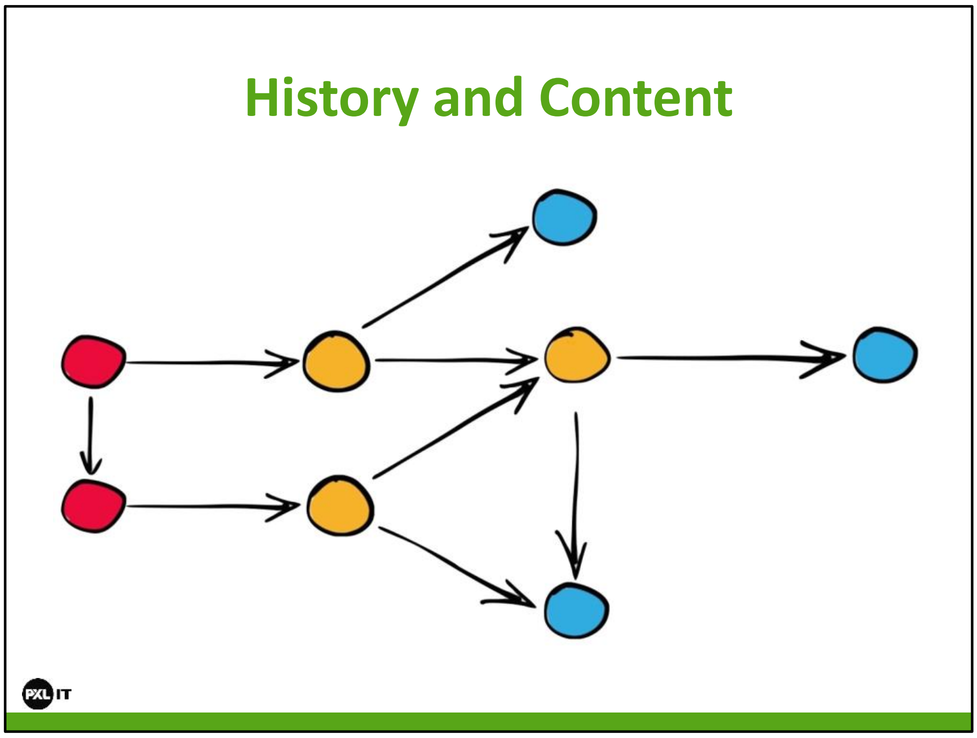
- A merge commit has two parents



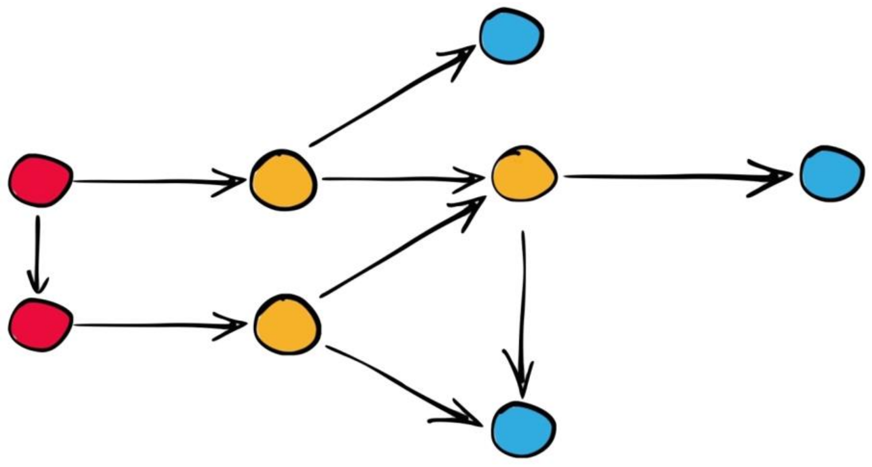
Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m2&clip=3&mode=live

Steps:

- Merge branch lisa into master
- A new commit is created (after resolving the conflict)
- git cat-file –p to this commit → you see it has two parents!

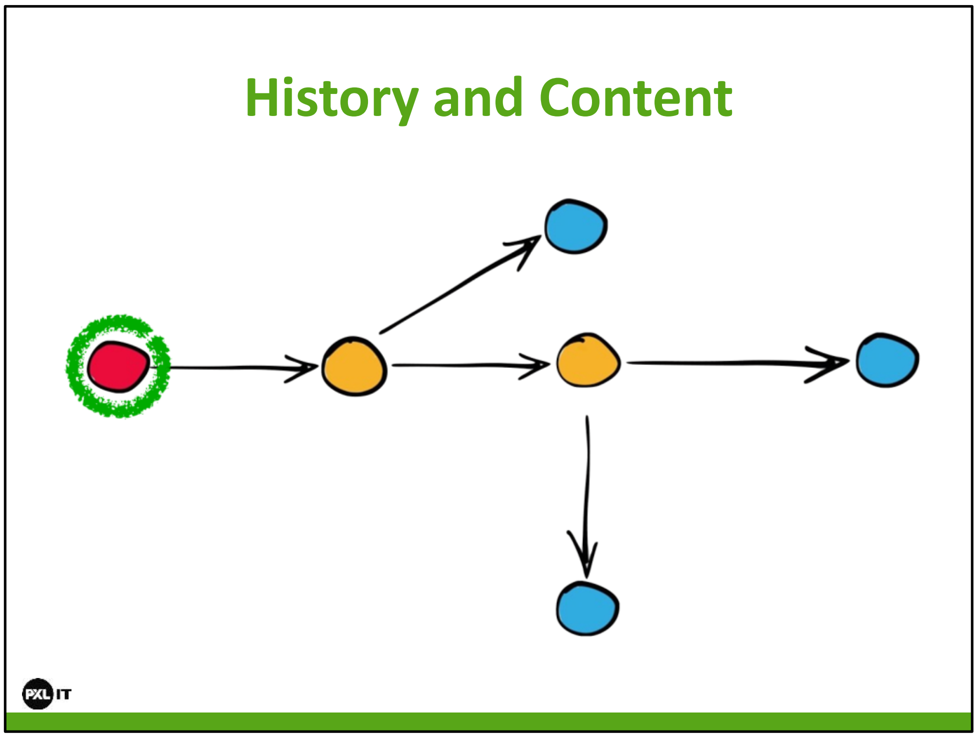


History and Content

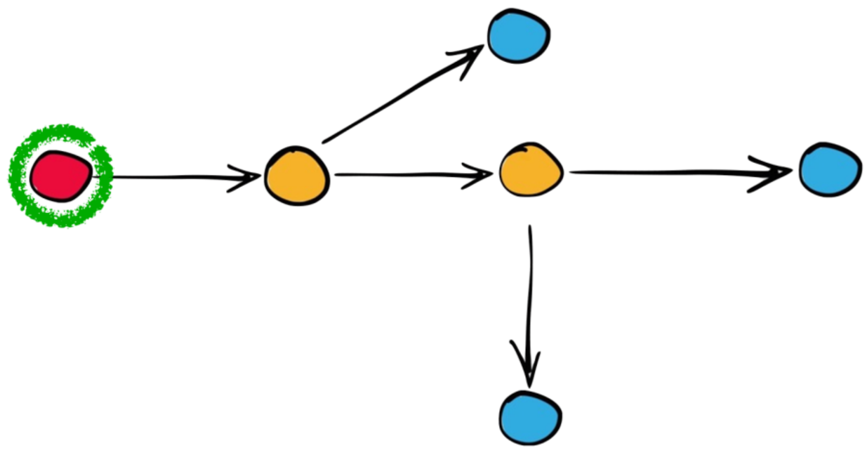


Git manages everything by means of a graph of objects:

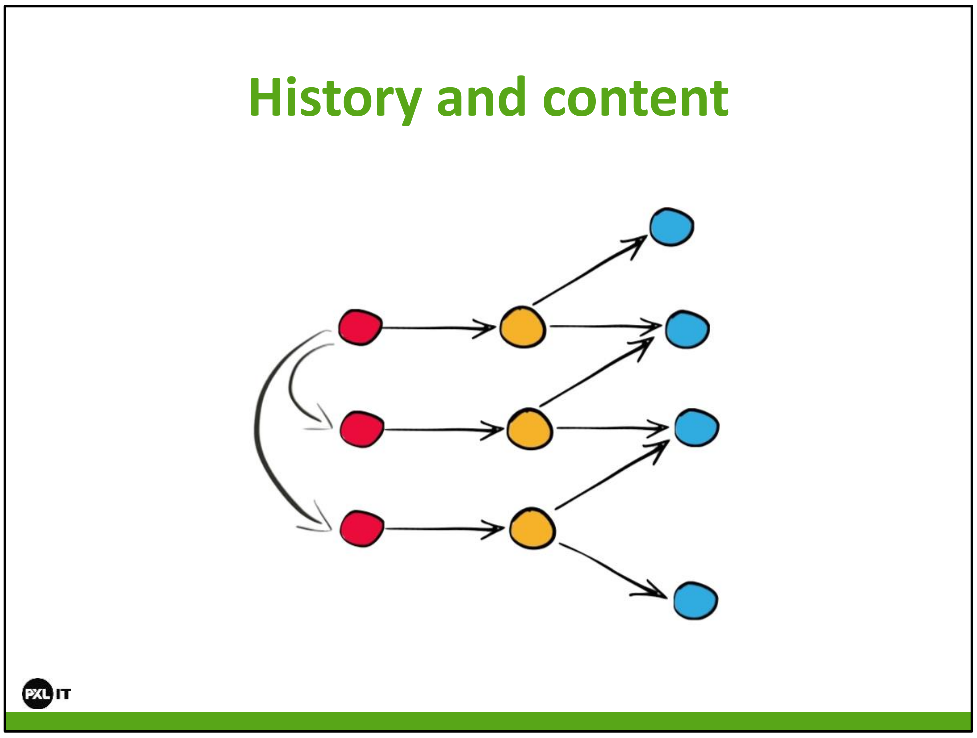
- Commits point to other commits: this is the history
- Commits pointing to trees and blobs: this is the content



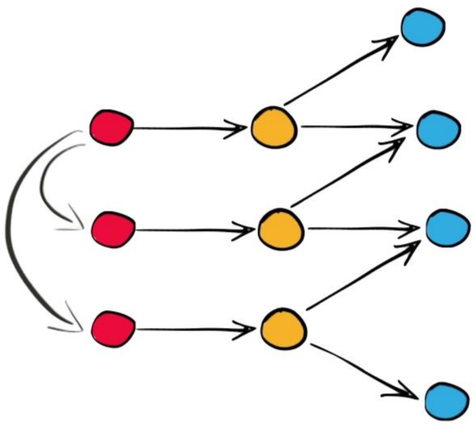
History and Content



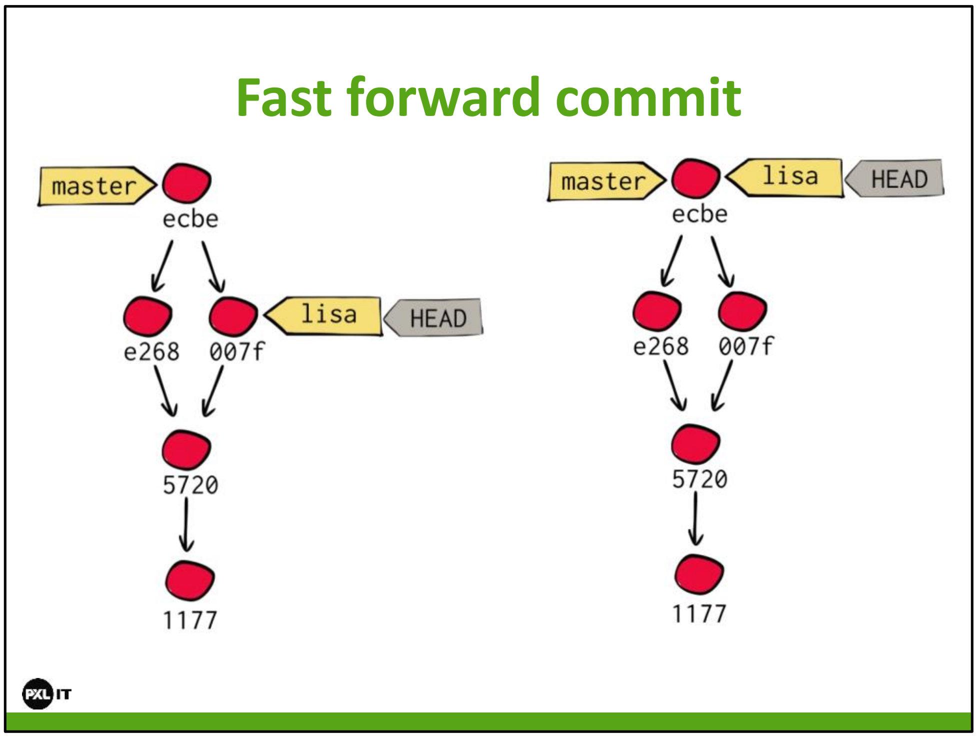
When you checkout a commit, it replaces everything in your working directory with the content of the object database. It isolates the commit, follows the trees and restores the content from the blobs.



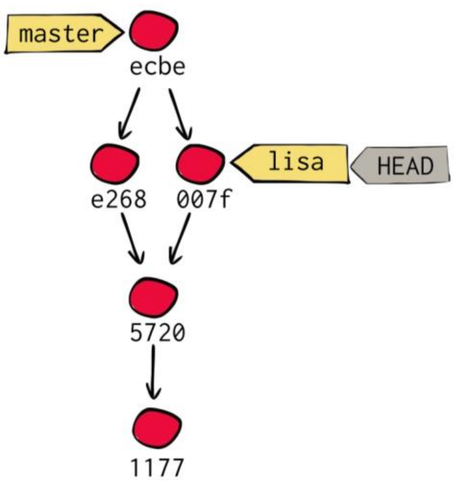
History and content

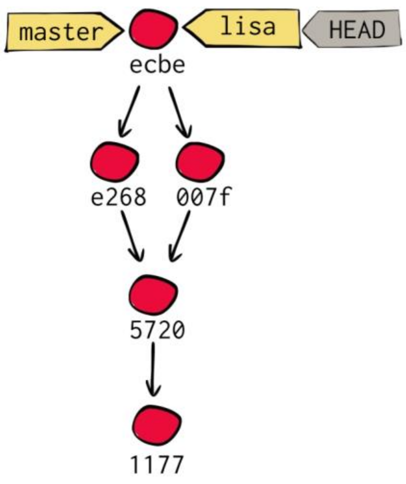


To checkout a merge commit, git follows the exact same pattern. Only: now you have to track additional commits (the parents) to restore all content.



Fast forward commit

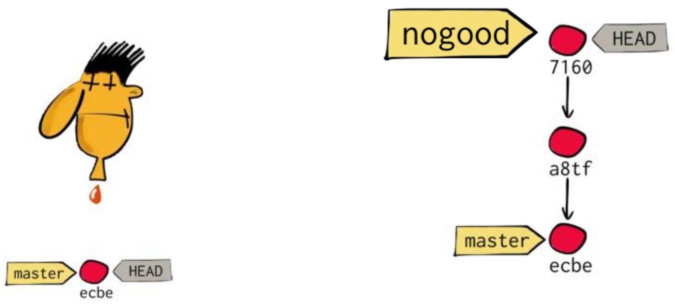




What happens when you merge master into lisa? Normally you should create a new merge commit with parents “ecbe” and “007f”, but because lisa’s change is exact the same as “ecbe”, git re-uses the content and fast forwards the commit.

A Detached Head

- When HEAD does not point to a branch, but to a COMMIT
- Useful for experimenting



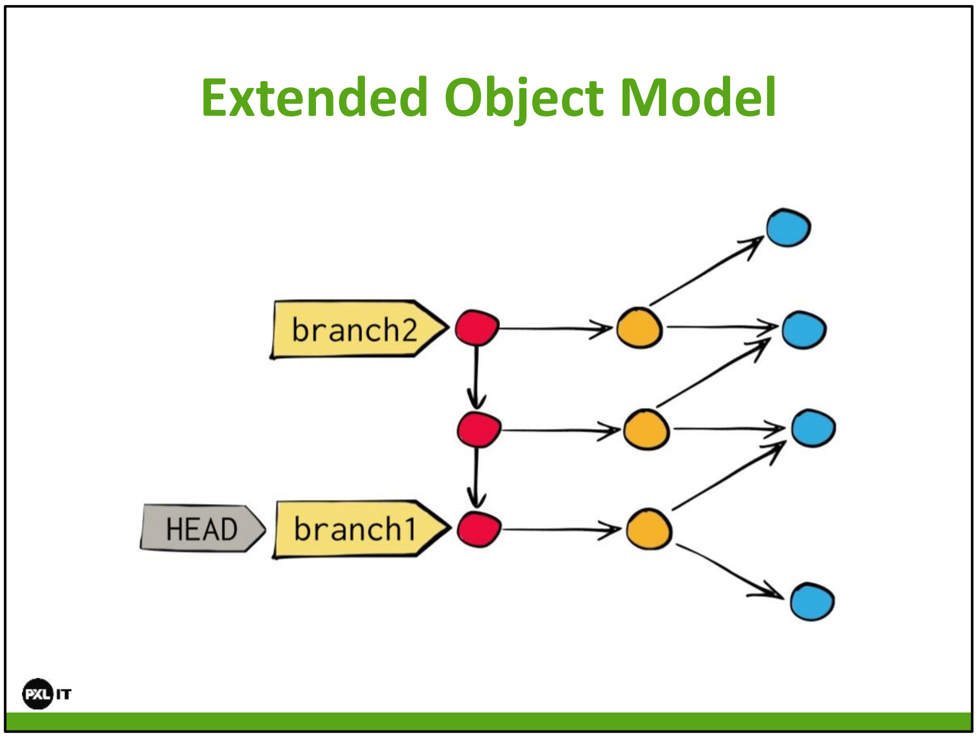
Demo: https://app.pluralsight.com/player?course=how-git-works&author=paolo-perrotta&name=how-git-works-m2&clip=6&mode=live

Step:

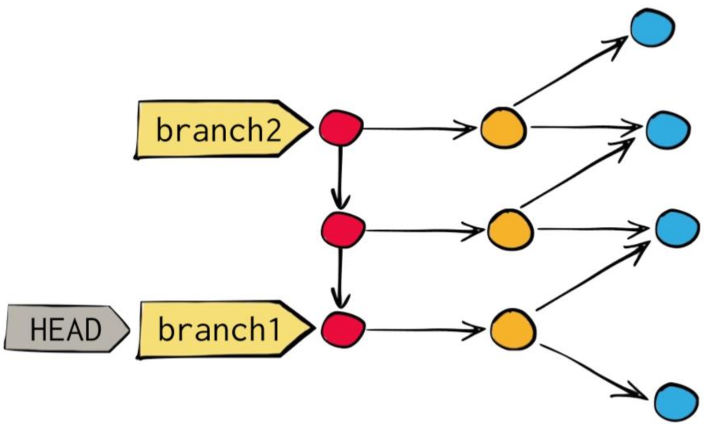
git checkout ecbe → this does a checkout of a commit instead of a branch!

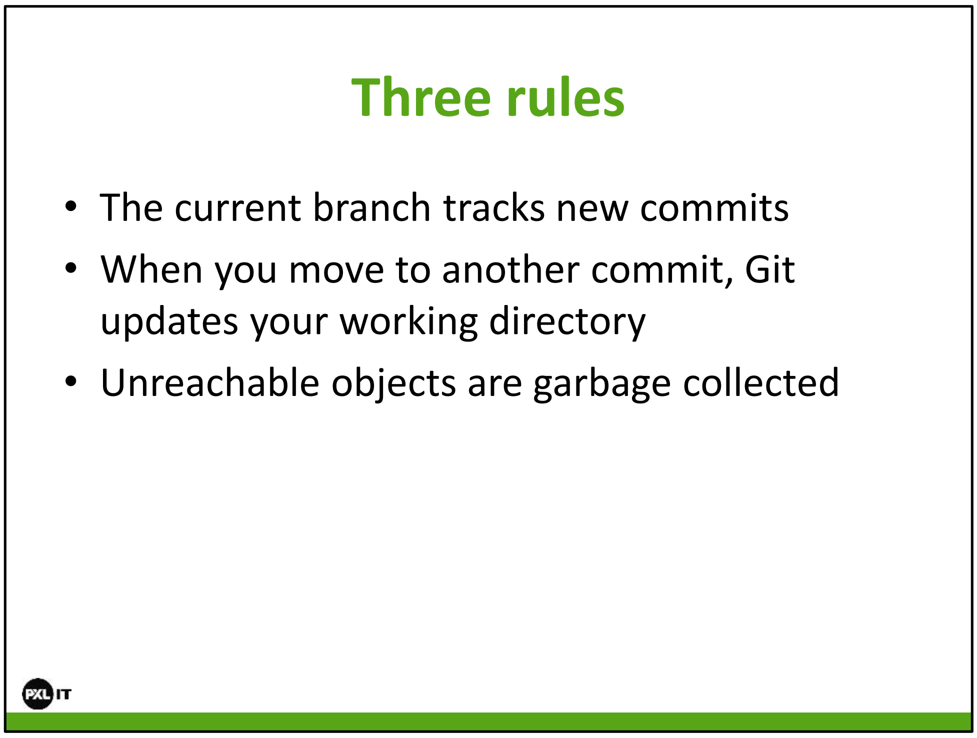
Now experiment and do a couple of commits → this moves HEAD, but this does not belong to a branch.
After a while, git will garbage collect these commits, unless you assign it a commit

git checkout 7160
git branch nogood



Extended Object Model





Three rules

- The current branch tracks new commits
- When you move to another commit, Git updates your working directory
- Unreachable objects are garbage collected