

GIT

An Introduction

Roeland Matthijssens & Bert Van Horenbeek

Osudio

1 February 2016

Git Object Hashmap

- internal storage git (./git/objects/)
- all content that represents history is stored here, ALL!
- stored in a hashmap of sha1 keys to content of objects
- different types of objects

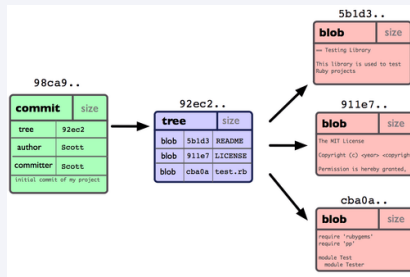
```
$ git cat-file -p <hash>
```

```
$ tree .git/objects
.git/objects/
| 9d/
|   '-- aeafb9864cf43055ae93beb0afd6c7d144bfa
'-- info/
'-- pack/
```

3 directories , 1 file

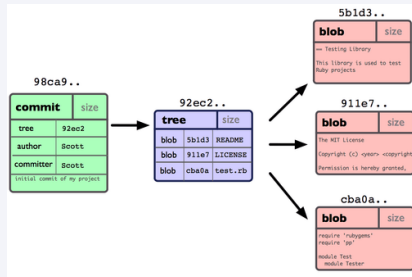
Git Object Hashmap

- blob
- tree
- commit
- tag (or are they?)



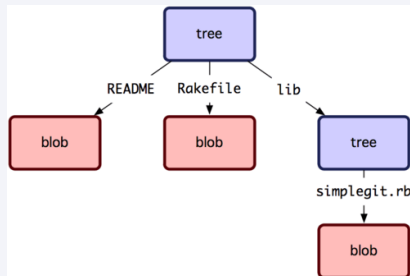
blob

- flat files
- the actual source code
- full copy
- has no further references to other objects



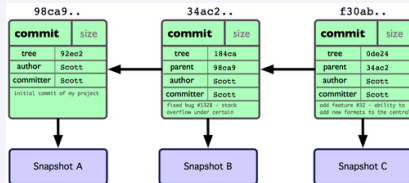
tree

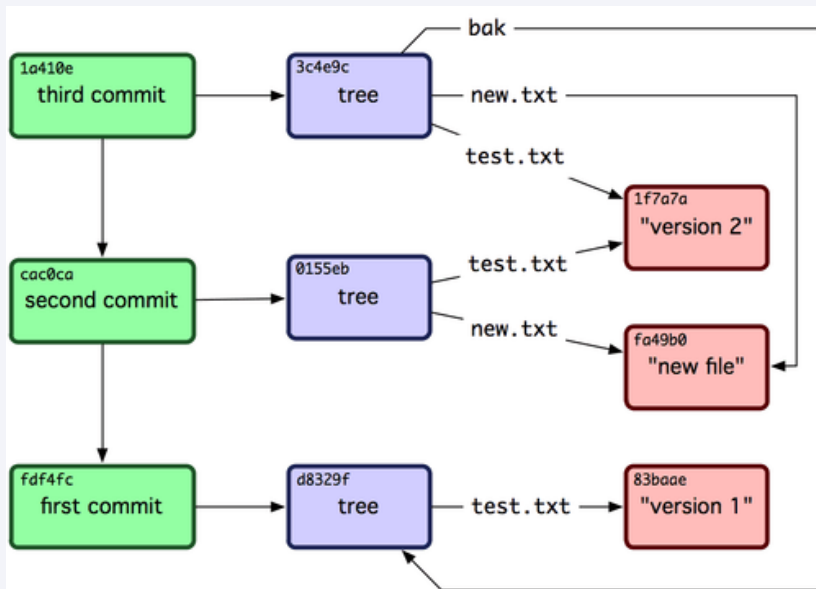
- represents a directory on disk
- retains file permissions (in unix notation)
- points to other trees and blobs



commit

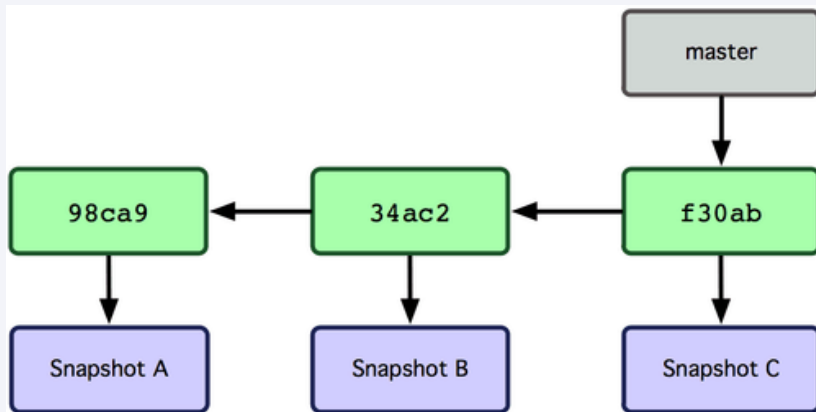
- stores author, timestamp, message, ...
- points to a tree
- points to 1 or more commits (parent commits)





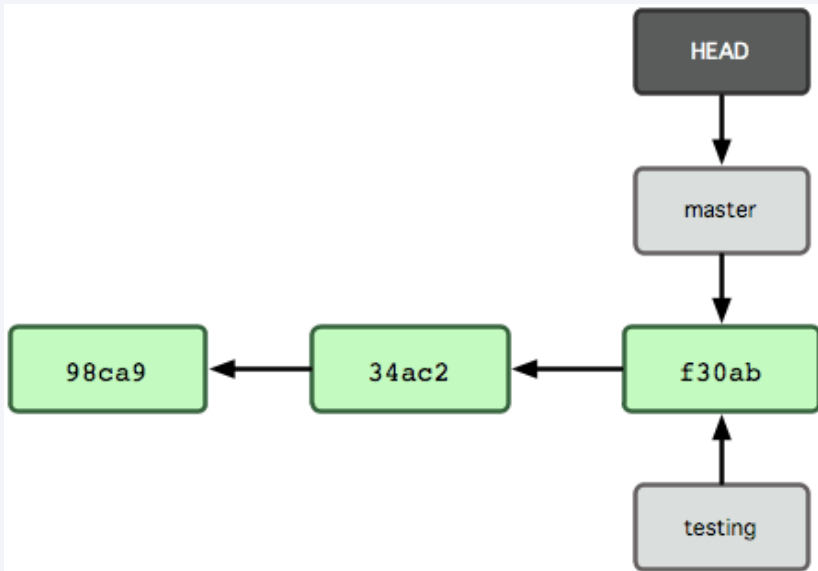
branches

- point to a commit (flat file with sha1 hash).
- moves pointer on commit, revert, rebase, reset, ...
- stored in `.git/refs/`



What is HEAD

- points to a branch
- flat file with relative path to branch
- or sha1 hash in detached head mode.
- moves on git checkout
- does NOT move on revert, rebase, reset, commit, ...



Tags vs Anotated Tags

Tags

- comparable to branch, can not be commit on
- located in `.git/refs/tags`
- does not create git objects

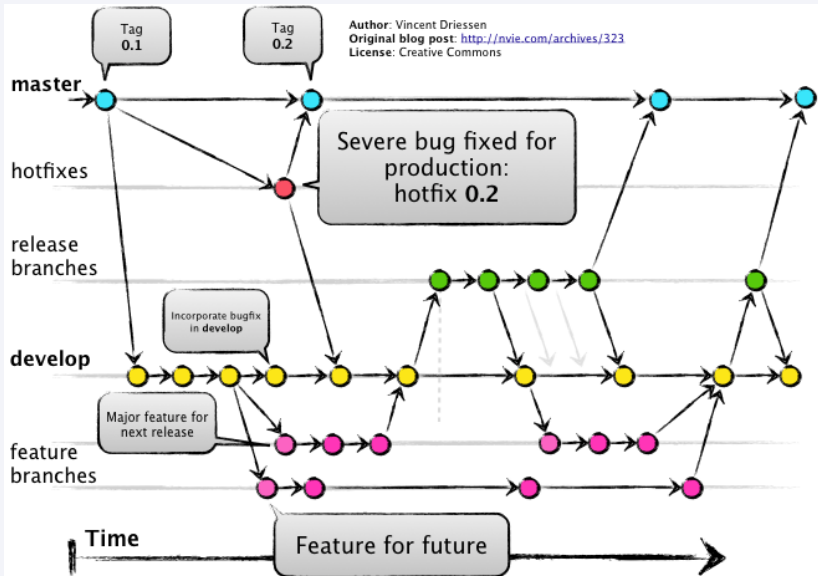
Anotated Tags

- comparable to branch, can not be commit on
- has tagger, time and message
- can be cryptographically signed! (REAL TAGS)
- does create git objects (`'git cat-file -p <tag-name>'`)
- will be used for releases.

And now, a demo

All hail the demo gods.

gitflow



gitflow branches

- master, develop (ever present)
- feature branch (each jira ticket)
- hotfix branch (production issues)
- release branch (only showstopper)

gitflow rules

- feature branch:
 - from develop
 - to develop
- hotfix branch:
 - from master
 - to master and develop (develop will also need this fix)
- release branch:
 - from develop
 - to master and develop

You are implementing a feature.

- create branch ('feature/JIRA-547')
- commit, commit, commit, ..., revert, commit again
- keep running unit tests
- create pull request
- reviewer merges back into develop
- ci deploys develop to TST

feature pull request breaks CI build

- merge develop in feature/JIRA-547
- commit, commit, commit, ... (hopefully fixing stuff)
- reopen pull request
- rinse and repeat

Preparing for a new release

- branch from develop ('release/release-version')
- deploy release branch to ACC
- have customer test everything
- fix showstoppers on release branch
- merge into master and back into develop
- tag merge commit on master branch

Production goes down

- branch from master ('hotfix/issue-description')
- commit, commit, commit, ...
- merge into master and develop
- tag merge commit on master branch
- deploy master to PRD

And now, a demo

All hail the demo gods.

merge vs rebase

