Tagging the latest ‘stable’ version in `git`

Javier Lopez-Gomez

March 22, 2021

ROOT project, EP/SFT, CERN

Objects and the project history[1]

Object: sequence of bytes stored in Git; identified by its SHA-1.

Types: *commit*, *tree*, *blob*, *tag*.

[1] https://github.com/jalopezg-r00t/ARCOS-Git_-_There_be_dragons/blob/master/GitTalk.pdf

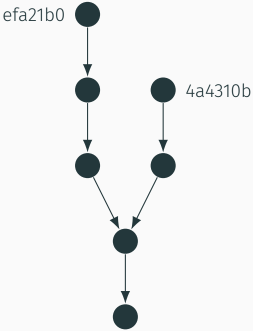
Objects and the project history[1]

Object: sequence of bytes stored in Git; identified by its SHA-1.

Types: *commit*, *tree*, *blob*, *tag*.

Project history: DAG in which each commit points to its parents.

Useless if we lack 'symbolic' names to refer to some objects, i.e. references.



[1] https://github.com/jalopezg-r00t/ARCOS-Git_-_There_be_dragons/blob/master/GitTalk.pdf

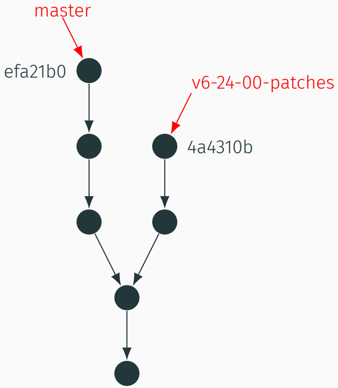
Objects and the project history[1]

Object: sequence of bytes stored in Git; identified by its SHA-1.

Types: *commit*, *tree*, *blob*, *tag*.

Project history: DAG in which each commit points to its parents.

Useless if we lack 'symbolic' names to refer to some objects, i.e. references.



[1] https://github.com/jalopezg-r00t/ARCOS-Git_-_There_be_dragons/blob/master/GitTalk.pdf

Objects and the project history[1]

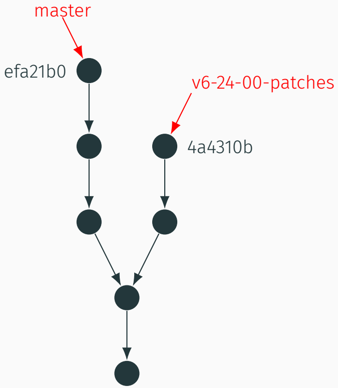
Object: sequence of bytes stored in Git; identified by its SHA-1.

Types: *commit*, *tree*, *blob*, *tag*.

Project history: DAG in which each commit points to its parents.

Useless if we lack 'symbolic' names to refer to some objects, i.e. references.

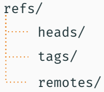
Commit-ish: A commit object... or something that can be recursively resolved to a commit object.



[1] https://github.com/jalopezg-r00t/ARCOS-Git_-_There_be_dragons/blob/master/GitTalk.pdf

References (refs) and their namespace

(Ref)erence: a name that points to an object, i.e. to its SHA-1. Hierarchical namespace rooted at `refs/`.



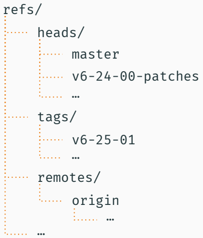
```
refs/
   heads/
   tags/
   remotes/
```

References (refs) and their namespace

(Ref)erence: a name that points to an object, i.e. to its SHA-1. Hierarchical namespace rooted at `refs/`.

Branch: a commit-ish under `refs/heads/`

Tag: a commit-ish under `refs/tags/`. Typically a ref that points to a tag object, but not necessarily…



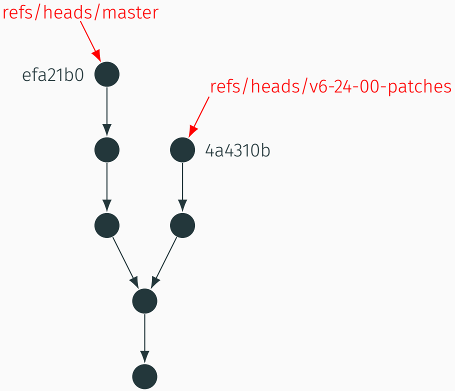
```
refs/
    heads/
        master
        v6-24-00-patches
        …
    tags/
        v6-25-01
        …
    remotes/
        origin
            …
    …
```

References (refs) and their namespace

(Ref)erence: a name that points to an object, i.e. to its SHA-1. Hierarchical namespace rooted at `refs/`.

Branch: a commit-ish under `refs/heads/`

Tag: a commit-ish under `refs/tags/`. Typically a ref that points to a tag object, but not necessarily...

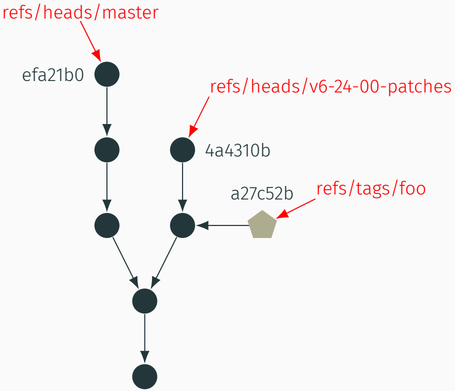


References (refs) and their namespace

(Ref)erence: a name that points to an object, i.e. to its SHA-1. Hierarchical namespace rooted at `refs/`.

Branch: a commit-ish under `refs/heads/`

Tag: a commit-ish under `refs/tags/`. Typically a ref that points to a tag object, but not necessarily...

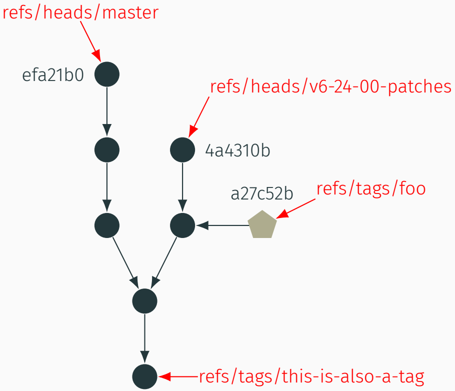


References (refs) and their namespace

(Ref)erence: a name that points to an object, i.e. to its SHA-1. Hierarchical namespace rooted at `refs/`.

Branch: a commit-ish under `refs/heads/`

Tag: a commit-ish under `refs/tags/`. Typically a ref that points to a tag object, but not necessarily...



Git refs: summary

- A reference is a name that points to an object.
- Refs are stored in the `refs/` hierarchy.
- To porcelain commands, something is a branch or a tag depending on the fully-qualified name of the ref.

Our ‘stable‘ tag

I propose:
to keep track of the stable version as a direct ref (not via a tag object) on the ‘/refs/tags/’ namespace.

How?
Git plumbing repertoire has `git-update-ref`, so that translates into:

```
$ COMMIT_TO_TAG=4a4310b9f284a580496ec6b0b823eb10f88e8e4e

$ git update-ref refs/tags/stable $COMMIT_TO_TAG
$ git push -f origin refs/tags/stable
```

When/what to retag as ‘`stable`’?

1. Manually?
2. As a GitHub action, triggered after creating a tag that looks like `vX-YY-ZZ`, where `YY` is an even number?
3. In Jenkins: after a build is green in a certain branch, e.g. `6-24-00-patches`?
4. ??