

# Git et Gitlab au quotidien

## Commandes Principales et Workflows

### Glossaire

Benoît Bayol

June 9, 2015

# Outline

Licence

Glossary

Topic

Licence

Glossary

- ▶ Les définitions sont extraites du man gitglossary
- ▶ Les graphiques présents dans les slides sont extraits du livre "Pro Git 2" qui est aussi présent dans l'archive.
- ▶ La licence de ce document est :  
<https://creativecommons.org/licenses/by-nc-sa/3.0/>

# Topic

Licence

Glossary

The action of updating all or part of the working tree with a tree object or blob from the object database, and updating the index and HEAD if the whole working tree has been pointed at a new branch.

# commit

As a noun: A single point in the Git history; the entire history of a project is represented as a set of interrelated commits. The word "commit" is often used by Git in the same places other revision control systems use the words "revision" or "version". Also used as a short hand for commit object.

As a verb: The action of storing a new snapshot of the project's state in the Git history, by creating a new commit representing the current state of the index and advancing HEAD to point at the new commit.

## evil merge

An evil merge is a merge that introduces changes that do not appear in any parent.



## fast-forward

A fast-forward is a special type of merge where you have a revision and you are "merging" another branch's changes that happen to be a descendant of what you have. In such these cases, you do not make a new mergecommit but instead just update to his revision. This will happen frequently on a remote-tracking branch of a remote repository.

## merge

As a verb: To bring the contents of another branch (possibly from an external repository) into the current branch. In the case where the merged-in branch is from a different repository, this is done by first fetching the remote branch and then merging the result into the current branch. This combination of fetch and merge operations is called a pull. Merging is performed by an automatic process that identifies changes made since the branches diverged, and then applies all those changes together. In cases where changes conflict, manual intervention may be required to complete the merge.

As a noun: unless it is a fast-forward, a successful merge results in the creation of a new commit representing the result of the merge, and having as parents the tips of the merged branches. This commit is referred to as a "merge commit", or sometimes just a "merge".

## origin

The default upstream repository. Most projects have at least one upstream project which they track. By default origin is used for that purpose. New upstream updates will be fetched into remote- tracking branches named `origin/name-of-upstream-branch`, which you can see using `git branch -r`.

## push

Pushing a branch means to get the branch's head ref from a remote repository, find out if it is a direct ancestor to the branch's local head ref, and in that case, putting all objects, which are reachable from the local head ref, and which are missing from the remote repository, into the remote object database, and updating the remote head ref. If the remote head is not an ancestor to the local head, the push fails.

# rebase

To reapply a series of changes from a branch to a different base, and reset the head of that branch to the result.

A name that begins with refs/ (e.g. refs/heads/master) that points to an object name or another ref (the latter is called a symbolic ref). For convenience, a ref can sometimes be abbreviated when used as an argument to a Git command; see gitrevisions(7) for details. Refs are stored in the repository.

The ref namespace is hierarchical. Different subhierarchies are used for different purposes (e.g. the refs/heads/ hierarchy is used to represent local branches).

There are a few special-purpose refs that do not begin with refs/. The most notable example is HEAD

A reflog shows the local "history" of a ref. In other words, it can tell you what the 3rd last revision in this repository was, and what was the current state in this repository, yesterday 9:14pm.

## remote-tracking branch

A ref that is used to follow changes from another repository. It typically looks like `refs/remotes/foo/bar` (indicating that it tracks a branch named `bar` in a remote named `foo`), and matches the right-hand-side of a configured fetch refspec. A remote-tracking branch should not contain direct modifications or have local commits made to it.



## tag

A ref under `refs/tags/` namespace that points to an object of an arbitrary type (typically a tag points to either a tag or a commit object). In contrast to a head, a tag is not updated by the commit command. A Git tag has nothing to do with a Lisp tag (which would be called an object type in Git's context). A tag is most typically used to mark a particular point in the commit ancestry chain.

## working tree

The tree of actual checked out files. The working tree normally contains the contents of the HEAD commit's tree, plus any local changes that you have made but not yet committed.