NAME

gitglossary − A GIT Glossary

SYNOPSIS

*

DESCRIPTION

alternate object database
: Via the alternates mechanism, a repository can inherit part of its object database from another object database, which is called "alternate".

bare repository
: A bare repository is normally an appropriately named directory with a .git suffix that does not have a locally checked−out copy of any of the files under revision control. That is, all of the git administrative and control files that would normally be present in the hidden .git sub−directory are directly present in the repository.git directory instead, and no other files are present and checked out. Usually publishers of public repositories make bare repositories available.

blob object
: Untyped object, e.g. the contents of a file.

branch
: A "branch" is an active line of development. The most recent commit on a branch is referred to as the tip of that branch. The tip of the branch is referenced by a branch head, which moves forward as additional development is done on the branch. A single git repository can track an arbitrary number of branches, but your working tree is associated with just one of them (the "current" or "checked out" branch), and HEAD points to that branch.

cache
: Obsolete for: index.

chain
: A list of objects, where each object in the list contains a reference to its successor (for example, the successor of a commit could be one of its parents).

changeset
: BitKeeper/cvsps speak for "commit". Since git does not store changes, but states, it really does not make sense to use the term "changesets" with git.

checkout
: 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.

cherry−picking
: In SCM jargon, "cherry pick" means to choose a subset of changes out of a series of changes (typically commits) and record them as a new series of changes on top of a different codebase. In GIT, this is performed by the "git cherry−pick" command to extract the change introduced by an existing commit and to record it based on the tip of the current branch as a new commit.

clean
: A working tree is clean, if it corresponds to the revision referenced by the current head. Also see "dirty".

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.

commit object
: An object which contains the information about a particular revision, such as parents, committer, author, date and the tree object which corresponds to the top directory of the stored revision.

core git
: Fundamental data structures and utilities of git. Exposes only limited source code management tools.

DAG
: Directed acyclic graph. The commit objects form a directed acyclic graph, because they have parents (directed), and the graph of commit objects is acyclic (there is no chain which begins and ends with the same object).

dangling object
: An unreachable object which is not reachable even from other unreachable objects; a dangling object has no references to it from any reference or object in the repository.

detached HEAD
: Normally the HEAD stores the name of a branch. However, git also allows you to check out an arbitrary commit that isn't necessarily the tip of any particular branch. In this case HEAD is said to be "detached".

dircache
: You are **waaaaay** behind. See index.

directory
: The list you get with "ls" :−)

dirty
: A working tree is said to be "dirty" if it contains modifications which have not been committed to the current branch.

ent
: Favorite synonym to "tree−ish" by some total geeks. See http://en.wikipedia.org/wiki/Ent_(Middle−earth) for an in−depth explanation. Avoid this term, not to confuse people.

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 merge commit but instead just update to his revision. This will happen frequently on a tracking branch of a remote repository.

fetch
: Fetching a branch means to get the branch's head ref from a remote repository, to find out which objects are missing from the local object database, and to get them, too. See also **git-fetch**(1).

file system
: Linus Torvalds originally designed git to be a user space file system, i.e. the infrastructure to hold files and directories. That ensured the efficiency and speed of git.

git archive
: Synonym for repository (for arch people).

grafts
: Grafts enables two otherwise different lines of development to be joined together by recording fake ancestry information for commits. This way you can make git pretend the set of parents a commit has is different from what was recorded when the commit was created. Configured via the .git/info/grafts file.

hash

In git’s context, synonym to object name.

head
: A named reference to the commit at the tip of a branch. Heads are stored in $GIT_DIR/refs/heads/, except when using packed refs. (See **git-pack-refs**(1).)

HEAD
: The current branch. In more detail: Your working tree is normally derived from the state of the tree referred to by HEAD. HEAD is a reference to one of the heads in your repository, except when using a detached HEAD, in which case it may reference an arbitrary commit.

head ref
: A synonym for head.

hook
: During the normal execution of several git commands, call−outs are made to optional scripts that allow a developer to add functionality or checking. Typically, the hooks allow for a command to be pre−verified and potentially aborted, and allow for a post−notification after the operation is done. The hook scripts are found in the $GIT_DIR/hooks/ directory, and are enabled by simply removing the .sample suffix from the filename. In earlier versions of git you had to make them executable.

index
: A collection of files with stat information, whose contents are stored as objects. The index is a stored version of your working tree. Truth be told, it can also contain a second, and even a third version of a working tree, which are used when merging.

index entry
: The information regarding a particular file, stored in the index. An index entry can be unmerged, if a merge was started, but not yet finished (i.e. if the index contains multiple versions of that file).

master
: The default development branch. Whenever you create a git repository, a branch named "master" is created, and becomes the active branch. In most cases, this contains the local development, though that is purely by convention and is not required.

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".

object
: The unit of storage in git. It is uniquely identified by the SHA1 of its contents. Consequently, an object can not be changed.

object database
: Stores a set of "objects", and an individual object is identified by its object name. The objects usually live in $GIT_DIR/objects/.

object identifier
: Synonym for object name.

object name
: The unique identifier of an object. The hash of the object’s contents using the Secure Hash Algorithm 1 and usually represented by the 40 character hexadecimal encoding of the hash of the object.

object type
: One of the identifiers "commit", "tree", "tag" or "blob" describing the type of an object.

octopus
: To merge more than two branches. Also denotes an intelligent predator.

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.

pack
: A set of objects which have been compressed into one file (to save space or to transmit them efficiently).

pack index
: The list of identifiers, and other information, of the objects in a pack, to assist in efficiently accessing the contents of a pack.

parent
: A commit object contains a (possibly empty) list of the logical predecessor(s) in the line of development, i.e. its parents.

pickaxe
: The term pickaxe refers to an option to the diffcore routines that help select changes that add or delete a given text string. With the −−pickaxe−all option, it can be used to view the full changeset that introduced or removed, say, a particular line of text. See **git-diff**(1).

plumbing
: Cute name for core git.

porcelain
: Cute name for programs and program suites depending on core git, presenting a high level access to core git. Porcelains expose more of a SCM interface than the plumbing.

pull
: Pulling a branch means to fetch it and merge it. See also **git-pull**(1).

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.

reachable
: All of the ancestors of a given commit are said to be "reachable" from that commit. More generally, one object is reachable from another if we can reach the one from the other by a chain that follows tags to whatever they tag, commits to their parents or trees, and trees to the trees or blobs that they contain.

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

ref
: A 40−byte hex representation of a SHA1 or a name that denotes a particular object. These may be stored in $GIT_DIR/refs/.

reflog
: 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. See **git-reflog**(1) for details.

refspec
: A "refspec" is used by fetch and push to describe the mapping between remote ref and local ref. They are combined with a colon in the format <src>:<dst>, preceded by an optional plus sign, +. For example: git fetch $URL refs/heads/master:refs/heads/origin means "grab the master branch head from the $URL and store it as my origin branch head". And git push $URL refs/heads/master:refs/heads/to−upstream means "publish my master branch head as to−upstream branch at $URL". See also **git-push**(1).

repository
: A collection of refs together with an object database containing all objects which are reachable from the refs, possibly accompanied by meta data from one or more porcelains. A repository can share an object database with other repositories via alternates mechanism.

resolve
: The action of fixing up manually what a failed automatic merge left behind.

revision
: A particular state of files and directories which was stored in the object database. It is referenced by a commit object.

rewind
: To throw away part of the development, i.e. to assign the head to an earlier revision.

SCM
: Source code management (tool).

SHA1
: Synonym for object name.

shallow repository
: A shallow repository has an incomplete history some of whose commits have parents cauterized away (in other words, git is told to pretend that these commits do not have the parents, even though they are recorded in the commit object). This is sometimes useful when you are interested only in the recent history of a project even though the real history recorded in the upstream is much larger. A shallow repository is created by giving the −−depth option to **git-clone**(1), and its history can be later deepened with **git-fetch**(1).

symref
: Symbolic reference: instead of containing the SHA1 id itself, it is of the format *ref: refs/some/thing* and when referenced, it recursively dereferences to this reference. *HEAD* is a prime example of a symref. Symbolic references are manipulated with the **git-symbolic-ref**(1) command.

tag
: A ref pointing to a tag or commit object. In contrast to a head, a tag is not changed by a commit. Tags (not tag objects) are stored in $GIT_DIR/refs/tags/. 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.

tag object
: An object containing a ref pointing to another object, which can contain a message just like a commit object. It can also contain a (PGP) signature, in which case it is called a "signed tag object".

topic branch
: A regular git branch that is used by a developer to identify a conceptual line of development. Since branches are very easy and inexpensive, it is often desirable to have several small branches that each contain very well defined concepts or small incremental yet related changes.

tracking branch
: A regular git branch that is used to follow changes from another repository. A tracking branch should not contain direct modifications or have local commits made to it. A tracking branch can usually be identified as the right−hand−side ref in a Pull: refspec.

tree
: Either a working tree, or a tree object together with the dependent blob and tree objects (i.e. a stored representation of a working tree).

tree object
: An object containing a list of file names and modes along with refs to the associated blob and/or tree objects. A tree is equivalent to a directory.

tree−ish
: A ref pointing to either a commit object, a tree object, or a tag object pointing to a tag or commit or tree object.

unmerged index
: An index which contains unmerged index entries.

unreachable object
: An object which is not reachable from a branch, tag, or any other reference.

upstream branch
: The default branch that is merged into the branch in question (or the branch in question is rebased onto). It is configured via branch.<name>.remote and branch.<name>.merge. If the upstream branch of *A* is *origin/B* sometimes we say "*A* is tracking *origin/B*".

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.

SEE ALSO

gittutorial(7), **gittutorial-2**(7), **gitcvs-migration**(7), **Everyday git**[1], **The Git User's Manual**[2]

GIT

Part of the **git**(1) suite.

NOTES

1. Everyday git
   file:///usr/share/doc/git-1.7.1/everyday.html
2. The Git User's Manual
   file:///usr/share/doc/git-1.7.1/user-manual.html