Guide to Git - Git Terms

In This Document

Overview

Git™ is a system that tracks and manages changes to files (a version control system). Whenever content changes, Git records it and stores the history of every change. Because of Git's complex functionality, it uses many terms that novice users may not immediately understand.

Common Git terms

> **Note:**
> Click a term to navigate to the official Git documentation for that concept or feature.

Archive

Archives store the contents of the current working tree (but not the `.git` directory or uncommitted changes) in a `.zip` or `.tar` file. You may wish to create an archive if, for example, you wish to provide a source download file.

Branch

Each branch in a repository represents a separate line of development, and all branches retain their own project history, working directory, and staging area.

- Each repository can contain as many branches as you wish to create.
- You can only work in one branch (the checked-out branch) at any given time.
- Generally, branches diverge from the original line of development with the intent to merge the branch's changes at a later time.

Check out

Git uses this term in two contexts:

- When you check out a branch or commit via the `git checkout` command, Git points HEAD to the specified branch or commit. This allows you to switch between multiple branches from the command line.
- When you check out files via the `git checkout` command, Git copies the version of that file from the specified commit or from the index. This allows you to revert committed or uncommitted changes.

Cherry-pick

When you cherry-pick changes via the `git cherry-pick` command, Git applies the specified changes from a commit and branch to a different branch's HEAD.

Clone

When you clone a repository via the `git clone` command, Git performs the following actions:

1. Git creates a new repository (the local repository) in the directory in which you ran the command.

   > **Note:**
   > When you clone a repository in cPanel's *Git Version Control* interface (*cPanel >> Home >> Files >> Git Version Control*), the system creates the repository in the *Repository Path* directory that you specify.

2. Git sets the repository that you wish to clone (the remote repository) as the `origin` remote repository.
3. Git fetches all of the commits and branches from the remote repository.
4. Git checks out the default branch (generally, the `master` branch).

Effectively, this copies the remote repository. You can then make changes to the local repository and push them to the remote repository.

Commit

Commits represent a point in Git's history. Git's entire history for a repository exists as a timeline of individual commits. When you commit changes, you create a new point in the history that represents the current state of the index. HEAD then points to the new commit.

Commit Object

Commit objects represent your committed revisions to a branch. Each commit object contains the commit's files (the tree object), parent commits, commit metadata (for example, the author and date), and a SHA-1 value that identifies the object.

Deployment

Deployment sends finished code into production. You can use different configurations to automatically (push deployment) or manually (pull deployment) deploy changes. For example, you can configure cPanel's *Git Version Control* feature (*cPanel >> Home >> Files >> Git Version Control*) to automatically deploy changes that a cPanel-managed repository receives to the directory for your website. For more information, read our Guide to Git - How to Set Up Deployment documentation.

Fetch

When you fetch changes via the `git fetch` command, Git automatically downloads new changes from the remote repository. However, it does **not** integrate (merge) these changes into the working tree for any local branch.

Fork

When you fork a repository, you create a new server-side copy of that repository. You can then experiment with changes to that repository without any impact on the original repository.

HEAD

The HEAD value represents the SHA-1 identifier for the most-recent commit or active branch. Whenever you commit changes to the active branch, Git automatically updates HEAD to that commit's SHA-1 identifier. If you use the `git checkout` command to check out a specific commit instead of a branch, Git enters the `detached HEAD` state.

Head

Heads are the SHA-1 identifiers for the most-recent commits to each branch. While only one HEAD commit exists, a repository generally contains many heads for each branch.

Hook

Hooks are scripts or other code that you can configure to trigger before or after specific Git actions. You can store these hooks in the `hooks` direc tory within the repository directory.

> **Note:**
> cPanel's *Git Version Control* feature (*cPanel >> Home >> Files >> Git Version Control*) automatically adds a `post-receive` hook to cPanel-managed repositories.

Index (Staging Area, Cache)

Indexes contain the files from your working tree that you add to a commit to the Git repository. Git also uses the index to store data during failed merges.

Log

The log contains the commit hash and commit metadata, such as the commit message, for every commit on the current branch. You can access this data via the `git log` command on the command line or via Gitweb in cPanel's *Git Version Control* interface (*cPanel >> Home >> Files >> Git Version Control*).

Master

Generally, the default branch for a repository is the `master` branch. When you commit changes to it, Git moves the `master` branch's HEAD to the most recent commit's SHA-1 identifier.

Merge

When you merge one or more commits, Git adds changes to the current branch. To perform a merge of this type, run the `git push` command.

You may also need to manually merge specific revisions if they conflict with changes that have already merged into the repository.

- This type of merge utilizes the `git merge` command.
- The term merge may also refer to the commit that this type of merge creates.

Origin

The origin is Git's default name for the remote repository from which you cloned a local repository. Most repositories include at least one origin repository. Software development often refers to origin as upstream.

Pull

When you pull changes from the remote repository via the `git pull` command, Git fetches remote changes and then merges them into the current branch.

> **Note:**
> You can use the *Pull from Remote* feature in cPanel's *Git Version Control* interface (*cPanel >> Home >> Files >> Git Version Control*) to automatically pull changes for a repository's active branch.

Push

When you push changes via the `git push` command, Git sends commits from your local branch to the remote repository.

Rebase

Rebases via the `git rebase` command reapply changes to the history of the active branch. To do this, rebases eliminate merge commits and create a new commit for each commit in the original branch.

Remote (Remote Repository)

The remote repository exists on a remote filesystem. When you fetch, pull, or push code, Git sends changes to or receives changes from the remote repository.

Repository

Repositories store all of the data that Git manages for a specific project. It contains objects and heads as well as the working tree.

SHA-1 (SHA-1 sum, Hash)

Technically, the algorithm that generates the names for Git objects. In Git's vernacular, this term also refers to the 40-character hexidecimal string that the algorithm generates.

Stash

An object that stores changes to the working tree and index for future reuse. The stash allows you to set aside changes to a branch and return to the HEAD state. You can then reapply the stashed changes or apply them to a different branch.

Version Control

Version control systems track changes in files and allow multiple users to coordinate those changes and view and manipulate the project's history. Git is a version control system.

Working Tree

The working tree contains the checked-out file system for a repository. The working tree includes the files for the HEAD commit and any local changes to those files.

Additional documentation

Suggested documentation For cPanel users For WHM users For developers

- Guide to Git - Deployment
- Guide to Git - For System Administrators
- Guide to Git
- Guide to Git - Common Git Commands

- Guide to Git - Host Git Repositories on a cPanel Account

Error rendering macro 'contentbylabel' : parameters should not be empty

Error rendering macro 'contentbylabel' : parameters should not be empty

Content by label

There is no content with the specified labels