Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'git-send-pack.1'

$ man git-send-pack.1

GIT-SEND-PACK(1) Git Manual GIT-SEND-PACK(1)

NAME

git-send-pack - Push objects over Git protocol to another repository

SYNOPSIS

git send-pack [--mirror] [--dry-run] [--force]

[--receive-pack=<git-receive-pack>]

[--verbose] [--thin] [--atomic]

[--[no-]signed | --signed=(true|false|if-asked)]

[<host>:]<directory> (--all | <ref>...)

DESCRIPTION

Usually you would want to use git push, which is a higher-level wrapper of this command, instead. See git-push(1).

Invokes git-receive-pack on a possibly remote repository, and updates it from the current repository, sending named refs.

OPTIONS

--receive-pack=<git-receive-pack>

Path to the git-receive-pack program on the remote end. Sometimes useful when pushing to a remote repository over ssh, and you do not have the program in a directory on the default $PATH.

--exec=<git-receive-pack>

Same as --receive-pack=<git-receive-pack>.

--all

Instead of explicitly specifying which refs to update, update all heads that locally

exist.

--stdin

Take the list of refs from stdin, one per line. If there are refs specified on the command line in addition to this option, then the refs from stdin are processed after those on the command line.

If --stateless-rpc is specified together with this option then the list of refs must be in packet format (pkt-line). Each ref must be in a separate packet, and the list must end with a flush packet.

--dry-run

Do everything except actually send the updates.

--force

Usually, the command refuses to update a remote ref that is not an ancestor of the local ref used to overwrite it. This flag disables the check. This means that the remote repository can lose commits; use it with care.

--verbose

Run verbosely.

--thin

Send a "thin" pack, which records objects in deltified form based on objects not included in the pack to reduce network traffic.

--atomic

Use an atomic transaction for updating the refs. If any of the refs fails to update then the entire push will fail without changing any refs.

--signed, --no-signed, --signed=(true|false|if-asked)

GPG-sign the push request to update refs on the receiving side, to allow it to be checked by the hooks and/or be logged. If false or --no-signed, no signing will be attempted. If true or --signed, the push will fail if the server does not support signed pushes. If set to if-asked, sign if and only if the server supports signed pushes. The push will also fail if the actual call to gpg --sign fails. See git-receive-pack(1) for the details on the receiving end.

--push-option=<string>

Pass the specified string as a push option for consumption by hooks on the server side. If the server doesn?t support push options, error out. See git-push(1) and githooks(5) for details.

<host>

A remote host to house the repository. When this part is specified, git-receive-pack is invoked via ssh.

<directory>

The repository to update.

<ref>...

The remote refs to update.

SPECIFYING THE REFS

There are three ways to specify which refs to update on the remote end.

With the --all flag, all refs that exist locally are transferred to the remote side. You cannot specify any <ref> if you use this flag.

Without --all and without any <ref>, the heads that exist both on the local side and on the remote side are updated.

When one or more <ref> are specified explicitly (whether on the command line or via --stdin), it can be either a single pattern, or a pair of such patterns separated by a colon ":" (this means that a ref name cannot have a colon in it). A single pattern <name> is just shorthand for <name>:<name>.

Each pattern pair consists of the source side (before the colon) and the destination side (after the colon). The ref to be pushed is determined by finding a match that matches the source side, and where it is pushed is determined by using the destination side. The rules used to match a ref are the same rules used by git rev-parse to resolve a symbolic ref name. See git-rev-parse(1).

? It is an error if <src> does not match exactly one of the local refs.

? It is an error if <dst> matches more than one remote ref.

? If <dst> does not match any remote ref, either

  ? it has to start with "refs/"; <dst> is used as the destination literally in this case.

  ? <src> == <dst> and the ref that matched the <src> must not exist in the set of remote refs; the ref matched <src> locally is used as the name of the destination.

Without --force, the <src> ref is stored at the remote only if <dst> does not exist, or <dst> is a proper subset (i.e. an ancestor) of <src>. This check, known as the "fast-forward check", is performed to avoid accidentally overwriting the remote ref and losing other people?s commits from there.

With --force, the fast-forward check is disabled for all refs.

Optionally, a <ref> parameter can be prefixed with a plus + sign to disable the fast-forward check only on that ref.

GIT

Part of the git(1) suite