**git-svn(1) Manual Page**

**NAME**

git-svn - Bidirectional operation between a Subversion repository and git

**SYNOPSIS**

*git svn* <command> [options] [arguments]

**DESCRIPTION**

*git svn* is a simple conduit for changesets between Subversion and git. It provides a bidirectional flow of changes between a Subversion and a git repository.

*git svn* can track a standard Subversion repository, following the common "trunk/branches/tags" layout, with the --stdlayout option. It can also follow branches and tags in any layout with the -T/-t/-b options (see options to *init* below, and also the *clone* command).

Once tracking a Subversion repository (with any of the above methods), the git repository can be updated from Subversion by the *fetch* command and Subversion updated from git by the *dcommit* command.

**COMMANDS**

*init*

Initializes an empty git repository with additional metadata directories for *git svn*. The Subversion URL may be specified as a command-line argument, or as full URL arguments to -T/-t/-b. Optionally, the target directory to operate on can be specified as a second argument. Normally this command initializes the current directory.

-T<trunk\_subdir>

--trunk=<trunk\_subdir>

-t<tags\_subdir>

--tags=<tags\_subdir>

-b<branches\_subdir>

--branches=<branches\_subdir>

-s

--stdlayout

These are optional command-line options for init. Each of these flags can point to a relative repository path (--tags=project/tags) or a full url (--tags=https://foo.org/project/tags). You can specify more than one --tags and/or --branches options, in case your Subversion repository places tags or branches under multiple paths. The option --stdlayout is a shorthand way of setting trunk,tags,branches as the relative paths, which is the Subversion default. If any of the other options are given as well, they take precedence.

--no-metadata

Set the *noMetadata* option in the [svn-remote] config. This option is not recommended, please read the *svn.noMetadata* section of this manpage before using this option.

--use-svm-props

Set the *useSvmProps* option in the [svn-remote] config.

--use-svnsync-props

Set the *useSvnsyncProps* option in the [svn-remote] config.

--rewrite-root=<URL>

Set the *rewriteRoot* option in the [svn-remote] config.

--rewrite-uuid=<UUID>

Set the *rewriteUUID* option in the [svn-remote] config.

--username=<user>

For transports that SVN handles authentication for (http, https, and plain svn), specify the username. For other transports (eg svn+ssh://), you must include the username in the URL, eg svn+ssh://foo@svn.bar.com/project

--prefix=<prefix>

This allows one to specify a prefix which is prepended to the names of remotes if trunk/branches/tags are specified. The prefix does not automatically include a trailing slash, so be sure you include one in the argument if that is what you want. If --branches/-b is specified, the prefix must include a trailing slash. Setting a prefix is useful if you wish to track multiple projects that share a common repository.

--ignore-paths=<regex>

When passed to *init* or *clone* this regular expression will be preserved as a config key. See *fetch* for a description of *--ignore-paths*.

--no-minimize-url

When tracking multiple directories (using --stdlayout, --branches, or --tags options), git svn will attempt to connect to the root (or highest allowed level) of the Subversion repository. This default allows better tracking of history if entire projects are moved within a repository, but may cause issues on repositories where read access restrictions are in place. Passing *--no-minimize-url* will allow git svn to accept URLs as-is without attempting to connect to a higher level directory. This option is off by default when only one URL/branch is tracked (it would do little good).

*fetch*

Fetch unfetched revisions from the Subversion remote we are tracking. The name of the [svn-remote "…"] section in the .git/config file may be specified as an optional command-line argument.

--localtime

Store Git commit times in the local timezone instead of UTC. This makes *git log* (even without --date=local) show the same times that svn log would in the local timezone.

This doesn’t interfere with interoperating with the Subversion repository you cloned from, but if you wish for your local Git repository to be able to interoperate with someone else’s local Git repository, either don’t use this option or you should both use it in the same local timezone.

--parent

Fetch only from the SVN parent of the current HEAD.

--ignore-paths=<regex>

This allows one to specify a Perl regular expression that will cause skipping of all matching paths from checkout from SVN. The *--ignore-paths* option should match for every *fetch* (including automatic fetches due to *clone*, *dcommit*, *rebase*, etc) on a given repository.

config key: svn-remote.<name>.ignore-paths

If the ignore-paths config key is set and the command line option is also given, both regular expressions will be used.

Examples:

Skip "doc\*" directory for every fetch

--ignore-paths="^doc"

Skip "branches" and "tags" of first level directories

--ignore-paths="^[^/]+/(?:branches|tags)"

*clone*

Runs *init* and *fetch*. It will automatically create a directory based on the basename of the URL passed to it; or if a second argument is passed; it will create a directory and work within that. It accepts all arguments that the *init* and *fetch* commands accept; with the exception of *--fetch-all*and *--parent*. After a repository is cloned, the *fetch* command will be able to update revisions without affecting the working tree; and the *rebase*command will be able to update the working tree with the latest changes.

*rebase*

This fetches revisions from the SVN parent of the current HEAD and rebases the current (uncommitted to SVN) work against it.

This works similarly to svn update or *git pull* except that it preserves linear history with *git rebase* instead of *git merge* for ease of dcommitting with *git svn*.

This accepts all options that *git svn fetch* and *git rebase* accept. However, *--fetch-all* only fetches from the current [svn-remote], and not all [svn-remote] definitions.

Like *git rebase*; this requires that the working tree be clean and have no uncommitted changes.

-l

--local

Do not fetch remotely; only run *git rebase* against the last fetched commit from the upstream SVN.

*dcommit*

Commit each diff from a specified head directly to the SVN repository, and then rebase or reset (depending on whether or not there is a diff between SVN and head). This will create a revision in SVN for each commit in git. It is recommended that you run *git svn* fetch and rebase (not pull or merge) your commits against the latest changes in the SVN repository. An optional revision or branch argument may be specified, and causes *git svn* to do all work on that revision/branch instead of HEAD. This is advantageous over *set-tree* (below) because it produces cleaner, more linear history.

--no-rebase

After committing, do not rebase or reset.

--commit-url <URL>

Commit to this SVN URL (the full path). This is intended to allow existing *git svn* repositories created with one transport method (e.g. svn:// or http:// for anonymous read) to be reused if a user is later given access to an alternate transport method (e.g. svn+ssh:// or https://) for commit.

config key: svn-remote.<name>.commiturl

config key: svn.commiturl (overwrites all svn-remote.<name>.commiturl options)

Using this option for any other purpose (don’t ask) is very strongly discouraged.

--mergeinfo=<mergeinfo>

Add the given merge information during the dcommit (e.g. --mergeinfo="/branches/foo:1-10"). All svn server versions can store this information (as a property), and svn clients starting from version 1.5 can make use of it. *git svn* currently does not use it and does not set it automatically.

*branch*

Create a branch in the SVN repository.

-m

--message

Allows to specify the commit message.

-t

--tag

Create a tag by using the tags\_subdir instead of the branches\_subdir specified during git svn init.

-d

--destination

If more than one --branches (or --tags) option was given to the *init* or *clone* command, you must provide the location of the branch (or tag) you wish to create in the SVN repository. The value of this option must match one of the paths specified by a --branches (or --tags) option. You can see these paths with the commands

git config --get-all svn-remote.<name>.branches

git config --get-all svn-remote.<name>.tags

where <name> is the name of the SVN repository as specified by the -R option to *init* (or "svn" by default).

--username

Specify the SVN username to perform the commit as. This option overrides the *username* configuration property.

--commit-url

Use the specified URL to connect to the destination Subversion repository. This is useful in cases where the source SVN repository is read-only. This option overrides configuration property *commiturl*.

git config --get-all svn-remote.<name>.commiturl

*tag*

Create a tag in the SVN repository. This is a shorthand for *branch -t*.

*log*

This should make it easy to look up svn log messages when svn users refer to -r/--revision numbers.

The following features from ‘svn log’ are supported:

-r <n>[:<n>]

--revision=<n>[:<n>]

is supported, non-numeric args are not: HEAD, NEXT, BASE, PREV, etc …

-v

--verbose

it’s not completely compatible with the --verbose output in svn log, but reasonably close.

--limit=<n>

is NOT the same as --max-count, doesn’t count merged/excluded commits

--incremental

supported

New features:

--show-commit

shows the git commit sha1, as well

--oneline

our version of --pretty=oneline

|  |  |
| --- | --- |
| **Note** | SVN itself only stores times in UTC and nothing else. The regular svn client converts the UTC time to the local time (or based on the TZ= environment). This command has the same behaviour. |

Any other arguments are passed directly to *git log*

*blame*

Show what revision and author last modified each line of a file. The output of this mode is format-compatible with the output of ‘svn blame’ by default. Like the SVN blame command, local uncommitted changes in the working copy are ignored; the version of the file in the HEAD revision is annotated. Unknown arguments are passed directly to *git blame*.

--git-format

Produce output in the same format as *git blame*, but with SVN revision numbers instead of git commit hashes. In this mode, changes that haven’t been committed to SVN (including local working-copy edits) are shown as revision 0.

*find-rev*

When given an SVN revision number of the form *rN*, returns the corresponding git commit hash (this can optionally be followed by a tree-ish to specify which branch should be searched). When given a tree-ish, returns the corresponding SVN revision number.

*set-tree*

You should consider using *dcommit* instead of this command. Commit specified commit or tree objects to SVN. This relies on your imported fetch data being up-to-date. This makes absolutely no attempts to do patching when committing to SVN, it simply overwrites files with those specified in the tree or commit. All merging is assumed to have taken place independently of *git svn* functions.

*create-ignore*

Recursively finds the svn:ignore property on directories and creates matching .gitignore files. The resulting files are staged to be committed, but are not committed. Use -r/--revision to refer to a specific revision.

*show-ignore*

Recursively finds and lists the svn:ignore property on directories. The output is suitable for appending to the $GIT\_DIR/info/exclude file.

*mkdirs*

Attempts to recreate empty directories that core git cannot track based on information in $GIT\_DIR/svn/<refname>/unhandled.log files. Empty directories are automatically recreated when using "git svn clone" and "git svn rebase", so "mkdirs" is intended for use after commands like "git checkout" or "git reset". (See the svn-remote.<name>.automkdirs config file option for more information.)

*commit-diff*

Commits the diff of two tree-ish arguments from the command-line. This command does not rely on being inside an git svn init-ed repository. This command takes three arguments, (a) the original tree to diff against, (b) the new tree result, (c) the URL of the target Subversion repository. The final argument (URL) may be omitted if you are working from a *git svn*-aware repository (that has been init-ed with *git svn*). The -r<revision> option is required for this.

*info*

Shows information about a file or directory similar to what ‘svn info’ provides. Does not currently support a -r/--revision argument. Use the --url option to output only the value of the *URL:* field.

*proplist*

Lists the properties stored in the Subversion repository about a given file or directory. Use -r/--revision to refer to a specific Subversion revision.

*propget*

Gets the Subversion property given as the first argument, for a file. A specific revision can be specified with -r/--revision.

*show-externals*

Shows the Subversion externals. Use -r/--revision to specify a specific revision.

*gc*

Compress $GIT\_DIR/svn/<refname>/unhandled.log files in .git/svn and remove $GIT\_DIR/svn/<refname>index files in .git/svn.

*reset*

Undoes the effects of *fetch* back to the specified revision. This allows you to re-*fetch* an SVN revision. Normally the contents of an SVN revision should never change and *reset* should not be necessary. However, if SVN permissions change, or if you alter your --ignore-paths option, a *fetch* may fail with "not found in commit" (file not previously visible) or "checksum mismatch" (missed a modification). If the problem file cannot be ignored forever (with --ignore-paths) the only way to repair the repo is to use *reset*.

Only the rev\_map and refs/remotes/git-svn are changed. Follow *reset* with a *fetch* and then *git reset* or *git rebase* to move local branches onto the new tree.

-r <n>

--revision=<n>

Specify the most recent revision to keep. All later revisions are discarded.

-p

--parent

Discard the specified revision as well, keeping the nearest parent instead.

Example:

Assume you have local changes in "master", but you need to refetch "r2".

r1---r2---r3 remotes/git-svn

\

A---B master

Fix the ignore-paths or SVN permissions problem that caused "r2" to be incomplete in the first place. Then:

git svn reset -r2 -p

git svn fetch

r1---r2'--r3' remotes/git-svn

\

r2---r3---A---B master

Then fixup "master" with *git rebase*. Do NOT use *git merge* or your history will not be compatible with a future *dcommit*!

git rebase --onto remotes/git-svn A^ master

r1---r2'--r3' remotes/git-svn

\

A'--B' master

**OPTIONS**

--shared[=(false|true|umask|group|all|world|everybody)]

--template=<template\_directory>

Only used with the *init* command. These are passed directly to *git init*.

-r <arg>

--revision <arg>

Used with the *fetch* command.

This allows revision ranges for partial/cauterized history to be supported. $NUMBER, $NUMBER1:$NUMBER2 (numeric ranges), $NUMBER:HEAD, and BASE:$NUMBER are all supported.

This can allow you to make partial mirrors when running fetch; but is generally not recommended because history will be skipped and lost.

-

--stdin

Only used with the *set-tree* command.

Read a list of commits from stdin and commit them in reverse order. Only the leading sha1 is read from each line, so *git rev-list --pretty=oneline* output can be used.

--rmdir

Only used with the *dcommit*, *set-tree* and *commit-diff* commands.

Remove directories from the SVN tree if there are no files left behind. SVN can version empty directories, and they are not removed by default if there are no files left in them. git cannot version empty directories. Enabling this flag will make the commit to SVN act like git.

config key: svn.rmdir

-e

--edit

Only used with the *dcommit*, *set-tree* and *commit-diff* commands.

Edit the commit message before committing to SVN. This is off by default for objects that are commits, and forced on when committing tree objects.

config key: svn.edit

-l<num>

--find-copies-harder

Only used with the *dcommit*, *set-tree* and *commit-diff* commands.

They are both passed directly to *git diff-tree*; see [git-diff-tree(1)](http://schacon.github.io/git/git-diff-tree.html) for more information.

config key: svn.l

config key: svn.findcopiesharder

-A<filename>

--authors-file=<filename>

Syntax is compatible with the file used by *git cvsimport*:

loginname = Joe User <user@example.com>

If this option is specified and *git svn* encounters an SVN committer name that does not exist in the authors-file, *git svn* will abort operation. The user will then have to add the appropriate entry. Re-running the previous *git svn* command after the authors-file is modified should continue operation.

config key: svn.authorsfile

--authors-prog=<filename>

If this option is specified, for each SVN committer name that does not exist in the authors file, the given file is executed with the committer name as the first argument. The program is expected to return a single line of the form "Name <email>", which will be treated as if included in the authors file.

-q

--quiet

Make *git svn* less verbose. Specify a second time to make it even less verbose.

--repack[=<n>]

--repack-flags=<flags>

These should help keep disk usage sane for large fetches with many revisions.

--repack takes an optional argument for the number of revisions to fetch before repacking. This defaults to repacking every 1000 commits fetched if no argument is specified.

--repack-flags are passed directly to *git repack*.

config key: svn.repack

config key: svn.repackflags

-m

--merge

-s<strategy>

--strategy=<strategy>

These are only used with the *dcommit* and *rebase* commands.

Passed directly to *git rebase* when using *dcommit* if a *git reset* cannot be used (see *dcommit*).

-n

--dry-run

This can be used with the *dcommit*, *rebase*, *branch* and *tag* commands.

For *dcommit*, print out the series of git arguments that would show which diffs would be committed to SVN.

For *rebase*, display the local branch associated with the upstream svn repository associated with the current branch and the URL of svn repository that will be fetched from.

For *branch* and *tag*, display the urls that will be used for copying when creating the branch or tag.

--use-log-author

When retrieving svn commits into git (as part of *fetch*, *rebase*, or *dcommit* operations), look for the first From: or Signed-off-by: line in the log message and use that as the author string.

--add-author-from

When committing to svn from git (as part of *commit-diff*, *set-tree* or *dcommit* operations), if the existing log message doesn’t already have aFrom: or Signed-off-by: line, append a From: line based on the git commit’s author string. If you use this, then --use-log-author will retrieve a valid author string for all commits.

**ADVANCED OPTIONS**

-i<GIT\_SVN\_ID>

--id <GIT\_SVN\_ID>

This sets GIT\_SVN\_ID (instead of using the environment). This allows the user to override the default refname to fetch from when tracking a single URL. The *log* and *dcommit* commands no longer require this switch as an argument.

-R<remote name>

--svn-remote <remote name>

Specify the [svn-remote "<remote name>"] section to use, this allows SVN multiple repositories to be tracked. Default: "svn"

--follow-parent

This is especially helpful when we’re tracking a directory that has been moved around within the repository, or if we started tracking a branch and never tracked the trunk it was descended from. This feature is enabled by default, use --no-follow-parent to disable it.

config key: svn.followparent

**CONFIG FILE-ONLY OPTIONS**

svn.noMetadata

svn-remote.<name>.noMetadata

This gets rid of the *git-svn-id:* lines at the end of every commit.

This option can only be used for one-shot imports as *git svn* will not be able to fetch again without metadata. Additionally, if you lose your .git/svn/**\*/.rev\_map.** files, *git svn* will not be able to rebuild them.

The *git svn log* command will not work on repositories using this, either. Using this conflicts with the *useSvmProps* option for (hopefully) obvious reasons.

This option is NOT recommended as it makes it difficult to track down old references to SVN revision numbers in existing documentation, bug reports and archives. If you plan to eventually migrate from SVN to git and are certain about dropping SVN history, consider [git-filter-branch(1)](http://schacon.github.io/git/git-filter-branch.html) instead. filter-branch also allows reformatting of metadata for ease-of-reading and rewriting authorship info for non-"svn.authorsFile" users.

svn.useSvmProps

svn-remote.<name>.useSvmProps

This allows *git svn* to re-map repository URLs and UUIDs from mirrors created using SVN::Mirror (or svk) for metadata.

If an SVN revision has a property, "svm:headrev", it is likely that the revision was created by SVN::Mirror (also used by SVK). The property contains a repository UUID and a revision. We want to make it look like we are mirroring the original URL, so introduce a helper function that returns the original identity URL and UUID, and use it when generating metadata in commit messages.

svn.useSvnsyncProps

svn-remote.<name>.useSvnsyncprops

Similar to the useSvmProps option; this is for users of the svnsync(1) command distributed with SVN 1.4.x and later.

svn-remote.<name>.rewriteRoot

This allows users to create repositories from alternate URLs. For example, an administrator could run *git svn* on the server locally (accessing via file://) but wish to distribute the repository with a public http:// or svn:// URL in the metadata so users of it will see the public URL.

svn-remote.<name>.rewriteUUID

Similar to the useSvmProps option; this is for users who need to remap the UUID manually. This may be useful in situations where the original UUID is not available via either useSvmProps or useSvnsyncProps.

svn-remote.<name>.pushurl

Similar to git’s *remote.<name>.pushurl*, this key is designed to be used in cases where *url* points to an SVN repository via a read-only transport, to provide an alternate read/write transport. It is assumed that both keys point to the same repository. Unlike *commiturl*, *pushurl* is a base path. If either *commiturl* or *pushurl* could be used, *commiturl* takes precedence.

svn.brokenSymlinkWorkaround

This disables potentially expensive checks to workaround broken symlinks checked into SVN by broken clients. Set this option to "false" if you track a SVN repository with many empty blobs that are not symlinks. This option may be changed while *git svn* is running and take effect on the next revision fetched. If unset, *git svn* assumes this option to be "true".

svn.pathnameencoding

This instructs git svn to recode pathnames to a given encoding. It can be used by windows users and by those who work in non-utf8 locales to avoid corrupted file names with non-ASCII characters. Valid encodings are the ones supported by Perl’s Encode module.

svn-remote.<name>.automkdirs

Normally, the "git svn clone" and "git svn rebase" commands attempt to recreate empty directories that are in the Subversion repository. If this option is set to "false", then empty directories will only be created if the "git svn mkdirs" command is run explicitly. If unset, *git svn* assumes this option to be "true".

Since the noMetadata, rewriteRoot, rewriteUUID, useSvnsyncProps and useSvmProps options all affect the metadata generated and used by *git svn*; they **must** be set in the configuration file before any history is imported and these settings should never be changed once they are set.

Additionally, only one of these options can be used per svn-remote section because they affect the *git-svn-id:* metadata line, except for rewriteRoot and rewriteUUID which can be used together.

**BASIC EXAMPLES**

Tracking and contributing to the trunk of a Subversion-managed project:

# Clone a repo (like git clone):

git svn clone http://svn.example.com/project/trunk

# Enter the newly cloned directory:

cd trunk

# You should be on master branch, double-check with 'git branch'

git branch

# Do some work and commit locally to git:

git commit ...

# Something is committed to SVN, rebase your local changes against the

# latest changes in SVN:

git svn rebase

# Now commit your changes (that were committed previously using git) to SVN,

# as well as automatically updating your working HEAD:

git svn dcommit

# Append svn:ignore settings to the default git exclude file:

git svn show-ignore >> .git/info/exclude

Tracking and contributing to an entire Subversion-managed project (complete with a trunk, tags and branches):

# Clone a repo (like git clone):

git svn clone http://svn.example.com/project -T trunk -b branches -t tags

# View all branches and tags you have cloned:

git branch -r

# Create a new branch in SVN

git svn branch waldo

# Reset your master to trunk (or any other branch, replacing 'trunk'

# with the appropriate name):

git reset --hard remotes/trunk

# You may only dcommit to one branch/tag/trunk at a time. The usage

# of dcommit/rebase/show-ignore should be the same as above.

The initial *git svn clone* can be quite time-consuming (especially for large Subversion repositories). If multiple people (or one person with multiple machines) want to use *git svn* to interact with the same Subversion repository, you can do the initial *git svn clone* to a repository on a server and have each person clone that repository with *git clone*:

# Do the initial import on a server

ssh server "cd /pub && git svn clone http://svn.example.com/project

# Clone locally - make sure the refs/remotes/ space matches the server

mkdir project

cd project

git init

git remote add origin server:/pub/project

git config --replace-all remote.origin.fetch '+refs/remotes/\*:refs/remotes/\*'

git fetch

# Prevent fetch/pull from remote git server in the future,

# we only want to use git svn for future updates

git config --remove-section remote.origin

# Create a local branch from one of the branches just fetched

git checkout -b master FETCH\_HEAD

# Initialize 'git svn' locally (be sure to use the same URL and -T/-b/-t options as were used on server)

git svn init http://svn.example.com/project

# Pull the latest changes from Subversion

git svn rebase

**REBASE VS. PULL/MERGE**

Originally, *git svn* recommended that the *remotes/git-svn* branch be pulled or merged from. This is because the author favored git svn set-tree B to commit a single head rather than the git svn set-tree A..B notation to commit multiple commits.

If you use git svn set-tree A..B to commit several diffs and you do not have the latest remotes/git-svn merged into my-branch, you should use git svn rebase to update your work branch instead of git pull or git merge. pull/merge can cause non-linear history to be flattened when committing into SVN, which can lead to merge commits reversing previous commits in SVN.

**MERGE TRACKING**

While *git svn* can track copy history (including branches and tags) for repositories adopting a standard layout, it cannot yet represent merge history that happened inside git back upstream to SVN users. Therefore it is advised that users keep history as linear as possible inside git to ease compatibility with SVN (see the CAVEATS section below).

**CAVEATS**

For the sake of simplicity and interoperating with Subversion, it is recommended that all *git svn* users clone, fetch and dcommit directly from the SVN server, and avoid all *git clone*/*pull*/*merge*/*push* operations between git repositories and branches. The recommended method of exchanging code between git branches and users is *git format-patch* and *git am*, or just 'dcommit’ing to the SVN repository.

Running *git merge* or *git pull* is NOT recommended on a branch you plan to *dcommit* from because Subversion users cannot see any merges you’ve made. Furthermore, if you merge or pull from a git branch that is a mirror of an SVN branch, *dcommit* may commit to the wrong branch.

If you do merge, note the following rule: *git svn dcommit* will attempt to commit on top of the SVN commit named in

git log --grep=^git-svn-id: --first-parent -1

You *must* therefore ensure that the most recent commit of the branch you want to dcommit to is the *first* parent of the merge. Chaos will ensue otherwise, especially if the first parent is an older commit on the same SVN branch.

*git clone* does not clone branches under the refs/remotes/ hierarchy or any *git svn* metadata, or config. So repositories created and managed with using *git svn* should use *rsync* for cloning, if cloning is to be done at all.

Since *dcommit* uses rebase internally, any git branches you *git push* to before *dcommit* on will require forcing an overwrite of the existing ref on the remote repository. This is generally considered bad practice, see the [git-push(1)](http://schacon.github.io/git/git-push.html) documentation for details.

Do not use the --amend option of [git-commit(1)](http://schacon.github.io/git/git-commit.html) on a change you’ve already dcommitted. It is considered bad practice to --amend commits you’ve already pushed to a remote repository for other users, and dcommit with SVN is analogous to that.

When using multiple --branches or --tags, *git svn* does not automatically handle name collisions (for example, if two branches from different paths have the same name, or if a branch and a tag have the same name). In these cases, use *init* to set up your git repository then, before your first *fetch*, edit the .git/config file so that the branches and tags are associated with different name spaces. For example:

branches = stable/\*:refs/remotes/svn/stable/\*

branches = debug/\*:refs/remotes/svn/debug/\*

**BUGS**

We ignore all SVN properties except svn:executable. Any unhandled properties are logged to $GIT\_DIR/svn/<refname>/unhandled.log

Renamed and copied directories are not detected by git and hence not tracked when committing to SVN. I do not plan on adding support for this as it’s quite difficult and time-consuming to get working for all the possible corner cases (git doesn’t do it, either). Committing renamed and copied files is fully supported if they’re similar enough for git to detect them.

**CONFIGURATION**

*git svn* stores [svn-remote] configuration information in the repository .git/config file. It is similar the core git [remote] sections except *fetch* keys do not accept glob arguments; but they are instead handled by the *branches* and *tags* keys. Since some SVN repositories are oddly configured with multiple projects glob expansions such those listed below are allowed:

[svn-remote "project-a"]

url = http://server.org/svn

fetch = trunk/project-a:refs/remotes/project-a/trunk

branches = branches/\*/project-a:refs/remotes/project-a/branches/\*

tags = tags/\*/project-a:refs/remotes/project-a/tags/\*

Keep in mind that the *\** (asterisk) wildcard of the local ref (right of the *:*) **must** be the farthest right path component; however the remote wildcard may be anywhere as long as it’s an independent path component (surrounded by */* or EOL). This type of configuration is not automatically created by *init* and should be manually entered with a text-editor or using *git config*.

It is also possible to fetch a subset of branches or tags by using a comma-separated list of names within braces. For example:

[svn-remote "huge-project"]

url = http://server.org/svn

fetch = trunk/src:refs/remotes/trunk

branches = branches/{red,green}/src:refs/remotes/branches/\*

tags = tags/{1.0,2.0}/src:refs/remotes/tags/\*

Note that git-svn keeps track of the highest revision in which a branch or tag has appeared. If the subset of branches or tags is changed after fetching, then .git/svn/.metadata must be manually edited to remove (or reset) branches-maxRev and/or tags-maxRev as appropriate.

**SEE ALSO**

[git-rebase(1)](http://schacon.github.io/git/git-rebase.html)

**GIT**

Part of the [git(1)](http://schacon.github.io/git/git.html) suite

Last updated 2011-05-05 01:29:54 UTC