git-cvsimport

git-cvsimport(1) ==========

NAME ---- git-cvsimport - Salvage your data out of another SCM people love to hate

SYNOPSIS --------

> 'git cvsimport' [-o <branch-for-HEAD>] [-h] [-v] [-d <CVSROOT>]
> [-A <author-conv-file>] [-p <options-for-cvsps>] [-P <file>]
> [-C <git-repository>] [-z <fuzz>] [-i] [-k] [-u] [-s <subst>]
> [-a] [-m] [-M <regex>] [-S <regex>] [-L <commit-limit>]
> [-r <remote>] [-R] [<CVS-module>]

DESCRIPTION ----------- **WARNING:** `git` `cvsimport` uses cvsps version 2, which is considered deprecated; it does not work with cvsps version 3 and later. If you are performing a one-shot import of a CVS repository consider using cvs2git or cvs-fast-export.

Imports a CVS repository into Git. It will either create a new repository, or incrementally import into an existing one.

Splitting the CVS log into patch sets is done by 'cvsps'. At least version 2.1 is required.

WARNING: for certain situations the import leads to incorrect results. Please see the section ISSUES for further reference.

You should **never** do any work of your own on the branches that are created by 'git cvsimport'. By default initial import will create and populate a "master" branch from the CVS repository's main branch which you're free to work with; after that, you need to 'git merge' incremental imports, or any CVS branches, yourself. It is advisable to specify a named remote via -r to separate and protect the incoming branches.

If you intend to set up a shared public repository that all developers can read/write, or if you want to use linkgit:git-cvsserver[1], then you probably want to make a bare clone of the imported repository, and use the clone as the shared repository. See linkgit:gitcvs-migration[7].

OPTIONS ------- -v:: Verbosity: let 'cvsimport' report what it is doing.

-d <CVSROOT>
The root of the CVS archive. May be local (a simple path) or remote; currently, only the :local:, :ext:

and :pserver: access methods are supported. If not given, ’git cvsimport’ will try to read it from `CVS/Root`. If no such file exists, it checks for the `CVSROOT` environment variable.

<CVS-module>
The CVS module you want to import. Relative to <CVSROOT>. If not given, ’git cvsimport’ tries to read it from `CVS/Repository`.

-C <target-dir>
The Git repository to import to. If the directory doesn’t exist, it will be created. Default is the current directory.

-r <remote>
The Git remote to import this CVS repository into. Moves all CVS branches into remotes/<remote>/<branch> akin to the way ’git clone’ uses ’origin’ by default.

-o <branch-for-HEAD>
When no remote is specified (via -r) the `HEAD` branch from CVS is imported to the ’origin’ branch within the Git repository, as `HEAD` already has a special meaning for Git. When a remote is specified the `HEAD` branch is named remotes/<remote>/master mirroring ’git clone’ behaviour. Use this option if you want to import into a different branch.

Use ’-o master’ for continuing an import that was initially done by the old cvs2git tool.

-i
Import-only: don’t perform a checkout after importing. This option ensures the working directory and index remain untouched and will not create them if they do not exist.

-k
Kill keywords: will extract files with ’-kk’ from the CVS archive to avoid noisy changesets. Highly recommended, but off by default to preserve compatibility with early imported trees.

-u
Convert underscores in tag and branch names to dots.

-s <subst>
Substitute the character "/" in branch names with <subst>

-p <options-for-cvsps>
Additional options for cvsps. The options `-u` and ’-A’ are implicit and should not be used here.

If you need to pass multiple options, separate them with a comma.

-z <fuzz>
Pass the timestamp fuzz factor to cvsps, in seconds. If unset, cvsps defaults to 300s.

-P <cvsps-output-file>
Instead of calling cvsps, read the provided cvsps output file. Useful for debugging or when cvsps is

being handled outside cvsimport.

-m
Attempt to detect merges based on the commit message. This option will enable default regexes that try to capture the source branch name from the commit message.

-M <regex>
Attempt to detect merges based on the commit message with a custom regex. It can be used with `-m` to enable the default regexes as well. You must escape forward slashes.

The regex must capture the source branch name in $1.

This option can be used several times to provide several detection regexes.

-S <regex>
Skip paths matching the regex.

-a
Import all commits, including recent ones. cvsimport by default skips commits that have a timestamp less than 10 minutes ago.

-L <limit>
Limit the number of commits imported. Workaround for cases where cvsimport leaks memory.

-A <author-conv-file>
CVS by default uses the Unix username when writing its commit logs. Using this option and an author-conv-file maps the name recorded in CVS to author name, e-mail and optional time zone:

```
exon=Andreas Ericsson <ae@op5.se>
spawn=Simon Pawn <spawn@frog-pond.org> America/Chicago
```

'git cvsimport' will make it appear as those authors had their GIT_AUTHOR_NAME and GIT_AUTHOR_EMAIL set properly all along. If a time zone is specified, GIT_AUTHOR_DATE will have the corresponding offset applied.

For convenience, this data is saved to `$GIT_DIR/cvs-authors` each time the '-A' option is provided and read from that same file each time 'git cvsimport' is run.

It is not recommended to use this feature if you intend to export changes back to CVS again later with 'git cvsexportcommit'.

-R
Generate a `$GIT_DIR/cvs-revisions` file containing a mapping from CVS revision numbers to newly-created Git commit IDs. The generated file will contain one line for each (filename, revision) pair imported; each line will look like

```
src/widget.c 1.1 1d862f173cdc7325b6fa6d2ae1cfd61fd1b512b7
```

The revision data is appended to the file if it already exists, for use when doing incremental imports.

This option may be useful if you have CVS revision numbers stored in commit messages, bug-tracking systems, email archives, and the like.

-h
Print a short usage message and exit.

OUTPUT ------ If `-v` is specified, the script reports what it is doing.

Otherwise, success is indicated the Unix way, i.e. by simply exiting with a zero exit status.

ISSUES ------ Problems related to timestamps:

- If timestamps of commits in the CVS repository are not stable enough to be used for ordering commits changes may show up in the wrong order.
- If any files were ever "cvs import"ed more than once (e.g., import of more than one vendor release) the HEAD contains the wrong content.
- If the timestamp order of different files cross the revision order within the commit matching time window the order of commits may be wrong.

Problems related to branches:

- Branches on which no commits have been made are not imported.
- All files from the branching point are added to a branch even if never added in CVS.
- This applies to files added to the source branch **after** a daughter branch was created: if previously no commit was made on the daughter branch they will erroneously be added to the daughter branch in git.

Problems related to tags:

- Multiple tags on the same revision are not imported.

If you suspect that any of these issues may apply to the repository you want to import, consider using cvs2git:

- cvs2git (part of cvs2svn), `https://subversion.apache.org/`

GIT --- Part of the linkgit:git[1] suite

□□□ #1
□ □□□ □□□ 2026-10-31 20:10:29 UTC
□ □□□ □□□ 2026-10-31 20:10:29 UTC