NAME

git-archive - Create an archive of files from a named tree

SYNOPSIS

git archive [--format=<fmt>] [--list] [--prefix=<prefix>/] [<extra>]
[-o <file> | --output=<file>] [--worktree-attributes]
[--remote=<repo> [--exec=<git-upload-archive>]] <tree-ish>
[<path>...]

DESCRIPTION

Creates an archive of the specified format containing the tree structure for the named tree, and writes it out to the standard output. If <prefix> is specified it is prepended to the filenames in the archive.

git archive behaves differently when given a tree ID versus when given a commit ID or tag ID. In the first case the current time is used as the modification time of each file in the archive. In the latter case the commit time as recorded in the referenced commit object is used instead. Additionally the commit ID is stored in a global extended pax header if the tar format is used; it can be extracted using *git get-tar-commit-id*. In ZIP files it is stored as a file comment.

OPTIONS

--format=<fmt>
: Format of the resulting archive. Possible values are **tar**, **zip**, **tar.gz**, **tgz**, and any format defined using the configuration option **tar.<format>.command**. If **--format** is not given, and the output file is specified, the format is inferred from the filename if possible (e.g. writing to **foo.zip** makes the output to be in the **zip** format). Otherwise the output format is **tar**.

-l, --list
: Show all available formats.

-v, --verbose
: Report progress to stderr.

--prefix=<prefix>/
: Prepend <prefix>/ to paths in the archive. Can be repeated; its rightmost value is used for all tracked files. See below which value gets used by **--add-file** and **--add-virtual-file**.

-o <file>, --output=<file>
: Write the archive to <file> instead of stdout.

--add-file=<file>
: Add a non-tracked file to the archive. Can be repeated to add multiple files. The path of the file in the archive is built by concatenating the value of the last **--prefix** option (if any) before this **--add-file** and the basename of <file>.

--add-virtual-file=<path>:<content>
: Add the specified contents to the archive. Can be repeated to add multiple files. The path of the file in the archive is built by concatenating the value of the last **--prefix** option (if any) before this **--add-virtual-file** and **<path>**.

  The **<path>** argument can start and end with a literal double-quote character; the contained file name is interpreted as a C-style string, i.e. the backslash is interpreted as escape character. The path must be quoted if it contains a colon, to avoid the colon from being misinterpreted as the separator between the path and the contents, or if the path begins or ends with a double-quote character.

  The file mode is limited to a regular file, and the option may be subject to platform-dependent command-line limits. For non-trivial cases, write an untracked file and use **--add-file** instead.

--worktree-attributes
: Look for attributes in .gitattributes files in the working tree as well (see the section called "ATTRIBUTES").

--mtime=<time>
: Set modification time of archive entries. Without this option the committer time is used if **<tree-ish>** is a commit or tag, and the current time if it is a tree.

<extra>
: This can be any options that the archiver backend understands. See next section.

--remote=<repo>
: Instead of making a tar archive from the local repository, retrieve a tar archive from a remote repository. Note that the remote repository may place restrictions on which sha1 expressions may be allowed in **<tree-ish>**. See **git-upload-archive**(1) for details.

--exec=<git-upload-archive>
: Used with --remote to specify the path to the *git-upload-archive* on the remote side.

<tree-ish>
: The tree or commit to produce an archive for.

<path>

Without an optional path parameter, all files and subdirectories of the current working directory are included in the archive. If one or more paths are specified, only these are included.

BACKEND EXTRA OPTIONS

zip

-<digit>

Specify compression level. Larger values allow the command to spend more time to compress to smaller size. Supported values are from **-0** (store-only) to **-9** (best ratio). Default is **-6** if not given.

tar

-<number>

Specify compression level. The value will be passed to the compression command configured in **tar.<format>.command**. See manual page of the configured command for the list of supported levels and the default level if this option isn't specified.

CONFIGURATION

tar.umask

This variable can be used to restrict the permission bits of tar archive entries. The default is 0002, which turns off the world write bit. The special value "user" indicates that the archiving user's umask will be used instead. See umask(2) for details. If **--remote** is used then only the configuration of the remote repository takes effect.

tar.<format>.command

This variable specifies a shell command through which the tar output generated by **git archive** should be piped. The command is executed using the shell with the generated tar file on its standard input, and should produce the final output on its standard output. Any compression-level options will be passed to the command (e.g., **-9**).

The **tar.gz** and **tgz** formats are defined automatically and use the magic command **git archive gzip** by default, which invokes an internal implementation of gzip.

tar.<format>.remote

If true, enable the format for use by remote clients via **git-upload-archive**(1). Defaults to false for user-defined formats, but true for the **tar.gz** and **tgz** formats.

ATTRIBUTES

export-ignore

Files and directories with the attribute export-ignore won't be added to archive files. See **gitattributes**(5) for details.

export-subst

If the attribute export-subst is set for a file then Git will expand several placeholders when adding this file to an archive. See **gitattributes**(5) for details.

Note that attributes are by default taken from the **.gitattributes** files in the tree that is being archived. If you want to tweak the way the output is generated after the fact (e.g. you committed without adding an appropriate export-ignore in its **.gitattributes**), adjust the checked out **.gitattributes** file as necessary and use **--worktree-attributes** option. Alternatively you can keep necessary attributes that should apply while archiving any tree in your **$GIT_DIR/info/attributes** file.

EXAMPLES

git archive --format=tar --prefix=junk/ HEAD | (cd /var/tmp/ && tar xf -)

Create a tar archive that contains the contents of the latest commit on the current branch, and extract it in the **/var/tmp/junk** directory.

git archive --format=tar --prefix=git-1.4.0/ v1.4.0 | gzip >git-1.4.0.tar.gz

Create a compressed tarball for v1.4.0 release.

git archive --format=tar.gz --prefix=git-1.4.0/ v1.4.0 >git-1.4.0.tar.gz

Same as above, but using the builtin tar.gz handling.

git archive --prefix=git-1.4.0/ -o git-1.4.0.tar.gz v1.4.0

Same as above, but the format is inferred from the output file.

git archive --format=tar --prefix=git-1.4.0/ v1.4.0^{tree} | gzip >git-1.4.0.tar.gz

Create a compressed tarball for v1.4.0 release, but without a global extended pax header.

git archive --format=zip --prefix=git-docs/ HEAD:Documentation/ > git-1.4.0-docs.zip

Put everything in the current head’s Documentation/ directory into *git-1.4.0-docs.zip*, with the prefix *git-docs/*.

git archive -o latest.zip HEAD

Create a Zip archive that contains the contents of the latest commit on the current branch. Note that the output format is inferred by the extension of the output file.

git archive -o latest.tar --prefix=build/ --add-file=configure --prefix= HEAD

Creates a tar archive that contains the contents of the latest commit on the current branch with no prefix and the untracked file *configure* with the prefix *build/*.

git config tar.tar.xz.command "xz -c"

Configure a "tar.xz" format for making LZMA-compressed tarfiles. You can use it specifying **--format=tar.xz**, or by creating an output file like **-o foo.tar.xz**.

SEE ALSO

gitattributes(5)

GIT

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