NAME

git-pack-objects - Create a packed archive of objects

SYNOPSIS

git pack-objects [-q | --progress | --all-progress] [--all-progress-implied]
[--no-reuse-delta] [--delta-base-offset] [--non-empty]
[--local] [--incremental] [--window=<n>] [--depth=<n>]
[--revs [--unpacked | --all]] [--stdout | base-name]
[--keep-true-parents] < object-list

DESCRIPTION

Reads list of objects from the standard input, and writes a packed archive with specified base-name, or to the standard output.

A packed archive is an efficient way to transfer a set of objects between two repositories as well as an access efficient archival format. In a packed archive, an object is either stored as a compressed whole or as a difference from some other object. The latter is often called a delta.

The packed archive format (.pack) is designed to be self-contained so that it can be unpacked without any further information. Therefore, each object that a delta depends upon must be present within the pack.

A pack index file (.idx) is generated for fast, random access to the objects in the pack. Placing both the index file (.idx) and the packed archive (.pack) in the pack/ subdirectory of $GIT_OBJECT_DIRECTORY (or any of the directories on $GIT_ALTERNATE_OBJECT_DIRECTORIES) enables Git to read from the pack archive.

The *git unpack-objects* command can read the packed archive and expand the objects contained in the pack into one-file one-object format; this is typically done by the smart-pull commands when a pack is created on-the-fly for efficient network transport by their peers.

OPTIONS

base-name
    Write into a pair of files (.pack and .idx), using <base-name> to determine the name of the created file. When this option is used, the two files are written in <base-name>-<SHA-1>.{pack,idx} files. <SHA-1> is a hash based on the pack content and is written to the standard output of the command.

--stdout
    Write the pack contents (what would have been written to .pack file) out to the standard output.

--revs
    Read the revision arguments from the standard input, instead of individual object names. The revision arguments are processed the same way as *git rev-list* with the --objects flag uses its commit arguments to build the list of objects it outputs. The objects on the resulting list are packed. Besides revisions, --not or --shallow <SHA-1> lines are also accepted.

--unpacked
    This implies --revs. When processing the list of revision arguments read from the standard input, limit the objects packed to those that are not already packed.

--all
    This implies --revs. In addition to the list of revision arguments read from the standard input, pretend as if all refs under refs/ are specified to be included.

--include-tag
    Include unasked-for annotated tags if the object they reference was included in the resulting packfile. This can be useful to send new tags to native Git clients.

--window=<n>, --depth=<n>

These two options affect how the objects contained in the pack are stored using delta compression. The objects are first internally sorted by type, size and optionally names and compared against the other objects within --window to see if using delta compression saves space. --depth limits the maximum delta depth; making it too deep affects the performance on the unpacker side, because delta data needs to be applied that many times to get to the necessary object. The default value for --window is 10 and --depth is 50.

--window-memory=<n>
> This option provides an additional limit on top of --window; the window size will dynamically scale down so as to not take up more than <*n*> bytes in memory. This is useful in repositories with a mix of large and small objects to not run out of memory with a large window, but still be able to take advantage of the large window for the smaller objects. The size can be suffixed with k, m, or g. --window-memory=0 makes memory usage unlimited, which is the default.

--max-pack-size=<n>
> Maximum size of each output pack file. The size can be suffixed with k, m, or g. The minimum size allowed is limited to 1 MiB. If specified, multiple packfiles may be created. The default is unlimited, unless the config variable pack.packSizeLimit is set.

--honor-pack-keep
> This flag causes an object already in a local pack that has a .keep file to be ignored, even if it would have otherwise been packed.

--incremental
> This flag causes an object already in a pack to be ignored even if it would have otherwise been packed.

--local
> This flag causes an object that is borrowed from an alternate object store to be ignored even if it would have otherwise been packed.

--non-empty
> Only create a packed archive if it would contain at least one object.

--progress
> Progress status is reported on the standard error stream by default when it is attached to a terminal, unless -q is specified. This flag forces progress status even if the standard error stream is not directed to a terminal.

--all-progress
> When --stdout is specified then progress report is displayed during the object count and compression phases but inhibited during the write-out phase. The reason is that in some cases the output stream is directly linked to another command which may wish to display progress status of its own as it processes incoming pack data. This flag is like --progress except that it forces progress report for the write-out phase as well even if --stdout is used.

--all-progress-implied
> This is used to imply --all-progress whenever progress display is activated. Unlike --all-progress this flag doesn't actually force any progress display by itself.

-q
> This flag makes the command not to report its progress on the standard error stream.

--no-reuse-delta
> When creating a packed archive in a repository that has existing packs, the command reuses existing deltas. This sometimes results in a slightly suboptimal pack. This flag tells the command not to reuse existing deltas but compute them from scratch.

--no-reuse-object
> This flag tells the command not to reuse existing object data at all, including non deltified object, forcing recompression of everything. This implies --no-reuse-delta. Useful only in the

obscure case where wholesale enforcement of a different compression level on the packed data is desired.

--compression=<n>
>   Specifies compression level for newly-compressed data in the generated pack. If not specified, pack compression level is determined first by pack.compression, then by core.compression, and defaults to -1, the zlib default, if neither is set. Add --no-reuse-object if you want to force a uniform compression level on all data no matter the source.

--thin
>   Create a thin pack by omitting the common objects between a sender and a receiver in order to reduce network transfer. This option only makes sense in conjunction with --stdout.

>   Note: A thin pack violates the packed archive format by omitting required objects and is thus unusable by Git without making it self-contained. Use git index-pack --fix-thin (see **git-index-pack(1)**) to restore the self-contained property.

--delta-base-offset
>   A packed archive can express the base object of a delta as either a 20-byte object name or as an offset in the stream, but ancient versions of Git don't understand the latter. By default, *git pack-objects* only uses the former format for better compatibility. This option allows the command to use the latter format for compactness. Depending on the average delta chain length, this option typically shrinks the resulting packfile by 3-5 per-cent.

>   Note: Porcelain commands such as git gc (see **git-gc(1)**), git repack (see **git-repack(1)**) pass this option by default in modern Git when they put objects in your repository into pack files. So does git bundle (see **git-bundle(1)**) when it creates a bundle.

--threads=<n>
>   Specifies the number of threads to spawn when searching for best delta matches. This requires that pack-objects be compiled with pthreads otherwise this option is ignored with a warning. This is meant to reduce packing time on multiprocessor machines. The required amount of memory for the delta search window is however multiplied by the number of threads. Specifying 0 will cause Git to auto-detect the number of CPU's and set the number of threads accordingly.

--index-version=<version>[,<offset>]
>   This is intended to be used by the test suite only. It allows to force the version for the generated pack index, and to force 64-bit index entries on objects located above the given offset.

--keep-true-parents
>   With this option, parents that are hidden by grafts are packed nevertheless.

SEE ALSO
git-rev-list(1) git-repack(1) git-prune-packed(1)

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