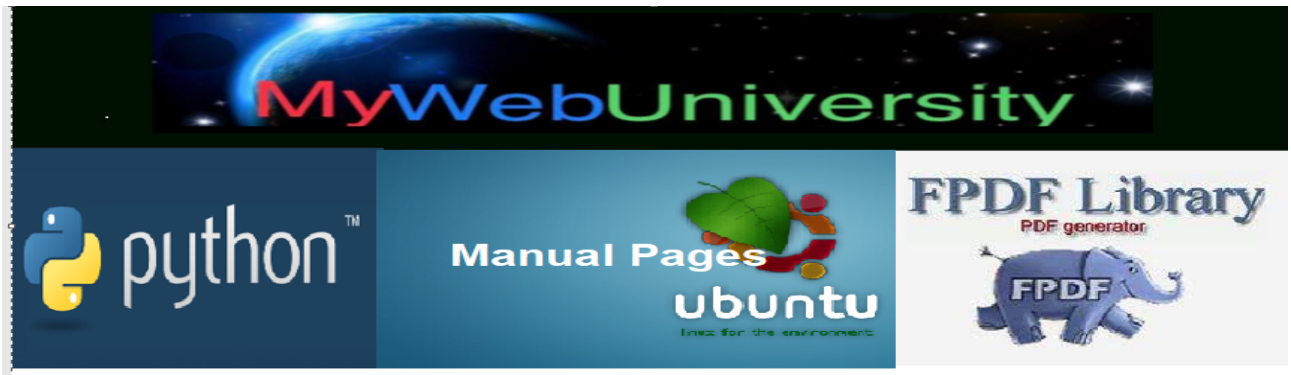


***Full credit is given to the above companies including the OS that this PDF file was generated!***

## *Linux Ubuntu 26.04 Manual Pages on command 'git-repack.1'*

***$ man git-repack.1***

GIT-REPACK(1) Git Manual GIT-REPACK(1)

NAME

git-repack - Pack unpacked objects in a repository

SYNOPSIS

git repack [-a] [-A] [-d] [-f] [-F] [-l] [-n] [-q] [-b] [-m]

[--window=<n>] [--depth=<n>] [--threads=<n>] [--keep-pack=<pack-name>]

[--write-midx] [--name-hash-version=<n>] [--path-walk]

DESCRIPTION

This command is used to combine all objects that do not currently reside in a "pack", into a pack. It can also be used to re-organize existing packs into a single, more efficient pack.

A pack is a collection of objects, individually compressed, with delta compression applied, stored in a single file, with an associated index file.

Packs are used to reduce the load on mirror systems, backup engines, disk storage, etc.

OPTIONS

-a

Instead of incrementally packing the unpacked objects, pack everything referenced into a single pack. Especially useful when packing a repository that is used for private development. Use with -d. This will clean up the objects that git prune leaves behind, but git fsck --full --dangling shows as dangling.

Note that users fetching over dumb protocols will have to fetch the whole new pack in order to get any contained object, no matter how many other objects in that pack they already have locally.

Promisor packfiles are repacked separately: if there are packfiles that have an

associated ".promisor" file, these packfiles will be repacked into another separate pack, and an empty ".promisor" file corresponding to the new separate pack will be written.

-A

Same as -a, unless -d is used. Then any unreachable objects in a previous pack become loose, unpacked objects, instead of being left in the old pack. Unreachable objects are never intentionally added to a pack, even when repacking. This option prevents unreachable objects from being immediately deleted by way of being left in the old pack and then removed. Instead, the loose unreachable objects will be pruned according to normal expiry rules with the next git gc invocation. See git-gc(1).

-d

After packing, if the newly created packs make some existing packs redundant, remove the redundant packs. Also run git prune-packed to remove redundant loose object files.

--cruft

Same as -a, unless -d is used. Then any unreachable objects are packed into a separate cruft pack. Unreachable objects can be pruned using the normal expiry rules with the next git gc invocation (see git-gc(1)). Incompatible with -k.

--cruft-expiration=<approxidate>

Expire unreachable objects older than <approxidate> immediately instead of waiting for the next git gc invocation. Only useful with --cruft -d.

--max-cruft-size=<n>

Override --max-pack-size for cruft packs. Inherits the value of --max-pack-size (if any) by default. See the documentation for --max-pack-size for more details.

--combine-cruft-below-size=<n>

When generating cruft packs without pruning, only repack existing cruft packs whose size is strictly less than <n> bytes, which can optionally be suffixed with "k", "m", or "g". Cruft packs whose size is greater than or equal to <n> are left as-is and not repacked. Useful when you want to avoid repacking large cruft pack(s) in repositories that have many and/or large unreachable objects.

--expire-to=<dir>

Write a cruft pack containing pruned objects (if any) to the directory <dir>. This option is useful for keeping a copy of any pruned objects in a separate directory as a backup. Only useful with --cruft -d.

-l

Pass the --local option to git pack-objects. See git-pack-objects(1).

-f

Pass the --no-reuse-delta option to git-pack-objects, see git-pack-objects(1).

-F

Pass the --no-reuse-object option to git-pack-objects, see git-pack-objects(1).

-q, --quiet

Show no progress over the standard error stream and pass the -q option to git pack-objects. See git-pack-objects(1).

-n

Do not update the server information with git update-server-info. This option skips updating local catalog files needed to publish this repository (or a direct copy of it) over HTTP or FTP. See git-update-server-info(1).

--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. The maximum depth is 4095.

--threads=<n>

This option is passed through to git pack-objects.

--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. The default is taken from the pack.windowMemory configuration variable. Note that the actual memory usage will be the limit multiplied by the number of threads used by git-pack-objects(1).

--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, which also prevents the creation of a bitmap index. The default is unlimited, unless the config variable pack.packSizeLimit is set. Note that this option may result in a larger and slower repository; see the discussion in pack.packSizeLimit.

--filter=<filter-spec>

Remove objects matching the filter specification from the resulting packfile and put them into a separate packfile. Note that objects used in the working directory are not filtered out. So for the split to fully work, it?s best to perform it in a bare repo and to use the -a and -d options along with this option. Also --no-write-bitmap-index (or the repack.writebitmaps config option set to false) should be used otherwise writing bitmap index will fail, as it supposes a single packfile containing all the objects. See git-rev-list(1) for valid <filter-spec> forms.

--filter-to=<dir>

Write the pack containing filtered out objects to the directory <dir>. Only useful with --filter. This can be used for putting the pack on a separate object directory that is accessed through the Git alternates mechanism.  WARNING: If the packfile containing the filtered out objects is not accessible, the repo can become corrupt as it might not be possible to access the objects in that packfile. See the objects and objects/info/alternates sections of gitrepository-layout(5).

-b, --write-bitmap-index

Write a reachability bitmap index as part of the repack. This only makes sense when used with -a, -A or -m, as the bitmaps must be able to refer to all reachable objects. This option overrides the setting of repack.writeBitmaps. This option has no effect if multiple packfiles are created, unless writing a MIDX (in which case a multi-pack bitmap is created).

--pack-kept-objects

Include objects in .keep files when repacking. Note that we still do not delete .keep packs after pack-objects finishes. This means that we may duplicate objects, but this makes the option safe to use when there are concurrent pushes or fetches. This option is generally only useful if you are writing bitmaps with -b or repack.writeBitmaps, as it ensures that the bitmapped packfile has the necessary objects.

--keep-pack=<pack-name>

Exclude the given pack from repacking. This is the equivalent of having .keep file on the pack. <pack-name> is the pack file name without leading directory (e.g. pack-123.pack). The option can be specified multiple times to keep multiple packs.

--unpack-unreachable=<when>

When loosening unreachable objects, do not bother loosening any objects older than <when>. This can be used to optimize out the write of any objects that would be immediately pruned by a follow-up git prune.

-k, --keep-unreachable

When used with -ad, any unreachable objects from existing packs will be appended to the end of the packfile instead of being removed. In addition, any unreachable loose objects will be packed (and their loose counterparts removed).

-i, --delta-islands

Pass the --delta-islands option to git-pack-objects, see git-pack-objects(1).

-g<factor>, --geometric=<factor>

Arrange resulting pack structure so that each successive pack contains at least <factor> times the number of objects as the next-largest pack.

git repack ensures this by determining a "cut" of packfiles that need to be repacked into one in order to ensure a geometric progression. It picks the smallest set of packfiles such that as many of the larger packfiles (by count of objects contained in that pack) may be left intact.

Unlike other repack modes, the set of objects to pack is determined uniquely by the set of packs being "rolled-up"; in other words, the packs determined to need to be combined in order to restore a geometric progression.

Loose objects are implicitly included in this "roll-up", without respect to their reachability. This is subject to change in the future.

When writing a multi-pack bitmap, git repack selects the largest resulting pack as the preferred pack for object selection by the MIDX (see git-multi-pack-index(1)).

-m, --write-midx

Write a multi-pack index (see git-multi-pack-index(1)) containing the non-redundant packs.

--name-hash-version=<n>

Provide this argument to the underlying git pack-objects process. See git-pack-objects(1) for full details.

--path-walk

Pass the --path-walk option to the underlying git pack-objects process. See git-pack-objects(1) for full details.

## CONFIGURATION

Various configuration variables affect packing, see git-config(1) (search for "pack" and "delta").

By default, the command passes --delta-base-offset option to git pack-objects; this typically results in slightly smaller packs, but the generated packs are incompatible with versions of Git older than version 1.4.4. If you need to share your repository with such ancient Git versions, either directly or via the dumb http protocol, then you need to set the configuration variable repack.UseDeltaBaseOffset to "false" and repack. Access from old Git versions over the native protocol is unaffected by this option as the conversion is performed on the fly as needed in that case.

Delta compression is not used on objects larger than the core.bigFileThreshold configuration variable and on files with the attribute delta set to false.

## SEE ALSO

git-pack-objects(1) git-prune-packed(1)

## GIT

Part of the git(1) suite