git-commit-graph(1) Manual Page

NAME

git-commit-graph - Write and verify Git commit-graph files

SYNOPSIS

git commit-graph verify [--object-dir <dir>] [--shallow] [--[no-]progress]
git commit-graph write <options> [--object-dir <dir>] [--[no-]progress]

DESCRIPTION

Manage the serialized commit-graph file.

OPTIONS

--object-dir

Use given directory for the location of packfiles and commit-graph file. This parameter exists to specify the location of an alternate that only has the objects directory, not a full `.git` directory. The commit-graph file is expected to be in the `<dir>/info` directory and the packfiles are expected to be in `<dir>/pack`. If the directory could not be made into an absolute path, or does not match any known object directory, `git commit-graph ...` will exit with non-zero status.

--[no-]progress

Turn progress on/off explicitly. If neither is specified, progress is shown if standard error is connected to a terminal.

COMMANDS

write

Write a commit-graph file based on the commits found in packfiles.

With the `--stdin-packs` option, generate the new commit graph by walking objects only in the specified pack-indexes. (Cannot be combined with `--stdin-commits` or `--reachable`.)

With the `--stdin-commits` option, generate the new commit graph by walking commits starting at the commits specified in stdin as a list of OIDs in hex, one OID per line. OIDs that resolve to non-commits (either directly, or by peeling tags) are silently ignored. OIDs that are malformed, or do not exist generate an error. (Cannot be combined with `--stdin-packs` or `--reachable`.)

With the `--reachable` option, generate the new commit graph by walking commits starting at all refs. (Cannot be combined with `--stdin-commits` or `--stdin-packs`.)

With the `--append` option, include all commits that are present in the existing commit-graph file.

With the `--changed-paths` option, compute and write information about the paths changed between a commit and its first parent. This operation can take a while on large repositories. It provides significant performance gains for getting history of a directory or a file with `git log -- <path>`. If this option is given, future commit-graph writes will automatically assume that this option was intended. Use `--no-changed-paths` to stop storing this data.

With the `--max-new-filters=<n>` option, generate at most `n` new Bloom filters (if `--changed-paths` is specified). If `n` is `-1`, no limit is enforced. Only commits present in the new layer count against this limit. To retroactively compute Bloom filters over earlier layers, it is advised to use `--split=replace`. Overrides the `commitGraph.maxNewFilters` configuration.

With the `--split[=<strategy>]` option, write the commit-graph as a chain of multiple commit-graph files stored in `<dir>/info/commit-graphs`. Commit-graph layers are merged based on the strategy and other splitting options. The new commits not

already in the commit-graph are added in a new "tip" file. This file is merged with the existing file if the following merge conditions are met:

- If `--split=no-merge` is specified, a merge is never performed, and the remaining options are ignored. `--split=replace` overwrites the existing chain with a new one. A bare `--split` defers to the remaining options. (Note that merging a chain of commit graphs replaces the existing chain with a length-1 chain where the first and only incremental holds the entire graph).
- If `--size-multiple=<X>` is not specified, let `X` equal 2. If the new tip file would have `N` commits and the previous tip has `M` commits and `X` times `N` is greater than `M`, instead merge the two files into a single file.
- If `--max-commits=<M>` is specified with `M` a positive integer, and the new tip file would have more than `M` commits, then instead merge the new tip with the previous tip.

  Finally, if `--expire-time=<datetime>` is not specified, let `datetime` be the current time. After writing the split commit-graph, delete all unused commit-graph whose modified times are older than `datetime`.

verify

Read the commit-graph file and verify its contents against the object database. Used to check for corrupted data.

With the `--shallow` option, only check the tip commit-graph file in a chain of split commit-graphs.

EXAMPLES

- Write a commit-graph file for the packed commits in your local `.git` directory.

  ```
  $ git commit-graph write
  ```

- Write a commit-graph file, extending the current commit-graph file using commits in `<pack-index>`.

  ```
  $ echo <pack-index> | git commit-graph write --stdin-packs
  ```

- Write a commit-graph file containing all reachable commits.

```
$ git show-ref -s | git commit-graph write --stdin-commits
```

- Write a commit-graph file containing all commits in the current commit-graph file along with those reachable from `HEAD`.

```
$ git rev-parse HEAD | git commit-graph write --stdin-commits --append
```