pg_checksums

- Description
- Options
- Environment
- Notes

pg_checksums

pg_checksums

1

Application

pg_checksums

enable, disable or check data checksums in a

PostgreSQL

database cluster

pg_checksums

option

-D

--pgdata

datadir

Description

pg_checksums checks, enables or disables data checksums in a PostgreSQL cluster. The server must be shut down cleanly before running pg_checksums. Checksums can also be enabled while

the cluster is running using ??? processing, see ??? for further details on the different approaches. When verifying checksums, the exit status is zero if there are no checksum errors, and nonzero if at least one checksum failure is detected. When enabling or disabling checksums, the exit status is nonzero if the operation failed.

If checksums were in the process of being enabled or disabled using ??? when the cluster was shut down, the control file still records that interrupted state, and pg_checksums refuses to run in any mode. Start the cluster and shut it down cleanly to reset the state, then retry; on a standby, let replication complete the transition first.

When verifying checksums, every file in the cluster is scanned. When enabling checksums, each relation file block with a changed checksum is rewritten in-place. Disabling checksums only updates the file `pg_control`.

Options

The following command-line options are available:

`-D datadir`; `--pgdata=datadir`
Specifies the directory where the database cluster is stored.

`-c`; `--check`
Checks checksums. This is the default mode if nothing else is specified.

`-d`; `--disable`
Disables checksums.

`-e`; `--enable`
Enables checksums.

`-f filenode`; `--filenode=filenode`
Only validate checksums in the relation with filenode <filenode>.

`-N`; `--no-sync`
By default, `pg_checksums` will wait for all files to be written safely to disk. This option causes `pg_checksums` to return without waiting, which is faster, but means that a subsequent operating system crash can leave the updated data directory corrupt. Generally, this option is useful for testing but should not be used on a production installation. This option has no effect when using `--check`.

`-P`; `--progress`
Enable progress reporting. Turning this on will deliver a progress report while checking or enabling checksums.

`--sync-method=method`
When set to `fsync`, which is the default, `pg_checksums` will recursively open and synchronize all files in the data directory. The search for files will follow symbolic links for the WAL directory and each configured tablespace.

On Linux, `syncfs` may be used instead to ask the operating system to synchronize the whole file systems that contain the data directory, the WAL files, and each tablespace. See ??? for information about the caveats to be aware of when using `syncfs`.

This option has no effect when `--no-sync` is used.

`-v`; `--verbose`
Enable verbose output. Lists all checked files.

`-V`; `--version`
Print the pg_checksums version and exit.

`-?`; `--help`
Show help about pg_checksums command line arguments, and exit.

Environment

`PGDATA`
Specifies the directory where the database cluster is stored; can be overridden using the `-D` option.

`PG_COLOR`
Specifies whether to use color in diagnostic messages. Possible values are `always`, `auto` and `never`.

Notes

Enabling checksums in a large cluster can potentially take a long time. During this operation, the cluster or other programs that write to the data directory must not be started or else data loss may occur.

Enabling or disabling checksums with pg_checksums changes only the local data directory; the new state is not replicated to any other node. In a replication setup the same change must be applied to every node:

1. All nodes participating in the replication must be stopped with a clean shutdown; pg_checksums refuses to run on a data directory left behind by an immediate shutdown. Before stopping a standby, make sure it has replayed all WAL of the primary: stop the primary first, read its “Latest checkpoint location” with ???, and check that `pg_last_wal_replay_lsn()` on the standby has advanced past it. Comparing the replay position with `pg_last_wal_receive_lsn()` is not enough, as it only shows that the WAL the standby received has been replayed.
2. Run pg_checksums on the data directory of each node in the replication setup. Nodes can be processed in parallel while they are shut down. Processing must complete successfully on all nodes before continuing.
3. Start the nodes normally, verify that ??? matches on all of them, and monitor the logs of the standbys for data checksum state mismatch warnings.

The replay requirement in step_title exists because an offline change is recorded only in the control file and has no defined ordering against WAL the node has not replayed yet; see ???. A node stopped before replaying an online checksum state change applies that change when it is restarted, overriding the offline change, and the states of the nodes silently diverge until a later checkpoint record triggers the warning described below. Because of this it is best not to mix the two mechanisms: change the state of a replication setup either with the offline procedure above or with an online transition, and make sure the previous change has reached every node before starting the next one.

If the change is applied inconsistently, each node keeps its own state, and a standby logs a warning when the state recorded in the replayed WAL differs from its own. The same warning can appear transiently while a standby catches up over WAL written before a consistent change; it stops once a checkpoint record carrying the new state has been replayed.

A standby whose data directory was never checksummed must not have checksums enabled by catching up this way. Converge the cluster by running pg_checksums on it while stopped as outlined above, by enabling checksums online, or by recreating it from a base backup. Note that an online enable only starts from a primary whose checksums are off, so if they are already enabled there, disable them online first.

If pg_checksums is aborted or killed while enabling or disabling checksums, the cluster's data checksum configuration remains unchanged, and pg_checksums can be re-run to perform the same operation.

Tools that copy relation file blocks directly between nodes, such as ???, require all nodes to be in the same data checksum state, else there is risk for data corruption.

The target cluster must have the same major version as pg_checksums.

□□□ #2
□ □□□ □□□ 2026-10-31 14:13:09 UTC
□ □□□ □□□ 2026-10-31 14:13:10 UTC