NAME

pg_verifybackup - verify the integrity of a base backup of a PostgreSQL cluster

SYNOPSIS

pg_verifybackup [*option*...]

DESCRIPTION

pg_verifybackup is used to check the integrity of a database cluster backup taken using **pg_basebackup** against a backup_manifest generated by the server at the time of the backup. The backup must be stored in the "plain" format; a "tar" format backup can be checked after extracting it.

It is important to note that the validation which is performed by pg_verifybackup does not and cannot include every check which will be performed by a running server when attempting to make use of the backup. Even if you use this tool, you should still perform test restores and verify that the resulting databases work as expected and that they appear to contain the correct data. However, pg_verifybackup can detect many problems that commonly occur due to storage problems or user error.

Backup verification proceeds in four stages. First, pg_verifybackup reads the backup_manifest file. If that file does not exist, cannot be read, is malformed, or fails verification against its own internal checksum, pg_verifybackup will terminate with a fatal error.

Second, pg_verifybackup will attempt to verify that the data files currently stored on disk are exactly the same as the data files which the server intended to send, with some exceptions that are described below. Extra and missing files will be detected, with a few exceptions. This step will ignore the presence or absence of, or any modifications to, postgresql.auto.conf, standby.signal, and recovery.signal, because it is expected that these files may have been created or modified as part of the process of taking the backup. It also won’t complain about a backup_manifest file in the target directory or about anything inside pg_wal, even though these files won’t be listed in the backup manifest. Only files are checked; the presence or absence of directories is not verified, except indirectly: if a directory is missing, any files it should have contained will necessarily also be missing.

Next, pg_verifybackup will checksum all the files, compare the checksums against the values in the manifest, and emit errors for any files for which the computed checksum does not match the checksum stored in the manifest. This step is not performed for any files which produced errors in the previous step, since they are already known to have problems. Files which were ignored in the previous step are also ignored in this step.

Finally, pg_verifybackup will use the manifest to verify that the write-ahead log records which will be needed to recover the backup are present and that they can be read and parsed. The backup_manifest contains information about which write-ahead log records will be needed, and pg_verifybackup will

use that information to invoke pg_waldump to parse those write-ahead log records. The --quiet flag will be used, so that pg_waldump will only report errors, without producing any other output. While this level of verification is sufficient to detect obvious problems such as a missing file or one whose internal checksums do not match, they aren't extensive enough to detect every possible problem that might occur when attempting to recover. For instance, a server bug that produces write-ahead log records that have the correct checksums but specify nonsensical actions can't be detected by this method.

Note that if extra WAL files which are not required to recover the backup are present, they will not be checked by this tool, although a separate invocation of pg_waldump could be used for that purpose. Also note that WAL verification is version-specific: you must use the version of pg_verifybackup, and thus of pg_waldump, which pertains to the backup being checked. In contrast, the data file integrity checks should work with any version of the server that generates a backup_manifest file.

OPTIONS

pg_verifybackup accepts the following command-line arguments:

-e
--exit-on-error
: Exit as soon as a problem with the backup is detected. If this option is not specified, pg_verifybackup will continue checking the backup even after a problem has been detected, and will report all problems detected as errors.

-i *path*
--ignore=*path*
: Ignore the specified file or directory, which should be expressed as a relative path name, when comparing the list of data files actually present in the backup to those listed in the backup_manifest file. If a directory is specified, this option affects the entire subtree rooted at that location. Complaints about extra files, missing files, file size differences, or checksum mismatches will be suppressed if the relative path name matches the specified path name. This option can be specified multiple times.

-m *path*
--manifest-path=*path*
: Use the manifest file at the specified path, rather than one located in the root of the backup directory.

-n
--no-parse-wal
: Don't attempt to parse write-ahead log data that will be needed to recover from this backup.

-q
--quiet
Don’t print anything when a backup is successfully verified.

-s
--skip-checksums
Do not verify data file checksums. The presence or absence of files and the sizes of those files will still be checked. This is much faster, because the files themselves do not need to be read.

-w *path*
--wal-directory=*path*
Try to parse WAL files stored in the specified directory, rather than in pg_wal. This may be useful if the backup is stored in a separate location from the WAL archive.

Other options are also available:

-V
--version
Print the pg_verifybackup version and exit.

-?
--help
Show help about pg_verifybackup command line arguments, and exit.

EXAMPLES

To create a base backup of the server at mydbserver and verify the integrity of the backup:

```
$ pg_basebackup -h mydbserver -D /usr/local/pgsql/data
$ pg_verifybackup /usr/local/pgsql/data
```

To create a base backup of the server at mydbserver, move the manifest somewhere outside the backup directory, and verify the backup:

```
$ pg_basebackup -h mydbserver -D /usr/local/pgsql/backup1234
$ mv /usr/local/pgsql/backup1234/backup_manifest /my/secure/location/backup_manifest.1234
$ pg_verifybackup -m /my/secure/location/backup_manifest.1234 /usr/local/pgsql/backup1234
```

To verify a backup while ignoring a file that was added manually to the backup directory, and also skipping checksum verification:

```
$ pg_basebackup -h mydbserver -D /usr/local/pgsql/data
$ edit /usr/local/pgsql/data/note.to.self
$ pg_verifybackup --ignore=note.to.self --skip-checksums /usr/local/pgsql/data
```

SEE ALSO

pg_basebackup(1)