[TUTORIAL] [Backup] Cluster config (pmxcfs) - /etc/pve

forum.proxmox.com/threads/backup-cluster-config-pmxcfs-etc-pve.154569

esi_y September 18, 2024

Backup

A no-nonsense way to safely backup your `/etc/pve` files (*pmxcfs* [1]) is actually very simple:

Bash:

```
sqlite3 /var/lib/pve-cluster/config.db .dump > ~/config.dump.$(date --utc
+%Z%Y%m%d%H%M%S).sql
```

This is *safe to execute on a running node* and is only necessary on any single node of the cluster, the results (at specific point in time) will be exactly the same.

Obviously, it makes more sense to save this somewhere else than the home directory `~`, especially if you have dependable shared storage *off the cluster*. Ideally, you want a *systemd timer*, *cron job* or a *hook* to your other favourite backup method launching this.

Recovery

You will ideally never need to recover from this backup. In case of single node's corrupt config database, you are best off to copy over `/var/lib/pve-cluster/config.db` (while inactive) from a healthy node and let the implantee catch up with the cluster.

However, *failing everything else*, you will want to stop cluster service, put aside the (possibly) corrupt database and get the last good state back:

Bash:

```
systemctl stop pve-cluster
killall pmxcfs
mv /var/lib/pve-cluster/config.db{,.corrupt}
sqlite3 /var/lib/pve-cluster/config.db < ~/config.dump.<timestamp>.sql
systemctl start pve-cluster
```

NOTE: Any leftover WAL will be ignored.

---

[1] https://pve.proxmox.com/wiki/Proxmox_Cluster_File_System_(pmxcfs)

Additional notes on SQLite CLI

The `.dump` command [1] reads the database as if with a SELECT statement within a single transaction. It will block concurrent writes, but once it finishes, you have a "snapshot". The result is a perfectly valid SQL set of commands to recreate your database.

There's an alternative `.save` command (equivalent to `.backup`), it would produce a valid copy of the actual `.db` file, and while it is non-blocking copying the base page by page, if they get dirty in the process, the process needs to start over. You could receive `Error: database is locked` failure on the attempt. If you insist on this method, you may need to append `.timeout <milliseconds>` to get more luck with it.

Another option yet would be to use `VACUUM` command with an `INTO` clause [2], but it does not*fsync* the result on its own!

If you *already* have a corrupt `.db` file at hand (and nothing better), you may try your luck with `.recover` [3].

[1] https://www.sqlite.org/cli.html#converting_an_entire_database_to_a_text_file
[2] https://www.sqlite.org/lang_vacuum.html
[3] https://www.sqlite.org/cli.html#recover_data_from_a_corrupted_database

Last edited: Sep 19, 2024

NOTE: I will likely expand this stub in the future, just wanted to have it floating here for easier reference.

NB Not sure why this is not suggested in the official docs since a while.

ALSO! Any feedback welcome, especially negative (as it makes the resulting piece of advice better)!

Reactions: waltar

Sep 20, 2024

> tismo said:
> Does this guide still apply if I only have one node?
> Thank you for putting up this guide by the way.
>
> Click to expand...

It surely does, the only difference is that the `config.db` is capturing file changes from either one or many nodes (in a cluster), it is the backend database holding the contents of the filesystem mounted into `/etc/pve` at runtime. So this is why it is also sufficient to backup configuration this way on any node of the cluster because it's replicated. In a single node install, it's not replicated, but everything still resides in that same `config.db` backed database.

Reactions: tismo
Oct 3, 2024

Related mini-tutorial on partial recovery of only some of the backed up files was added here:
https://forum.proxmox.com/threads/155374/