Backup Best Practices (Draft)

David Steele
Crunchy Data

November 16, 2017

Agenda

Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is, of course, a challenge.

Why Backup?

- Hardware Failure:
    - No amount of redundancy can prevent it.
- Replication:
    - WAL archive for when async streaming gets behind.
    - Sync replica from backup instead of master.
- Corruption:
    - Can be caused by hardware or software.
    - Detection is, of course, a challenge.

Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is, of course, a challenge.

Why Backup?

- Hardware Failure:
    - No amount of redundancy can prevent it.
- Replication:
    - WAL archive for when async streaming gets behind.
    - Sync replica from backup instead of master.
- Corruption:
    - Can be caused by hardware or software.
    - Detection is, of course, a challenge.

Why Backup?

- Hardware Failure:
    - No amount of redundancy can prevent it.
- Replication:
    - WAL archive for when async streaming gets behind.
    - Sync replica from backup instead of master.
- Corruption:
    - Can be caused by hardware or software.
    - Detection is, of course, a challenge.

Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

Why Backup?

- Accidents:
    - So you dropped a table?
    - Deleted your most important account?
- Development:
    - No more realistic data than production!
    - May not be practical due to size / privacy issues.
- Reporting:
    - Use backups to standup an independent reporting server.
    - Recover important data that was removed on purpose.

Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

Schrödingers Backup

The state of any backup is unknown until a restore is attempted.

Making Backups Useful

- Find a way to use your backups
  - Syncing / New Replicas
  - Offline reporting
  - Offline data archiving
  - Development
- Unused code paths will not work when you need them unless they are tested
  - Regularly scheduled automated failover using backups to restore the old primary
  - Regularly scheduled disaster recovery (during a maintenance window if possible) to test restore techniques

How to Backup?

- pg_dump
- pg_basebackup
- pgBackRest!

Backup Types

- Full

  A complete copy of the database.
- Incremental

  Copy only files that have changed since the last backup.
- Differential

  Like an incremental, but only copy files that have changed since the last **full** backup.

Backup Retention

A good default retention is:

```
    retention-full=5
    retention-diff=3
```

Run the full backup once a week and differential backups the other six days. With this schedule there will always be at least four weeks of backups to work with.

Backup Retention

A minimal retention is:

```
retention-full=2
retention-diff=1
```

Run the full backup once a week and differentials the other six days. With this schedule there will

always be at least a week of backups to work with. This not enough for many use cases and this retention should be considered "degraded" and only used temporarily until more space is allocated.

Checksums

When initializing a new cluster always specify the `-k` option.
This will enable page checksums which pgBackRest will verify on every backup. If corruption is detected early it can often be corrected.
If corruption is detected, don't panic! As long as older non-corrupted backups exist there are options.

Restore

Multi-processing can lead to dramatic reductions in restore time and network utilization. Use the `--delta` option with the `restore` command when time is of the essence.
Double-check where you are restoring. pgBackRest will refuse to overwrite a *running* cluster but it will happily overwrite a shutdown down cluster when `--delta` is used.

Archiving

Asynchronous parallel archiving allows compression and transfer to be offloaded to another process which maintains continuous connections to the remote server, improving throughput significantly.

A critical feature for databases with extremely high write volume.

Archiving

Asynchronous parallel archiving allows compression and transfer to be offloaded to another process which maintains continuous connections to the remote server, improving throughput significantly.

A critical feature for databases with extremely high write volume.

Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

Don’t Panic!

A failed backup should not be cause for panic. Ideally, there are a number of backups that can be used for a restore.

Remember:

Backup + WAL = Recoverable Database
OLDER Backup + WAL = Recoverable Database

Now Panic!

If Backup + WAL = Recoverable Database, then then a failing archive command means you won't be able to do a complete restore even if the backups have been successful.

Worse, the disk could fill up and cause PostgreSQL to panic and stop.

Even if the replica is up to date it is not a backup!

Questions?

website: `http://www.pgbackrest.org`

email: david@pgbackrest.org
email: david@crunchydata.com

releases: `https://github.com/pgbackrest/pgbackrest/releases`

slides & demo: `https://github.com/dwsteele/conference/releases`