Backup Best Practices (Draft)

David Steele
Crunchy Data

November 16, 2017

Agenda

1. Why and How to Backup
2. Best Practices
3. Questions?

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.

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.

Checksums

When initializing a new cluster always specify the `-k` option.
This will enable page checksums which pgBackRest will verify on every backup.

Delta Restore

- Backup manifest contains checksum and size for every file.
- On delta restore all files not present in the backup or with a different size are removed from PGDATA.
- The remaining files are checksummed and only files with a checksum mismatch are restored.
- Multi-processing can lead to dramatic reductions in restore time and network utilization.

Parallel Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous parallel archiving allows compression and transfer to be offloaded to another process which maintains continuous connections to the remote server, improving throughput significantly.
    - 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.

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`