Upgrade or Migrate Your PostgreSQL Database With The Least Possible Downtime

Avinash Vallarapu
Percona
Agenda

- Upgrade checklist
- Methods available to upgrade with and without downtime
- Demonstration
Pre-Upgrade Checklist

● Plan your hardware specifications
● Application to DB connectivity
● High Availability
● Performance testing
● Backup strategy
● Plan your postgresql.conf parameters
● Install all the required tools and extensions in advance
Methods Available to Upgrade Legacy PostgreSQL

Using `pg_dumpall`

Using `pg_dump/pg_restore`

Using logical replication

Using `Slony-I`

Using `pg_upgrade`
Downtime?

- May involve a huge downtime
  - `pg_dumpall`
  - `pg_dump` and `pg_restore`

- May not involve a huge downtime
  - *Logical replication or pg_logical*
  - Slony-I
  - `pg_upgrade` with hard links.
pg_dumpall

- Text-format dump of whole database cluster
- Single thread
- Single step approach
- May require double the space if it is an in-place upgrade.
- Removes table bloat
- A complete downtime for business (write-traffic)
pg_dump/pg_restore with pg_dumpall

- pg_dump and pg_restore using parallel jobs
- Requires pg_dumpall for globals
- May require double the space if it is an in-place upgrade
- Removes table bloat
- Faster when compared to an upgrade with pg_dumpall only
- Involves downtime for business (write traffic).
Slony - Overview

- Logical replication (publisher-subscriber)
- Primary key should be defined on each replicated table
- Trigger-based, additional C daemons (slon) are required
- Any PostgreSQL versions from and to 8.4 ⇔ 11
  - Useful for both upgrades and downgrades
- No support for:
  - DDL (CREATE/DROP/ALTER) - requires application change
  - BLOB (binary data supported, but not OID blobs)
- Application should be switched manually to subscriber
Slony - Additional features

- Monitoring and replication health checks
- Automation using altperl
- Ability to merge replication sets
Slony - Migration

● Migration by preserving existing replication chain:
  ○ Stop write transactions from the application and ensure no pending transactions
  ○ Use LOCK SET to lock the replication set against client updates
  ○ Use MOVE SET move replication set to new database which shifts the origin
  ○ Point the application to the new database

● Migration without preserving:
  ○ UNSUBSCRIBE SET which stops the subscriber from replicating the set
    ■ Table contents will be left and original triggers/rules/constraints will be restored
Logical replication and pglogical

- Uses publisher and subscriber model
- Logical Replication and Logical decoding
  - Replication between PostgreSQL 10.x and 11.x
- **pglogical (extension)**
  - Replication between PostgreSQL 9.4.x and PostgreSQL 11.x
- Requires primary key for tables to be replicated
- Switchover application to Subscriber upon replication
- May be a few minutes (or seconds) of downtime
pg_upgrade

- Time consuming when not using hard links
  - Similar to upgrade using pg_dump/pg_restore
  - Removes bloat from tables
  - Can work between 2 different file systems or servers

- Takes a few seconds when using hard links
  - Works on the same file system in the same server (not applicable for upgrade to a remote server).
  - No changes to the amount of bloat or fragmented space.
  - Does not require an application failover like pglogical or slony
  - May be a few seconds or minutes of downtime
Thank You to Our Sponsors
Rate My Session

Schedule
Timezone: Europe/Berlin +02:00

11:20
ClickHouse: High-Performance Distributed
11:20 - 12:10, Matterhorn 2
Introducing gh-ost: triggerless, painless, trusted online schema migrations
11:20 - 12:10, Matterhorn 2

Details
Introducing gh-ost: triggerless, painless, trusted online schema migrations
11:20 - 12:10
Matterhorn 2

Rate & Review
Tap a star to rate
Feedback (optional)

Anonymously
SUBMIT
Any Questions?