2ndQuadrant
Professional PostgreSQL

Bugs Fixed, Systems Integrated

Gianni Ciolli

PGConf.DE
Hamburg, 26-27 November 2015

Outline

Topics and Plan

Software

Architecture

Automation

Debate

Topics

- High Availability PostgreSQL cluster
- Integration between `repmgr` and PgBouncer

Plan

- Software
- Architecture
- Technical issues
- Diagnose and fix bugs
- Reliability
- Maintenance

Outline

Topics and Plan

Software

Architecture

Automation

Debate

`repmgr` Overview

- Clusterware for PostgreSQL replication
- Open source (GPL)
- Current version: 3.0.2
  - Released on 2 October 2015
- `http://www.repmgr.org/`

Some `repmgr` Features

- Monitoring
- Automatic Failover
- Base Backup with rsync or `pg_basebackup`
- Follow without restart
- Supports Cascading Replication
- Supports Replication Slots
- Event Logging and Commands

PgBouncer Overview

- Connection Pooling
- Open Source (BSD)
- Current version: 1.6.1
  - Released on 3 September 2015
- `http://pgbouncer.github.io/`

Some PgBouncer Features

- Connection Pooling
- Connection Concentration
- Lightweight
- Simple
- Flexible
- `PAUSE`, `RESUME`
  - "bounce" server smoothly!

Surprise guest!

- We also mention **Barman**
  - **Ba**ckup and **R**ecovery **Man**ager
- Why?

Surprise guest!

- We also mention **Barman**
  - **Ba**ckup and **R**ecovery **Man**ager
- Why?
- No Production Without Backup!

Surprise guest!

- We also mention **Barman**
  - **Ba**ckup and **R**ecovery **Man**ager
- Why?
- No Production Without Backup!

No Production Without Backup!

More generally. . .

- The primary is going to change regularly
  - Failover, Switchover, maintenance. . .
- Some maintenance must happen on the primary
  - Could be scheduled with cronjobs
  - Good to have an alias that doesn’t change
- When we say "Barman" think to all such procedures

Barman

- Disaster Recovery software
- Open source (GPL)
- Current version: 1.5.1
  - Released on 16 November 2015
- `http://www.pgbarman.org/`

Some Barman Features

- `.ini` Configuration File
- Configuration Overrides
  - Per user
  - Per server
- Retention Policies
- Monitoring
- Incremental Backup
- Backup from Standby

Some Barman Futures

- Copy Methods
  - `tar`, `pg_basebackup`
- Storage Strategies
  - `tar`, S3
- Backup Compression and Encryption
- Geo-Redundancy
- Import/Export
- …

Outline

Initial Architecture

- One Database Server (PostgreSQL)
- One Backup Server (Barman)

Initial Configuration

- barman.conf

```
[haclu]
ssh_command = ssh haclu-primary
conninfo = service=haclu-primary
description = Test HA cluster
```

Initial Configuration

- `~barman/.pg_service.conf`

```
[haclu-primary]
host=vm1.haclu
user=postgres
```

- `~barman/.ssh/config`

```
Host haclu-primary
     HostName 192.168.56.81
     User postgres
```

- Anything depending on state is placed in userspace
  - Our choice (good practice?)

Introducing `repmgr`

- Create `repmgr.conf`

```
cluster=haclu
node=1
node_name=vm1
conninfo=host=vm1 dbname=repmgr
```

repmgr Usage

```
repmgr master register

repmgr standby clone ...
repmgr standby register
repmgr standby unregister
repmgr standby promote
repmgr standby follow

repmgr witness create

repmgr cluster show
```

One Node

```
postgres@vm1:~$ repmgr master register

postgres@vm1:~$ repmgr cluster show
 Role      | Connection String
 * master  | host=vm1 dbname=repmgr
```

Another Node

```
postgres@vm2:~$ repmgr standby clone -h vm1

postgres@vm2:~$ repmgr standby register

postgres@vm2:~$ repmgr cluster show
 Role      | Connection String
 * master  | host=vm1 dbname=repmgr
   standby | host=vm2 dbname=repmgr
```

Introducing PgBouncer

- PgBouncer defines one or more *databases*
- Each PgBouncer database is a *connection string*
  - Local or Remote
- Clients connect to PgBouncer and are rerouted

PgBouncer Database Conf

- Our choice: separate reads and writes
  - Good practice

- `pgbouncer.ini` on vm1

```
[databases]
postgres_rw = host=vm1 dbname=postgres
postgres_ro = host=vm1 dbname=postgres
```

- `pgbouncer.ini` on vm2

```
[databases]
postgres_rw = host=vm1 dbname=postgres
postgres_ro = host=vm2 dbname=postgres
```

Bug!

- We found it on 28 January 2015
- Fix committed on 28 January 2015
  - Available since version 3.0.2
- Short story (from GitHub commit): «PgBouncer was allowing new server connections after `PAUSE db` »
- In other words: `PAUSE db` returned only after all clients disconnected from `db`
  - Much less useful. . .
- Only affecting `PAUSE db`, not `PAUSE`

What about Barman?

- Standbys are exact clones of the primary
- Many copies of one database server
- Barman only needs to see one
- Barman can backup from standbys. . .

What about Barman?

- Standbys are exact clones of the primary
- Many copies of one database server
- Barman only needs to see one
- Barman can backup from standbys. . .
  - (using `pgespresso`). . .

What about Barman?

- Standbys are exact clones of the primary
- Many copies of one database server
- Barman only needs to see one
- Barman can backup from standbys. . .
  - (using `pgespresso`). . .
  - but we use the primary
    - Keep things simple
    - Simmetry is useful

Outline

`repmgr` Automation

- Daemon `repmgrd`
  - Automatic Failover
  - Monitoring
- Extra automation:
  - When the state changes:
    reconfigure what needs to be reconfigured

Cluster State?

- A *standby* can replace the *master*
  - That’s what "stand by" means. . .
- Two different terms:
  - Switchover: planned
  - Failover: unplanned
- Crucial difference!
- The state of the cluster:
  - List of nodes
  - Which node is the master

New Primary via Switchover

```
postgres@vm1:~$ pg_ctl shutdown

postgres@vm2:~$ repmgr standby promote

postgres@vm3:~$ repmgr standby follow
postgres@vm4:~$ repmgr standby follow
...
postgres@vm100:~$ repmgr standby follow
```

Switchover Wishlist

- `repmgr standby switchover`
- That would be all!

Automatic Failover

```
failover=automatic
master_response_timeout=20
reconnect_attempts=3
reconnect_interval=5
promote_command=repmgr standby promote
follow_command=repmgr standby follow -W
```

- Can define node priority
  - Promote only if positive

Bug!

- Hit by a customer
- Reported on 27 July 2015
- Fix committed on 11 August 2015
  - Available since version 3.0.2
- Short story (from GitHub issue #90):
  - «If the master becomes available again after the first failed attempt, [automatic] failover still proceeds.»

Cluster State Change

- When the state changes:
  - We must update part of the configuration
- All in userspace:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`

Cluster State Change

- When the state changes:
  - We must update part of the configuration
- All in userspace:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`
- Well, almost. . .

Cluster State Change

- When the state changes:
  - We must update part of the configuration
- All in userspace:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`
- Well, almost. . .
- Not in userspace:
  - `/etc/pgbouncer/pgbouncer.ini`

Event Notification Commands

- Add to `repmgr.conf` (only two lines):

```
event_notification_command =
  repmgr-agent.sh repmgr.conf
  barman-server %n %e %s

event_notifications =
  master_register, standby_register,
  standby_promote
```

- Run a custom script in occasion of cluster events
  - A bit like AFTER triggers
- Only those that *change the status*

`repmgr-agent.sh`

- Script that updates the configuration
- Idempotent
- Prototype, to be contributed to `repmgr`
- Reads the cluster state
  - From any node in the cluster
- Rewrites:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`
  - `/etc/pgbouncer/pgbouncer.ini`

Outline

Topics and Plan

Software

Architecture

Automation

Debate

And now. . .

Questions?

And then…

Thank you!

`gianni@2ndquadrant.com`

`@GianniCiolli`

Licence