SYSTEM GUARDS
Consultores de PostgreSQL

The curious case of the point of sales
and why we still need pglogical

Jaime Casanova

SELECT * FROM me;

PostgreSQL contributor

Founder of PUG of Ecuador
- Mailing list: ecpug@postgresql.org
- twitter: @ecpug

Community support in spanish
- pgsql-es-ayuda@postgresql.org
- https://t.me/PostgreSQLes

Board member of
"PostgreSQL Community Association of Canada"

CEO of Systemguards

SELECT * FROM me;

Twitter: @JCasanovaEC / @systemguards

Mail: jcasanov@systemguards.com.ec

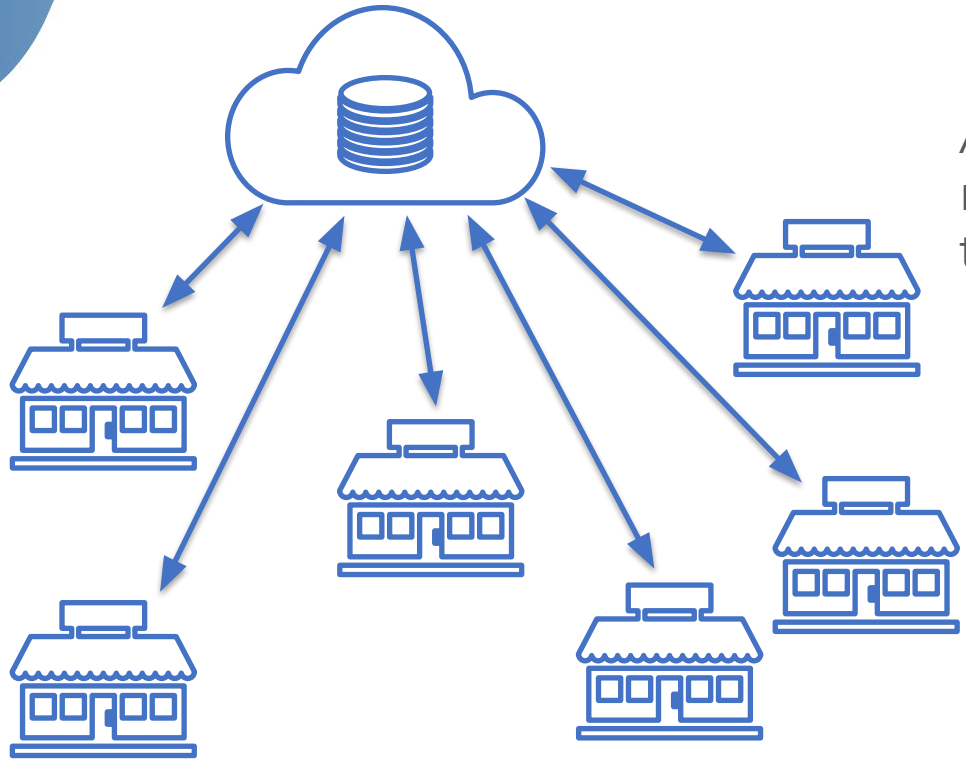
SYSTEM GUARDS
Consultores de PostgreSQL

Goals of this presentation

- A normal case that is not normal
- An initial solution
- A solution that gives some problems
- Next steps
- pglogical vs. native logical replication

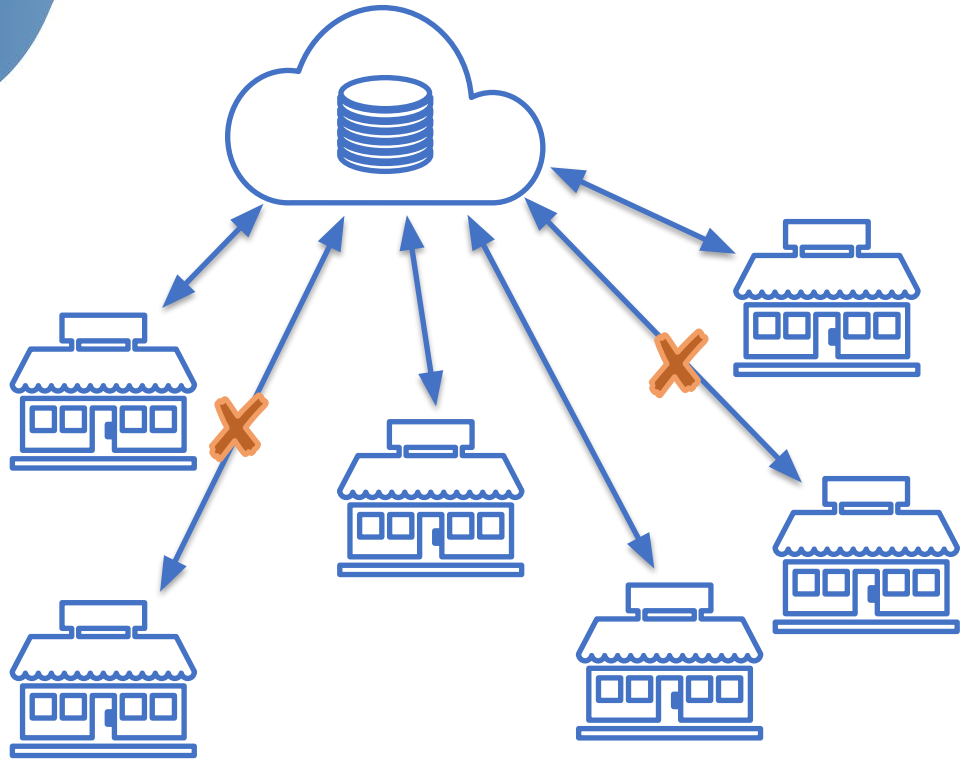
SYSTEM GUARDS
Consultores de PostgreSQL

A normal case that is not normal



A company in Ecuador that sales medicines in several points across the country:

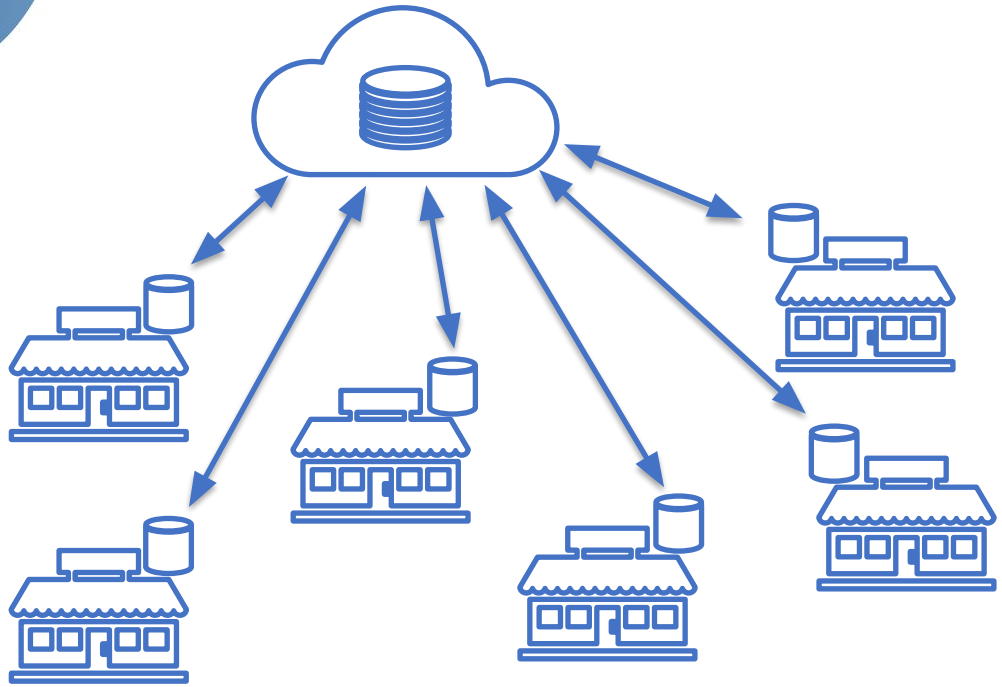
- 40+ POS
  - Inventory
  - Invoicing
- Old system in VB/MS SQL Server
- Migrated to Odoo/PostgreSQL 11



- Company works mostly on little cities and rural areas
- Available networks are:
  - unstable
  - slow

SYSTEM GUARDS
Consultores de PostgreSQL

An initial solution



- Every store has its own copy of the database
- Old system used a home-made script for synchronization
- We use pglogical 2

- Logical decoding feature was introduced on 9.4
- pglogical
  - was developed by 2ndQuadrant (now an EDB company)
  - current open source version: 2.4.2
    - https://github.com/2ndQuadrant/pglogical
    - runs on 9.4 upto 15
- Native logical replication introduced in v10, with limited features and heavily based on **pglogical** extension

Logical replication definitions

- It does a per database replication, not whole cluster
  - It does not use triggers but decode of WAL records
- We need to define what tables and sequences will be replicated and which operations (INSERT/UPDATE/DELETE)
  - replication_set on pglogical
  - PUBLICATION on native logical replication
- a subscriber (replica) asks the origin for the information published by a PUBLICATION (replication_set)

Common configuration

```
 1 ALTER SYSTEM SET shared_preload_libraries TO 'pglogical';
 2
 3 /* RESTART */
 4
 5 CREATE EXTENSION pglogical;
 6
 7 SELECT pglogical.create_node(
 8         node_name := 'node_name',
 9         dsn := 'host=this.node.ip port=5432 dbname=xxxx user=xxxxxx'
10 );
```

Configuring replication_sets

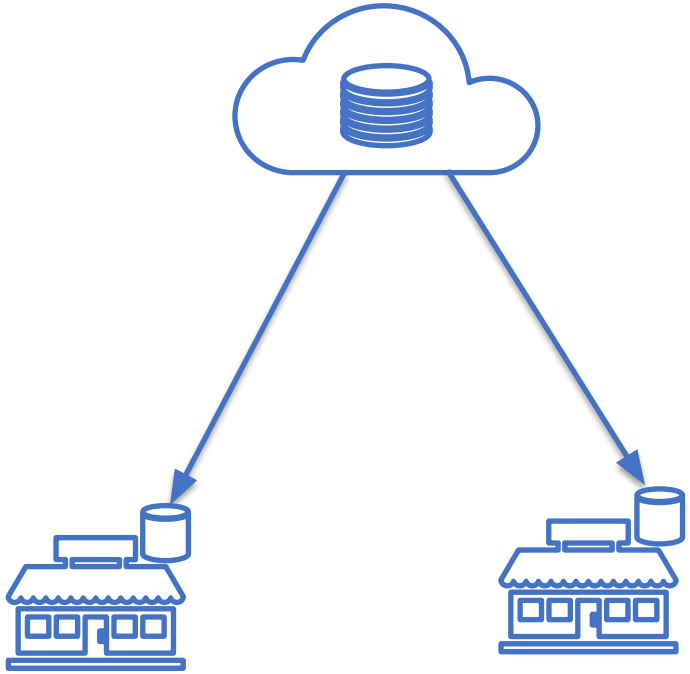
```
SELECT pglogical.create_replication_set(
        set_name := 'replication_set_name',
        replicate_insert:= true,
        replicate_update:= true,
        replicate_delete:= true,
        replicate_truncate:= true
);

SELECT pglogical.replication_set_add_table(
        set_name := 'replication_set_name',
        relation := 'public.table_name'::regclass,
        synchronize_data := false
);
```

Classifying tables

- 139 tables whose data originated on central and must be replicated to the stores
  - replication_set := set_acme_cat_down
- 35 tables whose data originated on the stores and must be replicated to the central
  - replication_set := set_acme_cat_up
- 5 tables that could be written anywhere and the changes must be propagated to all the stores and the central
  - replication_set := set_acme_shared_tables

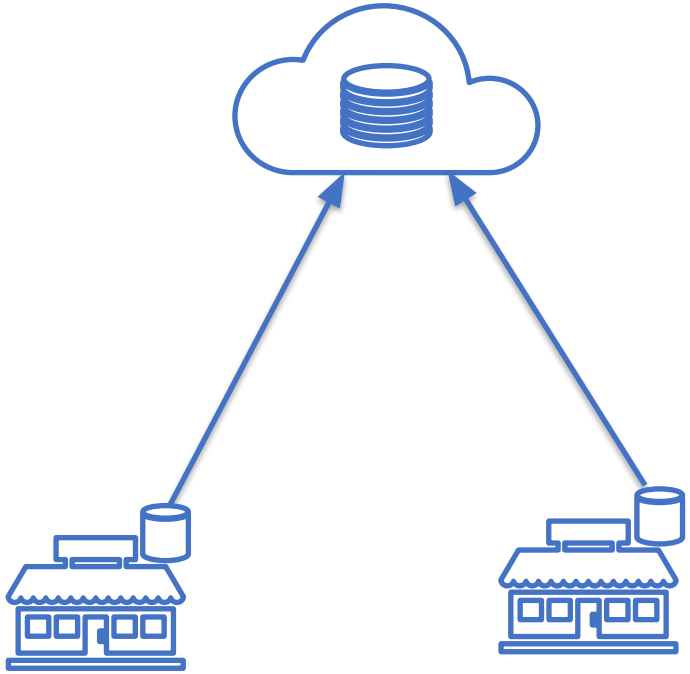
replication_set: set_acme_cat_down



On every store, to subscribe data generated on central

```
SELECT pglogical.create_subscription(
        subscription_name := 'sub_acme_cat_down_central_store1',
        provider_dsn := 'host=provider.ip port=5432 dbname=xxxx user=xxxxxx',
        replication_sets := '{set_acme_cat_down}',
        synchronize_structure := false,
        synchronize_data := true,
        forward_origins := '{}'
);
```

replication_set: set_acme_cat_up



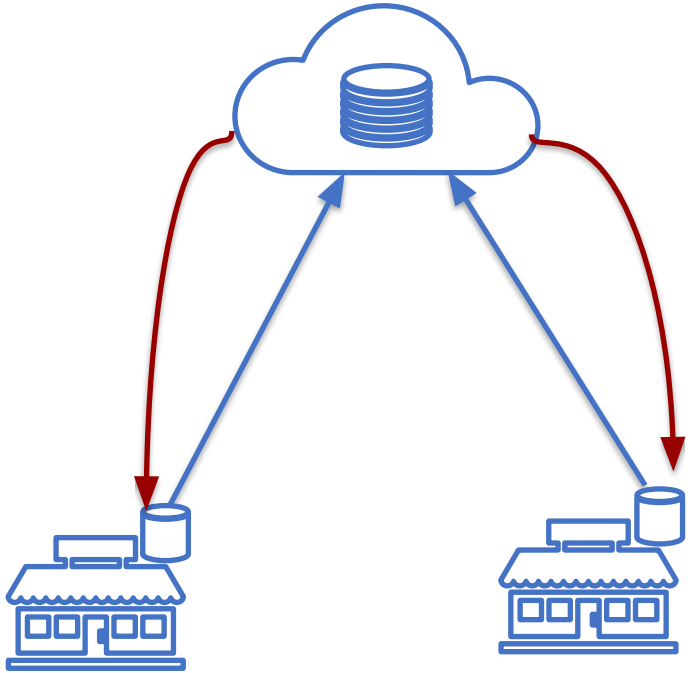
On central, to subscribe data generated on every store

```
1 SELECT pglogical.create_subscription(
2          subscription_name := 'sub_acme_cat_up_store1_central',
3          provider_dsn := 'host=provider.ip port=5432 dbname=xxxx user=xxxxxx',
4          replication_sets := '{set_acme_cat_up}',
5          synchronize_structure := false,
6          synchronize_data := true,
7          forward_origins := '{}'
8 );
```

Does replicated data generates conflicts?

- you can avoid them by:
  - having enough information on the table
  - using uuid as primary key
  - assign a range of values per store
  - Do as instagram to generate a conflict-free ID that is deterministic: https://instagram-engineering.com/sharding-ids-at-instagram-1cf5a71e5a5c

replication_set: set_acme_shared_tables



```
SELECT pglogical.create_subscription(
        subscription_name := 'sub_acme_shared_store1_central',
        provider_dsn := 'host=provider.ip port=5432 dbname=xxxx user=xxxxxx',
        replication_sets := '{set_acme_shared_tables}',
        synchronize_structure := false,
        synchronize_data := true,
        forward_origins := '{}'
);


SELECT pglogical.create_subscription(
        subscription_name := 'sub_acme_shared_central_store1',
        provider_dsn := 'host=provider.ip port=5432 dbname=xxxx user=xxxxxx',
        replication_sets := '{set_acme_shared_tables}',
        synchronize_structure := false,
        synchronize_data := true,
        forward_origins := '{all}'
);
```

SYSTEM GUARDS
Consultores de PostgreSQL

A solution that gives some problems

- The initial copy of data for set_acme_cat_down replication set could take upto 10 hours
  - We use pg_dump to move data "faster"
- After creation we need to recheck for missing rows

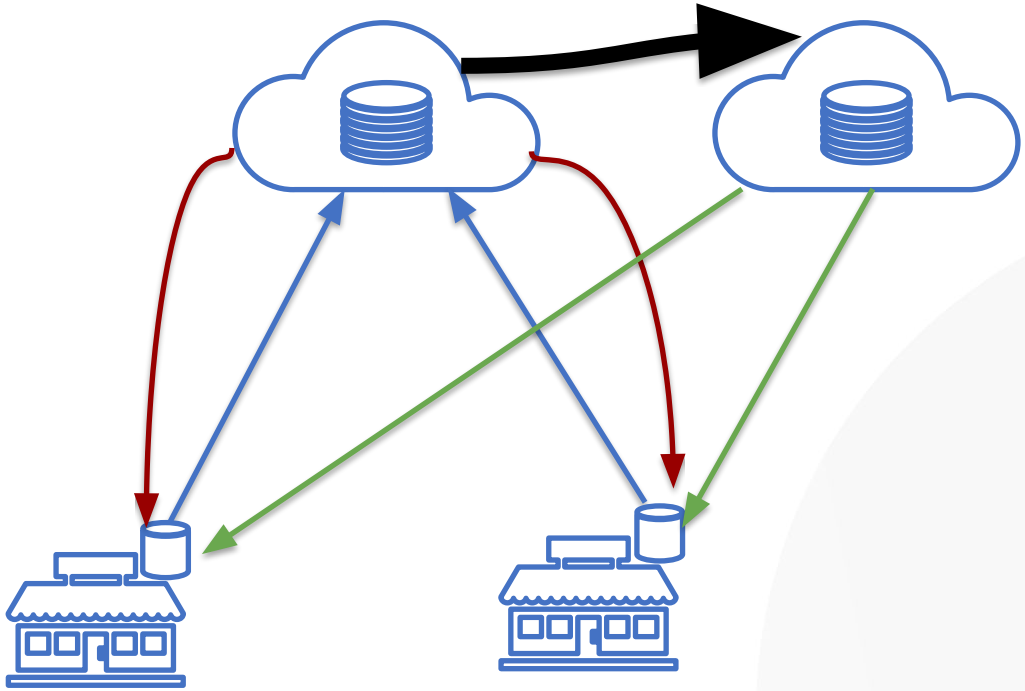
- A subscription generates one wal sender on the provider
  - There are two subscription getting data from central
- When the store # 70 was created... and we reached 140 decoding processes the server's temperature went up to a critical level

```
    89.06%     0.00%  postgres  postgres           [.] exec_replication_command
            |
            ---exec_replication_command
               |
               |--89.01%-- WalSndLoop
               |          |
               |          |--88.18%-- XLogSendLogical
               |          |          |
               |          |          |--75.11%-- GetFlushRecPtr
               |          |          |          |
               |          |          |          |--66.81%-- s_lock
```

- A patch for using LWLocks instead of spinlocks was provided to customer
  - Álvaro Herrera contributed an improved version of the patch using atomics
  - https://www.postgresql.org/message-id/flat/20200831182156.GA3983@alvherre.pgsql
- Starting 12.2 an independent commit of a patch from Pierre Ducroquet reduced the contention
  - https://www.postgresql.org/message-id/flat/2931018.Vxl9zapr77%40pierred-pdoc

SYSTEM GUARDS
Consultores de PostgreSQL

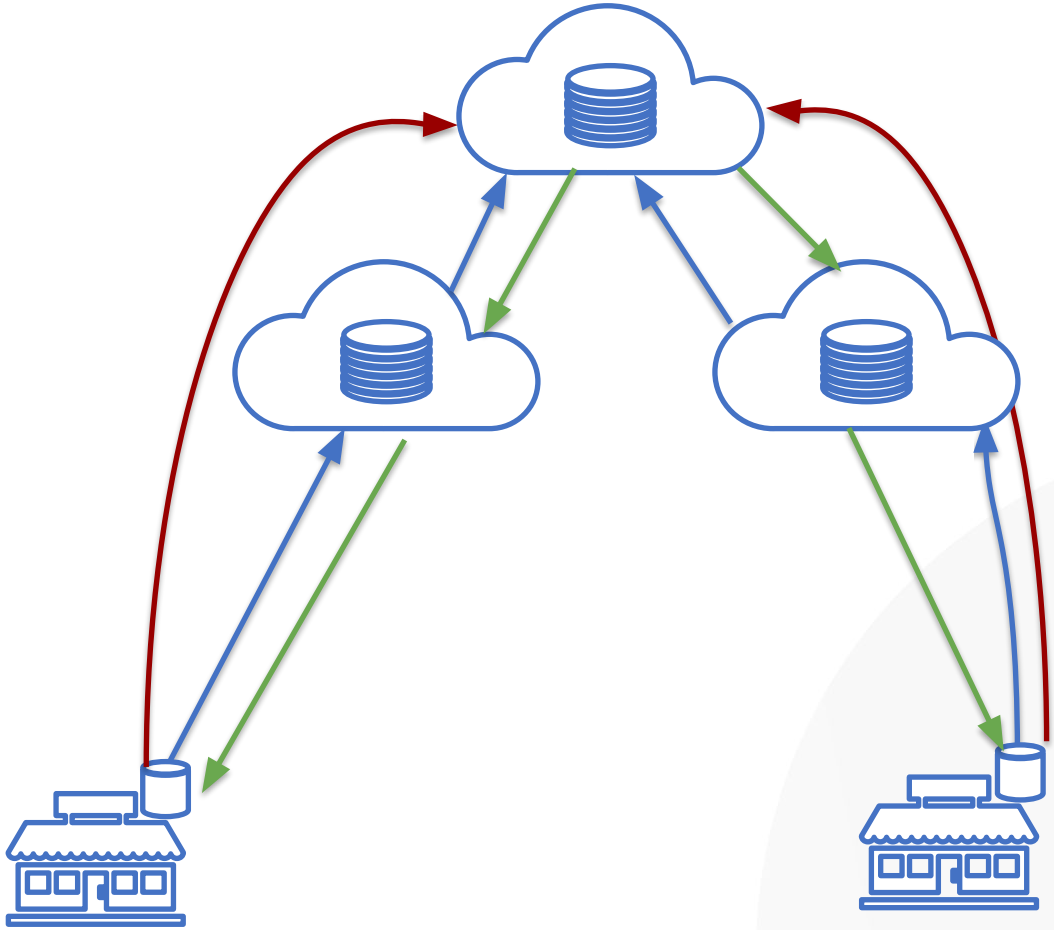
Next steps



SYSTEM GUARDS
Consultores de PostgreSQL

When the store # 142 was created...

```
1 ERROR:  out of memory
2 DETAIL:  Failed on request of size 724 in memory context "Tuples".
3 STATEMENT:  START_REPLICATION SLOT
```



SYSTEM GUARDS
Consultores de PostgreSQL

pglogical vs. native logical replication

Native logical replication

- continuously improved
- 14: large in-progress transactions could be streamed before commit
- 15: ALTER SUBSCRIPTION ... SKIP
- It always work on cascade mode
- there isn't any conflict resolution

Thank you

Any Questions?