Upgrade of a many TB postgres database with little down time

and not die trying

www.systemguards.com.ec

26.7.2022

SELECT * FROM me;

PostgreSQL contributor

Founder of Ecuador's PUG
- 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

Goals of this presentation

- Why to upgrade?
- Overview of standard ways of upgrading
- Study case
  - Constraints imposed by the customer
  - A not so standard solution

Why to upgrade?

Why to upgrade?

- ~~Minor~~ Security releases every 3 months
  - fix bugs
- Current supported versions: 10 to 14
- Versión 15 expected for sometime after september
  - 10 will lose support on november 2022

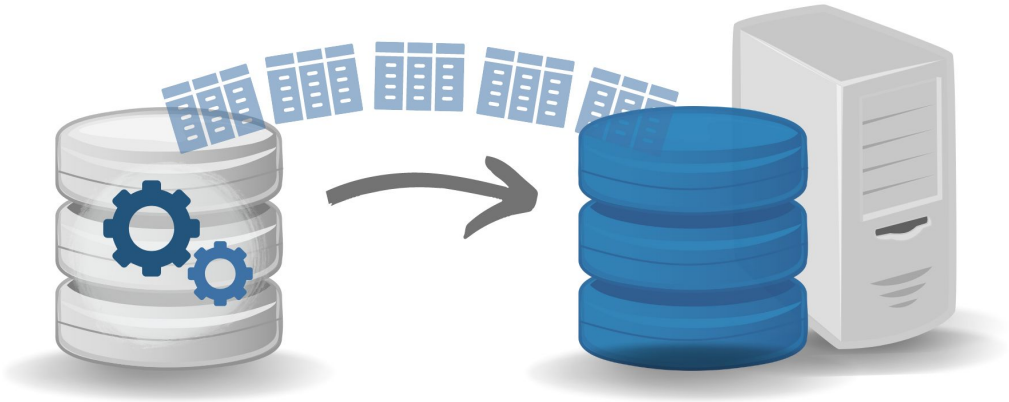
Overview of standard ways of upgrading

Pros
It's easy
It's the more tested
Cons
Apps should stop writing before the dump starts
To make it a bit more faster use directory format
All standbys become invalid

Overview of standard ways of upgrading

Pros
It's faster
Cons
Service must be shutdown to start the process
It doubled the used space (unless you use --link or --clone)
To make it a bit more faster use link option (no rollback)
All standbys become invalid

Overview of standard ways of upgrading



Logical replication

Pros
It could be done without disturbing current activity
It works between different versions of PostgreSQL
Cons
All replicated tables must have a PK
Initial copy could: • Take a long time • Cause bloat • Consume space on primary (WAL retention)

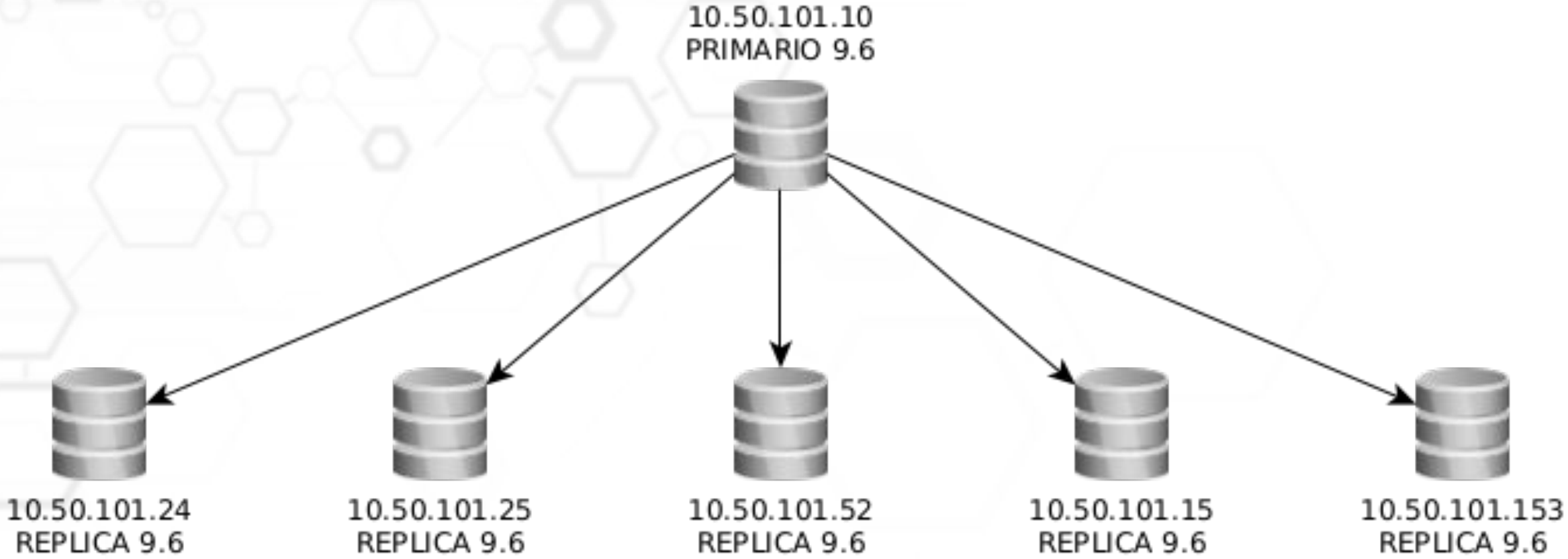
Study case

A financial system company managing lots of transactions from North and South America and Europe

- 1 primary (for writing mostly), 5 read only replicas
  - PostgreSQL version: 9.6
  - Total cluster size: 18Tb

- Mission critical
  - Every replica is used for one or various services
  - No service could be down for more than 5 minutes
  - No server has more than 1TB of free space
  - It generates an average of 10 WALs/sec (160MB/s)

Study case


10.50.101.10
PRIMARIO 9.6
10.50.101.24
REPLICA 9.6
10.50.101.25
REPLICA 9.6
10.50.101.52
REPLICA 9.6
10.50.101.15
REPLICA 9.6
10.50.101.153
REPLICA 9.6

pglogical

- Developed by 2ndQuadrant (now an EDB company)
  - Provides an API to implement logical replication (9.4+)
- shared_preload_libraries = 'pglogical'
- a *replication_set* is a list of tables, sequences and operations to replicate
  - what native logical replication call *publication*
- a *subscription* asks primary for the information in a replication_set (publication)

pglogical: on the primary

```
CREATE EXTENSION pglogical;

SELECT pglogical.create_node('primary', 'host=x.x.x.x dbname=db1');

SELECT pglogical.replication_set_add_all_tables('default',
          (select array_agg(nspname) from pg_namespace
            where nspname not like 'pg_%'
              and nspname <> 'information_schema')
          );
```

pglogical: on the new replica

- After restoring the database schema

```
CREATE EXTENSION pglogical;

SELECT pglogical.create_node('replica', 'host=y.y.y.y dbname=db1');

SELECT pglogical.create_subscription('subscription_name',
                'host=x.x.x.x dbname=db1', '{default}', false, true);
```

pglogical: on the new replica

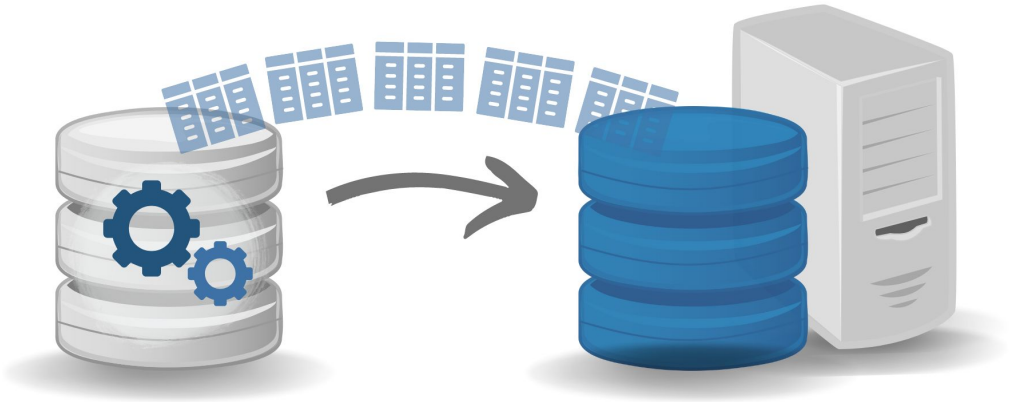
- After restoring the database schema

```
CREATE EXTENSION pglogical;

SELECT pglogical.create_node('replica', 'host=y.y.y.y dbname=db1');

SELECT pglogical.create_subscription('subscription_name',
                'host=x.x.x.x dbname=db1', '{default}', false, true);
```

Does logical replication work in this case?



Logical replication

Cons
All replicated tables must have a PK ● Index will be maintained while data gets loaded
Initial copy could: ● Take a long time ○ 1 table had > 5Tb ○ 10 hours over a 1Gbit/s network ● Cause bloat ● Consume space on primary (WAL retention) ○ ~5.49 Tb

A not so standard solution

- Use an existing standby for the initial copy of the data
- Transform it into a logical replica
- pg_upgrade it (~40min, ~375Gb)
- Continue with the logical replication in the already upgraded cluster

Transform a standby into a logical replica

```
/usr/pgsql-9.6/bin/pglogical_create_subscriber
    --pgdata=${PGDATA96}
    --subscriber-name="subscription_name"
    --subscriber-dsn="host=y.y.y.y"
    --provider-dsn="host=x.x.x.x"
    --databases="db1" --replication-sets=default
```

Before upgrading

- Disable the subscription

```
SELECT pglogical.alter_subscription_disable(sub_name, true)
  FROM pglogical.subscription;
```

Before upgrading

- Save information about replication origins

```
COPY (select external_id, remote_lsn from pg_replication_origin_status)
TO '/tmp/origin_status.txt'
```

Transform a standby into a logical replica

```
/usr/pgsql-13/bin/pg_upgrade
    -p 54321                    -P 54322
    -b /usr/pgsql-9.6/bin/      -B /usr/pgsql-13/bin/
    -d /var/lib/pgsql/9.6/data -D /var/lib/pgsql/13/data/
    --link
    --check
```

Transform a standby into a logical replica

```
/usr/pgsql-13/bin/pg_upgrade
    -p 54321                    -P 54322
    -b /usr/pgsql-9.6/bin/      -B /usr/pgsql-13/bin/
    -d /var/lib/pgsql/9.6/data -D /var/lib/pgsql/13/data/
    --link
```

After upgrading

- Restore information about replication origins

```
SELECT pg_replication_origin_create('external_id');
SELECT pg_replication_origin_advance('external_id', '0/000123'::pg_lsn);
```

After upgrading

- Enable the subscription

```
SELECT pglogical.alter_subscription_enable(sub_name, true)
  FROM pglogical.subscription;
```

Final thougths

- Before start creating the physical replicas of the logical replica make any compatible change you need
  - integer -> bigint
  - create new indexes

- if you haven't added sequences to the replication set, you need to setval() them before starts writing

- This procedure has been tested with pglogical, there could be possible to do the same with native logical replication but the exact steps may differ
  - pglogical could disappear *soon*

Questions?