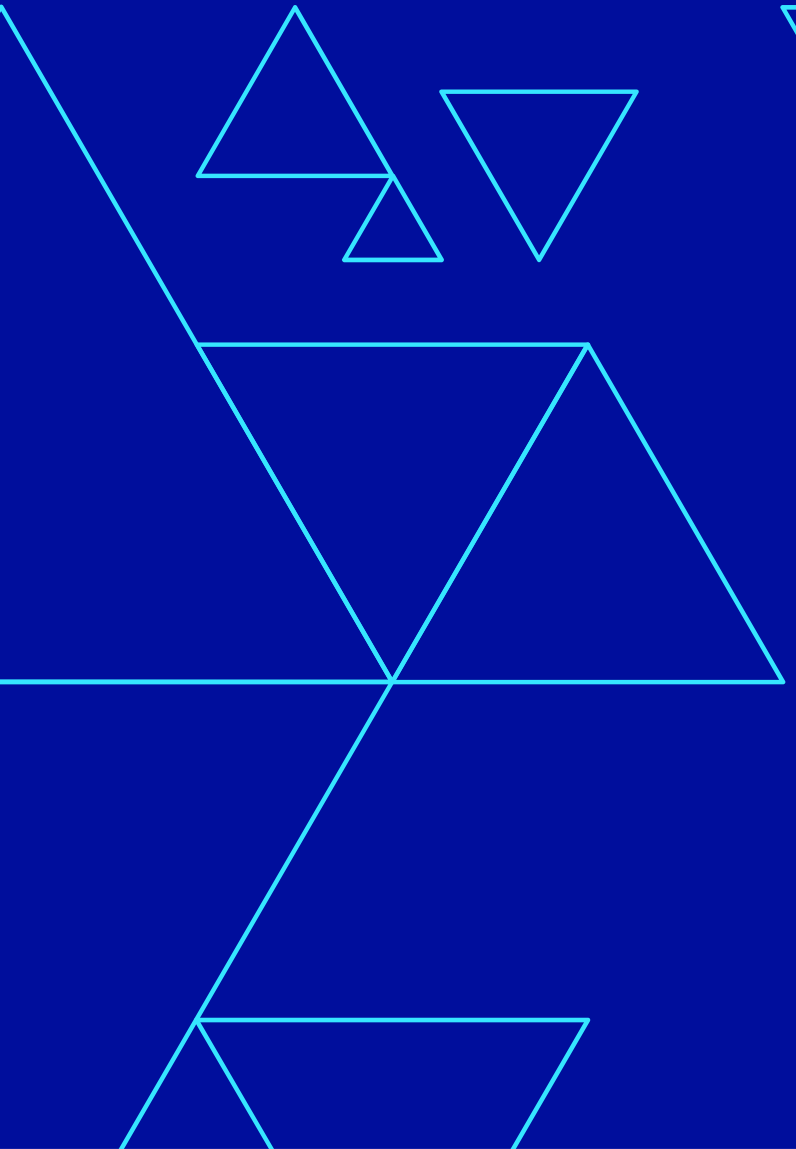


The Elephantine Upgrade

Julien Riou

PGConf NYC

September 23, 2022

Speaker

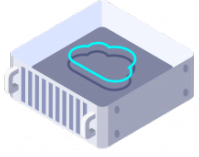
- Julien Riou
- DBA since 2012
- Tech lead at OVHcloud since 2015
- https://julien.riou.xyz

Context

Who are we?

Global cloud provider

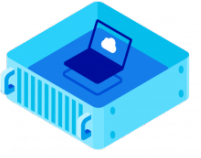
OVHcloud®



Bare Metal Cloud

Hosted Private Cloud

Public Cloud

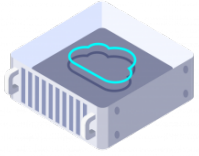


Web Cloud

Who are we?

Global cloud provider

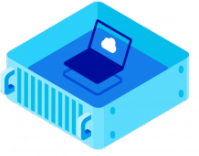
OVHcloud®



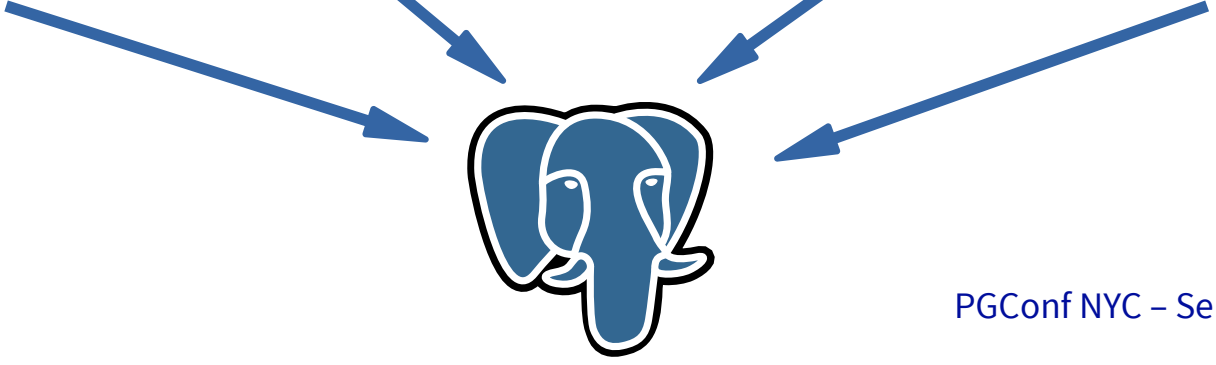
Bare Metal Cloud

Hosted Private Cloud

Public Cloud



Web Cloud

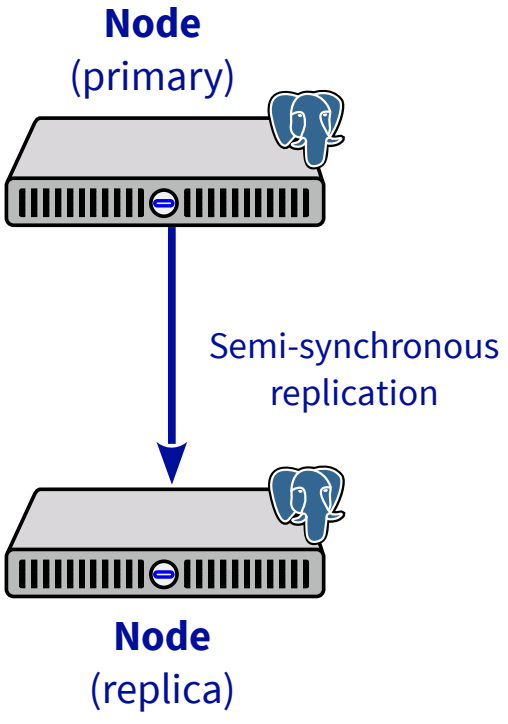


Infrastructure

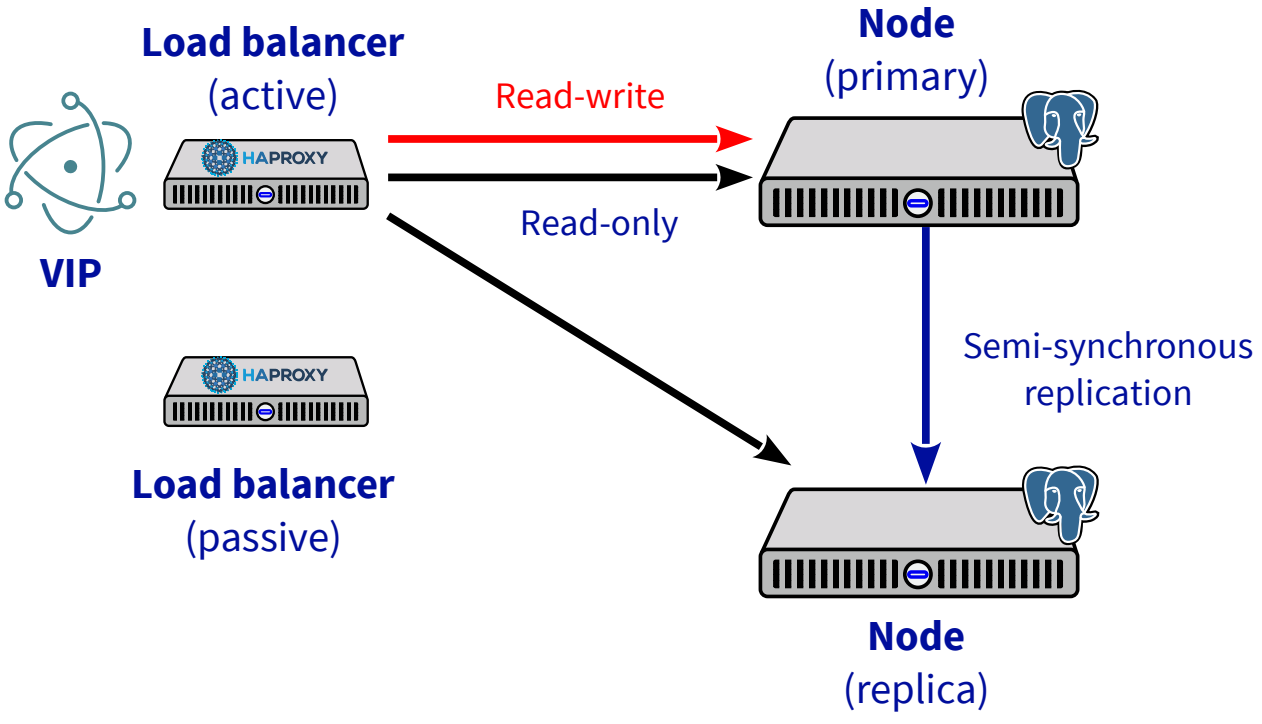
- **5 infrastructures** (production, development and more)
- **230+** PostgreSQL **databases**
- **50+** PostgreSQL **clusters**
- Some clusters are deployed in **highly-secured environments**

Cluster example

Patroni



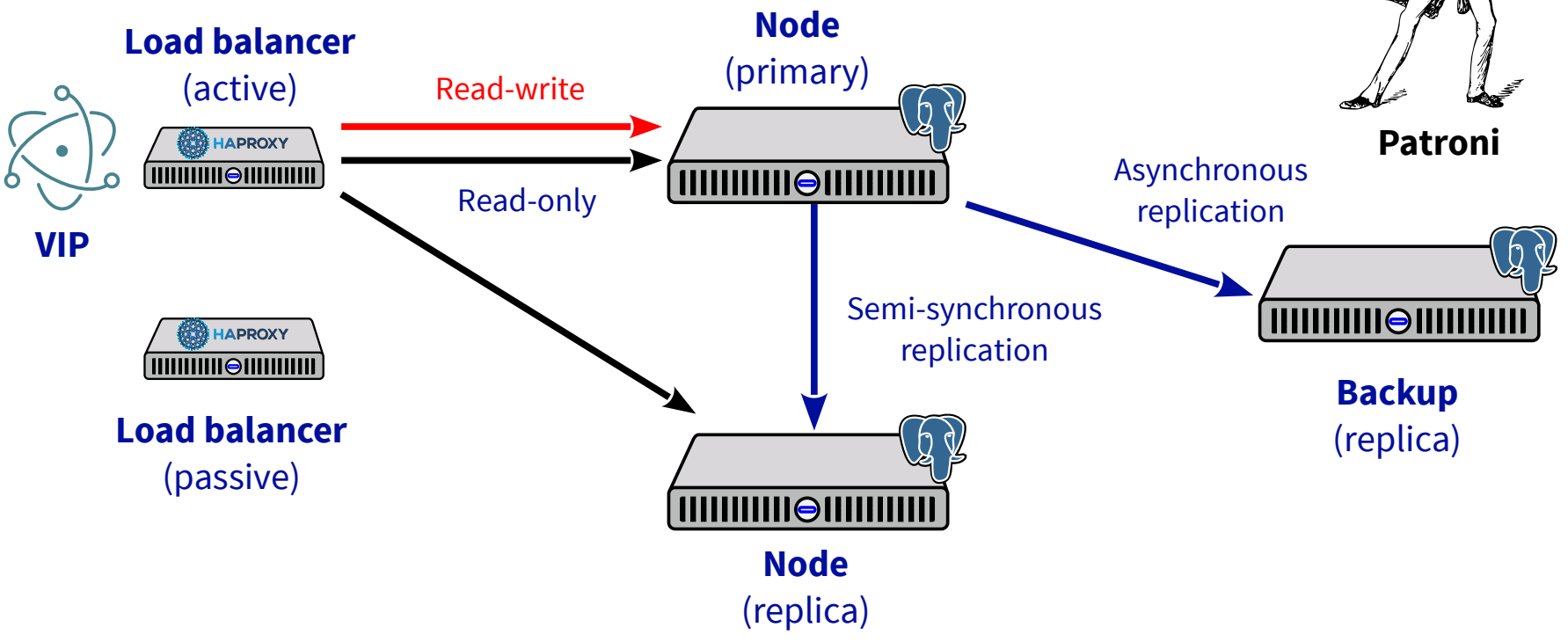
Cluster example



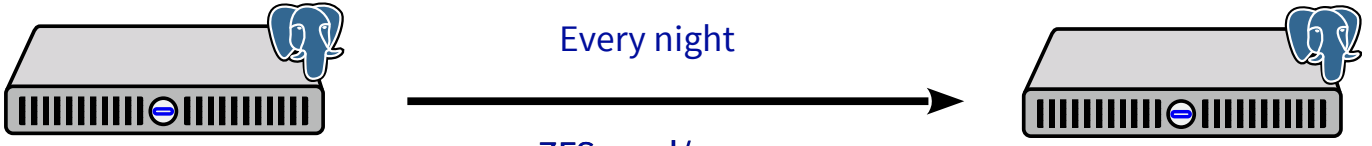
Patroni

Cluster example

Patroni



Cluster example



Backup
(PROD)

Anonymized instance
(DEV)

Highly-secured environments

- Fine-grained **firewall** and **pg_hba.conf** rules
- Connection with **TLS encryption enforced** with **recent ciphers**
- Data **encryption at rest** (disks)
- **Connection** and **disconnection logging**
- Logs are sent to an external system for **SIEM analysis**
- SSH connection only allowed via **bastions**
- **MFA** (Yubikey PIV + password) enabled on internal tools
- **CVE** monitoring
- **Audits** every year

Motivations

9.6


9.6

Why upgrade?

- PostgreSQL 9.6 **end of support** since **November 2021**

- 5 new major releases with:
  - More **features**
  - More **performance**
  - More **security**

What version?

Next major release available at the beginning of the project:

14

Opportunities

- **Operating system** upgrade (Debian 9 → 11)
- Replace SSD by **NVMe disks**
- Apply **PCI DSS security rules** to 100% of the database infrastructure
- Replace ZooKeeper by **Consul** for Patroni DCS

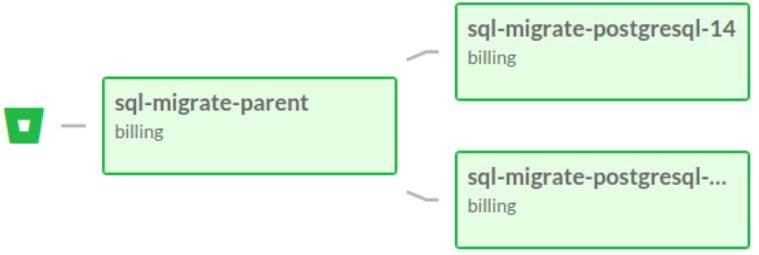
Constraints

- Identify and **avoid incompatibilities** with the new version early on
- **Near-to-zero downtime**
- **Automate** the migration process (because 50+ clusters!)
- Before the next **security audit** (FY22Q3)

Before the upgrade

Is my database schema compatible?

- Test schema migrations with **CDS**
- Test both major versions at the same time
- https://github.com/ovh/cds



Upgrade of a single cluster

Available tools

- pg_dump/pg_restore
- pg_upgrade
- Logical replication

pg_dump/pg_restore

- **pg_dump**
  - Logical export of a database at a given moment
- **pg_restore**
  - Import of a dump into a database

pg_dump/pg_restore

- Portable
- Remove bloat
- Small downtime for small databases
- Compatible with tables without primary or unique key

- Extended downtime for large databases

pg_upgrade

- Update the format of system tables
- Leave data untouched
- Restart the instance on new binaries

pg_upgrade

- Small read-write downtime
  - No matter how large is the database

- Hard to rollback
- Bloat not removed
- Not portable

Logical replication

- **WAL** (*Write-Ahead Log*)
  - Write operations of an instance

- **Physical replication** (*Streaming replication*)
  - Block copy of WAL content from one **instance** to another

- **Logical replication**
  - **Decode** WAL content to extract changes at the **database** level

Logical replication

- Available solutions to replicate data from PostgreSQL to PostgreSQL:
  - **pglogical** (9.4+)
  - **Built-in logical replication** (10+)

Logical replication

- Available solutions to replicate data from PostgreSQL to PostgreSQL:
  - **pglogical** (9.4+)
  - **Built-in logical replication** (10+)

Logical replication

- Small write downtime
- Portable
- No more bloat

- UTF-8 only (pglogical)
- Load on primary at initialization
- Primary or unique key required
- DDLs ignored

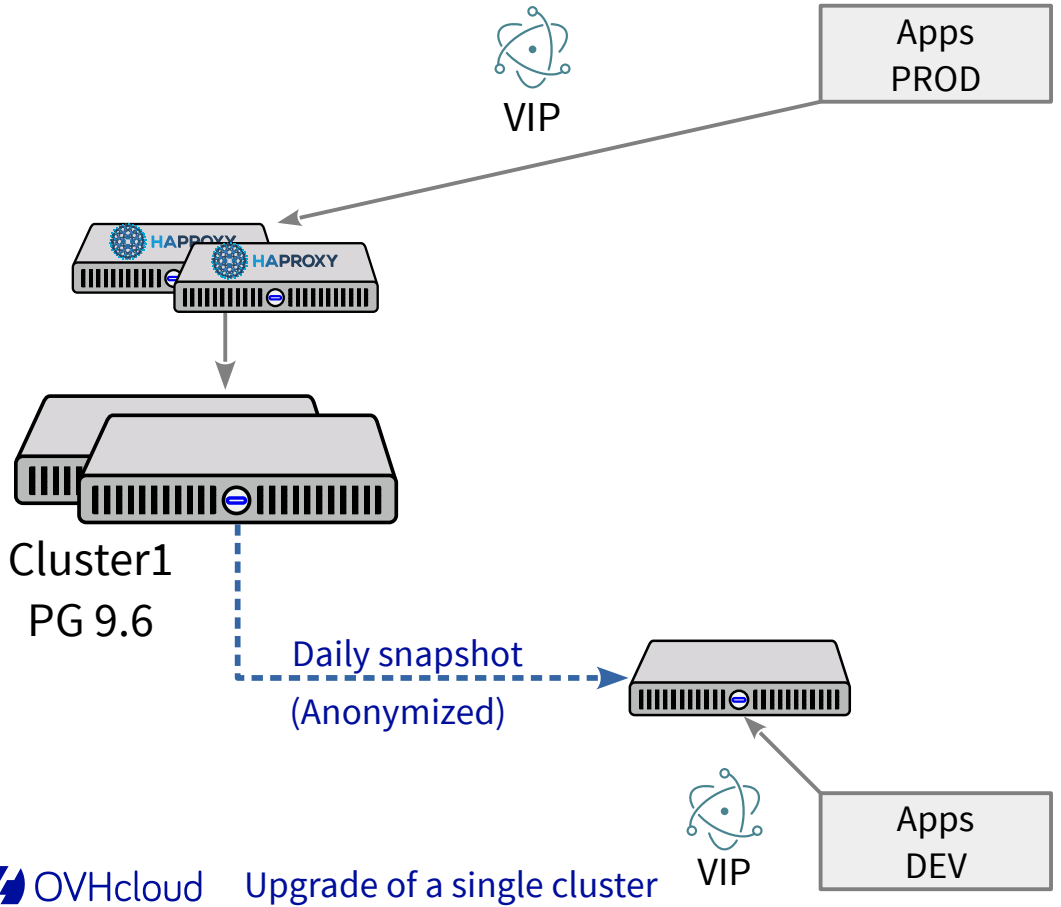
Logical replication

- Small write downtime
- Portable
- No more bloat

- UTF-8 only (pglogical)
- Load on primary at initialization
- Primary or unique key required
- DDLs ignored

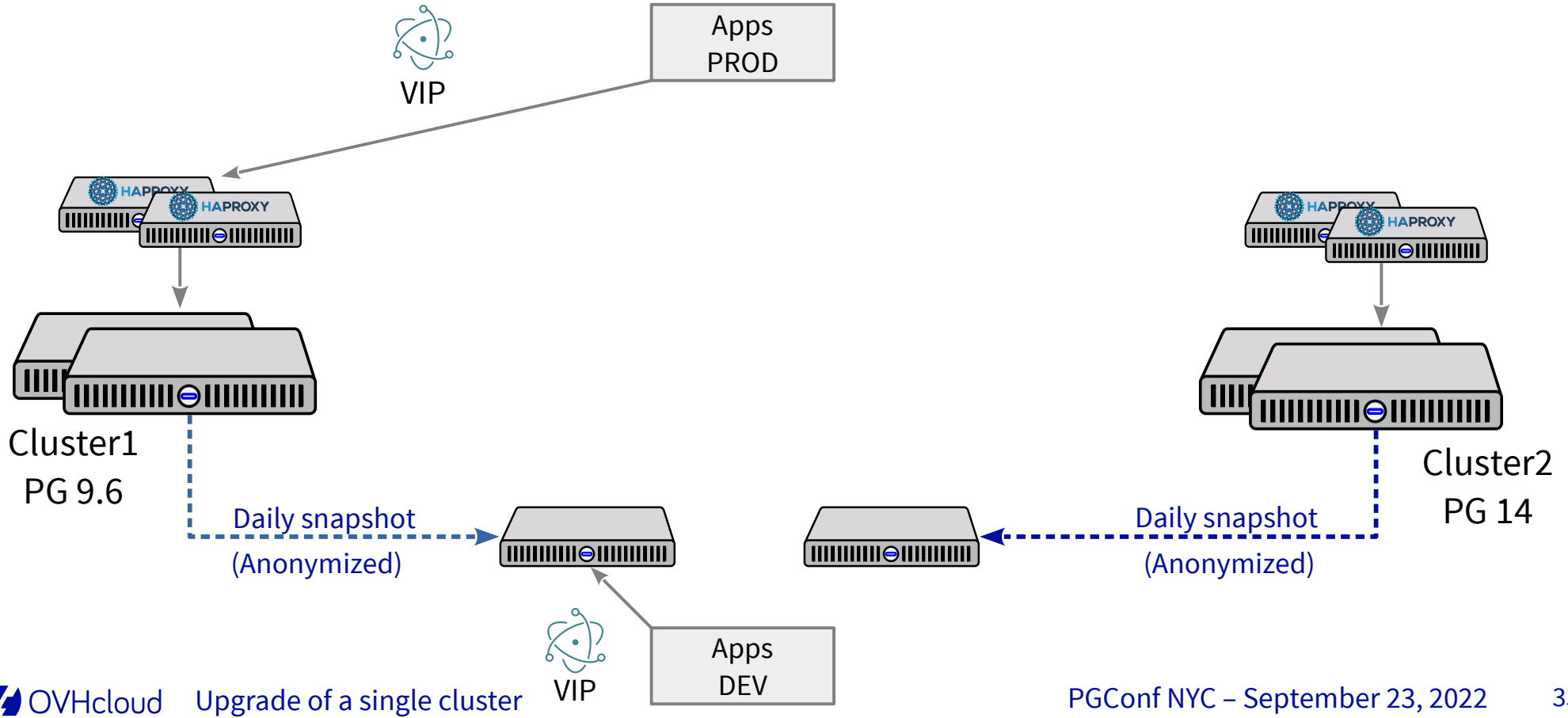
Logical replication

1. Initial state

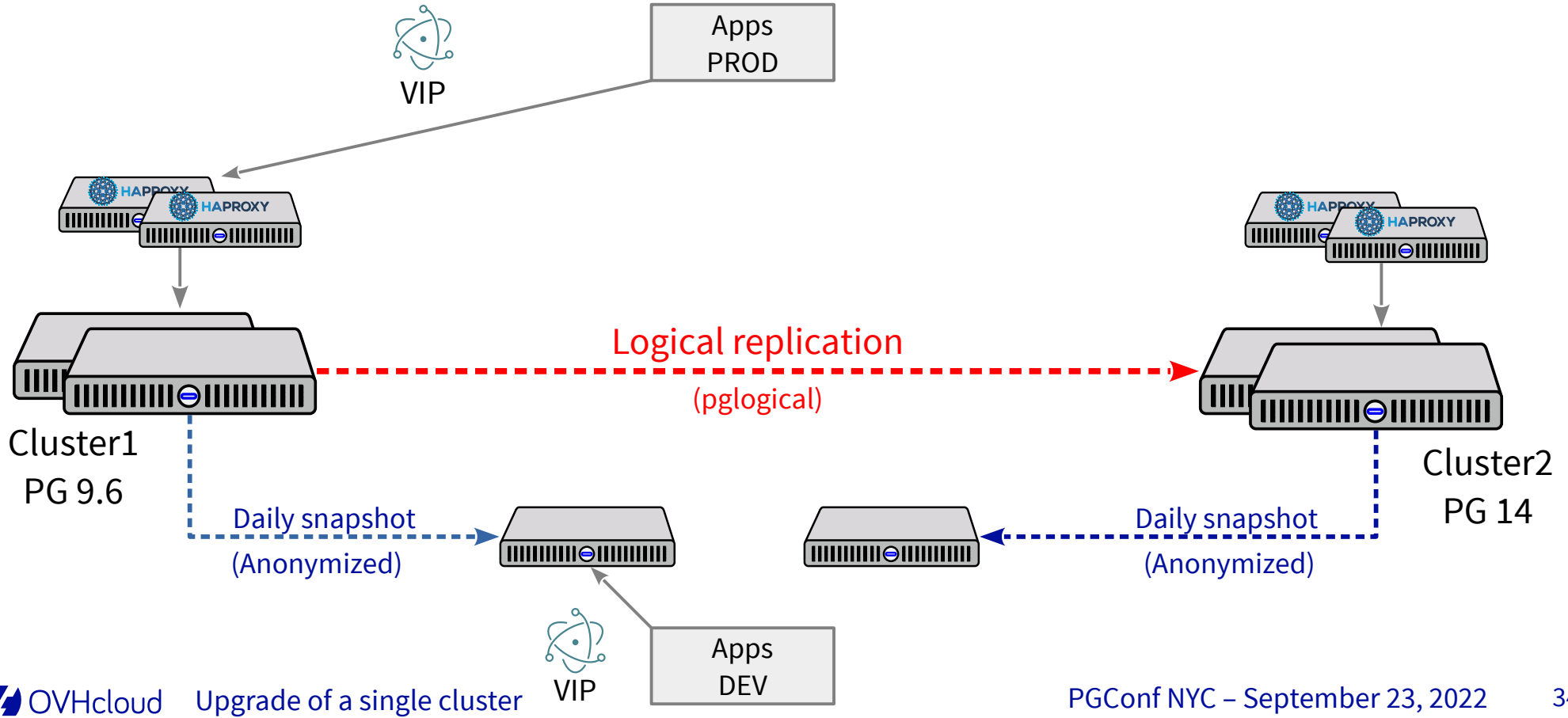


Logical replication

2. Add a cluster with new version

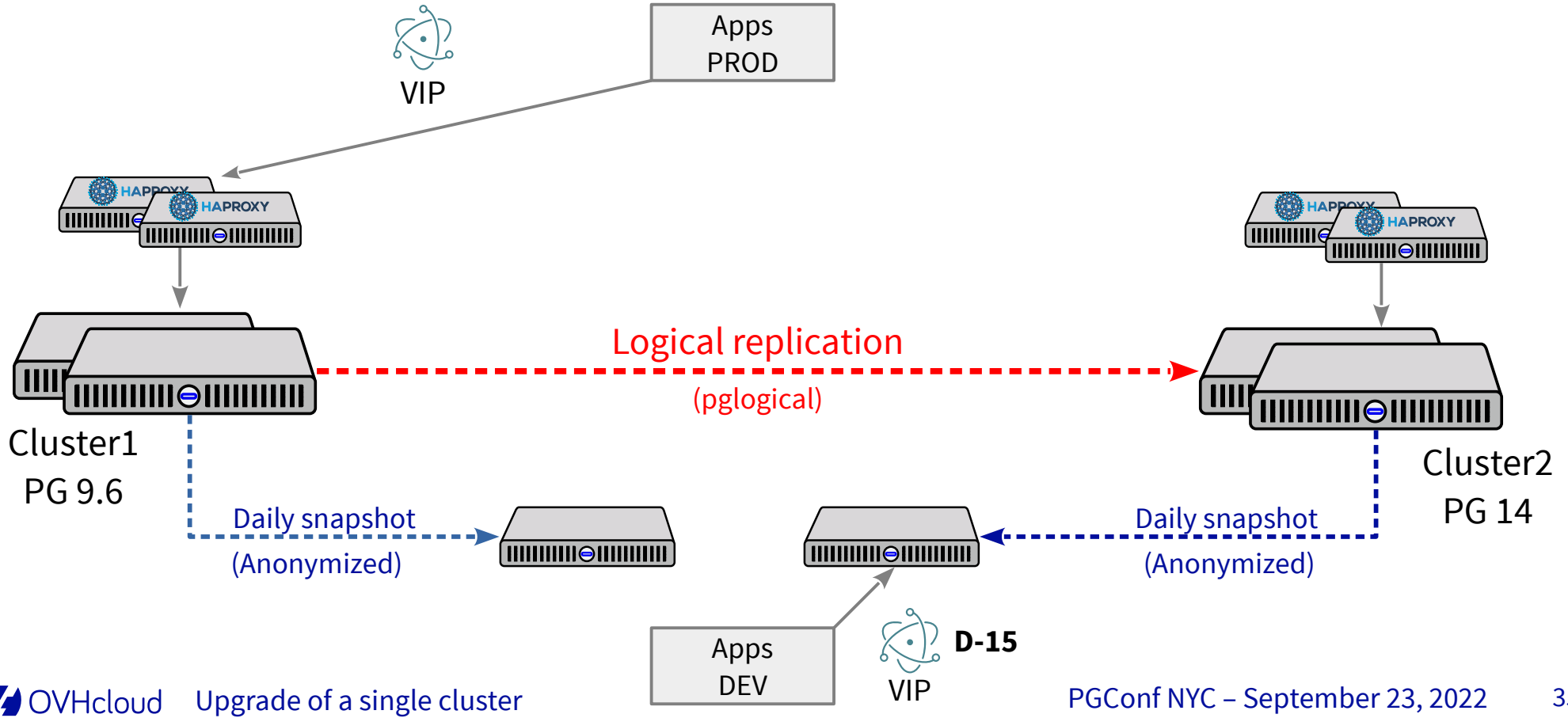


Logical replication



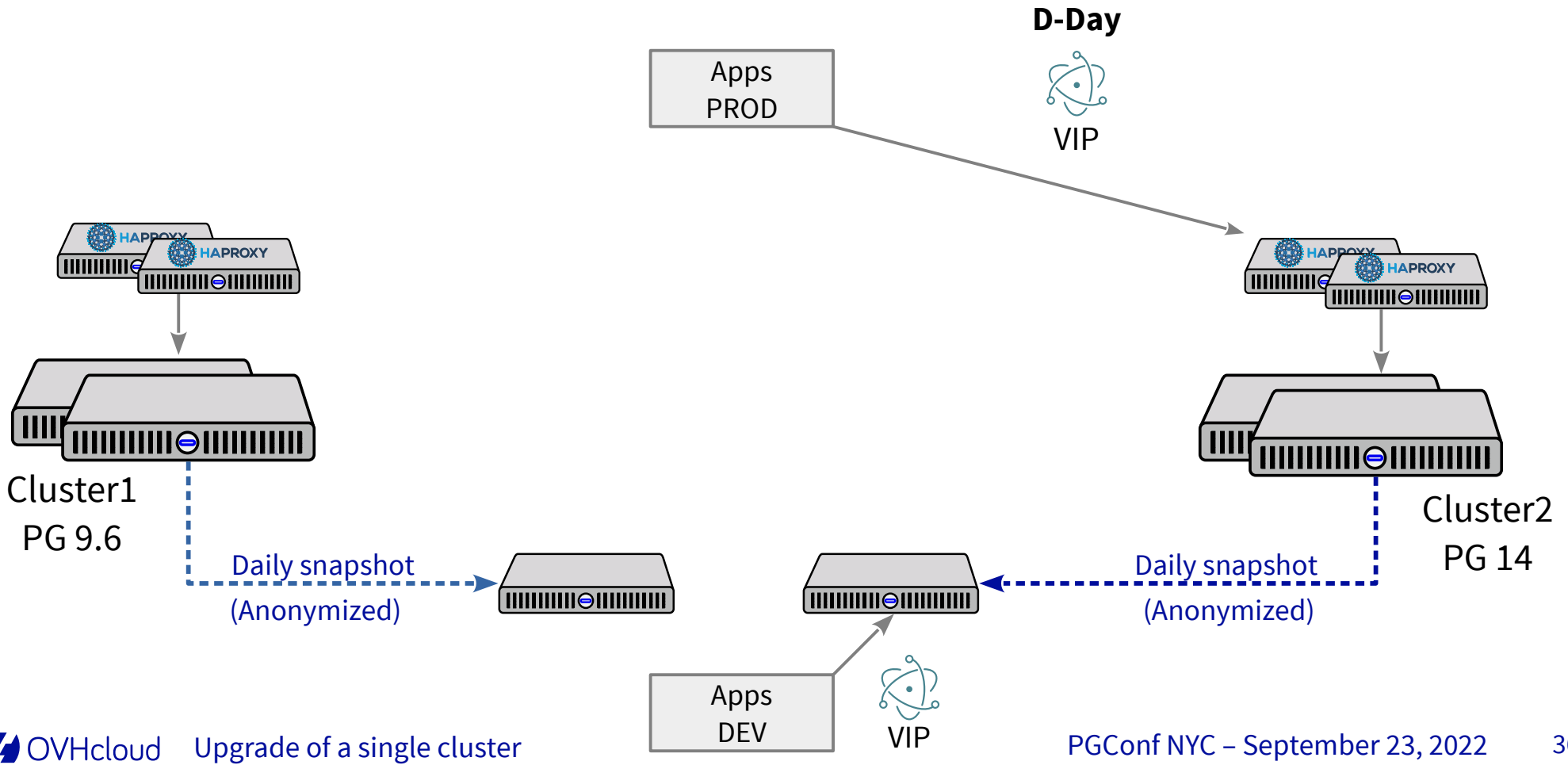
Logical replication

4. Switch DEV applications



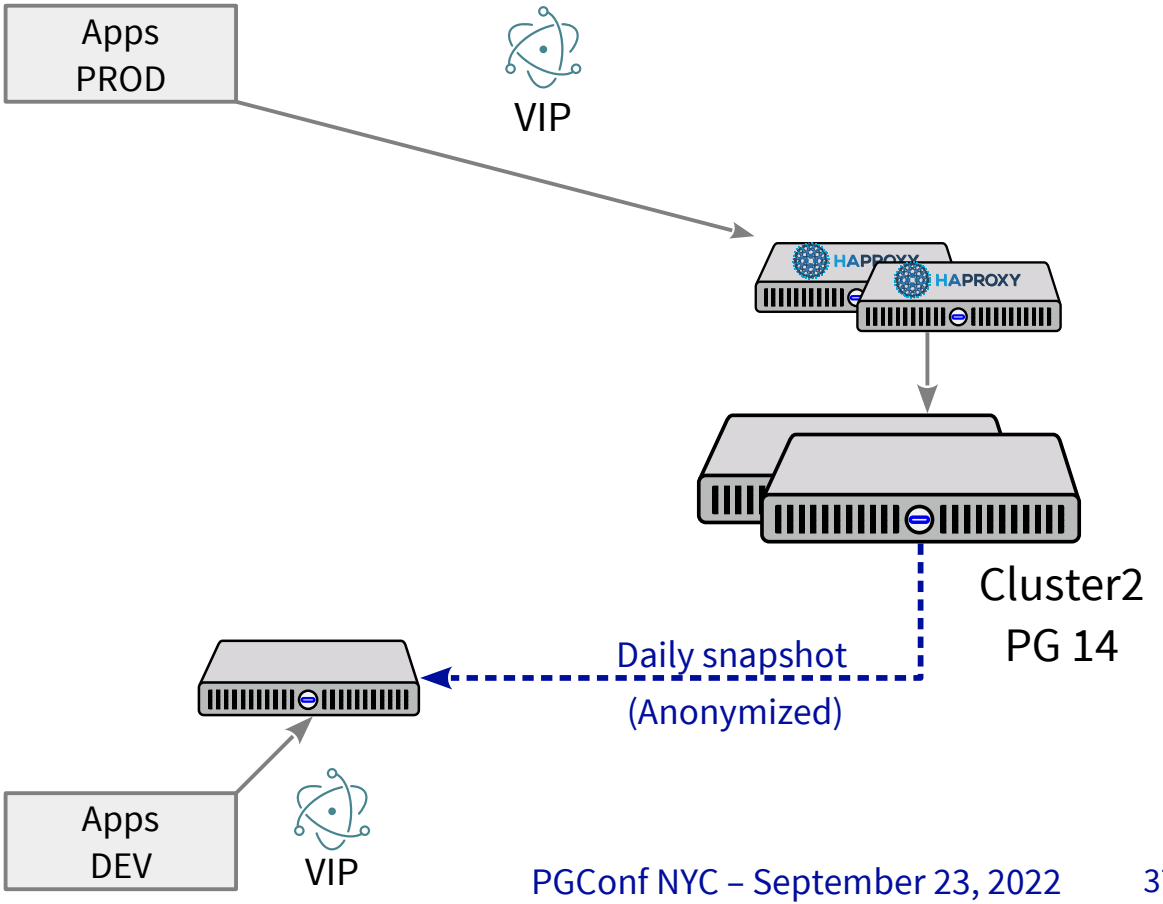
Logical replication

5. Switch PROD applications



Logical replication

6. Recycle previous cluster



Upgrade all clusters

Ansible

- Idempotent **playbooks** execution via SSH
- PostgreSQL modules
  - **postgresql_db:** create or delete databases
  - **postgresql_user:** create, alter or delete roles
  - **postgresql_privs:** grant or revoke privileges on objects
  - **postgresql_query:** arbitrary SQL query execution
- System modules (files, services, ...)

https://github.com/ansible/ansible
(Multiples licenses)

AWX

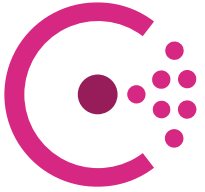
- **Ansible playbooks orchestration**
- Community release of **Ansible Tower**
- API REST, interface web, CLI
- SSO (SAML)
- Notifications
  - OpsGenie for alerting
  - Webex Teams for instant messages
- https://github.com/ansible/awx (Apache 2.0)

The Bastion

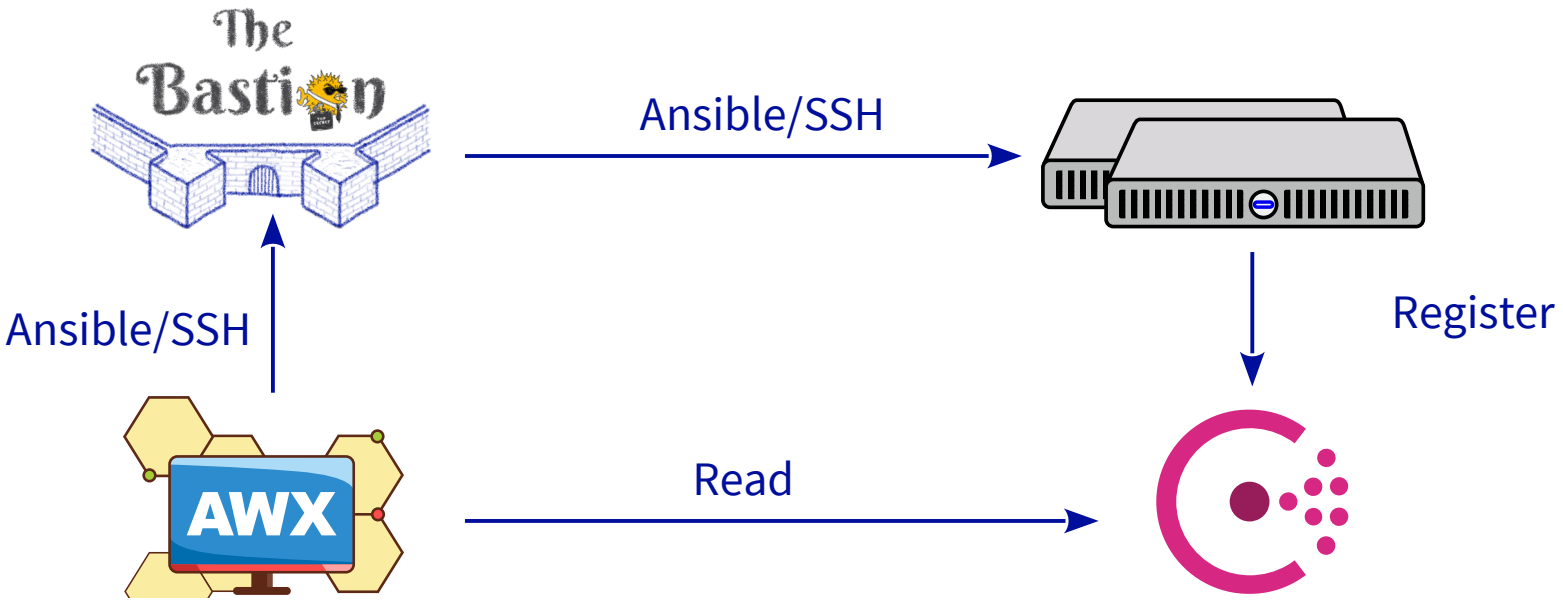
- SSH operations only through **The Bastion**
- **Fine-grained access** to the infrastructure
- **Sessions recorded** (ovh-ttyrec)
- Heavily used in **audited perimeters**
- https://github.com/ovh/the-bastion (Apache 2.0)
- https://github.com/ovh/the-bastion-ansible-wrapper (Apache 2.0)

Consul

- Dynamic **Ansible inventory** thanks to **Consul**
- Support of **node meta**, **service** and **services tag**
- Booleans interpretation
- https://github.com/wilfriedroset/consul-awx (MIT)



Overview



Logical replication playbooks

- **pglogical-create**
  - Setup logical replication between two **clusters**
- **cluster-migrate**
  - Move VIP from one cluster to another

pglogical-create

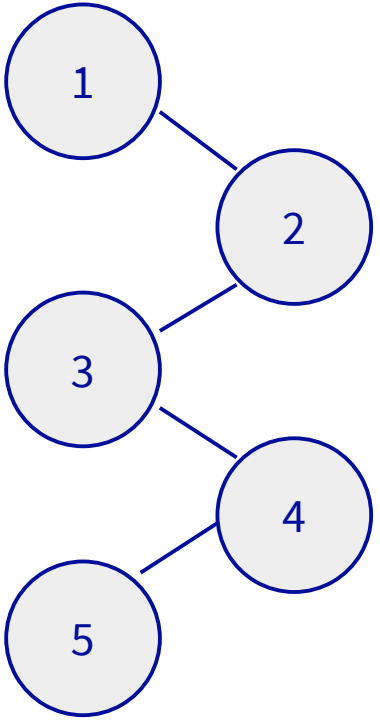
Setup logical replication between two clusters

Revoke DDL privileges from **source** cluster

Databases creation on **destination** cluster:

- CREATE DATABASE
- Dump and restore of schema from source

Creation of **application** and **bastion users** on **destination** cluster



Setup **pglogical** on **source** cluster on all databases:

- CREATE EXTENSION pglogical;
- “node” creation
- “set” creation (all tables and sequences of all schemas)

Setup **pglogical** on **destination** cluster on all databases:

- CREATE EXTENSION pglogical;
- “node” creation
- “subscription” creation

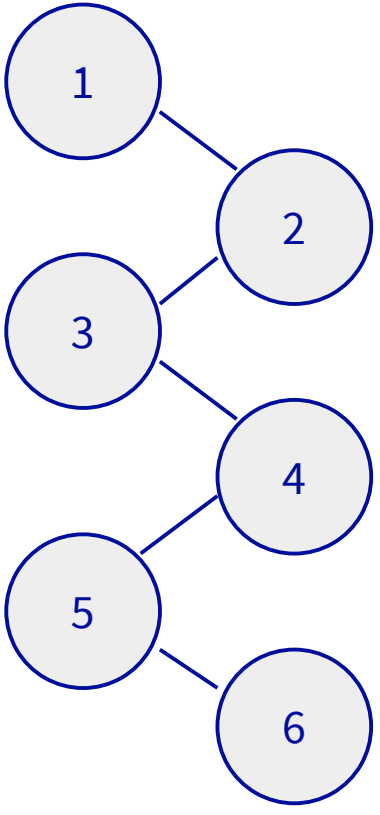
cluster-migrate

Migration from one PostgreSQL cluster to another using logical replication

Perform checks:
- Same databases
- Healthy subscriptions

Force **read-only** on **source**:
- `default_transaction_read_only=TRUE`
- Kill open sessions to force re-connection

Stop logical replication on **destination**:
- Re-check subscription health
- Delete subscriptions



Start keepalived on **destination**:
- VIP can be eligible but not promoted right away

Sequence synchronization:
- Using pglogical function
- Max value + 1000

Stop keepalived on **source**:
- VIP moves to the **destination** cluster (promotion)

Optimizations

Execution time of **pglogical-create**:

up to **1 hour**!

Optimizations

Simple iteration on all databases

```
- name: Ping databases
  postgresql_query:
    db: "{{ item }}"
    query: SELECT 1
  loop: "{{ databases_list }}"
```

Before (18 databases) → **42 seconds**

```
- name: Generate ping databases script
  template:
    src: databases-ping.sql.j2
    dest: /tmp/databases-ping.sql

# \set ON_ERROR_STOP true
# {% for database in databases_list %}
# \c {{ database }}
# SELECT 1;
# {% endfor %}

- name: Ping databases
  shell: psql < /tmp/databases-ping.sql
```

After (18 databases) → **14 seconds**

Optimizations

- **The Bastion Ansible Wrapper**
  - Call “`ansible-inventory --list`” (almost) every time
  - Takes between **1 and 3 seconds per call**
  - Implementation of a **cache** with `BASTION_ANSIBLE_INV_CACHE_FILE` and `BASTION_ANSIBLE_INV_CACHE_TIMEOUT` environment variables
- With cache (18 databases) → **5 seconds (-88%)**

pg_upgrade playbooks

- **primary-upgrade-check**
  - Check configurations and perform a pg_upgrade –check operation
- **primary-upgrade**
  - Stop replicas, setup and execute pg_upgrade on the primary
- **primary-upgrade-rollback**
  - Start replicas using previous version then reconfigure and start the primary
- **replica-upgrade**
  - Configure replicas to use the new version then start service

Attention points of the latest release

PostgreSQL 14 incompatibilities

- **Type changes**
  - `array_append(anyarray)` → `array_append(anycompatiblearray)`
  - `median(anyelement)` → `median(anycompatible)`
  - https://wiki.postgresql.org/wiki/Aggregate_Median

PostgreSQL 14 incompatibilities

- **Missing implicit oids**
  - **PG 9.6**

```
INSERT INTO pg_enum (enumtypid, enumsortorder, enumlabel)
VALUES (type_oid, sort_order, enum_value);
```

  - **PG 12+**

```
INSERT INTO pg_enum (oid, enumtypid, enumsortorder, enumlabel)
VALUES (pg_catalog.pg_nextoid('pg_catalog.pg_enum', 'oid',
'pg_catalog.pg_enum_oid_index'), type_oid, sort_order, enum_value);
```

PostgreSQL 14 incompatibilities

- Setup pglogical and **stop before starting the subscription**
- Update schema
- **Start the subscription**
- ~~Stop using pg_enum~~

Version too recent – External software

- Some **external software** don’t support the latest relase of PostgreSQL:

> ⚠ **Supported PostgreSQL Versions**
> Artifactory supports PostgreSQL version 13.x and below (9.5 and 9.6 were EOL in 2021).

Database	
PostgreSQL	✔ 13
	✔ 12
	✔ 11
	✔ 10
	✔ 9.6

Version too recent – External software

- Deployment of the latest **supported** release
- **PostgreSQL 13**

Version too recent – Debian packages

- Some **Debian packages** were **not available** at the beginning of the project
  - pglogical extension

Version too recent – Debian packages

- Deployment of **compiled binaries** on **test clusters**
- Official packages were available very quickly!

Version too recent – Internal software

- Some **internal tools** are not compatible with **Debian 11**:
  - **Installation images** with grsecurity patches
  - Schema migration binary
  - Session killer binary

Version too recent – Internal software

- Lower the priority of migrating to Debian 11
- Because **Debian 10 is still supported**

Insecure TLS ciphers

- PCI DSS hardening everywhere, blocking insecure TLS ciphers
- Applications running on old systems **still use deprecated TLS ciphers**
- Those applications are not planned to be PCI DSS certified

Insecure TLS ciphers

- **Disable TLS** with a **fine-grained** pg_hba.conf rule
- **Raise the priority** of an upgrade for those applications

Attention points of logical replication

Tables without primary or unique key

- Primary key or unique key **required** for logical replication

Tables without primary or unique key

- **Internal software**
  - Identify tables without primary or unique key
  - Add them

- **External software**
  - **pg_dump/pg_restore** for restore time under the minute or less critical databases
  - Otherwise, **pg_upgrade**

Maximum value for sequences

- **Constraint** on a column used by a sequence
- pglogical **adds 1000** to the max value at promotion
- **Insert errors**

Maximum value for sequences

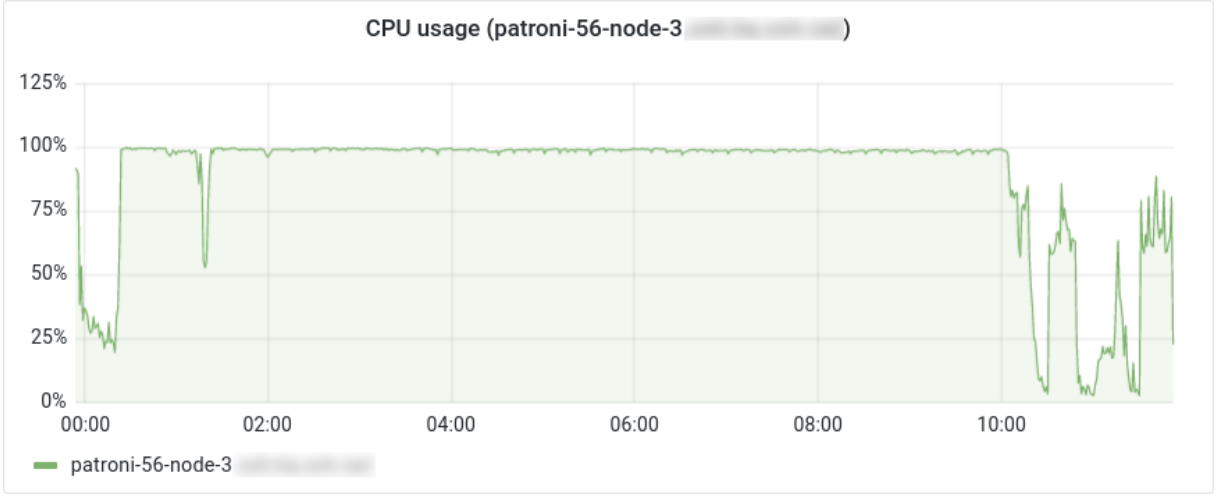
- Use maximum + 1 for the sequence number

```
SELECT max(c)+1 FROM t;
ALTER SEQUENCE … RESTART WITH …;
```

Load on the primary

- Logical replication connected to the **primary**
- Initialization phase using **COPY** operations running for **days**
  - Sequentially, table by table
- Nightly batches by the application
- Raise of **slow and consuming queries**
- 100% CPU

Load on the primary



Load on the primary

- Large tables are **not often updated**
- **pg_dump** of large tables from backup instance
- **pg_restore** on primary instance of the destination cluster
- Then **pglogical** setup
- Finally, manual integrity check

Dynamic tables

- Tables created every **1st of the month by the application**
- **DDLs forbidden** until the promotion day
- Migration scheduled for the **next month**

Dynamic tables

- Stop logical replication
- Allow DDLs
- Let the application create the monthly tables
- Setup logical replication

Attention points of the legacy

SQL_ASCII charset

- pglogical only supports the **UTF-8 charset**
- One database using **SQL_ASCII**
- More than **1TB**
- **Highly critical**

SQL_ASCII charset

- Use **pg_upgrade** to use version 14
- Convert data incompatible with UTF-8
- Use **built-in logical replication** from SQL_ASCII to UTF-8
  - Still in progress

What’s next?

The plan

- Use **built-in logical replication**
- Upgrade old applications
- **Restrict** the previously opened **pg_hba.conf and iptables rules**
- Upgrade to **Debian 11**
- Upgrade **more often**
  - By allowing multiple major releases at the same time
  - By migrating database by database
- Schedule **automatic major upgrades**

Special thanks

Thank you

Nicolas Payart
Project lead

Thank you all

The Team

Do you want to be part of this team?

We are hiring!

- **USA**
  - Contact me
- **France**
  - Database Reliability Engineer @ Critical Databases Team
  - Manager @ GIS Data Team
- **And more…**
  - OVHcloud US → https://us.ovhcloud.com/about/careers
  - OVHcloud Group → https://careers.ovhcloud.com/

Thanks for attending