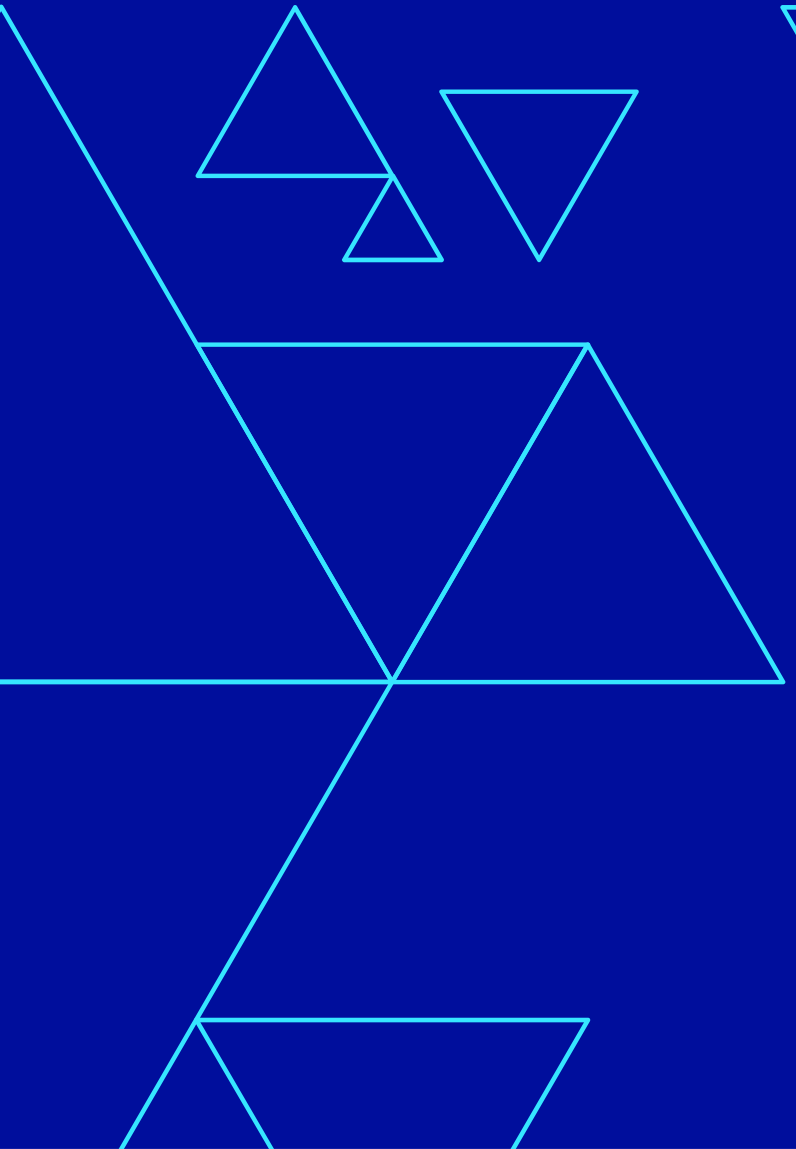


OVHcloud

# Automatisation d’une mise à jour éléphantesque

Julien Riou

PG Day France

22 Juin 2022

# Présentateur

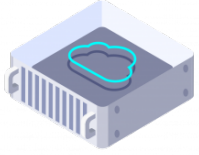
- Julien Riou
- DBA depuis 2012
- Tech lead chez OVHcloud depuis 2015
- https://julien.riou.xyz

# Contexte

# Qui sommes-nous ?

Leader européen du cloud

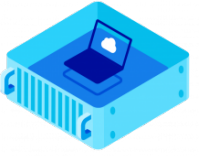
OVHcloud®



Bare Metal Cloud

Hosted Private Cloud

Public Cloud

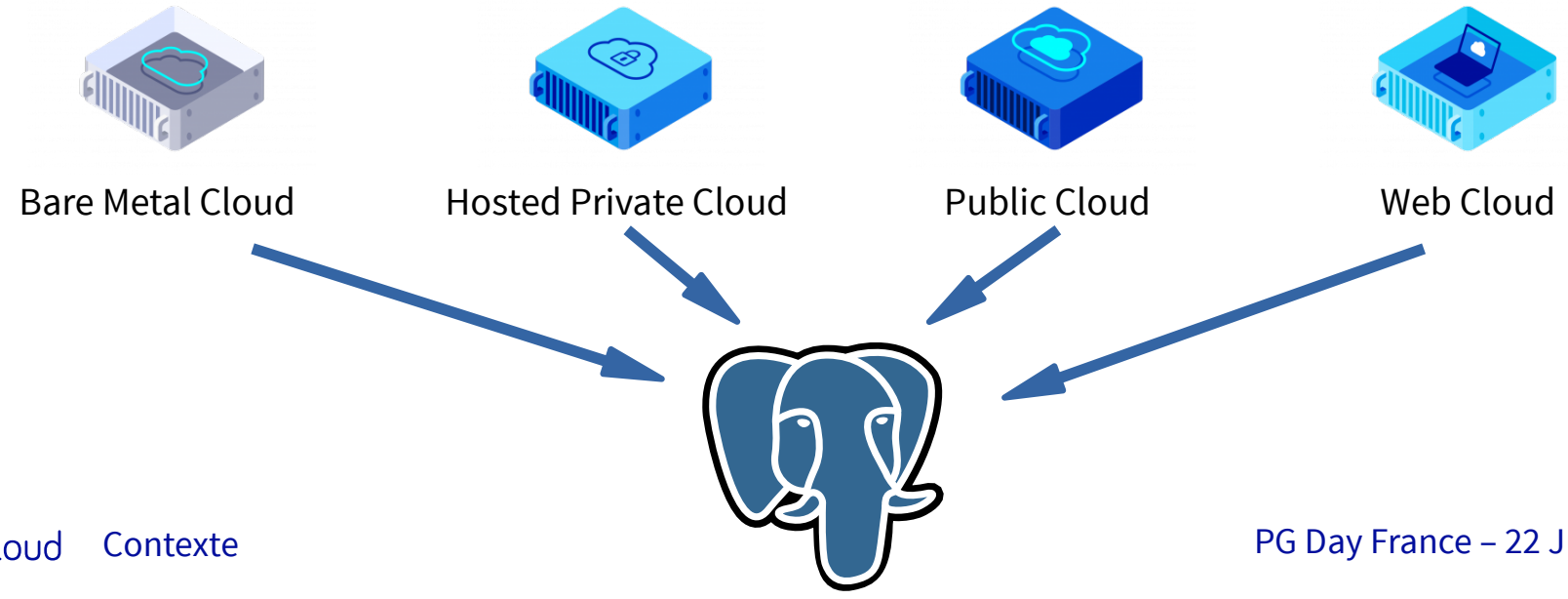


Web Cloud

# Qui sommes-nous ?

OVHcloud®

Leader européen du cloud

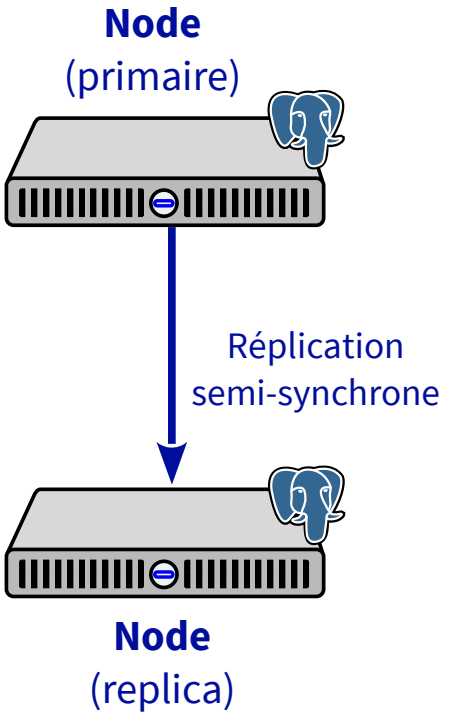


# L’infrastructure

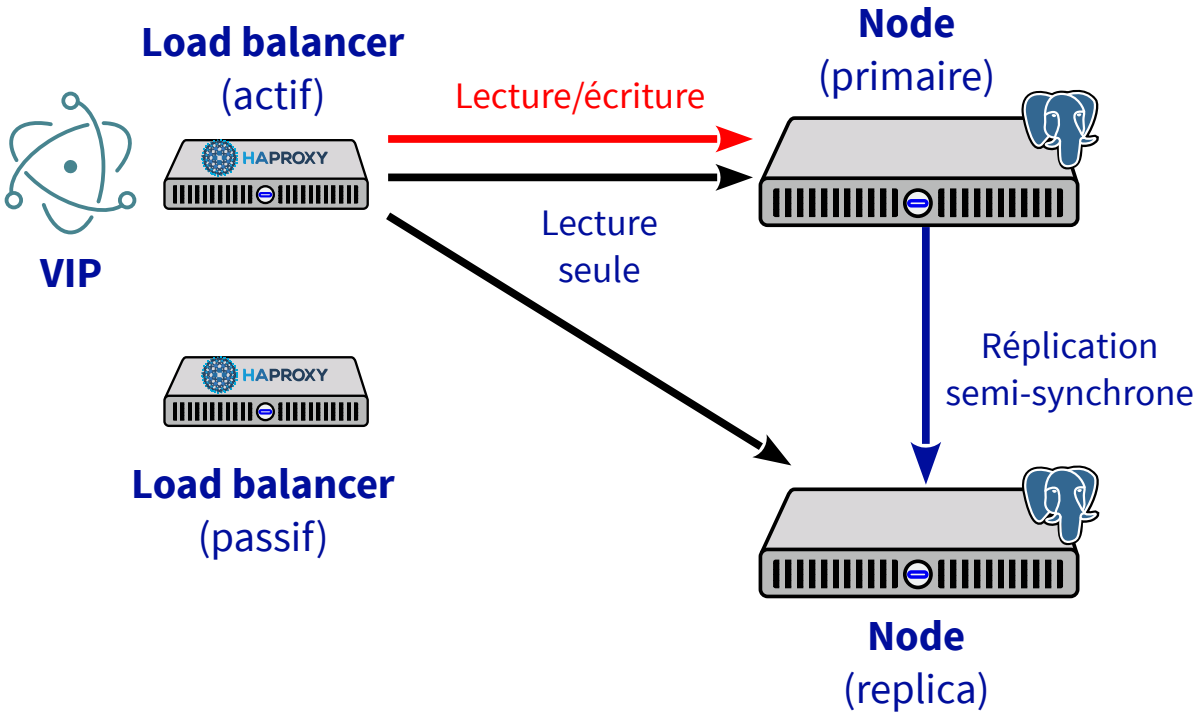
- **5 infrastructures** (productions, développement, etc)
- **230+ bases de données** PostgreSQL
- **50+ clusters** PostgreSQL
- Dont certains dans des périmètres **hautement sécurisés**

# Exemple de cluster

**Patroni**



# Exemple de cluster




Patroni

# Exemple de cluster

**Patroni**

**Load balancer**
(actif)

**VIP**

HAPROXY

Lecture/écriture

Lecture
seule

**Node**
(primaire)

Réplication
asynchrone

Réplication
semi-synchrone

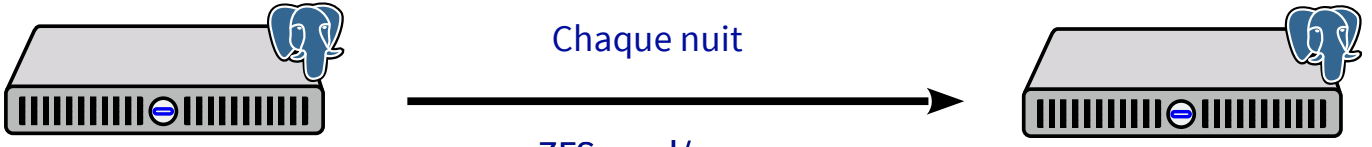
**Backup**
(replica)

HAPROXY

**Load balancer**
(passif)

**Node**
(replica)

# Exemple de cluster




**Backup**
(PROD)

**Dev anonymisé**
(DEV)

# Environnements hautement sécurisés

- Règles **firewall** fines
- Règles **pg_hba.conf** aux petits oignons
- **Chiffrement TLS** obligatoire à la connexion avec des **ciphers récents**
- **Chiffrement** des disques
- **Connexions/déconnexions** dans les logs
- Logs **externalisés** et **analysés** par le SIEM
- Accès **SSH via bastions** uniquement
- **MFA** (Yubikey PIV + mot de passe) activée sur de nombreux outils internes
- Supervision des **CVE**
- **Audits** chaque année

# Motivations

# 9.6


9.6

# Pourquoi mettre à jour ?

- **Fin du support** de la version 9.6 en **Novembre 2021**

- 5 nouvelles versions majeures disponibles avec :
  - Plus de **fonctionalités**
  - Plus de **performance**
  - Plus de **sécurité**

# Vers quelle version ?

La prochaine version majeure disponible au début du projet :

**14**

# Opportunités à saisir

- Mise à jour de la version du **système d'exploitation** (Debian 9 → 11)
- Remplacer les SSD par des disques **NVMe**
- Appliquer la **sécurité PCI DSS** sur 100% de notre périmètre
- Utiliser **Consul** comme Patroni DCS au lieu de ZooKeeper

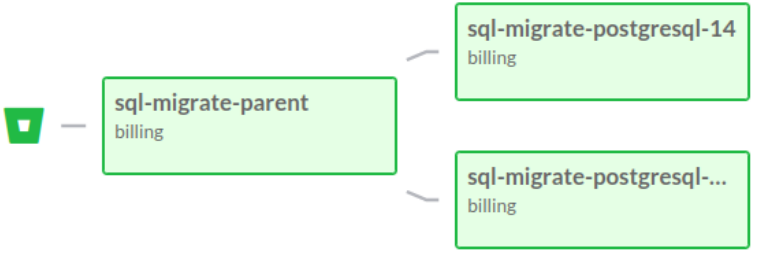
# Contraintes

- **Anticiper** au maximum **les incompatibilités** avec la nouvelle version
- Interruption de service le plus **proche de zéro**
- **Automatiser** le processus de migration (50+ clusters !)
- Avant le prochain **audit de sécurité** (FY22Q3)

# Avant la mise à jour

# Est-ce compatible ?

- Tests de validité des schémas avec **CDS**
- Tests des deux versions en même temps
- https://github.com/ovh/cds



# Mise à jour d’un cluster

# Outils disponibles

- pg_dump/pg_restore
- pg_upgrade
- Réplication logique

# pg_dump/pg_restore

- **pg_dump**
  - Export logique d’une base de données à un instant t
- **pg_restore**
  - Import d’un dump dans une base de données

# pg_dump/pg_restore

- Portabilité
- Données défragmentées
- Courte indisponibilité en écriture pour les petites bases
- Compatible avec tables sans clé primaire/unique

- Longue indisponibilité en écriture pour les bases volumineuses

# pg_upgrade

- Mise à jour du format des tables systèmes
- Ne touche pas aux données
- Redémarre sur les nouveaux binaires

# pg_upgrade

- Courte indisponibilité en écriture
  - Quelle que soit la volumétrie de données

- Rollback difficile
- Pas de défragmentation
- Pas portable

# Réplication logique

- **WAL** (*Write-Ahead Log*)
  - Ensemble des opérations d'écriture d'un cluster

- **Réplication physique** (*Streaming replication*)
  - Copie **bloc par bloc** le contenu des WAL d'une **instance** à une autre

- **Réplication logique** (*Logical replication*)
  - **Décode** le contenu des WAL pour extraire les changements **base par base**

# Réplication logique

- Plusieurs solutions de réplication PostgreSQL à PostgreSQL :
  - **pglogical** (9.4+)
  - **Réplication logique native** (*Built-in logical replication*, 10+)

# Réplication logique

- Plusieurs solutions de réplication PostgreSQL à PostgreSQL :
  - **pglogical** (9.4+)
  - **Réplication logique native** (*Built-in logical replication*, 10+)

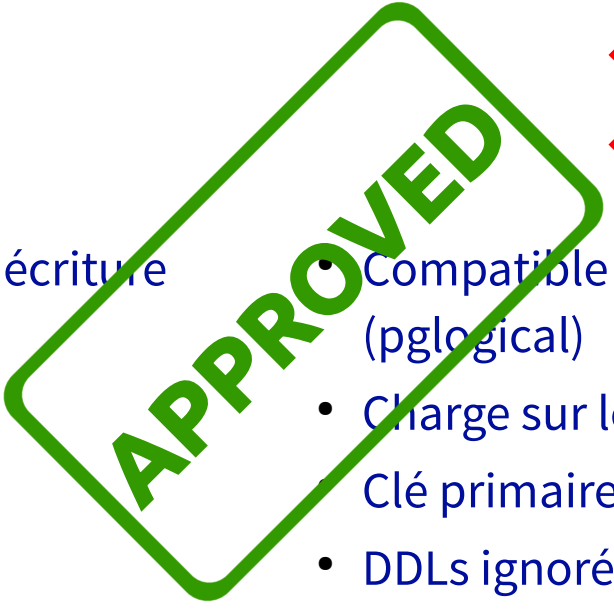
# Réplication logique

- Courte indisponibilité en écriture
- Portabilité
- Données défragmentées

- Compatible UTF-8 uniquement (pglogical)
- Charge sur le primaire à l'initialisation
- Clé primaire/unique obligatoire
- DDLs ignorés

# Réplication logique

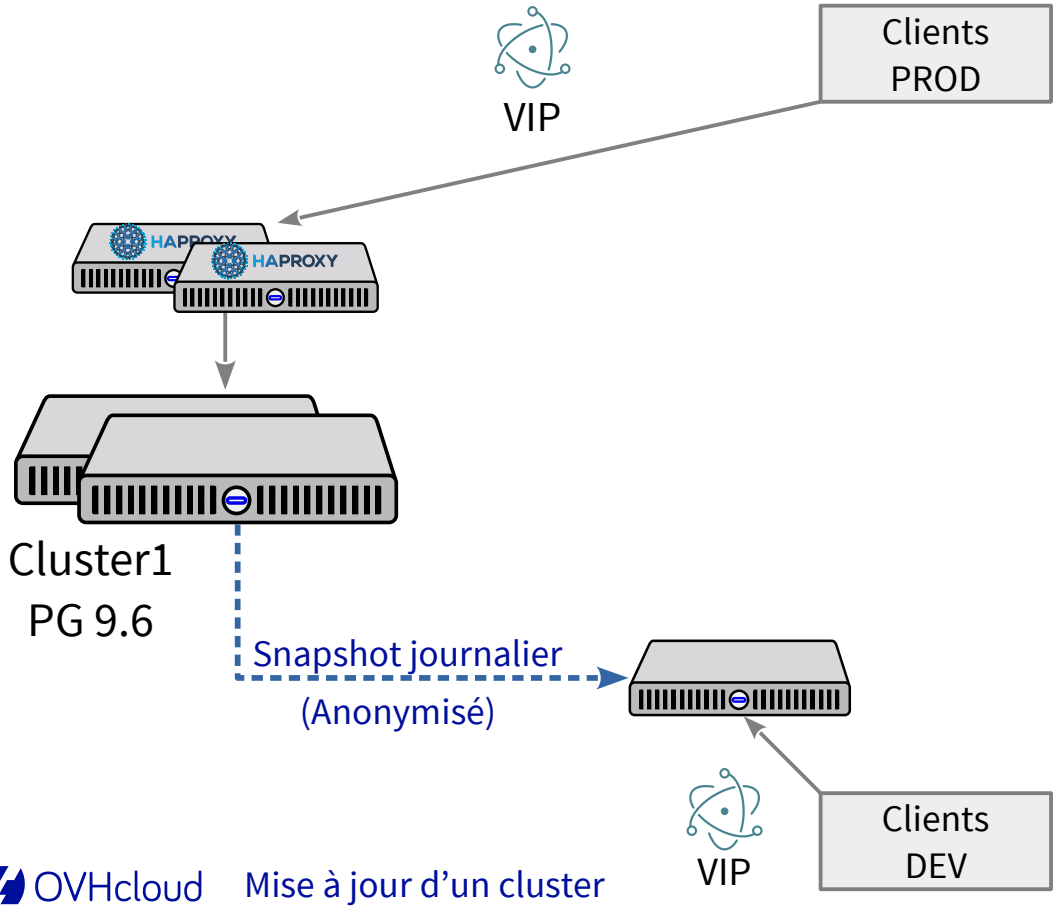
- Courte indisponibilité en écriture
- Portabilité
- Données défragmentées




- Compatible UTF-8 uniquement (pglogical)
- Charge sur le primaire à l'initialisation
- Clé primaire/unique obligatoire
- DDLs ignorés

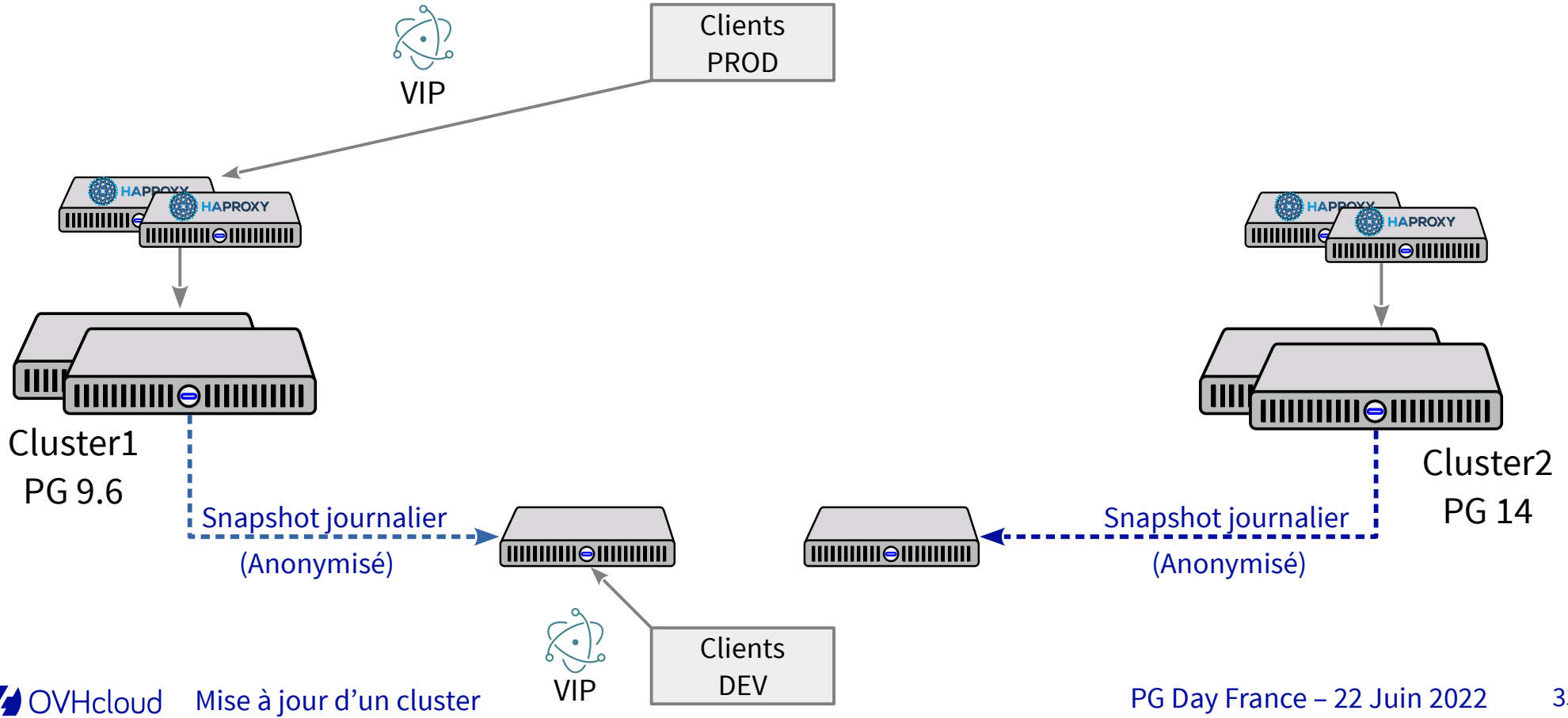
# Réplication logique

1. État initial



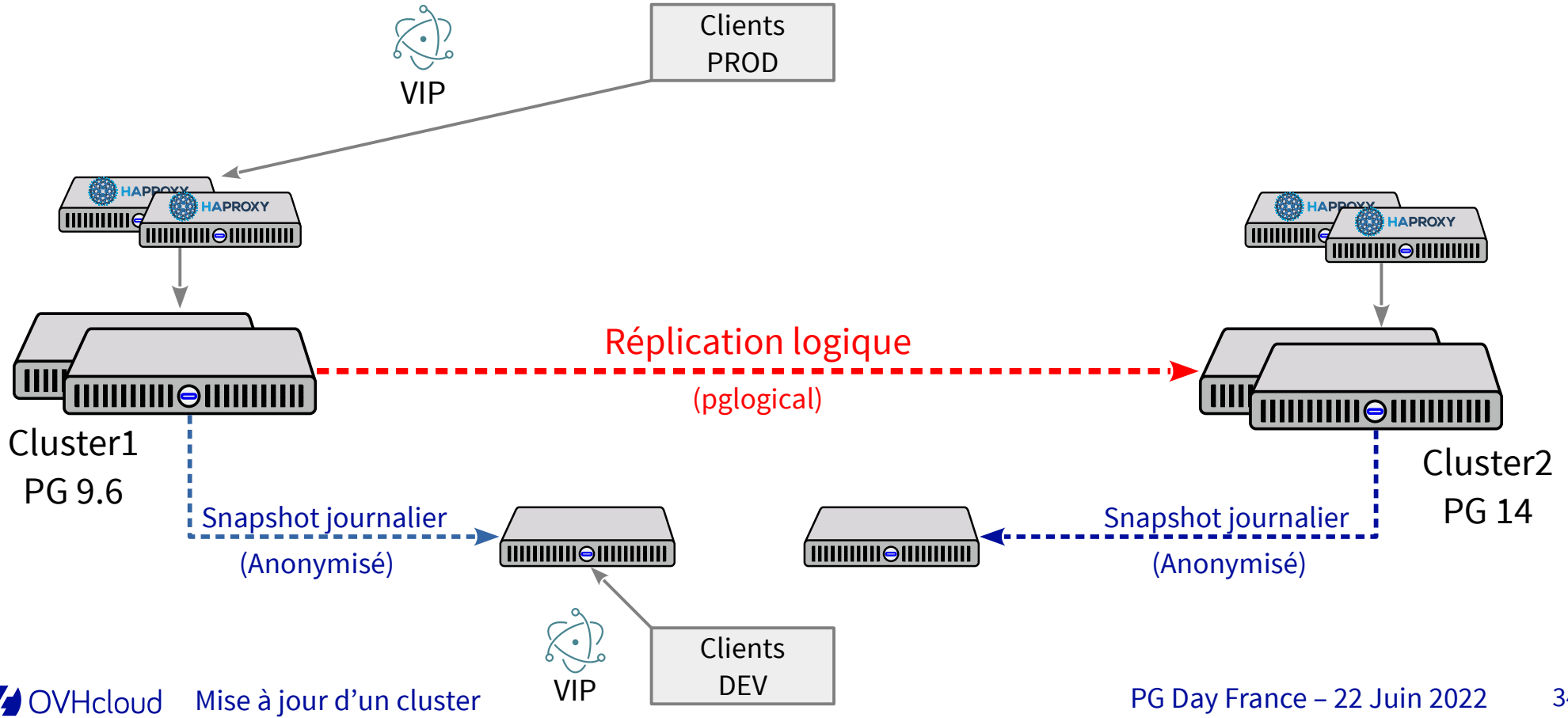
# Réplication logique

2. Ajout d'un cluster avec la nouvelle version



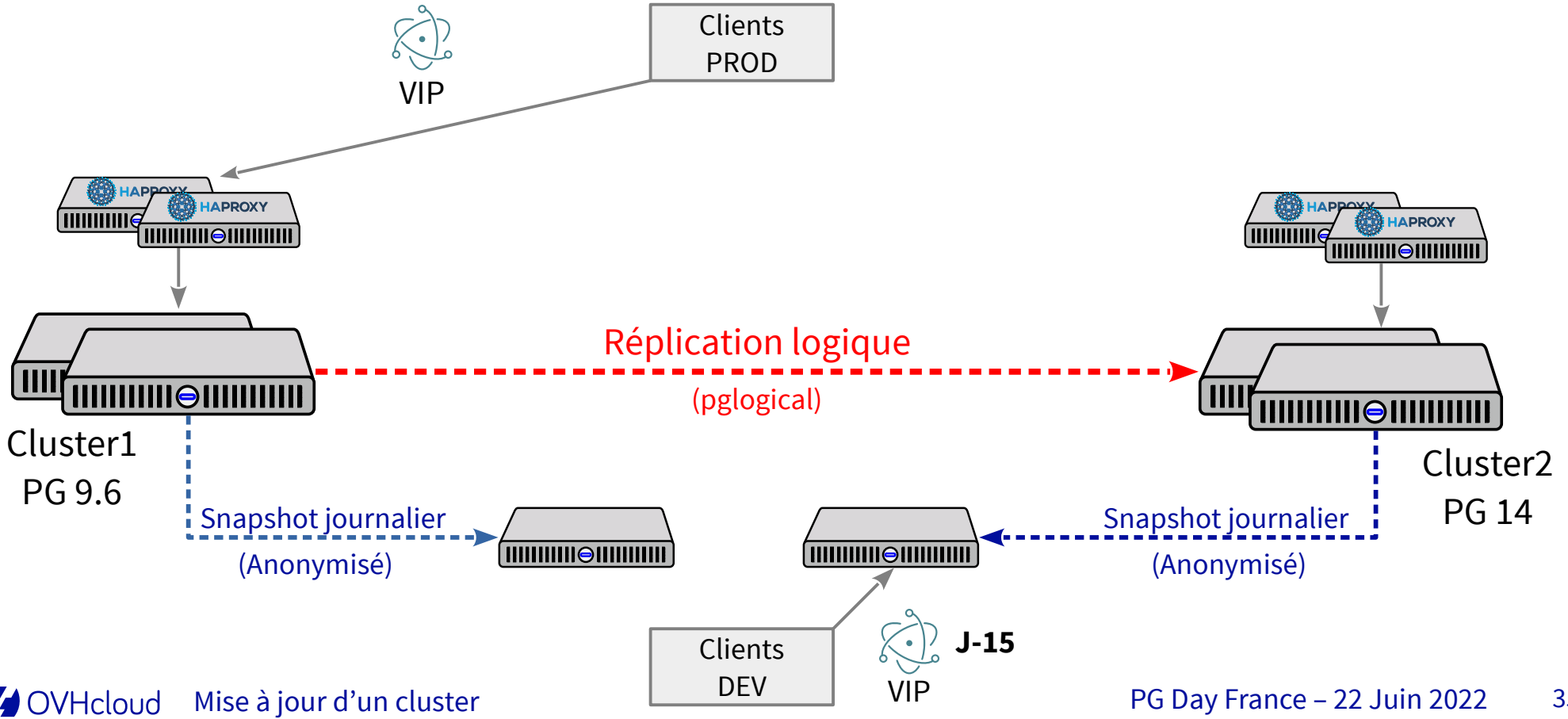
# Réplication logique





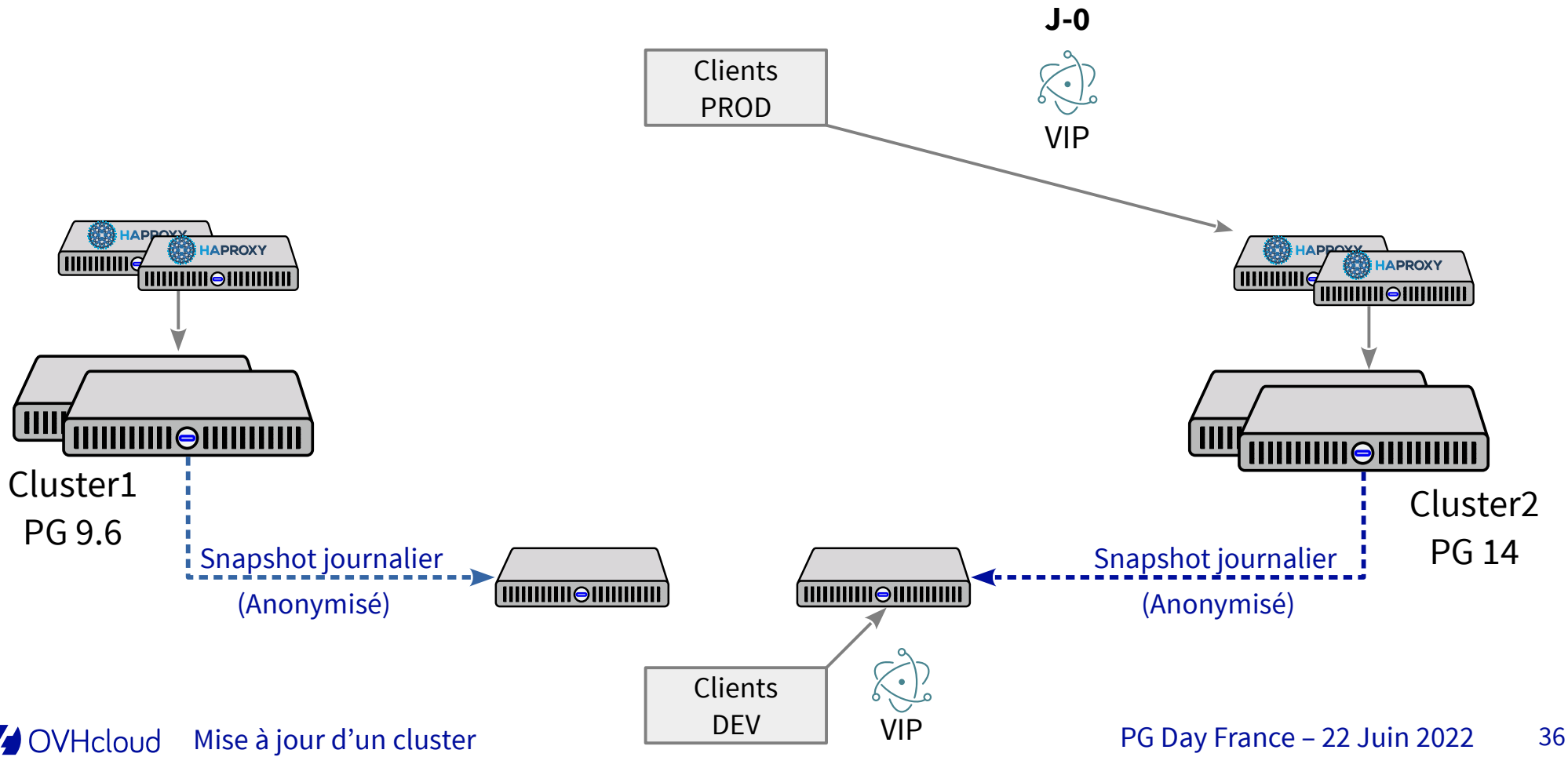
# Réplication logique





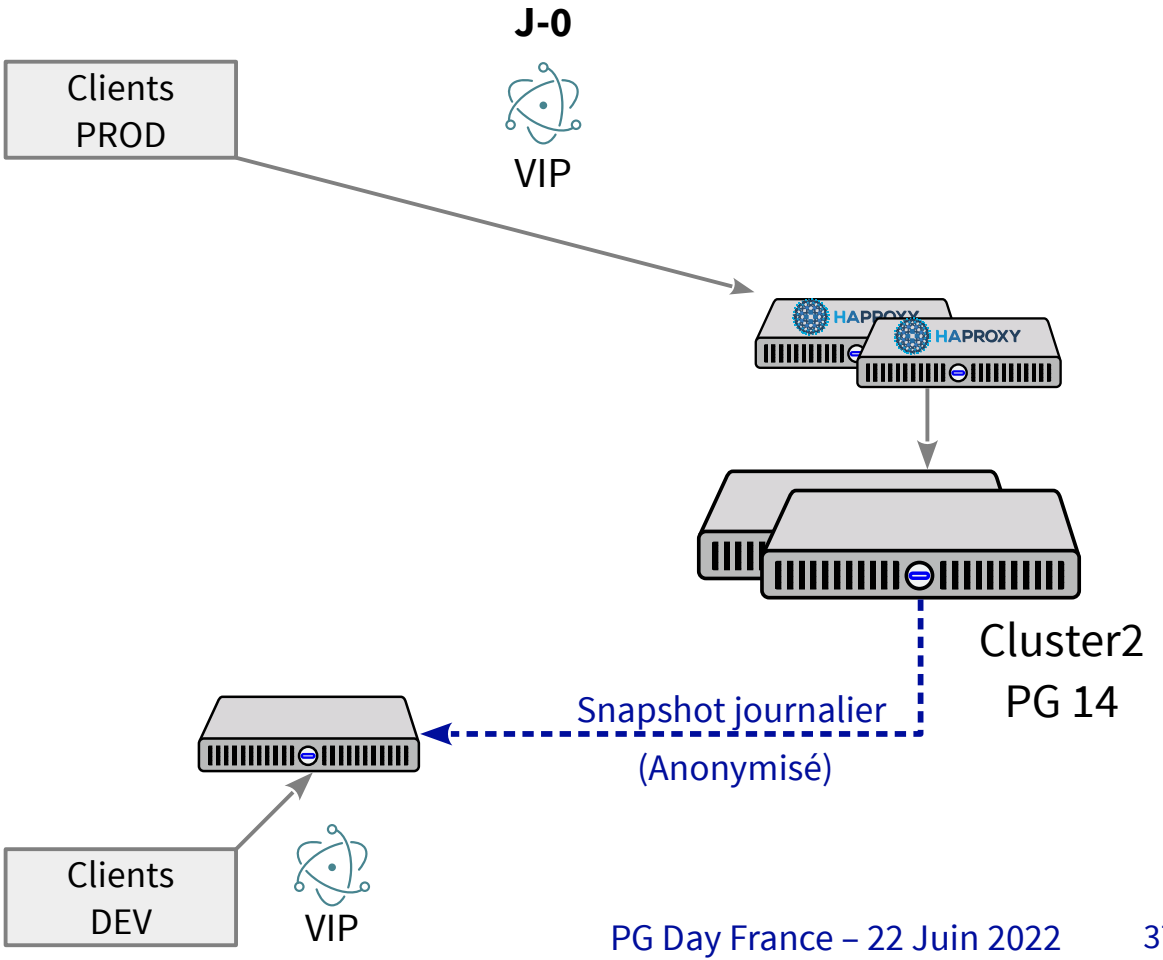
# Réplication logique

## 5. Bascule des clients en PROD



# Réplication logique

## 6. Recyclage de l'ancien cluster



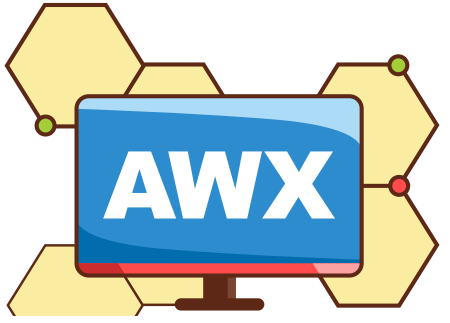
# Mise à jour de tous les clusters

# Ansible


- Exécution de **playbooks** idempotents via **SSH**
- Modules PostgreSQL
  - **postgresql_db :** créer ou supprimer des bases de données
  - **postgresql_user :** créer, altérer ou supprimer des roles
  - **postgresql_privs :** donner ou supprimer des droits d'accès sur des objets
  - **postgresql_query :** exécution de requête SQL
- Modules système (fichiers, services, ...)

https://github.com/ansible/ansible
(Licenses multiples)

# AWX



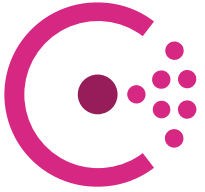

- **Orchestration de playbooks Ansible**
- Version communautaire d'**Ansible Tower**
- API REST, interface web, CLI
- Compatible SSO (SAML)
- Notifications
  - Compatible avec OpsGenie pour l'alerting
  - Compatible avec Webex Teams pour la messagerie instantanée
- https://github.com/ansible/awx (Apache 2.0)

# The Bastion

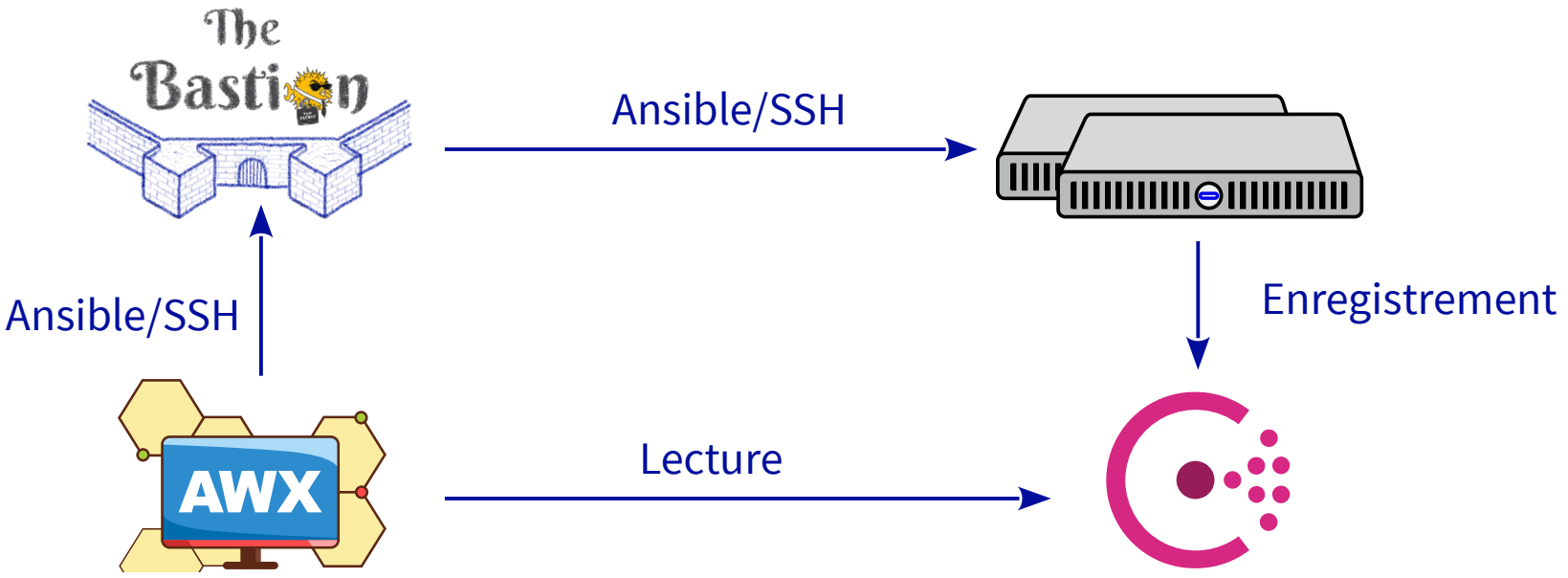
- Exécution d'opérations SSH à travers **The Bastion**
- **Accès fin** à l'infrastructure
- **Enregistrement** des sessions (ovh-ttyrec)
- Utilisé dans des environnements **sécurisés**
- https://github.com/ovh/the-bastion (Apache 2.0)
- https://github.com/ovh/the-bastion-ansible-wrapper (Apache 2.0)

# Consul

- **Inventaire Ansible** dynamique grâce à **Consul**
- Support des **node meta**, **service** et **services tag**
- Interprète les **booléens**
- https://github.com/wilfriedroset/consul-awx (MIT)



# Vue générale



# Playbooks de réplication logique

- Playbook **pglogical-create**
  - Configure une réplication logique entre deux clusters
- Playbook **cluster-migrate**
  - Bascule de VIP d'un cluster à un autre

# pglogical-create

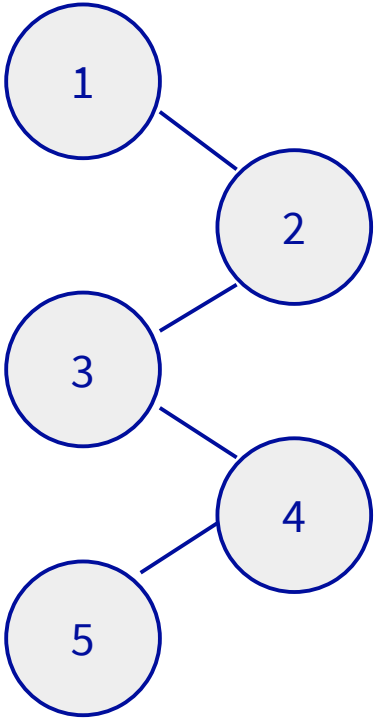
## Configuration d'une réplication logique entre deux clusters

**Suppression des droits DDL** sur le cluster **source**

**Création des bases de données** sur la **destination** :

- CREATE DATABASE
- Dump/restore du schéma depuis la source

Création des **utilisateurs applicatifs** et bastion sur la **destination**




Mise en place de **pglogical** sur le cluster **source** sur toutes les bases de données :

- CREATE EXTENSION pglogical;
- Création d'un « node »
- Création d'un « set » (toutes les tables et les séquences de tous les schémas)

Mise en place de **pglogical** sur le cluster **destination** sur toutes les bases de données :

- CREATE EXTENSION pglogical;
- Création d'un « node »
- Création d'une « subscription »

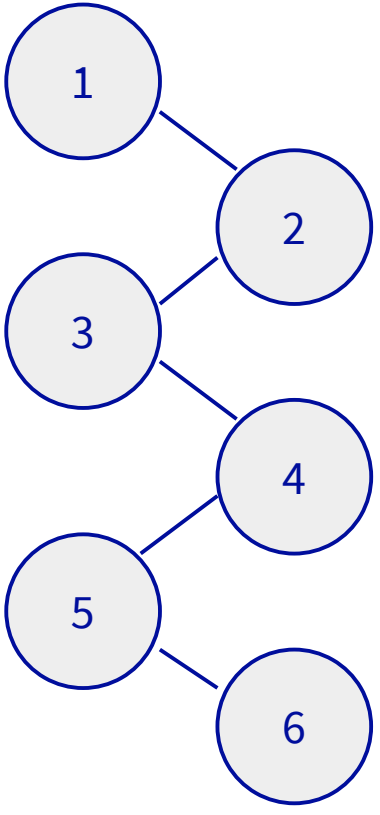
# cluster-migrate

Migration d'un cluster PostgreSQL vers un autre en réplication logique

**Vérification de l'installation** :
- Mêmes bases de données
- Subscriptions fonctionnelles

Passage en **lecture-seule** de la **source** :
- `default_transaction_read_only=TRUE`
- Kill des sessions ouvertes pour forcer la reconnexion

**Arrêt de la réplication logique** sur la destination :
- Re-vérification des subscriptions
- Suppression des subscriptions




**Démarrage de keepalived** sur la **destination** :
- VIP candidate à l'élection mais pas élue tout de suite

**Synchronisation des séquences** :
- Fonction intégrée à pglogical
- Valeur maximum + 1000

**Arrêt de keepalived** sur la **source** :
- Bascule de VIP vers la destination

# Optimisations

Durée d’exécution de **pglogical-create**

jusqu’à **1 heure** !

# Optimisations

## Simple itération sur l'ensemble des bases de données d'un cluster

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

Avant (18 bases) → **42 secondes**

```
- 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
```

Après (18 bases) → **14 secondes**

# Optimisations

- **The Bastion Ansible Wrapper**
  - Appel à « `ansible-inventory --list` » (presque) systématiquement
  - Entre 1 et 3 secondes par appel
  - Mise en place d'un **cache** avec `BASTION_ANSIBLE_INV_CACHE_FILE` et `BASTION_ANSIBLE_INV_CACHE_TIMEOUT`
- Avec cache pour 18 bases → **5 secondes (-88%)**

# Playbooks de pg_upgrade

- Playbook **primary-upgrade-check**
  - Vérifie que le cluster a les bonnes configurations pour pg_upgrade (--check)
- Playbook **primary-upgrade**
  - Arrête les réplicas, configure et exécute pg_upgrade sur le primaire
- Playbook **primary-upgrade-rollback**
  - Démarre les réplicas dans l'ancienne version, reconfigure et démarre le primaire
- Playbook **replica-upgrade**
  - Configure les réplicas pour démarrer Patroni sur la nouvelle version

# Particularités de la dernière version

# Incompatibilité avec PostgreSQL 14

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

# Incompatibilité avec PostgreSQL 14

- **Oids manquant à l'insertion**
  - **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);
```

# Incompatibilité avec PostgreSQL 14

- **pglogical** + modification de schéma avant démarrage de la souscription
- ~~Arrêter de jardiner pg_enum~~

# Version trop récente – Outils éditeur

- Certains **outils éditeur** ne supportent pas la dernière version de 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 trop récente – Outils éditeur

- Déploiement d’un cluster en version un peu moins récente mais supportée
- **PostgreSQL 13**

# Version trop récente – Paquets Debian officiels

- Certains **paquets Debian** n'étaient **pas disponibles** au début du projet
  - Extension pglogical

# Version trop récente – Paquets Debian officiels

- **Compilation** des binaires à destination des **clusters de test** uniquement
- Les paquets officiels sont arrivés très vite !

# Version trop récente – Paquets Debian internes

- Certains outils internes ne sont **pas disponibles sur Debian 11** :
  - **Images d'installation** avec les patchs grsecurity
  - Binaire de migration de schéma
  - Binaire de kill automatique des connexions idle

# Version trop récente – Paquets Debian internes

- Dé-priorisation du passage à Debian 11
- Utilisation de **Debian 10** encore supporté

# Anciens ciphers TLS

- Généralisation des configurations PCI DSS
- Certaines applications anciennes utilisent des **ciphers TLS dépréciés**
- Applications qui ne sont pas prévues pour être certifiées PCI DSS

# Anciens ciphers TLS

- **Désactivation du chiffrement** avec une **règle fine** dans le pg_hba.conf
- **Hausse de la priorité** de la mise à jour de ces clients

# Particularités de la réplication logique

# Tables sans clé primaire/unique

- Clé primaire ou contrainte d'unicité **requise** pour la **réplication logique**

# Tables sans clé primaire/unique

- Développement maison
  - Ajout par les équipes concernées

- Outil éditeur
  - **pg_dump/pg_restore** avec temps de restore inférieur à une minute ou pas critique
  - **pg_upgrade** pour les autres

# Valeur maximale pour des séquences

- **Contrainte** sur une colonne utilisée par une séquence
- pglogical détecte les **valeures maximales** utilisées par chaque séquence
- Et ajoute **1000**
- **Erreur** lors des insertions

# Valeur maximale pour des séquences

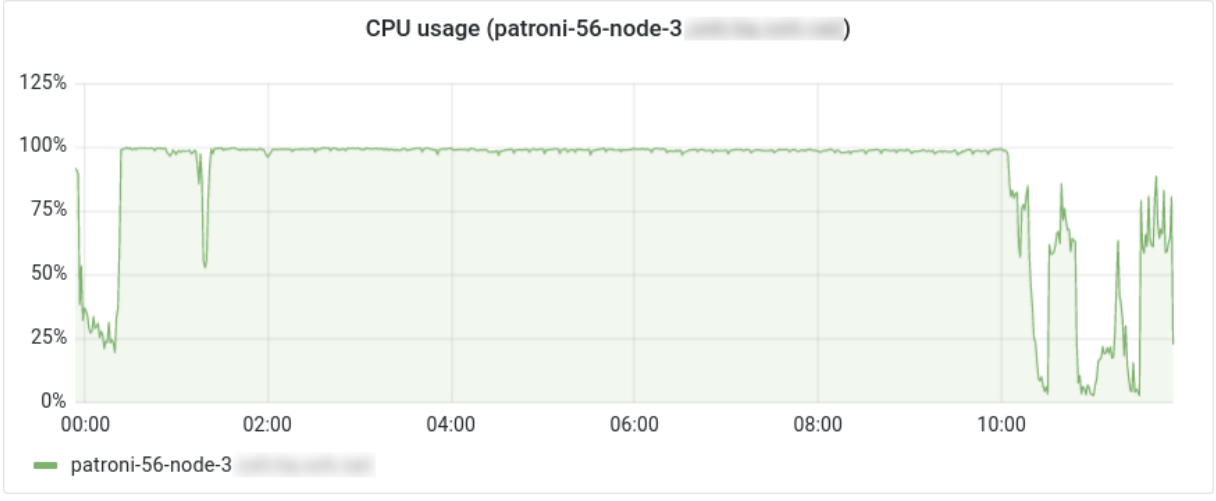
- Changement de la valeur courante à la valeur maximale + 1

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

# Charge sur le primaire

- Réplication logique connectée à l'instance **primaire**
- Phase d'initialisation avec **COPY** qui dure **plusieurs jours**
  - Séquentiellement, table par table
- Batchs de nuit par l'application
- Augmentation du nombre de **requêtes lentes** et consommatrices
- 100% CPU

# Charge sur le primaire



# Charge sur le primaire

- Les tables volumineuses sont **peu souvent modifiées**
- **pg_dump** des tables volumineuses depuis le serveur de **backup**
- **pg_restore** sur l'instance primaire du **nouveau cluster**
- Puis utilisation de **pglogical**
- Avec vérification manuelle de l'intégrité des données

# Tables dynamiques

- Nouvelles tables créées **le premier du mois par l'application**
- Désactivation des DDLs jusqu'au moment de la migration
- Migration prévue **le mois suivant**

# Tables dynamiques

- Arrêt de la réplication logique
- GRANT du role qui permet les DDLs
- Création des tables mensuelles par l’application
- Démarrage d’une nouvelle réplication logique

# Particularités du legacy

# Base en SQL_ASCII

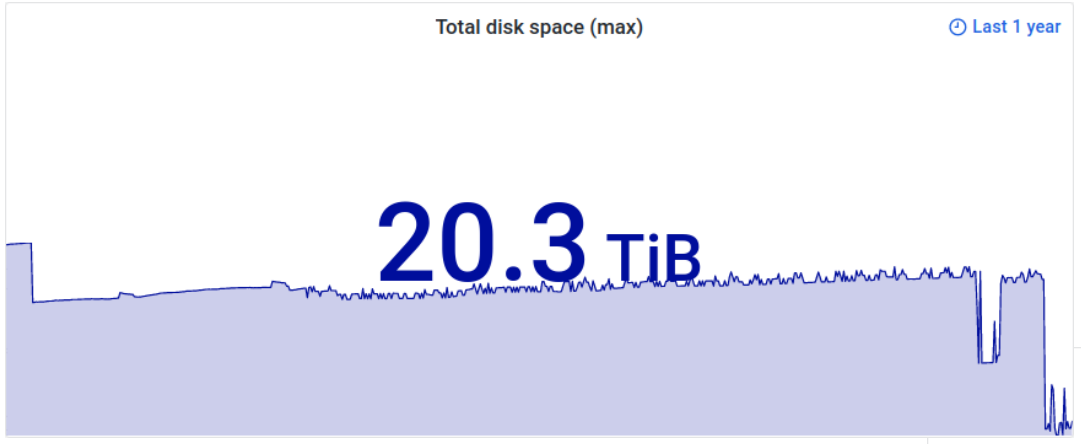
- pglogical **ne supporte que** le jeu de caractères **UTF-8**
- Une de nos bases de données est en **SQL_ASCII**
- Avec une taille de **plus de 1To**
- Base **hautement critique**

# Base en SQL_ASCII

- **pg_upgrade** pour passer en version 14
- Correction des données non-compatibles UTF-8
- **Réplication logique native** pour corriger le jeu de charactères
  - Toujours en cours

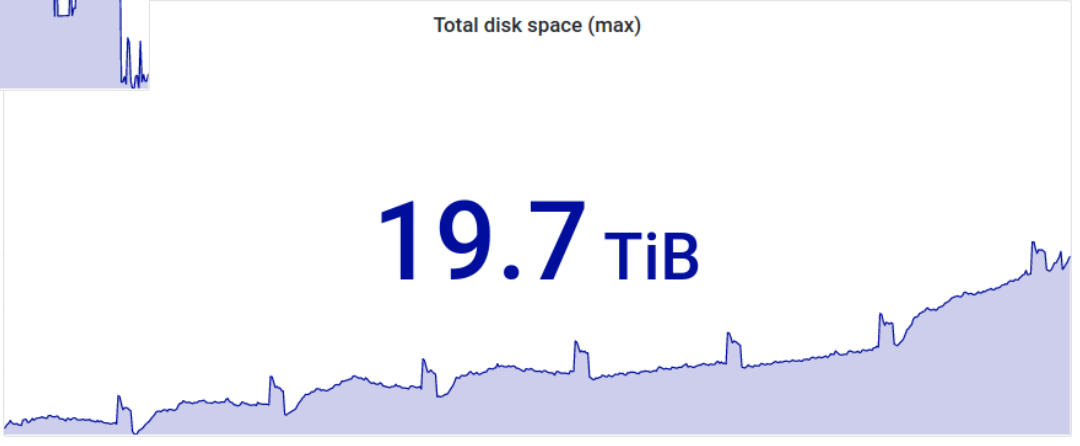
# Métriques de fin

# Espace disque




Avant

Après



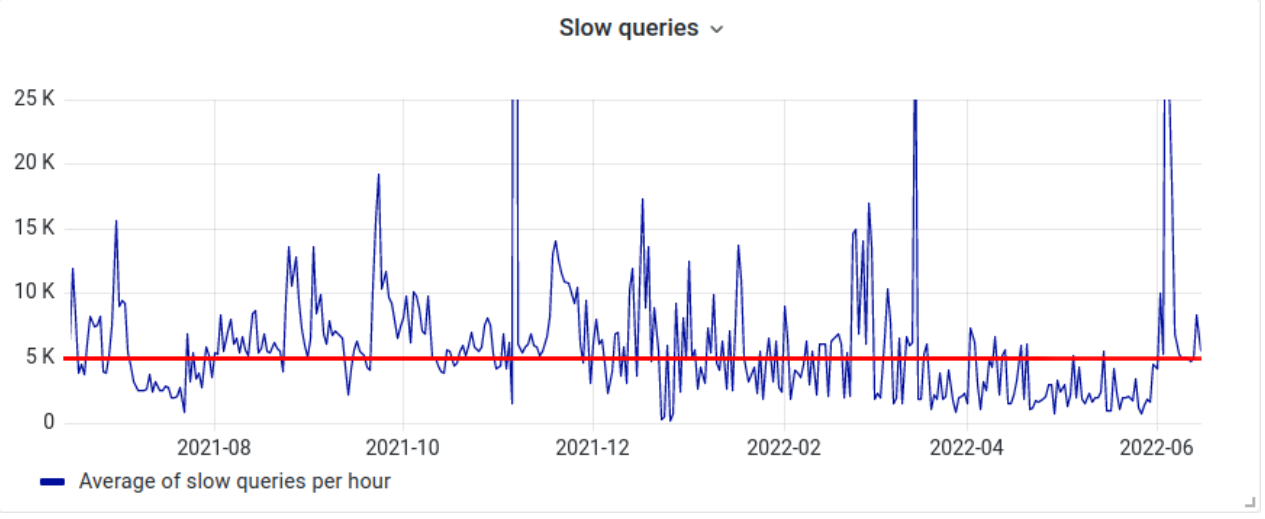

Espace occupé par la partition qui contient les données de tous les noeuds PostgreSQL

Valeur maximale sur 1 semaine entre il y a **1 an** et **aujourd'hui**

# Espace disque

- Quelques gigaoctets de **gagnés**
- Malgré une **croissance** de l'espace disque occupé par les données de **14,5%** (sur les derniers 6 mois avant mise à jour)

# Requêtes lentes




Moyenne du nombre de requêtes lentes détecté **chaque heure** sur les clusters PostgreSQL de production

# Requêtes lentes

- **Légère baisse** en dessous des 5000 requêtes lentes par heure
- Malgré la réduction du seuil de détection de **1 s** à **750 ms**
- Malgré l'augmentation du **nombre de bases de données**
- Malgré le passage à la **réplication entièrement synchrone** sur une partie des clusters
- **Pas de corrélation directe** avec la mise à jour majeure

# La suite

# Que faire ensuite ?

- Utiliser la **réplication logique native**
- **Affiner certaines règles** pg_hba.conf et iptables
- Mettre à jour les **anciens clients**
- Mettre à jour vers **Debian 11**
- Mettre à jour **plus régulièrement**
  - Plusieurs versions majeures en même temps au lieu d'une seule
  - Migration base par base
- Maintenance **automatique programmée**

# Remerciements

# Merci

**Nicolas Payart**
Lead sur le projet

# Merci

**Toute l’équipe**

# Pour nous rejoindre

# On recrute

- Database Reliability Engineer H/F @ Équipe Critical Databases
- Manager - Database Opensource H/F @ Équipe GIS Data
- Administrateur Bases de données - Opensource H/F @ Équipe GIS Data
- Et plus encore ! https://careers.ovhcloud.com/

# Merci à tous