

Workshop Patroni

Haute disponibilité avec Patroni



Dalibo & Contributors

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

Workshop Patroni

TITRE : Haute disponibilité avec Patroni

SOUS-TITRE : Workshop Patroni

REVISION: 19.02

DATE: février 2023

LICENCE: PostgreSQL

1 HAUTE DISPONIBILITÉ DE SERVICE AVEC PATRONI

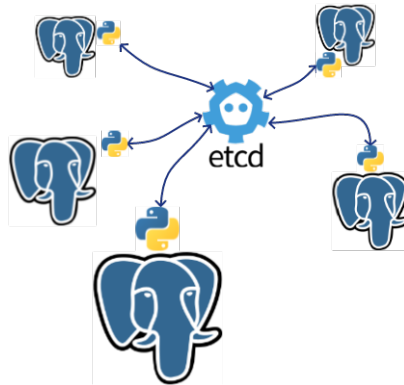


Figure 1: PostgreSQL

1.1 INTRODUCTION

- Principes
 - Mise en place
 - Installation et configuration des services
 - Construction d'un agrégat à bascule automatique
 - Création d'incidents
-

1.2 PRINCIPES

- Arbitrage par un quorum : DCS Etcd
 - Service PostgreSQL : désactivé
 - Contrôle complet par Patroni
-

Haute disponibilité avec Patroni

1.2.1 DCS : ETCD

- Arbitre en cas de bascules
- Stockage distribué de la configuration
- Jeton *leader* (Etcd)
- Instance primaire PostgreSQL

Pour arbitrer les bascules automatiques, confirmer le primaire PostgreSQL ou distribuer la configuration, Patroni utilise un *DCS* (*distributed configuration system*).

Pour ce rôle, nous utiliserons Etcd.

1.3 MISE EN PLACE DE L'INFRASTRUCTURE

- Connexion à la VM
- Récupération du *playbook Ansible*

Vous disposez d'une machine virtuelle dédiée dans laquelle nous construirons 7 conteneurs *lxc* :

- 3 Etcd
- 3 Patroni
- 1 backup optionnel : (sauvegardes, archivage)

1.3.1 CONNEXION À VOTRE MACHINE VIRTUELLE

un seul point d'entrée : eformation.dalibo.com un port attribué : 22XX

```
$ ssh -p 22XX dalibo@eformation.dalibo.com
```

Exemple de configuration pour une connexion simplifiée :

```
# .ssh/config
```

```
Host vm38
  Hostname eformation.dalibo.com
  User dalibo
  port 2238
```

```
$ ssh vm38
```

```
Last login: Wed Nov 10 13:23:26 2021 from 78.213.160.12
dalibo@vm38:~$
```

1.3.2 PLAYBOOK ANSIBLE

Récupération du *playbook* Ansible à cette adresse :

https://github.com/dalibo/workshops/tree/ws15_patroni/fr/patroni/playbook/etcd-patroni

Fichier	Description
inventory.yml	inventaire des machines
setup.yml	playbook principal
exchange_ssh_keys.yml	échange des clefs ssh
demarre_tout.sh	démarre tous les conteneurs
stoppe_tout.sh	arrête tous les conteneurs
teardown.yml	playbook de destruction massive

Quatre fichiers Yaml, deux scripts shell.

L'infrastructure complète peut être créée à l'aide des commandes :

```
$ sudo apt install -y ansible
$ sudo ansible-playbook -f 7 -i inventory.yml setup.yml
...
```

Cette opération peut durer jusqu'à une vingtaine de minutes.

Vous pouvez suivre l'évolution de la création des conteneurs dans un autre terminal :

```
$ watch -n 1 -d sudo lxc-ls -f

$ sudo lxc-ls -f
NAME    STATE   AUTOSTART  GROUPS IPV4  IPV6  UNPRIVILEGED
backup  STOPPED 0         -      -    -      false
e1      STOPPED 0         -      -    -      false
e2      STOPPED 0         -      -    -      false
e3      STOPPED 0         -      -    -      false
pg-1    STOPPED 0         -      -    -      false
pg-2    STOPPED 0         -      -    -      false
pg-3    STOPPED 0         -      -    -      false
```

L'état final de chaque conteneur étant *RUNNING* avec une adresse *IPV4* attribuée :

```
$ sudo lxc-ls -f
NAME    STATE   AUTOSTART  GROUPS IPV4        IPV6  UNPRIVILEGED
backup  RUNNING 0         -      10.0.3.204 -      false
e1      RUNNING 0         -      10.0.3.101 -      false
e2      RUNNING 0         -      10.0.3.102 -      false
e3      RUNNING 0         -      10.0.3.103 -      false
```

Haute disponibilité avec Patroni

pg-1	RUNNING	0	-	10.0.3.201	-	false
pg-2	RUNNING	0	-	10.0.3.202	-	false
pg-3	RUNNING	0	-	10.0.3.203	-	false

Sur toutes les machines, y compris l'hôte, le fichier `/etc/hosts` est automatiquement renseigné par le *playbook* et devrait contenir au moins :

```
10.0.3.101 e1
10.0.3.102 e2
10.0.3.103 e3
10.0.3.201 pg-1
10.0.3.202 pg-2
10.0.3.203 pg-3
10.0.3.204 backup
```

1.3.3 INSTALLATION D'ETCD

- Installation des paquets
- Configuration
- Démarrage du service
- Vérification

1.3.3.1 Installation des paquets

- Paquets essentiels :
 - etcd
 - curl
 - jq
 - iputils-ping

```
$ for node in e1 e2 e3; do
sudo ssh $node sudo apt-get install -qqy etcd curl iputils-ping jq
done
```

Le démarrage du service est automatique sous Debian.

```
$ for node in e1 e2 e3; do
sudo ssh $node "systemctl status etcd | grep -i active"
done
```

```
Active: active (running) since Wed 2021-11-10 17:48:26 UTC; 3min 28s ago
Active: active (running) since Wed 2021-11-10 17:48:36 UTC; 3min 18s ago
Active: active (running) since Wed 2021-11-10 17:48:46 UTC; 3min 8s ago
```

Vérification de l'état des nœuds

```
$ for node in e1 e2 e3; do
sudo ssh $node etcdctl member list
done

8e9e05c52164694d: name=e1 peerURLs=http://localhost:2380
clientURLs=http://localhost:2379 isLeader=true
8e9e05c52164694d: name=e2 peerURLs=http://localhost:2380
clientURLs=http://localhost:2379 isLeader=true
8e9e05c52164694d: name=e3 peerURLs=http://localhost:2380
clientURLs=http://localhost:2379 isLeader=true
```

Les nœuds sont tous des *leaders* indépendants, ce qui ne nous intéresse pas. Il faut donc les configurer pour qu'ils fonctionnent en collaboration.

Nous arrêtons donc les services :

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

```
$ for node in e1 e2 e3; do
sudo ssh $node "systemctl stop etcd && systemctl status etcd | grep -i active"
done
```

```
Active: inactive (dead) since Wed 2021-11-10 17:59:35 UTC; 2min 46s ago
Active: inactive (dead) since Wed 2021-11-10 17:59:35 UTC; 2min 46s ago
Active: inactive (dead) since Wed 2021-11-10 17:59:35 UTC; 2min 46s ago
```

1.3.3.2 Configuration du service Etcd

- Fichier : `/etc/default/etcd`

La configuration du service Etcd se trouve dans le fichier `/etc/default/etcd`, elle doit décrire notre agrégat sur chaque nœud :

Attention aux caractères invisibles ou aux sauts de ligne

Sur le nœud e1 :

```
ETCD_NAME='e1'
```

```
ETCD_DATA_DIR='/var/lib/etcd/default'
```

```
ETCD_LISTEN_PEER_URLS='http://127.0.0.1:2380,http://10.0.3.101:2380'
```

```
ETCD_LISTEN_CLIENT_URLS='http://127.0.0.1:2379,http://10.0.3.101:2379'
```

```
ETCD_INITIAL_ADVERTISE_PEER_URLS='http://10.0.3.101:2380'
```

```
ETCD_INITIAL_CLUSTER_STATE='new'
```

```
ETCD_INITIAL_CLUSTER_TOKEN='etcd-cluster'
```

```
ETCD_INITIAL_CLUSTER='e1=http://10.0.3.101:2380,e2=http://10.0.3.102:2380,e3=http://10.0.3.103:2380'
```

```
ETCD_ADVERTISE_CLIENT_URLS='http://10.0.3.101:2379'
```

```
ETCD_ENABLE_V2=true
```

Sur le nœud e2 :

```
ETCD_NAME='e2'
```

```
ETCD_DATA_DIR='/var/lib/etcd/default'
```

```
ETCD_LISTEN_PEER_URLS='http://127.0.0.1:2380,http://10.0.3.102:2380'
```

```
ETCD_LISTEN_CLIENT_URLS='http://127.0.0.1:2379,http://10.0.3.102:2379'
```

```
ETCD_INITIAL_ADVERTISE_PEER_URLS='http://10.0.3.102:2380'
```

1.3 Mise en place de l'infrastructure

```
ETCD_INITIAL_CLUSTER_STATE='new'
ETCD_INITIAL_CLUSTER_TOKEN='etcd-cluster'

ETCD_INITIAL_CLUSTER='e1=http://10.0.3.101:2380,e2=http://10.0.3.102:2380,e3=http://10.0.3.103:2380'

ETCD_ADVERTISE_CLIENT_URLS='http://10.0.3.102:2379'

ETCD_ENABLE_V2=true
```

Sur le nœud e3 :

```
ETCD_NAME='e3'

ETCD_DATA_DIR='/var/lib/etcd/default'

ETCD_LISTEN_PEER_URLS='http://127.0.0.1:2380,http://10.0.3.103:2380'
ETCD_LISTEN_CLIENT_URLS='http://127.0.0.1:2379,http://10.0.3.103:2379'
ETCD_INITIAL_ADVERTISE_PEER_URLS='http://10.0.3.103:2380'

ETCD_INITIAL_CLUSTER_STATE='new'
ETCD_INITIAL_CLUSTER_TOKEN='etcd-cluster'

ETCD_INITIAL_CLUSTER='e1=http://10.0.3.101:2380,e2=http://10.0.3.102:2380,e3=http://10.0.3.103:2380'

ETCD_ADVERTISE_CLIENT_URLS='http://10.0.3.103:2379'

ETCD_ENABLE_V2=true
```

1.3.3.3 Démarrage du service

- Réinitialisation des bases Etcd
- Démarrage du service `etcd`: `systemctl start etcd`

Avant de démarrer le service sur chaque nœud, il faut réinitialiser les répertoires de données des nœuds, afin qu'ils reparte sur un répertoire neuf.

Le nœud `e1`, que nous considérons comme premier *leader* sera démarré en premier mais il est possible qu'un autre nœud prenne le dessus s'il arrive à démarrer plus vite :

```
$ for node in e1 e2 e3; do
echo "$node : " ; sudo ssh $node "rm -rf ~etcd/default/*"
done

$ for node in e1 e2 e3; do
sudo ssh $node "systemctl start etcd" &
```

Haute disponibilité avec Patroni

```
sleep 1
done
```

En cas d'échec de démarrage, utilisez la commande *Systemd* pour en diagnostiquer la cause :

```
e1:~$ sudo journalctl -xfu etcd
```

Vérification :

```
$ for node in e1 e2 e3; do
echo "sur $node :"
sudo ssh $node "etcdctl member list"
done

sur e1 :
736293150f1cffb7: name=e1 peerURLs=http://10.0.3.101:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=true
7ef9d5bb55cefbcc: name=e3 peerURLs=http://10.0.3.103:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
97463691c7858a7b: name=e2 peerURLs=http://10.0.3.102:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
sur e2 :
736293150f1cffb7: name=e1 peerURLs=http://10.0.3.101:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=true
7ef9d5bb55cefbcc: name=e3 peerURLs=http://10.0.3.103:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
97463691c7858a7b: name=e2 peerURLs=http://10.0.3.102:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
sur e3 :
736293150f1cffb7: name=e1 peerURLs=http://10.0.3.101:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=true
7ef9d5bb55cefbcc: name=e3 peerURLs=http://10.0.3.103:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
97463691c7858a7b: name=e2 peerURLs=http://10.0.3.102:2380
clientURLs=http://10.0.3.101:2379,http://10.0.3.102:2379,http://10.0.3.103:2379 isLeader=false
```

Le *leader* doit être identique sur les trois nœuds, les trois nœuds doivent retourner la même liste de membres.

1.3.4 INSTALLATION DE POSTGRESQL / PATRONI

- Installation
 - PostgreSQL
 - Patroni
 - pgBackrest

Le dépôt *pgdg* est déjà préconfiguré dans les conteneurs pg-1, pg-2 et pg-3, l'installation est donc triviale :

```
$ for node in pg-1 pg-2 pg-3; do
sudo ssh $node "apt-get update && apt-get install -qqy postgresql patroni pgbackrest" &
done
```

Vérification :

```
$ for node in pg-1 pg-2 pg-3; do sudo ssh $node "dpkg -l postgresql patroni
pgbackrest | grep ^ii | cut -d ' ' -f 1,3"; done
ii patroni
ii pgbackrest
ii postgresql
ii patroni
ii pgbackrest
ii postgresql
ii patroni
ii pgbackrest
ii postgresql
```

Le service PostgreSQL doit être désactivé car la gestion totale de l'instance sera déléguée à Patroni :

```
$ for node in pg-1 pg-2 pg-3; do
sudo ssh $node "systemctl disable --now postgresql@15-main"
done
```

1.3.4.1 Configuration de Patroni

Sur tous les nœuds PostgreSQL/Patroni

- Configuration du DCS
 - `/etc/patroni/dcs.yml`
- Génération de la configuration
 - `pg_createconfig_patroni 15 main`

La configuration sous Debian se fait d'abord en renseignant comment contacter le DCS, puis en lançant le script de génération automatique de la configuration de Patroni.

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

Le port par défaut du service Etcd est le 2379.

```
$ sudo ssh pg-1
root@pg-1:~# vim /etc/patroni/dcs.yml

# /etc/patroni/dcs.yml
etcd:
  hosts:
    - 10.0.3.101:2379
    - 10.0.3.102:2379
    - 10.0.3.103:2379

root@pg-1:~# pg_createconfig_patroni 15 main"
```

La configuration `/etc/patroni/15-main.yml` est générée.

Ces opérations doivent être répétées sur tous les nœuds PostgreSQL/Patroni.

1.3.4.2 Création de l'agrégat

- Démarrage du primaire
- Création de l'utilisateur de réplication
- Suppression des instances secondaires
- Démarrage des instances secondaires

1.3.4.2.1 Démarrage du primaire

La création de l'agrégat commence par la mise en route du primaire sur le nœud `pg-1`, c'est lui qui sera la référence pour les secondaires.

```
root@pg-1:~# systemctl enable --now patroni@15-main
```

L'instance doit être promue pour pouvoir être accessible écriture :

```
root@pg-1:~# sudo -iu postgres psql -c 'select pg_promote();'
```

L'utilisateur permettant la mise en réplication doit être créé sur ce nœud, avec le mot de passe renseigné dans la configuration de Patroni :

```
root@pg-1:~# sudo -iu postgres psql -c "create user replicator replication password 'rep-pass';"
```

1.3.4.2.2 Superuser d'administration locale

Chaque nœud doit pouvoir récupérer la *timeline* et le *LSN* courants

1.3 Mise en place de l'infrastructure

```
root@pg-1:~# sudo -iu postgres psql -c "create user dba superuser password 'admin'"
```

Si l'utilisateur est différent de **postgres**, il faudra désactiver le socket unix sinon Patroni essaiera la connexion locale authentifiée par la méthode **peer**. L'utilisateur **dba** n'existant pas au niveau système, il y aurait échec.

La configuration de chaque nœud doit être modifiée :

```
#/etc/patroni/15-main.yaml
```

```
postgresql:
```

```
#...
```

```
use_unix_socket: false
```

```
#...
```

```
superuser:
```

```
  username: "dba"
```

```
  password: "admin"
```

```
#...
```

1.3.4.2.3 Suppression des instances secondaires

Les instances secondaires ont été initialisées lors de l'installation du paquet Debian, il faut donc vider leur répertoire de données car Patroni refusera d'écraser des données existantes. Nous utilisons le *wrapper* Debian :

pg-1 étant notre primaire :

```
$ for node in pg-2 pg-3; do sudo ssh $node "pg_dropcluster 15 main"; done
```

Les secondaires seront recréés automatiquement par Patroni, depuis le primaire par répliation.

Le primaire doit être redémarré :

```
postgres@pg-1:~ $ patronictl restart 15-main pg-1 --force
```

les nœuds secondaires doivent être démarrés :

```
$ for node in pg-2 pg-3; do sudo ssh $node "systemctl start patroni@15-main"; done
```

La vérification se fait dans les traces de Patroni des nœuds secondaires :

```
Feb 09 17:12:38 pg-3 patroni@15-main[1029]: 2023-02-09 17:12:38,984 INFO: Lock owner: pg-1; I am p
Feb 09 17:12:38 pg-3 patroni@15-main[1029]: 2023-02-09 17:12:38,986 INFO: Local timeline=7 lsn=0/5
Feb 09 17:12:39 pg-3 patroni@15-main[1029]: 2023-02-09 17:12:39,037 INFO: master_timeline=7
```

1.3.4.2.4 Vérifications

- Liste des nœuds Patroni
- Test de bascule manuelle vers chaque nœud

Liste des nœuds Patroni

Sur chaque nœud Patroni, modifier le `.profile` de l'utilisateur `postgres` en ajoutant :

```
export PATRONICTL_CONFIG_FILE=/etc/patroni/15-main.yml
```

```
$ sudo ssh pg-1 sudo -iu postgres patronictl list

+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201 | Leader | running | 3 |           |
| pg-2    | 10.0.3.202 | Replica | running | 3 |           0 |
| pg-3    | 10.0.3.203 | Replica | running | 3 |           0 |
```

Test de bascule manuelle vers chaque nœud

```
$ sudo ssh pg-1 sudo -iu postgres patronictl switchover
```

Master [pg-1]:

Candidate ['pg-2', 'pg-3'] []: pg-2

When should the switchover take place (e.g. 2021-11-12T12:21) [now]:

Current cluster topology

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201 | Leader | running | 3 |           |
| pg-2    | 10.0.3.202 | Replica | running | 3 |           0 |
| pg-3    | 10.0.3.203 | Replica | running | 3 |           0 |
+-----+-----+-----+-----+-----+-----+
```

Are you sure you want to switchover cluster 15-main, demoting current master

pg-1? [y/N]: y

2021-11-12 11:21:20.08091 Successfully switched over to "pg-2"

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201 | Replica | stopped |   | unknown |
| pg-2    | 10.0.3.202 | Leader | running | 3 |           |
| pg-3    | 10.0.3.203 | Replica | running | 3 |           0 |
+-----+-----+-----+-----+-----+-----+
```


1.4 CRÉATION D'INCIDENTS

- Perte totale du DCS
- Freeze du nœud primaire Patroni
- Bascule manuelle

1.4.1 PERTE TOTALE DU DCS

- Perte de tous les nœuds Etcd

Nous simulons un incident majeur au niveau du DCS :

```
$ for node in e1 e2 e3; do
  sudo lxc-freeze $node
done
```

La commande classique `patronictl list` échoue faute de DCS pour la renseigner.

Nous interrogeons directement sur les instances :

```
$ for node in pg-1 pg-2 pg-3; do
  echo "$node :"
  sudo ssh $node "sudo -iu postgres psql -c 'select pg_is_in_recovery()'"
done
```

```
pg-1 :
pg_is_in_recovery
-----
t
(1 ligne)
```

```
pg-2 :
pg_is_in_recovery
-----
t
(1 ligne)
```

```
pg-3 :
pg_is_in_recovery
-----
t
(1 ligne)
```

Nous constatons que l'intégralité des nœuds est passée en lecture seule (*stand-by*).

Nous débloquons la situation :

```
$ for node in e1 e2 e3; do
  echo "$node :"
```

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

```
sudo lxc-unfreeze $node
done
```

Nous pouvons observer le retour à la normale :

```
postgres@pg-1:~$ patronictl list -ew 1
```

1.4.2 PERTE DU NŒUD PRIMAIRE PATRONI

- Perte du primaire courant

Dans un autre terminal, nous observons l'état de l'agrégat sur le nœud **pg-2** :

```
postgres@pg-2:~$ patronictl list -ew 1
```

Nous simulons une perte du primaire **pg-1** :

```
$ sudo lxc-freeze pg-1
```

Nous observons la disparition de **pg-1** de la liste des nœuds et une bascule automatique se déclenche vers un des nœuds secondaires disponibles :

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-2   | 10.0.3.202 | Replica | running | 7 | 0 |
| pg-3   | 10.0.3.203 | Leader | running | 7 |  |
+-----+-----+-----+-----+-----+-----+
```

Nous rétablissons la situation :

```
$ sudo lxc-unfreeze pg-1
```

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Replica | running | 6 | 0 |
| pg-2   | 10.0.3.202 | Replica | running | 7 | 0 |
| pg-3   | 10.0.3.203 | Leader | running | 7 |  |
+-----+-----+-----+-----+-----+-----+
```

Pour un retour à l'état nominal, il suffit de procéder à une bascule manuelle (adapter la commande si votre primaire n'est pas **pg-3**) :

```
postgres@pg-1:~$ patronictl switchover --master pg-3 --candidate pg-1 --force
```

Current cluster topology

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Replica | running | 7 | 0 |
```

```

| pg-2 | 10.0.3.202 | Replica | running | 7 | 0 |
| pg-3 | 10.0.3.203 | Leader | running | 7 |  |
+-----+-----+-----+-----+-----+
2021-11-12 13:18:36.05884 Successfully switched over to "pg-1"
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+
| pg-1 | 10.0.3.201 | Leader | running | 7 |  |
| pg-2 | 10.0.3.202 | Replica | running | 7 | 0 |
| pg-3 | 10.0.3.203 | Replica | stopped |  | unknown |
+-----+-----+-----+-----+-----+

```

1.5 MODIFICATION DE LA CONFIGURATION

- `patronictl edit-config`

L'un des avantages de bénéficier d'une configuration distribuée est qu'il est possible de modifier cette configuration pour tous les nœuds en une seule opération.

Si le paramètre nécessite un rechargement de la configuration, elle sera lancée sur chaque nœud.

Si la modification nécessite un redémarrage, l'indicateur *pending restart* sera positionné sur toutes les instances et attendrons une action de votre part pour l'effectuer.

L'installation de la commande `less` est un pré-requis :

```

$ for node in pg-1 pg-2 pg-3; do
  ssh $node apt install less
done

```

La modification peut se faire sur n'importe quel nœud :

```
postgres@pg-2:~$ patronictl edit-config
```

Nous ajoutons une ligne fille de la ligne `parameters:`

```

loop_wait: 10
maximum_lag_on_failover: 1048576
postgresql:
  parameters:
    max_connections: 123
...

```

Une confirmation est demandée après la sortie de l'éditeur :

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

```
patronictl edit-config
---
+++
@@ -1,7 +1,8 @@
loop_wait: 10
maximum_lag_on_failover: 1048576
postgresql:
- parameters: null
+ parameters:
+   max_connections: 123
pg_hba:
- local      all                all                                peer
- host       all                all                                127.0.0.1/32      md5
```

Apply these changes? [y/N]: y
Configuration changed

Après modification, il convient de regarder si notre modification ne nécessite pas de redémarrage :

```
postgres@pg-2:~$ patronictl list -e
```

Member	Host	Role	State	TL	Lag in MB	Pending restart
pg-1	10.0.3.201	Leader	running	8		*
pg-2	10.0.3.202	Replica	running	8	0	*
pg-3	10.0.3.203	Replica	running	8	0	*

Dans notre cas, un redémarrage de toutes les instances est nécessaire :

```
postgres@pg-2:~$ patronictl restart 15-main
```

Member	Host	Role	State	TL	Lag in MB	Pending restart
pg-1	10.0.3.201	Leader	running	8		*
pg-2	10.0.3.202	Replica	running	8	0	*
pg-3	10.0.3.203	Replica	running	8	0	*

When should the restart take place (e.g. 2021-11-12T14:37) [now]:
Are you sure you want to restart members pg-3, pg-2, pg-1? [y/N]: y
Restart if the PostgreSQL version is less than provided (e.g. 9.5.2) []:
Success: restart on member pg-3
Success: restart on member pg-2
Success: restart on member pg-1

```
postgres@pg-2:~$ patronictl list -e
```

```
+ Cluster: 15-main (7029596050496494965) -+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB | Pending restart |
+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Leader | running | 8 |          |                |
| pg-2   | 10.0.3.202 | Replica | running | 8 |          |                |
| pg-3   | 10.0.3.203 | Replica | running | 8 |          |                |
+-----+-----+-----+-----+-----+-----+

```

```
$ for node in pg-1 pg-2 pg-3; do
  echo "$node : "
  sudo ssh $node "sudo -iu postgres psql -c 'show max_connections'"
done

```

```
pg-1 :
max_connections
-----
123
(1 ligne)

```

```
pg-2 :
max_connections
-----
123
(1 ligne)

```

```
pg-3 :
max_connections
-----
123
(1 ligne)

```

L'application d'un paramètre qui ne nécessite pas de redémarrage est transparente, le rechargement de la configuration sur tous les nœuds est automatiquement déclenchée par Patroni.

1.6 SAUVEGARDES

- Installation pgBackrest
- Configuration
- Détermination du primaire
- Archivage
- Sauvegarde

Haute disponibilité avec Patroni

1.6.1 DÉTERMINATION DU PRIMAIRE

Nous proposons de déclencher la sauvegarde sur le primaire courant, il faut donc d'abord l'identifier.

Le script suivant est une solution permettant de récupérer le primaire de notre agrégat à partir d'un nœud Etcd et de l'API mise à disposition :

```
#!/bin/bash
SCOPE=$(grep -i scope: /etc/patroni/15-main.yml | cut -d '"' -f 2)
curl -s http://e1:2379/v2/keys/postgresql-common/"$SCOPE"/leader | jq -r .node.value
```

1.6.2 CONFIGURATION DE PGBACKREST

Sur chacun des nœuds, il faut configurer le *stanza* et l'initialiser :

```
# /etc/pgbackrest.conf
[main]
pg1-path=/var/lib/postgresql/15/main
pg1-socket-path=/var/run/postgresql
pg1-port=5432

[global]
log-level-file=detail
log-level-console=detail
repo1-host=backup
repo1-host-user=postgres
```

Tous les nœuds doivent permettre la connexion *ssh* sans mot de passe, le *playbook Ansible* nommé *exchange_ssh_keys* permet de faire ce travail rapidement :

```
$ sudo ansible-playbook -i inventory.yml exchange_ssh_keys.yml -f 7
```

La première connexion *ssh* entre *backup* et les nœuds PostgreSQL demande une confirmation. Il faut donc lancer les trois commandes :

```
postgres@backup:~ $ ssh pg-1
```

```
postgres@backup:~ $ ssh pg-2
```

```
postgres@backup:~ $ ssh pg-3
```

Nous pouvons alors tenter de créer le *stanza* sur le primaire :

```
postgres@pg-1:~$ pgbackrest --stanza main stanza-create
postgres@pg-1:/var/lib/pgbackrest$ pgbackrest --stanza main check
ERROR: [087]: archive_mode must be enabled
```

L'archivage est en erreur puisque non configuré.

1.6.2.1 Configuration de l'archivage

Toutes les instances doivent être en mesure d'archiver leurs journaux de transactions au moyen de pgBackrest :

```
postgres@pg-1:~$ patronictl edit-config
postgresql:
  parameters:
    max_connections: 123
    archive_mode: 'on'
    archive_command: pgbackrest --stanza=main archive-push %p
```

Notre configuration n'a pas encore été appliquée sur les instances car un redémarrage est requis :

```
postgres@pg-1:~$ patronictl list -e
+ Cluster: 15-main (7029596050496494965) +-----+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB | Pending restart |
+-----+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Leader | running | 8 |          | *               |
| pg-2   | 10.0.3.202 | Replica | running | 8 |          | *               |
| pg-3   | 10.0.3.203 | Replica | running | 8 |          | *               |
+-----+-----+-----+-----+-----+-----+

postgres@pg-1:~$ patronictl restart 15-main --force
+ Cluster: 15-main (7029596050496494965) +-----+-----+-----+
| Member | Host      | Role   | State | TL | Lag in MB | Pending restart |
+-----+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Leader | running | 8 |          | *               |
| pg-2   | 10.0.3.202 | Replica | running | 8 |          | *               |
| pg-3   | 10.0.3.203 | Replica | running | 8 |          | *               |
+-----+-----+-----+-----+-----+-----+

Success: restart on member pg-1
Success: restart on member pg-3
Success: restart on member pg-2
```

Test de la configuration de l'archivage sur le nœud **pg-1** :

```
postgres@pg-1:~$ pgbackrest --stanza main check

2021-11-12 15:57:04.000 P00 INFO: check command begin 2.35: --exec-id=13216-4a7c4a92 --log-level-console=detail --log-level-file=detail --pg1-path=/var/lib/postgresql/15/main --pg1-port=5432 --pg1-socket-path=/var/run/postgresql --repo1-host=backup --repo1-host-user=postgres --stanza=main
2021-11-12 15:57:04.616 P00 INFO: check repo1 configuration (primary)
2021-11-12 15:57:05.083 P00 INFO: check repo1 archive for WAL (primary)
2021-11-12 15:57:08.425 P00 INFO: WAL segment 00000008000000000000000000000000
successfully archived to '/var/lib/pgbackrest/archive/main/15-1/
http://dalibo.com/formations
```

Haute disponibilité avec Patroni

```
0000000800000000/000000080000000000000005-  
b0929d740c7996974992ecd7b9b189b37d06a896.gz' on repo1  
2021-11-12 15:57:08.528 P00 INFO: check command end: completed successfully  
(4531ms)
```

1.6.2.2 Configuration sur la machine hébergeant les sauvegardes

Sur la machine **backup**, créer le script de détermination du **leader** (le rendre exécutable) :

```
postgres@backup:~$ vim ~/leader.sh && chmod +x leader.sh  
#!/bin/bash  
SCOPE='15-main'  
curl -s http://e1:2379/v2/keys/postgresql-common/"$SCOPE"/leader | jq -r .node.value
```

1.6.2.2.1 Configuration de pgBackrest

Nous avons choisi d'opérer en mode *pull*, les sauvegardes seront exécutées sur la machine **backup** et récupérées depuis le primaire courant.

La configuration se fait dans le fichier **/etc/pgbackrest.conf** :

```
[global]  
repo1-path=/var/lib/pgbackrest  
repo1-retention-full=2  
start-fast=y  
log-level-console=info  
log-level-file=info  
  
[main]  
pg1-path=/var/lib/postgresql/15/main  
pg1-host-user=postgres  
pg1-user=postgres  
pg1-port=5432
```

On déterminera l'instance qui sera utilisée pour récupérer la sauvegarde, au moment de la sauvegarde.

Test d'une sauvegarde

```
postgres@backup:~$ pgbackrest --stanza main --pg1-host=$(./leader.sh) backup  
--type=full
```

```
2023-02-09 18:43:38.424 P00 INFO: backup command begin 2.44: --exec-id=1116-00f26290  
--log-level-console=info --log-level-file=info --pg1-host=pg-1 --pg1-host-user=postgres --  
pg1-path=/var/lib/postgresql/15/main --pg1-port=5432 --pg1-user=postgres --repo1-path=/var/  
24
```


1.6 Sauvegardes

```
lib/pgbackrest --repo1-retention-full=2 --stanza=main --start-fast --type=full
2023-02-09 18:43:39.363 P00 INFO: execute non-exclusive backup start: backup begins
after the requested immediate checkpoint completes
2023-02-09 18:43:40.475 P00 INFO: backup start archive = 0000000700000000000000013, lsn = 0/13000
2023-02-09 18:43:40.475 P00 INFO: check archive for prior segment 0000000700000000000000012
2023-02-09 18:44:06.600 P00 INFO: execute non-exclusive backup stop and wait for all WAL segment
2023-02-09 18:44:06.905 P00 INFO: backup stop archive = 0000000700000000000000013, lsn = 0/130001
2023-02-09 18:44:06.955 P00 INFO: check archive for segment(s)
0000000700000000000000013:0000000700000000000000013
2023-02-09 18:44:07.533 P00 INFO: new backup label = 20230209-184339F
2023-02-09 18:44:07.798 P00 INFO: full backup size = 29.2MB, file total = 1261
2023-02-09 18:44:07.819 P00 INFO: backup command end: completed successfully (29396ms)
2023-02-09 18:44:07.819 P00 INFO: expire command begin 2.44: --exec-id=1116-00f26290
--log-level-console=info --log-level-file=info --repo1-path=/var/lib/pgbackrest
--repo1-retention-full=2 --stanza=main
2023-02-09 18:44:07.822 P00 INFO: repo1: expire full backup 20230209-183850F
2023-02-09 18:44:07.892 P00 INFO: repo1: remove expired backup 20230209-183850F
2023-02-09 18:44:07.971 P00 INFO: repo1: 15-1 remove archive, start = 0000000700000000000000008,
stop = 000000070000000000000000A
2023-02-09 18:44:07.971 P00 INFO: expire command end: completed successfully (152ms)
```

Vérification de l'état de la sauvegarde :

```
postgres@backup:~$ pgbackrest --stanza main info

stanza: main
  status: ok
  cipher: none

db (current)
  wal archive min/max (14): 000000010000000000000001/00000008000000000000000B

  full backup: 20211112-163233F
    timestamp start/stop: 2021-11-12 16:32:33 / 2021-11-12 16:32:45
    wal start/stop: 00000008000000000000000B / 00000008000000000000000B
    database size: 25.1MB, database backup size: 25.1MB
    repo1: backup set size: 3.2MB, backup size: 3.2MB
```

1.7 RÉPLICATION SYNCHRONE

La réplication synchrone permet de garantir que les données sont écrites sur un ou plusieurs secondaires lors de la validation des transactions.

Elle permet de réduire quasi-totalement la perte de donnée lors d'un incident (RPO).

Il faut un minimum de trois paramètres ajoutés à la configuration dynamique pour décrire la réplication synchrone :

```
synchronous_mode: true
synchronous_node_count: 1
synchronous_standby_names: '*'
```

Après quelques secondes, l'un des réplicas passe en mode synchrone :

```
postgres@pg-1:~$ patronictl list
+ Cluster: 15-main (7198182122558146054) -----+-----+-----+
| Member | Host          | Role          | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201    | Leader        | running | 7 |           |
| pg-2    | 10.0.3.202    | Replica        | running | 7 |           |
| pg-3    | 10.0.3.203    | Sync Standby  | running | 7 |           |
+-----+-----+-----+-----+-----+-----+
```

- La prochaine bascule ne sera possible que sur le nœud synchrone.
- Si le nœud synchrone est défaillant, un des secondaires restant passera en mode synchrone (**synchronous_standby_names** et **synchronous_node_count** l'autorisent)

Perte du secondaire synchrone :

```
$ sudo lxc-freeze pg-3
```

Quelques secondes plus tard :

```
+ Cluster: 15-main (7198182122558146054) -----+-----+-----+
| Member | Host          | Role          | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201    | Leader        | running | 7 |           |
| pg-2    | 10.0.3.202    | Sync Standby  | running | 7 |           |
+-----+-----+-----+-----+-----+-----+
```

1.8 MISE À JOUR MINEURE SANS INTERRUPTION DE SERVICE

Rappel : la réplication physique peut être mise en œuvre entre deux instances de versions mineures différentes. (ex: 15.1 vers 15.2)

La mise à jour mineure peut être faite nœud par nœud en commençant par les secondaires asynchrones, puis par les secondaires synchrones.

Dès qu'un deuxième secondaire synchrone est présent, la mise peut être faite sur le premier secondaire synchrone.

Une fois tous les secondaires mis à jour, une bascule sur un des secondaires synchrone à jour pourra être faite et l'ancien primaire sera alors mis à jour de la même manière, puis redémarré :

État de départ :

```
+ Cluster: 15-main (7198182122558146054) -----+-----+
| Member | Host      | Role      | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Leader    | running | 7 |          |
| pg-2   | 10.0.3.202 | Replica    | running | 7 |          0 |
| pg-3   | 10.0.3.203 | Sync Standby | running | 7 |          0 |
+-----+-----+-----+-----+-----+-----+-----+
```

Mise à jour jour et redémarrage du premier secondaire asynchrone **pg-2 :**

```
...
$ patronictl restart 15-main pg-2
```

Passage à 2 nœuds synchrones :

```
$ patronictl edit-config
...
synchronous_node_count: 2
...

+ Cluster: 15-main (7198182122558146054) -----+-----+
| Member | Host      | Role      | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1   | 10.0.3.201 | Leader    | running | 10 |          |
| pg-2   | 10.0.3.202 | Sync Standby | running | 10 |          0 |
| pg-3   | 10.0.3.203 | Sync Standby | running | 10 |          0 |
+-----+-----+-----+-----+-----+-----+-----+
```

Mise à jour du nœud synchrone **pg-3 :**

<http://dalibo.com/formations>

Haute disponibilité avec Patroni

...

```
$ patronictl restart 15-main pg-3
```

Bascule vers le secondaire synchrone mis à jour :

...

```
$ patronictl switchover --master pg-1 --candidate pg-3 --force
```

Mise à jour du primaire :

...

```
$ patronictl restart 15-main pg-1 --force
```

Effectuer la promotion et remettre le nombre de nœuds synchrone à 1 :

```
$ patronictl switchover --candidate pg-1 --master pg-3 --force
```

```
+ Cluster: 15-main (7198182122558146054) -----+-----+-----+
| Member | Host      | Role      | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201 | Leader    | running | 11 |           |
| pg-2    | 10.0.3.202 | Sync Standby | running | 11 |           0 |
| pg-3    | 10.0.3.203 | Sync Standby | running | 11 |           0 |
+-----+-----+-----+-----+-----+-----+
```

```
$ patronictl edit-config
```

...

```
synchronous_node_count: 1
```

...

Après quelques secondes :

```
+ Cluster: 15-main (7198182122558146054) -----+-----+-----+
| Member | Host      | Role      | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg-1    | 10.0.3.201 | Leader    | running | 11 |           |
| pg-2    | 10.0.3.202 | Sync Standby | running | 11 |           0 |
| pg-3    | 10.0.3.203 | Replica    | running | 11 |           0 |
+-----+-----+-----+-----+-----+-----+
```

Aucun arrêt de service et aucune perte de données due à l'opération.

1.9 RÉFÉRENCES

- Etcd : <https://etcd.io/docs/>
 - Patroni : <https://patroni.readthedocs.io/en/latest/>
 - Formation HAPAT : <https://dalibo.com/formation-postgresql-haute-disponibilite>
 - Dalibo : <https://dalibo.com>
-

NOTES

NOTES

NOTES

NOTES

NOS AUTRES PUBLICATIONS

FORMATIONS

- **DBA1 : Administration PostgreSQL**
<https://dali.bo/dba1>
- **DBA2 : Administration PostgreSQL avancé**
<https://dali.bo/dba2>
- **DBA3 : Sauvegarde et réplication avec PostgreSQL**
<https://dali.bo/dba3>
- **DEVPG : Développer avec PostgreSQL**
<https://dali.bo/devpg>
- **PERF1 : PostgreSQL Performances**
<https://dali.bo/perf1>
- **PERF2 : Indexation et SQL avancés**
<https://dali.bo/perf2>
- **MIGORPG : Migrer d'Oracle à PostgreSQL**
<https://dali.bo/migorpg>
- **HAPAT : Haute disponibilité avec PostgreSQL**
<https://dali.bo/hapat>

LIVRES BLANCS

- Migrer d'Oracle à PostgreSQL
- Industrialiser PostgreSQL
- Bonnes pratiques de modélisation avec PostgreSQL
- Bonnes pratiques de développement avec PostgreSQL

TÉLÉCHARGEMENT GRATUIT

Les versions électroniques de nos publications sont disponibles gratuitement sous licence open-source ou sous licence Creative Commons. Contactez-nous à l'adresse contact@dalibo.com pour plus d'information.

DALIBO, L'EXPERTISE POSTGRESQL

Depuis 2005, DALIBO met à la disposition de ses clients son savoir-faire dans le domaine des bases de données et propose des services de conseil, de formation et de support aux entreprises et aux institutionnels.

En parallèle de son activité commerciale, DALIBO contribue aux développements de la communauté PostgreSQL et participe activement à l'animation de la communauté francophone de PostgreSQL. La société est également à l'origine de nombreux outils libres de supervision, de migration, de sauvegarde et d'optimisation.

Le succès de PostgreSQL démontre que la transparence, l'ouverture et l'auto-gestion sont à la fois une source d'innovation et un gage de pérennité. DALIBO a intégré ces principes dans son ADN en optant pour le statut de SCOP : la société est contrôlée à 100 % par ses salariés, les décisions sont prises collectivement et les bénéfices sont partagés à parts égales.