Atelier

Haute disponibilité avec Patroni & Etcd

DALIBO
L'expertise PostgreSQL

1.6

Franck BOUDEHEN

www.dalibo.com

Haute disponibilité avec Patroni & Etcd

Atelier

TITRE : Haute disponibilité avec Patroni & Etcd
SOUS-TITRE : Atelier

REVISION: 1.6
DATE: 17 novembre 2020

Table des Matières

ATELIER PATRONI ETCD POSTGRESQL

SCHÉMA

Cluster Etcd	Cluster Patroni
etcd1: 10.0.3.64	pg-patroni1: 10.0.3.141
etcd2: 10.0.3.78	pg-patroni2: 10.0.3.201
etcd3: 10.0.3.32	(pg-patroni3: 10.0.3.68)

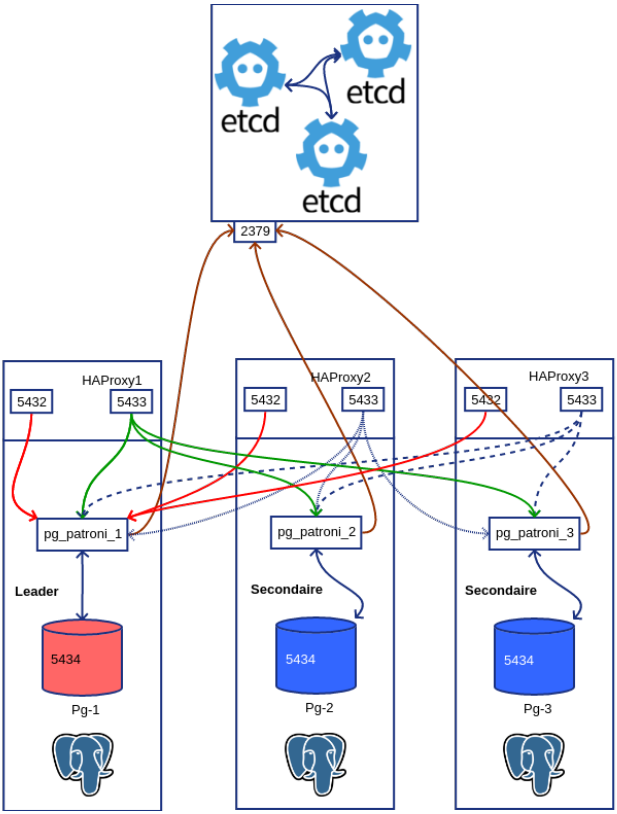

Figure 1: Schéma complet

PRÉ-REQUIS

- LXC

CONTAINERS LXC

création des 3 containers :

```
sudo lxc-create -n etcd1 -t debian -- -r stretch
sudo lxc-create -n etcd2 -t debian -- -r stretch
sudo lxc-create -n etcd3 -t debian -- -r stretch
```

ETCD

```
sudo lxc-start -n etcd1
sudo lxc-attach -n etcd1
```

SUR ETCD1, ETCD2 ET ETCD3

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y curl wget sudo vim iputils-ping
```

INSTALLATION ETCD

INSTALLATION DEPUIS L'ARCHIVE CHEZ GITHUB

```
$ sudo -s
...
curl -s https://api.github.com/repos/etcd-io/etcd/releases/latest \
  | grep browser_download_url \
  | grep linux-amd64 \
  | cut -d '"' -f 4 \
  | wget -qi -


gzip -d etcd-v3.4.1-linux-amd64.tar.gz
tar xf etcd-v3.4.1-linux-amd64.tar
cd etcd-v3.4.1-linux-amd64

$ sudo mv etcd* /usr/local/bin/

$ sudo useradd -m -d /var/lib/etcd etcd
```

INSTALLATION DEPUIS LES PAQUETS DEBIAN

```
$ sudo apt install etcd
```

Configuration etcd par fichier de configuration

Configuration du premier nœud etcd, changer le nom et l'adresse ip pour les 2 suivants :

```
# /etc/etcd/etcd.conf.yml

# Human-readable name for this member.
name: 'etcd1'

# Path to the data directory.
data-dir: /var/lib/etcd/my-cluster.etcd

# List of comma separated URLs to listen on for peer traffic.
listen-peer-urls: http://10.0.3.64:2380

# List of comma separated URLs to listen on for client traffic.
# remplacer par l'ip de etcd2
listen-client-urls: http://localhost:2379,http://10.0.3.64:2379

# The URLs needed to be a comma-separated list.
# remplacer par l'ip de etcd2
initial-advertise-peer-urls: http://10.0.3.64:2380

# List of this member's client URLs to advertise to the public.
# The URLs needed to be a comma-separated list.
# remplacer par l'ip de etcd2

advertise-client-urls: http://10.0.3.64:2379

# Initial cluster configuration for bootstrapping.
initial-cluster: etcd1=http://10.0.3.64:2380,etcd2=http://10.0.3.78:2380

# Initial cluster token for the etcd cluster during bootstrap.
initial-cluster-token: 'patroni'

# Initial cluster state ('new' or 'existing').
initial-cluster-state: 'new'
```

Configuration dans le service

```
INT_NAME="eth0"
ETCD_HOST_IP=$(ETCD_HOST_IP=$(ip addr sho eth0 | grep 'inet\b' | cut -d/ -f 1 | awk '{print $2}')
ETCD_NAME=$(hostname -s)
```

etcd1

```
[Unit]
Description=etcd - highly-available key value store
Documentation=https://github.com/coreos/etcd
Documentation=man:etcd
After=network.target
Wants=network-online.target

[Service]
Environment=DAEMON_ARGS=
Environment=ETCD_NAME=%H
Environment=ETCD_DATA_DIR=/var/lib/etcd/my-cluster
Environment=INT_NAME="eth0"
Environment=ETCD_HOST_IP=10.0.3.64
Environment=ETCD_NAME=etcd1
EnvironmentFile=-/etc/default/%p
Type=notify
User=etcd
PermissionsStartOnly=true


ExecStart=/usr/bin/etcd $DAEMON_ARGS \
 --name ${ETCD_NAME} \
 --data-dir ${ETCD_DATA_DIR} \
 --listen-peer-urls http://${ETCD_HOST_IP}:2380 \
 --listen-client-urls http://${ETCD_HOST_IP}:2379,http://127.0.0.1:2379 \
 --advertise-client-urls http://${ETCD_HOST_IP}:2379 \
 --initial-cluster-token etcd-cluster-0 \
 --initial-cluster etcd1=http://etcd1:2380,etcd2=http://etcd2:2380,etcd3=http://etcd3:2380 \
 --initial-cluster-state new \
 --heartbeat-interval 1000 \
 --election-timeout 5000

Restart=on-failure
RestartSec=5

Restart=on-abnormal
#RestartSec=10s
LimitNOFILE=65536

[Install]
```

```
WantedBy=multi-user.target
Alias=etcd2.service
```

etcd2

```
[Unit]
Description=etcd - highly-available key value store
Documentation=https://github.com/coreos/etcd
Documentation=man:etcd
After=network.target
Wants=network-online.target

[Service]
Environment=DAEMON_ARGS=
Environment=ETCD_NAME=%H
Environment=ETCD_DATA_DIR=/var/lib/etcd/my-cluster
Environment=INT_NAME="eth0"
Environment=ETCD_HOST_IP=10.0.3.78
Environment=ETCD_NAME=etcd2
EnvironmentFile=-/etc/default/%p
Type=notify
User=etcd
PermissionsStartOnly=true

ExecStart=/usr/bin/etcd $DAEMON_ARGS \
  --name ${ETCD_NAME} \
  --data-dir ${ETCD_DATA_DIR} \
  --initial-advertise-peer-urls http://${ETCD_HOST_IP}:2380 \
  --listen-peer-urls http://${ETCD_HOST_IP}:2380 \
  --listen-client-urls http://${ETCD_HOST_IP}:2379,http://127.0.0.1:2379 \
  --advertise-client-urls http://${ETCD_HOST_IP}:2379 \
  --initial-cluster-token etcd-cluster-0 \
  --initial-cluster etcd1=http://etcd1:2380,etcd2=http://etcd2:2380,etcd3=http://etcd3:2380 \
  --initial-cluster-state new \
  --heartbeat-interval 1000 \
  --election-timeout 5000

Restart=on-abnormal
#RestartSec=10s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
Alias=etcd2.service
```

> La présence de 2 nœuds Etcd n'est pas suffisante pour la haute disponibilité du service. Etcd s'arrête si le quorum n'est pas respecté (nombre de nœuds/2 +1 disponibles).

etcd3

```
[Unit]
Description=etcd - highly-available key value store
Documentation=https://github.com/coreos/etcd
Documentation=man:etcd
After=network.target
Wants=network-online.target

[Service]
Environment=DAEMON_ARGS=
Environment=ETCD_NAME=%H
Environment=ETCD_DATA_DIR=/var/lib/etcd/my-cluster
Environment=INT_NAME="eth0"
Environment=ETCD_HOST_IP=10.0.3.32
Environment=ETCD_NAME=etcd3
EnvironmentFile=-/etc/default/%p
Type=notify
User=etcd
PermissionsStartOnly=true
#ExecStart=/bin/sh -c "GOMAXPROCS=$(nproc) /usr/bin/etcd $DAEMON_ARGS"
ExecStart=/usr/bin/etcd $DAEMON_ARGS \
  --name ${ETCD_NAME} \
  --data-dir=${ETCD_DATA_DIR} \
  --initial-advertise-peer-urls http://${ETCD_HOST_IP}:2380 \
  --listen-peer-urls http://${ETCD_HOST_IP}:2380 \
  --listen-client-urls http://${ETCD_HOST_IP}:2379,http://127.0.0.1:2379 \
  --advertise-client-urls http://${ETCD_HOST_IP}:2379 \
  --initial-cluster-token etcd-cluster-0 \
  --initial-cluster etcd1=http://etcd1:2380,etcd2=http://etcd2:2380,etcd3=http://etcd3:2380 \
  --initial-cluster-state new \
  --heartbeat-interval 1000 \
  --election-timeout 5000


Restart=on-abnormal
#RestartSec=10s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

```
Alias=etcd2.service
```

Lancement du daemon avec le fichier de configuration

```
$ sudo -iu etcd
$ etcd --config-file /etc/etcd/etcd.conf.yml
```

Lancement du daemon avec systemd

```
# arrêt de l'instance par défaut
$ sudo systemctl stop etcd
# rechargement de la configuration du service
$ sudo systemctl daemon-reload
# activation et lancemeznt
$ sudo systemctl enable etcd
$ sudo systemctl start etcd
$ sudo systemctl status etcd
```

Vérification des services

Par l'*API*

```
$ for i in  64 78 32; do curl http://10.0.3.$i:2379/v2/machines; echo;done
http://10.0.3.64:2379, http://10.0.3.78:2379, http://10.0.3.32:2379
http://10.0.3.64:2379, http://10.0.3.78:2379, http://10.0.3.32:2379
http://10.0.3.64:2379, http://10.0.3.78:2379, http://10.0.3.32:2379
```

Chacun des 3 nœuds doit retourner la liste de tous les nœuds.

Vérification par scan de ports

```
$ nmap -P0 --open -p 2379,2380 10.0.3.64,78,32
```

Réinitialisation de la base d'Etcd

On peut avoir besoin de recréer totalement le cluster et donc de réinitialiser la configuration distribuée contenue dans le cluster Etcd :

```
$ sudo -iu etcd etcdctl rm /service/my-cluster initialize
```

PATRONI

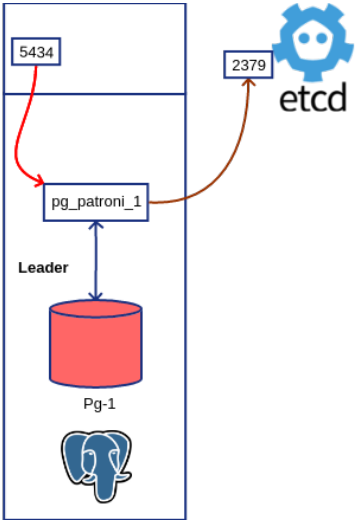

Figure 2: Schéma d'un module Patroni

Création des containers PostgreSQL/Patroni :

```
$ sudo lxc-create -n pg-patroni1 -t debian -- -r stretch
$ sudo lxc-create -n pg-patroni2 -t debian -- -r stretch
```

SUR PG-PATRONI1

```
$ sudo  apt install vim sudo curl wget gpg
$ sudo  vim /etc/apt/sources.list.d/pgpdg.list
```

ajout du dépôt pgdg

Remplacer *stretch* par *buster* si vous êtes en debian 10

```
deb http://apt.postgresql.org/pub/repos/apt/ stretch-pgdg main
```

Import de la clef gpg :

```
$ curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

Installation de Postgresql :

```
$ sudo apt update
$ sudo apt install postgresql-13
$ sudo pg_ctlcluster 13 main stop
```

```
$ sudo pg_dropcluster 13 main
$ sudo systemctl disable postgresql-13
```

Installation de Patroni :

```
$ sudo apt install patroni
$ sudo cp /etc/patroni/config.yml.in /etc/patroni/config.yml
$ sudo chown postgres:postgres /etc/patroni/config.yml
```

Installation de Patroni sans paquet pour info :

```
$ sudo apt install python3-pip python3-psycopg2
$ sudo pip3 install python-etcd
$ sudo pip3 install patroni
$ sudo patroni
Usage: /usr/local/bin/patroni config.yml
    Patroni may also read the configuration from the PATRONI_CONFIGURATION environment variable
```

Config de pg-patroni1

Création du fichier /etc/patroni/config.yml

```
scope: my-ha-cluster
name: pg-1

restapi:
  listen: 0.0.0.0:8008
  connect_address: 127.0.0.1:8008


etcd:
  host: 10.0.3.64:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        hot_standby: "on"
        wal_keep_segment: 8
        max_wal_senders: 5
        max_relication_slots: 5
```

```
        checkpoint_timeout: 30min

  initdb: UTF8

  pg_hba:
  - host all dba all md5
  - host replication repl all md5

  users:
    dba:
      password: secret
      options:
       - createrole
       - createdb
    repl:
      password: secret
      options:
        - replication:

postgresql:
  listen: 10.0.3.141:5434
  connect_address: 10.0.3.141:5434
  data_dir: /var/lib/postgresql/data
  bin_dir: /usr/lib/postgresql/12/bin
  authentication:
    replication:
      username: repl
      password: secret
    superuser:
      username: dba
      password: secret
  parameters:
    unix_socket_directories: '/tmp'
```

Comme notre cluster Etcd a plusieurs nœuds, nous pouvons les communiquer à Patroni, afin qu'il tente la connexion sur chacun des 3 nœuds en cas d'échec :

dans la section etcd :

```
etcd:
  hosts: 10.0.3.64:2379, 10.0.3.78:2379, 10.0.3.32:2379
```

> Notez que la sous-section `hosts` est passée au pluriel.

lancement de patroni

```
$ sudo -iu postgresql
$ patroni /etc/patroni/config.yml

2019-10-09 15:08:24,783 INFO: Failed to import patroni.dcs.consul
2019-10-09 15:08:24,793 INFO: Selected new etcd server http://10.0.3.64:2379
2019-10-09 15:08:24,800 INFO: No PostgreSQL configuration items changed, nothing to re
2019-10-09 15:08:24,809 WARNING: Postgresql is not running.
2019-10-09 15:08:24,810 INFO: Lock owner: None; I am pg-1
2019-10-09 15:08:24,812 INFO: pg_controldata:
  Backup start location: 0/0
  Latest checkpoint's oldestCommitTsXid: 0
  Database cluster state: shut down
  Size of a large-object chunk: 2048
  pg_control version number: 1201
  Bytes per WAL segment: 16777216
  max_wal_senders setting: 10
  Float8 argument passing: by value
  Latest checkpoint's REDO WAL file: 0000000A0000000000000003
  Maximum size of a TOAST chunk: 1996
  Latest checkpoint's NextXID: 0:492
  Latest checkpoint's NextOID: 16393
  wal_level setting: replica
  track_commit_timestamp setting: off
  Mock authentication nonce: 8acd62300c302ccd7e96b515e1de24171c625933d7c483e3b056e0ed9
  Blocks per segment of large relation: 131072
  Database system identifier: 6745811559071353594
  Latest checkpoint's TimeLineID: 10
  pg_control last modified: Wed Oct  9 15:07:16 2019
  Latest checkpoint's REDO location: 0/301DFA0
  max_connections setting: 100
  Time of latest checkpoint: Wed Oct  9 15:07:16 2019
  Database block size: 8192
  max_worker_processes setting: 8
  Backup end location: 0/0
  Latest checkpoint's newestCommitTsXid: 0
  Latest checkpoint's NextMultiXactId: 1
  Latest checkpoint's oldestXID: 480
  Maximum data alignment: 8
  WAL block size: 8192
  Latest checkpoint's oldestMulti's DB: 1
```

```
  Maximum length of identifiers: 64
  Latest checkpoint's oldestXID's DB: 1
  Latest checkpoint location: 0/301DFA0
  Data page checksum version: 0
  End-of-backup record required: no
  Latest checkpoint's full_page_writes: on
  Float4 argument passing: by value
  Min recovery ending loc's timeline: 0
  Latest checkpoint's oldestMultiXid: 1
  max_locks_per_xact setting: 64
  wal_log_hints setting: on
  Date/time type storage: 64-bit integers
  Latest checkpoint's oldestActiveXID: 0
  Latest checkpoint's PrevTimeLineID: 10
  Maximum columns in an index: 32
  Fake LSN counter for unlogged rels: 0/1
  max_prepared_xacts setting: 0
  Latest checkpoint's NextMultiOffset: 0
  Minimum recovery ending location: 0/0
  Catalog version number: 201909212

2019-10-09 15:08:24,814 INFO: Lock owner: None; I am pg-1
2019-10-09 15:08:24,822 INFO: Lock owner: None; I am pg-1
2019-10-09 15:08:24,827 INFO: starting as a secondary
2019-10-09 15:08:24,843 INFO: postmaster pid=11557
10.0.3.141:5432 - no response
2019-10-09 15:08:24.858 UTC [11557] LOG:  starting PostgreSQL 12.0 (Debian 12.0-
  1.pgdg90+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 2
2019-10-09 15:08:24.859 UTC [11557] LOG:  en écoute sur IPv4, adresse «
10.0.3.141 », port 5434
2019-10-09 15:08:24.863 UTC [11557] LOG:  écoute sur la socket Unix « /tmp/
.s.PGSQL.5432 »
2019-10-09 15:08:24.881 UTC [11559] LOG:  le système de bases de données a été
arrêté à 2019-10-09 15:07:16 UTC
2019-10-09 15:08:24.881 UTC [11559] ATTENTION:  specified neither
primary_conninfo nor restore_command
2019-10-09 15:08:24.881 UTC [11559] ASTUCE :  Le serveur de la base de données
va régulièrement interroger le sous-répertoire
    pg_wal pour vérifier les fichiers placés ici.
```

```
2019-10-09 15:08:24.881 UTC [11559] LOG:  entre en mode standby
2019-10-09 15:08:24.884 UTC [11559] LOG:  état de restauration cohérent atteint
à 0/301E030
2019-10-09 15:08:24.884 UTC [11559] LOG:  longueur invalide de l'enregistrement
à 0/301E030 : voulait 24, a eu 0
2019-10-09 15:08:24.885 UTC [11557] LOG:  le système de bases de données est
prêt pour accepter les connexions en lecture seule
10.0.3.141:5432 - accepting connections
10.0.3.141:5432 - accepting connections
2019-10-09 15:08:25,904 INFO: establishing a new patroni connection to the
postgres cluster
2019-10-09 15:08:25,939 WARNING: Could not activate Linux watchdog device:
"Can't open watchdog device: [Errno 2] No such file or directory: '/dev/
watchdog'"
2019-10-09 15:08:25,947 INFO: promoted self to leader by acquiring session lock
serveur en cours de promotion
2019-10-09 15:08:25.951 UTC [11559] LOG:  a reçu une demande de promotion
2019-10-09 15:08:25.951 UTC [11559] LOG:  la ré-exécution n'est pas nécessaire
2019-10-09 15:08:25,951 INFO: cleared rewind state after becoming the leader
2019-10-09 15:08:25.958 UTC [11559] LOG:  identifiant d'un timeline
nouvellement sélectionné : 11
2019-10-09 15:08:26.065 UTC [11559] LOG:  restauration terminée de l'archive
2019-10-09 15:08:26.078 UTC [11557] LOG:  le système de bases de données est
prêt pour accepter les connexions
2019-10-09 15:08:26,979 INFO: Lock owner: pg-1; I am pg-1
2019-10-09 15:08:27,029 INFO: no action.  i am the leader with the lock
^C
```

Nous arrêtons patroni après ce test.

Configuration du service

```
# /etc/systemd/system/patroni.service

[Unit]
Description=patroni service
Documentation=https://github.com/zalando/patroni

[Service]
Type=simple
User=postgres
Group=postgres
```

```
ExecStart=/usr/bin/patroni /etc/patroni/config.yml

# only patroni killed
KillMode=process
TimeoutSec=30
# don't restart on failure
Restart=no

[Install]
WantedBy=multi-user.target
```

Prise en compte par systemd :

```
$ sudo systemctl daemon-reload
$ sudo systemctl start patroni
```

Vérification dans les journaux applicatifs :

```
$ sudo journalctl -f
```

SUR PG-PATRONI2

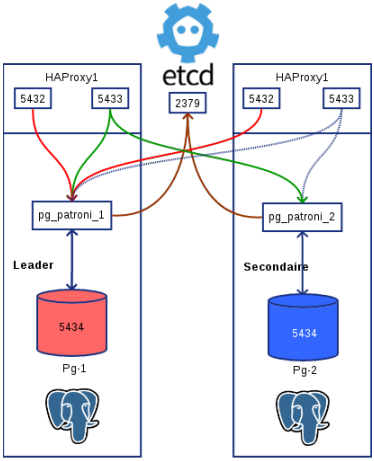

Figure 3: Ajout d'un module Patroni

La deuxième instance que l'on va raccrocher au cluster.

Config de pg-patroni2

```
scope: my-ha-cluster
name: pg-2

restapi:
  listen: 0.0.0.0:8008
  connect_address: 127.0.0.1:8008


etcd:
  host: 10.0.3.64:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        hot_standby: "on"
        wal_keep_segment: 8
        max_wal_senders: 5
        max_relication_slots: 5
        checkpoint_timeout: 30

  initdb: UTF8

  pg_hba:
  - host all dba all md5
  - host replication repl all md5

  users:
    dba:
      password: secret
      options:
       - createrole
       - createdb
    repl:
      password: secret
      options:
        - replication:
```

```
postgresql:
  listen: 10.0.3.201:5434
  connect_address: 10.0.3.201:5434
  data_dir: /var/lib/postgresql/data
  bin_dir: /usr/lib/postgresql/12/bin
  authentication:
    replication:
      username: repl
      password: secret
    superuser:
      username: dba
      password: secret
  parameters:
    unix_socket_directories: '/tmp'
```

Lancement de patroni sur pg-patroni2

```
$ patroni /etc/patroni/config.yml
```

```
2019-10-09 15:57:37,553 INFO: Failed to import patroni.dcs.consul
2019-10-09 15:57:37,564 INFO: Selected new etcd server http://10.0.3.64:2379
2019-10-09 15:57:37,572 INFO: No PostgreSQL configuration items changed, nothing to re
2019-10-09 15:57:37,583 INFO: Lock owner: pg-1; I am pg-2
2019-10-09 15:57:37,586 INFO: trying to bootstrap from leader 'pg-1'
2019-10-09 15:57:38,259 INFO: replica has been created using basebackup
2019-10-09 15:57:38,261 INFO: bootstrapped from leader 'pg-1'
2019-10-09 15:57:38,280 INFO: postmaster pid=11530
10.0.3.201:5432 - no response
2019-10-09 15:57:38.293 UTC [11530] LOG:  starting PostgreSQL 12.0 (Debian 12.0-
  1.pgdg90+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 2
2019-10-09 15:57:38.293 UTC [11530] LOG:  en écoute sur IPv4, adresse «
10.0.3.201 », port 5432
2019-10-09 15:57:38.297 UTC [11530] LOG:  écoute sur la socket Unix « /tmp/
.s.PGSQL.5432 »
2019-10-09 15:57:38.320 UTC [11532] LOG:  le système de bases de données a été
interrompu ; dernier lancement connu à 2019-10-09 15:57:37 UTC
2019-10-09 15:57:38.396 UTC [11532] LOG:  entre en mode standby
2019-10-09 15:57:38.400 UTC [11532] LOG:  la ré-exécution commence à 0/6000028
2019-10-09 15:57:38.402 UTC [11532] LOG:  état de restauration cohérent atteint
à 0/6000100
2019-10-09 15:57:38.402 UTC [11530] LOG:  le système de bases de données est
prêt pour accepter les connexions en lecture seule
2019-10-09 15:57:38.409 UTC [11536] FATAL:  n'a pas pu démarrer l'envoi des WAL
```

```
: ERREUR:  le slot de réplication « pg_2 » n'existe pas
2019-10-09 15:57:38.414 UTC [11537] FATAL:  n'a pas pu démarrer l'envoi des WAL
: ERREUR:  le slot de réplication « pg_2 » n'existe pas
10.0.3.201:5432 - accepting connections
10.0.3.201:5432 - accepting connections
2019-10-09 15:57:39,318 INFO: Lock owner: pg-1; I am pg-2
2019-10-09 15:57:39,319 INFO: does not have lock
2019-10-09 15:57:39,319 INFO: establishing a new patroni connection to the
postgres cluster
2019-10-09 15:57:39,336 INFO: no action.  i am a secondary and i am following a
leader
2019-10-09 15:57:43.425 UTC [11545] LOG:  Commence le flux des journaux depuis
le principal à 0/7000000 sur la timeline 20
2019-10-09 15:57:48,680 INFO: Lock owner: pg-1; I am pg-2
2019-10-09 15:57:48,680 INFO: does not have lock
2019-10-09 15:57:48,689 INFO: no action.  i am a secondary and i am following a
leader
2019-10-09 15:57:58,675 INFO: Lock owner: pg-1; I am pg-2
2019-10-09 15:57:58,675 INFO: does not have lock
2019-10-09 15:57:58,685 INFO: no action.  i am a secondary and i am following a
leader
```

Patroni effectue un basebackup sur le primaire, configure et lance le secondaire en réplication physique raccrochée au primaire.

TEST DU FAIL-OVER

On arrête brutalement `pg-patroni1` (arrêt du processus patroni sur pg-patroni1)

`pg-patroni2` prend la relève :

```
2019-10-09 16:02:52,027 INFO: Lock owner: pg-1; I am pg-2
2019-10-09 16:02:52,028 INFO: does not have lock
2019-10-09 16:02:52,033 INFO: no action.  i am a secondary and i am following a leader
2019-10-09 16:03:00.489 UTC [11709] LOG:  réplication terminée par le serveur primaire
2019-10-09 16:03:00.489 UTC [11709] DÉTAIL:  Fin du WAL atteint sur la timeline 22 à 0
2019-10-09 16:03:00.489 UTC [11709] FATAL:  n'a pas pu transmettre le message
de fin d'envoi de flux au primaire : aucun COPY en cours
2019-10-09 16:03:00.489 UTC [11705] LOG:  longueur invalide de l'enregistrement à 0/70
2019-10-09 16:03:00.493 UTC [11716] FATAL:  n'a pas pu se connecter au serveur
principal : n'a pas pu se connecter au serveur : Connexion refusée
        Le serveur est-il actif sur l'hôte « 10.0.3.141 » et accepte-t-il les connexio
```

```
        TCP/IP sur le port 5434 ?
2019-10-09 16:03:00,532 INFO: Got response from pg-1 http://127.0.0.1:8008/
patroni: b'{"state": "running", "server_version": 120000,
"postmaster_start_time": "2019-10-09 16:02:47.254 UTC", "role": "replica",
"xlog": {"paused": false, "replayed_timestamp": null, "replayed_location":
117441704, "received_location": 117441704}, "patroni": {"version": "1.6.0",
"scope": "my-ha-cluster"}, "database_system_identifier": "6745811559071353594",
"cluster_unlocked": true, "timeline": 22}'
2019-10-09 16:03:00,634 WARNING: Could not activate Linux watchdog device:
"Can't open watchdog device: [Errno 2] No such file or directory: '/dev/
watchdog'"
2019-10-09 16:03:00,638 INFO: promoted self to leader by acquiring session lock
serveur en cours de promotion
2019-10-09 16:03:00.641 UTC [11705] LOG:  a reçu une demande de promotion
2019-10-09 16:03:00.641 UTC [11705] LOG:  ré-exécution faite à 0/7000430
2019-10-09 16:03:00,641 INFO: cleared rewind state after becoming the leader
2019-10-09 16:03:00.645 UTC [11705] LOG:  identifiant d'un timeline
nouvellement sélectionné : 23
2019-10-09 16:03:00.721 UTC [11705] LOG:  restauration terminée de l'archive
2019-10-09 16:03:00.730 UTC [11703] LOG:  le système de bases de données est
prêt pour accepter les connexions
2019-10-09 16:03:01,659 INFO: Lock owner: pg-2; I am pg-2
2019-10-09 16:03:01,710 INFO: no action.  i am the leader with the lock
```

Le secondaire est promu.

TEST DU FAIL-BACK

Quand l'ancien primaire est rétabli, il devient secondaire, accroché au nouveau primaire promu.

```
patroni /etc/patroni/config.yml
2019-10-09 16:05:54,352 INFO: Failed to import patroni.dcs.consul
2019-10-09 16:05:54,363 INFO: Selected new etcd server http://10.0.3.64:2379
2019-10-09 16:05:54,371 INFO: No PostgreSQL configuration items changed, nothing to re
2019-10-09 16:05:54,378 WARNING: Postgresql is not running.
2019-10-09 16:05:54,378 INFO: Lock owner: pg-2; I am pg-1
2019-10-09 16:05:54,380 INFO: pg_controldata:
 max_wal_senders setting: 10
 Latest checkpoint's NextOID: 16399
 Latest checkpoint's oldestMulti's DB: 1
```

```
Database cluster state: shut down
Latest checkpoint's full_page_writes: on
Latest checkpoint's PrevTimeLineID: 23
Database system identifier: 6745811559071353594
Fake LSN counter for unlogged rels: 0/1
Min recovery ending loc's timeline: 0
Maximum data alignment: 8
Bytes per WAL segment: 16777216
max_connections setting: 100
max_prepared_xacts setting: 0
track_commit_timestamp setting: off
Backup start location: 0/0
Latest checkpoint's newestCommitTsXid: 0
wal_log_hints setting: on
Mock authentication nonce: 8acd62300c302ccd7e96b515e1de24171c625933d7c483e3b056e0ed95
Minimum recovery ending location: 0/0
Maximum length of identifiers: 64
Maximum size of a TOAST chunk: 1996
wal_level setting: replica
Latest checkpoint's REDO WAL file: 000000170000000000000007
Blocks per segment of large relation: 131072
Latest checkpoint's NextMultiOffset: 0
Latest checkpoint's REDO location: 0/7000588
Data page checksum version: 0
Database block size: 8192
Latest checkpoint location: 0/7000588
Latest checkpoint's oldestXID's DB: 1
Latest checkpoint's NextXID: 0:495
Latest checkpoint's oldestMultiXid: 1
Maximum columns in an index: 32
Time of latest checkpoint: Wed Oct  9 16:05:46 2019
Catalog version number: 201909212
Latest checkpoint's oldestActiveXID: 0
Latest checkpoint's oldestCommitTsXid: 0
Latest checkpoint's TimeLineID: 23
max_locks_per_xact setting: 64
pg_control version number: 1201
Latest checkpoint's oldestXID: 480
WAL block size: 8192
```

```
 max_worker_processes setting: 8
 Latest checkpoint's NextMultiXactId: 1
 pg_control last modified: Wed Oct  9 16:05:46 2019
 Float8 argument passing: by value
 Backup end location: 0/0
 End-of-backup record required: no
 Float4 argument passing: by value
 Date/time type storage: 64-bit integers
 Size of a large-object chunk: 2048

2019-10-09 16:05:54,383 INFO: Lock owner: pg-2; I am pg-1
2019-10-09 16:05:54,397 INFO: Local timeline=23 lsn=0/7000588
2019-10-09 16:05:54,401 INFO: master_timeline=24
2019-10-09 16:05:54,401 INFO: master: history=1 0/1640480   no recovery target specifi

2   0/1640610   no recovery target specified
3   0/16407A0   no recovery target specified
4   0/1640930   no recovery target specified
5   0/30001C0   no recovery target specified
6   0/3000350   no recovery target specified
7   0/3000468   no recovery target specified
8   0/301D768   no recovery target specified
9   0/301DEC0   no recovery target specified
10  0/301E030   no recovery target specified
11  0/301E1C0   no recovery target specified
12  0/301E2D8   no recovery target specified
13  0/301E468   no recovery target specified
14  0/40000A0   no recovery target specified
15  0/40001B8   no recovery target specified
16  0/50000A0   no recovery target specified
17  0/50001F8   no recovery target specified
18  0/50001F8   no recovery target specified
19  0/501C2E0   no recovery target specified
20  0/70001C0   no recovery target specified
21  0/7000318   no recovery target specified
22  0/70004A8   no recovery target specified
23  0/7000638   no recovery target specified

2019-10-09 16:05:54,401 INFO: Lock owner: pg-2; I am pg-1
```

```
2019-10-09 16:05:54,415 INFO: starting as a secondary
2019-10-09 16:05:54,427 INFO: postmaster pid=12090
10.0.3.141:5432 - no response
2019-10-09 16:05:54.438 UTC [12090] LOG:  starting PostgreSQL 12.0 (Debian 12.0-
  1.pgdg90+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 2
2019-10-09 16:05:54.438 UTC [12090] LOG:  en écoute sur IPv4, adresse «
10.0.3.141 », port 5432
2019-10-09 16:05:54.445 UTC [12090] LOG:  écoute sur la socket Unix « /tmp/
.s.PGSQL.5434 »
2019-10-09 16:05:54.459 UTC [12092] LOG:  le système de bases de données a été
arrêté à 2019-10-09 16:05:46 UTC
2019-10-09 16:05:54.460 UTC [12092] LOG:  entre en mode standby
2019-10-09 16:05:54.463 UTC [12092] LOG:  état de restauration cohérent atteint
à 0/7000600
2019-10-09 16:05:54.463 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
2019-10-09 16:05:54.463 UTC [12090] LOG:  le système de bases de données est
prêt pour accepter les connexions en lecture seule
2019-10-09 16:05:54.470 UTC [12096] LOG:  récupération du fichier historique
pour la timeline 24 à partir du serveur principal
2019-10-09 16:05:54.474 UTC [12096] LOG:  Commence le flux des journaux depuis
le principal à 0/7000000 sur la timeline 23
2019-10-09 16:05:54.474 UTC [12096] LOG:  réplication terminée par le serveur
primaire
2019-10-09 16:05:54.474 UTC [12096] DÉTAIL:  Fin du WAL atteint sur la timeline
23 à 0/7000638
2019-10-09 16:05:54.475 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
2019-10-09 16:05:54.475 UTC [12096] FATAL:  arrêt du processus walreceiver
suite à la demande de l'administrateur
2019-10-09 16:05:54.475 UTC [12092] LOG:  la nouvelle timeline cible est 24
2019-10-09 16:05:54.475 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
2019-10-09 16:05:54.475 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
2019-10-09 16:05:54.475 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
10.0.3.141:5432 - accepting connections
10.0.3.141:5432 - accepting connections
```

```
2019-10-09 16:05:55,471 INFO: Lock owner: pg-2; I am pg-1
2019-10-09 16:05:55,471 INFO: does not have lock
2019-10-09 16:05:55,472 INFO: establishing a new patroni connection to the
postgres cluster
2019-10-09 16:05:55,494 INFO: no action.  i am a secondary and i am following a
leader
2019-10-09 16:05:59.478 UTC [12092] LOG:  longueur invalide de l'enregistrement
à 0/7000600 : voulait 24, a eu 0
2019-10-09 16:06:00,436 INFO: Lock owner: pg-2; I am pg-1
2019-10-09 16:06:00,437 INFO: does not have lock
2019-10-09 16:06:00,454 INFO: no action.  i am a secondary and i am following a
leader
```

Il se resynchronise avec le nouveau primaire et devient un secondaire.

AJOUT D'UN NOUVEAU SECONDAIRE PATRONI AU CLUSTER

Création du container

```
$ sudo lxc-create -n pg-patroni3 -t debian -- -r stretch
$ sudo lxc-start -n pg-patroni3
```

sur pg-patroni3

```
$ sudo apt update
$ sudo apt upgrade -y
$ sudo apt install -y vim sudo curl wget gpg
$ sudo  vim /etc/apt/source.list.d/pgpdg.list
```

ajout du dépôt pgdg deb http://apt.postgresql.org/pub/repos/apt/ stretch-pgdg main

```
$ curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
$ sudo apt update
$ sudo  apt install postgresql-12
$ sudo  apt install python3-pip
$ sudo  apt install python3-psycopg2
$ sudo pip3 install python-etcd
$ sudo  pip3 install patroni
$ sudo  patroni
Usage: /usr/local/bin/patroni config.yml
    Patroni may also read the configuration from the PATRONI_CONFIGURATION environment variable
```

Config de pg-patroni3

Création du fichier /etc/patroni/config.yml, on le raccroche au cluster de configuration etcd (10.0.3.64) et on lui demande d'écouter sur 10.0.3.32:5434 (adresse affectée par `lxc-create`).

```
scope: my-ha-cluster
name: pg-3

restapi:
  listen: 0.0.0.0:8008
  connect_address: 127.0.0.1:8008


etcd:
  host: 10.0.3.64:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        hot_standby: "on"
        wal_keep_segment: 8
        max_wal_senders: 5
        max_relication_slots: 5
        checkpoint_timeout: 30

  initdb: UTF8

  pg_hba:
  - host all dba all md5
  - host replication repl all md5

  users:
    dba:
      password: secret
      options:
       - createrole
       - createdb
    repl:
```

```
      password: secret
      options:
        - replication:

postgresql:
  listen: 10.0.3.68:5434
  connect_address: 10.0.3.68:5434
  data_dir: /var/lib/postgresql/data/main
  bin_dir: /usr/lib/postgresql/12/bin
  authentication:
    replication:
      username: repl
      password: secret
    superuser:
      username: dba
      password: secret
  parameters:
    unix_socket_directories: '/tmp'
```

Au lancement de Patroni, pg-patroni3 récupère l'instance sur le leader et se raccroche à la timeline de celui-ci.

Il devient un nouveau secondaire.

```
$ patroni /etc/patroni/config.yml
2019-10-09 16:37:14,848 INFO: Failed to import patroni.dcs.consul
2019-10-09 16:37:14,859 INFO: Selected new etcd server http://10.0.3.64:2379
2019-10-09 16:37:14,866 INFO: No PostgreSQL configuration items changed, nothing to re
2019-10-09 16:37:14,885 INFO: Lock owner: pg-2; I am pg-3
2019-10-09 16:37:14,890 INFO: trying to bootstrap from leader 'pg-2'
2019-10-09 16:37:16,088 INFO: replica has been created using basebackup
2019-10-09 16:37:16,090 INFO: bootstrapped from leader 'pg-2'
2019-10-09 16:37:16,117 INFO: postmaster pid=10448 10.0.3.68:5432 - no response
2019-10-09 16:37:16.134 UTC [10448] LOG:  starting PostgreSQL 12.0 (Debian 12.0-
  1.pgdg90+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 2
2019-10-09 16:37:16.134 UTC [10448] LOG:  en écoute sur IPv4, adresse «
10.0.3.68 », port 5432
2019-10-09 16:37:16.137 UTC [10448] LOG:  écoute sur la socket Unix « /tmp/.s.PGSQL.54
2019-10-09 16:37:16.152 UTC [10450] LOG:  le système de bases de données a été
interrompu ; dernier lancement connu à 2019-10-09 16:37:15 UTC
2019-10-09 16:37:16.220 UTC [10450] LOG:  entre en mode standby
2019-10-09 16:37:16.224 UTC [10450] LOG:  la ré-exécution commence à 0/8000028
2019-10-09 16:37:16.225 UTC [10450] LOG:  état de restauration cohérent atteint à 0/80
2019-10-09 16:37:16.226 UTC [10448] LOG:  le système de bases de données est
```

```
prêt pour accepter les connexions en lecture seule
2019-10-09 16:37:16.231 UTC [10454] FATAL:  n'a pas pu démarrer l'envoi des WAL
: ERREUR:  le slot de réplication « pg_3 » n'existe pas
2019-10-09 16:37:16.236 UTC [10455] FATAL:  n'a pas pu démarrer l'envoi des WAL
: ERREUR:  le slot de réplication « pg_3 » n'existe pas
10.0.3.68:5432 - accepting connections
10.0.3.68:5432 - accepting connections
2019-10-09 16:37:17,163 INFO: Lock owner: pg-2; I am pg-3
2019-10-09 16:37:17,164 INFO: does not have lock
2019-10-09 16:37:17,164 INFO: establishing a new patroni connection to the postgres cl
2019-10-09 16:37:17,187 INFO: no action.  i am a secondary and i am following a leader
2019-10-09 16:37:20,437 INFO: Lock owner: pg-2; I am pg-3
2019-10-09 16:37:20,437 INFO: does not have lock
2019-10-09 16:37:20,442 INFO: no action.  i am a secondary and i am following a leader
2019-10-09 16:37:21.248 UTC [10464] LOG:  Commence le flux des journaux depuis
le principal à 0/9000000 sur la timeline 24
2019-10-09 16:37:30,425 INFO: Lock owner: pg-2; I am pg-3
2019-10-09 16:37:30,425 INFO: does not have lock
2019-10-09 16:37:30,431 INFO: no action.  i am a secondary and i am following a leader
```

SÉCURISATION DU CLUSTER ETCD

Changement de la configuration des instances patroni

Pour garantir la continuité de service du cluster Etcd et le fonctionnement de Patroni, nous indiquons à chaque nœud Patroni, la liste de tous les nœuds etcd :

```
etcd:
  hosts: etcd1:2379,etcd2:2379,etcd3:2379
```

> Remarquez que `hosts` est au pluriel pour permettre l'utilisation d'une liste.

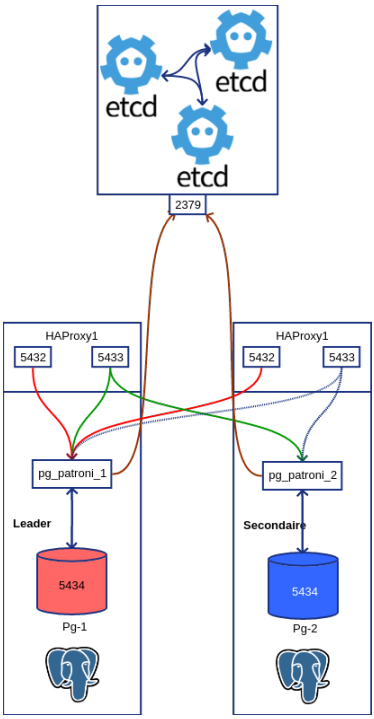

Figure 4: Schéma 2 modules Patroni et proxy etcd

Activation de l'authentification Etcd

L'activation de l'authentification se fait en définissant les utilisateurs et leur mot de passe, puis en activant l'authentification.

NB: activer l'authentification va de paire avec le chiffrement des communications (https)

SE CONNECTER AUTOMATIQUEMENT AU LEADER

Pour se connecter au cluster patroni, il faut déterminer qui est le *leader* et qui sont les secondaires. L'*API* de Patroni permet de récupérer cette information en se connectant sur le port 8008 en http, le statut 200 indique que nous sommes en présence du *leader*.

HA PROXY SUR LE CLUSTER PATRONI

Sur chaque nœud patroni, on installe HaProxy :

```
$ sudo apt-get install haproxy
```

On se propose de configurer un round-robin en lecture sur les secondaires et une redirection vers le leader en cas d'accès en écriture.

> Les lectures seront effectuées sur le port 5433 et redirigées vers le port 5434 d'un des secondaires Les écritures seront effectuées sur le port 5432 et redirigées vers le port 5434 du *leader*

Mise en place

- Toutes les instances écoutent sur le port 5434.
- Sur chacun des noeuds, /etc/hosts contiendra les adresses ip des noeuds :
  - pg-patroni1, p1 ou pg-1
  - pg-patroni2, p2 ou pg-2
  - pg-patroni3, p3 ou pg-3

CONFIGURER HA PROXY

Sur toutes les instances Patroni, configurer HA proxy comme tel :

```
listen production
    bind    *:5432
    option  httpchk OPTIONS /master
    http-check      expect status 200
    default-server inter 2s fastinter 1s rise 2 fall 1 on-marked-down shutdown-sessions
    server  pg-1    pg-1:5434 check port 8008
    server  pg-2    pg-2:5434 check port 8008
    server  pg-3    pg-3:5434 check port 8008

listen standby
    bind    *:5433
    option  httpchk OPTIONS /replica
    http-check      expect status 200
    balance roundrobin
    default-server inter 2s fastinter 1s rise 2 fall 1 on-marked-down shutdown-sessions
    server  pg-1    pg-1:5434 check port 8008
    server  pg-2    pg-2:5434 check port 8008
    server  pg-3    pg-3:5434 check port 8008
```

HAProxy doit être en mesure de déterminer via le *http-check*, qui est le *leader* et le rendre disponible sur le port 5432.

Les autres nœuds en lecture seule, doivent être accessibles à tour de rôle sur le port 5433. L'api de Patroni réponds avec un *status 200* sur l'url /replica pour tous les secondaires que nous avons organisés en *round-robin*.

VÉRIFICATION SUR TOUS LES NŒUDS

Pour vérifier que Haproxy est fonctionnel, les stats sont disponibles sur le port 7000 aux *url* :

- http://p1:7000
- http://p2:7000
- http://p3:7000

VÉRIFICATION DE LA CONNEXION AU CLUSTER SUR LES 2 PORTS

```
nmap  -p 5432,5433 p1 p2 p3

Starting Nmap 7.40 ( https://nmap.org ) at 2020-01-09 17:03 CET
Nmap scan report for p1 (10.0.3.141)
Host is up (0.00018s latency).
rDNS record for 10.0.3.141: pg-patroni1
PORT     STATE SERVICE
5432/tcp open  postgresql
5433/tcp open  pyrrho

Nmap scan report for p2 (10.0.3.201)
Host is up (0.00011s latency).
rDNS record for 10.0.3.201: pg-patroni2
PORT     STATE SERVICE
5432/tcp open  postgresql
5433/tcp open  pyrrho

Nmap scan report for p3 (10.0.3.68)
Host is up (0.000080s latency).
rDNS record for 10.0.3.68: pg-patroni3
PORT     STATE SERVICE
5432/tcp open  postgresql
5433/tcp open  pyrrho
```

Vérification du *round-robin* pour l'accès en lecture seule

```
$ while : ; do psql -P pager -h pg-2 -p 5433 -U dba -c "show primary_conninfo;" dba;
  sleep 1;  done
                                    primary_conninfo
-------------------------------------------------------------------------------------------------
 user=repl password=* host=10.0.3.141 port=5434 sslmode=prefer application_name=pg-3
(1 row)

                                    primary_conninfo
-------------------------------------------------------------------------------------------------
 user=repl password=* host=10.0.3.68 port=5434 sslmode=prefer application_name=pg-1
(1 row)


                                    primary_conninfo
-------------------------------------------------------------------------------------------------
 user=repl password=* host=10.0.3.141 port=5434 sslmode=prefer application_name=pg-3
(1 row)

                                    primary_conninfo
-------------------------------------------------------------------------------------------------
```

```
 user=repl password=* host=10.0.3.68 port=5434 sslmode=prefer application_name=pg-1
(1 row)
```

Répéter le test sur les 2 autres nœuds :

```
$ while : ; do psql -P pager -h pg-1 -p 5433 -U dba -c "show primary_conninfo;"
 dba;  sleep 1;  done
...
$ while : ; do psql -P pager -h pg-3 -p 5433 -U dba -c "show primary_conninfo;"
 dba;  sleep 1;  done
```

On doit obtenir le même résultat.

VÉRIFICATION DU LEADER

```
$ while : ; do psql -h  p1,p2,p3 -P pager -p 5432 -U dba -c "show primary_conninfo;"
 dba ; sleep 1 ;clear;done
```

Seul le *leader* doit répondre à une demande de connexion sur le port 5432.

On peut simuler une perte du leader et vérifier la bascule vers un des secondaires promu.

HAUTE DISPONIBILITÉ

Enfin, pour bénéficier de la haute disponibilité du cluster, les clients doivent se connecter en mentionnant les 3 serveurs du cluster dans leur chaîne de connexion pour ainsi se connecter au premier disponible qui se chargera de la redirection, via HaProxy :

Connexion en écriture, port 5432

```
while : ; do psql -P pager -h pg-1,pg-2,pg-3 -p 5432 -U dba -c "show primary_conninfo;"
 dba;  sleep 1;  done
```

Connexion en lecture, port 5433

```
while : ; do psql -P pager -h pg-1,pg-2,pg-3 -p 5433 -U dba -c "show primary_conninfo;"
 dba;  sleep 1;  done
```

En cas de perte du nœud sur laquelle elle est connectée, l'application n'aura qu'à attendre le temps de la promotion initiée par Patroni et ré-initier la même connexion.

HaProxy basculera sur un nœud disponible, que ce soit le nouveau primaire ou un autre secondaire.

QUELQUES TESTS

FAILOVER

Vérifier que le secondaire descendu, disparaît du *round-robin*

FAILBACK

Le nœud rétabli, n'est accessible que lorsqu'il a raccroché à la *timeline* du *leader*.

Vérifier qu'il est réinséré dans le *roundrobin*.

NB

L'application doit pouvoir gérer le défaut de connexion temporaire lorsqu'un secondaire ou le primaire tombe (essai à nouveau 2s plus tard pour que cela soit transparent).

PASSAGE DU CLUSTER ETCD EN HTTPS ?

On se propose de chiffrer les accès au cluster `etcd` par SSL.

Sur le premier nœud

```
$ sudo -iu etcd
$ mkdir ~/bin
$ curl -s -L -o ~/bin/cfssl https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
$ curl -s -L -o ~/bin/cfssljson https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
$ chmod +x ~/bin/{cfssl,cfssljson}
$ export PATH=$PATH:~/bin

$ mkdir ~/etcd-ca
$ cd ~/etcd-ca

$ echo '{"CN":"CA","key":{"algo":"rsa","size":2048}}'
  | cfssl gencert -initca -
  | cfssljson -bare ca -

$ echo '{"signing":{"default":{"expiry":"43800h","usages":["signing",
"key encipherment", "server auth","client auth"]}}}' > ca-config.json
```

Recopie du ca-key.pem et ca.pem sur les autres nœuds depuis l'extérieur du cluster :

```
# cp etcd1/rootfs/var/lib/etcd/etcd-ca/ca.pem etcd2/rootfs/var/lib/etcd/etcd-ca/
# cp etcd1/rootfs/var/lib/etcd/etcd-ca/ca-key.pem etcd2/rootfs/var/lib/etcd/etcd-ca/
```

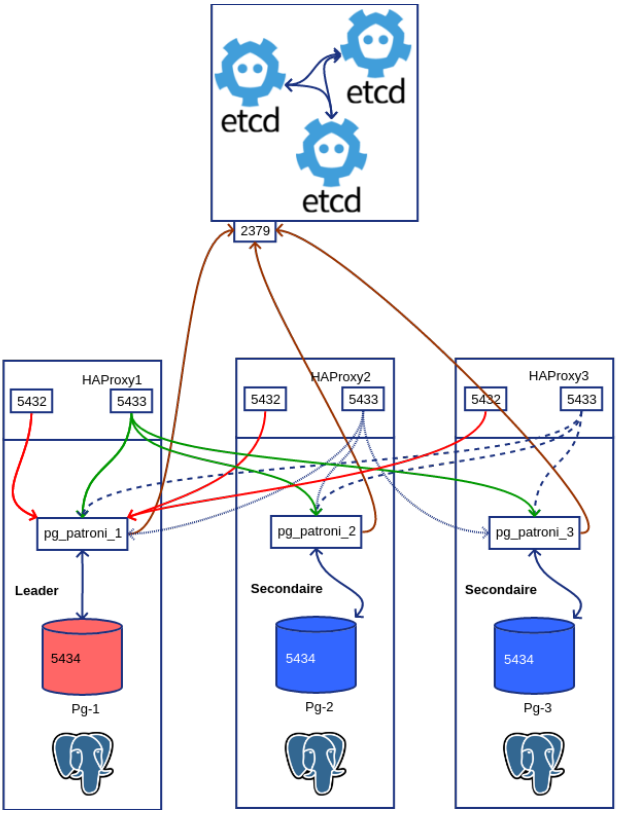

Figure 5: Schéma complet

```
# cp etcd1/rootfs/var/lib/etcd/etcd-ca/ca.pem etcd3/rootfs/var/lib/etcd/etcd-ca/
# cp etcd1/rootfs/var/lib/etcd/etcd-ca/ca-key.pem etcd3/rootfs/var/lib/etcd/etcd-ca/
```

Correction des droits (le uid 1000 peut différer) :

```
# chown 1000:0 etcd2/rootfs/var/lib/etcd/etcd-ca/ca-key.pem
# chown 1000:0 etcd3/rootfs/var/lib/etcd/etcd-ca/ca-key.pem
```

Génération d'un certificat sur chaque nœud:

```
$ export NAME=etcd1
$ export ADDRESS=10.0.3.64,$NAME.mydomain.com,$NAME

$ echo '{"CN":"'$NAME'","hosts":[""],"key":{"algo":"rsa","size":2048}}'
| cfssl gencert -config=ca-config.json -ca=ca.pem -ca-key=ca-key.pem
-hostname="$ADDRESS" - | cfssljson -bare $NAME
```

Déployer la clef et le certificat dans `/etc/etcd/` :

```
# cp ca.pem /etc/etcd/etcd-ca.crt
# cp etcd1.pem /etc/etcd/server.crt
# cp etcd1-key.pem root@etcd1:/etc/etcd/server.key

# sudo chmod 600 /etc/etcd/server.key
```

Le certificat ca.pem sera déployé sur chacun des nœuds etcd dans le répertoire `/etc/` pour qu'il soit commun à tous les nœuds du cluster `etcd`.

Déploiement du même certificats sur tous le nœuds :

En dehors de tout container :

```
# cd /var/lib/lxc

# cp etcd1/rootfs/etc/etcd/etcd-ca.crt etcd2/rootfs/etc/etcd/
# cp etcd1/rootfs/etc/etcd/etcd-ca.crt etcd3/rootfs/etc/etcd/
```

Génération des certificats sur chaque nœud, de la même façon que pour etcd1 en changeant le nom de l'hôte.

Au final on obtient :

```
/lxc# ls -ul etcd1/rootfs/etc/etcd/
total 20
-rw-r--r-- 1 root root 1119 août  25 14:21 etcd-ca.crt
-rw-r--r-- 1 frbn frbn 3800 mars  12 14:11 etcd.conf.yml
-rw-r--r-- 1 root root 1220 août  25 14:09 server.crt
```

```
-rw------- 1 frbn root 1679 août  25 14:09 server.key

/lxc# ls -ul etcd2/rootfs/etc/etcd/
total 20
-rw-r--r-- 1 root root 1119 août  25 14:43 etcd-ca.crt
-rw-r--r-- 1 frbn frbn 3800 mars  12 14:12 etcd.conf.yml
-rw-r--r-- 1 root root 1220 août  25 14:43 server.crt
-rw------- 1 root root 1679 août  25 14:44 server.key

/lxc# ls -ul etcd3/rootfs/etc/etcd/
total 16
-rw-r--r-- 1 root root 1119 août  25 14:43 etcd-ca.crt
-rw-r--r-- 1 frbn frbn 3667 août  24 17:55 etcd.conf.yml
-rw-r--r-- 1 root root 1220 août  25 14:43 server.crt
-rw------- 1 root root 1679 août  25 14:44 server.key
```

Correction des droits sur le certificat sur chaque nœud, depuis l'intérieur du container :

```
root@etcd2:/etc/etcd# chown etcd:root server.key
...
root@etcd3:/etc/etcd# chown etcd:root server.key
```

Modification de la configuration d'`etcd` sur chaque nœud, nous chiffrons tous accès, pour les clients et pour les nœuds entre-eux :

```
client-transport-security:
  # Path to the client server TLS cert file.
  cert-file: /etc/etcd/server.crt

  # Path to the client server TLS key file.
  key-file: /etc/etcd/server.key

  # Enable client cert authentication.
  client-cert-auth: true

  # Path to the client server TLS trusted CA cert file.
  trusted-ca-file: /etc/etcd/etcd-ca.crt

  # Client TLS using generated certificates
  auto-tls: false
```

```
peer-transport-security:
  # Path to the peer server TLS cert file.
  cert-file: /etc/etcd/server.crt

  # Path to the peer server TLS key file.
  key-file: /etc/etcd/server.key

  # Enable peer client cert authentication.
  client-cert-auth: true

  # Path to the peer server TLS trusted CA cert file.
  trusted-ca-file: /etc/etcd/etcd-ca.crt

  # Peer TLS using generated certificates.
  auto-tls: false
```

l'utilisateur qui lance `etcd` doit pouvoir accéder en lecture au fichier `/etc/etcd/server.key` (`chown etcd:root`).

Vérification de l'api :

sur etcd1

```
etcd@etcd1:~$ etcdctl  --endpoints=https://etcd1:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd1:~$ etcdctl  --endpoints=https://etcd2:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd1:~$ etcdctl  --endpoints=https://etcd3:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false
```

sur etcd2

```
etcd@etcd2:~$ etcdctl  --endpoints=https://etcd1:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd2:~$ etcdctl  --endpoints=https://etcd2:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd2:~$ etcdctl  --endpoints=https://etcd3:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false
```

sur etcd3

```
etcd@etcd3:~$ etcdctl  --endpoints=https://etcd1:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd3:~$ etcdctl  --endpoints=https://etcd2:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false

etcd@etcd3:~$ etcdctl  --endpoints=https://etcd3:2379 --cacert=/etc/etcd/etcd-ca.crt
 --cert=/etc/etcd/server.crt --key=/etc/etcd/server.key member list
203f65fca4ab0c0c, started, etcd1, https://etcd1:2380, https://etcd1:2379, false
5803d57c81942472, started, etcd3, https://etcd3:2380, https://etcd3:2379, false
fed3383484bd04ca, started, etcd2, https://etcd2:2380, https://etcd2:2379, false
```

configuration de Patroni sur pg-patroni1

Maintenant que notre cluster `etcd` est passé en https, nous modifions le protocole de communication de Patroni afin d'utiliser ce chiffrement.

Génération des certificats sur les instances Patroni

Nous utilisons cfssl (openssl ferait l'affaire) :

```
postgres@pg-1:~$ mkdir bin
postgres@pg-1:~$ curl -s -L -o ~/bin/cfssl https://pkg.cfssl.org/R1.2/
cfssl_linux-amd64
postgres@pg-1:~$ curl -s -L -o ~/bin/cfssljson https://pkg.cfssl.org/R1.2/
cfssljson_linux-amd64
postgres@pg-1:~$ chmod +x ~/bin/{cfssl,cfssljson}
postgres@pg-1:~$ export PATH=$PATH:~/bin
postgres@pg-1:~$ export NAME=pg-1
postgres@pg-1:~$ export ADDRESS=10.0.3.141,$NAME.mydomain.com,$NAME

postgres@pg-1:~$ echo '{"signing":{"default":{"expiry":"43800h",
"usages":["signing","keyb encipherment","server auth","client auth"]}}}'
> ca-config.json

postgres@pg-1:~$ echo '{"CN":"'$NAME'","hosts":[""],"key":{"algo":"rsa",
"size":2048}}' | cfssl gencert -config=ca-config.json -ca=ca.pem
-ca-key=ca-key.pem -hostname="$ADDRESS" - | cfssljson -bare $NAME

postgres@pg-1:~$ cp pg-1.pem /etc/patroni/server.crt
postgres@pg-1:~$ cp pg-1-key.pem /etc/patroni/server.key
```

Test de l'accès au cluster etcd depuis le nœud pg-patroni1 :

installer etcdctl

```
postgres@pg-1:~$curl -s https://api.github.com/repos/etcd-io/etcd/releases/latest
| grep browser_download_url   | grep linux-amd64   | cut -d '"' -f 4
| wget -qi -

postgres@pg-1:~$tar xzf etcd-v3.3.25-linux-amd64.tar.gz
postgres@pg-1:~$ cp etcd-v3.3.25-linux-amd64/etcdctl ~/bin/
```

Accès chiffré au cluster

```
$etcdctl  --endpoints=https://etcd1:2379 --ca-file=/etc/patroni/etcd-ca.crt
```

```
--cert-file=/etc/patroni/server.crt --key-file=/etc/patroni/server.key
member list

203f65fca4ab0c0c: name=etcd1 peerURLs=https://etcd1:2380
clientURLs=https://etcd1:2379 isLeader=true
5803d57c81942472: name=etcd3 peerURLs=https://etcd3:2380
clientURLs=https://etcd3:2379 isLeader=false
fed3383484bd04ca: name=etcd2 peerURLs=https://etcd2:2380
clientURLs=https://etcd2:2379 isLeader=false
```

À vérifier en passant par les 2 autres *endpoints* `etcd2` et `etcd3`.

> La syntaxe de la commande etcdctl a changé dans la nouvelle version.

Passage en https des accès à etcd dans la configuration de patroni sur pg-1

Dans un premier temps on vérifie que patroni arrive à dialoguer en https avec etcd1 :

```
etcd:
  protocol: https
  host: etcd1:2379
  cacert: /etc/patroni/etcd-ca.crt
  cert: /etc/patroni/server.crt
  key: /etc/patroni/server.key
```

Dans un 2ème temps, qu'il arrive à le faire avec etcd2 et etcd3 :

```
etcd:
  protocol: https
  host: etcd2:2379
  cacert: /etc/patroni/etcd-ca.crt
  cert: /etc/patroni/server.crt
  key: /etc/patroni/server.key
```

```
etcd:
  protocol: https
  host: etcd3:2379
  cacert: /etc/patroni/etcd-ca.crt
  cert: /etc/patroni/server.crt
  key: /etc/patroni/server.key
```

Les 3 :

```
etcd:
  protocol: https
  hosts: etcd1:2379,etcd2:2379,etcd3:2379
  cacert: /etc/patroni/etcd-ca.crt
  cert: /etc/patroni/server.crt
  key: /etc/patroni/server.key
```

> Il convient de vérifier que le cluster Patroni est fonctionnel malgré une perte d'un des nœuds `etcd`.

Conclusion du passage en HTTPS

Les opérations de ce chapitre doivent être répétées sur les 2 autres nœuds pg-patroni2 et pg-patroni3.

RÉPLICATION SYNCHRONE AVEC PATRONI 2

La version 2 de Patroni supporte la réplication synchrone sur un ou plusieurs réplicats Les options à activer sont :

```
postgresql:
  ...
  parameters:
  ...
synchronous_commit: 'on'
synchronous_standby_names: "*"
```

Il faut redémarrer chaque nœud et passer en réplication synchrone en modifiant la configuration dynamique du cluster :

```
postgres $ patronictl -c /etc/patroni/config.yml edit-config
```

La configuration modifiable s'affiche :

```
loop_wait: 10
maximum_lag_on_failover: 1048576
postgresql:
  checkpoint_timeout: 30
  hot_standby: 'on'
  max_relication_slots: 5
  max_wal_senders: 5
  parameters: null
```

```
  use_pg_rewind: true
  use_slots: true
  wal_keep_segment: 8
  wal_level: replica
retry_timeout: 10
synchronous_node_count: 1
ttl: 30
```

Le paramètre qui active la réplication synchrone est :

```
synchronous_mode: true
```

Dès la sauvegarde de la configuration, Patroni effectue un rechargement sur toutes les instances et la réplication devient synchrone sur un réplica.

```
$ patronictl -c  /etc/patroni/config.yml list
+ Cluster: my-ha-cluster (6876375338380834518) ----+----+-----------+
| Member | Host           | Role         | State   | TL | Lag in MB |
+--------+----------------+--------------+---------+----+-----------+
| pg-1   | 10.0.3.85:5434 | Leader       | running | 14 |           |
| pg-2   | 10.0.3.35:5434 | Sync Standby | running | 14 |         0 |
| pg-3   | 10.0.3.70:5434 | Replica      | running | 14 |         0 |
+--------+----------------+--------------+---------+----+-----------+
```

Le paramètre `synchronous_node_count` détermine le nombre de réplicas synchrones désiré. Il faut attendre un peu plus longtemps pour que le deuxième secondaire soit déclaré synchrone :

```
$ patronictl -c  /etc/patroni/config.yml list
+ Cluster: my-ha-cluster (6876375338380834518) ----+----+-----------+
| Member | Host           | Role         | State   | TL | Lag in MB |
+--------+----------------+--------------+---------+----+-----------+
| pg-1   | 10.0.3.85:5434 | Leader       | running | 15 |           |
| pg-2   | 10.0.3.35:5434 | Sync Standby | running | 15 |         0 |
| pg-3   | 10.0.3.70:5434 | Sync Standby | running | 15 |         0 |
+--------+----------------+--------------+---------+----+-----------+
```

AMÉLIORATION : INTÉGRATION PGBACKREST POUR LA CRÉATION DES INSTANCES (FAIL-BACK OU AJOUT D'UN NŒUD)