PostgreSQL Patroni Cluster Introduction, Configuration, Implementation And Administration Step by Step Document

Author : Ahsan Iqbal
Created date : 11th August, 2025
Doc Version : V-1.0

Reviewer:

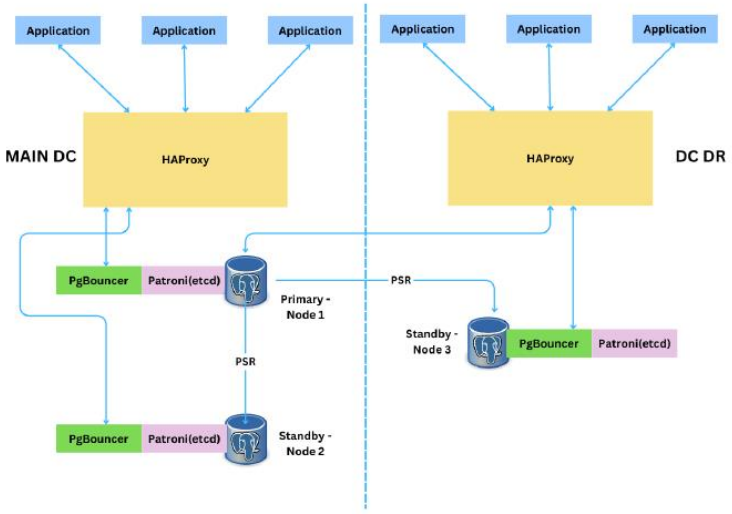
V#	Name	Designation	Email Address	Company
V-1.1	Syed Asad	Database Administrator	syed.asad@securemaxtech.com	Securemax
V-1.1	Sharukh Shaikh	Database Administrator	Sharukhshaikh1807@gmail.com	Securemax

1. Introduction 4
2. Overview of Architecture Components 4
3. Pre-installation Prerequisites 5
3.1 Open the network Ports 5
3.2 Download software / Media 6
3.3 Modify the /etc/hosts file to include the hostnames and IP addresses of the nodes 6
4. Install PostgreSQL, Patroni, ETCD & Pgbouncer on Data nodes 7
4.1 Install PostgreSQL 7
4.2 Install Patroni 7
4.3 Install ETCD 7
4.4 Install Pgbouncer 7
4.5 Check the status of PostgreSQL, patroni, etcd 8
4.6 Disable the services 8
5. Configure etcd distributed store 9
5.1 Modify the configuration file 9
5.2 Enable and start the etcd service on all nodes 10
5.3 Check the etcd cluster members and leader Node 11
6. Configure Patroni 11
6.1 Create the /etc/patroni/patroni.yml configuration file and add the following configuration for each Patroni Node 11
6.2 Check the patroni. service file if it is not created create it manually & Reload the systemd to be aware of the new service 16
6.3 Start the Patroni service on each Node one by one 16
6.4 Check for any errors of the patroni services 16
6.5 Check the cluster status 16
7. Configure the PgBouncer 17
8. Configure the HAProxy 18
9. Basic Administration and Troubleshooting Commands 19
10. References 20

1. Introduction

This document outlines the installation and configuration process for a high-availability (HA) database architecture designed to ensure robust performance and failover capabilities. The architecture leverages a combination of HAProxy load balancers, PostgreSQL databases with Patroni for automated failover, and PgBouncer for connection pooling. It is structured to support a primary Data Center (MAIN DC) with a Disaster Recovery (DC DR) site, ensuring continuous operation and data integrity under various failure scenarios.

2. Overview of Architecture Components



HAProxy: Acts as a load balancer and reverse proxy, distributing incoming application traffic across the database nodes in both MAIN DC and DC DR. It ensures high availability and efficient load distribution.

Patroni: A template for high-availability PostgreSQL clusters, managing automated failover and replication between primary and standby nodes to maintain data consistency and availability.

PgBouncer: A lightweight connection pooler for PostgreSQL, optimizing database connections by reusing them, reducing resource usage, and improving performance.

PostgreSQL Nodes: Include a Primary Node (active) and Standby Nodes (replicas) in both MAIN DC and DC DR. The Primary Node handles read/write operations, while Standby Nodes provide redundancy and support read-only queries.

PSR (Primary-Standby Replication): Facilitates real-time data replication from the Primary Node to Standby Nodes, ensuring data synchronization across the architecture.

Applications: Client applications that interact with the database infrastructure through HAProxy,

enabling seamless access to the underlying PostgreSQL clusters.

3. Pre-installation Prerequisites

3.1 Server and their Roles.

Network Ports Used by different components:

S#	Component Name	Ports
1	PostgreSQL	15432
2	Patroni	8010
3	Etcd	23790 & 23805
4	Pgbouncer	5005
5	HAProxy	5005

IP	Role
192.168.100.1	Primary server
192.168.100.2	Replica server
192.168.100.3	HA proxy server in Main Datacenter
192.166.100.4	Replica Server
192.166.100.5	HA Proxy Server in DR

3.2 Download software

Download PostgreSQL software using below links:

https://download.postgresql.org/pub/repos/yum/16/redhat/rhel-8-x86_64/postgresql16-16.6-1PGDG.rhel8.x86_64.rpm

https://download.postgresql.org/pub/repos/yum/16/redhat/rhel-8-x86_64/postgresql16-contrib-16.6-1PGDG.rhel8.x86_64.rpm

https://download.postgresql.org/pub/repos/yum/16/redhat/rhel-8-x86_64/postgresql16-libs-16.6-1PGDG.rhel8.x86_64.rpm

https://download.postgresql.org/pub/repos/yum/16/redhat/rhel-8-x86_64/postgresql16-server-16.6-1PGDG.rhel8.x86_64.rpm

Download Patroni Media using below links:

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/patroni-4.0.4-1PGDG.rhel8.noarch.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/patroni-etcd-4.0.4-1PGDG.rhel8.noarch.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/python3-etcd-0.4.5-45.rhel8.noarch.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/python3-cdiff-1.0-1.rhel8.noarch.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/python3-ydiff-1.2-10.rhel8.noarch.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/python3-psycopg2-2.9.5-3.rhel8.x86_64.rpm

https://download.postgresql.org/pub/repos/yum/common/redhat/rhel-8-x86_64/python3-psycopg3-3.2.3-1PGDG.rhel8.noarch.rpm

Download Etcd Media using below links:

https://download.postgresql.org/pub/repos/yum/common/pgdg-rhel8-extras/redhat/rhel-8.5-x86_64/etcd-3.5.17-1PGDG.rhel8.x86_64.rpm

Download HAProxy Media using below links:

https://download.postgresql.org/pub/repos/yum/common/pgdg-rhel8-extras/redhat/rhel-8.5-x86_64/haproxy-3.1.1-1PGDG.rhel8.x86_64.rpm

Download Pgbouncer using below link:

https://www.pgbouncer.org/downloads/files/1.24.0/pgbouncer-1.24.0.tar.gz

3.3 Modify the /etc/hosts file to include the hostnames and IP addresses of the nodes

Add the following changes on each node in hosts file. (vi /etc/hosts)

```
192.168.100.1   pg-primary-main.sm.net
192.168.100.2   pg-replica-main.sm.net
192.168.100.3   haproxy-main.sm.net
192.166.100.4   pg-replica-dr.sm.net
192.166.100.5   haproxy-dr.sm.net
```

4. Install PostgreSQL, Patroni, ETCD & Pgbouncer on Data nodes

In this section we will install PostgreSQL, Patroni, ETCD, and pgbouncer on all the three data nodes.

4.1 Install PostgreSQL

On Data nodes:

```
yum install postgresql16-16.6-1PGDG.rhel8.x86_64.rpm
yum install postgresql16-contrib-16.6-1PGDG.rhel8.x86_64.rpm
yum install postgresql16-libs-16.6-1PGDG.rhel8.x86_64.rpm
yum install postgresql16-server-16.6-1PGDG.rhel8.x86_64.rpm
```

4.2 Install Patroni

On Data nodes:

```
yum install python3-cdiff   >>> dependencies python3 3.6
yum install python3-click
yum install python3-prettytable
yum install python3-ydiff
yum install patroni-4.0.4-1PGDG.rhel8.noarch.rpm
yum install python3-etcd-0.4.5-45.rhel8.noarch.rpm
```

4.3 Install ETCD

On Data nodes:

```
yum install etcd-3.5.9-1.rhel8.x86_64.rpm
```

4.4 Install Pgbouncer

PgBouncer depends on few things to get compiled:

- GNU Make 3.81+
- Libevent 2.0+
- pkg-config
- OpenSSL 1.0.1+ for TLS support

When dependencies are installed just run:

```
$ ./configure --prefix=/usr/local
$ make
$ make install
```

4.5 Check the status of PostgreSQL, patroni, etcd

```
# systemctl status {postgresql-16.service,etcd,patroni}

● postgresql-16.service - PostgreSQL 16 database server
   Loaded: loaded (/usr/lib/systemd/system/postgresql-16.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://www.postgresql.org/docs/16/static/

● etcd.service - Etcd Server
   Loaded: loaded (/usr/lib/systemd/system/etcd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)

● patroni.service - Runners to orchestrate a high-availability PostgreSQL
   Loaded: loaded (/usr/lib/systemd/system/patroni.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
```

4.6 Disable the services

```
# systemctl disable {postgresql-16.service,etcd,patroni}
```

5. Configure etcd distributed store

5.1 Modify the configuration file

Node-1:

```
# cd /etc/etcd

# ls -l
total 4
-rw-r--r--. 1 root root 1568 Feb  9 07:06 etcd.conf

/*Change the settigns accordingly*/


# cat etcd.conf

# [member]
ETCD_NAME='node-1'
ETCD_DATA_DIR="/var/lib/etcd/etcd.new"
ETCD_LISTEN_PEER_URLS="http://192.168.100.1:23805,http://127.0.0.1:23805"
ETCD_LISTEN_CLIENT_URLS="http://192.168.100.1:23790,http://127.0.0.1:23790"

#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.100.1:23805"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e.
"test=http://..."
ETCD_INITIAL_CLUSTER="node-1=http://192.168.100.1:23805,node-2=http://192.168.100.2:23805,node-
3=http://192.166.100.4:23805"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="Postgres_HA_Cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.100.1:23790"
```

Node-2:

```
# cd /etc/etcd

# ls -l
total 4
-rw-r--r--. 1 root root 1568 Feb  9 07:06 etcd.conf


# [member]
ETCD_NAME='node-2'
ETCD_DATA_DIR="/var/lib/etcd/etcd.new"
ETCD_LISTEN_PEER_URLS="http://192.168.100.2:23805,http://127.0.0.1:23805"
ETCD_LISTEN_CLIENT_URLS="http://192.168.100.2:23790,http://127.0.0.1:23790"
```

```
#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.100.2:23805"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e. "test=http://..."
ETCD_INITIAL_CLUSTER="node-1=http://192.168.100.1:23805,node-2=http://192.168.100.2:23805,node-
3=http://192.166.100.4:23805"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="Postgres_HA_Cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.100.2:23790"
```

Node-3:

```
# cd /etc/etcd

# ls -l
total 4
-rw-r--r--. 1 root root 1568 Feb  9 07:06 etcd.conf

# [member]
ETCD_NAME='node-3'
ETCD_DATA_DIR="/var/lib/etcd/etcd.new"
ETCD_LISTEN_PEER_URLS="http://192.166.100.4:23805,http://127.0.0.1:23805"
ETCD_LISTEN_CLIENT_URLS="http://192.166.100.4:23790,http://127.0.0.1:23790"

#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.166.100.4:23805"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e. "test=http://..."
ETCD_INITIAL_CLUSTER="node-1=http://192.168.100.1:23805,node-2=http://192.168.100.2:23805,node-
3=http://192.166.100.4:23805"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="Postgres_HA_Cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.166.100.4:23790"
```

5.2 Enable and start the etcd service on all nodes:

```
systemctl enable --now etcd
systemctl start etcd
systemctl status etcd

/*Try starting all nodes at the same time for the etcd cluster to be created.
It may fail during startup try to start it again and double check the configuration file. */
```

5.3 Check the etcd cluster members and leader Node

ETCDCTL_API=3 etcdctl --endpoints=http://192.168.100.1:23790,http://192.168.100.2:23790,http://192.166.100.4:23790 -w table member list

ID	STATUS	NAME	PEER ADDRS	CLIENT ADDRS	IS LEARNER
4b2bfbcd0afc0ced	started	node-3	http://192.168.100.1:23805	http://192.168.100.1:23790	false
7d83e2fc470e6c72	started	node-2	http://192.168.100.2:23805	http://192.168.100.2:23790	false
e79754086f603f62	started	node-1	http://192.166.100.4:23805	http://192.166.100.4:23790	false

ETCDCTL_API=3 etcdctl --endpoints=http://192.168.100.1:23790,http://192.168.100.2:23790,http://192.166.100.4:23790 -w table member endpoint status

ENDPOINT	ID	VERSION	DB SIZE	IS LEADER	IS LEARNER	RAFT TERM	RAFT INDEX	RAFT APPLIED INDEX	ERRORS
192.168.100.1:23790	e79754086f603f62	3.5.9	20 kB	true	false	2	9	9	
192.168.100.2:23790	7d83e2fc470e6c72	3.5.9	20 kB	false	false	2	9	9	
192.166.100.4:23790	4b2bfbcd0afc0ced	3.5.9	20 kB	false	false	2	9	9	

6. Configure Patroni

6.1 Create the /etc/patroni/patroni.yml configuration file and add the following configuration for each Patroni Node

Node-1:

```
scope: pci-cluster
namespace: db
name: pg-primary-main.sm.net
restapi:
  listen: 192.168.100.1:8010
  connect_address: 192.168.100.1:8010
etcd3:
  hosts: 192.168.100.1:23790,192.168.100.2:23790,192.166.100.4:23790
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      shared_buffers: 4GB
```

```
        max_wal_size: 2GB
        min_wal_size: 1GB
        log_rotation_size: 100MB
        max_connections: 300
        wal_level: replica
        wal_log_hints: "on"
        hot_standby: "on"
        max_wal_senders: 10
        max_replication_slots: 10
        archive_mode: "on"
        archive_command: 'cp %p /pgarchive/16/archive/%f'
        log_rotation_size: 100MB
        log_destination: 'stderr'
        logging_collector: 'on'
        log_directory: '/pglog/postgres_log'
        log_filename: 'postgresql-%Y-%m-%d_%H%M%S.log'

    pg_hba:
      - host replication replicator 192.168.100.1/0 md5
      - host replication replicator 192.168.100.2/0 md5
      - host replication replicator 192.166.100.4/0 md5
      - host replication all 0.0.0.0/0 md5
      - host all all 0.0.0.0/0 md5
postgresql:
    listen: 0.0.0.0:15432
    connect_address: 192.168.100.1:15432
    data_dir: /pgdata/16
    bin_dir: /usr/pgsql-16/bin
    authentication:
      replication:
        username: replicator
        password: replicator
      superuser:
        username: postgres
        password: postgres
    parameters:
      unix_socket_directories: '/tmp/'
watchdog:
    mode: required
    device: /dev/watchdog
    safety_margin: 5
tags:
    failover_priority: 2
    noloadbalance: false
    clonefrom: false
    nosync: false
log:
    level: DEBUG
    dir: /pglog/patroni_log
    file_size: 100000000
    file_num: 10
```

```
scope: pci-cluster
namespace: db
name: pg-replica-main.sm.net

restapi:
  listen: 192.168.100.2:8010
  connect_address: 192.168.100.2:8010

etcd3:
  hosts: 192.168.100.1:23790,192.168.100.2:23790,192.166.100.4:23790

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      shared_buffers: 4GB
      max_wal_size: 2GB
      min_wal_size: 1GB
      log_rotation_size: 100MB
      max_connections: 300
      wal_level: replica
      wal_log_hints: "on"
      hot_standby: "on"
      max_wal_senders: 10
      max_replication_slots: 10
      archive_mode: "on"
      archive_command: 'cp %p /pgarchive/16/archive/%f'
      log_rotation_size: 100MB
      log_destination: 'stderr'
      logging_collector: 'on'
      log_directory: '/pglog/postgres_log'
      log_filename: 'postgresql-%Y-%m-%d_%H%M%S.log'
  pg_hba:
    - host replication replicator 192.168.100.1/0 md5
    - host replication replicator 192.168.100.2/0 md5
    - host replication replicator 192.166.100.4/0 md5
    - host replication all 0.0.0.0/0 md5
    - host all all 0.0.0.0/0 md5

postgresql:
  listen: 0.0.0.0:15432
  connect_address: 192.168.100.2:15432
  data_dir: /pgdata/16
  bin_dir: /usr/pgsql-16/bin
  authentication:
    replication:
      username: replicator
      password: replicator
```

```
    superuser:
      username: postgres
      password: postgres
  parameters:
    unix_socket_directories: '/tmp/'

watchdog:
  mode: required
  device: /dev/watchdog
  safety_margin: 5

tags:
  failover_priority: 2
  noloadbalance: false
  clonefrom: false
  nosync: false

log:
  level: DEBUG
  dir: /pglog/patroni_log
  file_size: 100000000
  file_num: 10
```

Node-3:

```
scope: pci-cluster
namespace: db
name: pg-replica-dr.sm.net
restapi:
  listen: 192.166.100.4:8010
  connect_address: 192.166.100.4:8010
etcd3:
  hosts: 192.168.100.1:23790,192.168.100.2:23790,192.166.100.4:23790
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      shared_buffers: 4GB
      max_wal_size: 2GB
      min_wal_size: 1GB
      log_rotation_size: 100MB
      max_connections: 300
      wal_level: replica
```

```
        wal_log_hints: "on"
        hot_standby: "on"
        max_wal_senders: 10
        max_replication_slots: 10
        archive_mode: "on"
        archive_command: 'cp %p /pgarchive/16/archive/%f'
        log_rotation_size: 100MB
        log_destination: 'stderr'
        logging_collector: 'on'
        log_directory: '/pglog/postgres_log'
        log_filename: 'postgresql-%Y-%m-%d_%H%M%S.log'

    pg_hba:
      - host replication replicator 192.168.100.1/0 md5
      - host replication replicator 192.168.100.2/0 md5
      - host replication replicator 192.166.100.4/0 md5
      - host replication all 0.0.0.0/0 md5
      - host all all 0.0.0.0/0 md5

postgresql:
    listen: 0.0.0.0:15432
    connect_address: 192.166.100.4:15432
    data_dir: /pgdata/16
    bin_dir: /usr/pgsql-16/bin
    authentication:
      replication:
        username: replicator
        password: replicator
      superuser:
        username: postgres
        password: postgres
    parameters:
      unix_socket_directories: '/tmp/'

watchdog:
    mode: required
    device: /dev/watchdog
    safety_margin: 5

tags:
    failover_priority: 2
    noloadbalance: false
    clonefrom: false
    nosync: false

log:
    level: DEBUG
    dir: /pglog/patroni_log
    file_size: 100000000
    file_num: 10
```

6.2 Check the patroni. service file if it is not created create it manually & Reload the systemd to be aware of the new service

```
sudo systemctl daemon-reload
```

Repeat the above steps on each node.

6.3 Start the Patroni service on each Node one by one:

Note:

Follow these steps to apply the commands:

- Start with node1 – Run the commands and wait for the service to become active.
- Proceed to the next node – Only after node1 is live, move to node2 and apply the commands.
- Verify synchronization – Ensure the new node syncs with the primary before proceeding.
- Repeat for remaining nodes – Continue this process one node at a time.

/*start the patroni service on node-1*/

```
sudo systemctl enable --now patroni
sudo systemctl restart patroni
sudo systemctl status patroni
```

6.4 Check for any errors of the patroni services:

```
sudo journalctl -u patroni. service --follow
```

6.5 Check the cluster status

```
patronictl -c /etc/patroni/patroni.yml list
```

```
+ Cluster: pci_cluster (7460502510827950504) ------+-----------+----+------------+
| Member                | Host           | Role    | State     | TL | Lag in MB |
+-----------------------+----------------+---------+-----------+----+-----------+
| pg-primary-main.sm.net | 192.168.100.1  | Leader  | running   |  1 |           |
| pg-replica-main.sm.net | 192.168.100.2  | Replica | streaming |  1 |         0 |
| pg-replica-dr.sm.net   | 192.166.100.4  | Replica | streaming |  1 |         0 |
+-----------------------+----------------+---------+-----------+----+-----------+
```

7. Configure the Pgbouncer

1. PgBouncer Configuration File (pgbouncer.ini)

```
[databases]

# To set the pool mode at the database level
# [Database Name] = host=[Host Name] port= 15432 auth_user=postgres pool_mode=session

pms_prod = host=pg-primary-main.sm.net port=15432 auth_user=postgres pool_mode=session

* = host=pg-primary-main.sm.net port=15432 auth_user=postgres

[pgbouncer]
logfile = /pglog/pgbouncer/pgbouncer.log
pidfile = /pghome/pgbouncer/pgbouncer.pid
listen_addr = 0.0.0.0
listen_port = 25432
auth_type = md5
auth_file = /pghome/pgbouncer/userlist.txt
auth_query = SELECT usename, passwd FROM pg_shadow WHERE usename=$1
admin_users = postgres
stats_users = postgres
pool_mode = transaction
ignore_startup_parameters = extra_float_digits
max_client_conn = 500
default_pool_size = 100
reserve_pool_size = 15
reserve_pool_timeout = 3
server_lifetime = 300
server_idle_timeout = 120
server_connect_timeout = 5
server_login_retry = 1
query_timeout = 300
query_wait_timeout = 60
client_idle_timeout = 90
client_login_timeout = 60
```

2. User Authentication Configuration (userlist.txt)

```
"postgres" "Admin#$321"
```

Follow the same steps on the other nodes as well and change the hostname accordingly.

3. Starting, Stopping & Restarting PgBouncer

8. Configure the HAProxy

1. Edit the haproxy.cfg file using root user.

```
global
   maxconn     2000
   user haproxy
   group haproxy
   daemon
defaults
   mode                tcp
   log                 global
   option              tcplog
   retries             3
   timeout queue         1m
   timeout connect       4s
   timeout client        60m
   timeout server        60m
   timeout check         5s
   maxconn             1000
listen stats
   mode http
   bind *:7000
   stats enable
   stats uri /

listen primary
   bind *:5005
   mode tcp
   option ssl-hello-chk  # Optional: Ensures SSL handshake is valid
   option httpchk OPTIONS /master
   http-check expect status 200
   default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
   server pg-primary-main.sm.net 192.168.100.1:25432 maxconn 700 check port 8010
   server pg-replica-main.sm.net 192.168.100.2:25432 maxconn 700 check port 8010
   server pg-replica-dr.sm.net   192.166.100.4:25432 maxconn 700 check port 8010
```

2. Stop & Start HAProxy.

```
sudo systemctl stop haproxy.service
sudo systemctl start haproxy.service
sudo systemctl status haproxy.service
```

3. Monitoring and Troubleshooting.

```
sudo journalctl -u haproxy.service -f
```

Similarly configure the other HAProxy Node.

9. Basic Administration and Troubleshooting Commands

In this section we will cover some basic commands for etcd, patroni & pgbouncer.

Basic Commands

S#	Commands	Description
	Etcd Commands	
01	ETCDCTL_API=3 etcdctl --write-out=table endpoint health	To check etcd cluster health
02	ETCDCTL_API=3 etcdctl --write-out=table endpoint status	To check cluster Status
03	ETCDCTL_API=3 etcdctl --write-out=table member list	List All Members
04	ETCDCTL_API=3 etcdctl --write-out=table alarm list	Check Alarm List (for issues like no space, corruption)
	Patroni Commands	
07	patronictl -c /etc/patroni/patroni.yml list	To check patroni cluster Status, leader, replication lag
08	patronictl -c /etc/patroni/patroni.yml show-config	Displays Patroni configuration (PostgreSQL parameters)
09	journalctl -u patroni -f	Check PostgreSQL Logs
10	patronictl -c /etc/patroni/patroni.yml switchover	Graceful Switchover
11	patronictl -c /etc/patroni/patroni.yml pause [cluster name]	Pause Auto-Failove (useful for maintenance)
12	patronictl -c /etc/patroni/patroni.yml resume [cluster name]	Resume Auto-Failover
13	patronictl -c /etc/patroni/patroni.yml reload	Reload Configuration: Applies configuration changes without a full restart.
	Pgbouncer Commands	
14	psql -p 25432 -U postgres -d pgbouncer	Basic Connection Test
15	SHOW HELP;	Lists All Admin Commands
16	SHOW DATABASES;	Configured databases
17	SHOW POOLS;	Connection pool stats
18	SHOW CLIENTS;	Active clients
19	SHOW SERVERS;	PostgreSQL backend connections
20	SHOW STATS;	Traffic statistics (queries, bytes)
21	pgbouncer -R -d /pghome/pgbouncer/pgbouncer.ini **OR** reload;	Reload Configuration (Without Restart)
22	SHOW CONFIG;	Runtime Configuration

References

https://www.cybertec-postgresql.com/en/introduction-and-how-to-etcd-clusters-for-patroni/
https://docs.percona.com/postgresql/16/solutions/ha-setup-yum.html#configure-patroni
https://etcd.io/docs/v3.5/install/
https://patroni.readthedocs.io/en/master/installation.html
https://www.pgbouncer.org/install.html
https://www.haproxy.com/downloads
https://www.haproxy.org/download/