Setup a Simple HA PostgreSQL Cluster Using Patroni, Etcd, HAProxy, and Keepalived

by Biondi Septian S

Objective

To simulate the building of a simple highly available PostgreSQL Patroni Cluster from scratch in a development environment.

Assumptions

- I don't care about the firewall in this demo environment, therefore I'm turning off the ufw service in this demo.
- I'm using default values of PostgreSQL Memory Parameters because the VMs that I'm using are small in terms of specification.
- I'm doing this demo in my VirtualBox, but I won't specify step by step related to the virtualbox management. I'm focusing only on the setup of the PostgreSQL Cluster.

Prerequisites

- 2 VMs for PostgreSQL and Patroni:
  bizdb01, bizdb02

- 2 VMs for HAProxy:
  haproxy01, haproxy02

- 3 Etcd nodes will be installed on shared VMs:
  bizdb01, bizdb02, haproxy01

- OS: Ubuntu 20.04

- PostgreSQL version: 14

- PostgreSQL cluster name: bizdb_cluster

- RAM:
  bizdb01: 4 GB
  bizdb02: 4 GB
  haproxy01: 4 GB

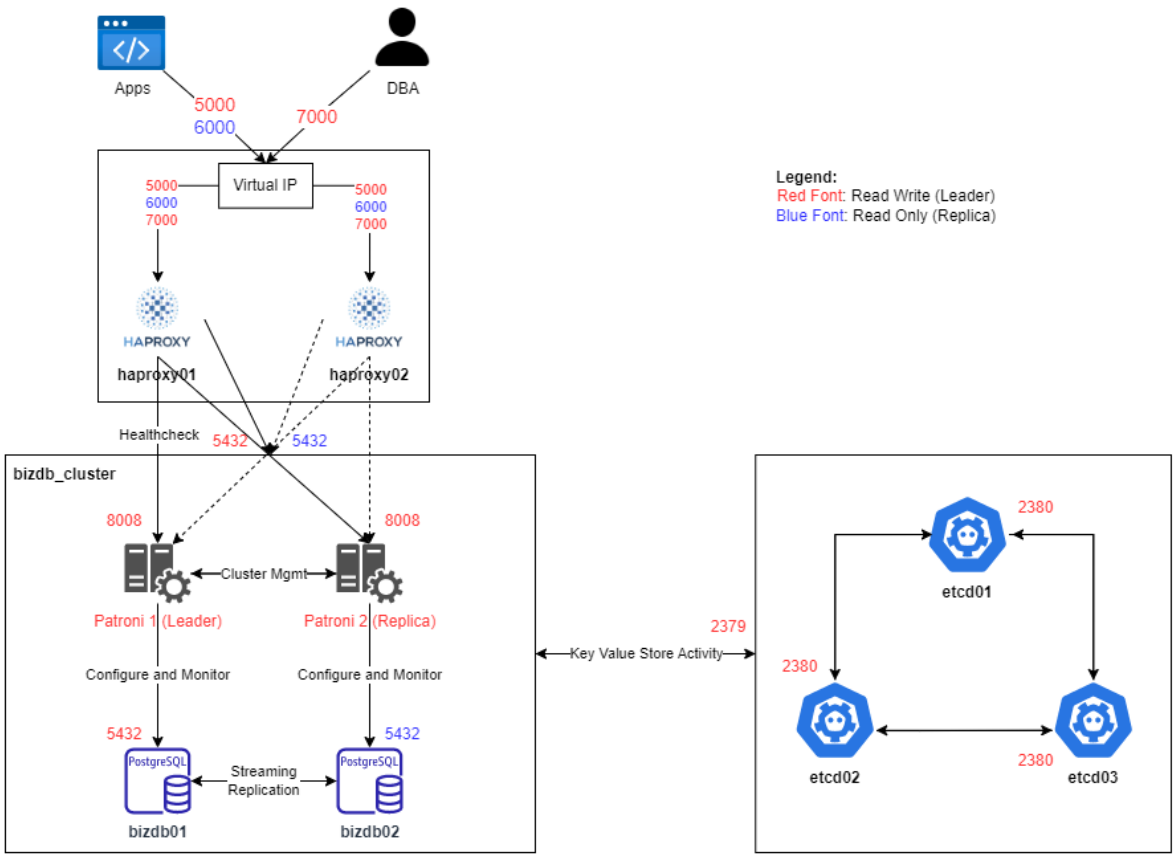
haproxy02: 2 GB

- CPU:
  bizdb01: 2 CPU
  bizdb02: 2 CPU
  haproxy01: 2 CPU
  haproxy02: 2 CPU

- Storage:
  bizdb01: 20 GB
  bizdb02: 20 GB
  haproxy01: 20 GB
  haproxy02: 15 GB

- IP Address:
  bizdb01 (PG Node 1): 192.168.150.100
  bizdb02 (PG Node 2): 192.168.150.101
  haproxy01 (HAProxy Node 1): 192.168.150.110
  haproxy02 (HAProxy Node 2): 192.168.150.111
  Virtual IP Address Reserved for HAProxy: 192.168.150.200

Architecture



Steps

1. Install Etcd on 3 Nodes (bizdb01, bizdb02, haproxy01)

Do Below Steps on 3 Nodes (bizdb01, bizdb02, haproxy01), Change the Hostname and IP Address according to the Hosts:

Install Required Tools

root@bizdb01:~# **sudo apt-get install wget curl -y**

Download etcd Binary From Github Using wget

root@bizdb01:~# **wget**
https://github.com/etcd-io/etcd/releases/download/v3.5.7/etcd-v3.5.7-linux-amd64.tar.gz
….

….
etcd-v3.5.7-linux-amd64.tar.gz
100%[===>]
17.60M 4.46MB/s   in 5.0s

2023-03-15 09:23:11 (3.55 MB/s) - ‘etcd-v3.5.7-linux-amd64.tar.gz’ saved [18458320/18458320]

Unzip the etcd Binary Files and Rename Directory to etcd

root@bizdb01:~# **tar xvf etcd-v3.5.7-linux-amd64.tar.gz**
root@bizdb01:~# **mv etcd-v3.5.7-linux-amd64 etcd**

Move All Binary Files Into /usr/local/bin Directory

root@bizdb01:~# **cd etcd/**
root@bizdb01:~/etcd# **sudo mv etcd etcdctl etcdutl /usr/local/bin/**

Check etcd Binaries Version

root@bizdb01:~/etcd# **etcd --version**
etcd Version: 3.5.7
Git SHA: 215b53cf3
Go Version: go1.17.13
Go OS/Arch: linux/amd64

root@bizdb01:~/etcd# **etcdctl version**
etcdctl version: 3.5.7
API version: 3.5

root@bizdb01:~/etcd# **etcdutl version**
etcdutl version: 3.5.7
API version: 3.5

Setup Etcd Service

Edit /etc/hosts to Give Alias For Each Etcd Cluster Members

root@bizdb01:~# **nano /etc/hosts**
192.168.150.100 etcd01
192.168.150.101 etcd02
192.168.150.110 etcd03

Create Group and User for Etcd

root@bizdb01:~# **sudo groupadd --system etcd**

root@bizdb01:~# **sudo useradd -s /sbin/nologin --system -g etcd etcd**

Create Data Directory and Config Directory

root@bizdb01:~# **sudo mkdir -p /var/lib/etcd/**
root@bizdb01:~# **sudo mkdir /etc/etcd/**

Change Ownership and Permission Of /var/lib/etcd Directory To Etcd User

root@bizdb01:~# **sudo chown -R etcd:etcd /var/lib/etcd/**
root@bizdb01:~# **sudo chmod -R 700 /var/lib/etcd/**

Identify Primary Network Interface To Run Etcd

root@bizdb01:~# **ip ad**
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:dd:a2:da brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 78324sec preferred_lft 78324sec
    inet6 fe80::a00:27ff:fedd:a2da/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:39:88:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.100/24 brd 192.168.150.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe39:8862/64 scope link
       valid_lft forever preferred_lft forever

Load and Check Some Environment Variables For Used in Service Daemon

root@bizdb01:~# **NET_NAME="enp0s8"**
root@bizdb01:~# **ETCD_HOST_IP=$(ip addr show $NET_NAME | grep "inet\b" | awk '{print $2}' | cut -d/ -f1)**
root@bizdb01:~# **ETCD_NAME=$(cat /etc/hosts | grep "$(ip addr show $NET_NAME | grep "inet\b" | awk '{print $2}' | cut -d/ -f1)" | awk '{print $2}')**
root@bizdb01:~# **echo $NET_NAME**
enp0s8
root@bizdb01:~# **echo $ETCD_HOST_IP**

192.168.150.100
root@bizdb01:~# **echo $ETCD_NAME**
etcd01

#NET_NAME is used to store the name of the primary network interface of etcd server
#ETCD_HOST_IP is used to store the IP of the host of the Etcd member in the clusters.
#ETCD_NAME is used to store the unique name of the Etcd member by using hostname.

Create Etcd Unit File in Systemd To Manage Etcd Service Using Following Configuration

root@bizdb01:~# **cat <<EOF | sudo tee /etc/systemd/system/etcd.service**
[Unit]
Description=etcd - highly-available key value store
Documentation=https://etcd.io/docs
Documentation=man:etcd

[Service]
Type=notify
User=etcd
**ExecStart=/usr/local/bin/etcd **
** --name ${ETCD_NAME} **
** --data-dir=/var/lib/etcd **
** --initial-advertise-peer-urls http://${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 **
** --initial-cluster etcd01=http://etcd01:2380,etcd02=http://etcd02:2380,etcd03=http://etcd03:2380**
****
** --initial-cluster-state new **
** --enable-v2=true **

[Install]
WantedBy=multi-user.target
EOF

Reload Daemon and Enable The Service To Start On Boot

root@bizdb01:~# **sudo systemctl daemon-reload**
root@bizdb01:~# **sudo systemctl enable etcd.service**
Created symlink /etc/systemd/system/multi-user.target.wants/etcd.service →
/etc/systemd/system/etcd.service.

Start The Etcd Service on 3 Nodes

root@bizdb01:~# **sudo systemctl start etcd**

Check Status of The Etcd Services

```
root@bizdb01:~# sudo systemctl status etcd
● etcd.service - etcd - highly-available key value store
     Loaded: loaded (/etc/systemd/system/etcd.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-03-17 11:21:07 UTC; 15min ago
       Docs: https://etcd.io/docs
             man:etcd
   Main PID: 3711 (etcd)
      Tasks: 8 (limit: 4612)
     Memory: 16.1M
     CGroup: /system.slice/etcd.service
             └─3711 /usr/local/bin/etcd --name etcd01 --data-dir=/var/lib/etcd --initial-advertise-peer-urls
http://192.168.150.100:2380

root@bizdb01:~# etcdctl endpoint status --write-out=table
--endpoints=http://etcd01:2379,http://etcd02:2379,http://etcd03:2379
+-------------------+------------------+---------+---------+-----------+------------+-----------+------------+--------------------
-+--------+
|     ENDPOINT      |        ID        | VERSION | DB SIZE | IS LEADER | IS LEARNER | RAFT TERM |
RAFT INDEX | RAFT APPLIED INDEX | ERRORS |
+-------------------+------------------+---------+---------+-----------+------------+-----------+------------+--------------------
-+--------+
| http://etcd01:2379 | b8b747c74aaea686 |   3.5.7 |   20 kB |      true |      false |         7 |         26 |
26 |        |
| http://etcd02:2379 |  b3504381e8ba3cb |   3.5.7 |   20 kB |     false |      false |         7 |         26 |
26 |        |
| http://etcd03:2379 | f572fdfc5cb68406 |   3.5.7 |   20 kB |     false |      false |         7 |         26 |
26 |        |
+-------------------+------------------+---------+---------+-----------+------------+-----------+------------+--------------------
-+--------+
```

2. Install PostgreSQL 14 on 2 Nodes (bizdb01, bizdb02)

Install GNUPG

```
root@bizdb01:~# sudo apt install gnupg2
```

Add PostgreSQL 14 Repository for Ubuntu 20.04 Release

root@bizdb01:~# **sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'**

Import GPG Signing Key For The Repository

root@bizdb01:~# **wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -**
OK

Update APT Package List

root@bizdb01:~# **sudo apt update**
Hit:1 http://id.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://id.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://id.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://id.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://id.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2,425 kB]
Get:6 http://id.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1,038 kB]
Get:7 http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease [91.6 kB]
Get:8 http://apt.postgresql.org/pub/repos/apt focal-pgdg/main amd64 Packages [258 kB]
Fetched 4,149 kB in 3s (1,579 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
28 packages can be upgraded. Run 'apt list --upgradable' to see them.

Install PostgreSQL 14 Package

root@bizdb01:~# **sudo apt -y install postgresql-14**
….
….
Creating config file /etc/postgresql-common/createcluster.conf with new version
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
'/etc/apt/trusted.gpg.d/apt.postgresql.org.gpg' ->
'/usr/share/postgresql-common/pgdg/apt.postgresql.org.gpg'
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service →
/lib/systemd/system/postgresql.service.
Setting up postgresql-14 (14.7-1.pgdg20.04+1) ...
Creating new PostgreSQL cluster 14/main ...
/usr/lib/postgresql/14/bin/initdb -D /var/lib/postgresql/14/main --auth-local peer --auth-host scram-sha-256 --no-instructions
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/14/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Asia/Jakarta
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
update-alternatives: using /usr/share/postgresql/14/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Processing triggers for systemd (245.4-4ubuntu3.20) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) …

Drop Existing main Cluster From PostgreSQL Default Installation

We don’t need this built-in cluster from installation, because later Patroni will take care of new database cluster initialization
root@bizdb01:~# **pg_dropcluster --stop 14 main**

Edit PostgreSQL Default Config File, Change listen_addresses

root@bizdb01:~# **cd /etc/postgresql/14/main**
root@bizdb01:/etc/postgresql/14/main# **nano postgresql.conf**
- Connection Settings -

listen_addresses = '*'

root@bizdb01:/etc/postgresql/14/main# **sudo systemctl stop postgresql**
root@bizdb01:/etc/postgresql/14/main# **sudo systemctl start postgresql**

3. Setup Patroni on 2 Postgresql Nodes (bizdb01, bizdb02)

Install Patroni

root@bizdb01:~# **curl https://bootstrap.pypa.io/pip/3.6/get-pip.py -o /tmp/get-pip.py -k**

```
root@bizdb01:~# python3 /tmp/get-pip.py
Collecting pip<22.0
  Downloading pip-21.3.1-py3-none-any.whl (1.7 MB)
     |████████████████████████████████| 1.7 MB 1.6 MB/s
Collecting wheel
  Downloading wheel-0.40.0-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 1.1 MB/s
Installing collected packages: wheel, pip
Successfully installed pip-21.3.1 wheel-0.40.0

root@bizdb01:~# pip install psycopg2-binary
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(3.0 MB)
     |████████████████████████████████| 3.0 MB 1.5 MB/s
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.5

root@bizdb01:~# pip install patroni[etcd,consul]
Collecting patroni[consul,etcd]
  Downloading patroni-3.0.1-py3-none-any.whl (214 kB)
     |████████████████████████████████| 214 kB 251 kB/s
Collecting ydiff>=1.2.0
  Downloading ydiff-1.2.tar.gz (42 kB)
     |████████████████████████████████| 42 kB 372 kB/s
  Preparing metadata (setup.py) ... done
Requirement already satisfied: click>=4.1 in /usr/lib/python3/dist-packages (from patroni[consul,etcd])
(7.0)
Collecting prettytable>=0.7
  Downloading prettytable-3.6.0-py3-none-any.whl (27 kB)
Requirement already satisfied: six>=1.7 in /usr/lib/python3/dist-packages (from patroni[consul,etcd])
(1.14.0)
Requirement already satisfied: urllib3!=1.21,>=1.19.1 in /usr/lib/python3/dist-packages (from
patroni[consul,etcd]) (1.25.8)
Requirement already satisfied: PyYAML in /usr/lib/python3/dist-packages (from patroni[consul,etcd])
(5.3.1)
Collecting psutil>=2.0.0
  Downloading
psutil-5.9.4-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl (280 kB)
     |████████████████████████████████| 280 kB 5.1 MB/s
Collecting python-dateutil
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 2.0 MB/s
```

```
Collecting python-consul>=0.7.1
  Downloading python_consul-1.1.0-py2.py3-none-any.whl (24 kB)
Collecting python-etcd<0.5,>=0.4.3
  Downloading python-etcd-0.4.5.tar.gz (37 kB)
  Preparing metadata (setup.py) ... done
Collecting wcwidth
  Downloading wcwidth-0.2.6-py2.py3-none-any.whl (29 kB)
Requirement already satisfied: requests>=2.0 in /usr/lib/python3/dist-packages (from
python-consul>=0.7.1->patroni[consul,etcd]) (2.22.0)
Collecting dnspython>=1.13.0
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
     |████████████████████████████████| 283 kB 7.6 MB/s
Building wheels for collected packages: python-etcd, ydiff
  Building wheel for python-etcd (setup.py) ... done
  Created wheel for python-etcd: filename=python_etcd-0.4.5-py3-none-any.whl size=38501
sha256=a253b2a3321b305b1834f77ad0af292b630da9e25adc551f697009567fca294e
  Stored in directory:
/root/.cache/pip/wheels/10/fa/7a/8ee59bc1789d1a1efffb172d6efbd0d24c5b85ec7ddce32ee4
  Building wheel for ydiff (setup.py) ... done
  Created wheel for ydiff: filename=ydiff-1.2-py3-none-any.whl size=16631
sha256=c7302c15bffdef40241f8e26f9452bbd84d7ab9ff5e4507d228c3fa42f5e8131
  Stored in directory:
/root/.cache/pip/wheels/3b/64/6b/1fffe46e533173749dd54bd9cf053291ae9d2bcc53dffba1ff
Successfully built python-etcd ydiff
Installing collected packages: wcwidth, ydiff, python-dateutil, psutil, prettytable, dnspython, python-etcd,
python-consul, patroni
Successfully installed dnspython-2.3.0 patroni-3.0.1 prettytable-3.6.0 psutil-5.9.4 python-consul-1.1.0
python-dateutil-2.8.2 python-etcd-0.4.5 wcwidth-0.2.6 ydiff-1.2
```

Setup SSL Certificates for PostgreSQL On 2 Nodes (bizdb01, bizdb02)

Create Directory For Storing SSL Certificates on Both Nodes

root@bizdb01:~# **mkdir -p /usr/patroni/conf**
root@bizdb02:~# **mkdir -p /usr/patroni/conf**

Generate Public/Private Key File Pair (**Only On Node 1 And Then Copy Into Node 2**)

root@bizdb01:~# **openssl genrsa -out server.key 2048**
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@bizdb01:~# **openssl req -new -x509 -days 3650 -key server.key -out server.crt -subj "/C=ID/CN=192.168.150.200"**

Note: /CN filled with Shared Virtual IP address of the HAProxy we created before.

Copy Cert and Key To Directory Created Before (/usr/patroni/conf) On Node 1

root@bizdb01:~# **cp server.crt /usr/patroni/conf/server.crt**
root@bizdb01:~# **cp server.key /usr/patroni/conf/server.key**

Change Cert and Key Files Mode and Owner On Node 1

root@bizdb01:~# **cd /usr/patroni/conf**
root@bizdb01:/usr/patroni/conf# **chmod 400 server.crt**
root@bizdb01:/usr/patroni/conf# **chmod 400 server.key**
root@bizdb01:/usr/patroni/conf# **chown postgres:postgres server.key**
root@bizdb01:/usr/patroni/conf# **chown postgres:postgres server.crt**

Setup Passwordless SSH Connection Between bizdb01 And bizdb02

Edit Etc Hosts File on Both Nodes To Include bizdb01 and bizdb02:

root@bizdb01:~# **nano /etc/hosts**
….
….
192.168.150.100 etcd01 **bizdb01**
192.168.150.101 etcd02 **bizdb02**

root@bizdb02:~# **nano /etc/hosts**
….
….
192.168.150.100 etcd01 **bizdb01**
192.168.150.101 etcd02 **bizdb02**

Change Postgres OS User Password on Both Nodes:

root@bizdb01:~# **passwd postgres**
New password:
Retype new password:
passwd: password updated successfully

root@bizdb02:~# **passwd postgres**
New password:
Retype new password:
passwd: password updated successfully

Generate Public/Private Key Pair Using Postgres User on Both Nodes:

```
root@bizdb01:~# sudo -i -u postgres
postgres@bizdb01:~$ ssh-keygen -t rsa -N ''
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/postgresql/.ssh/id_rsa):
Created directory '/var/lib/postgresql/.ssh'.
Your identification has been saved in /var/lib/postgresql/.ssh/id_rsa
Your public key has been saved in /var/lib/postgresql/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dpOBn0ZdxsM7l1Vbr5NJg3Jg3QTkHF7kp86nyEMkKmU postgres@bizdb01
The key's randomart image is:
+---[RSA 3072]----+
|          oo+B* +|
|         o *.B= =|
|        . + B ++=|
|        Eo.*..oO.|
|       oS.Bo  *o |
|      ...o ..o . |
|       .   .  o .|
|           ... o |
|            o..  |
+----[SHA256]-----+


root@bizdb02:~# sudo -i -u postgres
postgres@bizdb02:~$ ssh-keygen -t rsa -N ''
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/postgresql/.ssh/id_rsa):
Created directory '/var/lib/postgresql/.ssh'.
Your identification has been saved in /var/lib/postgresql/.ssh/id_rsa
Your public key has been saved in /var/lib/postgresql/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:VVtzX5tOFPkCm+l6g4iXgQTYGAON6k35HpRqcpDz4ok postgres@bizdb02
The key's randomart image is:
+---[RSA 3072]----+
|**           . oo=|
|=.o         ..o.+=|
|... . .    . .+ +o|
|.+.o o    .. o + .|
|..*.+   S. .   o |
| +.*.o    .      |
|o *..+.. .       |
|Eo. +.. +        |
|   .     o       |
```

+----[SHA256]-----+

Copy the Public Key From bizdb01 to bizdb02:

postgres@bizdb01:~$ **ssh-copy-id postgres@bizdb02**
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/postgresql/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
postgres@bizdb02's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'postgres@bizdb02'"
and check to make sure that only the key(s) you wanted were added.

Copy the Public Key From bizdb02 to bizdb01:

postgres@bizdb02:~$ **ssh-copy-id postgres@bizdb01**
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/postgresql/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
postgres@bizdb01's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'postgres@bizdb01'"
and check to make sure that only the key(s) you wanted were added.

Test Connect From Both Nodes to Both Nodes:

postgres@bizdb01:~$ **ssh postgres@bizdb02**
postgres@bizdb02:~$ **ssh postgres@bizdb01**

Copy Server.crt and Server.key To Node 2

Create Temporary Directory On Node 2 To Store Cert And Key:

root@bizdb02:/usr/patroni# **sudo -i -u postgres**
postgres@bizdb02:~$ **pwd**
/var/lib/postgresql
postgres@bizdb02:~$ **mkdir tempcrt**

SCP Server.crt and Server.key From Node 1 To Temporary Directory On Node 2:

```
root@bizdb01:~# sudo -i -u postgres
postgres@bizdb01:~$ scp -p /usr/patroni/conf/server.crt
postgres@bizdb02:/var/lib/postgresql/tempcrt/server.crt
server.crt                                    100% 1164   439.7KB/s
00:00
postgres@bizdb01:~$ scp -p /usr/patroni/conf/server.key
postgres@bizdb02:/var/lib/postgresql/tempcrt/server.key
server.key                                    100% 1675   233.1KB/s
00:00
```

On Node 2 As Root User, Copy Server.crt and Server.key To /usr/patroni/conf Directory:

```
root@bizdb02:~# cd /var/lib/postgresql/tempcrt/
root@bizdb02:/var/lib/postgresql/tempcrt# ls -ltr
total 8
-r-------- 1 postgres postgres 1164 Mar 19 11:35 server.crt
-r-------- 1 postgres postgres 1675 Mar 19 11:35 server.key
root@bizdb02:/var/lib/postgresql/tempcrt# cp -p server.crt /usr/patroni/conf/
root@bizdb02:/var/lib/postgresql/tempcrt# cp -p server.key /usr/patroni/conf/
```

Create Config File for Patroni (YAML)

Create Configuration File By Touch

```
root@bizdb01:~# touch /usr/patroni/conf/postgresql.yml
```

Prepare The YAML Config On Node 1 (bizdb01), The Green Highlight is Unique Value on Each Node:

```
scope: postgres
namespace: /bizdb_cluster/
name: bizdb01
restapi:
  listen: 192.168.150.100:8008
  connect_address: 192.168.150.100:8008
etcd:
  hosts: 192.168.150.100:2379, 192.168.150.101:2379, 192.168.150.110:2379
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
```

```
    maximum_lag_on_failover: 1048576
    maximum_lag_on_syncnode: 15000000
    synchronous_mode: false
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        shared_buffers: 128MB
        work_mem: 4MB
        maintenance_work_mem: 64MB
        max_worker_processes: 8
        wal_buffers: 4MB
        max_wal_size: 1GB
        min_wal_size: 80MB
        effective_cache_size: 4GB
        fsync: on
        checkpoint_completion_target: 0.9
        log_rotation_size: 10MB
        listen_addresses: "*"
        max_connections: 100
        temp_buffers: 4MB
        ssl: true
        ssl_cert_file: /usr/patroni/conf/server.crt
        ssl_key_file: /usr/patroni/conf/server.key
  initdb:
    - encoding: UTF8
    - data-checksums
  pg_hba:
    - host replication replicator 127.0.0.1/32 md5
    - host replication replicator 192.168.150.100/32 md5
    - host replication replicator 192.168.150.101/32 md5
    - host all all 0.0.0.0/0 md5
users:
  admin:
    password: admin
    options:
      - createrole
      - createdb

postgresql:
  listen: 192.168.150.100:5432
  connect_address: 192.168.150.100:5432
  data_dir: /var/lib/postgresql/14/bizdb_cluster
  bin_dir: /usr/lib/postgresql/14/bin
```

```
  pgpass: /tmp/pgpass
  authentication:
    replication:
      username: replicator
      password: replicator
    superuser:
      username: postgres
      password: postgres
    rewind:
      username: pgrewind
      password: pgrewind
tags:
  nofailover: false
  noloadbalance: false
  clonefrom: false
  nosync: true
```

Prepare The YAML File On Node 2 (bizdb02), The Green Highlight is Unique Value on Each Node:

```
scope: postgres
namespace: /bizdb_cluster/
name: bizdb02
restapi:
  listen: 192.168.150.101:8008
  connect_address: 192.168.150.101:8008
etcd:
  hosts: 192.168.150.100:2379, 192.168.150.101:2379, 192.168.150.110:2379
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    maximum_lag_on_syncnode: 15000000
    synchronous_mode: false
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        shared_buffers: 128MB
        work_mem: 4MB
        maintenance_work_mem: 64MB
        max_worker_processes: 8
        wal_buffers: 4MB
```

```
      max_wal_size: 1GB
      min_wal_size: 80MB
      effective_cache_size: 4GB
      fsync: on
      checkpoint_completion_target: 0.9
      log_rotation_size: 10MB
      listen_addresses: "*"
      max_connections: 100
      temp_buffers: 4MB
      ssl: true
      ssl_cert_file: /usr/patroni/conf/server.crt
      ssl_key_file: /usr/patroni/conf/server.key
  initdb:
    - encoding: UTF8
    - data-checksums
  pg_hba:
    - host replication replicator 127.0.0.1/32 md5
    - host replication replicator 192.168.150.100/32 md5
    - host replication replicator 192.168.150.101/32 md5
    - host all all 0.0.0.0/0 md5
users:
  admin:
    password: admin
    options:
      - createrole
      - createdb

postgresql:
  listen: 192.168.150.101:5432
  connect_address: 192.168.150.101:5432
  data_dir: /var/lib/postgresql/14/bizdb_cluster
  bin_dir: /usr/lib/postgresql/14/bin
  pgpass: /tmp/pgpass
  authentication:
    replication:
      username: replicator
      password: replicator
    superuser:
      username: postgres
      password: postgres
    rewind:
      username: pgrewind
      password: pgrewind
tags:
```

nofailover: false
noloadbalance: false
clonefrom: false
nosync: true

Edit Configuration File On Both Node 1 and Node 2 (bizdb01, bizdb02) Using Nano and Fill Each With YAML Config Node 1 and Node 2 Above:

root@bizdb01:~# **nano /usr/patroni/conf/postgresql.yml**
root@bizdb02:~# **nano /usr/patroni/conf/postgresql.yml**

Create Patroni Service

root@bizdb01:~# **nano /usr/lib/systemd/system/patroni.service**

[Unit]
Description=patroni
Documentation=https://patroni.readthedocs.io/en/latest/index.html
After=syslog.target network.target etcd.target
Wants=network-online.target

[Service]
Type=simple
User=postgres
Group=postgres
PermissionsStartOnly=true
ExecStart=/usr/local/bin/patroni /usr/patroni/conf/postgresql.yml
ExecReload=/bin/kill -HUP $MAINPID
LimitNOFILE=65536
KillMode=process
KillSignal=SIGINT
Restart=on-abnormal
RestartSec=30s
TimeoutSec=0

[Install]
WantedBy=multi-user.target

Start Patroni on Both Nodes (bizdb01, bizdb02)

root@bizdb01:~# **sudo systemctl daemon-reload**
root@bizdb01:~# **sudo systemctl start patroni**
root@bizdb01:~# **sudo systemctl status patroni**
● patroni.service - patroni
Loaded: loaded (/lib/systemd/system/patroni.service; disabled; vendor preset: enabled)

```
     Active: active (running) since Sun 2023-03-19 14:54:12 WIB; 2min 59s ago
       Docs: https://patroni.readthedocs.io/en/latest/index.html
   Main PID: 4301 (patroni)
      Tasks: 14 (limit: 4609)
     Memory: 136.4M
     CGroup: /system.slice/patroni.service
             ├─4301 /usr/bin/python3 /usr/local/bin/patroni /usr/patroni/conf/postgresql.yml
             ├─4342 /usr/lib/postgresql/14/bin/postgres -D /var/lib/postgresql/14/bizdb_cluster
--config-file=/var/lib/postgresql/14/bizdb_cluster>
             ├─4346 postgres: postgres: checkpointer
             ├─4347 postgres: postgres: background writer
             ├─4348 postgres: postgres: walwriter
             ├─4349 postgres: postgres: autovacuum launcher
             ├─4350 postgres: postgres: stats collector
             ├─4351 postgres: postgres: logical replication launcher
             ├─4354 postgres: postgres: postgres postgres 192.168.150.100(49336) idle
             └─4365 postgres: postgres: walsender replicator 192.168.150.101(48170) streaming
0/3000060

Mar 19 14:55:34 bizdb01 patroni[4301]: 2023-03-19 14:55:34,383 INFO: no action. I am (bizdb01), the
leader with the lock
```

```
root@bizdb02:~# sudo systemctl daemon-reload
root@bizdb02:~# sudo systemctl start patroni
root@bizdb02:~# sudo systemctl enable patroni
root@bizdb02:~# sudo systemctl status patroni
● patroni.service - patroni
     Loaded: loaded (/lib/systemd/system/patroni.service; disabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-03-19 14:54:30 WIB; 3min 26s ago
       Docs: https://patroni.readthedocs.io/en/latest/index.html
   Main PID: 3190 (patroni)
      Tasks: 12 (limit: 4609)
     Memory: 108.6M
     CGroup: /system.slice/patroni.service
             ├─3190 /usr/bin/python3 /usr/local/bin/patroni /usr/patroni/conf/postgresql.yml
             ├─3213 /usr/lib/postgresql/14/bin/postgres -D /var/lib/postgresql/14/bizdb_cluster
--config-file=/var/lib/postgresql/14/bizdb_cluster>
             ├─3215 postgres: postgres: startup recovering 000000010000000000000003
             ├─3219 postgres: postgres: checkpointer
             ├─3220 postgres: postgres: background writer
             ├─3221 postgres: postgres: stats collector
             ├─3227 postgres: postgres: postgres postgres 192.168.150.101(54304) idle
             └─3232 postgres: postgres: walreceiver streaming 0/3000060
```

Mar 19 14:56:24 bizdb02 patroni[3190]: 2023-03-19 14:56:24,391 INFO: no action. I am (bizdb02), a secondary, and following a leader (bizdb01)

4. Install HAProxy on 2 Nodes (haproxy01, haproxy02)

Enable PPA

root@haproxy01:~# **sudo apt-get install --no-install-recommends software-properties-common**
root@haproxy01:~# **sudo add-apt-repository ppa:vbernat/haproxy-2.6**
 HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web sites crawling under very high loads while needing persistence or Layer7 processing. Supporting tens of thousands of connections is clearly realistic with todays hardware. Its mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the Net.

This PPA contains packages for HAProxy 2.6.
 More info: https://launchpad.net/~vbernat/+archive/ubuntu/haproxy-2.6
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Install HAProxy

root@haproxy01:~# **sudo apt-get install haproxy=2.6.***
Reading package lists... Done
Building dependency tree
Reading state information... Done
Selected version '2.6.11-1ppa1~focal' (HAProxy 2.6:20.04/focal [amd64]) for 'haproxy'
The following additional packages will be installed:
  liblua5.3-0
Suggested packages:
  vim-haproxy haproxy-doc
The following NEW packages will be installed:
  haproxy liblua5.3-0
0 upgraded, 2 newly installed, 0 to remove and 26 not upgraded.
Need to get 1,806 kB of archives.
After this operation, 4,480 kB of additional disk space will be used.
Do you want to continue? [Y/n] **y**

Check HAproxy Version

root@haproxy01:~# **sudo haproxy -v**
HAProxy version 2.6.11-1ppa1~focal 2023/03/18 - https://haproxy.org/
Status: long-term supported branch - will stop receiving fixes around Q2 2027.
Known bugs: http://www.haproxy.org/bugs/bugs-2.6.11.html

Running on: Linux 5.4.0-144-generic #161-Ubuntu SMP Fri Feb 3 14:49:04 UTC 2023 x86_64

Backup Existing Config File of HAProxy

root@haproxy01:~# **cp -p /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak**

Replace The Content of HAProxy Config File With Below

oot@haproxy01:~# **cat /dev/null > /etc/haproxy/haproxy.cfg**
root@haproxy01:~# **nano /etc/haproxy/haproxy.cfg**

```
global
  log 127.0.0.1 local2
  chroot /var/lib/haproxy
  pidfile /var/run/haproxy.pid
  maxconn 6000
  user haproxy
  group haproxy
  daemon
  stats socket /var/lib/haproxy/stats
defaults
  mode tcp
  log global
  retries 3
  timeout queue 1m
  timeout connect 10s
  timeout client 31m
  timeout server 31m
  timeout check 10s
  maxconn 3000
listen stats
  mode http
  bind *:7000
  stats enable
  stats uri /
listen postgres
  bind *:5000
  option httpchk
  http-check expect status 200
  default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
  server bizdb01 192.168.150.100:5432 maxconn 100 check port 8008
  server bizdb02 192.168.150.101:5432 maxconn 100 check port 8008
listen postgres-readonly
  bind *:6000
  option httpchk GET /replica
```

http-check expect status 200
default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
server bizdb01 192.168.150.100:5432 maxconn 100 check port 8008
server bizdb02 192.168.150.101:5432 maxconn 100 check port 8008

Edit HAProxy Service Script and Add LimitNOFILE Param

root@haproxy01:~# **nano /usr/lib/systemd/system/haproxy.service**
….
….
[Service]
LimitNOFILE=6000

Reload Daemon and Restart HAProxy Service

root@haproxy01:~# sudo systemctl daemon-reload
root@haproxy01:~# sudo systemctl stop haproxy
root@haproxy01:~# sudo systemctl start haproxy

5. Test Connect To PostgreSQL Cluster Through HAProxy

Using Psql (Connect Without SSL)

root@bizdb01:~# **sudo -i -u postgres**
postgres@bizdb01:~$ **psql -h 192.168.150.110 -p 5000 -d postgres -U postgres**
Password for user postgres:
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \q
postgres@bizdb01:~$ **psql -h 192.168.150.111 -p 5000 -d postgres -U postgres**
Password for user postgres:
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \q

Using Psql (Connect With SSL)

postgres@bizdb01:~$ **psql "host=192.168.150.110 port=5000 dbname=postgres user=postgres password=postgres sslmode=verify-ca sslrootcert=/usr/patroni/conf/server.crt"**
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)

```
Type "help" for help.

postgres=# \q
postgres@bizdb01:~$ psql "host=192.168.150.111 port=5000 dbname=postgres user=postgres
password=postgres sslmode=verify-ca sslrootcert=/usr/patroni/conf/server.crt"
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \q
```

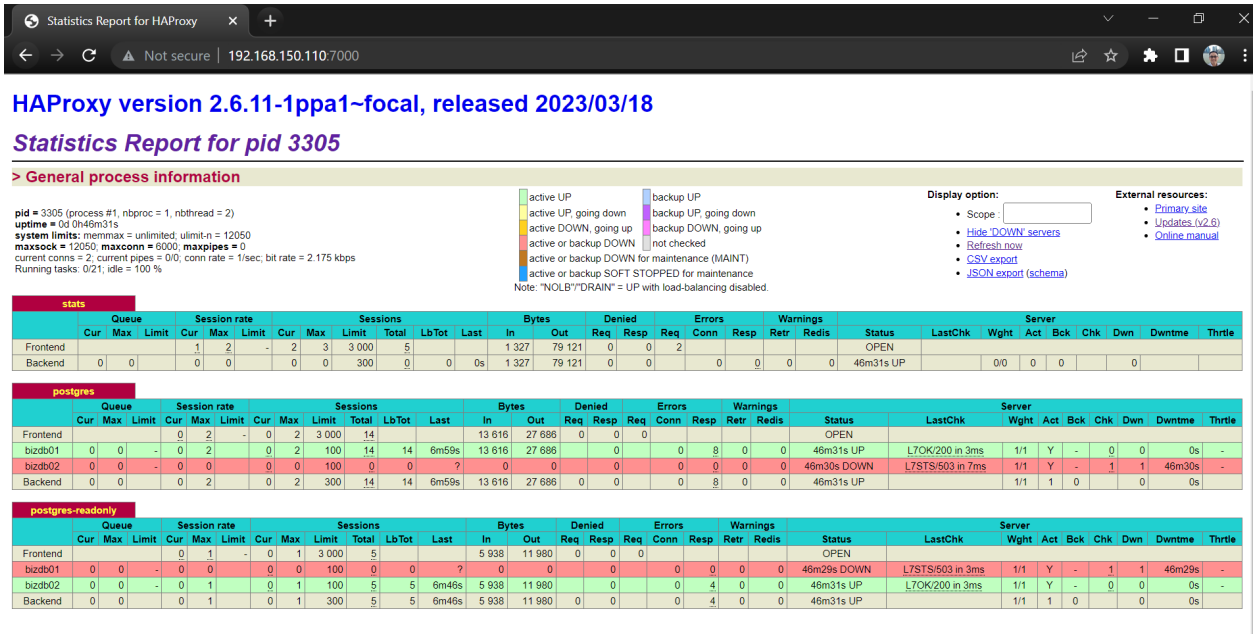
Using Psql, Check Read-Write or Read-Only Connection on Different Ports

```
postgres@bizdb01:~$ psql -h 192.168.150.110 -U postgres -p 5000 -c "SELECT
pg_is_in_recovery();"
Password for user postgres:
 pg_is_in_recovery
-------------------
 f
(1 row)

postgres@bizdb01:~$ psql -h 192.168.150.110 -U postgres -p 6000 -c "SELECT
pg_is_in_recovery();"
Password for user postgres:
 pg_is_in_recovery
-------------------
 t
(1 row)
```

Check PostgreSQL Cluster Through HAProxy Stats Dashboard

http://192.168.150.110:7000



6. Patronictl Commands

List Nodes:

root@bizdb01:~# **patronictl -c /usr/patroni/conf/postgresql.yml list**

```
+ Cluster: postgres --------+---------+---------+----+-----------+--------------+
| Member  | Host           | Role    | State   | TL | Lag in MB | Tags         |
+---------+----------------+---------+---------+----+-----------+--------------+
| bizdb01 | 192.168.150.100 | Leader  | running |  4 |           | nosync: true |
| bizdb02 | 192.168.150.101 | Replica | running |  4 |         0 | nosync: true |
+---------+----------------+---------+---------+----+-----------+--------------+
```

Failover:

root@bizdb01:~# **patronictl -c /usr/patroni/conf/postgresql.yml failover**

Current cluster topology

```
+ Cluster: postgres --------+---------+---------+----+-----------+--------------+
| Member  | Host           | Role    | State   | TL | Lag in MB | Tags         |
+---------+----------------+---------+---------+----+-----------+--------------+
| bizdb01 | 192.168.150.100 | Leader  | running |  4 |           | nosync: true |
| bizdb02 | 192.168.150.101 | Replica | running |  4 |         0 | nosync: true |
+---------+----------------+---------+---------+----+-----------+--------------+
```

Candidate ['bizdb02'] []: **bizdb02**

Are you sure you want to failover cluster postgres, demoting current leader bizdb01? [y/N]: **y**

2023-03-20 17:21:38.43431 Successfully failed over to "bizdb02"

```
+ Cluster: postgres --------+---------+---------+----+-----------+--------------+
| Member  | Host           | Role    | State   | TL | Lag in MB | Tags         |
```

```
+---------+----------------+---------+---------+----+-----------+--------------+
| bizdb01 | 192.168.150.100 | Replica | stopped |    |   unknown | nosync: true |
| bizdb02 | 192.168.150.101 | Leader  | running |  4 |           | nosync: true |
+---------+----------------+---------+---------+----+-----------+--------------+
root@bizdb01:~# patronictl -c /usr/patroni/conf/postgresql.yml list
+ Cluster: postgres --------+---------+---------+----+-----------+--------------+
| Member  | Host           | Role    | State   | TL | Lag in MB | Tags         |
+---------+----------------+---------+---------+----+-----------+--------------+
| bizdb01 | 192.168.150.100 | Replica | running |  5 |         0 | nosync: true |
| bizdb02 | 192.168.150.101 | Leader  | running |  5 |           | nosync: true |
+---------+----------------+---------+---------+----+-----------+--------------+
```

7. Install Keepalived on HAProxy Nodes

Determine The Default Network Interface on Both Nodes For Virtual IP Address

```
root@haproxy01:~# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe17:643a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:17:64:3a  txqueuelen 1000  (Ethernet)
        RX packets 7018  bytes 10120224 (10.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1011  bytes 80841 (80.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.150.110  netmask 255.255.255.0  broadcast 192.168.150.255
        inet6 fe80::a00:27ff:fe75:5361  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:75:53:61  txqueuelen 1000  (Ethernet)
        RX packets 1017273  bytes 103387316 (103.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1005278  bytes 103836782 (103.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 9364  bytes 506298 (506.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9364  bytes 506298 (506.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

In my case, I will use **enp0s8** because it is the interface where HAProxy IP resides.

Install Keepalived From APT Repository on Both Nodes

root@haproxy01:~# **sudo apt-get install keepalived -y**
root@haproxy02:~# **sudo apt-get install keepalived -y**

Using Nano, Create keepalived.conf on Node 1

root@haproxy01:~# **nano /etc/keepalived/keepalived.conf**

```
global_defs {
  router_id gd_ha01
  default_interface enp0s8
}
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2                  # every 2 seconds
  fall 2
  rise 2
}
vrrp_instance VI_1 {
  interface enp0s8            # interface to monitor
  nopreempt
  state BACKUP                # use nopreempt, so BACKUP on both of haproxy01 and haproxy02
  virtual_router_id 91        # use a unique id shared between haproxy01 and haproxy02
  priority 101                # 101 on haproxy01, 100 on haproxy02
  advert_int 1
  unicast_src_ip 192.168.150.110
  unicast_peer {
     192.168.150.111
  }
  authentication {
    auth_type PASS
    auth_pass haproxy1234 # specify the same password for haproxy01 and haproxy02
  }
  virtual_ipaddress {
    192.168.150.200 # specify a virtual ip address agreed before when generating SSL for
postgresql
  }
  track_script {
    chk_haproxy
  }
}
```

Using Nano, Create keepalived.conf on Node 2

root@haproxy02:~# **nano /etc/keepalived/keepalived.conf**

```
global_defs {
  router_id gd_ha02
  default_interface enp0s8
}
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2                  # every 2 seconds
  fall 2
  rise 2
}
vrrp_instance VI_1 {
  interface enp0s8            # interface to monitor
  nopreempt
  state BACKUP                # use nopreempt, so BACKUP on both of haproxy01 and haproxy02
  virtual_router_id 91        # use a unique id shared between haproxy01 and haproxy02
  priority 100                # 101 on haproxy01, 100 on haproxy02
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass haproxy1234 # specify the same password for haproxy01 and haproxy02
  }
  unicast_src_ip 192.168.150.111
  unicast_peer {
    192.168.150.110
  }
  virtual_ipaddress {
    192.168.150.200 # specify a virtual ip address agreed before when generating SSL for
postgresql
  }
  track_script {
    chk_haproxy
  }
}
```

Configure IP Forwarding and Non-local Binding on Both HAProxy Nodes

root@haproxy01:~# **nano /etc/sysctl.conf**

….

….

```
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1
```

root@haproxy01:~# **sysctl -p**
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1

root@haproxy02:~# **nano /etc/sysctl.conf**
….
….
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1

root@haproxy02:~# **sysctl -p**
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1

Start Keepalived Services on Both HAProxy Nodes

root@haproxy01:~# **sudo systemctl enable keepalived**
Synchronizing state of keepalived.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable keepalived
root@haproxy01:~# **sudo systemctl start keepalived**
root@haproxy01:~# **sudo systemctl status keepalived**
● keepalived.service - Keepalive Daemon (LVS and VRRP)
Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2023-03-20 18:27:26 WIB; 13s ago
Main PID: 4970 (keepalived)
Tasks: 2 (limit: 4609)
Memory: 3.2M
CGroup: /system.slice/keepalived.service
├─4970 /usr/sbin/keepalived --dont-fork
└─4971 /usr/sbin/keepalived --dont-fork

Check The Virtual IP Address On Both Nodes

root@haproxy01:~# **ip addr show dev enp0s8**
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:75:53:61 brd ff:ff:ff:ff:ff:ff
inet 192.168.150.110/24 brd 192.168.150.255 scope global enp0s8
valid_lft forever preferred_lft forever
inet 192.168.150.200/32 scope global enp0s8
valid_lft forever preferred_lft forever

```
    inet6 fe80::a00:27ff:fe75:5361/64 scope link
       valid_lft forever preferred_lft forever

root@haproxy02:~# ip addr show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 08:00:27:9e:4a:3a brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.111/24 brd 192.168.150.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe9e:4a3a/64 scope link
       valid_lft forever preferred_lft forever

# The IP Address should only appear on one of the two nodes of the HAProxy hosts.
```

Change Bind Address From * to The Virtual IP Address in Both HAProxy Config

root@haproxy01:~# **nano /etc/haproxy/haproxy.cfg**

```
listen stats
….
  bind 192.168.150.200:7000
….
listen postgres
….
  bind 192.168.150.200:5000
….
listen postgres-readonly
….
  bind 192.168.150.200:6000
….
```

```
defaults
  mode tcp
  log global
  retries 3
  timeout queue 1m
  timeout connect 10s
  timeout client 31m
  timeout server 31m
  timeout check 10s
  maxconn 3000
listen stats
  mode http
  bind 192.168.150.200:7000
  stats enable
  stats uri /
listen postgres
  bind 192.168.150.200:5000
  option httpchk
  http-check expect status 200
  default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
  server bizdb01 192.168.150.100:5432 maxconn 100 check port 8008
  server bizdb02 192.168.150.101:5432 maxconn 100 check port 8008
listen postgres-readonly
  bind 192.168.150.200:6000
  option httpchk GET /replica
  http-check expect status 200
  default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
  server bizdb01 192.168.150.100:5432 maxconn 100 check port 8008
  server bizdb02 192.168.150.101:5432 maxconn 100 check port 8008
```

root@haproxy01:~# **sudo systemctl stop haproxy**
root@haproxy01:~# **sudo systemctl start haproxy**

Test Connect to PostgreSQL Cluster Using Virtual IP Address

Without SSL:

root@bizdb01:~# **sudo -i -u postgres**
postgres@bizdb01:~$ **psql -h 192.168.150.200 -p 5000 -d postgres -U postgres**
Password for user postgres:
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# **\q**

With SSL:

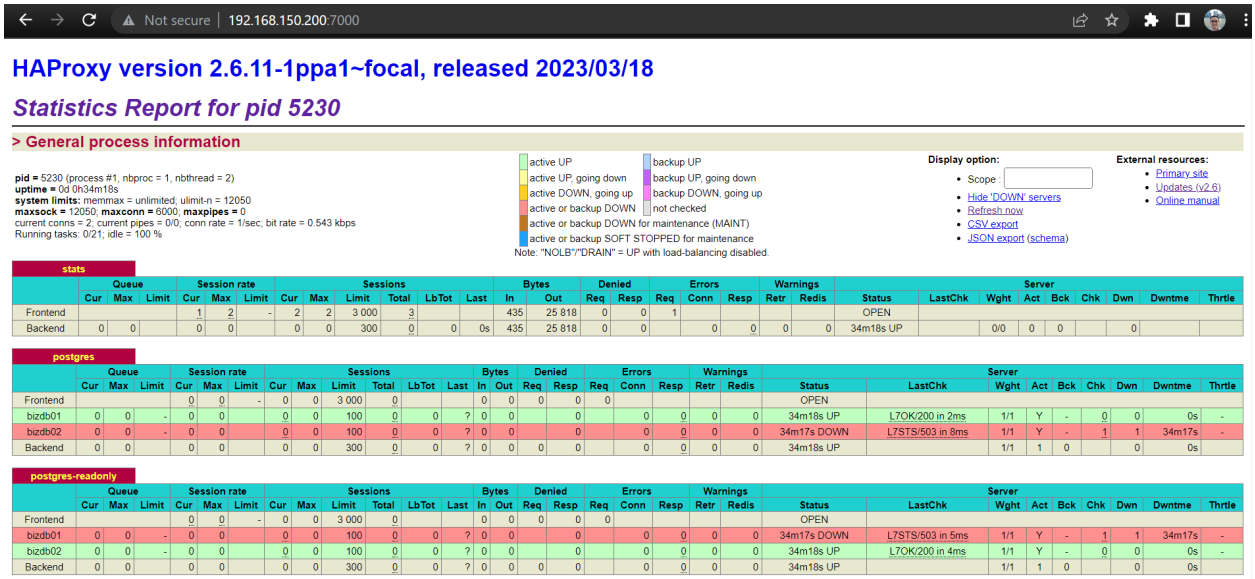
postgres@bizdb01:~$ **psql "host=192.168.150.200 port=5000 dbname=postgres user=postgres password=postgres sslmode=verify-full sslrootcert=/usr/patroni/conf/server.crt"**
psql (14.7 (Ubuntu 14.7-1.pgdg20.04+1))

SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# **\q**

Check PostgreSQL Cluster Through HAProxy Stats Dashboard Using Virtual IP Address

http://192.168.150.200:7000



8. Test Keepalived Service For HAProxy

Check Keepalived Service Logs on Both Nodes

root@haproxy01:~# **journalctl -u keepalived -f**
-- Logs begin at Fri 2023-03-17 19:21:01 WIB. --
Mar 20 18:27:26 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 18:27:29 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE
Mar 20 19:02:37 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 1
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering FAULT STATE
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) sent 0 priority
Mar 20 19:02:47 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 0
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 19:02:53 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE

root@haproxy02:~# **journalctl -u keepalived -f**

-- Logs begin at Fri 2023-03-17 19:21:01 WIB. --
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: SECURITY VIOLATION - scripts are being executed but script_security not enabled.
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: Registering gratuitous ARP shared channel
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) succeeded
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering BACKUP STATE
Mar 20 19:02:32 haproxy02 Keepalived_vrrp[4083]: Script `chk_haproxy` now returning 1
Mar 20 19:02:34 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:02:34 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering FAULT STATE
Mar 20 19:02:52 haproxy02 Keepalived_vrrp[4083]: Script `chk_haproxy` now returning 0
Mar 20 19:02:54 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:02:54 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering BACKUP STATE

Simulate HAProxy Service Problem on Node 1

root@haproxy01:~# **sudo systemctl stop haproxy**

Keepalived Log on Node 1 (Now Node 1 becomes FAULT):

```
root@haproxy01:~# journalctl -u keepalived -f
-- Logs begin at Fri 2023-03-17 19:21:01 WIB. --
Mar 20 18:27:26 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 18:27:29 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE
Mar 20 19:02:37 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 1
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering FAULT STATE
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) sent 0 priority
Mar 20 19:02:47 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 0
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 19:02:53 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE
Mar 20 19:21:43 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 1
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering FAULT STATE
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: (VI_1) sent 0 priority
```

Keepalived Log on Node 2 (Now Node 2 becomes MASTER):

```
root@haproxy02:~# journalctl -u keepalived -f
-- Logs begin at Fri 2023-03-17 19:21:01 WIB. --
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: SECURITY VIOLATION - scripts are being executed but script_security not enabled.
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: Registering gratuitous ARP shared channel
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) succeeded
Mar 20 18:27:33 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering BACKUP STATE
Mar 20 19:02:32 haproxy02 Keepalived_vrrp[4083]: Script `chk_haproxy` now returning 1
Mar 20 19:02:34 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:02:34 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering FAULT STATE
Mar 20 19:02:52 haproxy02 Keepalived_vrrp[4083]: Script `chk_haproxy` now returning 0
Mar 20 19:02:54 haproxy02 Keepalived_vrrp[4083]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:02:54 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering BACKUP STATE
Mar 20 19:21:45 haproxy02 Keepalived_vrrp[4083]: (VI_1) Backup received priority 0 advertisement
Mar 20 19:21:45 haproxy02 Keepalived_vrrp[4083]: (VI_1) Backup received priority 0 advertisement
Mar 20 19:21:46 haproxy02 Keepalived_vrrp[4083]: (VI_1) Entering MASTER STATE
```

Check Virtual IP Address on HAProxy Node 2, The Virtual IP now is switch to there:
root@haproxy02:~# **ip addr show dev enp0s8**
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:9e:4a:3a brd ff:ff:ff:ff:ff:ff
inet 192.168.150.111/24 brd 192.168.150.255 scope global enp0s8
valid_lft forever preferred_lft forever
inet 192.168.150.200/32 scope global enp0s8
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe9e:4a3a/64 scope link
valid_lft forever preferred_lft forever

Simulate HAProxy Service on Node 1 is Backed Up Again

root@haproxy01:~# **sudo systemctl start haproxy**

Keepalived Log on Node 1 (Now Node 1 becomes BACKUP):

```
root@haproxy01:~# journalctl -u keepalived -f
-- Logs begin at Fri 2023-03-17 19:21:01 WIB. --
Mar 20 18:27:26 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 18:27:29 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE
Mar 20 19:02:37 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 1
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering FAULT STATE
Mar 20 19:02:39 haproxy01 Keepalived_vrrp[4971]: (VI_1) sent 0 priority
Mar 20 19:02:47 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 0
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:02:49 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
Mar 20 19:02:53 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering MASTER STATE
Mar 20 19:21:43 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 1
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) failed (exited with status 1)
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering FAULT STATE
Mar 20 19:21:45 haproxy01 Keepalived_vrrp[4971]: (VI_1) sent 0 priority
Mar 20 19:31:18 haproxy01 Keepalived_vrrp[4971]: Script `chk_haproxy` now returning 0
Mar 20 19:31:20 haproxy01 Keepalived_vrrp[4971]: VRRP_Script(chk_haproxy) succeeded
Mar 20 19:31:20 haproxy01 Keepalived_vrrp[4971]: (VI_1) Entering BACKUP STATE
```