VitalPBX Hight Availability

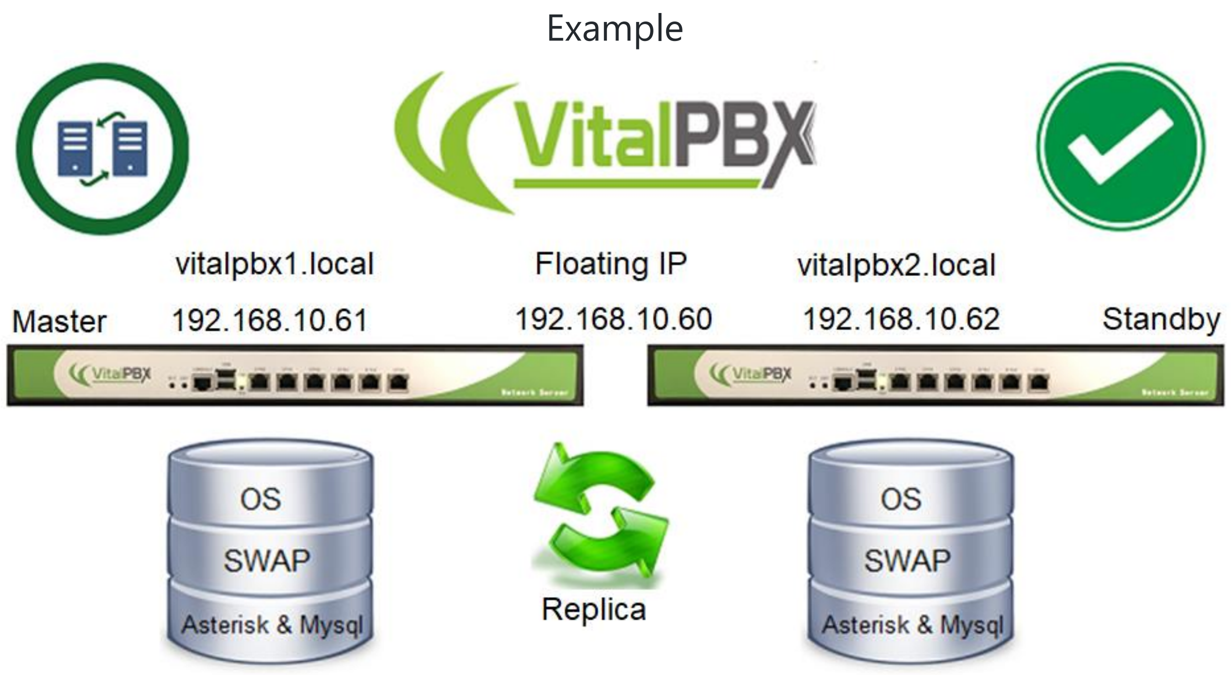
INDEX

1.- Introduction

1.1.- VitalPBX High Availability

High availability is a characteristic of a system which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

Make a high-availability cluster out of any pair of VitalPBX servers. VitalPBX can detect a range of failures on one VitalPBX server and automatically transfer control to the other server, resulting in a telephony environment with minimal down time.



1.2.- Prerequisites

In order to install VitalPBX in high availability you need the following:

a.- 3 IP addresses.
b.- Install VitalPBX Version 3.0 in two servers with similar characteristics.
c.- MariaDB Galera (include in VitalPBX 3)
d.- Corosync, Pacemaker, PCS and lsyncd.

2.- Configurations

2.1- IP Configuration and Hostname.

We will configure in each server the IP address and the host name.

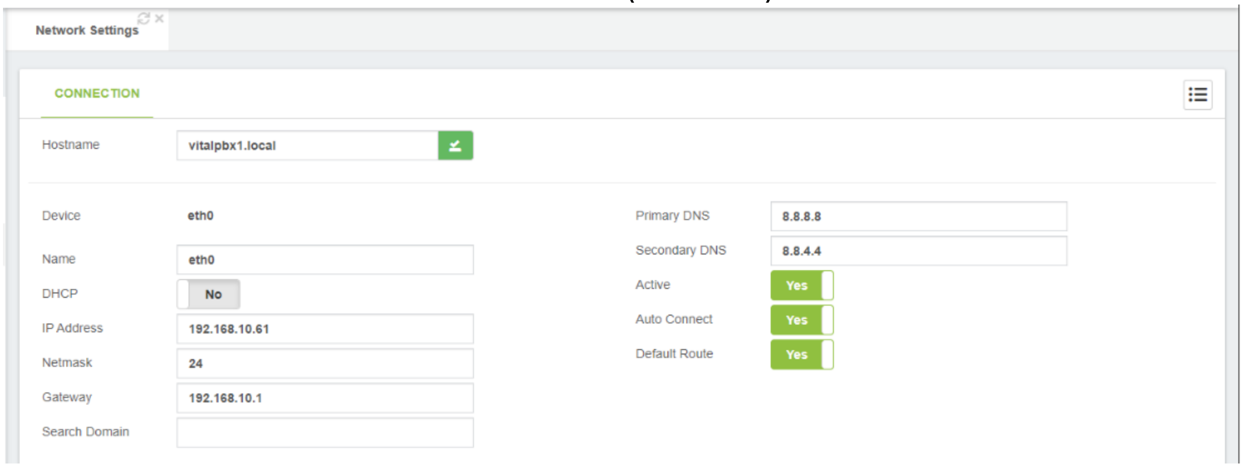
First, we will go to the web interface under:
Admin>System Settings>Network Settings

Disable DHCP and configure the selected IP and hostname. In our example we will use the following values.

Name	**Master**	**Standby**
Hostname	vitalpbx1.local	vitalpbx2.local
IP Address	192.168.10.61	192.168.10.62
Netmask	255.255.255.0	255.255.255.0
Gateway	192.168.10.1	192.168.10.1
Primary DNS	8.8.8.8	8.8.8.8
Secondary DNS	8.8.4.4	8.8.4.4

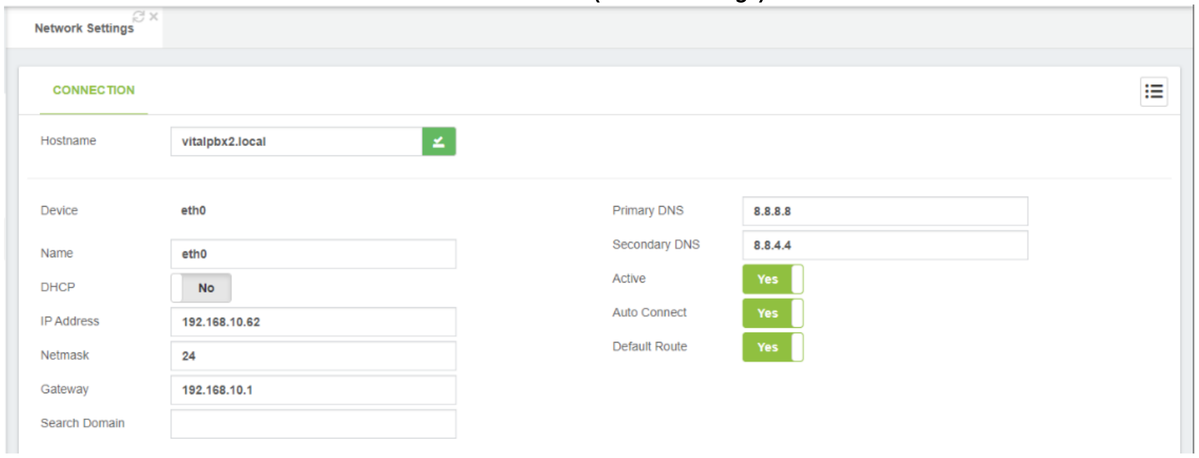
First change the Hostname, remember press the **Check button (🟩)** next to it to apply the new hostname.

Server 1 (Master)



Hostname	vitalpbx1.local
Device	eth0
Name	eth0
DHCP	No
IP Address	192.168.10.61
Netmask	24
Gateway	192.168.10.1
Search Domain	
Primary DNS	8.8.8.8
Secondary DNS	8.8.4.4
Active	Yes
Auto Connect	Yes
Default Route	Yes

Server 2 (Standby)


Network Settings

CONNECTION

Hostname	vitalpbx2.local		
Device	eth0	Primary DNS	8.8.8.8
Name	eth0	Secondary DNS	8.8.4.4
DHCP	No	Active	Yes
IP Address	192.168.10.62	Auto Connect	Yes
Netmask	24	Default Route	Yes
Gateway	192.168.10.1		
Search Domain			

You can also change the hostname from the console using the following command:

Server 1

```
[root@ vitalpbx ~]# hostnamectl set-hostname vitalpbx1.local
```

Server 2

```
[root@ vitalpbx ~]# hostnamectl set-hostname vitalpbx2.local
```

2.2.- Installing the necessary software dependencies

For High Availability services we need to install in both servers Corosync and Pacemaker

```
[root@ vitalpbx1-2 ~]# yum -y install corosync pacemaker pcs
```

We are going to synchronize some directories in both servers. For this we need to install lsync in both Server

```
[root@ vitalpbx1-2 ~]# yum install lsyncd -y
```

2.3.- Create authorization key for the Access between the two servers without credentials

Create key in Server 1

```
[root@ vitalpbx1 ~]# ssh-keygen -f /root/.ssh/id_rsa -t rsa -N '' >/dev/null
[root@ vitalpbx1 ~]# ssh-copy-id root@192.168.10.62
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.10.62's password: (remote server root's password)

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@192.168.10.62'"
and check to make sure that only the key(s) you wanted were added.

[root@vitalpbx1 ~]#
```

Create key in Server 2

```
[root@ vitalpbx2 ~]# ssh-keygen -f /root/.ssh/id_rsa -t rsa -N '' >/dev/null
[root@ vitalpbx2 ~]# ssh-copy-id root@192.168.10.61
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.10.61's password: (remote server root's password)

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@192.168.10.61'"
and check to make sure that only the key(s) you wanted were added.

[root@vitalpbx2 ~]#
```

2.4.- Installing from Scripts

If you want to continue step by step go to step 2.5, but if you want to create the configuration automatically, run the following script in Server 1:

```
[root@ vitalpbx1 ~]# mkdir /usr/share/vitalpbx/ha
[root@ vitalpbx1 ~]# cd /usr/share/vitalpbx/ha
[root@ vitalpbx1 ~]# wget https://raw.githubusercontent.com/VitalPBX/vitalpbx_ha/master/vpbxha.sh
[root@ vitalpbx1 ~]# chmod +x vpbxha.sh
[root@ vitalpbx1 ~]# ./vpbxha.sh

************************************************************
*  Welcome to the VitalPBX high availability installation  *
*                All options are mandatory                 *
************************************************************
IP Master................ > 192.168.10.61
IP Standby............... > 192.168.10.62
Floating IP.............. > 192.168.10.60
Floating IP Mask (SIDR).. > 24
hacluster password....... > MyPassword (any password)
************************************************************
*                   Check Information                      *
*        Make sure you have internet on both servers       *
************************************************************
Are you sure to continue with this settings? (yes,no) > yes
```

This process may take a couple of minutes, and once it is done, VitalPBX High Availability will be ready to use.
Always remember to use floating ip to manage your VitalPBX. In this example it is 192.168.10.60.

Some interesting commands that we must keep in mind for any problem we have with our cluster

To destroy the cluster and start again we use the following command

```
[root@ vitalpbx1 ~]# ./vpbxha.sh destroy
```

To rebuild the cluster after destroying it

```
[root@ vitalpbx1 ~]# ./vpbxha.sh rebuild
```

And lastly to force a refresh of the cluster

```
[root@ vitalpbx1 ~]# pcs resource refresh --full
```

Remember that in all cases we must have the script on our server

Note:

In some cases, the destroy command does not respond because the corosync service cannot be stopped, if this happens, we recommend stopping the service forcibly with the “kill PID” command (the service PID). Do this on both servers.

```
[root@ vitalpbx1 ~]# systemctl status corosync | grep "Main PID"
Main PID: 5702 (corosync)
[root@vitalpbx1-2 ~]#
```

Kill the service in both servers

```
[root@ vitalpbx1 ~]# kill 5702
```

The PID is different on both servers, so we recommend searching for the PID separately.

Important Note:

Please read the section on 3.- Resources Troubleshooting, this will help you solve any problem you have with your cluster.

2.5.- Pairing both Servers (Continuing with the step by step)

Now, we will pair the two servers because we need them to have the same main Tenant ID.

Server 1

```
[root@ vitalpbx1.local ~]# mysql -uroot ombutel -e "select path from ombu_tenants" | awk 'NR==2'
af739029bb237e9e
```

Remember this ID.

Server 2

```
[root@ vitalpbx2.local ~]# mysql -uroot ombutel -e "select path from ombu_tenants" | awk 'NR==2'
633225cc70d86221
```

Next, in Server 2 Update the Tenant ID with the value of Server 1

```
[root@ vitalpbx2.local ~]# mysql -uroot ombutel -e "update ombu_tenants set path='af739029bb237e9e'"
```

And rename the main Tenant path in Server 2

```
[root@ vitalpbx2.local ~]# mv /var/lib/vitalpbx/static/633225cc70d86221 /var/lib/vitalpbx/static/af739029bb237e9e
```

2.6.- Firewall

In both Servers.

In VitalPBX GUI go to Admin > Firewall > Services and add the following Services.

Server Name	**Port**	**Protocol**
MariaDB Client	3306	TCP
MariaDB Galera Traffic	4567-4568	TCP
MariaDB Galera SST	4444	TCP
HA2224	2224	TCP
HA3121	3121	TCP
HA5403	5403	TCP
HA5404-5405	5404-5405	UPD
HA21064	21064	TCP
HA9929	9929	BOTH

Then, go to Admin > Firewall > Rules and add the Rules for the services created and in the Source for security use the local network (In my case is: 192.168.10.0/24). Then remember to apply changes in both servers.

2.7.- Hostname

Next, we will connect through ssh to each of the servers and we configure the hostname of each server in the /etc/hots file, so that both servers see each other with the hostname.

```
[root@vitalpbx1-2.local ~]# echo -e "192.168.10.61 \tvitalpbx1.local" >> /etc/hosts
[root@vitalpbx1-2.local ~]# echo -e "192.168.10.62 \tvitalpbx2.local" >> /etc/hosts
```

2.8.- Configure MariaDB Galera to replicate the Database in Both VitalPBX Servers.

In Server 1 configure Galera

```
[root@ vitalpbx1.local ~]# nano /etc/my.cnf.d/server.cnf
[galera]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.10.61,192.168.10.62"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="192.168.10.61"
wsrep_node_name="Server 1"
```

Remember to change the IP with your IP Addressing.

In Server 2 configure Galera

```
[root@ vitalpbx2.local ~]# nano /etc/my.cnf.d/server.cnf
[galera]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.10.61,192.168.10.62"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="192.168.10.62"
wsrep_node_name="Server 2"
```

Remember to change the IP with your IP Addressing

Now in Server 1 we proceed to create the Cluster.

```
[root@ vitalpbx1 ~]# systemctl stop mariadb
[root@ vitalpbx1 ~]# galera_new_cluster
```

In Server 2 restart mariadb service

```
[root@ vitalpbx2 ~]# systemctl restart mariadb
```

2.9.- Create Hight Availability for the Server

Afterwards, create a monitor directory in both servers

```
[root@ vitalpbx1-2 ~]# mkdir /var/spool/asterisk/monitor
[root@ vitalpbx1-2 ~]# chown asterisk:asterisk /var/spool/asterisk/monitor
```

2.9.1.- Then, configure lsync in Server 1

```
[root@ vitalpbx1 ~]# nano /etc/lsyncd.conf
----
-- User configuration file for lsyncd.
--
-- Simple example for default rsync.
--
settings {
        logfile    = "/var/log/lsyncd/lsyncd.log",
        statusFile = "/var/log/lsyncd/lsyncd-status.log",
        statusInterval = 20,
        nodaemon   = true,
        insist = true,
}

sync {
        default.rsync,
        source="/var/spool/asterisk/monitor",
        target="192.168.10.62:/var/spool/asterisk/monitor",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/",
        target="192.168.10.62:/var/lib/asterisk/",
        rsync = {
                binary = "/usr/bin/rsync",
                owner = true,
                group = true,
                archive = "true",
                _extra = {
                        "--include=astdb.sqlite3",
                        "--exclude=*"
                        }
                }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/agi-bin/",
        target="192.168.10.62:/var/lib/asterisk/agi-bin/",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/priv-callerintros/",
        target="192.168.10.62:/var/lib/asterisk/priv-callerintros",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/sounds/",
        target="192.168.10.62:/var/lib/asterisk/sounds/",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/vitalpbx",
        target="192.168.10.62:/var/lib/vitalpbx",
        rsync = {
                binary = "/usr/bin/rsync",
                owner = true,
                group = true,
                archive = "true",
                _extra = {
                                "--exclude=*.lic",
                                "--exclude=*.dat",
                        "--exclude=dbsetup-done",
                                "--exclude=cache"
                         }
                }
}

sync {
        default.rsync,
        source="/etc/asterisk",
        target="192.168.10.62:/etc/asterisk",
        rsync={
                owner = true,
                group = true
        }
}
```

Remember to change the IP with your IP Addressing

In Server 2

```
[root@ vitalpbx2 ~]# nano /etc/lsyncd.conf
----
-- User configuration file for lsyncd.
--
-- Simple example for default rsync.
--
settings {
        logfile    = "/var/log/lsyncd/lsyncd.log",
        statusFile = "/var/log/lsyncd/lsyncd-status.log",
        statusInterval = 20,
        nodaemon   = true,
        insist = true,
}

sync {
        default.rsync,
        source="/var/spool/asterisk/monitor",
        target="192.168.10.61:/var/spool/asterisk/monitor",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/",
        target="192.168.10.61:/var/lib/asterisk/",
        rsync = {
                binary = "/usr/bin/rsync",
                owner = true,
                group = true,
                archive = "true",
                _extra = {
                        "--include=astdb.sqlite3",
                        "--exclude=*"
                         }
                }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/agi-bin/",
        target="192.168.10.61:/var/lib/asterisk/agi-bin/",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/priv-callerintros/",
        target="192.168.10.61:/var/lib/asterisk/priv-callerintros",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/asterisk/sounds/",
        target="192.168.10.61:/var/lib/asterisk/sounds/",
        rsync={
                owner = true,
                group = true
        }
}

sync {
        default.rsync,
        source="/var/lib/vitalpbx",
        target="192.168.10.61:/var/lib/vitalpbx",
        rsync = {
                binary = "/usr/bin/rsync",
                owner = true,
                group = true,
                archive = "true",
                _extra = {
                                "--exclude=*.lic",
                                "--exclude=*.dat",
                         "--exclude=dbsetup-done",
                                "--exclude=cache"
                         }
                }
}

sync {
        default.rsync,
        source="/etc/asterisk",
        target="192.168.10.61:/etc/asterisk",
        rsync={
            owner = true,
            group = true
      }
}
```

Remember to change the IP for your IP Address

2.9.2.- Configure the start of services on both servers

```
[root@ vitalpbx1-2 ~]# systemctl start pcsd
[root@ vitalpbx1-2 ~]# systemctl enable pcsd.service
[root@ vitalpbx1-2 ~]# systemctl enable corosync.service
[root@ vitalpbx1-2 ~]# systemctl enable pacemaker.service
```

2.9.3.- Create the password of the hacluster user on both nodes

```
[root@ vitalpbx1-2 ~]# echo MyPassword | passwd --stdin hacluster
```

2.9.4.- Server Authenticate in Server 1

```
[root@ vitalpbx1 ~]# pcs cluster auth vitalpbx1.local vitalpbx2.local -u hacluster -p MyPassword
vitalpbx1.local: Authorized
vitalpbx2.local: Authorized
```

2.9.5.- Create the cluster and configure parameters, perform only on the Server 1

```
[root@ vitalpbx1 ~]# pcs cluster setup --name cluster_vitalpbx vitalpbx1.local vitalpbx2.local
```

2.9.6.- Starting Cluster in Server 1

```
[root@ vitalpbx1 ~]# pcs cluster start --all
[root@ vitalpbx1 ~]# pcs cluster enable --all
[root@ vitalpbx1 ~]# pcs property set stonith-enabled=false
[root@ vitalpbx1 ~]# pcs property set no-quorum-policy=ignore
```

2.9.7.- Stop services and disable in both servers

```
[root@ vitalpbx1-2 ~]#  systemctl stop asterisk
[root@ vitalpbx1-2 ~]#  systemctl disable asterisk
```

2.9.8.- Create resource for the use of the Floating IP

```
[root@ vitalpbx1 ~]# pcs resource create virtual_ip ocf:heartbeat:IPaddr2 ip=192.168.10.60
cidr_netmask=24 op monitor interval=30s on-fail=restart
[root@ vitalpbx1 ~]# pcs cluster cib drbd_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push drbd_cfg
```

2.9.9.- Create asterisk Service in Server 1

```
[root@ vitalpbx1 ~]# pcs resource create asterisk service:asterisk op monitor interval=30s
[root@ vitalpbx1 ~]# pcs cluster cib fs_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint colocation add asterisk with virtual_ip INFINITY
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint order virtual_ip then asterisk
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg –config
[root@ vitalpbx1 ~]# pcs resource update asterisk op stop timeout=120s
```

```
[root@ vitalpbx1 ~]# pcs resource update asterisk op start timeout=120s
[root@ vitalpbx1 ~]# pcs resource update asterisk op restart timeout=120s
```

Note:

Changing these values from 15s (default) to 120s is very important since depending on the server and the number of extensions the Asterisk can take more than 15s to start

2.9.10.- Create lsyncd Service in Server 1

```
[root@ vitalpbx1 ~]# pcs resource create lsyncd service:lsyncd.service op monitor interval=30s
[root@ vitalpbx1 ~]# pcs cluster cib fs_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint colocation add lsyncd with virtual_ip INFINITY
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint order asterisk then lsyncd
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
```

2.10.- Create useful commands for the maintenance of our cluster

2.10.1.- Create “bascul” command in both servers

```
[root@ vitalpbx1-2 ~]# nano /usr/local/bin/bascul
#!/bin/bash
set -e
# Authors:      Rodrigo Cuadra
#               with Collaboration of Jose Miguel Rivera
#               4-Jul-2020
# Support:      rcuadra@aplitel.com
#
#funtion for draw a progress bar
#You must pass as argument the amount of secconds that the progress bar will run
#progress-bar 10 --> it will generate a progress bar that will run per 10 seconds

progress-bar() {
        local duration=${1}

        already_done() { for ((done=0; done<$elapsed; done++)); do printf ">"; done }
        remaining() { for ((remain=$elapsed; remain<$duration; remain++)); do printf " "; done }
        percentage() { printf "| %s%%" $(( (($elapsed)*100)/($duration)*100/100 )); }
        clean_line() { printf "\r"; }

        for (( elapsed=1; elapsed<=$duration; elapsed++ )); do
                already_done; remaining; percentage
                sleep 1
                clean_line
        done
        clean_line
}

server_a=`pcs status | awk 'NR==10 {print $3}'`
server_b=`pcs status | awk 'NR==10 {print $4}'`
server_master=`pcs status resources | awk 'NR==1 {print $4}'`

#Perform some validations
if [ "${server_a}" = "" ] || [ "${server_b}" = "" ]
then
    echo -e "\e[41m There are problems with high availability, please check with the command *pcs status* (we recommend applying
the command *pcs cluster unstandby* in both servers) \e[0m"
    exit;
fi

if [[ "${server_master}" = "${server_a}" ]]; then
        host_master=$server_a
        host_standby=$server_b
else
        host_master=$server_b
        host_standby=$server_a
fi
```

```
# Print a warning message and ask to the user if he wants to continue
echo -e "*************************************************************"
echo -e "*     Change the roles of servers in high availability     *"
echo -e "*\e[41m WARNING-WARNING-WARNING-WARNING-WARNING-WARNING-WARNING  \e[0m*"
echo -e "*All calls in progress will be lost and the system will be *"
echo -e "*     be in an unavailable state for a few seconds.        *"
echo -e "*************************************************************"

#Perform a loop until the users confirm if wants to proceed or not
while [[ $perform_bascul != yes && $perform_bascul != no ]]; do
        read -p "Are you sure to switch from $host_master to $host_standby? (yes,no) > " perform_bascul
done

if [[ "${perform_bascul}" = "yes" ]]; then
        #Unstandby both nodes
        pcs cluster unstandby $host_master
        pcs cluster unstandby $host_standby

        #Do a loop per resource
        pcs status resources | grep "^s.*s(.*):s.*" | awk '{print $1}' | while read -r resource ; do
                #Skip moving the virutal_ip resource, it will be moved at the end
                if [[ "${resource}" != "virtual_ip" ]]; then
                        echo "Moving ${resource} from ${host_master} to ${host_standby}"
                        pcs resource move  ${host_standby}
                fi
        done

        sleep 5 && pcs cluster standby $host_master & #Standby current Master node after five seconds
        sleep 20 && pcs cluster unstandby $host_master & #Automatically Unstandby current Master node after$

        #Move the Virtual IP resource to standby node
        echo "Moving virutal_ip from ${host_master} to ${host_standby}"
        pcs resource move virtual_ip ${host_standby}

        #End the script
        echo "Becoming ${host_standby} to Master"
        progress-bar 10
        echo "Done"
else
        echo "Nothing to do, bye, bye"
fi

sleep 5
role
```

Add permissions

```
[root@ vitalpbx1-2 ~]# chmod +x /usr/local/bin/bascul
```

2.10.2.- Create "role" command in both servers

```
[root@ vitalpbx1-2 ~]# nano /usr/local/bin/role
#Bash Colour Codes
green="\033[00;32m"
txtrst="\033[00;0m"
if [ -f /etc/redhat-release ]; then
        linux_ver=`cat /etc/redhat-release`
        vitalpbx_ver=`rpm -qi vitalpbx |awk -F: '/^Version/ {print $2}'`
        vitalpbx_release=`rpm -qi vitalpbx |awk -F: '/^Release/ {print $2}'`
elif [ -f /etc/debian_version ]; then
        linux_ver="Debian "`cat /etc/debian_version`
        vitalpbx_ver=`dpkg -l vitalpbx |awk '/ombutel/ {print $3}'`
else
        linux_ver=""
        vitalpbx_ver=""
        vitalpbx_release=""
fi
vpbx_version="${vitalpbx_ver}-${vitalpbx_release}"
asterisk_version=`rpm -q --qf "%{VERSION}" asterisk`
server_master=`pcs status resources | awk 'NR==1 {print $4}'`
host=`hostname`
if [[ "${server_master}" = "${host}" ]]; then
```

```
        server_mode="Master"
else
                server_mode="Standby"
fi
logo='
 _    _ _         _ ______ ______ _    _
| |  | (_)_       | (______ (_____  \ \  / /
| |  | |_| |_ _____| |______) )___)  ) \/ /
 \ \/ /| |  _) / _  | |  _____/  __  ( )  (
  \  / | | |_( ( | | | |    | |__)  ) /\ \
   \/  |_|\___)_||_|_|_|    |______/_/  \_\
'
echo -e "
${green}
${logo}
${txtrst}
 Role          : $server_mode
 Version       : ${vpbx_version//[[:space:]]}
 Asterisk      : ${asterisk_version}
 Linux Version : ${linux_ver}
 Welcome to    : `hostname`
 Uptime        : `uptime | grep -ohe 'up .*' | sed 's/up //g' | awk -F "," '{print $1}'`
 Load          : `uptime | grep -ohe 'load average[s:][: ].*' | awk '{ print "Last Minute: " $3" Last 5 Minutes: "$4" Last 15 Minutes: "$5 }'`
 Users         : `uptime | grep -ohe '[0-9.*] user[s,]'`
 IP Address    : ${green}`ip addr | sed -En 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.){3}[0-9]*).*/\2/p' | xargs`${txtrst}
 Clock         :`timedatectl | sed -n '/Local time/ s/^[ \t]*Local time:\(.*$\)/\1/p'`
 NTP Sync.     :`timedatectl |awk -F: '/NTP sync/ {print $2}'`
"
echo -e ""
echo -e "****************************************************************"
echo -e "*                    Servers Status                            *"
echo -e "****************************************************************"
echo -e "Master"
pcs status resources
echo -e ""
echo -e "Servers Status"
pcs cluster pcsd-status
```

Add permissions and make a copy to /etc/profile.d/vitalwelcome.sh

```
[root@ vitalpbx1-2 ~]# chmod +x /usr/local/bin/role
[root@ vitalpbx1-2 ~]# cp -rf /usr/local/bin/role /etc/profile.d/vitalwelcome.sh
```

2.10.3.- Create “recovery_galera” command and service in both servers

In Server 1

```
[root@ vitalpbx1 ~]# nano /usr/local/bin/recovery_galera
#!/bin/bash
set -e

mariadb_state=`systemctl is-active mariadb >/dev/null 2>&1 && echo YES || echo NO`
if [ "${mariadb_state}" = 'NO' ] ;then
        galera_state_server_a=`cat /var/lib/mysql/grastate.dat | grep 'safe_to_bootstrap:'`

        if ping -c 3 192.168.10.62; then
                galera_state_server_b=\`ssh root@192.168.10.62 "cat /var/lib/mysql/grastate.dat | grep 'safe_to_bootstrap:'"\`

                if [ "\${galera_state_server_a}" = 'safe_to_bootstrap: 1' ] ;then
                        galera_new_cluster
                        ssh root@192.168.10.62 "systemctl start mariadb"
                        systemctl status mariadb
                        exit;
                fi

                if [ "\${galera_state_server_b}" = 'safe_to_bootstrap: 1' ] ;then
                        ssh root@192.168.10.62 "galera_new_cluster"
                        systemctl start mariadb
                        systemctl status mariadb
                        exit;
                fi
        else
```

```
                                        sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap: 1/g' /var/lib/mysql/grastate.dat
                                        galera_new_cluster
                    fi

else
                    echo -e "*************************************************************"
                    echo -e "*             MariaDB Service is Running               *"
                    echo -e "*************************************************************"
fi
EOF
```

Add permissions in server 1

```
[root@ vitalpbx1 ~]# chmod +x /usr/local/bin/recovery_galera
```

Create the service in Server 1

```
[root@ vitalpbx1 ~]# nano /etc/systemd/system/recovery-galera.service
[Unit]
Description=Run script at startup after network and mariadb becomes reachable
After=network.target
After=mariadb.service

[Service]
Type=Recovery Galera
RemainAfterExit=yes
ExecStart=/usr/local/bin/recovery_galera
TimeoutStartSec=0

[Install]
WantedBy=default.target
```

Enable the Service in server 1

```
[root@ vitalpbx1 ~]# systemctl daemon-reload
[root@ vitalpbx1 ~]# systemctl enable recovery-galera.service
```

In Server 2

```
[root@ vitalpbx2 ~]# nano /usr/local/bin/recovery_galera
#!/bin/bash
set -e

mariadb_state=`systemctl is-active mariadb >/dev/null 2>&1 && echo YES || echo NO`
if [ "${mariadb_state}" = 'NO' ] ;then
                    galera_state_server_a=`cat /var/lib/mysql/grastate.dat | grep 'safe_to_bootstrap:'`

                    if ping -c 3 192.168.10.62; then
                                        galera_state_server_b=\`ssh root@192.168.10.61 "cat /var/lib/mysql/grastate.dat | grep 'safe_to_bootstrap:'"\`

                                        if [ "\${galera_state_server_a}" = 'safe_to_bootstrap: 1' ] ;then
                                                            galera_new_cluster
                                                            ssh root@192.168.10.61 "systemctl start mariadb"
                                                            systemctl status mariadb
                                                            exit;
                                        fi

                                        if [ "\${galera_state_server_b}" = 'safe_to_bootstrap: 1' ] ;then
                                                            ssh root@192.168.10.61 "galera_new_cluster"
                                                            systemctl start mariadb
                                                            systemctl status mariadb
                                                            exit;
                                        fi
                    else
                                        sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap: 1/g' /var/lib/mysql/grastate.dat
                                        galera_new_cluster
```

```
                fi

else
                echo -e "*************************************************************"
                echo -e "*            MariaDB Service is Running              *"
                echo -e "*************************************************************"
fi
EOF
```

Add permissions in server 2

```
[root@ vitalpbx2 ~]# chmod +x /usr/local/bin/recovery_galera
```

Create the service in Server 1

```
[root@ vitalpbx2 ~]# nano /etc/systemd/system/recovery-galera.service
[Unit]
Description=Run script at startup after network and mariadb becomes reachable
After=network.target
After=mariadb.service

[Service]
Type=Recovery Galera
RemainAfterExit=yes
ExecStart=/usr/local/bin/recovery_galera
TimeoutStartSec=0

[Install]
WantedBy=default.target
```

Enable the Service in server 2

```
[root@ vitalpbx2 ~]# systemctl daemon-reload
[root@ vitalpbx2 ~]# systemctl enable recovery-galera.service
```

2.11.- Test

2.11.1.- Test bascul

```
[root@vitalpbx1 ~]# bascul
************************************************************
*     Change the roles of servers in high availability     *
* WARNING-WARNING-WARNING-WARNING-WARNING-WARNING-WARNING  *
*All calls in progress will be lost and the system will be *
*     be in an unavailable state for a few seconds.        *
************************************************************
Are you sure to switch from vitalpbx1.local to vitalpbx2.local? (yes,no) > yes
Moving virutal_ip from vitalpbx1.local to vitalpbx2.local
Becoming vitalpbx2.local to Master
Done>>>>>>| 100%

************************************************************
*                   Servers Status                         *
************************************************************
Master
 virtual_ip   (ocf::heartbeat:IPaddr2):     Started vitalpbx2.local
 asterisk     (service:asterisk):    Started vitalpbx2.local
 lsyncd (service:lsyncd.service):     Started vitalpbx2.local
```

```
Servers Status
  vitalpbx1.local: Online
  vitalpbx2.local: Online
[root@vitalpbx1 ~]#
```

2.11.2.- Show the Cluster status

```
[root@vitalpbx1 ~]# role

************************************************************
*                   Servers Status                         *
************************************************************
Master
 virtual_ip    (ocf::heartbeat:IPaddr2):     Started vitalpbx2.local
 asterisk      (service:asterisk):    Started vitalpbx2.local
 lsyncd (service:lsyncd.service):      Started vitalpbx2.local

Servers Status
  vitalpbx1.local: Online
  vitalpbx2.local: Online
[root@vitalpbx1 ~]#
```

3.- Resources troubleshooting

3.1.- Very useful commands

If a resource has failed, a failure message appears when you display the cluster status. If you resolve that resource, you can clear that failure status with the pcs resource cleanup command. This command resets the resource status and fail count, telling the cluster to forget the operation history of a resource and re-detect its current state.
The following command cleans up the resource specified by resource_id.

```
[root@vitalpbx1-2 ~]# pcs resource cleanup resource_id
```

If you do not specify a resource_id, this command resets the resource status and fail count for all resources.

As of Red Hat Enterprise Linux 7.5, the pcs resource cleanup command probes only the resources that display as a failed action. To probe all resources on all nodes you can enter the following command:

```
[root@vitalpbx1-2 ~]# pcs resource refresh
```

By default, the pcs resource refresh command probes only the nodes where a resource's state is known. To probe all resources even if the state is not known, enter the following command:

```
[root@vitalpbx1-2 ~]# pcs resource refresh --full
```

Show all the parameters associate with resources

```
[root@vitalpbx1-2 ~]# pcs resource show (asterisk, lsyncd or virtual_ip)
```

Is possible update a parameter after finish to create the cluster with the fallowing command. For change the timeout in start for the resource asterisk do this:

```
[root@vitalpbx1-2 ~]# pcs resource update asterisk op start timeout=120s
```

3.2.- Mariadb does not start correctly

If you shut down all nodes at the same time, then you have effectively terminated the cluster. Of course, the cluster's data still exists, but the running cluster no longer exists. When this happens, you'll need to bootstrap the cluster again.

If the cluster is not bootstrapped and mysqld on the first node is just started normally, then the node willl try to connect to at least one of the nodes listed in the wsrep_cluster_address option. If no nodes are currently running, then this will fail. Bootstrapping the first node solves this problem.

In some cases Galera will refuse to bootstrap a node if it detects that it might not be the most advanced node in the cluster. Galera makes this determination if the node was not the last one in the cluster to be shut down or if the node crashed. In those cases, manual intervention is needed.
If you experience this issue the **recovery_galera** command solves it

```
[root@vitalpbx1 ~]# recovery_galera
```

If we cannot recover with the recovery_galera command it means that we will have to do it manually, for which on the server we will edit the file **/var/lib/mysql/grastate.dat** and change the value of **safe_to_bootstrap: 0** to **safe_to_bootstrap: 1** on the server that we believe has the most up-to-date data from the databases.

```
[root@vitalpbx# ~]# nano /var/lib/mysql/grastate.dat
```

Then on the same server we execute the following command:

```
[root@vitalpbx# ~]# galera_new_cluster
```

And on the other server we start mariadb normally

```
[root@vitalpbx$ ~]# systemctl restart mariadb
```

With this, our mariadb cluster should be normalized.

3.3.- I can't get into the VitalPBX interface, it gives me a sql error

A command may fail with ER_UNKNOWN_COM_ERROR producing 'WSREP has not yet prepared node for application use' (or 'Unknown command' in older versions) error message. It happens when a cluster is suspected to be split and the node is in a smaller part — for example, during a network glitch, when nodes temporarily lose each other. It can also occur during state transfer. The node takes this measure to prevent data inconsistency. Its usually a temporary state which can be detected by checking wsrep_ready value. The node, however, allows SHOW and SET command during this period.

In the Server that have the Issue

```
[root@vitalpbx# ~]# mysql -uroot
[root@vitalpbx# ~]# SET GLOBAL wsrep_provider_options="pc.bootstrap=yes";
[root@vitalpbx# ~]# exit;
```

Now on the other server

```
[root@vitalpbx# ~]# systemctl restart mariadb
```

3.4.- Cluster Destroy

In some cases, the cluster is not working as we expect, sometimes it is necessary to make the cluster again, so the steps to follow are as follows.

To destroy the cluster on both servers run the following command

```
[root@vitalpbx1-2 ~]# pcs cluster destroy
```

If you are configuring step by step you must follow the following steps

3.5.- Create the Cluster again

3.5.1.- Create the cluster and configure parameters, perform only on the Server 1

```
[root@ vitalpbx1 ~]# pcs cluster setup --name cluster_vitalpbx vitalpbx1.local vitalpbx2.local
```

3.5.2.- Starting Cluster in Server 1

```
[root@ vitalpbx1 ~]# pcs cluster start --all
[root@ vitalpbx1 ~]# pcs cluster enable --all
[root@ vitalpbx1 ~]# pcs property set stonith-enabled=false
[root@ vitalpbx1 ~]# pcs property set no-quorum-policy=ignore
```

3.5.3.- Stop services and disable in both servers

```
[root@ vitalpbx1-2 ~]#  systemctl stop asterisk
[root@ vitalpbx1-2 ~]#  systemctl disable asterisk
```

3.5.4.- Create resource for the use of the Floating IP

```
[root@ vitalpbx1 ~]# pcs resource create virtual_ip ocf:heartbeat:IPaddr2 ip=192.168.10.60
cidr_netmask=24 op monitor interval=30s on-fail=restart
[root@ vitalpbx1 ~]# pcs cluster cib drbd_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push drbd_cfg
```

3.5.5.- Create asterisk Service in Server 1

```
[root@ vitalpbx1 ~]# pcs resource create asterisk service:asterisk op monitor interval=30s
[root@ vitalpbx1 ~]# pcs cluster cib fs_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint colocation add asterisk with virtual_ip INFINITY
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint order virtual_ip then asterisk
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg –config
```

```
[root@ vitalpbx1 ~]# pcs resource update asterisk op stop timeout=120s
[root@ vitalpbx1 ~]# pcs resource update asterisk op start timeout=120s
[root@ vitalpbx1 ~]# pcs resource update asterisk op restart timeout=120s
```

3.5.6.- Create lsyncd Service in Server 1

```
[root@ vitalpbx1 ~]# pcs resource create lsyncd service:lsyncd.service op monitor interval=30s
[root@ vitalpbx1 ~]# pcs cluster cib fs_cfg
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint colocation add lsyncd with virtual_ip INFINITY
[root@ vitalpbx1 ~]# pcs -f fs_cfg constraint order asterisk then lsyncd
[root@ vitalpbx1 ~]# pcs cluster cib-push fs_cfg --config
```

3.5.7.- Server 1 and 2

```
[root@vitalpbx1-2 ~]# pcs resource refresh --full
```

Wait 30 seconds and run the role command to ensure everything is fine.

```
[root@vitalpbx1 ~]# role
```

3.6.- Destroy the Cluster and Normalize both servers

If for any reason you want to become independent again the two servers must follow the following procedure.

Destroy the cluster on both servers run the following command

```
[root@vitalpbx1-2 ~]# pcs cluster destroy
```

Note:

In some cases, the destroy command does not respond because the corosync service cannot be stopped, if this happens, we recommend stopping the service forcibly with the “kill PID” command (the service PID). Do this on both servers.

```
[root@ vitalpbx1-2 ~]# systemctl status corosync | grep "Main PID"
Main PID: 5702 (corosync)
[root@vitalpbx1-2 ~]#
```

Kill the service in both servers

```
[root@ vitalpbx1 ~]# kill 5702
```

The PID is different on both servers, so we recommend searching for the PID separately.

Now that we have destroyed the cluster we are going to remove the Galera cluster.

It's easy. You should gracefully stop all nodes. After this remove all Galera configuration from /etc/my.cnf.d/server.cnf and start MySQL. Better to do it on latest stopped node, to prevent data loss.

```
[root@ vitalpbx1-2 ~]# systemctl stop mariadb
```

In Server 1 remove all the yellow data, do the same in server 2.

```
[root@ vitalpbx1.local ~]# nano /etc/my.cnf.d/server.cnf
```

```
[galera]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.10.61,192.168.10.62"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="192.168.10.61"
wsrep_node_name="Server 1"
```

Now start mariadb in both servers

```
[root@ vitalpbx1-2 ~]# systemctl start mariadb
```

Now normalize the start of the asterisk service on both servers

```
[root@ vitalpbx1-2 ~]# systemctl enable asterisk
[root@ vitalpbx1-2 ~]# systemctl start asterisk
```

Now go to the /etc/profile.d/vitalwelcome.sh file and delete the lines that are yellow on both servers

```
[root@ vitalpbx1-2 ~]# nano /etc/profile.d/vitalwelcome.sh
#Bash Colour Codes
green="\033[00;32m"
txtrst="\033[00;0m"
if [ -f /etc/redhat-release ]; then
        linux_ver=`cat /etc/redhat-release`
        vitalpbx_ver=`rpm -qi vitalpbx |awk -F: '/^Version/ {print $2}'`
        vitalpbx_release=`rpm -qi vitalpbx |awk -F: '/^Release/ {print $2}'`
elif [ -f /etc/debian_version ]; then
        linux_ver="Debian "`cat /etc/debian_version`
        vitalpbx_ver=`dpkg -l vitalpbx |awk '/ombutel/ {print $3}'`
else
        linux_ver=""
        vitalpbx_ver=""
        vitalpbx_release=""
fi
vpbx_version="${vitalpbx_ver}-${vitalpbx_release}"
asterisk_version=`rpm -q --qf "%{VERSION}" asterisk`
server_master=`pcs status resources | awk 'NR==1 {print $4}'`
host=`hostname`
if [[ "${server_master}" = "${host}" ]]; then
        server_mode="Master"
else
                server_mode="Standby"
fi
logo='

 _       _ _             _ ______  ______ _     _
| |     | (_) _          | (_____  (_____  \ \   / /
| |     | |_| |_   _____| |______) )____)    ) \/ /
 \ \/  /| |   _) /  _   | |    ____/   __    ( )    (
   \   /  |  |  |_(  (  |  |  |  |        |  |__)   )  /\  \
    \/   |_|\___)_||_|_|_|        |_______/_/   \_\
'
echo -e "
${green}
${logo}
${txtrst}
 Role          : $server_mode
 Version       : ${vpbx_version//[[:space:]]}
 Asterisk      : ${asterisk_version}
 Linux Version  : ${linux_ver}
 Welcome to    : `hostname`
 Uptime        : `uptime | grep -ohe 'up .*' | sed 's/up //g' | awk -F "," '{print $1}'`
 Load          : `uptime | grep -ohe 'load average[s:][: ].*' | awk '{ print "Last Minute: " $3" Last 5 Minutes: "$4" Last 15 Minutes: "$5 }'`
 Users         : `uptime | grep -ohe '[0-9.*] user[s,]'`
 IP Address    : ${green}`ip addr | sed -En 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.){3}[0-9]*).*/\2/p' | xargs`${txtrst}
 Clock         :`timedatectl | sed -n '/Local time/ s/^[ \t]*Local time:\(.*$\)/\1/p'`
 NTP Sync.     :`timedatectl |awk -F: '/NTP sync/ {print $2}'`
"
echo -e ""
echo -e "************************************************************"
echo -e "*                  Servers Status                          *"
echo -e "************************************************************"
echo -e "Master"
pcs status resources
echo -e ""
echo -e "Servers Status"
pcs cluster pcsd-status
```

Remove all relative file and service in both server

```
[root@ vitalpbx1-2 ~]# systemctl stop recovery_galera
[root@ vitalpbx1-2 ~]# systemctl disable recovery_galera
[root@ vitalpbx1-2 ~]# rm /usr/local/bin/recovery_galera
[root@ vitalpbx1-2 ~]# rm /usr/local/bin/role
[root@ vitalpbx1-2 ~]# rm /usr/local/bin/bascul
[root@ vitalpbx1-2 ~]# rm /etc/systemd/system/recovery-galera.service
```

3.7.- Recommendations

- If you have to turn off both servers at the same time, we recommend that you start by turning off the one in Standby and then the Master.
- If the two servers stopped abruptly, always start first that you think you have the most up-to-date information and a few minutes later the other server.
- If you want to update the version of VitalPBX we recommend you do it first on Server 1, then do a bascul and do it again on Server 2.