# High Availability V4 – Debian 11

# CONTENTS

## Contents

CONTENTS ..... 2

**VITALPBX HIGH AVAILABILITY ..... 3**

1.- INTRODUCTION ..... 3
1.1.- DISTRIBUTED REPLICATED BLOCK DEVICE (DRBD): ..... 3
1.2.- PACEMAKER: ..... 3
1.3.- COROSYNC: ..... 3
1.4.- DEBIAN 11: ..... 3
1.5.- IP PBX: ..... 4
2.- REQUIREMENTS TO INSTALL VITALPBX IN HIGH AVAILABILITY ..... 5
3.- INSTALLATION ..... 6
4.- CONFIGURATIONS ..... 11
4.1- IP AND HOSTNAME CONFIGURATION. ..... 11
4.2.- INSTALL DEPENDENCIES ..... 12
4.3.- HOSTNAME ..... 12
4.4.- CREATE THE PARTITION ON BOTH SERVERS ..... 13
4.5.- USING SCRIPT ..... 13
4.6.- FIREWALL ..... 14
4.7.- FORMAT THE PARTITION ..... 16
4.8.- CONFIGURING DRBD ..... 16
4.9.- CONFIGURE CLUSTER ..... 18
4.10.- BIND ADDRESS ..... 21
4.11.- CREATE "BASCUL" COMMAND IN BOTH SERVERS ..... 21
4.12.- CREATE "ROLE" COMMAND IN BOTH SERVERS ..... 23
5.- ADD NEW SERVICES ..... 25
5.1.- ADD SONATA SWITCHBOARD ..... 25
5.2.- ADD SONATA RECORDING ..... 25
5.3.- ADD SONATA BILLING ..... 26
5.4.- ADD SONATA STATS ..... 26
5.5.- ADD SONATA DIALER ..... 27
5.6.- ADD VITXI ..... 28
5.7.- ADD OPENVPN ..... 29
5.8.- ADD MAINTENANCE ..... 29
6.- TEST ..... 30
7.- UPDATE VITALPBX OR ADD-ONS ..... 31
8.- CHANGING ONE OF THE SERVERS ..... 31
8.1.- CHANGING THE SECONDARY SERVER ..... 31
8.2.- CHANGING THE PRIMARY SERVER ..... 33
9.- SOME USEFUL COMMANDS ..... 36
10.- SOLVE A DRBD SPLIT-BRAIN IN 4 STEPS ..... 37
11.- CREDITS ..... 38
11.1 SOURCES OF INFORMATION ..... 38

Copyright © 2023 VitalPBX. All rights reserved.

# VitalPBX High Availability

## 1.- Introduction

Welcome, tech enthusiasts, to our blog on the cutting-edge world of high availability! In today's digital landscape, where uninterrupted connectivity and seamless operations are paramount, the need for robust systems that can withstand failures and ensure continuous service has never been more critical. Whether you're managing mission-critical applications, virtualized environments, or communication systems like IP PBX, a reliable high availability solution is essential to keep your operations running smoothly.

In this blog, we delve into the realm of High Availability (HA) and explore how a combination of powerful open-source technologies can create a rock-solid infrastructure capable of weathering unexpected challenges. At the core of our discussion lie four key pillars: DRBD, Pacemaker, Corosync, and Debian 11, which together form a formidable alliance to achieve fault tolerance, data redundancy, and seamless failover.

### 1.1.- Distributed Replicated Block Device (DRBD):

Imagine a scenario where data loss due to hardware failure could cripple your business. DRBD, an open-source Linux kernel module, is here to save the day. This revolutionary technology ensures real-time data replication across multiple networked devices, creating mirrored storage that guarantees data integrity and redundancy. Join us as we unravel the inner workings of DRBD and how it forms the foundation of our high availability solution.

### 1.2.- Pacemaker:

Bringing intelligence to the table, Pacemaker is an indispensable cluster resource manager. As the heartbeat of our HA setup, Pacemaker monitors the health of nodes and services, making critical decisions to maintain service availability. We'll explore its advanced capabilities, such as resource allocation, failover policies, and even how it collaborates with DRBD to orchestrate data synchronization and prevent downtime.

### 1.3.- Corosync:

Communication is the key to any successful relationship, and that's precisely where Corosync comes into play. As a messaging layer, Corosync facilitates the exchange of cluster status information among nodes, ensuring they remain in sync and can make informed decisions collectively. Our blog will highlight Corosync's vital role in ensuring seamless cooperation among various components within our high availability architecture.

### 1.4.- Debian 11:

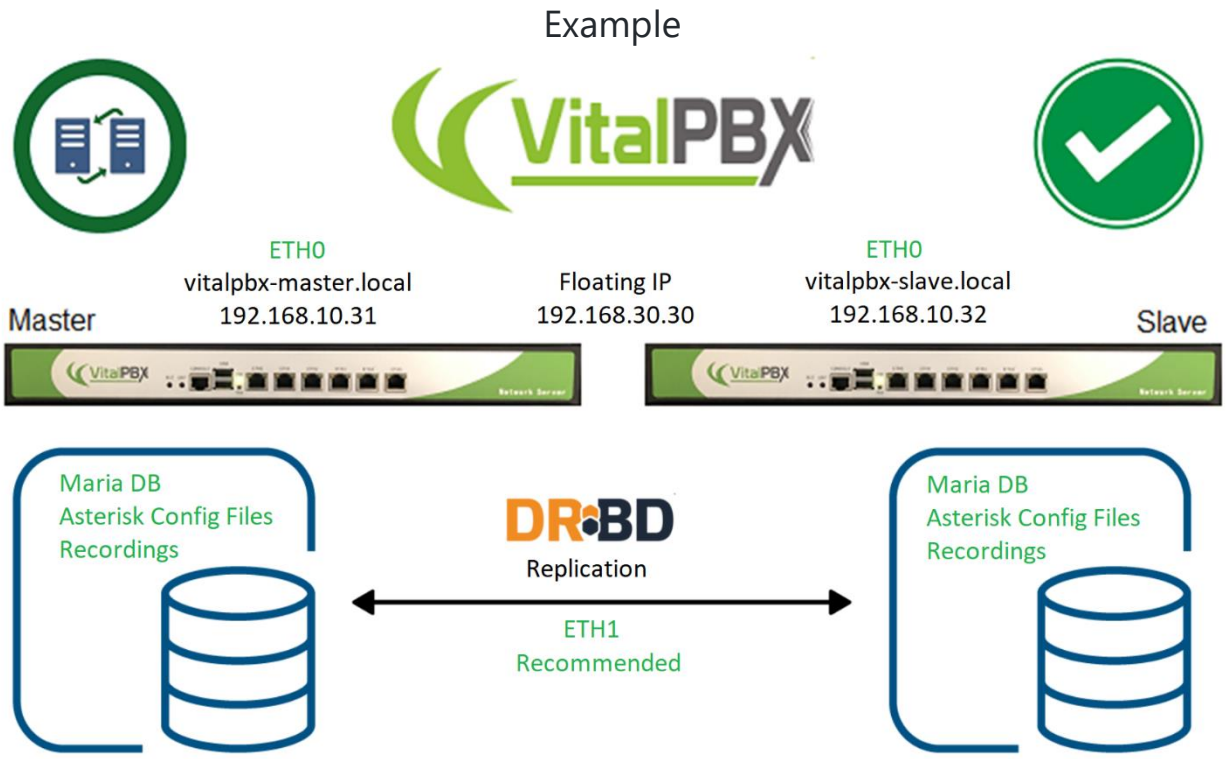
At the heart of our HA setup lies Debian 11, a rock-solid, community-driven operating system known for its stability and reliability. As the chosen platform for our HA journey, we'll explore how Debian 11's

robustness complements our selected technologies, providing a secure and adaptable environment for our high availability system to thrive.

## 1.5.- IP PBX:

Diving into real-world application, we'll demonstrate how our high availability architecture extends its protective embrace to critical communication systems like IP PBX. Discover how the combination of DRBD, Pacemaker, Corosync, and Debian 11 ensures that voice services remain uninterrupted, assuring seamless communication and business continuity.

Together, we will uncover the intricacies of setting up this formidable high availability solution and its potential applications across diverse industries. So, whether you're an IT professional, a system administrator, or a curious tech enthusiast, join us as we embark on this enlightening journey through the world of high availability, where cutting-edge technologies converge to safeguard the lifeline of businesses and organizations alike. Let's fortify our systems and brace ourselves for a future of resilience and uninterrupted operations. Stay tuned!



# 2.- Requirements to install VitalPBX in High Availability

Setting up High Availability (HA) with DRBD (Distributed Replicated Block Device) involves ensuring critical systems and services are available with minimal downtime in case of failures. DRBD enables real-time data replication between nodes to ensure data availability and integrity. Here are some general requirements to set up High Availability with DRBD:

1.- Physical Infrastructure and Networking:
Two or more identical or very similar nodes (servers) to implement redundancy.
Reliable and low-latency network connection between the nodes. This can be a dedicated replication network (preferred) or a shared network if it's of high quality.

2.- Operating System – VitalPBX version:
Nodes should run the same operating system, preferably the same version and the same version of VitalPBX.

3.- Disk Partitioning:
The storage device to be replicated should be partitioned and accessible on both nodes.
Each node should have sufficient storage space to accommodate replication and data.

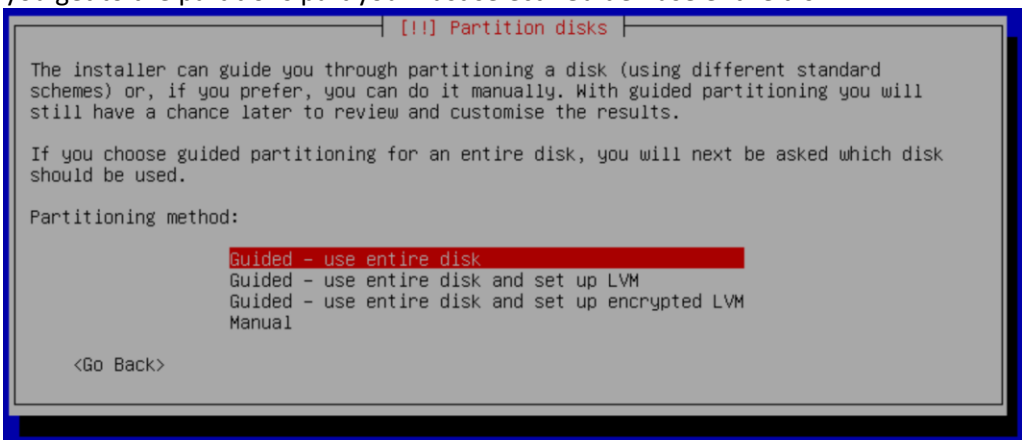
4.- Network Configuration:
Each node should have static IP addresses and resolve correctly in the local DNS system or in the /etc/hosts file of the other node.
Hostnames should be consistent across nodes.

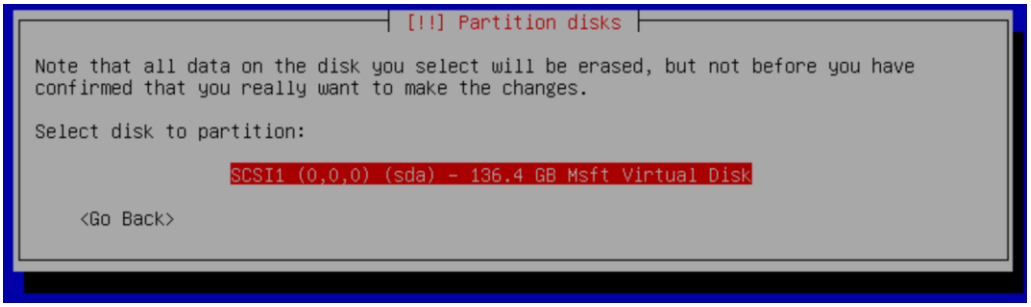
5.- DRBD:
Install and configure DRBD on both nodes.
Configure DRBD resources that define which devices will be replicated and how replication will be established.
At the time of installation leave the largest amount of space on the hard drive to store the variable data on both servers.
Define node roles: primary and secondary nodes.

# 3.- Installation

We are going to start by installing VitalPBX on two servers

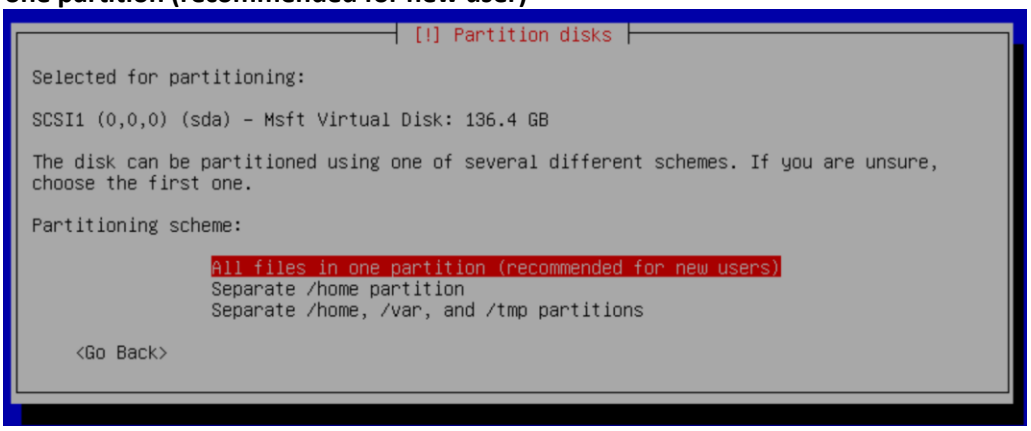
a.- When you get to the partitions part you must select “Guide - use entire disk”:



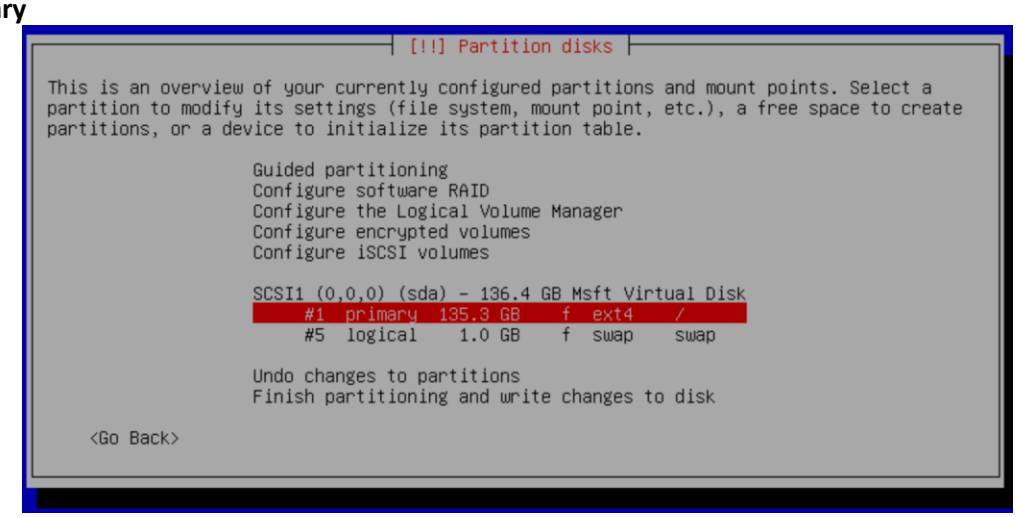

b.- Select:

**All file in one partition (recommended for new user)**

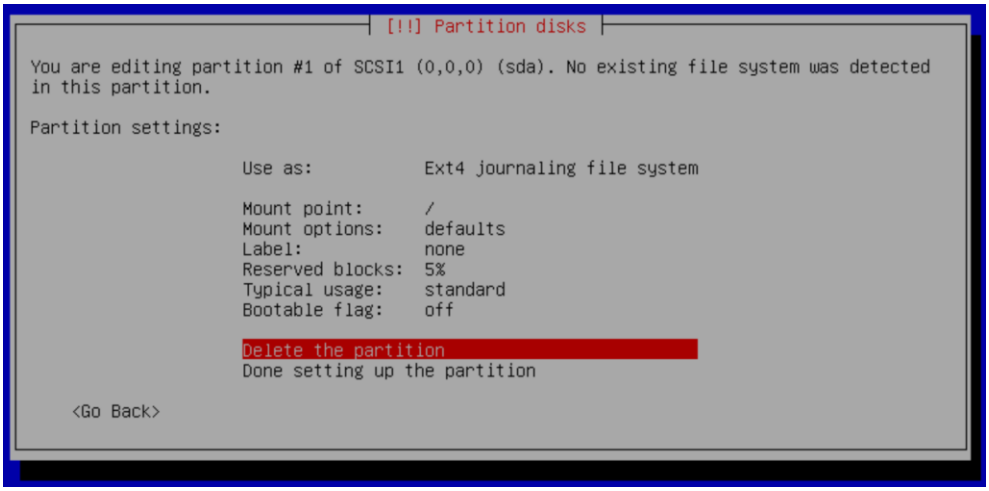


c.- Select primary:

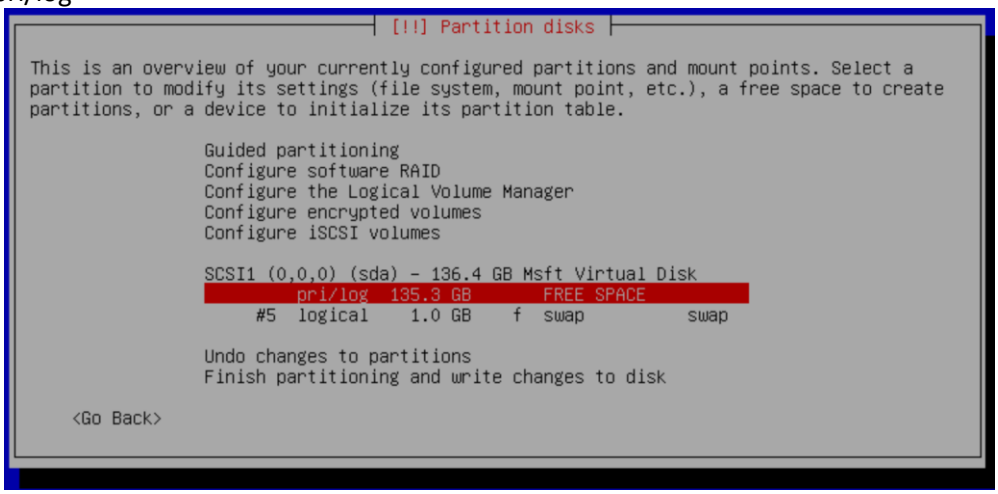
**#1 primary**



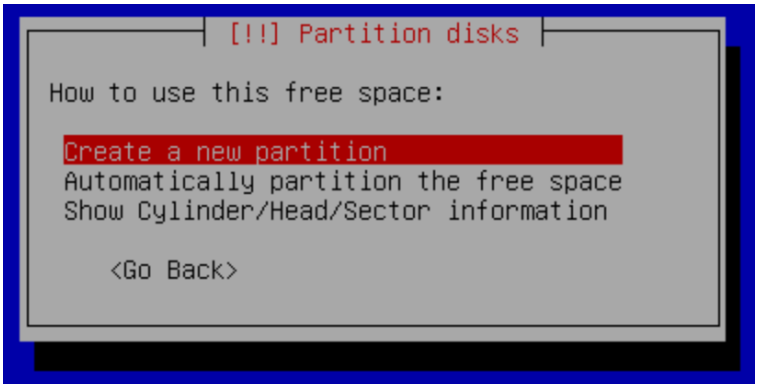

d.- Delete partition




e.- Select pri/log



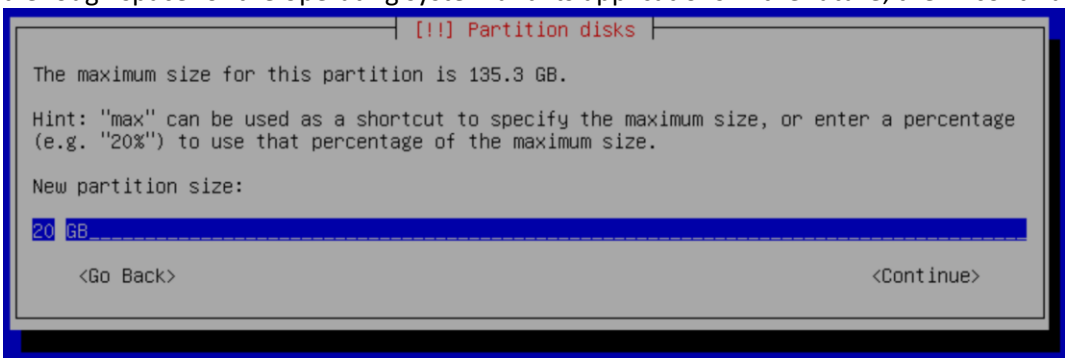
f.- Create New Partition




g.- Change the capacity to:

**New partition size: 20GB**

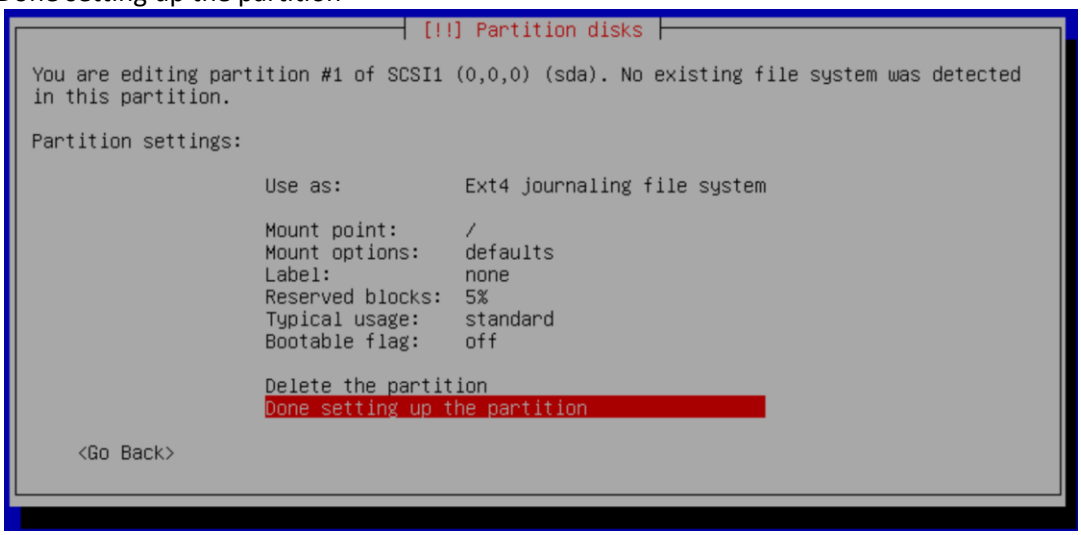
We need enough space for the operating system and its applications in the future; then <continue>



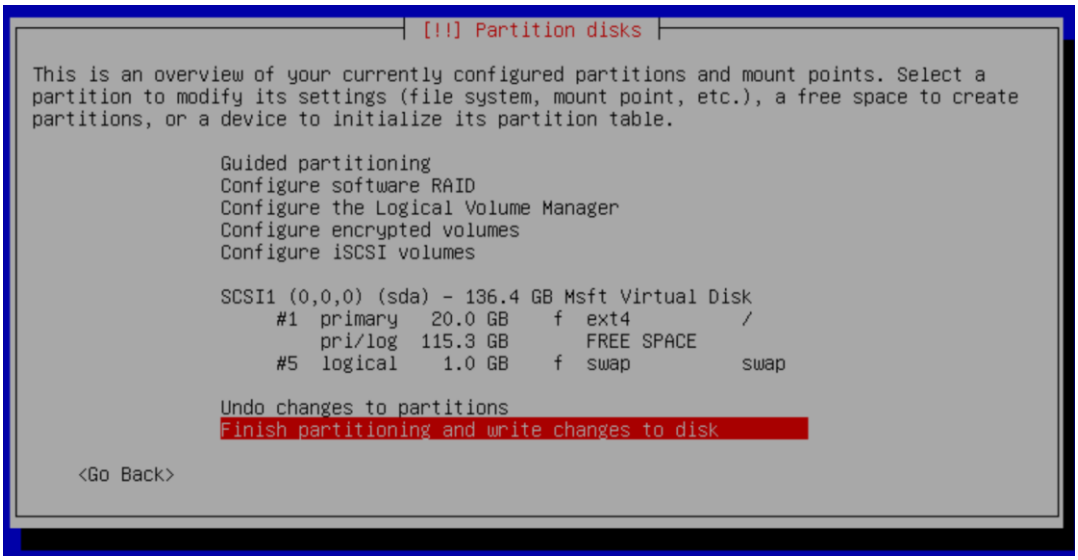

Select Primary

Select Beginning

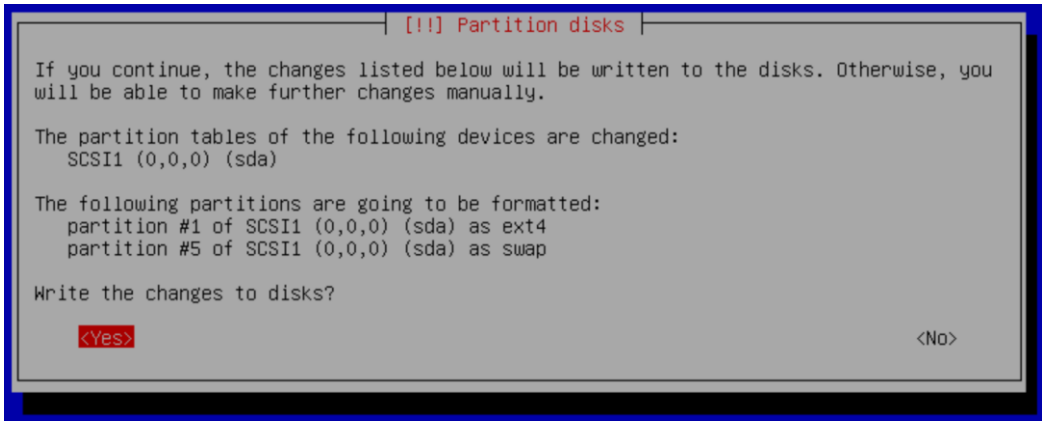
Select Done setting up the partition




h.- Finish



Pres **Yes** to confirm




And continue with the installation.

# 4.- Configurations

## 4.1- IP and Hostname Configuration.

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

| Name | Master | Slave |
|---|---|---|
| Hostname | vitalpbx-master.local | vitalpbx-slave.local |
| IP Address | 192.168.10.31 | 192.168.10.32 |
| Netmask | 255.255.254.0 | 255.255.254.0 |
| 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 |

### 4.1.1.- Remote access with the root user.

As a security measure, the root user is not allowed to remote access the server. If you wish to unblock it, remember to set a complex password, and perform the following steps.

Enter the Command Line Console with the root user directly on your server with the password you set up earlier.
Edit the following file using nano, /etc/ssh/sshd_config.

```
root@vitalpbx-master-slave:~# nano /etc/ssh/sshd_config
```

Change the following line

```
#PermitRootLogin prohibit-password
```

With

```
PermitRootLogIn yes
```

Save, Exit and restart the sshd service.

```
root@vitalpbx-master-slave:~# systemctl restart sshd
```

### 4.1.2.- Changing your IP Address to a static address.

By default, our system's IP address in Debian is obtained through DHCP, however, you can modify it and assign it a static IP address using the following steps:

Edit the following file with nano, /etc/network/interfaces.

```
root@vitalpbx-master-slave:~# nano /etc/network/interfaces
```

Change

```
#The primary network interface
allow-hotplug eth0
iface eth0 inet dchp
```

With the following. Sever Master.

```
#The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 192.168.10.31
netmask 255.255.254.0
gateway 192.168.10.1
```

With the following. Sever Slave.

```
#The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 192.168.10.32
netmask 255.255.254.0
gateway 192.168.10.1
```

Lastly, reboot both servers and you can now log in via ssh.

## 4.2.- Install Dependencies

Install dependencies on both servers

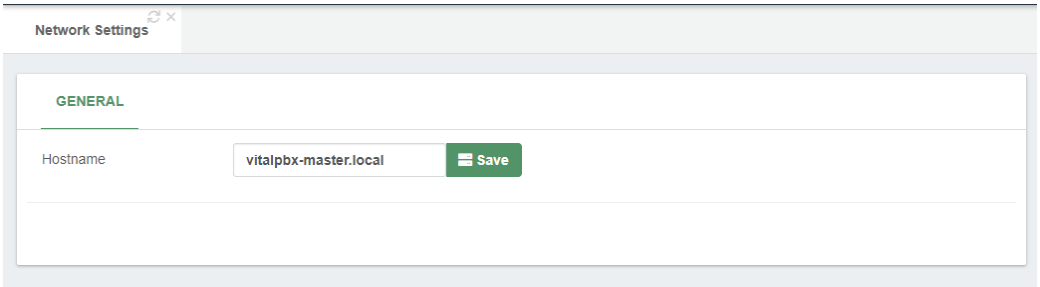
```
root@vitalpbx-master-slave:~# apt -y install drbd-utils corosync pacemaker pcs chrony xfsprogs
```

## 4.3.- Hostname

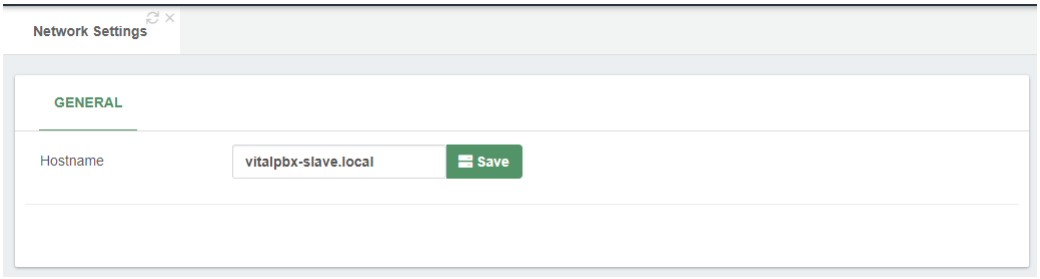
Go to the web interface to:
**ADMIN>System Settings>Network Settings**

First, change the Hostname in both servers, and remember to press the **Save** button.

Master




Slave



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

Server Master

```
root@vitalpbx-master:~# hostname vitalpbx-master.local
```

Server Slave

```
root@vitalpbx-slave:~# hostname vitalpbx-slave.local
```

Server Master and Slave

```
root@vitalpbx-master-slave:~# nano /etc/hosts
192.168.10.31 vitalpbx-master.local
192.168.10.32 vitalpbx-slave.local
```

## 4.4.- Create the partition on both servers

Initialize the partition to allocate the available space on the hard disk. Do these on both servers.

```
root@vitalpbx-master-slave:~# fdisk /dev/sda
Command (m for help): n
Partition type:
  p   primary (3 primary, 0 extended, 1 free)
  e   extended
Select (default e): p
Selected partition 3 (take note of the assigned partition number as we will need it later)
First sector (35155968-266338303, default 35155968): [Enter]
Last sector, +sectors or +size{K,M,G} (35155968-266338303, default 266338303): [Enter]
Using default value 266338303
Partition 4 of type Linux and of size 110.2 GiB is set
Command (m for help): t
Partition number (1-4, default 4): 3
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
Command (m for help): w
```

Then, restart the servers so that the new table is available.

```
root@vitalpbx-master-slave:~# reboot
```

## 4.5.- Using Script

We have two ways to configure the Cluster. Using the following Script or following this manual step by step. If you decide to use the following Script, the next step you should follow in this manual is 5 if you consider it necessary. Otherwise conti
nue with step 4.6.

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

Create key in Server Master

```
root@vitalpbx-master:~# ssh-keygen -f /root/.ssh/id_rsa -t rsa -N '' >/dev/null
root@vitalpbx-master:~# ssh-copy-id root@192.168.10.32
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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@192.168.10.32'"
and check to make sure that only the key(s) you wanted were added.

root@vitalpbx-master:~#
```

Create key in Server Slave

```
root@vitalpbx-slave:~# ssh-keygen -f /root/.ssh/id_rsa -t rsa -N '' >/dev/null
root@vitalpbx-slave:~# ssh-copy-id root@192.168.10.31
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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@192.168.10.31'"
and check to make sure that only the key(s) you wanted were added.

root@vitalpbx-slave:~#
```

Now copy and run the following script in Master Server.

```
root@vitalpbx-master:~# mkdir /usr/share/vitalpbx/ha
root@vitalpbx-master:~# cd /usr/share/vitalpbx/ha
root@vitalpbx-master:~# wget
https://raw.githubusercontent.com/VitalPBX/vitalpbx4_drbd_ha/main/vpbxha.sh
root@vitalpbx-master:~# chmod +x vpbxha.sh
root@vitalpbx-master:~# ./vpbxha.sh
```

Now we enter all the information requested.

```
************************************************************
*  Welcome to the VitalPBX high availability installation  *
*                All options are mandatory                 *
************************************************************
IP Server1.............. > 192.168.10.31
IP Server2.............. > 192.168.10.32
Floating IP............. > 192.168.10.30
Floating IP Mask (SIDR).. > 24
Disk (sdax)............. > sda3
hacluster password...... > MyPassword
************************************************************
*                   Check Information                      *
*        Make sure you have internet on both servers       *
************************************************************
Are you sure to continue with this settings? (yes,no) > yes
```

Note:

Before doing any high availability testing, make sure that the data has finished syncing. To do this, use the **cat /proc/drbd** command.

CONGRATULATIONS, you have installed high availability in VitalPBX 4

## 4.6.- Firewall

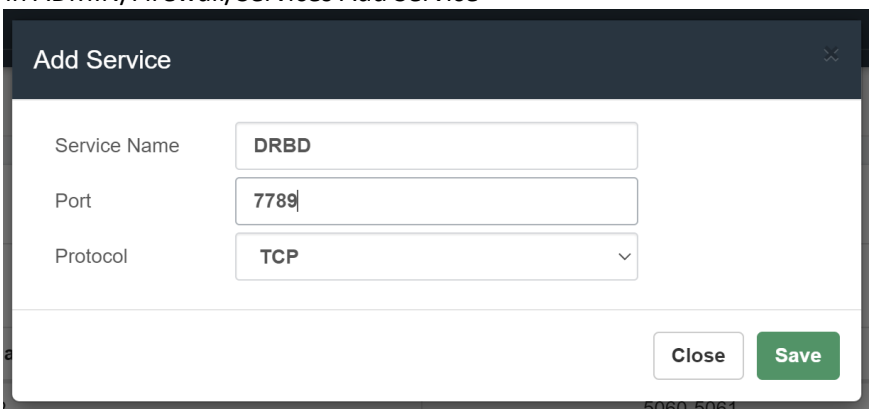
Configure the Firewall (Both Servers)

| Port | Description |
|---|---|
| TCP 2224 | Required on all nodes (needed by the pcsd Web UI and required for node-to-node communication)<br>It is crucial to open port 2224 in such a way that pcs from any node can talk to all nodes in the cluster, including itself. When using the Booth cluster ticket manager or a quorum device you must open port 2224 on all related hosts, such as Booth arbiters or the quorum device host. |
| TCP 3121 | Required on all nodes if the cluster has any Pacemaker Remote nodes<br>Pacemaker's crmd daemon on the full cluster nodes will contact the pacemaker_remoted daemon on Pacemaker Remote nodes at port 3121. If a separate interface is used for cluster communication, the port only needs to be open on that interface. At a minimum, the port should open on Pacemaker Remote nodes to full cluster nodes. Because users may convert a host between a full node and a remote node, or run a remote node inside a container using the host's network, it can be useful to |

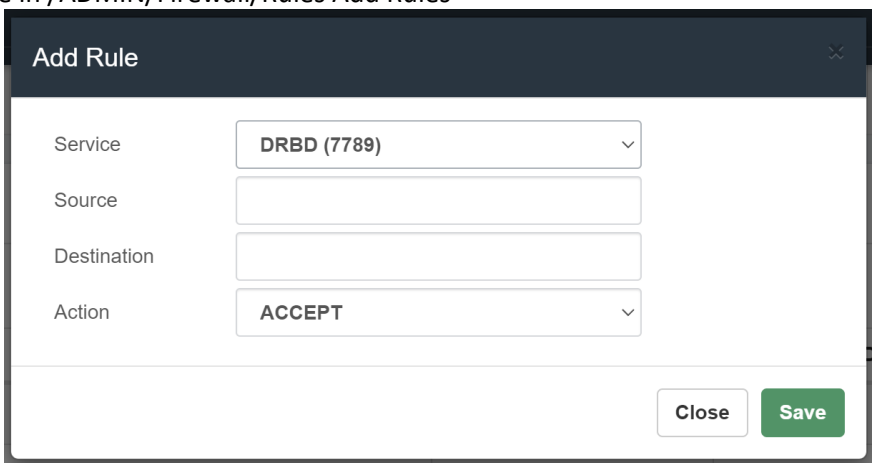
| | |
|---|---|
| | open the port to all nodes. It is not necessary to open the port to any hosts other than nodes. |
| TCP 5403 | Required on the quorum device host when using a quorum device with corosync-qnetd. The default value can be changed with the -p option of the corosync-qnetd command. |
| UDP 5404 | Required on corosync nodes if corosync is configured for multicast UDP. |
| UDP 5405 | Required on all corosync nodes (needed by corosync) |
| TCP 21064 | Required on all nodes if the cluster contains any resources requiring DLM (such as clvm or GFS2) |
| TCP 9929, UDP 9929 | Required to be open on all cluster nodes and booth arbitrator nodes to connections from any of those same nodes when the Booth ticket manager is used to establish a multi-site cluster. |
| TCP 7789 | Required by DRBD to synchronize information. |

Add in both Servers the Service and Rule with this Service.

Add the Service in ADMIN/Firewall/Services Add Service




We add the Rule in /ADMIN/Firewall/Rules Add Rules




And we apply changes.

## 4.7.- Format the partition

Now, we will proceed to format the new partition in both servers with the following command:

```
root@vitalpbx-master-slave:~# mkdir /vpbx_data
root@vitalpbx-master-slave:~# mke2fs -j /dev/sda3
root@vitalpbx-master-slave:~# dd if=/dev/zero bs=1M count=500 of=/dev/sda3; sync
```

## 4.8.- Configuring DRBD

Load the module and enable the service on both nodes, using the follow command:

```
root@vitalpbx-master-slave:~# modprobe drbd
root@vitalpbx-master-slave:~# systemctl enable drbd.service
```

Create a new global_common.conf file on both nodes with the following contents:

```
root@vitalpbx-master-slave:~# mv /etc/drbd.d/global_common.conf
/etc/drbd.d/global_common.conf.orig
root@vitalpbx-master-slave:~# nano /etc/drbd.d/global_common.conf
global {
  usage-count no;
}
  common {
net {
  protocol C;
  }
}
```

Next, we will need to create a new configuration file called /etc/drbd.d/drbd0.res for the new resource named drbd0, with the following contents:

```
root@vitalpbx-master-slave:~# nano /etc/drbd.d/drbd0.res
resource drbd0 {
startup {
        wfc-timeout  5;
        outdated-wfc-timeout 3;
        degr-wfc-timeout 3;
        outdated-wfc-timeout 2;
}
syncer {
        rate 10M;
        verify-alg md5;
}
net {
    after-sb-0pri discard-older-primary;
    after-sb-1pri discard-secondary;
    after-sb-2pri call-pri-lost-after-sb;
}
handlers {
    pri-lost-after-sb "/sbin/reboot";
}
on vitalpbx-master.local {
        device /dev/drbd0;
        disk /dev/$disk;
        address 192.168.10.31:7789;
        meta-disk internal;
}
on vitalpbx-slave.local {
        device /dev/drbd0;
        disk /dev/$disk;
        address 192.168.10.32:7789;
        meta-disk internal;
        }
}
```

Note:
Although the access interfaces can be used, which in this case is ETH0. It is recommended to use an interface (ETH1) for synchronization, this interface must be directly connected between both servers.

Initialize the meta data storage on each nodes by executing the following command on both nodes

```
root@vitalpbx-master-slave:~# drbdadm create-md drbd0
Writing meta data...
New drbd meta data block successfully created.
```

Let's define the DRBD Primary node as first node "vitalpbx-master"

```
root@vitalpbx-master:~# drbdadm up drbd0
root@vitalpbx-master:~# drbdadm primary drbd0 --force
```

On the Secondary node "vitalpbx-slave" run the following command to start the drbd0

```
root@vitalpbx-slave:~# drbdadm up drbd0
```

You can check the current status of the synchronization while it's being performed. The ***cat /proc/drbd*** command displays the creation and synchronization progress of the resource.

### 4.8.1.- Formatting and Test DRBD Disk

In order to test the DRBD functionality we need to Create a file system, mount the volume and write some data on primary node "vitalpbx-master" and finally switch the primary node to "vitalpbx-slave"

Run the following command on the primary node to create an xfs filesystem on /dev/drbd0 and mount it to the mnt directory, using the following commands.

```
root@vitalpbx-master:~# mkfs.xfs /dev/drbd0
root@vitalpbx-master:~# mount /dev/drbd0 /vpbx_data
```

Create some data using the following command:

```
root@vitalpbx-master:~# touch /vpbx_data/file{1..5}
root@vitalpbx-master:~# ls -l /vpbx_data
-rw-r--r-- 1 root root 0 Nov 17 11:28 file1
-rw-r--r-- 1 root root 0 Nov 17 11:28 file2
-rw-r--r-- 1 root root 0 Nov 17 11:28 file3
-rw-r--r-- 1 root root 0 Nov 17 11:28 file4
-rw-r--r-- 1 root root 0 Nov 17 11:28 file5
```

Let's now switch primary mode "vitalpbx-server" to second node "vitalpbx-slave" to check the data replication works or not.
First, we have to unmount the volume drbd0 on the first drbd cluster node "vitalpbx-master" and change the primary node to secondary node on the first drbd cluster node "vitalpbx-master"

```
root@vitalpbx-master:~# umount /vpbx_data
root@vitalpbx-master:~# drbdadm secondary drbd0
```

Change the secondary node to primary node on the second drbd cluster node "vitalpbx-slave"

```
root@vitalpbx-slave:~#  drbdadm primary drbd0 --force
```

Mount the volume and check the data available or not.

```
root@vitalpbx-slave:~# mount /dev/drbd0 /vpbx_data
root@vitalpbx-slave:~# ls -l  /vpbx_data
total 0
-rw-r--r-- 1 root root 0 Nov 17 11:28 file1
-rw-r--r-- 1 root root 0 Nov 17 11:28 file2
-rw-r--r-- 1 root root 0 Nov 17 11:28 file3
-rw-r--r-- 1 root root 0 Nov 17 11:28 file4
-rw-r--r-- 1 root root 0 Nov 17 11:28 file5
```

Normalize Server-Slave

```
root@vitalpbx-slave:~# umount /vpbx_data
root@vitalpbx-slave:~# drbdadm secondary drbd0
```

Normalize Server-Master

```
root@vitalpbx-master:~# drbdadm primary drbd0
root@vitalpbx-master:~# mount /dev/drbd0 /vpbx_data
```

## 4.9.- Configure Cluster

Create the password of the hacluster user on both nodes

```
root@vitalpbx-master-slave:~# echo hacluster:Mypassword | chpasswd
```

Start PCS on both servers

```
root@vitalpbx-master-slave:~# systemctl start pcsd
```

Configure the start of services on both nodes

```
root@vitalpbx-master-slave:~# systemctl enable pcsd.service
root@vitalpbx-master-slave:~# systemctl enable corosync.service
root@vitalpbx-master-slave:~# systemctl enable pacemaker.service
```

Server Authenticate in Master

On vitalpbx-master, use pcs cluster auth to authenticate as the hacluster user.

```
root@vitalpbx-master:~# pcs cluster destroy
root@vitalpbx-master:~# pcs host auth vitalpbx-master.local vitalpbx-slave.local -u hacluster
-p Mypassword

vitalpbx-master.local: Authorized
vitalpbx-slave.local: Authorized
```

Next, use pcs cluster setup on the vitalpbx-master to generate and synchronize the corosync configuration.

```
root@vitalpbx-master:~# pcs cluster setup cluster_vitalpbx vitalpbx-master.local vitalpbx-
slave.local --force
```

Starting Cluster in Master

```
root@vitalpbx-master:~# pcs cluster start --all
root@vitalpbx-master:~# pcs cluster enable --all
root@vitalpbx-master:~# pcs property set stonith-enabled=false
root@vitalpbx-master:~# pcs property set no-quorum-policy=ignore
```

Prevent Resources from Moving after Recovery

In most circumstances, it is highly desirable to prevent healthy resources from being moved around the cluster. Moving resources almost always requires a period of downtime. For complex services such as databases, this period can be quite long.

```
root@vitalpbx-master:~# pcs resource defaults update resource-stickiness=INFINITY
```

Create resource for the use of Floating IP

```
root@vitalpbx-master:~# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.10.30
cidr netmask=24 op monitor interval=30s on-fail=restart
root@vitalpbx-master:~# pcs cluster cib drbd_cfg
root@vitalpbx-master:~# pcs cluster cib-push drbd_cfg --config
```

Create resource for the use of DRBD

```
root@vitalpbx-master:~# pcs cluster cib drbd_cfg
root@vitalpbx-master:~# pcs -f drbd_cfg resource create DrbdData ocf:linbit:drbd
drbd_resource=drbd0 op monitor interval=60s
root@vitalpbx-master:~# pcs -f drbd_cfg resource promotable DrbdData promoted-max=1 promoted-
node-max=1 clone-max=2 clone-node-max=1 notify=true
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push drbd_cfg --config
```

Create FILESYSTEM resource for the automated mount point

```
root@vitalpbx-master:~# pcs cluster cib drbd_cfg
root@vitalpbx-master:~# pcs -f fs_cfg resource create DrbdFS Filesystem device="/dev/drbd0"
directory="/vpbx_data" fstype="xfs"
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add DrbdFS with DrbdData-clone
INFINITY with-rsc-role=Master
root@vitalpbx-master:~# pcs -f fs_cfg constraint order promote DrbdData-clone then start
DrbdFS
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add DrbdFS with ClusterIP INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order DrbdData-clone then DrbdFS
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

Stop and disable all services in both servers

```
root@vitalpbx-master-slave:~# systemctl stop mariadb
root@vitalpbx-master-slave:~# systemctl disable mariadb
root@vitalpbx-master-slave:~# systemctl stop fail2ban
root@vitalpbx-master-slave:~# systemctl disable fail2ban
root@vitalpbx-master-slave:~# systemctl stop asterisk
root@vitalpbx-master-slave:~# systemctl disable asterisk
root@vitalpbx-master-slave:~# systemctl stop vpbx-monitor
root@vitalpbx-master-slave:~# systemctl disable vpbx-monitor
```

Create resource for the use of MariaDB in Master

```
root@vitalpbx-master:~# mkdir /vpbx_data/mysql
root@vitalpbx-master:~# mkdir /vpbx_data/mysql/data
root@vitalpbx-master:~# cp -aR /var/lib/mysql/* /vpbx_data/mysql/data
root@vitalpbx-master:~# chown -R mysql:mysql /vpbx_data/mysql
root@vitalpbx-master-slave:~# sed -i 's/var\/lib\/mysql/vpbx_data\/mysql\/data/g'
/etc/mysql/mariadb.conf.d/50-server.cnf

root@vitalpbx-master:~# pcs resource create mysql service:mariadb op monitor interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add mysql with ClusterIP INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order DrbdFS then mysql
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

Path Asterisk service in both servers

```
root@vitalpbx-master-slave:~# sed -i 's/RestartSec=10/RestartSec=1/g'
/usr/lib/systemd/system/asterisk.service
root@vitalpbx-master-slave:~# sed -i 's/Wants=mariadb.service/#Wants=mariadb.service/g'
/usr/lib/systemd/system/asterisk.service
root@vitalpbx-master-slave:~# sed -i 's/After=mariadb.service/#After=mariadb.service/g'
/usr/lib/systemd/system/asterisk.service
```

Create resource for Asterisk

```
root@vitalpbx-master:~# pcs resource create asterisk service:asterisk op monitor interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add asterisk with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order mysql then asterisk
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs resource update asterisk op stop timeout=120s
root@vitalpbx-master:~# pcs resource update asterisk op start timeout=120s
root@vitalpbx-master:~# pcs resource update asterisk op restart timeout=120s
```

Copy folders and files the DRBD partition on the server1

```
root@vitalpbx-master:~# tar -zcvf var-asterisk.tgz /var/log/asterisk
root@vitalpbx-master:~# tar -zcvf var-lib-asterisk.tgz /var/lib/asterisk
root@vitalpbx-master:~# tar -zcvf var-lib-vitalpbx.tgz /var/lib/vitalpbx
root@vitalpbx-master:~# tar -zcvf etc-vitalpbx.tgz /etc/vitalpbx
root@vitalpbx-master:~# tar -zcvf usr-lib-asterisk.tgz /usr/lib/asterisk
root@vitalpbx-master:~# tar -zcvf var-spool-asterisk.tgz /var/spool/asterisk
root@vitalpbx-master:~# tar -zcvf etc-asterisk.tgz /etc/asterisk

root@vitalpbx-master:~# tar xvfz var-asterisk.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz var-lib-asterisk.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz var-lib-vitalpbx.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz etc-vitalpbx.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz usr-lib-asterisk.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz var-spool-asterisk.tgz -C /vpbx_data
root@vitalpbx-master:~# tar xvfz etc-asterisk.tgz -C /vpbx_data
root@vitalpbx-master:~# chmod -R 775 /vpbx_data/var/log/asterisk

root@vitalpbx-master:~# rm -rf /var/log/asterisk
root@vitalpbx-master:~# rm -rf /var/lib/asterisk
root@vitalpbx-master:~# rm -rf /var/lib/vitalpbx
root@vitalpbx-master:~# rm -rf /etc/vitalpbx
root@vitalpbx-master:~# rm -rf /usr/lib/asterisk
root@vitalpbx-master:~# rm -rf /var/spool/asterisk
root@vitalpbx-master:~# rm -rf /etc/asterisk

root@vitalpbx-master:~# ln -s /vpbx_data/var/log/asterisk /var/log/asterisk
root@vitalpbx-master:~# ln -s /vpbx_data/var/lib/asterisk /var/lib/asterisk
root@vitalpbx-master:~# ln -s /vpbx_data/var/lib/vitalpbx /var/lib/vitalpbx
root@vitalpbx-master:~# ln -s /vpbx data/etc/vitalpbx /etc/vitalpbx
root@vitalpbx-master:~# ln -s /vpbx_data/usr/lib/asterisk /usr/lib/asterisk
root@vitalpbx-master:~# ln -s /vpbx_data/var/spool/asterisk /var/spool/asterisk
root@vitalpbx-master:~# ln -s /vpbx_data/etc/asterisk /etc/asterisk

root@vitalpbx-master:~# rm -rf var-asterisk.tgz
root@vitalpbx-master:~# rm -rf var-lib-asterisk.tgz
root@vitalpbx-master:~# rm -rf var-lib-vitalpbx.tgz
root@vitalpbx-master:~# rm -rf etc-vitalpbx.tgz
root@vitalpbx-master:~# rm -rf usr-lib-asterisk.tgz
root@vitalpbx-master:~# rm -rf var-spool-asterisk.tgz
root@vitalpbx-master:~# rm -rf etc-asterisk.tgz
```

Configure symbolic links on the server-slave

```
root@vitalpbx-slave:~#  rm -rf /var/log/asterisk
root@vitalpbx-slave:~#  rm -rf /var/lib/asterisk
root@vitalpbx-slave:~#  rm -rf /var/lib/vitalpbx
root@vitalpbx-slave:~#  rm -rf /etc/vitalpbx
root@vitalpbx-slave:~#  rm -rf /usr/lib/asterisk
root@vitalpbx-slave:~#  rm -rf /var/spool/asterisk
root@vitalpbx-slave:~#  rm -rf /etc/asterisk

root@vitalpbx-slave:~#  ln -s /vpbx_data/var/log/asterisk /var/log/asterisk
root@vitalpbx-slave:~#  ln -s /vpbx_data/var/lib/asterisk /var/lib/asterisk
root@vitalpbx-slave:~#  ln -s /vpbx_data/var/lib/vitalpbx /var/lib/vitalpbx
root@vitalpbx-slave:~#  ln -s /vpbx_data/etc/vitalpbx /etc/vitalpbx
root@vitalpbx-slave:~#  ln -s /vpbx_data/usr/lib/asterisk /usr/lib/asterisk
root@vitalpbx-slave:~#  ln -s /vpbx_data/var/spool/asterisk /var/spool/asterisk
root@vitalpbx-slave:~#  ln -s /vpbx_data/etc/asterisk /etc/asterisk
```

Create VitalPBX Service

```
root@vitalpbx-master:~# pcs resource create vpbx-monitor service:vpbx-monitor op monitor
interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add vpbx-monitor with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order asterisk then vpbx-monitor
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

Create fail2ban Service

```
root@vitalpbx-master:~# pcs resource create fail2ban service:fail2ban op monitor interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add fail2ban with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order asterisk then fail2ban
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

Note:
All configuration is stored in the file: /var/lib/pacemaker/cib/cib.xml
Show the Cluster Status

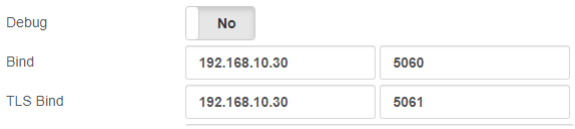
```
root@vitalpbx-master:~# pcs status resources
  * ClusterIP  (ocf::heartbeat:IPaddr2):        Started vitalpbx-master.local
  * Clone Set: DrbdData-clone [DrbdData] (promotable):
    * Masters: [ vitalpbx-master.local ]
    * Slaves: [ vitalpbx-slave.local ]
  * DrbdFS      (ocf::heartbeat:Filesystem):     Started vitalpbx-master.local
  * mysql       (ocf::heartbeat:mysql):  Started vitalpbx-master.local
  * asterisk    (service:asterisk):      Started vitalpbx-master.local
  * vpbx-monitor        (service:vpbx-monitor):  Started vitalpbx-master.local
  * fail2ban    (service:fail2ban):      Started vitalpbx-master.local
```

Note:
Before doing any high availability testing, make sure that the data has finished syncing. To do this, use the cat/proc/drbd command.

## 4.10.- Bind Address

Managing the bind address also is critical if you use multple IP addresses on the same NIC [as, for example, when using a floating IP address in an HA cluster]. In that circumstance, asterisk has a rather nasty habit of listening for SIP/IAX on the virtual IP address but replying on the base address of the NIC causing phones/trunks to fail to register.

In the Master server go to SETTINGS/PJSIP Settings and configure the Floating IP that we are going to use in "Bind" and "TLS Bind". Also do it in SETTINGS/SIP Settings Tab NETWORK fields "TCP Bind Address" and "TLS Bind Address".




## 4.11.- Create “bascul” command in both servers

The bascul command permanently moves services from one server to another. If you want to return the services to the main server you must execute the command again.
Download file

```
root@vitalpbx-master-slave:~#  wget
https://raw.githubusercontent.com/VitalPBX/vitalpbx4_drbd_ha/main/bascul
```

Or create file

```
root@vitalpbx-master-slave:~# nano bascul
```

Paste

```
#!/bin/bash
# This code is the property of VitalPBX LLC Company
# License: Proprietary
# Date: 1-Agu-2023
# Change the status of the servers, the Master goes to Stanby and the Standby goes to Master.
#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
set -e
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==11 {print $4}'`
server_b=`pcs status | awk 'NR==11 {print $5}'`
server_master=`pcs status resources | awk 'NR==1 {print $5}'`
#Perform some validations
if [ "${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
arg=$1
if [ "$arg" = 'yes' ] ;then
        perform_bascul='yes'
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 node unstandby $host_master
        pcs node unstandby $host_standby
        #Do a loop per resource
        pcs status resources | awk '{print $2}' | while read -r resource ; do
                #Skip moving the virutal_ip resource, it will be moved at the end
                if [[ "${resource}" != "ClusterIP" ]] && [[ "${resource}" != "Clone" ]] && [[
"${resource}" != "Masters:" ]] && [[ "${resource}" != "Slaves:" ]]; then
                        echo "Moving ${resource} from ${host_master} to ${host_standby}"
                        pcs resource move ${resource} ${host_standby}
                fi
```

```
        done
        sleep 5 && pcs node standby $host_master & #Standby current Master node after five
seconds
        sleep 20 && pcs node unstandby $host_master & #Automatically Unstandby current Master
node after$
        #Move the ClusterIP resource to standby node
        echo "Moving ClusterIP from ${host_master} to ${host_standby}"
        pcs resource move ClusterIP ${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 15
role
```

### Add permissions and move to folder /usr/local/bin

```
root@vitalpbx-master-slave:~# chmod +x bascul
root@vitalpbx-master-slave:~# mv bascul /usr/local/bin
```

## 4.12.- Create “role” command in both servers

### Download file

```
root@vitalpbx-master-slave:~#  wget
https://raw.githubusercontent.com/VitalPBX/vitalpbx4_drbd_ha/main/role
```

### Or create file

```
root@vitalpbx-master-slave:~# nano role
```

### Paste

```
#!/bin/bash
# This code is the property of VitalPBX LLC Company
# License: Proprietary
# Date: 8-Agu-2023
# Show the Role of Server.
#Bash Colour Codes
green="\033[00;32m"
yellow="\033[00;33m"
txtrst="\033[00;0m"
linux_ver=`cat /etc/os-release | grep -e PRETTY_NAME | awk -F '=' '{print $2}' | xargs`
vpbx_version=`aptitude versions vitalpbx | awk '{ print $2 }'`
server_master=`pcs status resources | awk 'NR==1 {print $5}'`
host=`hostname`
if [[ "${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 -rx "core show version" 2>/dev/null| grep -ohe 'Asterisk [0-9.]*'`
```

```
 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 service/ {print $2}'`
"
echo -e ""
echo -e "************************************************************"
if [[ "${server_mode}" = "Master" ]]; then
echo -e "*                Server Status: ${green}$server_mode${txtrst}                    *"
else
echo -e "*                Server Status: ${yellow}$server_mode${txtrst}                     *"
fi
echo -e "************************************************************"
pcs status resources
echo -e "************************************************************"
echo -e ""
echo -e "Servers Status"
pcs cluster pcsd-status
```

Add permissions and copy to folder /etc/profile.d/

```
root@vitalpbx-master-slave:~# cp -rf role /etc/profile.d/vitalwelcome.sh
root@vitalpbx-master-slave:~# chmod 755 /etc/profile.d/vitalwelcome.sh
```

Now add permissions and move to folder /usr/local/bin

```
root@vitalpbx-master-slave:~# chmod +x role
root@vitalpbx-master-slave:~# mv role /usr/local/bin
```

# 5.- Add New Services

## 5.1.- Add Sonata Switchboard

To install Sonata Switchboard make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Sonata Switchboard and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf switchboard-settings.tgz /etc/switchboard/settings.conf
root@vitalpbx-master:~# tar xvfz switchboard-settings.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /etc/switchboard/settings.conf
root@vitalpbx-master:~# ln -s /vpbx_data/etc/switchboard/settings.conf
/etc/switchboard/settings.conf
root@vitalpbx-master:~# rm -rf switchboard-settings.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Install Sonata Switchboard

7.- Create the virtual link from the settings.conf file.

```
root@vitalpbx-slave:~# rm -rf /etc/switchboard/settings.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/etc/switchboard/settings.conf
/etc/switchboard/settings.conf
```

8.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

## 5.2.- Add Sonata Recording

To install Sonata Recording make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Sonata Recording and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf recording-settings.tgz /etc/rec-manager/settings.conf
root@vitalpbx-master:~# tar xvfz recording-settings.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /etc/rec-manager/settings.conf
root@vitalpbx-master:~# ln -s /vpbx_data/etc/rec-manager/settings.conf  /etc/rec-
manager/settings.conf
root@vitalpbx-master:~# rm -rf recording-settings.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Install Sonata Recording

7.- Create the virtual link from the settings.conf file.

```
root@vitalpbx-slave:~# rm -rf /etc/rec-manager/settings.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/etc/rec-manager/settings.conf  /etc/rec-
manager/settings.conf
```

8.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

9.- We added the Sonata Recordings automation service.

```
root@vitalpbx-master-slave:~# systemctl stop recordings
root@vitalpbx-master-slave:~# systemctl disable recordings
root@vitalpbx-master:~# pcs resource create sonata-recordings service:recordings op monitor
interval=30s
```

```
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add sonata-recordings with
ClusterIP INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then sonata-recordings
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

## 5.3.- Add Sonata Billing

To install Sonata Billing make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Sonata Billing and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf billing-settings.tgz /etc/billing/settings.conf
root@vitalpbx-master:~# tar xvfz billing-settings.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /etc/billing/settings.conf
root@vitalpbx-master:~# ln -s /vpbx_data/etc/billing/settings.conf  /etc/billing/settings.conf
root@vitalpbx-master:~# rm -rf billing-settings.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Install Sonata Billing

7.- Create the virtual link from the settings.conf file.

```
root@vitalpbx-slave:~# rm -rf /etc/billing/settings.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/etc/billing/settings.conf  /etc/billing/settings.conf
```

8.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

## 5.4.- Add Sonata Stats

To install Sonata Stats make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Sonata Stats and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf stats-env.tgz /usr/share/queues-stats/backend/.env
root@vitalpbx-master:~# tar -zcvf stats-wizard.tgz /var/lib/sonata/stats/wizard.conf
root@vitalpbx-master:~# tar xvfz stats-env.tgz -C /vpbx_data/
root@vitalpbx-master:~# tar xvfz stats-wizard.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /usr/share/queues-stats/backend/.env
root@vitalpbx-master:~# ln -s /vpbx_data/usr/share/queues-stats/backend/.env
/usr/share/queues-stats/backend/.env
root@vitalpbx-master:~# rm -rf /var/lib/sonata/stats/wizard.conf
root@vitalpbx-master:~# ln -s /vpbx_data/var/lib/sonata/stats/wizard.conf
/var/lib/sonata/stats/wizard.conf
root@vitalpbx-master:~# rm -rf stats-env.tgz
root@vitalpbx-master:~# rm -rf stats-wizard.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Create the virtual link from the .env file.

```
root@vitalpbx-slave:~# rm -rf /usr/share/queues-stats/backend/.env
root@vitalpbx-slave:~# ln -s /vpbx_data/usr/share/queues-stats/backend/.env /usr/share/queues-
stats/backend/.env
root@vitalpbx-slave:~# rm -rf /var/lib/sonata/stats/wizard.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/var/lib/sonata/stats/wizard.conf
/var/lib/sonata/stats/wizard.conf
```

```
root@vitalpbx-slave:~# cd /usr/share/queues-stats/backend/
root@vitalpbx-slave:~# php artisan config:cache && php artisan route:cache && php artisan
view:cache
```

7.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

8.- We added the Sonata Stats automation service.

```
root@vitalpbx-master-slave:~# systemctl stop sonata-stats
root@vitalpbx-master-slave:~# systemctl disable sonata-stats
root@vitalpbx-master:~# pcs resource create sonata-stats service:sonata-stats op monitor
interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add sonata-stats with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then sonata-stats
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

## 5.5.- Add Sonata Dialer

To install Sonata Dialer make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Sonata Dialer and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf dialer-env.tgz /usr/share/dialer/backend/.env
root@vitalpbx-master:~# tar -zcvf dialer-wizard.tgz /var/lib/sonata/dialer/wizard.conf
root@vitalpbx-master:~# tar xvfz dialer-env.tgz -C /vpbx_data/
root@vitalpbx-master:~# tar xvfz dialer-wizard.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /usr/share/dialer/backend/.env
root@vitalpbx-master:~# ln -s /vpbx_data/usr/share/dialer/backend/.env
/usr/share/dialer/backend/.env
root@vitalpbx-master:~# rm -rf /var/lib/sonata/dialer/wizard.conf
root@vitalpbx-master:~# ln -s /vpbx_data/var/lib/sonata/dialer/wizard.conf
/var/lib/sonata/dialer/wizard.conf
root@vitalpbx-master:~# rm -rf dialer-env.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

7.- Create the virtual link from the .env file.

```
root@vitalpbx-slave:~# rm -rf /usr/share/dialer/backend/.env
root@vitalpbx-slave:~# ln -s /vpbx_data/usr/share/dialer/backend/.env
/usr/share/dialer/backend/.env
root@vitalpbx-slave:~# rm -rf /var/lib/sonata/dialer/wizard.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/var/lib/sonata/dialer/wizard.conf
/var/lib/sonata/dialer/wizard.conf
root@vitalpbx-slave:~# cd /usr/share/dialer/backend/
root@vitalpbx-slave:~# php artisan config:cache && php artisan route:cache && php artisan
view:cache
```

8.- We added the Sonata Dialer automation service.

```
root@vitalpbx-master-slave:~# systemctl stop sonata-dialer
root@vitalpbx-master-slave:~# systemctl disable sonata-dialer
root@vitalpbx-master:~# pcs resource create sonata-dialer service:sonata-dialer op monitor
interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add sonata-dialer with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then sonata-dialer
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

## 5.6.- Add VitXi

To install VitXi make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install VitXi and execute the Wizard.

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf vitxi-env.tgz /usr/share/vitxi/backend/.env
root@vitalpbx-master:~# tar -zcvf vitxi-storage.tgz /usr/share/vitxi/backend/storage
root@vitalpbx-master:~# tar -zcvf vitxi-wizard.tgz /var/lib/vitxi/wizard.conf
root@vitalpbx-master:~# tar xvfz vitxi-env.tgz -C /vpbx_data/
root@vitalpbx-master:~# tar xvfz vitxi-storage.tgz -C /vpbx_data/
root@vitalpbx-master:~# tar xvfz vitxi-wizard.tgz -C /vpbx_data/

root@vitalpbx-master:~# rm -rf /usr/share/vitxi/backend/.env
root@vitalpbx-master:~# rm -rf /usr/share/vitxi/backend/storage
root@vitalpbx-master:~# rm -rf /var/lib/vitxi/wizard.conf

root@vitalpbx-master:~# ln -s /vpbx_data/usr/share/vitxi/backend/.env
/usr/share/vitxi/backend/.env
root@vitalpbx-master:~# ln -s /vpbx_data/usr/share/vitxi/backend/storage
/usr/share/vitxi/backend/storage
root@vitalpbx-master:~# ln -s /vpbx_data/var/lib/vitxi/wizard.conf  /var/lib/vitxi/wizard.conf

root@vitalpbx-master:~# rm -rf vitxi-env.tgz
root@vitalpbx-master:~# rm -rf vitxi-storage.tgz
root@vitalpbx-master:~# rm -rf vitxi-wizard.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

5.- From your browser, go to ip 192.168.10.30 again

6.- Install VitXi

7.- Create the virtual link from the .env file.

```
root@vitalpbx-slave:~# rm -rf /usr/share/vitxi/backend/.env
root@vitalpbx-slave:~# ln -s /vpbx_data/usr/share/vitxi/backend/.env
/usr/share/vitxi/backend/.env
root@vitalpbx-slave:~# rm -rf /var/lib/vitxi/wizard.conf
root@vitalpbx-slave:~# ln -s /vpbx_data/var/lib/vitxi/wizard.conf  /var/lib/vitxi/wizard.conf
root@vitalpbx-slave:~# rm -rf /usr/share/vitxi/backend/storage
root@vitalpbx-slave:~# ln -s /vpbx_data/usr/share/vitxi/backend/storage
/usr/share/vitxi/backend/storage

root@vitalpbx-slave:~# cd /usr/share/vitxi/backend/
root@vitalpbx-slave:~# php artisan config:cache && php artisan route:cache && php artisan
view:cache
```

8.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

9.- We added the VitXi automation service.

```
root@vitalpbx-master-slave:~# systemctl stop vitxi
root@vitalpbx-master-slave:~# systemctl disable vitxi
root@vitalpbx-master:~# pcs resource create vitxi service:vitxi op monitor interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add vitxi with ClusterIP INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then vitxi
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

## 5.7.- Add OpenVPN

To install OpenVPN make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install OpenVPN

3.- Copy the files that are going to be in the replica.

```
root@vitalpbx-master:~# tar -zcvf etc-openvpn.tgz /etc/openvpn
root@vitalpbx-master:~# tar xvfz etc-openvpn.tgz -C /vpbx_data/
root@vitalpbx-master:~# rm -rf /etc/openvpn
root@vitalpbx-master:~# ln -s /vpbx_data/etc/openvpn /etc/openvpn
root@vitalpbx-master:~# rm -rf etc-openvpn.tgz
```

4.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

4.- From your browser, go to ip 192.168.10.30 again

5.- Install OpenVPN

6.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

7.- Create the virtual link from OpenVPN path.

```
root@vitalpbx-slave:~# rm -rf /etc/openvpn
root@vitalpbx-slave:~# ln -s /vpbx data/etc/openvpn /etc/openvpn
```

8.- We added the OpenVPN automation service.

```
root@vitalpbx-master-slave:~# systemctl stop openvpn
root@vitalpbx-master-slave:~# systemctl disable openvpn
root@vitalpbx-master:~# pcs resource create openvpn service:openvpn op monitor interval=30s
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add openvpn with ClusterIP
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then openvpn
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg --config
```

## 5.8.- Add Maintenance

To install Maintenance make sure **vitalpbx-master.local** is set to Master. And follow the procedure below.

1.- From your browser, go to ip 192.168.10.30

2.- Install Maintenance module.

3.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

4.- From your browser, go to ip 192.168.10.30 again

5.- Install Maintenance module.

6.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

7.- Because the maintenance cron file is copied to a shared folder (/etc/cron.d/vpbx_maintenace) that has other system cron services active, we recommend first creating the maintenance on the master server and then bascul and create the same maintenance on the slave server.

Because the databases are synchronized, it is only necessary to go to ADMIN/Tools/Maintenance and press **Save** and **Apply Changes**

# 6.- Test

To execute the process of changing the role, we recommend using the following command:

```
root@vitalpbx-master-slave:~# 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 vitalpbx-master.local to vitalpbx-slave.local? (yes,no) >
```

This action convert the vitalpbx-master.local to Slave and vitalpbx-slave.local to Master. If you want to return to default do the same again.

If you want to know the current state of High Availability from a phone, please create the following context.

```
root@vitalpbx-master:~# nano /etc/asterisk/vitalpbx/extensions__61-ha-test.conf
[cos-all](+)
exten => *777,1,NoOp(Say Asterisk IP Address)
 same => n,Answer()
 same => n,Set(ASTERISK_IP=${SHELL(/usr/bin/hostname -I | awk -F " " '{print $1}'`)})
 same => n,NoOp(IP Address: ${ASTERISK_IP})
 same => n,SayDigits(${CUT(ASTERISK_IP,.,1)})
 same => n,Playback(letters/dot)
 same => n,SayDigits(${CUT(ASTERISK_IP,.,2)})
 same => n,Playback(letters/dot)
 same => n,SayDigits(${CUT(ASTERISK_IP,.,3)})
 same => n,Playback(letters/dot)
 same => n,SayDigits(${CUT(ASTERISK_IP,.,4)})
 same => n,PlayBack(silence/1&vm-goodbye)
 same => n,Hangup()
```

Now for the context to be used we must restart the Asterisk dial plan.

```
root@vitalpbx-master:~# asterisk -rx"dialplan reload"
```

To try just dial *777

Warning Note ⚠

When you must turn off the servers, when you turn it on always try to start with the Master, wait for the Master to start and then turn on the Slave.

# 7.- Update VitalPBX or Add-Ons

To update VitalPBX to the latest version just follow the following steps:
1.- From your browser, go to ip 192.168.10.30
2.- Update VitalPBX from the interface
3.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

4.- From your browser, go to ip 192.168.10.30 again
5.- Update VitalPBX from the interface
6.- Execute the following command in Master console

```
root@vitalpbx-master:~# bascul
```

**CONGRATULATIONS, you have installed and tested the high availability in VitalPBX**

# 8.- Changing one of the Servers

Sometimes it is necessary to change one of the servers either because it is damaged or because of some hardware modification.
For this it is necessary to follow the following procedure.

## 8.1.- Changing the secondary Server

The method to change the secondary server differs a bit from the primary server method, here are the steps to change the secondary server.

First we proceed to destroy the cluster in **master-vitalpbx.local**

```
root@vitalpbx-master:~# pcs cluster stop
root@vitalpbx-master:~# pcs cluster destroy
root@vitalpbx-master:~# systemctl disable pcsd.service
root@vitalpbx-master:~# systemctl disable corosync.service
root@vitalpbx-master:~# systemctl disable pacemaker.service
root@vitalpbx-master:~# systemctl stop pcsd.service
root@vitalpbx-master:~# systemctl stop corosync.service
root@vitalpbx-master:~# systemctl stop pacemaker.service
```

Now, if we still have access to **vitalpbx-slave.local**, we proceed to destroy the cluster as well.

```
root@vitalpbx-slave:~# pcs cluster stop --force
root@vitalpbx-slave:~# pcs cluster destroy
root@vitalpbx-slave:~# systemctl disable pcsd.service
root@vitalpbx-slave:~# systemctl disable corosync.service
root@vitalpbx-slave:~# systemctl disable pacemaker.service
root@vitalpbx-slave:~# systemctl stop pcsd.service
root@vitalpbx-slave:~# systemctl stop corosync.service
root@vitalpbx-slave:~# systemctl stop pacemaker.service
```

Because when destroying the Cluster the DRBD unit is unmounted, in **vitalpbx-master.local** we must mount it again manually to avoid interrupting the normal operation of our services.

```
root@vitalpbx-master:~# drbdadm up drbd0
root@vitalpbx-master:~# drbdadm primary drbd0 --force
root@vitalpbx-master:~# mount /dev/drbd0 /vpbx data
```

Now enable the services to make sure our server continues to function normally.

```
root@vitalpbx-master:~# systemctl enable asterisk
root@vitalpbx-master:~# systemctl restart asterisk
root@vitalpbx-master:~# systemctl enable mariadb
root@vitalpbx-master:~# systemctl restart mariadb
root@vitalpbx-master:~# systemctl enable fail2ban
root@vitalpbx-master:~# systemctl restart fail2ban
root@vitalpbx-master:~# systemctl enable vpbx-monitor
root@vitalpbx-master:~# systemctl restart vpbx-monitor
```

Our **vitalpbx-master.local** server is now working normally. Now we proceed to configure the replica with our new server.

First we prepare our new server following the steps in section 3, 4.1, 4.2, 4.3, 4.4, 4.6. It should only be configured in **vitalpbx-slave.local**.

Now, we will proceed to format the new partition on **vitalpbx-slave.local** with the following command:

```
root@vitalpbx-slave:~# mkdir /vpbx_data
root@vitalpbx-slave:~# mke2fs -j /dev/sda3
root@vitalpbx-slave:~# dd if=/dev/zero bs=1M count=500 of=/dev/sda3; sync
```

Load the module and enable the service on **vitalpbx-slave.local**, using the follow command:

```
root@vitalpbx-slave:~# modprobe drbd
root@vitalpbx-slave:~# systemctl enable drbd.service
```

Create a new global_common.conf file on **vitalpbx-slave.local** with the following contents:

```
root@vitalpbx-slave:~# mv /etc/drbd.d/global_common.conf /etc/drbd.d/global_common.conf.orig
root@vitalpbx-slave:~# nano /etc/drbd.d/global_common.conf
global {
  usage-count no;
}
  common {
net {
  protocol C;
  }
}
```

Next, we will need to create on **vitalpbx-slave.local** a new configuration file called /etc/drbd.d/drbd0.res for the new resource named drbd0, with the following contents:

```
root@vitalpbx-slave:~# nano /etc/drbd.d/drbd0.res
resource drbd0 {
startup {
        wfc-timeout  5;
        outdated-wfc-timeout 3;
        degr-wfc-timeout 3;
        outdated-wfc-timeout 2;
}
syncer {
        rate 10M;
        verify-alg md5;
}
net {
    after-sb-0pri discard-older-primary;
    after-sb-1pri discard-secondary;
    after-sb-2pri call-pri-lost-after-sb;
}
handlers {
    pri-lost-after-sb "/sbin/reboot";
}
on vitalpbx-master.local {
        device /dev/drbd0;
        disk /dev/$disk;
```

```
        address 192.168.10.31:7789;
        meta-disk internal;
}
on vitalpbx-slave.local {
        device /dev/drbd0;
        disk /dev/$disk;
        address 192.168.10.32:7789;
        meta-disk internal;
        }
}
```

Note:
Although the access interfaces can be used, which in this case is ETH0. It is recommended to use an interface (ETH1) for synchronization, this interface must be directly connected between both servers.

Initialize the meta data storage on each nodes by executing the following command on **vitalpbx-slave.local**.

```
root@vitalpbx-slave:~# drbdadm create-md drbd0
Writing meta data...
New drbd meta data block successfully created.
```

On the **vitalpbx-slave.local** run the following command to start the drbd0

```
root@vitalpbx-slave:~# drbdadm up drbd0
```

You can check the current status of the synchronization while it's being performed. The ***cat /proc/drbd*** command displays the creation and synchronization progress of the resource.

Now follow steps 4.9, 4.11, 4.12, 4.13.

**CONGRATULATIONS, you have updated the high availability in VitalPBX**

## 8.2.- Changing the primary Server

The method to change the primary server differs a bit from the secundary server method, here are the steps to change the primary server.

First we proceed to destroy the cluster in **master-vitalpbx.local**

```
root@vitalpbx-master:~# pcs cluster stop
root@vitalpbx-master:~# pcs cluster destroy
root@vitalpbx-master:~# systemctl disable pcsd.service
root@vitalpbx-master:~# systemctl disable corosync.service
root@vitalpbx-master:~# systemctl disable pacemaker.service
root@vitalpbx-master:~# systemctl stop pcsd.service
root@vitalpbx-master:~# systemctl stop corosync.service
root@vitalpbx-master:~# systemctl stop pacemaker.service
```

Now, if we still have access to **vitalpbx-slave.local**, we proceed to destroy the cluster as well.

```
root@vitalpbx-slave:~# pcs cluster stop --force
root@vitalpbx-slave:~# pcs cluster destroy
root@vitalpbx-slave:~# systemctl disable pcsd.service
root@vitalpbx-slave:~# systemctl disable corosync.service
root@vitalpbx-slave:~# systemctl disable pacemaker.service
root@vitalpbx-slave:~# systemctl stop pcsd.service
root@vitalpbx-slave:~# systemctl stop corosync.service
root@vitalpbx-slave:~# systemctl stop pacemaker.service
```

Because when destroying the Cluster the DRBD unit is unmounted, in **vitalpbx-slave.local** we must mount it again manually to avoid interrupting the normal operation of our services.

```
root@vitalpbx-slave:~# drbdadm up drbd0
root@vitalpbx-slave:~# drbdadm primary drbd0 --force
root@vitalpbx-slave:~# mount /dev/drbd0 /vpbx_data
```

Now enable the services to make sure our server continues to function normally.

```
root@vitalpbx-slave:~# systemctl enable asterisk
root@vitalpbx-slave:~# systemctl restart asterisk
root@vitalpbx-slave:~# systemctl enable mariadb
root@vitalpbx-slave:~# systemctl restart mariadb
root@vitalpbx-slave:~# systemctl enable fail2ban
root@vitalpbx-slave:~# systemctl restart fail2ban
root@vitalpbx-slave:~# systemctl enable vpbx-monitor
root@vitalpbx-slave:~# systemctl restart vpbx-monitor
```

Our **vitalpbx-slave.local** server is now working normally. Now we proceed to configure the replica with our new server.

First we prepare our new server following the steps in section 3, 4.1, 4.2, 4.3, 4.4, 4.6. It should only be configured in **vitalpbx-slave.local**.

Now, we will proceed to format the new partition on **vitalpbx-master.local** with the following command:

```
root@vitalpbx-master:~# mkdir /vpbx data
root@vitalpbx-master:~# mke2fs -j /dev/sda3
root@vitalpbx-master:~# dd if=/dev/zero bs=1M count=500 of=/dev/sda3; sync
```

Load the module and enable the service on **vitalpbx-master.local**, using the follow command:

```
root@vitalpbx-master:~# modprobe drbd
root@vitalpbx-master:~# systemctl enable drbd.service
```

Create a new global_common.conf file on **vitalpbx-slave.local** with the following contents:

```
root@vitalpbx-master:~# mv /etc/drbd.d/global_common.conf /etc/drbd.d/global_common.conf.orig
root@vitalpbx-master:~# nano /etc/drbd.d/global_common.conf
global {
  usage-count no;
}
  common {
net {
  protocol C;
  }
}
```

Next, we will need to create on **vitalpbx-master.local** a new configuration file called /etc/drbd.d/drbd0.res for the new resource named drbd0, with the following contents:

```
root@vitalpbx-master:~# nano /etc/drbd.d/drbd0.res
resource drbd0 {
startup {
        wfc-timeout  5;
        outdated-wfc-timeout 3;
        degr-wfc-timeout 3;
        outdated-wfc-timeout 2;
}
syncer {
        rate 10M;
        verify-alg md5;
}
net {
    after-sb-0pri discard-older-primary;
    after-sb-1pri discard-secondary;
```

```
    after-sb-2pri call-pri-lost-after-sb;
}
handlers {
    pri-lost-after-sb "/sbin/reboot";
}
on vitalpbx-master.local {
        device /dev/drbd0;
        disk /dev/$disk;
        address 192.168.10.31:7789;
        meta-disk internal;
}
on vitalpbx-slave.local {
        device /dev/drbd0;
        disk /dev/$disk;
        address 192.168.10.32:7789;
        meta-disk internal;
        }
}
```

Note:
Although the access interfaces can be used, which in this case is ETH0. It is recommended to use an interface (ETH1) for synchronization, this interface must be directly connected between both servers.

Initialize the meta data storage on each nodes by executing the following command on **vitalpbx-master.local**.

```
root@vitalpbx-master:~# drbdadm create-md drbd0
Writing meta data...
New drbd meta data block successfully created.
```

On the **vitalpbx-slave.local** run the following command to start the drbd0

```
root@vitalpbx-master:~# drbdadm up drbd0
```

You can check the current status of the synchronization while it's being performed. The ***cat /proc/drbd*** command displays the creation and synchronization progress of the resource.

Now follow steps 4.9, 4.11, 4.12, 4.13.

**CONGRATULATIONS, you have updated the high availability in VitalPBX**

# 9.- Some useful commands

- **bascul**, is used to change roles between high availability servers. If all is well, a confirmation question should appear if we wish to execute the action. The bascul command permanently moves services from one server to another. If you want to return the services to the main server you must execute the command again.
- **role**, shows the status of the current server. If all is well you should return Masters or Slaves.
- **pcs resource refresh --full**, to poll all resources even if the status is unknown, enter the following command.
- **pcs node standby host**, Stops the server node where it is running. The node status is shown as stopped.
- **pcs node unstandby host**, in some cases the bascul command does not finish tilting, which causes one of the servers to be in standby (stop), with this command the state is restored to normal.
- **pcs resource delete**, removes the resource so it can be created.
- **pcs resource create**, create the resource.
- **corosync-cfgtool -s**, to check whether cluster communication is happy.
- **ps axf**, confirmed that Corosync is functional, we can check the rest of the stack. Pacemaker has already been started, so verify the necessary processes are running.
- **pcs status**, check the pcs status output.
- **crm_verify -L -V**, check the validity of the configuration.
- **drbdadm status**, shows the integrity status of the disks that are being shared between both servers in high availability. If for some reason the status of Connecting or Standalone returns to us, wait a while and if the state remains it is because there are synchronization problems between both servers, and you should execute the drbdsplit command.
- **cat /proc/drbd**, the state of your device is kept in /proc/drbd.
- **drbdadm connect drbd0,** on the other node (the split brain survivor), if its connection state is also StandAlone, you would enter.
- **drbdadm role drbd0**, another way to check the role of the block device.
- **drbdadm primary drbd0**, switch the DRBD block device to Primary using drbdadm.
- **drbdadm secondary drbd0**, switch the DRBD block device to Secondary using drbdadm.
- **/usr/share/vitalpbx/ha/ ./vpbxha.sh,** create the cluster automatically.
- **/usr/share/vitalpbx/ha/ ./destroy.sh,** completely destroy the cluster, leaving the DRBD intact.
- **/usr/share/vitalpbx/ha/ ./rebuild.sh,** recreates the cluster starting from the fact that the DRBD is already configured on both servers.

# 10.- Solve a DRBD split-brain in 4 steps

Whenever a DRBD setup runs into a situation where the replication network is disconnected and fencing policy is set to dont-care (default), there is the potential risk of a split-brain. Even with resource level fencing or STONITH setup, there are corner cases that will end up in a split-brain.

When your DRBD resource is in a split-brain situation, don't panic! Split-brain means that the contents of the backing devices of your DRBD resource on both sides of your cluster started to diverge. At some point in time, the DRBD resource on both nodes went into the Primary role while the cluster nodes themselves were disconnected from each other.
Different writes happened to both sides of your cluster afterwards. After reconnecting, DRBD doesn't know which set of data is "right" and which is "wrong".

## *Indications of a Split-Brain*

The symptoms of a split-brain are that the peers will not reconnect on DRBD startup but stay in connection state StandAlone or WFConnection. The latter will be shown if the remote peer detected the split-brain earlier and was faster at shutdown its connection. In your kernel logs you will see messages like:

```
kernel: block drbd0: Split-Brain detected, dropping connection!
```

## *4 Steps to solve the Split-Brain*

### Step 1

Manually choose a node which data modifications will be discarded.
We call it the split brain victim. Choose wisely, all modifications will be lost! When in doubt run a backup of the victim's data before you continue.
When running a Pacemaker cluster, you can enable maintenance mode. If the split brain victim is in Primary role, bring down all applications using this resource. Now switch the victim to Secondary role:

```
root@vitalpbx-victim:~# drbdadm up drbd0
root@vitalpbx-victim:~# drbdadm secondary drbd0
```

### Step 2

Disconnect the resource if it's in connection state WFConnection:

```
root@vitalpbx-victim:~# drbdadm disconnect drbd0
```

### Step 3

Force discard of all modifications on the split brain victim:

```
root@vitalpbx-victim:~# drbdadm connect --discard-my-data drbd0
```

### Step 4

Resync will start automatically if the survivor was in WFConnection network state. If the split brain survivor is still in Standalone connection state, reconnect it:

```
root@vitalpbx-survivor:~# drbdadm connect drbd0
```

At the latest now the resynchronization from the survivor (SyncSource) to the victim (SyncTarget) starts immediately. There is no full sync initiated but all modifications on the victim will be overwritten by the survivor's data and modifications on the survivor will be applied to the victim.

### *Background: What happens?*

With the default after-split-brain policies of disconnect this will happen always in dual primary setups. It can happen in single primary setups if one peer changes at least once its role from Secondary to Primary while disconnected from the previous (before network interruption) Primary.

There are a variety of automatic policies to solve a split brain but some of them will overwrite (potentially valid) data without further inquiry. Even with theses policies in place a unresolvable split-brain can occur.

The split-brain is detected once the peers reconnect and do their DRBD protocol handshake which also includes exchanging of the Generation Identifiers (GIs).

# 11.- Credits

## 11.1 Sources of Information

- voip-info.org
- asterisk.org
- DRBD Website (https://www.linbit.com/en/)
- Pacemaker Website (https://clusterlabs.org/pacemaker/)
- https://github.com/VitalPBX/vitalpbx4_drbd_ha
- https://xahteiwi.eu/resources/hints-and-kinks/solve-drbd-split-brain-4-steps/