



High Availability V4 – Debian 11

CONTENTS

Contents

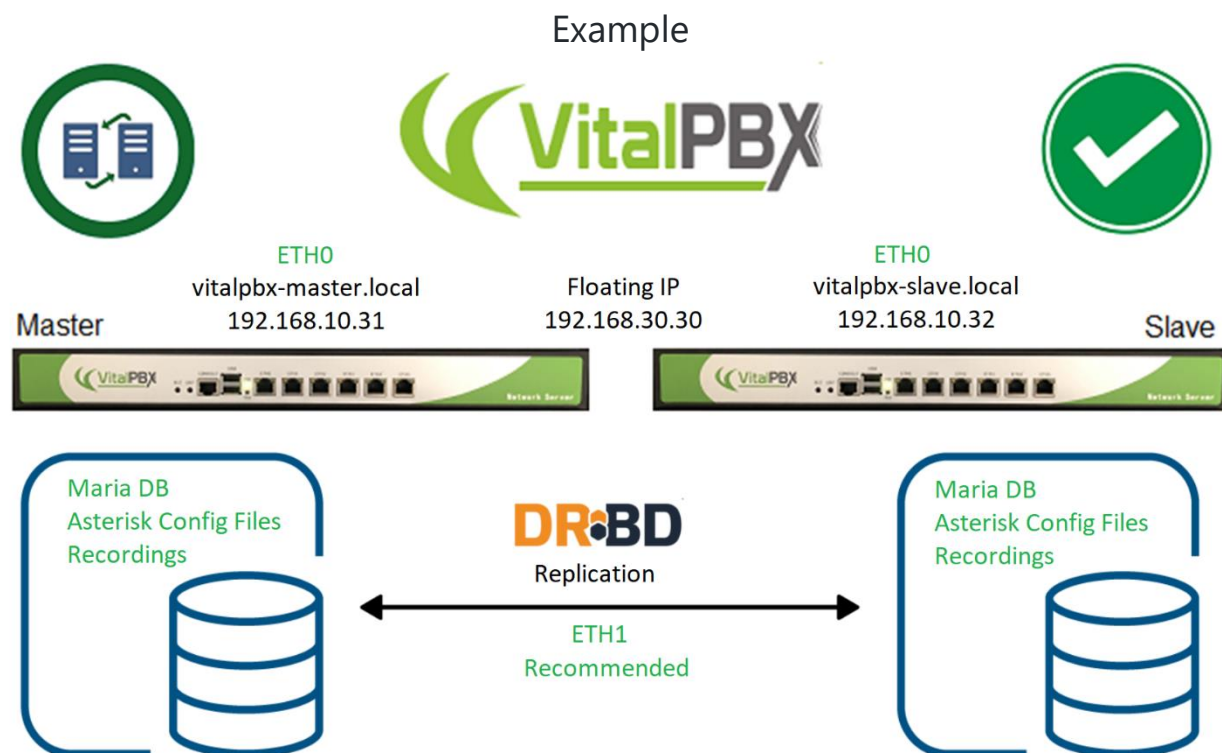
VITALPBX HIGH AVAILABILITY	3
1.- INTRODUCTION	3
2.- PREREQUISITES	3
3.- INSTALLATION	4
3.- CONFIGURATIONS.....	10
3.1- IP AND HOSTNAME CONFIGURATION.	10
3.2.- INSTALL DEPENDENCIES	11
3.3.- HOSTNAME	11
3.4.- FIREWALL.....	12
3.5.- CREATE THE PARTITION ON BOTH SERVERS	14
3.6.- FORMAT THE PARTITION	14
3.7.- CONFIGURING DRBD.....	14
3.8.- CONFIGURE CLUSTER	16
3.9.- CREATE "BASCUL" COMMAND IN BOTH SERVERS	20
3.10.- CREATE "ROLE" COMMAND IN BOTH SERVERS	22
3.11.- CREATE "DRBDSPLIT" COMMAND IN BOTH SERVERS	23
4.- ADD NEW SERVICES.....	24
4.1.- ADD SONATA SWITCHBOARD	24
4.2.- ADD SONATA STATS.....	24
4.3.- ADD SONATA RECORDING	24
4.4.- ADD SONATA BILLING	25
4.5.- ADD SONATA DIALER	25
4.6.- ADD VITXI	25
4.7.- ADD OPENVPN	26
5.- TEST	27
6.- TURN OFF AND TURN ON	27
7.- UPDATE VITALPBX OR ADD-ONS	27
8.- SOME USEFUL COMMANDS.....	28
9.- CREDITS.....	28
9.1 SOURCES OF INFORMATION	28

VitalPBX High Availability

1.- Introduction

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.



2.- Prerequisites

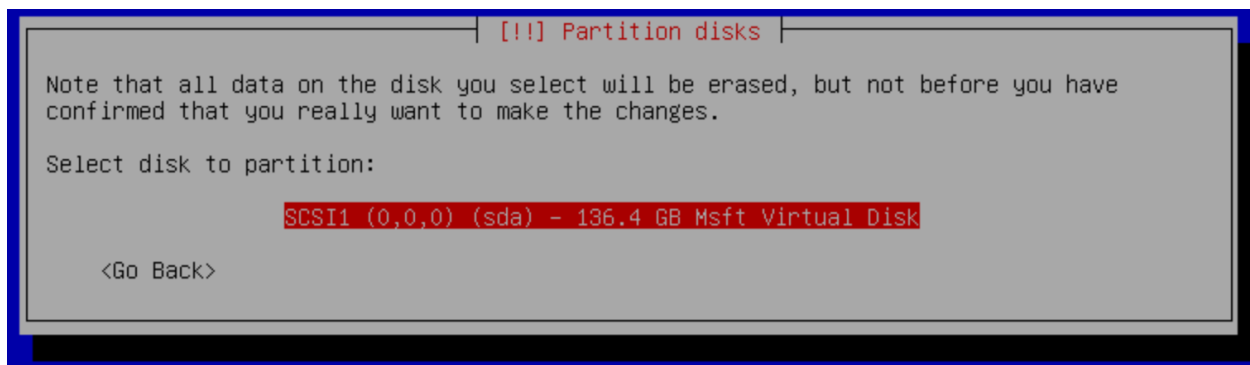
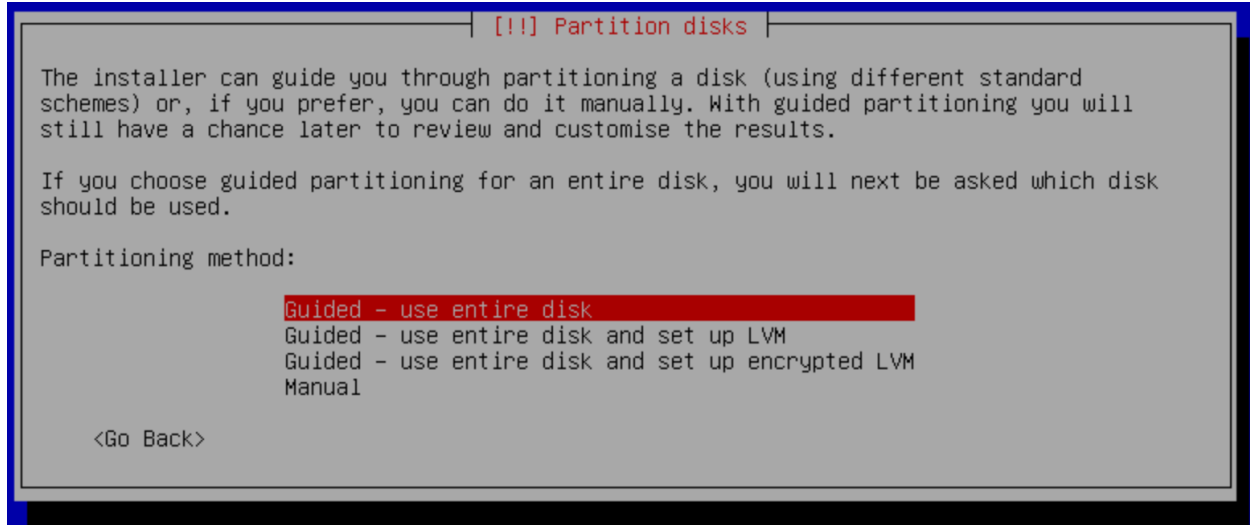
In order to install VitalPBX in high availability you need the following:

- a.- 3 IP addresses for access and 2 IP addresses for synchronization (DRBD).
- b.- Install VitalPBX on two servers with similar characteristics.
- c.- At the time of installation leave the largest amount of space on the hard drive to store the variable data on both servers.

3.- Installation

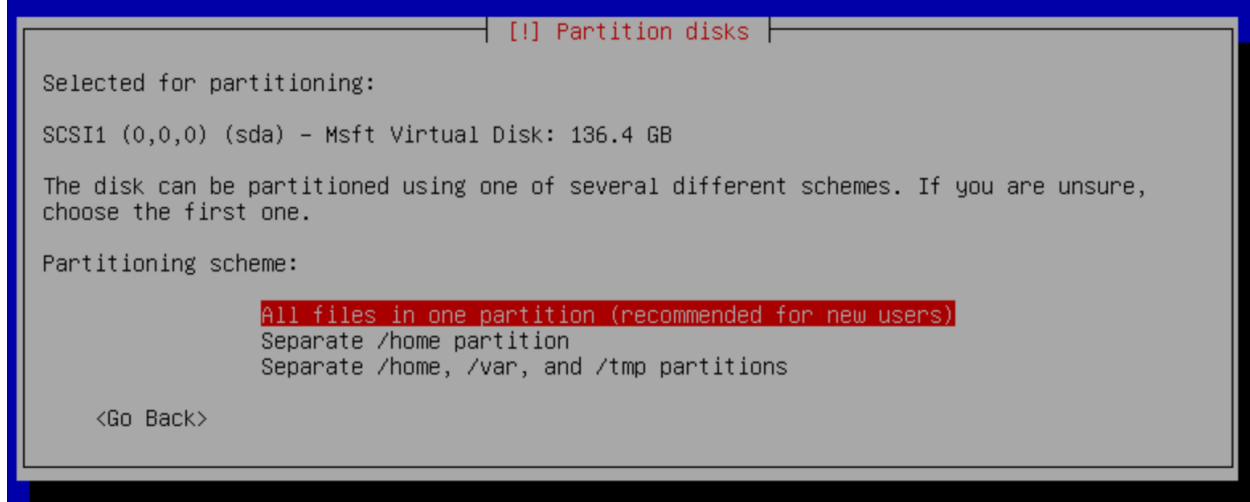
We are going to start by installing VitalPBX on two servers

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



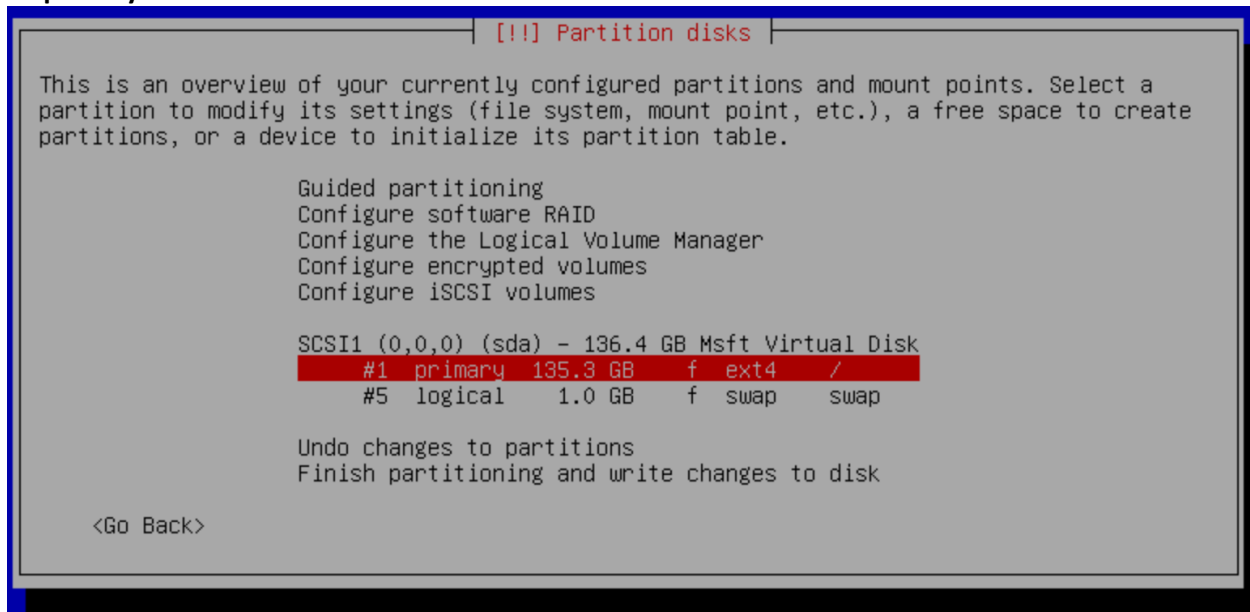
b.- Select:

All file in one partition (recommended for new user)

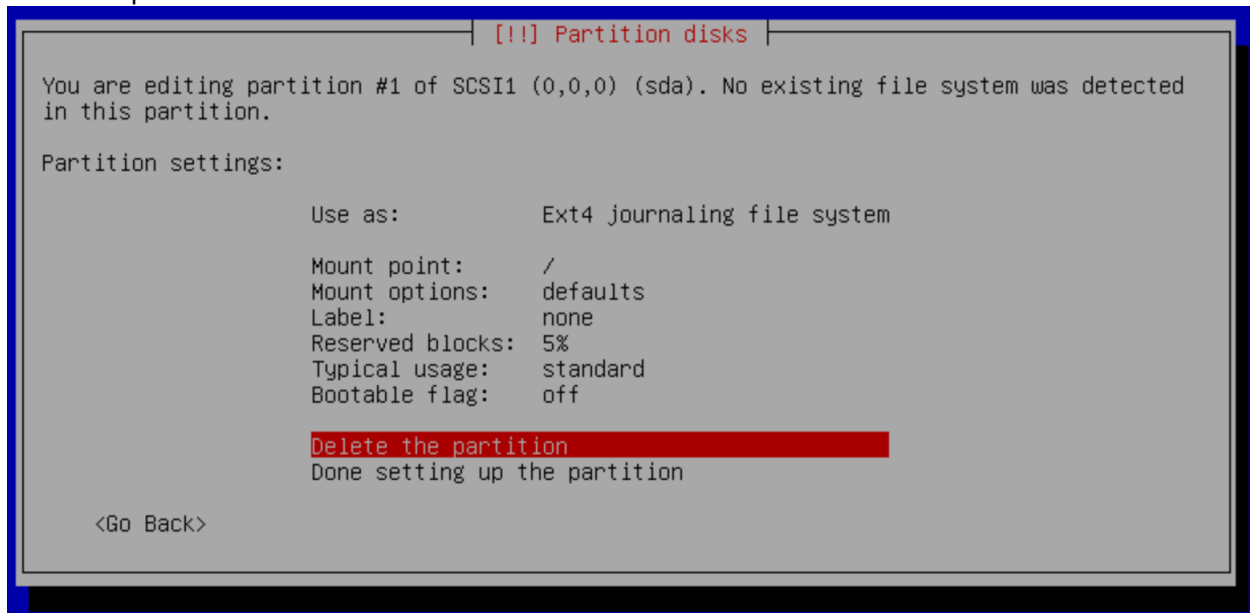


c.- Select primary:

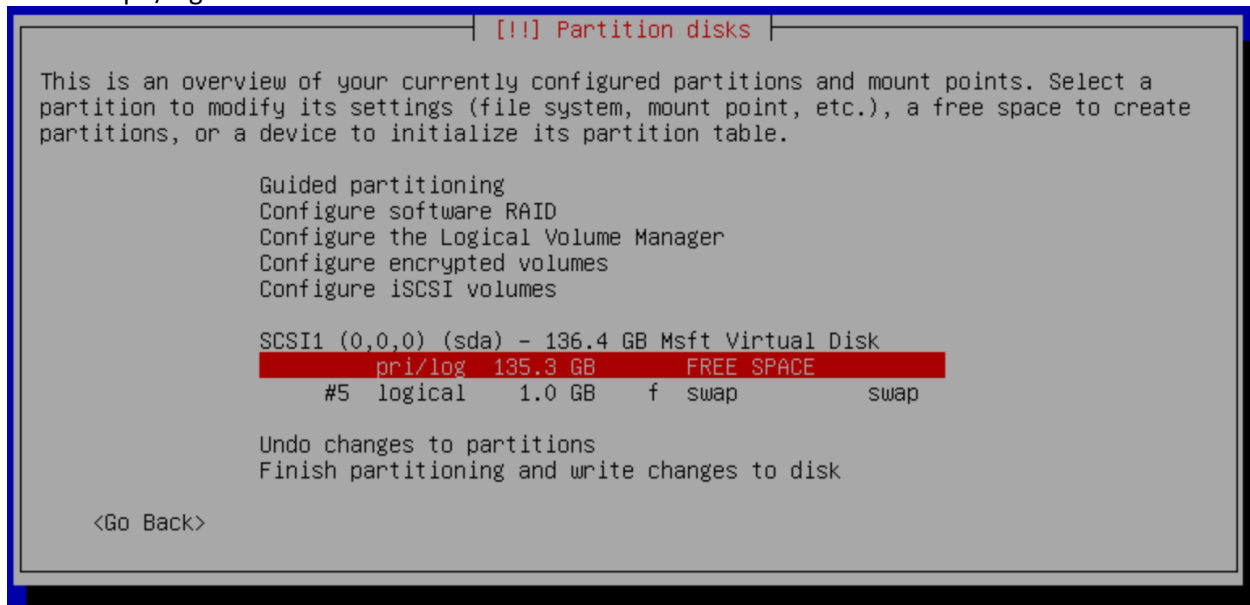
#1 primary



d.- Delete partition



e.- Select pri/log



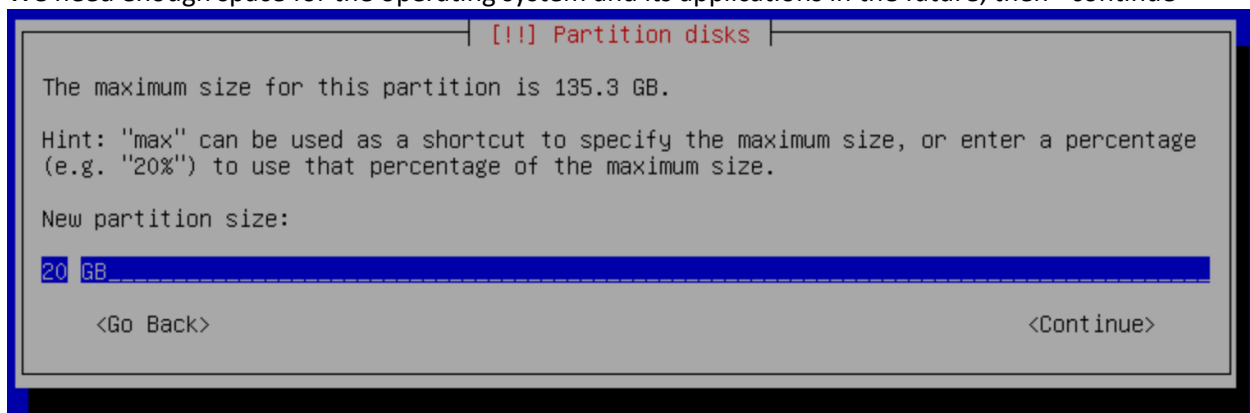
f.- Create New Partition



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

!!! Partition disks

Please choose whether you want the new partition to be created at the beginning or at the end of the available space.

Location for the new partition:

Beginning

End

<Go Back>

Select Done setting up the partition

!!! Partition disks

You are editing partition #1 of SCSI1 (0,0,0) (sda). No existing file system was detected in this partition.

Partition settings:

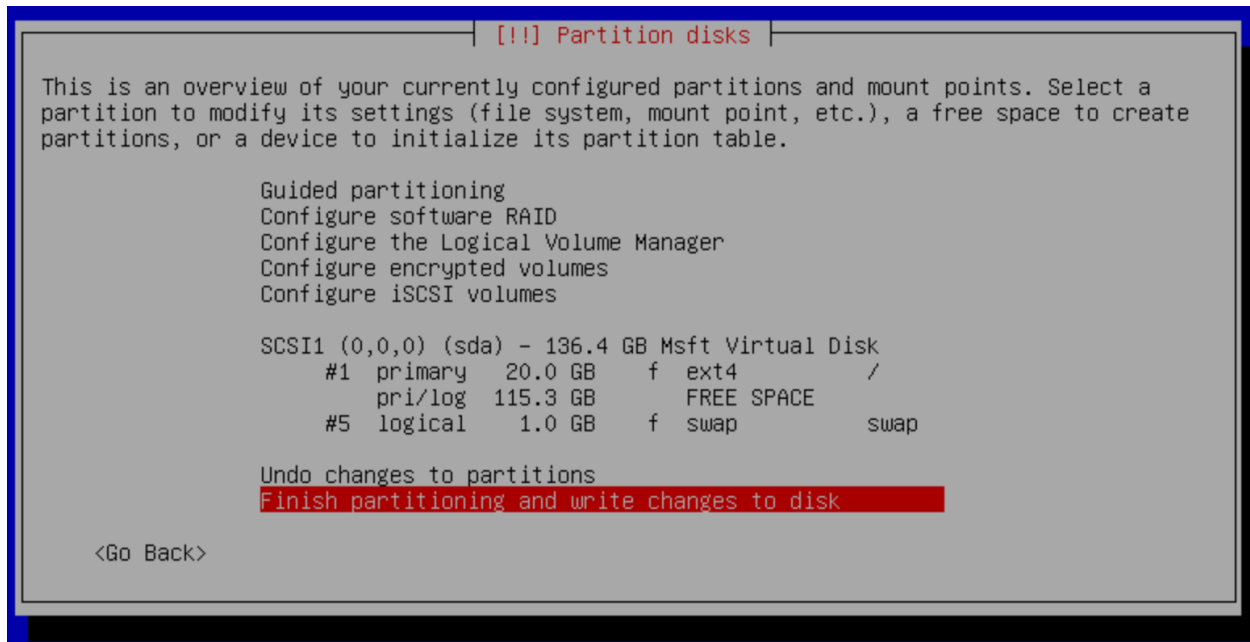
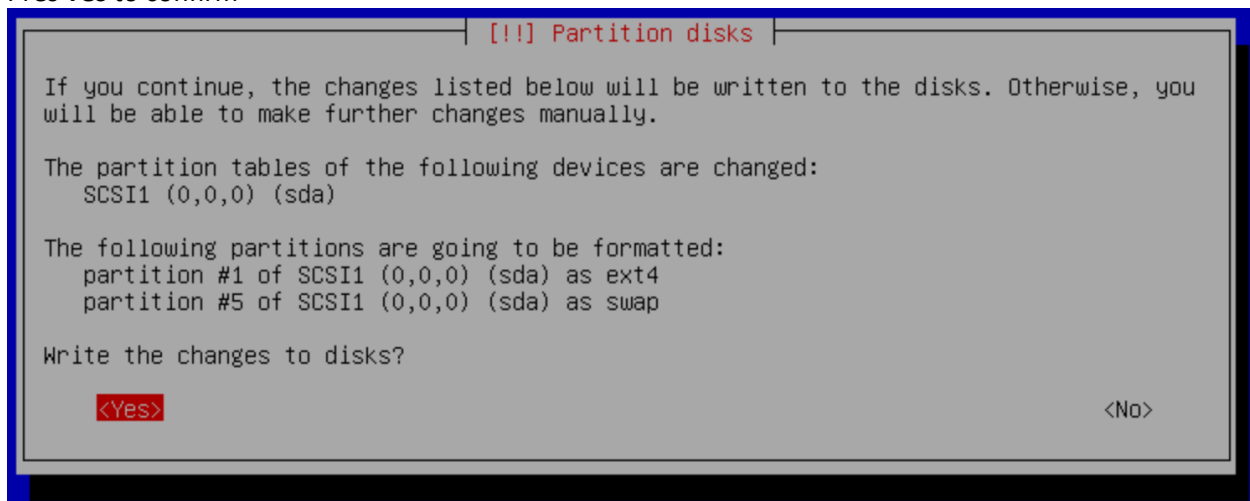
Use as:	Ext4 journaling file system
Mount point:	/
Mount options:	defaults
Label:	none
Reserved blocks:	5%
Typical usage:	standard
Bootable flag:	off

Delete the partition

Done setting up the partition

<Go Back>

h.- Finish

Pres **Yes** to confirm

And continue with the installation.

3.- Configurations

3.1- 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

3.1.1.- 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
```

3.1.2.- 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 dhcp
```

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.

3.2.- Install Dependencies

Install dependencies on both servers

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

3.3.- 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

The screenshot shows the 'Network Settings' web interface for the Master server. The 'GENERAL' tab is selected. The 'Hostname' field contains 'vitalpbx-master.local'. A green 'Save' button is visible to the right of the field.

Slave

The screenshot shows the 'Network Settings' web interface for the Slave server. The 'GENERAL' tab is selected. The 'Hostname' field contains 'vitalpbx-slave.local'. A green 'Save' button is visible to the right of the field.

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
```

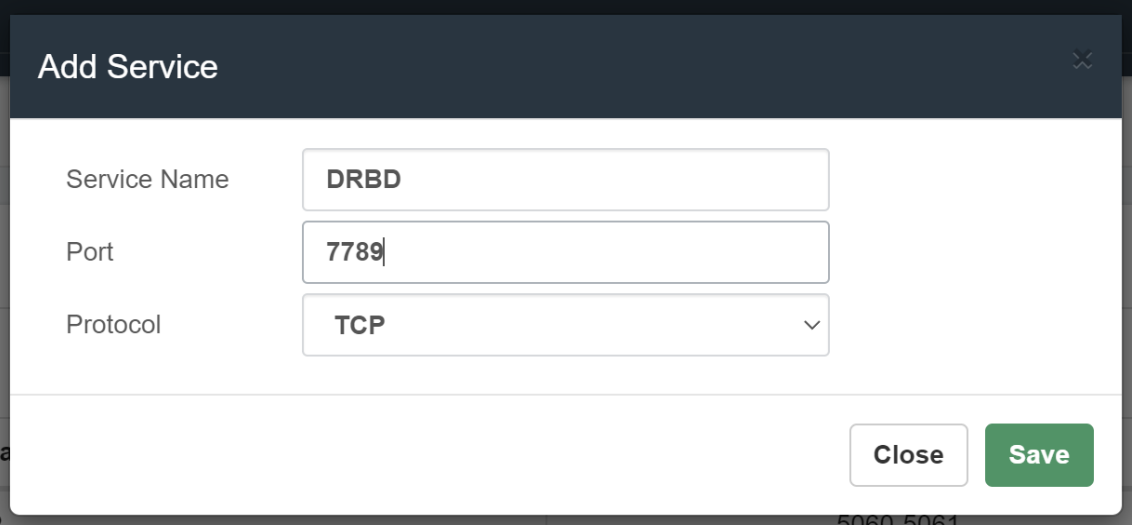
3.4.- Firewall

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) 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 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



Add Service

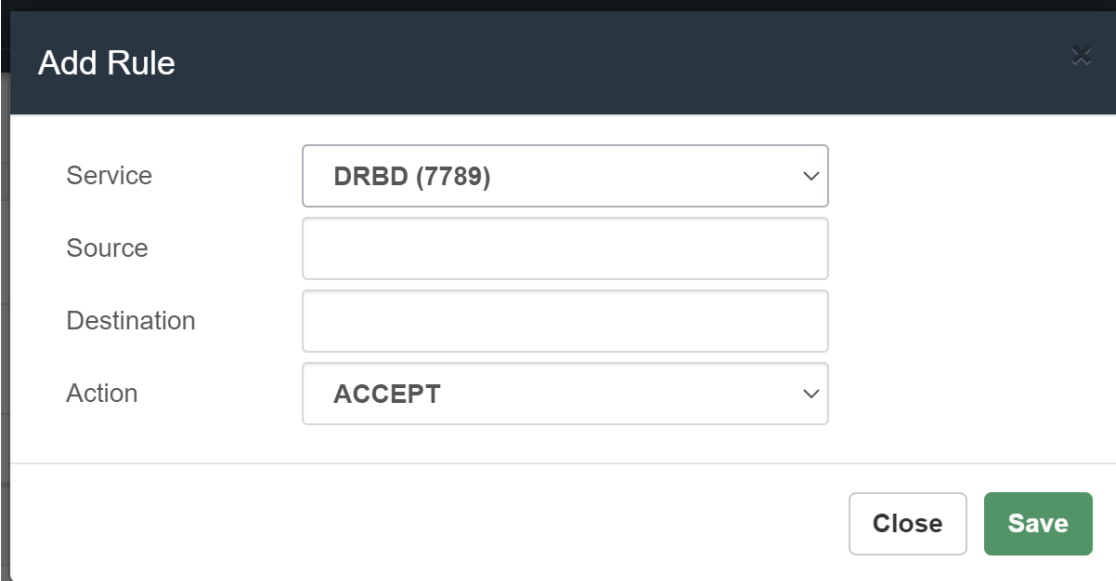
Service Name **DRBD**

Port **7789**

Protocol **TCP**

Close **Save**

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



Add Rule

Service **DRBD (7789)**

Source

Destination

Action **ACCEPT**

Close **Save**

And we apply changes

3.5.- 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
```

3.6.- Format the partition

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

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

3.7.- 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 yes;
}
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 {
    on vitalpbx-master.local {
        device /dev/drbd0;
        disk /dev/sda3;
        address 192.168.10.31:7789;
        meta-disk internal;
    }
    on vitalpbx-slave.local {
        device /dev/drbd0;
        disk /dev/sda3;
        address 192.168.10.32:7789;
```

```

        meta-disk internal;
    }
    handlers {
        split-brain "/usr/lib/drbd/notify-split-brain.sh root";
    }
    net {
        after-sb-0pri discard-zero-changes;
        after-sb-1pri discard-secondary;
        after-sb-2pri disconnect;
    }
}

```

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, as shown here:

3.7.1.- 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 /mnt

```

Create some data using the following command:

```

root@vitalpbx-master:~# touch /mnt/file{1..5}
root@vitalpbx-master:~# ls -l /mnt
-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 /mnt
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 /mnt
root@vitalpbx-slave:~# ls -l /mnt
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 /mnt
root@vitalpbx-slave:~# drbdadm secondary drbd0
```

Normalize Server-Master

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

3.8.- Configure Cluster

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 4 if you consider it necessary. Otherwise continue with step 3.8.2.

3.8.1.- Using Script

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
hacluster password..... > MyPassword
*****
* Check Information *
* Make sure you have internet on both servers *
*****
Are you sure to continue with this settings? (yes,no) > yes
```

CONGRATULATIONS, you have installed high availability in VitalPBX 4

3.8.2.- Creating the Cluster step by step.

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

```
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
```

Create the cluster and configure parameters, perform only on the server-master

```
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
```

Create resource for the use of Floating IP

```
root@vitalpbx-master:~# pcs resource create virtual_ip 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
```

Create resource for the use of DRBD

```
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-push drbd_cfg
```

Create FILESYSTEM resource for the automated mount point

```
root@vitalpbx-master:~# pcs cluster cib fs_cfg
root@vitalpbx-master:~# pcs -f fs_cfg resource create DrbdFS Filesystem device="/dev/drbd0"
directory="/mnt" 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 virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order DrbdData-clone then DrbdFS
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg
```

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 /mnt/mysql
root@vitalpbx-master:~# mkdir /mnt/mysql/data
root@vitalpbx-master:~# cp -aR /var/lib/mysql/* /mnt/mysql/data
root@vitalpbx-master:~# chown -R mysql:mysql /mnt/mysql
root@vitalpbx-master-slave:~# sed -i 's/var\lib\mysql/mnt\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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add mysql with virtual_ip INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order DrbdFS then mysql
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg
```

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 virtual_ip
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 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 /mnt/
root@vitalpbx-master:~# tar xvfz var-lib-asterisk.tgz -C /mnt/
root@vitalpbx-master:~# tar xvfz var-lib-vitalpbx.tgz -C /mnt/
root@vitalpbx-master:~# tar xvfz usr-lib-asterisk.tgz -C /mnt/
root@vitalpbx-master:~# tar xvfz var-spool-asterisk.tgz -C /mnt/
root@vitalpbx-master:~# tar xvfz etc-asterisk.tgz -C /mnt/

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

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

root@vitalpbx-slave:~# ln -s /mnt/var/log/asterisk /var/log/asterisk
root@vitalpbx-slave:~# ln -s /mnt/var/lib/asterisk /var/lib/asterisk
root@vitalpbx-slave:~# ln -s /mnt/var/lib/vitalpbx /var/lib/vitalpbx
root@vitalpbx-slave:~# ln -s /mnt/usr/lib/asterisk /usr/lib/asterisk
root@vitalpbx-slave:~# ln -s /mnt/var/spool/asterisk /var/spool/asterisk
root@vitalpbx-slave:~# ln -s /mnt/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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add vpbx-monitor with virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order asterisk then vpbx-monitor
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg

```

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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add fail2ban with virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order asterisk then fail2ban
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg
```

Initialize the services of corosync and pacemaker in server-slave

```
root@vitalpbx-slave:~# systemctl restart corosync.service
root@vitalpbx-slave:~# systemctl restart pacemaker.service
```

Show the Cluster Status

```
root@vitalpbx-master:~# pcs status resources
* virtual_ip (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
```

3.9.- Create “bascul” command in both servers

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
#!/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 Standby and the Standby goes to Master.
#function for draw a progress bar
#You must pass as argument the amount of seconds 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-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 $2}' | while read -r resource ; do
            #Skip moving the virtual_ip resource, it will be moved at the end
            if [[ "${resource}" != "virtual_ip" ]] && [[ "${resource}" != "Clone" ]] && [[
"${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 Virtual IP resource to standby node
        echo "Moving virtual_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 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

```

22

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
```

3.11.- Create “drbdsplit” command in both servers

Split brain can be caused by intervention by cluster management software or human error during a period of failure for network links between cluster nodes, causing both nodes to switch to the primary role while disconnected.

Split brain occurs when both High Availability nodes switch into the primary role while disconnected. This behavior can allow data to be modified on either node without being replicated on the peer, leading to two diverging sets of data on each node, which can be difficult to merge.

Download file

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

Or create file

```
root@vitalpbx-master-slave:~# nano drbdsplit
#!/bin/bash
set -e
# This code is the property of VitalPBX LLC Company
# License: Proprietary
# Date: 1-Agu-2023
# DRBD split-brain recovery
#
drbdadm secondary drbd0
drbdadm disconnect drbd0
drbdadm -- --discard-my-data connect drbd0
echo "Disk Status"
drbdadm status
```

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

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

4.- Add New Services

4.1.- Add Sonata Switchboard

If we decide to install Sonata Switchboard, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install Sonata Switchboard
- 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 Sonata Switchboard
- 6.- Execute the following command in Master console

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

4.2.- Add Sonata Stats

If we decide to install Sonata Stats, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install Sonata Stats
- 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 Sonata Stats
- 6.- Execute the following command in Master console

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

- 7.- We added the Sonata Stats automation service.

```
root@vitalpbx-master:~# systemctl stop sonata-stats
root@vitalpbx-master:~# 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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add sonata-stats with virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then sonata-stats
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg
```

4.3.- Add Sonata Recording

If we decide to install Sonata Recording, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install Sonata Recording
- 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 Sonata Recording
- 6.- Execute the following command in Master console

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

- 7.- We added the Sonata Recordings automation service.

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

```

4.4.- Add Sonata Billing

If we decide to install Sonata Billing, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install Sonata Billing
- 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 Sonata Billing
- 6.- Execute the following command in Master console

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

4.5.- Add Sonata Dialer

If we decide to install Sonata Dialer, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install Sonata Dialer
- 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 Sonata Dialer
- 6.- Execute the following command in Master console

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

- 7.- We added the Sonata Dialer automation service.

```

root@vitalpbx-master:~# systemctl stop sonata-dialer
root@vitalpbx-master:~# 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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add sonata-dialer with virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then sonata-dialer
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg

```

4.6.- Add VitXi

If we decide to install VitXi, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install VitXi
- 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 VitXi
- 6.- Execute the following command in Master console

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

7.- We added the VitXi automation service.

```

root@vitalpbx-master:~# tar -zcvf vitxi-storage.tgz /usr/share/vitxi/backend/storage
root@vitalpbx-master:~# tar xvfz vitxi-storage.tgz -C /mnt/
root@vitalpbx-master:~# rm -rf /usr/share/vitxi/backend/storage
root@vitalpbx-master:~# ln -s /mnt/usr/share/vitxi/backend/storage
/usr/share/vitxi/backend/storage
root@vitalpbx-master:~# rm -rf vitxi-storage.tgz

root@vitalpbx-slave:~# rm -rf /usr/share/vitxi/backend/storage
root@vitalpbx-slave:~# ln -s /mnt/usr/share/vitxi/backend/storage
/usr/share/vitxi/backend/storage

root@vitalpbx-master:~# systemctl stop vitxi
root@vitalpbx-master:~# 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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add vitxi with virtual_ip INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then vitxi
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg

```

4.7.- Add OpenVPN

If we decide to install OpenVPN, then we show the procedure.

- 1.- From your browser, go to ip 192.168.10.30
- 2.- Install OpenVPN
- 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 OpenVPN
- 6.- Execute the following command in Master console

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

7.- We added the OpenVPN automation service.

```

root@vitalpbx-master:~# systemctl stop openvpn
root@vitalpbx-master:~# 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
root@vitalpbx-master:~# pcs -f fs_cfg constraint colocation add openvpn with virtual_ip
INFINITY
root@vitalpbx-master:~# pcs -f fs_cfg constraint order vpbx-monitor then openvpn
root@vitalpbx-master:~# pcs cluster cib-push fs_cfg

```

5.- 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.

6.- Turn off and turn on

Warning Note ⚠

When you must turn off the servers, when you turn it on always 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.- 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.
- **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 cluster 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.
- **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.
- **drbdsplit**, solves DRBD split brain recovery.
- **cat /proc/drbd**, the state of your device is kept in /proc/drbd.
- **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.

9.- Credits

9.1 Sources of Information

- voip-info.org
- asterisk.org
- DRBD Website (<https://www.linbit.com/en/>)
- Pacemaker Website (<https://clusterlabs.org/pacemaker/>)