# Practice M6: Clustering and High Availability (CentOS)

This practice assumes that you are working in an on-premise environment

All tasks can be achieved under different configurations (host OS and/or virtualization solution) with the appropriate adjustments

This practice is oriented towards **CentOS 8.4** and **Fedora Server 34**

## Part 1: Load Balancing

For this part we will need an infrastructure like this one:

Chart

Description automatically generated

Machines can be with or without graphical environment

Network settings shown on the picture reflect the ones, used during the demonstration. You should adjust them according to your setup

Please note that **M1** should be set as a default gateway for **M2** and **M3**

### Linux Virtual Server

#### Installation

Let’s log on to the first machine (**M1**) and install the prerequisites

**sudo dnf install ipvsadm**

#### Preparation

We can use at least two methods here, but let’s go with the first one

##### IP Forwarding (Method 1)

Enable IP forwarding

**echo 'net.ipv4.ip\_forward = 1' | sudo tee -a /etc/sysctl.conf**

Read the values from configuration files

**sudo sysctl -p**

Check if the value has been set

**sysctl net.ipv4.ip\_forward**

##### IP Forwarding (Method 2)

Put both network interfaces in the appropriate firewall zones (make sure to adjust the adapter/connection names)

**sudo nmcli conn modify eth0 connection.zone external**

**sudo nmcli conn modify eth1 connection.zone internal**

#### Configuration

Check the content of the **IPVS** configuration file

**sudo cat /etc/sysconfig/ipvsadm-config**

It appears that we must ensure that **/etc/sysconfig/ipvsadm** file exists, so let’s create it

**sudo touch /etc/sysconfig/ipvsadm**

Now, enable and start the service

**sudo systemctl enable --now ipvsadm**

And check if everything is okay

**systemctl status ipvsadm**

Let’s check if there are any existing rules

**sudo ipvsadm -l**

We can achieve the same with

**cat /proc/net/ip\_vs**

Let’s clean the rules even if there aren’t any

**sudo ipvsadm -C**

Now, let’s add a virtual service that will listen on port **80/tcp** and will use the **round-robin** distribution method

**sudo ipvsadm -A -t <ext-ip-of-machine-1>:80 -s rr**

And now, we must add the two backend (or real) servers with

**sudo ipvsadm -a -t <ext-ip-of-machine-1>:80 -r <ip-of-machine-2>:80 -m**

**sudo ipvsadm -a -t <ext-ip-of-machine-1>:80 -r <ip-of-machine-3>:80 -m**

Let’s check what are the results so far

**sudo ipvsadm -l**

We should see the rules

It would be nice to open the necessary firewall ports on **M1**

*If during the preparation, you did not used the zone placement as an option to enable IP forwarding, then instead of specifying the zones now, skip them to allow the port in the default zone (execute the command once)*

**sudo firewall-cmd --add-service http --permanent --zone external**

**sudo firewall-cmd --add-service http --permanent --zone internal**

**sudo firewall-cmd --reload**

#### Testing

In order to prove that our setup is working correctly, we must install a web server on both **M2** and **M3**

Install **Apache** web server

**sudo dnf install httpd**

Enable and start it

**sudo systemctl enable --now httpd**

Set a custom index page

**echo 'Hello from <machine-name>' | sudo tee /var/www/html/index.html**

It would be nice to open the necessary firewall ports as well

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

Make sure that the above steps were executed successfully on both stations

Open a browser tab on the host (or on the **M1** machine) and navigate to **<ext-ip-of-machine-1>**

Now refresh a few times to see that the page is served by different nodes

Should you have issues accessing the resource, try to disable the firewall on **M1**

#### Persistent Configuration

In order to retain the configuration after reboot, we must execute

**sudo ipvsadm-save -n | sudo tee /etc/sysconfig/ipvsadm**

Or we can change the appropriate settings in **/etc/sysconfig/ipvsadm-config**

Alternatively, we can use the following pair of commands to save and load the rules

**sudo ipvsadm -Sn > rules.txt**

**sudo ipvsadm -R < rules.txt**

### Linux Virtual Server + Keepalived

For this part, we will need an infrastructure like this

Chart, diagram, waterfall chart

Description automatically generated

#### Installation (Load Balancers)

Log on to **M1**

Install the required packages

**sudo dnf install keepalived ipvsadm**

Repeat the installation on **M2** as well

#### Installation (Web Servers)

Let’s prepare the other two machines as well

Log on to **M3** and execute

**sudo dnf install httpd**

Enable and start it

**sudo systemctl enable --now httpd**

Repeat this on **M4** as well

#### Configuration (Web Servers)

Set a custom index page

**echo 'Hello from WEB1' | sudo tee /var/www/html/index.html**

It would be nice to open the necessary firewall ports as well

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --permanent --direct --add-rule ipv4 nat PREROUTING 0 -d <ip-address-m3> -j REDIRECT**

**sudo firewall-cmd --permanent --direct --add-rule ipv4 nat PREROUTING 0 -d <vip-address> -j REDIRECT**

**sudo firewall-cmd --reload**

Repeat the above on **M4** as well

#### Configuration (Load Balancers)

Log on to **M1**

Create a copy of the initial configuration file if you like

**sudo cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.bak**

Open the original file for editing

**sudo vi /etc/keepalived/keepalived.conf**

And change it to match the configuration provided as **LSAA-M6-P1-KA1.txt**

Don’t forget to adjust the IP addresses, names, and other parameters you see fit

Now, enable and start the **keepalived** service

**sudo systemctl enable --now keepalived**

Check if the virtual address is there (it may take some time, retry after a while)

**ip a**

Adjust the following settings in **/etc/sysctl.conf**

**echo 'net.ipv4.ip\_forward = 1' | sudo tee -a /etc/sysctl.conf**

**echo 'net.ipv4.conf.all.rp\_filter = 0' | sudo tee -a /etc/sysctl.conf**

Apply the changes with

**sudo sysctl -p**

Create an empty file **/etc/sysconfig/ipvsadm**

**sudo touch /etc/sysconfig/ipvsadm**

Enable and start the **ipvsadm** service

**sudo systemctl enable --now ipvsadm**

Check if the rules appeared with

**sudo ipvsadm -Ln**

*If no rules appear, restart the* ***keepalived*** *service and check again*

Don’t forget to adjust the firewall as well

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

Repeat the procedure on **M2** but change the **/etc/keepalived/keepalived.conf** to match the **LSAA-M6-P1-KA2.txt**

Adjust the configuration if needed

#### Testing (Host)

Open a browser tab and navigate to **<virtual-ip>**

Now refresh a few times to see that the page is served by different nodes

Should you have issues accessing the resource, try to disable the firewall on **M1** (and then on **M2**)

Hm, it seems that one and the same host replies

Open the **/etc/keepalived/keepalived.conf** file and comment out the **persistence\_timeout 50** setting (line 34)

Save and close

Change the configuration on both servers (**M1** and **M2**)

Restart the service (**keepalived**) on both servers

Check the generated rules

Refresh the browser tab several times

Now, the result is different

Now, you can stop (and then start) one of the load balancer nodes and try to refresh again

Then you can stop (and then start) one of the web servers and check again

### Load Balancing (HAProxy)

For this part, we will need an infrastructure like this

Waterfall chart

Description automatically generated

#### Preparation (Web Servers)

Let’s first prepare the two web servers

Log on to **M2** and install the software

**sudo dnf install httpd**

Set a custom index page

**echo 'Hello from WEB1' | sudo tee /var/www/html/index.html**

It would be nice to open the necessary firewall ports as well

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

Open the main configuration file of **Apache**

**sudo vi /etc/httpd/conf/httpd.conf**

Change (substitute **%h** with **\"%{X-Forwarded-For}i\"**) the log format setting (row 199) to

**LogFormat "\"%{X-Forwarded-For}i\" %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined**

Save and close the file

Enable and start it

**sudo systemctl enable --now httpd**

Repeat this on **M3** as well

#### Preparation (Load Balancer)

Log on to **M1** and install the software

**sudo dnf install haproxy**

Let’s adjust the configuration file **/etc/haproxy/haproxy.cfg** to match our needs

**sudo vi /etc/haproxy/haproxy.cfg**

Comment out (or delete) the existing **frontend** and **backend** sections

Create a new **frontend** section like this

**frontend http-in**

**bind \*:80**

**default\_backend web\_servers**

**option forwardfor**

Create a new **backend** section like this

**backend web\_servers**

**balance roundrobin**

**server m2 <ip-address-m2>:80 check**

**server m3 <ip-address-m3>:80 check**

Be sure to adjust the parameters according to your setup

Save and close the file

We can configure the **rsyslog** daemon to capture the logs from **haproxy**

Open the **/etc/rsyslog.conf** file for editing

**sudo vi /etc/rsyslog.conf**

Uncomment the lines (19 and 20) about **imudp**

Just after them add this one

**$AllowedSender UDP, 127.0.0.1**

In the list on the line (47) about **/var/log/messages** add

**;local2.none**

As next line add

**local2.\* /var/log/haproxy.log**

Save and close the file

Restart the **rsyslog** service

**sudo systemctl restart rsyslog**

Start and enable the **haproxy**

**sudo systemctl enable --now haproxy**

Open the **HTTP** service in the firewall

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

#### Testing

Open a browser tab and navigate to **<load-balancer-ip>**

Refresh a few times

## Part 2: Failover Clusters

For this part we will need an infrastructure like this one:

Chart

Description automatically generated with medium confidence

Machines can be with or without graphical environment

Network settings shown on the picture reflect the ones, used during the demonstration. You should adjust them according to your setup

### Preparation

Let’s first log on **M1** and install **Apache**

**sudo dnf install httpd**

Create a new file **/etc/httpd/conf.d/server-status.conf**

**sudo vi /etc/httpd/conf.d/server-status.conf**

With the following content

**<Location /server-status>**

**SetHandler server-status**

**Require local**

**</Location>**

Create a sample **index.html** page

**echo 'Demo page running on WEB1' | sudo tee /var/www/html/index.html**

Adjust the firewall rules

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

Repeat the steps on **M2** as well

### Installation

Log on to **M1** and install the required packages

First, we must enable the high availabilityrepository

**sudo dnf config-manager --set-enabled ha**

Then, we can start with the installation of the main packages

**sudo dnf install pacemaker pcs**

Once installed, enable and start the **pcsd** service

**sudo systemctl enable --now pcsd**

Check if the **hacluster** user has been created

**grep hacluster /etc/passwd**

Set its password, for example to **Password1**

**sudo passwd hacluster**

Adjust the firewall rules

**sudo firewall-cmd --add-service=high-availability --permanent**

**sudo firewall-cmd --reload**

Repeat the steps so far on the second (**M2**) node as well

### Configuration

Return on the first node (**M1**)

Authorize among the nodes (use the **hacluster** user and its password)

**sudo pcs host auth m1.lsaa.lab m2.lsaa.lab**

Generate the cluster configuration

**sudo pcs cluster setup demo m1.lsaa.lab m2.lsaa.lab**

Start the cluster

**sudo pcs cluster start --all**

Enable cluster auto-start

**sudo pcs cluster enable --all**

Check the status of the cluster

**sudo pcs cluster status**

List all cluster properties

**sudo pcs property list --all**

Let’s set a property

**sudo pcs property set stonith-enabled=false**

Check membership status

**sudo pcs status corosync**

We can view the cluster configuration in raw format

**sudo pcs cluster cib**

We can check the configuration consistency of a running cluster with

**sudo crm\_verify -LV**

Multiple **V** letters can be added to increase the verbosity level

### Resources

Before we create any resources, let’s explore what types are there

Check resource standards

**sudo pcs resource standards**

Alternatively, for older installations the **crmsh** (or **crm**) tool can be used to interact with the cluster

Get list of the available resource providers

**sudo pcs resource providers**

Retrieve the list of all resource agents

**sudo pcs resource agents**

Narrow down the agents to a particular provider

**sudo pcs resource agents ocf:heartbeat**

Now, that we are more or less aware, let’s create the virtual IP address for the cluster

**sudo pcs resource create clha\_ip ocf:heartbeat:IPaddr2 ip=<virtual-ip-address> cidr\_netmask=24 op monitor interval=30s**

Now, we can define the highly available web site as additional resource

**sudo pcs resource create clha\_web ocf:heartbeat:apache configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" op monitor interval=1min**

And then check the status

**sudo pcs status**

We can list just the resources with

**sudo pcs resource**

We may see that both resources reside on different nodes

We can define a constraint to guarantee that both will be together

**sudo pcs constraint colocation add clha\_web with clha\_ip INFINITY**

If we ask once again for the status

**sudo pcs status**

We will see that both are on one and the same node

We can go even further. We can set an order for starting the resources

**sudo pcs constraint order clha\_ip then clha\_web**

We can explore just the constraints with

**sudo pcs constraint**

Or add the **--full** flag at the end to see their IDs as well

### Testing

We can open a browser tab and navigate to **<clha\_ip>** or the **<virtual-ip-address>** we used for its creation

Next, we can stop one of the nodes

First check where the resources are running

**sudo pcs status**

Now stop the node, for example **M1**

**sudo pcs cluster stop m1.lsaa.lab**

Log on to the other node and check the status

**sudo pcs status**

Re-open the browser tab and check if the site is still available

Start the stopped node again

**sudo pcs cluster start m1.lsaa.lab**

Check the status again

**sudo pcs status**

## Part 3: Failover Clusters and Storage

### LVM Shared Storage

We can either start from the beginning, or continue from the previous part

No matter which way we will go, we must have the following infrastructure

Waterfall chart

Description automatically generated with medium confidence

#### Storage Preparation

First, we must have an **iSCSI** target (**M1**)

Then, we must authenticate both nodes with the target (**M2** and **M3**)

##### Target Preparation

Log on to **M1**

Install the required package

**sudo dnf install targetcli**

Create a folder to store the **iSCSI** disk files

**sudo mkdir /var/lib/iscsi-images**

Start the administration tool

**sudo targetcli**

Switch to the **fileio** backend

**cd backstores/fileio**

Create an **iSCSI** disk

**create D1 /var/lib/iscsi-images/D1.img 2G**

Switch to the **iscsi** functions

**cd /iscsi**

Define a new target

**create iqn.2021-09.lab.lsaa:m1.tgt1**

Enter the target

**cd iqn.2021-09.lab.lsaa:m1.tgt1/tpg1/luns**

Create a LUN using the disk created earlier

**create /backstores/fileio/D1**

Adjust the access to the resource

**cd ../acls**

Register the initiators 1 (**M2**) and 2 (**M3**)

**create iqn.2021-09.lab.lsaa:m2.init1**

**create iqn.2021-09.lab.lsaa:m3.init1**

Enter the record (if not there already)

**cd iqn.2021-09.lab.lsaa:m2.init1/**

Set user and password

**set auth userid=demo**

**set auth password=demo**

Switch to the other one and set the credentials

**cd ../iqn.2021-09.lab.lsaa:m3.init1/**

**set auth userid=demo**

**set auth password=demo**

Exit the administrative tool

**exit**

Adjust the firewall

**sudo firewall-cmd --add-service iscsi-target --permanent**

**sudo firewall-cmd --reload**

Enable and start the **target** service

**sudo systemctl enable --now target**

##### Initiator Preparation

Log on to the **M2** machine

Install the initiator package

**sudo dnf install iscsi-initiator-utils**

Open the initiator configuration file for editing

**sudo vi /etc/iscsi/initiatorname.iscsi**

Set the name to match to your situation, for example **iqn.2021-09.lab.lsaa:m2.init1**

Save and close the file

Adjust the authentication settings in **/etc/iscsi/iscsid.conf** file

**sudo vi /etc/iscsi/iscsid.conf**

Uncomment **node.session.auth.authmethod** = CHAP (line 58)

Uncomment and adjust **node.session.auth.username** and **node.session.auth.password** (lines 69 and 70)

Save and close

Initiate a target discovery with

**sudo iscsiadm -m discovery -t sendtargets -p <m1-name>**

Confirm what we have discovered

**sudo iscsiadm -m node -o show**

Login to the target

**sudo iscsiadm -m node --login**

Confirm the established session

**sudo iscsiadm -m session -o show**

Repeat the procedure on **M3** as well

#### Cluster Preparation

Next step is to spin up the cluster on nodes **M2** and **M3**

Remember to use the **FQDNs** (for example, **m2.lsaa.lab** instead of **m2**) where applicable

Refer to sections ***Installation*** and***Configuration*** in***Part 2: Failover Clusters***

#### Shared LVM

Log on to machine **M2**

Open the file **/etc/lvm/lvm.conf** for editing

**sudo vi /etc/lvm/lvm.conf**

And change the **system\_id\_source** (line 1235) to

**system\_id\_source = "uname"**

Save and close the file

Check that the **LVM** system ID matches the node name returned by **uname**

**sudo lvm systemid**

**uname -n**

Repeat the procedure on **M3** as well

Return on **M2**

Create (or re-create) the partition with

**sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 16384s -0m set 1 lvm on**

Initialize it as physical volume

**sudo pvcreate /dev/sdb1**

Create a volume group

**sudo vgcreate vg\_ha /dev/sdb1**

Check if the system id is correctly applied

**sudo vgs -o+systemid**

Create a logical volume

**sudo lvcreate -l 100%FREE -n lv\_ha vg\_ha**

We can check the result with

**sudo lvs**

Create a filesystem

**sudo mkfs.ext4 /dev/vg\_ha/lv\_ha**

#### Turn Off Automounting

We must make sure that the volume groups that will be managed by Pacemaker won’t be auto-loaded by the system

Let’s check currently known volume groups on **M2**

**sudo vgs --noheadings -o vg\_name**

Depending on our configuration we may see one (we just created) or more

Open again the **LVM** configuration file

**sudo vi /etc/lvm/lvm.conf**

Go to row 1416 and paste the following

**auto\_activation\_volume\_list = []**

Please note that if on your system there are other (for example system) volume groups, their names should be included in this list. Just the one managed by **Pacemaker** should be absent

Save and close the file

Rebuild the **initramfs** by executing

**sudo dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)**

And reboot the node

Repeat the procedure on **M3** as well

#### Resources and Resource Groups

Return on **M2** to finalize the setup

Check that everything with the cluster is okay

**sudo pcs cluster status**

Create the mount point on all nodes (**M2** and **M3** in our case)

**sudo mkdir /shared-lvm**

Being on **M2**, create a new cluster resource

**sudo pcs resource create lvm\_ha ocf:heartbeat:LVM-activate vgname=vg\_ha vg\_access\_mode=system\_id --group ha\_group**

We can check the status

**sudo pcs status**

Or just the resources

**sudo pcs resource status**

Next, we must create the file system resource with

**sudo pcs resource create lvm\_fs ocf:heartbeat:Filesystem device=/dev/vg\_ha/lv\_ha directory=/shared-lvm fstype=ext4 --group ha\_group**

Let’s check again the cluster and the resources and if the mount was successful

**sudo pcs status**

**sudo pcs resource status**

**df -hT**

Create a simple text file there with

**echo 'Hello from Shared LVM' | sudo tee /shared-lvm/readme.txt**

#### Failover Test

We can stop first node **M2**

**sudo pcs node standby m2.lsaa.lab**

And then on the second node **M3** we can check if the resource is there and working

**sudo pcs status**

**lsblk**

**cat /shared-lvm/readme.txt**

Now, while on the second node bring back the first node

**sudo pcs node unstandby m2.lsaa.lab**

Check the status again

**sudo pcs status**

And move back the resource (if you want) to it with

**sudo pcs resource move lvm\_ha m2.lsaa.lab**

### NFS Cluster Resource

Let’s build on the previous set of tasks by adding **NFS** capabilities to our cluster while utilizing the shared **LVM**

Chart, waterfall chart

Description automatically generated

We can make a use of one additional station though – a client

#### Configuration

Log on to the first node **M2**

Adjust the firewall for **NFS**

**sudo firewall-cmd --add-service nfs --permanent**

**sudo firewall-cmd --add-service=mountd --permanent**

**sudo firewall-cmd --add-service=rpc-bind --permanent**

**sudo firewall-cmd --reload**

Repeat the firewall related steps on **M3** as well

Return on the node that runs the resources (assuming its node **M2**) and create required folders

The base folder we already have - **/shared-lvm**

We must create one more, for the shared information for the **nfsserver** resource

**sudo mkdir -p /shared-lvm/info**

And the actual exported folder

**sudo mkdir -p /shared-lvm/exports/share1**

Then, we must create the **nfsserver** resource

**sudo pcs resource create nfs\_daemon ocf:heartbeat:nfsserver nfs\_shared\_infodir=/shared-lvm/info nfs\_no\_notify=true --group ha\_group**

*You must adjust the names (resource, group, path) to match your settings*

Add the root **exportfs** resources to export the **/shared-lvm/exports** directory

**sudo pcs resource create nfs\_root ocf:heartbeat:exportfs clientspec=<network-address>/255.255.255.0 options=rw,sync,no\_root\_squash directory=/shared-lvm/exports fsid=0 --group ha\_group**

*You must adjust the names (resource, group, path) and the address to match your settings*

Add the actual share (export)

**sudo pcs resource create nfs\_share1 ocf:heartbeat:exportfs clientspec=<network-address>/255.255.255.0 options=rw,sync,no\_root\_squash directory=/shared-lvm/exports/share1 fsid=1 --group ha\_group**

*You must adjust the names (resource, group, path) and the address to match your settings*

Add the **floating IP** (or virtual IP) address resource that **NFS** clients will use to access the **NFS** share

**sudo pcs resource create nfs\_vip ocf:heartbeat:IPaddr2 ip=<virtual-ip-address> cidr\_netmask=24 --group ha\_group**

*You must adjust the names (resource, group) and the address to match your settings*

Add a **nfsnotify** resource for sending **NFSv3** reboot notifications once the entire **NFS** deployment has initialized

**sudo pcs resource create nfs\_notify ocf:heartbeat:nfsnotify source\_host=<virtual-ip-address> --group ha\_group**

*You must adjust the names (resource, group) and the address to match your settings*

Check the status

**sudo pcs status**

While still on the active node, check the **NFS** exports with

**sudo showmount -e**

#### Testing

Log on to the client (**M4**)

Install the required package

**sudo dnf install nfs-utils**

Check if the exports can be seen there

**sudo showmount -e <virtual-ip-address>**

Try to mount using **NFSv3**

**sudo mount -o "vers=3" <virtual-ip-address>:/shared-lvm/exports/share1 /mnt**

Check information about the mounted filesystem

**df -hT /mnt**

And unmount it

**sudo umount /mnt**

Try to mount the export but this time using **NFSv4**

**sudo mount -t nfs4 <virtual-ip-address>:share1 /mnt**

Check information about the mounted filesystem

**df -hT /mnt**

#### Failover Test

Return on the active node (for example **M2**)

Check the status of the cluster

**sudo pcs status**

And put the active node to standby

**sudo pcs node standby m2.lsaa.lab**

Go to the other (now the active one) node and check the status again

**sudo pcs status**

Go to the client machine and check if the mounted NFS export is working

Return on the second node and bring back the first node

**sudo pcs node unstandby m2.lsaa.lab**

Check the status again

**sudo pcs status**

And move back the resource (if you want or if not moved automatically) to it with

**sudo pcs resource move lvm\_ha m2.lsaa.lab**