# Practice M6: Clustering and High Availability (openSUSE)

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 **openSUSE Leap 15.3**

## 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 zypper install ipvsadm**

#### Preparation

##### IP Forwarding (Method 1)

Use this one, if using **wicked** as network management solution

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)

Use this one, if using **Network Manager** as network management solution

Put both network interfaces in the appropriate firewall zones

**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**

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 zypper install apache2**

Enable and start it

**sudo systemctl enable --now apache2**

Set a custom index page

**echo 'Hello from <machine-name>' | sudo tee /srv/www/htdocs/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/ipvsadm.rules**

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

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 zypper 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 zypper install apache2**

Enable and start it

**sudo systemctl enable --now apache2**

Repeat this on **M4** as well

#### Configuration (Web Servers)

Set a custom index page

**echo 'Hello from WEB1' | sudo tee /srv/www/htdocs/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

**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**

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

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 zypper install apache2**

Set a custom index page

**echo 'Hello from WEB1' | sudo tee /srv/www/htdocs/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 log configuration file

**sudo vi /etc/apache2/mod\_log\_config.conf**

Change (substitute **%h** with **\"%{X-Forwarded-For}i\"**) the log format setting (row 15) to

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

Save and close the file

Enable and start it

**sudo systemctl enable --now apache2**

Repeat this on **M3** as well

#### Preparation (Load Balancer)

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

**sudo zypper install haproxy**

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

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

Change the port for the statistics listener from **80** to **8080** on rows 30 and 31

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**

We can add a separate line like

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

Otherwise, the **HAProxy** logs will go to **/var/log/localmessages**

Save and close the file

Restart the **rsyslog** service if you made any changes

**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**

Should you want to be able to see the statistics as well, open the port you used above

**sudo firewall-cmd --add-port 8080/tcp --permanent**

And then reload the configuration

**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 zypper install apache2**

Create a new file **/etc/apache2/conf.d/server-status.conf**

**sudo vi /etc/apache2/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 served by WEB1' | sudo tee /srv/www/htdocs/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

Lock the following package as will cause conflicts later

**sudo zypper al ruby2.5-rubygem-railties-5.2**

Install the required packages

**sudo zypper install ha-cluster-bootstrap**

Reboot the virtual machine if needed

Start the cluster creation procedure

**sudo ha-cluster-init**

On the first question answer with **Y**

Confirm the proposed network settings

On the question about the **SBD** answer with **N**

On the question about the **Virtual IP (admin)** answer with **Y**

Enter the address you want (for example, **192.168.81.160**)

On the question about the **QDevice** answer with **N**

Done. The cluster, even with one node is there

We can check its status with

**sudo crm status**

Now, log on to **M2**

Lock the following package as will cause conflicts later

**sudo zypper al ruby2.5-rubygem-railties-5.2**

Install the required packages

**sudo zypper install ha-cluster-bootstrap**

Reboot the virtual machine if needed

Then try to join **M2** to the cluster

**sudo ha-cluster-join -c m1.lsaa.lab**

When asked for the **root** password, enter it

On the first question answer with **Y**

Done. Our cluster should have two nodes now. Check it with

**sudo crm status**

Now, we can use either IP addresses (**<m1-ip-address>** or **<m2-ip-address>**) to log in to the web administrative console of the cluster. We can use even the **<virtual-admin-ip-address>** to do the same

For example, open a browser tab on the host and enter https://<virtual-admin-ip-address>:7630

Use the credentials generated during the initialization - user **hacluster** and password **linux**

### Resources

We can use the web interface (try it and you will like it) but instead, we will go with the command line approach

#### Resources (GUI)

Being in the admin interface click on **CONFIGURATION** and then on **Add Resource**

Select the **Primitive** option

For **Resource ID** enter **clha\_web**

Leave the **Template** selection empty

For **Class** select **ocf**

For **Provider** select **heartbeat**

For **Type** select **apache**

In the **Parameters** section select **configfile** from the drop-down list

Then select **statusurl** and enter **http://localhost/server-status**

Click on **Create**

Then return to the main (**Status**) page

Click the **Start** button next to the resource and confirm with **OK**

#### Resources (CLI)

Let’s create the virtual IP address for the cluster

**sudo crm configure primitive 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 crm configure primitive clha\_web ocf:heartbeat:apache configfile=/etc/apache2/httpd.conf statusurl="http://localhost/server-status" op monitor interval=1min**

And then check the status

**sudo crm status**

We may see that both resources reside on different nodes

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

**sudo crm configure colocation clha\_ip\_and\_web inf: clha\_web clha\_ip**

If we ask once again for the status

**sudo crm 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 crm configure order clha\_ip\_before\_web mandatory: clha\_ip clha\_web**

We can combine the following commands to explore deeper the status of the cluster

**sudo crm status**

**sudo crm configure show**

### 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 crm status**

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

**sudo crm node standby m1**

Log on to the other node and check the status

**sudo crm status**

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

Start the stopped node again

**sudo crm node online m1**

Check the status again

**sudo crm 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 zypper install targetcli-fb**

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 targetcli.service**

##### Initiator Preparation

Log on to the **M2** machine

Install the initiator package

**sudo zypper install open-iscsi**

Reboot the system and log on again

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**

Change **node.startup** mode to **automatic** on line 45

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**

Install the required packages

**sudo zypper install lvm2**

Reboot the machine

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

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

And change the **system\_id\_source** (line 1154) 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 1320 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 **the cluster** should be absent

Save and close the file

Rebuild the **initrd** by executing

**sudo dracut -H -f /boot/initrd-$(uname -r) $(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 crm status**

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

**sudo mkdir /shared-lvm**

Being on **M2**, create the two new cluster resources but this time in a different way

**sudo crm configure<<EOF**

**primitive lvm\_ha ocf:heartbeat:LVM-activate params vgname=vg\_ha vg\_access\_mode=system\_id**

**primitive lvm\_fs ocf:heartbeat:Filesystem params device=/dev/vg\_ha/lv\_ha directory=/shared-lvm fstype=ext4**

**group ha\_group lvm\_ha lvm\_fs**

**commit**

**EOF**

We can check the status

**sudo crm status**

Go to the node where the resources are and execute

**df -hT**

And then create a simple text file there with

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

#### Failover Test

We can stop the node on which the resources are working

**sudo crm node standby <node-with-resources>**

And then on the other node we can check if the resource is there and working

**sudo crm status**

**lsblk**

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

Now, bring back the stopped node

**sudo crm node online <stand-by-node>**

Check the status again

**sudo crm status**

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

**sudo crm group move ha\_group <other-node>**

### 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**

Install the required packages

**sudo zypper install nfs-kernel-server**

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 above (package installation + firewall) 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 necessary resources

**sudo crm configure<<EOF**

**primitive nfs\_daemon ocf:heartbeat:nfsserver params nfs\_shared\_infodir=/shared-lvm/info nfs\_no\_notify=true**

**primitive nfs\_root ocf:heartbeat:exportfs params clientspec=<network-address>/255.255.255.0 options=rw,sync,no\_root\_squash directory=/shared-lvm/exports fsid=0**

**primitive nfs\_share1 ocf:heartbeat:exportfs params clientspec=<network-address>/255.255.255.0 options=rw,sync,no\_root\_squash directory=/shared-lvm/exports/share1 fsid=1**

**primitive nfs\_vip ocf:heartbeat:IPaddr2 params ip=<virtual-ip-address> cidr\_netmask=24**

**modgroup ha\_group add nfs\_daemon**

**modgroup ha\_group add nfs\_root**

**modgroup ha\_group add nfs\_share1**

**modgroup ha\_group add nfs\_vip**

**commit**

**EOF**

Check the status

**sudo crm 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 zypper install nfs-client**

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 crm status**

And put the active node to standby

**sudo crm node standby <node-with-resources>**

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

**sudo crm 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 crm node online <stand-by-node>**

Check the status again

**sudo crm status**

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

**sudo crm group move ha\_group <other-node>**