Configure and Maintain High Availability/Clustering

Basically,**clusters** fall into four major types that is this:-

(1.)**Storage**: provide a consistent file system image across servers in a cluster, allowing the servers to simultaneously read and write to a single shared file system.

(2.) **High Availability:** eliminate single points of failure and by failing over services from one cluster node to another in case a node goes becomes inoperative.

(3.) **Load Balancing:** dispatch network service requests to multiple cluster nodes to balance the request load among the cluster nodes.

(4.) **High Performance:** carry out parallel or concurrent processing, thus helping to improve performance of applications.

To setup a cluster,Today we will use two Linux virtual machine like servers:

node1 **192.168.122.97**

node2 192.168.122.113

Configuring Local DNS Settings on Each Server/VM

in this step i used two servers to communicate to each other,i need to configure the appropriate local DNS setting in the **/etc /hosts** files on boths servers or vm.

$ Sudo vim /etc /hosts

Add the following entry with actual IP address of vitual machine like this-

192.168.122.97 node1.ha.com

192.168.122.113 node2.ha.com

Installing Nginx Server and epel-release on both Virtual machine

In this step install **epel-release** and **nginx server** on node1 virtual machine following run command-

[root@node1 ~]#yum install epel-release

[root@node1 ~]# yum install nginx

```
[root@node1 ~]# yum install nginx
Extra Packages for Enterprise Linux 8 - x86_64                  137 kB/s |  11 MB     01:22
Extra Packages for Enterprise Linux Modular 8 - x86_64           59 kB/s | 979 kB     00:16
Last metadata expiration check: 0:00:12 ago on Saturday 22 January 2022 09:01:18 PM IST.
Dependencies resolved.
================================================================================================================================
 Package                          Architecture      Version                                    Repository             Size
================================================================================================================================
Installing:
 nginx                            x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream             570 k
Installing dependencies:
 nginx-all-modules                noarch            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              23 k
 nginx-filesystem                 noarch            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              24 k
 nginx-mod-http-image-filter      x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              35 k
 nginx-mod-http-perl              x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              45 k
 nginx-mod-http-xslt-filter       x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              33 k
 nginx-mod-mail                   x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              64 k
 nginx-mod-stream                 x86_64            1:1.14.1-9.module_el8.0.0+184+e34fea82     appstream              85 k
Enabling module streams:
 nginx                                              1.14
```

After that, start the Nginx service for now and enable it to auto-start at boot time, then check if it's up and running using the systemctl command in both vm.

```
[root@node1 ~]# systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@node1 ~]# systemctl start nginx
[root@node1 ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2022-01-22 21:02:32 IST; 6s ago
  Process: 3383 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 3381 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 3380 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 3385 (nginx)
    Tasks: 3 (limit: 11117)
   Memory: 10.2M
   CGroup: /system.slice/nginx.service
           ├─3385 nginx: master process /usr/sbin/nginx
           ├─3386 nginx: worker process
           └─3387 nginx: worker process

Jan 22 21:02:30 node1.ha.com systemd[1]: Starting The nginx HTTP and reverse proxy server...
Jan 22 21:02:32 node1.ha.com nginx[3381]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jan 22 21:02:32 node1.ha.com nginx[3381]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jan 22 21:02:32 node1.ha.com systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Jan 22 21:02:32 node1.ha.com systemd[1]: Started The nginx HTTP and reverse proxy server.
```

After staring nginx service i need to create custom web page for testing operation on both vm.here we will modifiy the content of default file that is **index.html,** run following commands on both vm(node1 and node2)-

```
[root@node1 ~]# echo "This is the default page for node1.ha.com" | sudo tee /usr/share/nginx/html/index.html
This is the default page for node1.ha.com
[root@node1 ~]#
```

Installing and Configuring Corosync and Pacemaker

Next, i have to install **Pacemaker, Corosync,** and **Pcs** on both vm run following command-

```
[root@node1 ~]# yum --enablerepo=ha -y install pacemaker pcs
CentOS Linux 8 - HighAvailability                                  272 kB/s | 523 kB     00:01
Last metadata expiration check: 0:00:01 ago on Saturday 22 January 2022 09:35:09 PM IST.
Dependencies resolved.
================================================================================================================================
 Package                          Architecture     Version                                  Repository           Size
================================================================================================================================
Installing:
 pacemaker                        x86_64           2.1.0-8.el8                              ha                  456 k
 pcs                              x86_64           0.10.8-1.el8                             ha                   13 M
Installing dependencies:
```

Once the installation is complete on both vm, make sure that **pcs daemon** is running on both vm.

```
[root@node1 ~]# systemctl status pcsd
● pcsd.service - PCS GUI and remote configuration interface
   Loaded: loaded (/usr/lib/systemd/system/pcsd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:pcsd(8)
           man:pcs(8)
[root@node1 ~]# systemctl enable pcsd
Created symlink /etc/systemd/system/multi-user.target.wants/pcsd.service → /usr/lib/systemd/system/pcsd.
[root@node1 ~]# systemctl start pcsd
[root@node1 ~]# systemctl status pcsd
● pcsd.service - PCS GUI and remote configuration interface
   Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2022-01-22 21:40:13 IST; 1min 23s ago
     Docs: man:pcsd(8)
           man:pcs(8)
 Main PID: 5088 (pcsd)
    Tasks: 1 (limit: 11117)
   Memory: 27.4M
   CGroup: /system.slice/pcsd.service
           └─5088 /usr/libexec/platform-python -Es /usr/sbin/pcsd
```

Creating the Cluster

Installation time automatically, **“hacluster”** user is created. So here i need to set up the authentication needed for pcs. so Let’s start by creating a new password for the “hacluster” user with help of passwd command as you can see.

```
[root@node1 ~]# passwd hacluster
Changing password for user hacluster.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@node1 ~]#
[root@node1 ~]#
```

next step, here you can run the following command for authentication needed for **pcs** on node1.

```
[root@node1 ~]# pcs host auth node1.ha.com node2.ha.com
Username: hacluster
Password:
node1.ha.com: Authorized
node2.ha.com: Authorized
[root@node1 ~]#
```

In this step i have used **newcluster** on Node1 server as you can see given below-

```
[root@node1 ~]# pcs cluster setup --name newcluster node1.ha.com node2.ha.com
Error: Specified option '--name' is not supported in this command
Hint: Syntax has changed from previous version. See 'man pcs' -> Changes in pcs-0.10.
[root@node1 ~]# pcs cluster setup newcluster node1.ha.com node2.ha.com
No addresses specified for host 'node1.ha.com', using 'node1.ha.com'
No addresses specified for host 'node2.ha.com', using 'node2.ha.com'
Destroying cluster on hosts: 'node1.ha.com', 'node2.ha.com'...
node1.ha.com: Successfully destroyed cluster
node2.ha.com: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'node1.ha.com', 'node2.ha.com'
node1.ha.com: successful removal of the file 'pcsd settings'
node2.ha.com: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'node1.ha.com', 'node2.ha.com'
node1.ha.com: successful distribution of the file 'corosync authkey'
node1.ha.com: successful distribution of the file 'pacemaker authkey'
node2.ha.com: successful distribution of the file 'corosync authkey'
node2.ha.com: successful distribution of the file 'pacemaker authkey'
Sending 'corosync.conf' to 'node1.ha.com', 'node2.ha.com'
node1.ha.com: successful distribution of the file 'corosync.conf'
node2.ha.com: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
[root@node1 ~]#
```

Now enable the cluster on boot and start the service run command like this.

```
root@node1 ~]# pcs cluster enable --all
ode1.ha.com: Cluster Enabled
ode2.ha.com: Cluster Enabled
root@node1 ~]#
root@node1 ~]# pcs cluster start --all
ode2.ha.com: Starting Cluster...
ode1.ha.com: Starting Cluster...
```

Now i want to check if the cluster service is up and running using the following command.

[root@node1 ~]# **pcs status**

```
[root@node1 ~]# pcs status
Cluster name: newcluster

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: node2.ha.com (version 2.1.0-8.el8-7c3f660707) - partition with quorum
  * Last updated: Sat Jan 22 22:13:33 2022
  * Last change:  Sat Jan 22 22:13:27 2022 by hacluster via crmd on node2.ha.com
  * 2 nodes configured
  * 0 resource instances configured

Node List:
  * Online: [ node1.ha.com node2.ha.com ]

Full List of Resources:
  * No resources

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

As you can see above the output,there is a warning about no **STONITH** devices yet the **STONITH** is still enabled in the cluster.

First STONITH (or Shoot The Other Node In The Head), the fencing implementation on Pacemaker.

This component helps to protect your data from being **corrupted by concurrent access**.i will disable it since i have not configured any devices.

[root@node1 ~]# **pcs property set stonith-enabled=false**

also ignore the **Quorum** policy by running the following command:

[root@node1 ~]# **pcs property set no-quorum-policy=ignore**

```
[root@node1 ~]#
[root@node1 ~]# pcs property set stonith-enabled=false
[root@node1 ~]# pcs property set no-quorum-policy=ignore
[root@node1 ~]#
```

In this step run the following command to see the **property list** and important that the above options, **stonith** and the **quorum** policy are disabled.

```
[root@node1 ~]# pcs property list
Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: newcluster
 dc-version: 2.1.0-8.el8-7c3f660707
 have-watchdog: false
 no-quorum-policy: ignore
 stonith-enabled: false
```

Adding a Cluster Service

I will used configure a f**loating IP** which is the IP address that can be instantly moved from one vm/Server to another within the same network,which is **"floating IP"** is used for IPs which are not bound strictly to one single interface.

In our case, it will be used to support failover in a high-availability cluster.they have a few other use cases.like i need to configure the cluster in such a way that only the active member of the cluster “owns” or responds to the floating IP at any given time.

(1.) i am adding two cluster resources, **floating IP** address resource called **“floating_ip”** and a resource for the Nginx web server called **“http_server**”.

Our floating IP address is **192.168.122.20**

```
[root@node1 ~]#
[root@node1 ~]# pcs resource create floating_ip ocf:heartbeat:IPaddr2 ip=192.168.122.20 cidr_netmask=24 op monitor interval=60s
[root@node1 ~]# pcs resource create http_server ocf:heartbeat:nginx configfile="/etc/nginx/nginx.conf" op monitor timeout="20s" interva
l="60s"
```

(2.) add the second resource, named **http_server**. Here, resource agent of the service is **ocf:heartbeat:nginx.**

Once you have added the cluster services, issue the following command to check the status of resources**.**

```
[root@node1 ~]# pcs status resources
  * floating_ip (ocf::heartbeat:IPaddr2):        Started node1.ha.com
  * http_server (ocf::heartbeat:nginx):  Started node2.ha.com
```

you need to allow all traffic to Nginx and all **high availability** services through the firewall for proper communication between nodes.

```
[root@node1 ~]#
[root@node1 ~]# firewall-cmd --permanent --add-service=http
success
[root@node1 ~]# firewall-cmd --permanent --add-service=high-availability
Warning: ALREADY_ENABLED: high-availability
success
[root@node1 ~]# firewall-cmd --reload
success
```

Testing High Availability/Clustering

Open a web browser and hit IP address **192.168.122.20** you should see the default Nginx page from the **node2.ha.com** as shown in the screenshot.

If you check failure, run the following command to stop the cluster on the **node2.ha.com.**

```
node2.ha.com: Starting Cluster...
[root@node1 ~]# pcs status resources
  * floating_ip (ocf::heartbeat:IPaddr2):        Started node1.ha.com
  * http_server (ocf::heartbeat:nginx):  Started node2.ha.com
[root@node1 ~]# pcs cluster stop node2.ha.com
node2.ha.com: Stopping Cluster (pacemaker)...
node2.ha.com: Stopping Cluster (corosync)...
```

Note:- If we refresh on browser our web page so we got default page description (default web page node1.ha.com) and again when second time we refresh our web page so we got our web page showing (default for node2.ha.com),It means our cluster is working properly.

Then reload the page at **192.168.122.20**, you should now access the default Nginx web page from the node1.ha.com-