**AWS Resources**

**1 VPC --> 3 Subnets --> 1 IGW --> 2 Route Tables**

**3 EC2 on each subnet --> 2 Security Group Rule --> 1 IAM Role (3 permissions)**

**Follow the below steps to create the above stack:**

1. Create a VPC with CIDR range 172.31.0.0/16 (you can choose any range as per your need, make sure it doesn’t conflict with other VPC)
2. Create 3 Subnets using the created VPC and assign the given CIDR range (172.31.0.0/24, 172.31.1.0/24, 172.31.2.0/24)
3. Enable the auto assign IP allocation on each subnet
4. Create an IGW and attach it to the created VPC to get Internet access
5. Create a two Route tables for 3 subnets (one for 2 subnets and one for 1 subnet)
6. Once VPC setup is done, create a Redhat or Ubuntu based EC2 Instance using the created VPC and subnets (choose subnet as per your requirement, here one for each)
7. Create the SG and assign the rules per requirement and Launch the Instance
8. After the Instance comes Running status, note down the private IP’s of each instance and assign the name of it(you can assign it via tags while creating)
9. So we have three nodes 2 app nodes and 1 quorum node now, Master --> Backup --> Quorum
10. Add the below additional route into the created Route table. No need to add this route on all route tables, just create one RT and add this route and attach it to both app nodes.

170.30.0.10 <Select eni id of eth0 on the main instance>

1. Create three permissions individually and give a suitable name using below code.

* **Add a policy which allows the instances to modify their routing table**

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "Stmt1424870324001",  "Effect": "Allow",  "Action": [  "ec2:DescribeInstances",  "ec2:DescribeInstanceAttribute",  "ec2:DescribeTags"  ],  "Resource": "\*"  },  {  "Sid": "Stmt1424860166261",  "Action": [  "ec2:CreateRoute",  "ec2:DeleteRoute",  "ec2:DescribeRouteTables",  "ec2:ReplaceRoute"  ],  "Effect": "Allow",  "Resource": "\*"  }  ]  } |

* **Add a policy to stop and start the instances**

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "Stmt1424870324002",  "Effect": "Allow",  "Action": [  "ec2:DescribeInstances",  "ec2:DescribeInstanceAttribute",  "ec2:DescribeTags"  ],  "Resource": "\*"  },  {  "Sid": "Stmt1424860166262",  "Action": [  "ec2:ModifyInstanceAttribute",  "ec2:RebootInstances",  "ec2:StartInstances",  "ec2:StopInstances"  ],  "Effect": "Allow",  "Resource": "\*"  }  ]  } |

* Add a policy to allow publishing to an SNS topic for cluster services

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement":  [  {  "Effect":"Allow",  "Action":"sns:Publish",  "Resource":"arn:aws:sns:us-east-1:593322718047:snsAnpOpsgenie" }  ]  } |

1. Create a Role with a suitable name and attach these 3 permissions into it. Once done, attach this role itself on each of the created EC2 instances.
2. Below is the topic which needs to be used while publishing the message via SNS.

|  |
| --- |
| aws sns publish --topic-arn "arn:aws:sns:us-east-1:593322718047:snsOpsGenie" --message "SVFE Node 1 has failed over to Node 2" --subject "SVFE Node 1 Failed Over" |

Once the nodes are created and able to communicate within the same network, then perform the below tasks on the created EC2 instance.

**Tasks on EC2 Instance:**

1. You must need to logically assign the VIP which comes out of our created VPC CIDR range to both app nodes (Remember its not on quorum node)

Here the VIP we are choosing is 172.30.0.10/32

1. Create a new network interface (ifcfg-eth0:0) file on this path /etc/sysconfig/network-scripts and then copy/paste the below contents to allocate the above VIP to both app nodes.

|  |
| --- |
| Edit /etc/sysconfig/network-scripts/ifcfg-eth0:0 (assuming the main network interface is eth0)  DEVICE=eth0:0  Type=Ethernet  ONBOOT=yes  NM\_CONTROLLED=no  BOOTPROTO=none  IPADDR=172.30.0.10  PREFIX=16  IPV6INIT="no"  CLOUD\_NETCONFIG\_MANAGE='no' |

1. Reboot the instance after saving the above file and verify the VIP using ifconfig command.
2. Once VIP created, Install aws cli tools on all the three nodes and configure your access into it using below commands.

|  |
| --- |
| **Installation:**  curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"  yum -y install unzip wget; unzip awscli-bundle.zip  ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws  **Configuration:**  aws configure  AWS Access Key ID [None]:  AWS Secret Access Key [None]:  Default region name [None]: ap-south-1 <choose region where you created your resources>  Default output format [None]: |

1. Next, Install the Keepalived software/tool on both app nodes (not on quorum node) using below commands

|  |
| --- |
| yum install psmisc make openssl-devel libnl3-devel ipset-devel iptables-devel file-devel glib2-devel pcre2-devel libnftnl-devel libmnl-devel  cd /opt  sudo wget https://www.keepalived.org/software/keepalived-2.0.18.tar.gz  sudo tar -zxvf keepalived-2.0.18.tar.gz  cd keepalived-2.0.18  ./configure  make  sudo make install |

1. After installation, confirm the keepalived.conf file creation on /usr/local/etc/keepalived path where you need to specify scripts and config files which keepalived needs to look up.
2. Then, edit the /usr/local/etc/sysconfig/keepalived file and add the below path.

|  |
| --- |
| KEEPALIVED\_OPTIONS="-D -P -f /usr/local/etc/keepalived/keepalived.conf" |

1. Again edit the /usr/lib/systemd/system/keepalived.service file and confirm the below contents.

Note: Comment out the PIDFile line so that PIDFile is not defined. The service acts buggy if this isn’t done.

|  |
| --- |
| [Unit]  Description=LVS and VRRP High Availability Monitor  After=syslog.target network-online.target  [Service]  Type=forking  **#**PIDFile=/var/run/keepalived.pid  KillMode=process  # Confirm the correct environment file path below, this is the place where you set an env variable.  EnvironmentFile=-/usr/local/etc/sysconfig/keepalived  # Confirm the below execution path as well  ExecStart=/usr/local/sbin/keepalived $KEEPALIVED\_OPTIONS  ExecReload=/bin/kill -HUP $MAINPID  [Install]  WantedBy=multi-user.target |

1. Once the changes are done on the service file, then make sure to reload the daemon and disable the keepalived service on boot up.

|  |
| --- |
| sudo systemctl daemon-reload  sudo systemctl disable keepalived |

1. Now remove the contents of keepalived.conf file and paste the below contents over there. Just do some changes as per your requirements.

* **Master**

|  |
| --- |
| ! Configuration File for keepalived  global\_defs {  router\_id testapp server1  }  vrrp\_script check {    # Choose the correct path and your script name  script "/usr/local/etc/keepalived/check.sh"  #This script would exit 1 if bad, 0 if good  interval 120  }  vrrp\_instance testappservers {  state MASTER  interface eth0  virtual\_router\_id 33  priority 200  unicast\_src\_ip 172.31.1.64  unicast\_peer {  172.31.0.197  }  authentication {  auth\_type PASS  auth\_pass password  }  track\_script {  check  }  notify /usr/local/etc/keepalived/notify.sh  } |

* **Backup**

|  |
| --- |
| ! Configuration File for keepalived  global\_defs {  router\_id testappserver2  }  vrrp\_script check {    # Choose the correct path and your script name  script "/usr/local/etc/keepalived/check.sh"  #This script would exit 1 if bad, 0 if good  interval 120  }  vrrp\_instance testappservers {  state BACKUP  interface eth0  virtual\_router\_id 33  priority 100  unicast\_src\_ip 172.31.0.197  unicast\_peer {  172.31.1.64  }  authentication {  auth\_type PASS  auth\_pass password  }  track\_script {  check  }  notify /usr/local/etc/keepalived/notify.sh  } |

1. Save the config file and restart the keepalived.

**Scripts needs to be added:**

1. **check.sh**
2. **notify.sh**
3. **quoramcheck.svfe.sh**

**Master and Backup**

On both app nodes, make sure to add to both check.sh and notify.sh script files under /usr/local/etc/keepalived directory.

Once added provide the execution permissions to both scripts and change the below lines into the scripts.

In check.sh, modify the following variable

|  |
| --- |
| EC2\_QUORUM\_IP="<Private-IP>" #This should be the IP of the quorum node |

Change the service name unser check service block in script as per your requirements

In notify.sh, modify the following variables

|  |
| --- |
| EC2\_VIP\_CIDR="<Private-IP>" #This is the VIP CIDR that was decided on earlier  EC2\_QUORUM\_IP="<Private-IP>" #This should be the IP of the quorum node |

**Quoram**

Add the quoram.svfe.sh fileunderthe /root directory and provide the execution permission.

Edit the file, and modify the following variables

|  |
| --- |
| EC2\_INSTANCE\_IP="<Private-IP>" #This is the master instance IP  EC2\_OTHER\_INSTANCE\_IP="<Private-IP>" #This is the backup instance IP |

After the above changes, create cron rule for every 5 minutes to execute this script on quoram node.

|  |
| --- |
| \*/5 \* \* \* \* /root/quorumcheck.svfe.sh |

**Reboot**

Must follow this steps while rebooting

1. On Backup node, stop the keepalived

|  |
| --- |
| sudo systemctl stop keepalived |

1. Check the status of keepalived and service

|  |
| --- |
| sudo systemctl status keepalived  sudo systemctl status <service-name> |

1. On Master node, stop the keepalived

|  |
| --- |
| sudo systemctl stop keepalived |

1. Check the status of keepalived and service

|  |
| --- |
| sudo systemctl status keepalived  sudo systemctl status <service-name> |

1. On Master node, start the keepalived

|  |
| --- |
| sudo systemctl start keepalived |

1. Check the status of keepalived and service

|  |
| --- |
| sudo systemctl status keepalived  sudo systemctl status <service-name> |

1. Also check the logs and confirm the Master state.
2. On Backup node, start the keepalived and check the logs to confirm the Backup state.

|  |
| --- |
| sudo systemctl start keepalived |

**Troubleshooting**

1. Always check the iptable rules on your node and flush it completely if you don’t want to specify any rule (iptables -F).
2. Make sure the execution path which you have mentioned on the service file located at /usr/lib/systemd/system/keepalived.service is correct.
3. Confirm the aws configure in all nodes using aws cli commands, if you get access denied then aws not configured yet.
4. While aws configuring, make sure you choose the appropriate region where you created all the resources.
5. Follow the reboot order properly always and don’t enable any service manually after installation. Just mention the service name in script, keepalived do the rest process to start or stop the service.
6. Always confirm that you haven’t added VIP and Keepalived on quoram node.
7. Ensure the VIP route has been created on the main instance subnet route table.
8. Both app nodes must need to use the same route table rule to exchange the VIP’s whenever failover happens.
9. Confirm the ping communication in between for all the nodes before processing the script.
10. Always check the logs for all the troubleshooting (/var/log/messages)
11. Confirm the priority values of master and backup as master always needs to be higher priority than backup.
12. Commands can help to check the packets transfer

|  |
| --- |
| tcpdump proto 112  tshark -f “vrrp” |

### **Testing and Management**

### **Starting keepalived**

* The keepalived service does not start automatically, so we can keep a handle on the situation when there might be faults or issues that cause the machine to shut itself down.
* The starting sequence should be Master first, then the Backup. (systemctl start keepalived)
* A notification will be sent of the node going into BACKUP state first, then a notification that the node went to MASTER state. For the backup, it will just notify that it went into BACKUP state.

### **Stopping keepalived**

* If the intent is to reload keepalived on both nodes, then the sequence should be to stop the Backup first, then restart the Master, then start the Backup, so the Backup doesn't try to take over.
* It is best practice to verify that all mounts and services are actually stopped when stopping keepalived, or unexpected corruption could occur. Some services may not stop cleanly or leave behind lock files, etc. These must all be cleaned up if a clean startup of services is going to happen in the future. Here are some known lock files

|  |
| --- |
| SVFE  /tmp/svfe\_lock  /var/lock/subsys/svfe-adapter |

* There will be no notification of the actual stopping of the service.

### **Rebooting one of the nodes**

Because of the crontab script check running on the quorum node, you must disable this crontab check on the quorum node if you intend to reboot one of the primary or backup nodes. Then reenable the script after everything is back to normal. If you do not do this, the quorum node will shut it down and cause issues.

### **Faults**

* If the backup faults, it will recover if the fault goes away
* If the master faults, and gets shut down, if it is brought back up, it is possible it will try to become master, and if there is still an issue, it will first become master, then realize the fault and failover again. If the fault has been resolved, it will take over master again.

### **Testing**

* Checking the activity of keepalived can be done by watching the output of /var/log/messages
* To quickly check the last reported state change in the log:

|  |
| --- |
| grep -E '\_vrrp.\*Entering .\* STATE' /var/log/messages|tail -n1 |

* Save it as a script so you can run it quickly.

|  |
| --- |
| grep -E '\_vrrp.\*Entering .\* STATE' /var/log/messages|tail -n1 > /root/keepalivedstatus  chmod +x /root/keepalivedstatus |

#### **Application failure**

* To test failover due to application failure, we stop any of the services and wait for failover. The Backup should take over, and should shut down the Master.
* A notification should be received that the backup has taken over
* We should test all of the possible application failures once all of the applications and mount points are added to the check.sh script and notify.sh script.

#### **Split brain where the quorum node is the one that is split**

* On the quorum node, run the following

|  |
| --- |
| iptables -A INPUT -s main.instance.ip.address -j DROP  iptables -A INPUT -s backup.instance.ip.address -j DROP |

This will block traffic to the quorum node from both nodes, and will simulate a split where the quorum node is the only AZ that is split. The master and backup nodes will continue to talk to each other, so they will not failover.

Verify that the Master does not fail over, and that the master and backup both remain as master and backup, with no faults

There will be no notification of the quorum node being split, as it is not a fault condition. We could notify, but we need a way to not continually send emails every time it checks. It would be best to handle this from a monitoring perspective.

When done, flush iptables on the quorum node

|  |
| --- |
| sudo iptables -F |

#### **Split brain where the backup has split from the master, and the backup is the one that is inaccessible from the quorum node**

For this test it is recommended to set the frequency of the script on the quorum node to something higher, like every 5 minutes, so the chances of the quorum node shutting down the backup node before test confirmation is less. You can then wait for a few minutes after confirmation to see it get shut down by the quorum node.

On the quorum node, run the following

|  |
| --- |
| iptables -A INPUT -s backup.instance.ip.address -j DROP |

On the backup node, run the following

|  |
| --- |
| iptables -A INPUT -s master.instance.ip.address -j DROP |

On the master node, run the following

|  |
| --- |
| iptables -A INPUT -s backup.instance.ip.address -j DROP |

This will stop communication from the backup node to the quorum node, and from the master node to the backup node. The backup node should attempt to go into master mode, and fail when it doesn’t get contact to the quorum node or the other node, then it will go into fault mode, and the master node should remain the master

A notification should be received that the backup node has gone into master then fault mode

If the condition is left long enough, the quorum node will shut down the backup node. It should be brought back up only after clearing the iptables below on the master and quorum nodes.

When done, flush iptables on the quorum node

|  |
| --- |
| sudo iptables -F |

and on the backup node

|  |
| --- |
| sudo iptables -F |

and on the master node

i

|  |
| --- |
| sudo iptables -F |

The backup node should go from FAULT state to BACKUP state.

#### **Split brain where the master has split from the backup, and the master is the one that is inaccessible from the quorum node**

Run the following three steps in quick succession

On the quorum node, run the following

|  |
| --- |
| iptables -A INPUT -s master.instance.ip.address -j DROP |

On the backup node

|  |
| --- |
| iptables -A INPUT -s master.instance.ip.address -j DROP |

On the master node

|  |
| --- |
| iptables -A INPUT -s backup.instance.ip.address -j DROP |

The master should detect that it has gone into fault, since it can't ping the quorum node or the other node.

Around the same time, the backup node should become master, since it cannot ping the other node, and it can ping the quorum node.

Both the backup node and the quorum node will attempt to shutdown (STONITH) the original master node.

Verify that the original master node gets shut down, and that the backup node has become the master.

A notification should be received that the backup has become the master

When done, run the following on the quorum node

|  |
| --- |
| sudo iptables -F |

and this on the backup node

|  |
| --- |
| sudo iptables -F |

and restart the master node from the AWS control panel.

You will need to manually start keepalived on the master, and verify that it takes back over.

#### **Neither node can communicate with the quorum node or each other**

The backup node will want to become master due to loss of communication with the master node, but it will go into a fault state when it can't ping the quorum node. The master node will also go into fault due to loss of communication with the quorum node and it's partner.

On the quorum node, do the following

|  |
| --- |
| iptables -A INPUT -s master.instance.ip.address -j DROP  iptables -A INPUT -s backup.instance.ip.address -j DROP |

On the backup node

|  |
| --- |
| iptables -A INPUT -s master.instance.ip.address -j DROP |

On the master node

|  |
| --- |
| iptables -A INPUT -s backup.instance.ip.address -j DROP |

Verify that both nodes are in fault state

A notification should be received that the backup went into master and then fault state. The master should send notification that it went into fault state also.

When done

on the backup node,

|  |
| --- |
| sudo systemctl stop keepalived  sudo iptables -F |

on the master node,

|  |
| --- |
| sudo iptables -F |

and on the quorum node

|  |
| --- |
| sudo iptables -F |

The master node will recover from fault status and take over as MASTER

After this has happened, on the backup node

|  |
| --- |
| systemctl restart keepalived |

The master node will recover from fault status and take over as MASTER

## Additional Notes About HA Service Monitoring

### **SVFE**

**Ports**

|  |
| --- |
| 9000 svfe\_adapter  7777 tcpcomms7  7051 tcpcomms  7055 tcpcomms  7057 tcpcomms  4083 tcpcomms |

#### 