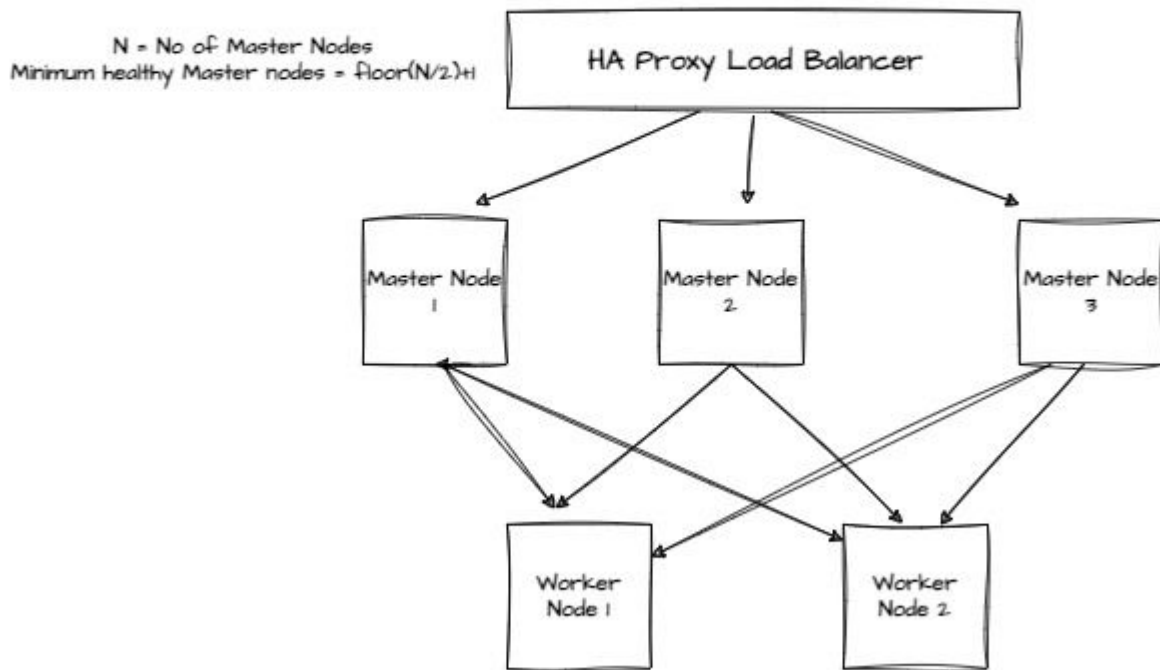


High Availability Kubernetes Cluster



Highly Available K8s Cluster

To set up a highly available Kubernetes cluster with two master nodes and three worker nodes without using a cloud load balancer, you can use a virtual machine to act as a load balancer for the API server. Here are the detailed steps for setting up such a cluster:

Prerequisites

- 3 master nodes
- 3 worker nodes
- 1 load balancer node
- All nodes should be running a Linux distribution like Ubuntu

Step 1: Prepare the Load Balancer Node

1. Install HAProxy:

```
sudo apt-get update
```

```
sudo apt-get install -y haproxy
```

2. Configure HAProxy: Edit the HAProxy configuration file (/etc/haproxy/haproxy.cfg):

```
sudo nano /etc/haproxy/haproxy.cfg
```

Add the following configuration:

```
frontend kubernetes-frontend
  bind *:6443
  option tcplog
  mode tcp
  default_backend kubernetes-backend

backend kubernetes-backend
  mode tcp
  balance roundrobin
  option tcp-check
  server master1 <MASTER1_IP>:6443 check
  server master2 <MASTER2_IP>:6443 check
```

3. Restart HAProxy:

```
sudo systemctl restart haproxy
```

Step 2: Prepare All Nodes (Masters and Workers)

1. Install Docker, kubeadm, kubelet, and kubectl:

```
sudo apt-get update
sudo apt install docker.io -y
sudo chmod 666 /var/run/docker.sock
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg
sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt update
sudo apt install -y kubeadm=1.30.0-1.1 kubelet=1.30.0-1.1 kubectl=1.30.0-1.1
```

Step 3: Initialize the First Master Node

1. Initialize the first master node:

```
sudo kubeadm init --control-plane-endpoint "LOAD_BALANCER_IP:6443" --upload-certs --pod-
network-cidr=10.244.0.0/16
```

2. Set up kubeconfig for the first master node:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

3. Install Calico network plugin:

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

4. Install Ingress-NGINX Controller:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-
v0.49.0/deploy/static/provider/baremetal/deploy.yaml
```

Step 4: Join the Second & third Master Node

1. You can now join any number of the control-plane node running the following command on each as root:

```
kubeadm join 13.234.120.104:6443 --token bfugml.64jh62uu73814a3s \
--discovery-token-ca-cert-hash sha256:07a6d4a097c1427d541f9f057d7584ec08f49c23859e133883eece6f334ff2b0 \
--control-plane --certificate-key 8a3820db3d4c5342d0a654c9a29ceadc73c79aa8959f5098bc0ea78822878448
```

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

i-088ff315cc59a1415 (K8s-M1)

PublicIPs: 3.109.143.146 PrivateIPs: 172.31.37.251

2. Set up kubeconfig for the second and third master node:

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 5: Join the Worker Nodes

1. Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 13.234.120.104:6443 --token bfugml.64jh62uu73814a3s \
--discovery-token-ca-cert-hash sha256:07a6d4a097c1427d541f9f057d7584ec08f49c23859e133883eece6f334ff2b0
ubuntu@ip-172-31-37-251:~$
```

```
i-088ff315cc59a1415 (K8s-M1)
```

```
PublicIPs: 3.109.143.146 PrivateIPs: 172.31.37.251
```

Step 6: Verify the Cluster

```
ubuntu@ip-172-31-37-251:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-37-251    Ready    control-plane   12m   v1.30.0
ip-172-31-40-145    Ready    <none>         72s   v1.30.0
ip-172-31-40-188    Ready    <none>         60s   v1.30.0
ip-172-31-40-73     Ready    control-plane   6m12s v1.30.0
ip-172-31-42-100    Ready    control-plane   6m29s v1.30.0
ubuntu@ip-172-31-37-251:~$
ubuntu@ip-172-31-37-251:~$
```

```
i-088ff315cc59a1415 (K8s-M1)
```

```
PublicIPs: 3.109.143.146 PrivateIPs: 172.31.37.251
```

By following these steps, you will have a highly available Kubernetes cluster with three master nodes and two worker nodes, and a load balancer distributing traffic between the master nodes.

This setup ensures that if one master node fails, the other will continue to serve the API requests.

Verification

Step 1: Install etcdctl

1. Install etcdctl using apt(On Master-1)

```
sudo apt-get update
```

```
sudo apt-get install -y etcd-client
```

2. Check the health of the etcd cluster:

```
sudo ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --
```

```
cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/peer.crt --
```

```
key=/etc/kubernetes/pki/etcd/peer.key endpoint health
```

Step 2: Verify HAProxy Configuration and Functionality

1. Configure HAProxy Stats:

Add the stats configuration to (/etc/haproxy/haproxy.cfg)

```
listen stats
```

```
bind *:8404
```

```
mode http
```

```
stats enable
```

```
stats uri /
```

```
stats refresh 10s
```

```
stats admin if LOCALHOST
```

2. Restart HAProxy:

```
sudo systemctl restart haproxy
```

3. Check HAProxy Stats:

Access the stats page at `http://<LOAD_BALANCER_IP>:8404`.

Now let's move to Master Node 3 and save any deployment yaml file (Board Game).

You can find this file on github <https://github.com/Bijan1235/Boardgame.git>

Save this file as ds.yml

And the yaml file is(deployment-service.yaml);

apiVersion: apps/v1

kind: Deployment # Kubernetes resource kind we are creating

metadata:

name: boardgame-deployment

spec:

selector:

matchLabels:

app: boardgame

replicas: 2 # Number of replicas that will be created for this deployment

template:

metadata:

labels:

app: boardgame

spec:

containers:

- name: boardgame

image: bijan9438/boardgame:latest # Image that will be used to containers in the cluster

imagePullPolicy: Always

ports:

- containerPort: 8080 # The port that the container is running on in the cluster

apiVersion: v1 # Kubernetes API version

kind: Service # Kubernetes resource kind we are creating

For checking our cluster is Highly Available or not we have to remove one Master Node,
On my case I removed Master Node-3 (M3)

Successfully initiated termination (deletion) of i-094cd6506b93b25f9

Instances (1/6) info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
K8s-HALB	i-0282a9f0d0297ae04	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-234-120-104.ap...	13.234.120.104
K8s-M1	i-088ff315cc59a1415	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-3-109-143-146.ap...	3.109.143.146
K8s-M2	i-0f654274b4b4f2dfa	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-232-34-32.ap-s...	13.232.34.32
<input checked="" type="checkbox"/> K8s-M3	i-094cd6506b93b25f9	Terminated	t2.medium	-	View alarms +	ap-south-1a	-	-
K8s-W1	i-0ddcadea5c3f6877	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-201-46-220.ap-...	13.201.46.220

```
ubuntu@ip-172-31-40-73:~$  
Broadcast message from root@ip-172-31-40-73 (Sun 2024-08-18 08:45:38 UTC):  
The system will power off now!
```

i-094cd6506b93b25f9 (K8s-M3)

PublicIPs: 3.111.171.209 PrivateIPs: 172.31.40.73

HAProxy version 2.8.5-1ubuntu3, released 2024/04/01

Statistics Report for pid 2555

> General process information

pid = 2555 (process #1, nproc = 1, nthread = 2)
uptime = 0d 0h 16m 20s; warnings = 1
system limits: memmax = unlimited, ulimit-n = 524287
maxsock = 524287; maxconn = 262116; reached = 0; maxpipes = 0
current conn = 8; current pipes = 0; conn rate = 0/sec; bit rate = 0.244 kbps
Running tasks: 0/25; idle = 100 %

Display option: Scope: Hide DOWN servers Disable refresh Refresh now CSV export JSON export (schema)

External resources: Primary site Updates (v2.8) Online manual

kubernetes-frontent									
Queue	Cur	Max	Limit	Session rate	Cur	Max	Limit	Sessions	Bytes
Frontend	0	8	-	5	9	262116	27	27	208019

kubernetes-backend									
Queue	Cur	Max	Limit	Session rate	Cur	Max	Limit	Sessions	Bytes
master1	0	0	-	0	3	2	4	10	119698
master2	0	0	-	0	3	3	4	10	21198
master3	0	0	-	0	2	0	2	7	125127
Backend	0	0	-	0	8	5	9	262116	208019

stats									
Queue	Cur	Max	Limit	Session rate	Cur	Max	Limit	Sessions	Bytes
Frontend	0	2	-	1	2	262116	3	3	44677
Backend	0	0	-	0	0	262116	0	0	44677

Now to check go to other master nodes and use kubectl get all

```
ubuntu@ip-172-31-37-251:~$ kubectl get all  
NAME                                READY   STATUS    RESTARTS   AGE  
pod/boardgame-deployment-759944f5c7-6mrqg    1/1     Running   0           10m  
pod/boardgame-deployment-759944f5c7-tp5vm    1/1     Running   0           10m  
  
NAME                                TYPE           CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE  
service/boardgame-ssvc              LoadBalancer  10.97.81.169 <pending>     8080:31948/TCP   10m  
service/kubernetes                   ClusterIP      10.96.0.1    <none>        443/TCP          34m  
  
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/boardgame-deployment  2/2     2             2           10m  
  
NAME                                DESIRED   CURRENT   READY   AGE  
replicaset.apps/boardgame-deployment-759944f5c7  2         2         2       10m  
ubuntu@ip-172-31-37-251:~$
```

i-088ff315cc59a1415 (K8s-M1)

PublicIPs: 3.109.143.146 PrivateIPs: 172.31.37.251

Now refresh the application again and check the status...!!!

Number of Instances

EC2 Dashboard

EC2 Global View

Events

Instances

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshot

Instances (6/6) info

Find instance by attribute or tag (case-sensitive)

All states

Connect

Instance state

Actions

Launch instances

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input checked="" type="checkbox"/>	K8s-HALB	i-0282a9f0d0297ae04	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-234-120-104.ap...	13.234.120.104
<input checked="" type="checkbox"/>	K8s-M1	i-088ff315cc59a1415	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-3-109-143-146.ap...	3.109.143.146
<input checked="" type="checkbox"/>	K8s-M2	i-0f654274b4b4f2dfa	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-232-34-32.ap...	13.232.34.32
<input checked="" type="checkbox"/>	K8s-M3	i-094cd6506b93b25f9	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-3-111-171-209.ap...	3.111.171.209
<input checked="" type="checkbox"/>	K8s-W1	i-0ddcadea5c33f6877	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-201-46-220.ap...	13.201.46.220
<input checked="" type="checkbox"/>	K8s-W2	i-0af33ea2a46ea8bea	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-43-204-29-122.ap...	43.204.29.122

Ports that are used

Inbound rules (16)

Search

Manage tags

Edit inbound rules

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-01f4ucdbz57edcaa	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-02efca046ca5d5ed1	IPv4	Custom TCP	TCP	2000 - 11000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0288cc521e6b3217f	IPv4	Custom TCP	TCP	9100	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0f78a689d6699e3bd	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0b75f4ad67f308040	IPv4	Custom TCP	TCP	6443	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0bde85a3721c7c3b1	IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-07e4634b474d64...	IPv4	Custom TCP	TCP	6443	172.31.0.0/16	-
<input type="checkbox"/>	-	sgr-06975cc57d07f191a	IPv4	SMTP	TCP	25	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-068e603bb9bac0a...	IPv4	Custom TCP	TCP	2380	172.31.0.0/16	-
<input type="checkbox"/>	-	sgr-0a177e79bfb773111	IPv4	Custom TCP	TCP	587	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0cac31347b9321621	IPv4	SMTPS	TCP	465	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0afcea989108ed3e	IPv4	Custom TCP	TCP	2379	172.31.0.0/16	-
<input type="checkbox"/>	-	sgr-0e1b8f331720178e4	IPv4	Custom TCP	TCP	27017	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0807b7126614d2...	IPv4	Custom TCP	TCP	3000 - 10000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0b1cca1d2ef9f97b9	IPv4	HTTPS	TCP	443	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0f8c626790cb54e79	IPv4	HTTP	TCP	80	0.0.0.0/0	-

Acknowledgment

Special thanks to Aditya Jaiswal of "DevOps Shack" on YouTube for his invaluable guidance and tutorials, which were instrumental in the successful completion of this project. Thank you, Aditya!

