**Creating a Resilient Kubernetes Cluster: A Complete Setup Guide**

In today's rapidly evolving infrastructure landscape, Kubernetes has become a cornerstone for managing containerized applications with precision. Ensuring high availability within our Kubernetes environment is crucial for maintaining continuous service and resilience. In this guide, we’ll walk through the steps to establish a robust Kubernetes cluster with 3 master nodes, 3 worker nodes, and 1 High Availability (HA) node. Let’s dive in and build a reliable infrastructure together!

**Prerequisites**

Before diving in, ensure you have the following:

1. Operating System: Ubuntu 24.04 installed on all nodes.

2. SSH Access: SSH connectivity from the bastion host to all nodes.

**1. Configure the Bastion Host**

a. System Update:

>> sudo apt update && sudo apt upgrade -y

b. Install Required Packages:

>> sudo apt install -y curl apt-transport-https

c. Install Kubernetes CLI (`kubectl`):

>> curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo touch /etc/apt/sources.list.d/kubernetes.list

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update

sudo apt install -y kubectl

**2. Set Up the Master Nodes**

a. Install Docker and Kubernetes Components:

>> sudo apt update

>> sudo apt install -y docker.io

>> sudo apt install -y kubelet kubeadm kubectl

>> sudo apt-mark hold kubelet kubeadm kubectl

b. Initialize the Kubernetes Cluster on the Primary Master Node:

>> sudo kubeadm init --control-plane-endpoint "<bastion-host-ip>:<port>" --upload-certs --pod-network-cidr=10.244.0.0/16

**Replace `<bastion-host-ip>:<port>` with your HAProxy IP and port (usually 6443).**

c. Set Up `kubectl` for Access:

>> mkdir -p $HOME/.kube

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

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

d. Deploy a Network Plugin (Flannel):

>> kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

e. Join Additional Master Nodes:

Execute the join command provided by the `kubeadm init` output of the first master node on each additional master node.

**3. Configure the Worker Nodes**

a. Install Docker and Kubernetes Components:

>> sudo apt update

>> sudo apt install -y docker.io

>> sudo apt install -y kubelet kubeadm kubectl

>> sudo apt-mark hold kubelet kubeadm kubectl

b. Join Worker Nodes to the Cluster:

Run the join command from the `**kubeadm init**` output of the **master node** on each **worker node**.

**4. Set Up High Availability (HA)**

a. Configure HAProxy or Alternative Load Balancer on the HA Node:

For HAProxy, install it:

>> sudo apt install -y haproxy

Modify the HAProxy Configuration (`/etc/haproxy/haproxy.cfg`):

frontend kubernetes-api

bind \*:6443

default\_backend kubernetes-masters

backend kubernetes-masters

balance roundrobin

server master1 Master1\_IP:6443 check

server master2 Master2\_IP:6443 check

server master3 Master3\_IP:6443 check

Restart HAProxy:

>> sudo systemctl restart haproxy

b. Update Master Nodes to Use the HAProxy Endpoint:

Adjust the Kubernetes configuration files and settings to point to the HAProxy IP and port.

**5. Verify the Cluster**

a. Check Cluster Health:

>> kubectl get nodes

>> kubectl get pods --all-namespaces

b. Test Your Deployment:

Deploy a sample application to ensure the cluster is functioning correctly.

**Conclusion**

By following these steps, we have a robust, highly available Kubernetes cluster ready to support our containerized applications. A high-availability setup ensures that our infrastructure is resilient and maintains service continuity even in the face of node failures.