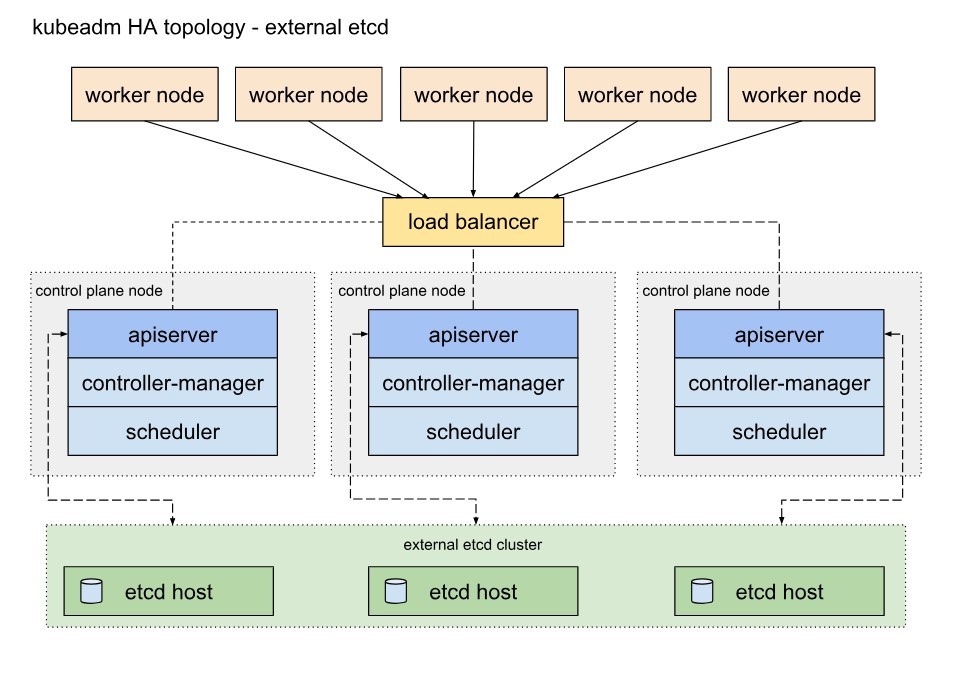
****

To set up this Kubernetes cluster, I am choosing to do it using the AWS cloud platform due to the following reasons:

* Elasticity and Scalability: AWS provides scalable infrastructure services, allowing the Kubernetes cluster to scale up or down based on demand, ensuring high availability and performance.
* Wide Range of Services: AWS offers a comprehensive suite of services, including networking, storage, security, and management tools, making it easier to integrate with Kubernetes for a robust and feature-rich environment.
* Security: AWS has a strong security framework, providing features such as IAM, VPC, and security groups, ensuring that the Kubernetes cluster is secure from potential threats.
* Global Reach: With data centers in multiple regions worldwide, AWS allows for the deployment of the Kubernetes cluster in various geographical locations, ensuring low latency and improved performance for users across the globe.
* Managed Kubernetes Services: AWS provides managed Kubernetes services like Amazon EKS (Elastic Kubernetes Service), which simplify the management of Kubernetes clusters by handling the underlying infrastructure, allowing you to focus more on the applications.
* Integration with Other AWS Services: AWS seamlessly integrates with various other AWS services, such as CloudWatch for monitoring, CloudTrail for auditing, and services like Amazon RDS, Amazon S3, and Amazon DynamoDB, which can be utilized by applications running in the Kubernetes cluster.
* Comprehensive Documentation and Support: AWS has extensive documentation, tutorials, and a large community, making it easier to find support and resources to set up and manage the Kubernetes cluster effectively.

The set up steps include:

1. Security Configuration Setup on the Base Image:

* Start with an Amazon Machine Image (AMI) with the necessary security configurations.
* Configure security groups to restrict access to necessary ports and services.
* Enable AWS Identity and Access Management (IAM) for secure access control.

1. Setting up Kubernetes Master Nodes:

* Launch two EC2 instances as Kubernetes master nodes in separate availability zones for high availability.
* Install the etcd database, on the master nodes.
* Install Kubeadm, Kubelet, and Kubectl on both master nodes.
* Initialize the Kubernetes cluster on the first master node and join the second master node to the cluster.

1. VIP Setup between the Two Master Nodes:

* Implement an Elastic IP address for the primary master node.
* Use Route 53 or an Elastic Load Balancer (ELB) with the primary node's Elastic IP as the virtual IP for the Kubernetes master nodes.

**NB**

I am not using Keepalived for the virtual IP because it is not most optimal approach in AWS due to the highly available services AWS provides, such as Elastic Load Balancing and Route 53.

Keepalived is commonly used in environments where direct control over IP failover is necessary, especially in on-premises or non-cloud environments. AWS has its own suite of services specifically designed for managing high availability, failover, and load balancing, which can be more effectively utilized to ensure the high availability of the Kubernetes cluster.

1. Setting up Kubernetes Worker Nodes:

* Launch two EC2 instances as Kubernetes worker nodes in separate availability zones.
* Install Kubeadm, Kubelet, and Kubectl on both worker nodes.
* Join the worker nodes to the Kubernetes cluster.

1. Monitoring Setup for All Servers:

* Utilize AWS CloudWatch for monitoring the Kubernetes cluster nodes and applications.
* Set up custom CloudWatch metrics and alarms for monitoring performance.
* Configure CloudWatch Logs for centralized log collection.

1. Centralized Log Server:

* Use Amazon Elasticsearch Service (Amazon ES) for centralized logging.
* Set up Fluentd or Fluent Bit on each node to collect and ship logs to Amazon ES.
* Use Kibana for log visualization and analysis.

7. Additional Security Measures

* Implement AWS Identity and Access Management (IAM) roles and policies for controlling access to AWS resources.
* Use Amazon VPC for network security and set up Security Groups for the EC2 instances and network ACLs for the subnets, it will enhance fine-grained control.

8. Regular Backups:

* Implement automated backups for critical data and configurations using AWS Backup or custom scripts.
* Store backups in Amazon S3 or Amazon EBS for durability and easy recovery.

9. Ongoing Maintenance and Updates:

* Set up AWS Systems Manager for automating patch management and maintaining the health of the EC2 instances.
* Regularly update Kubernetes components and perform necessary security patches and updates.