Certified Kubernetes Administrator Prep

Making Kubernetes Highly Available

Making Kubernetes Highly Available

- The challenges associated with making any platform “H.A.” are many.

- While the exam isn’t going to make you deploy a full HA kubernetes cluster, we should take a look at the process from a high level to gain an understanding of how it should work.

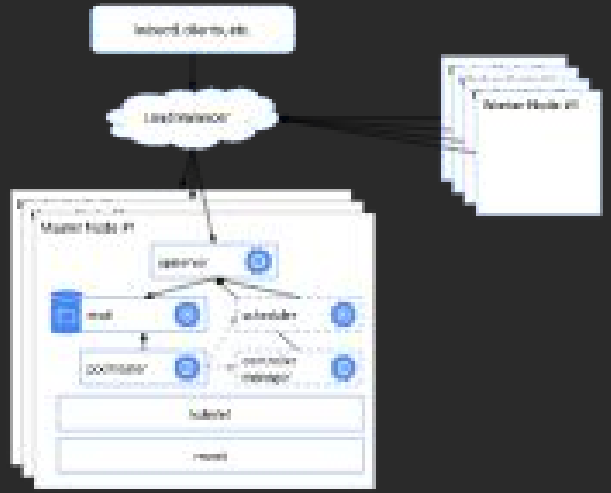
The Process

- Create the reliable nodes that will form our cluster.
- Set up a redundant and reliable storage service with a multinode deployment of etcd.
- Start replicated and load balanced Kubernetes API servers.
- Set up a master-elected kubernetes scheduler and controller manager daemons.

The Process

- Here is what our final system is going to look like.

- Notice that everything that needs the API server or any other service on the master goes through the load balancer, including the worker nodes.



Step One

- Make the Master node reliable.
- Ensure that the services automatically restart if they fail
- Kubelet already does this, so it's a convenient piece to use
- If the kubelet goes down, though, we need something to restart it
- Monit on Debian systems or systemctl on systemd-based systems.

Step Two - Storage Layer

- If we're going to make it H.A., then we've got to make sure that persistent storage is rock solid.
- Protect the data!
- If you have the data, you can rebuild almost anything else.
- Once the data is gone, the cluster is gone too.

Step Two - Storage Layer

- Clustered etcd already replicates the storage to all master instances in your cluster.
- To lose data, all three nodes would need to have their disks fail at the same time.
- The probability that this occurs is relatively low, so just running a replicated etcd cluster is reliable enough.
- Add additional reliability by increasing the size of the cluster from three to five nodes

Step Two - Storage Layer

- A multinode cluster of etcd

- If you use a cloud provider, then they usually provide this for you.
  - Example: Persistent Disk on the Google Cloud Platform.
- If you are running on physical machines, you can also use network attached redundant storage using an iSCSI or NFS interface. Alternatively, you can run a clustered file system like Gluster or Ceph.
- RAID array on each physical machine

Step Three - Replicated API Services

- Create the initial log file so that Docker will mount a file instead of a directory:
  - touch /var/log/kube-apiserver.log

- Next, we create a /srv/kubernetes/ directory on each node which should include:
  - basic_auth.csv - basic auth user and password
  - ca.crt - Certificate Authority cert
  - known_tokens.csv - tokens that entities (e.g. the kubelet) can use to talk to the apiserver
  - kubecfg.crt - Client certificate, public key
  - kubecfg.key - Client certificate, private key
  - server.cert - Server certificate, public key
  - server.key - Server certificate, private key

Step Three - Replicated API Services

- Either create this manually or copy it from a master node on a working cluster.
- Once these files exist, copy the kube-apiserver.yaml into /etc/kubernetes/manifests/ on each of our master nodes.
- The kubelet monitors this directory, and will automatically create an instance of the kube-apiserver container using the pod definition specified in the file.

Step Three - Replicated API Services

- If a network load balancer is set up, then access the cluster using the VIP and see traffic balancing between the apiserver instances.

- Setting up a load balancer will depend on the specifics of your platform.

- For external users of the API (e.g., the kubectl command line interface, continuous build pipelines, or other clients) remember to configure them to talk to the external load balancer's IP address.

Step Four - Controller/Scheduler Daemons

- Now we need to allow our state to change.
- Controller managers and scheduler.
- These processes must not modify the cluster's state simultaneously, use a lease-lock
- Each scheduler and controller manager can be launched with a --leader-elect flag

Step Four - Controller/Scheduler Daemons

- The scheduler and controller-manager can be configured to talk to the API server that is on the same node (i.e. 127.0.0.1).
  - It can also be configured to communicate using the load balanced IP address of the API servers.
- The scheduler and controller-manager will complete the leader election process mentioned before when using the --leader-elect flag.
- In case of a failure accessing the API server, the elected leader will not be able to renew the lease, causing a new leader to be elected.
- This is especially relevant when configuring the scheduler and controller-manager to access the API server via 127.0.0.1, and the API server on the same node is unavailable.

Installing Configuration Files

- Create empty log files on each node, so that Docker will mount the files and not make new directories:
  - touch /var/log/kube-scheduler.log
  - touch /var/log/kube-controller-manager.log

- Set up the descriptions of the scheduler and controller manager pods on each node by copying kube-scheduler.yaml and kube-controller-manager.yaml into the /etc/kubernetes/manifests/ directory.

- And once that's all done, we've now made our cluster highly available! Remember, if a worker node goes down, kubernetes will rapidly detect that and spin up replacement pods elsewhere in the cluster.