Consider the following example: In a HA cluster with 3 masters, a cold backup is taken sequentially starting from master 1, then 2, then 3. Each master is powered off (using the recommended approach in the previous sections), backed up, then powered on again. During this time period, master 2 and 3 are still running, and the cluster is still fully operational for end users. When restoring, the same order is used so master 1 is restored first, then 2, then 3.

The issue here is that each backup for the master nodes is taken at a completely different time, with master 3 being the last. This means that master 3 will have different data when restored, and when brought online last it will attempt to communicate with the existing cluster. Therefore, it's essential to ensure that in the event of a cluster failure, and all 3 masters need to be restored from a backup, that the order of restoration is the *reverse* of the order of backup. This process applies to restoring both hot and cold backups.

Verifying a restored backup

Restoring a cluster entirely depends on the hosting virtualization platform. The examples in this section uses VMWare to host the IBM Cloud Private installation, and therefore uses the Virtual Machine (VM) snapshot capabilities provided with the platform. Note that a VMWare snapshot is not a complete backup mechanism and is used here to demonstrate the practical application of the backup methods described in this chapter.

Using the previous example of a HA cluster, master nodes 1, 2, and 3 were backed up using VMWare snapshots while the machines were offline (cold snapshot) in sequence starting from master 1. Nginx workloads were deployed to simulate user deployments so that the cluster state in etcd was changed. To simulate a disaster, masters 1 and 2 were destroyed. At this point etcd is now running as a single node cluster, in read-only mode and cannot accept any write requests from Kubernetes.

The CoreOS documentation (found at `https://coreos.com/etcd/docs/latest/op-guide/failures.html`) provides a good explanation about different etcd failure scenarios and what it can or can't tolerate. In this situation, quorum is lost and as you cannot currently replace IBM Cloud Private master nodes using different IP addresses, you cannot recover. Therefore, the snapshots must be used to deploy masters 1 and 2, and restore all 3 masters to a previously working state.

Restore all 3 masters to a previous snapshot, power on the nodes and allow some time for all the containers to start. To ensure the etcd cluster has started successfully and the cluster is healthy, etcd provides a useful command line utility to query the cluster status, which can be downloaded to your local machine or ran from the etcd container itself. To get the cluster health status, complete the following steps from the IBM Cloud Private boot node (or whatever system has cluster access with `kubectl`):

1. Run `kubectl -n kube-system get pods | grep k8s-etcd` to retrieve the etc pod name (the format is `k8s-etcd-<node-ip>`):

   ```
   [root@icp-ha-boot ~]# kubectl -n kube-system get pods -o name| grep k8s-etcd
   k8s-etcd-172.24.19.201
   k8s-etcd-172.24.19.202
   k8s-etcd-172.24.19.203
   ```

2. Use any one of the pod names returned to execute an **`etcdctl cluster-health`** command, using one of the master node IP addresses in the `endpoint` parameter:

   ```
   [root@icp-ha-boot ~]# kubectl -n kube-system exec k8s-etcd-172.24.19.201 -- sh
   -c "export ETCDCTL_API=2; etcdctl --cert-file /etc/cfc/conf/etcd/client.pem
   --key-file /etc/cfc/conf/etcd/client-key.pem --ca-file
   /etc/cfc/conf/etcd/ca.pem --endpoints https://172.24.19.201:4001
   cluster-health"
   ```