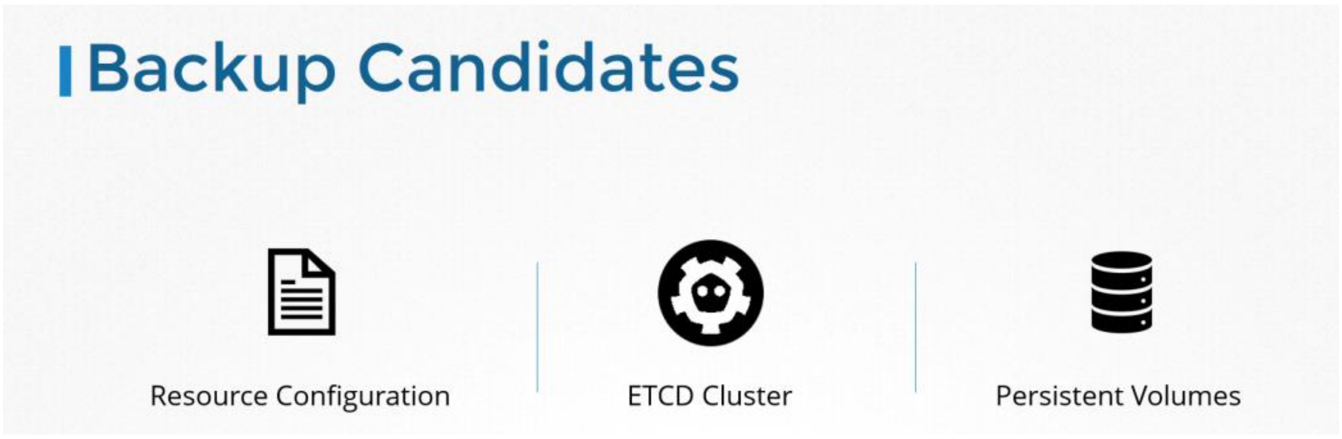
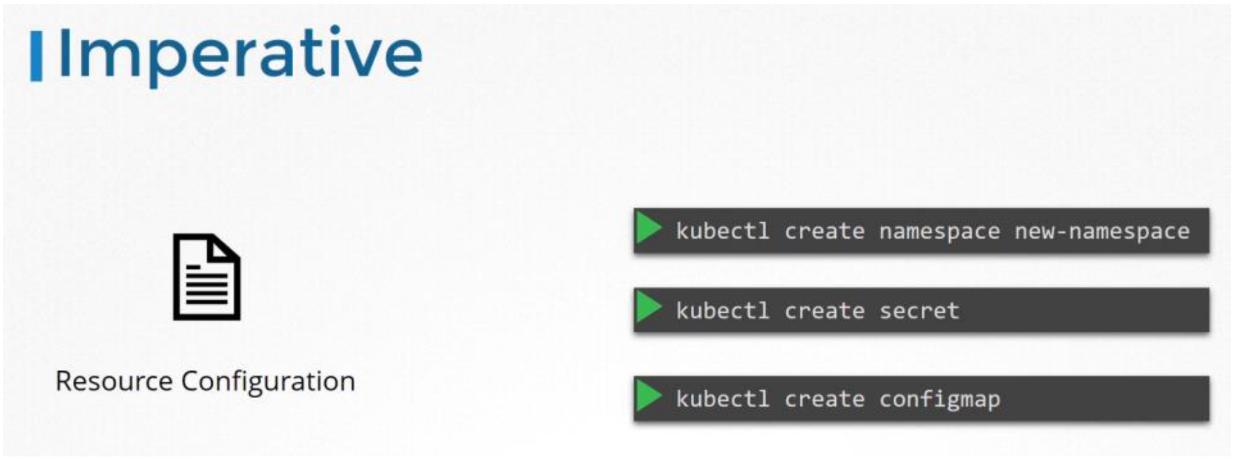
Monday, 24 October 2022 14.10

Backup and Restore



Backup Candidates



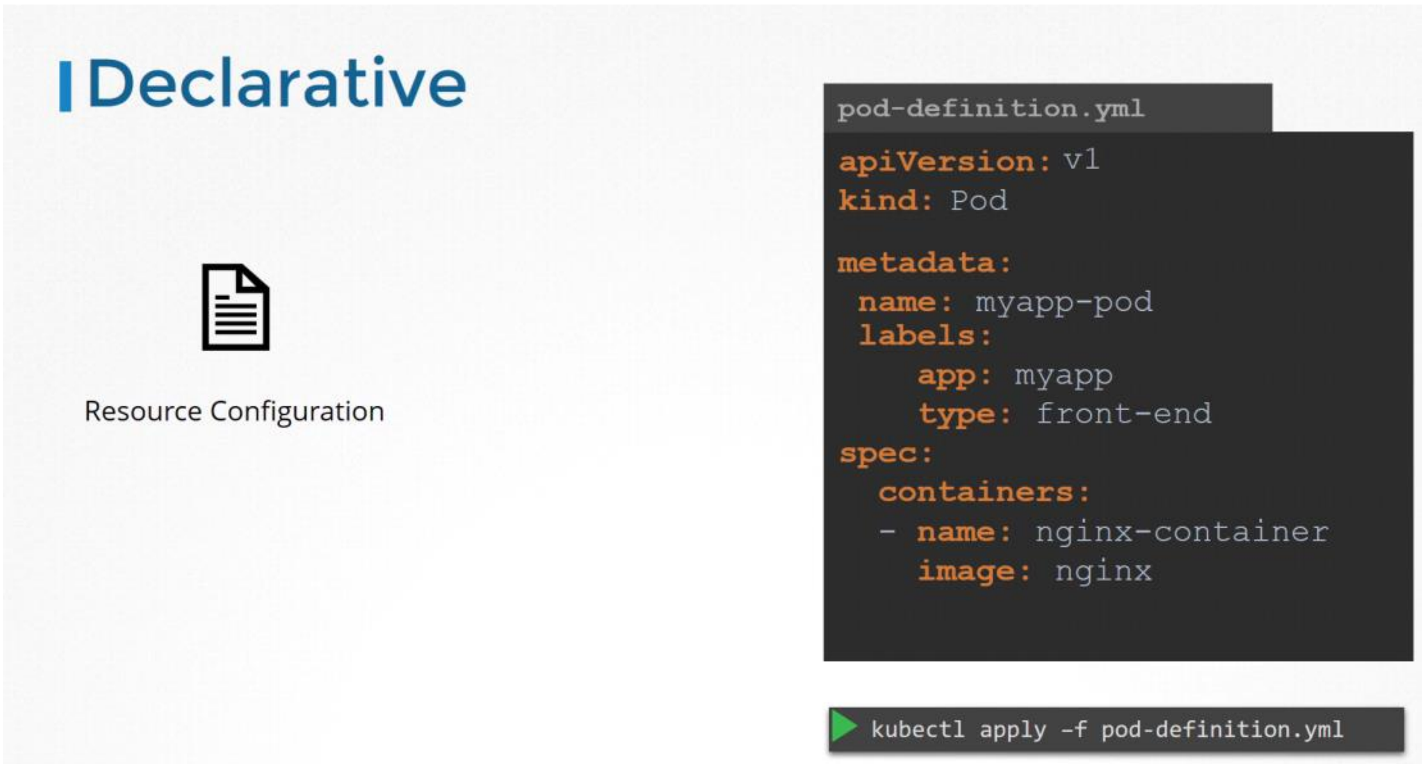
Imperative

Resource Configuration

```
kubectl create namespace new-namespace
```

```
kubectl create secret
```

```
kubectl create configmap
```



Declarative

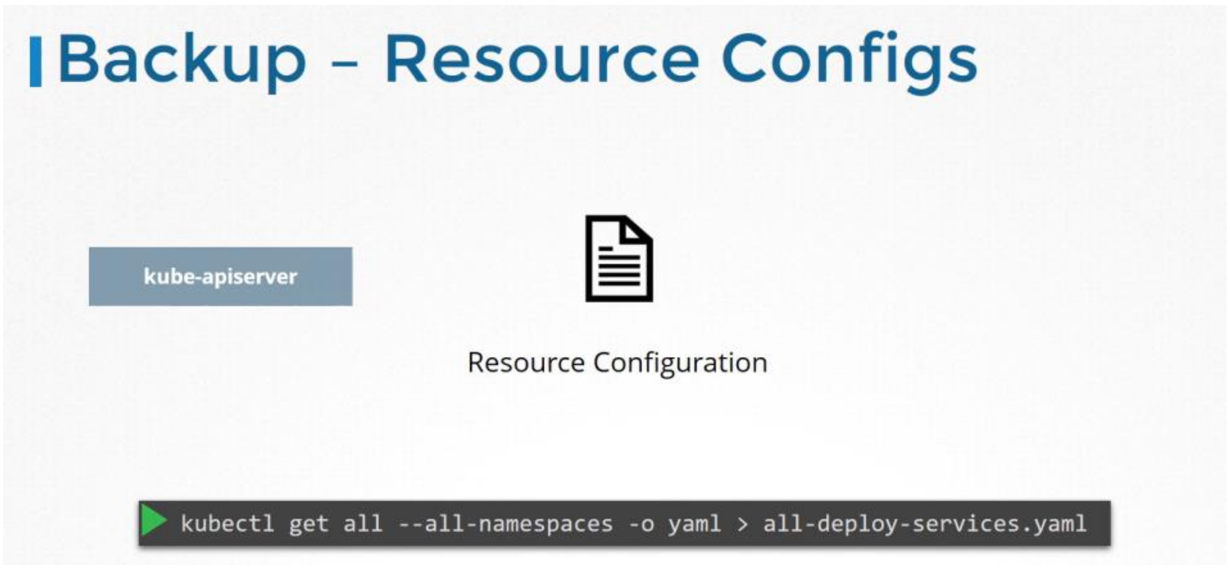
Resource Configuration

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

```
kubectl apply -f pod-definition.yml
```

GitHub



Backup – Resource Configs

kube-apiserver

Resource Configuration

```
kubectl get all --all-namespaces -o yaml > all-deploy-services.yaml
```

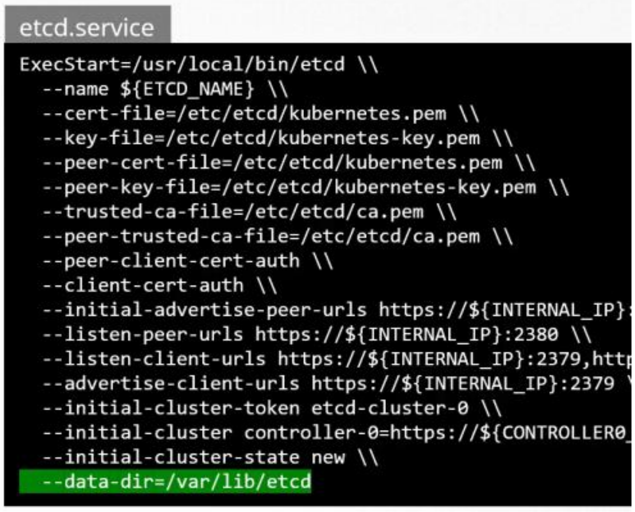
VELERO

Formerly called ARK by HeptIO

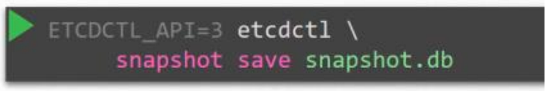
Backup - ETCD

ETCD Cluster

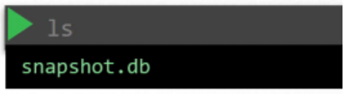
ETCD
M
ETCD
M
ETCD
M


etcd.service

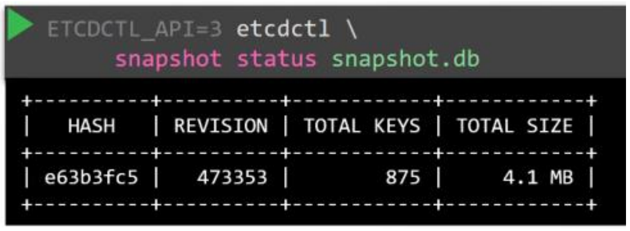
```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,http
  --advertise-client-urls https://${INTERNAL_IP}:2379 \
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster controller-0=https://${CONTROLLER0_
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
```



```
ETCDCTL_API=3 etcdctl \
      snapshot save snapshot.db
```



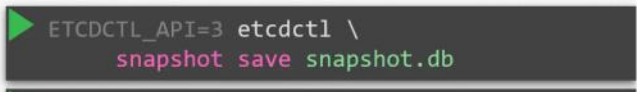
```
ls
snapshot.db
```



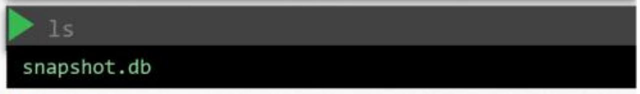
```
ETCDCTL_API=3 etcdctl \
      snapshot status snapshot.db

+----------+----------+------------+------------+
|   HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| e63b3fc5 |   473353 |        875 |     4.1 MB |
+----------+----------+------------+------------+
```

Restore - ETCD

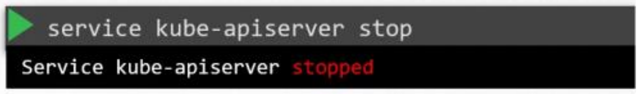
ETCD Cluster



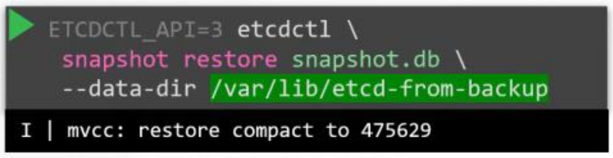
```
ETCDCTL_API=3 etcdctl \
      snapshot save snapshot.db
```



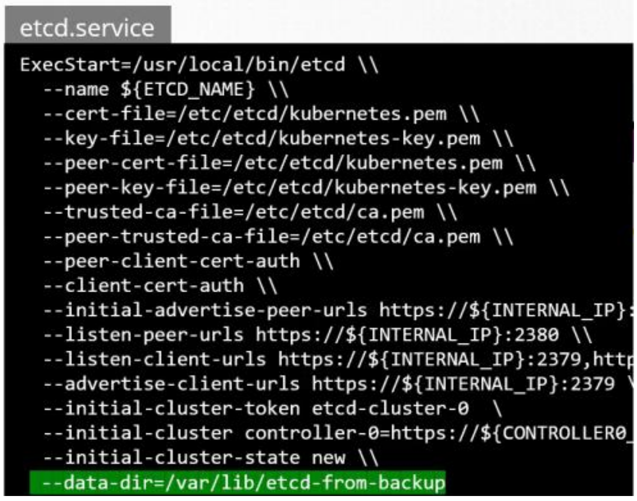
```
ls
snapshot.db
```



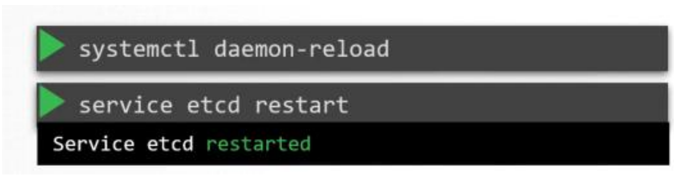
```
service kube-apiserver stop
Service kube-apiserver stopped
```



```
ETCDCTL_API=3 etcdctl \
  snapshot restore snapshot.db \
  --data-dir /var/lib/etcd-from-backup
I | mvcc: restore compact to 475629
```

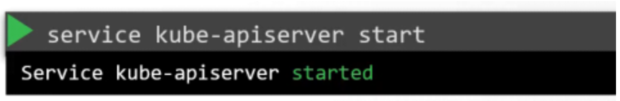

etcd.service

```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,http
  --advertise-client-urls https://${INTERNAL_IP}:2379 \
  --initial-cluster-token etcd-cluster-0  \
  --initial-cluster controller-0=https://${CONTROLLER0_
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd-from-backup
```

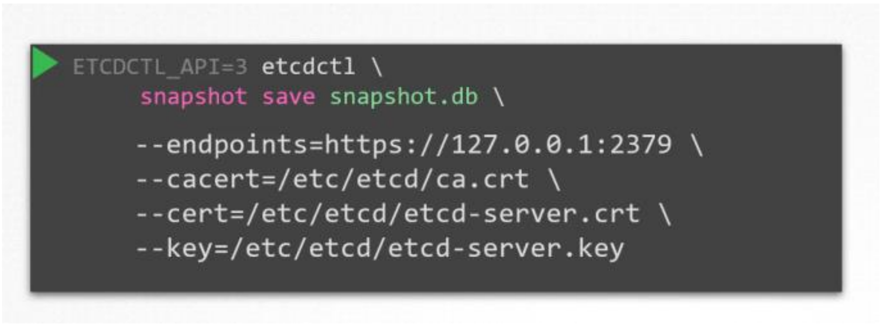


```
systemctl daemon-reload
```

```
service etcd restart
Service etcd restarted
```



```
service kube-apiserver start
Service kube-apiserver started
```



```
ETCDCTL_API=3 etcdctl \
     snapshot save snapshot.db \
     --endpoints=https://127.0.0.1:2379 \
     --cacert=/etc/etcd/ca.crt \
     --cert=/etc/etcd/etcd-server.crt \
     --key=/etc/etcd/etcd-server.key
```

Working with ETCDCTL

`etcdctl` is a command line client for etcd.

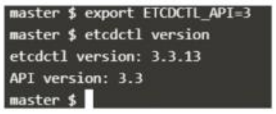
In all our Kubernetes Hands-on labs, the ETCD key-value database is deployed as a static pod on the master. The version used is v3.

To make use of etcdctl for tasks such as back up and restore, make sure that you set the ETCDCTL_API to 3.

You can do this by exporting the variable ETCDCTL_API prior to using the etcdctl client. This can be done as follows:

`export ETCDCTL_API=3`

On the **Master Node**:



```
master $ export ETCDCTL_API=3
master $ etcdctl version
etcdctl version: 3.3.13
API version: 3.3
master $
```

To see all the options for a specific sub-command, make use of the **-h or --help** flag.

For example, if you want to take a snapshot of etcd, use:

`etcdctl snapshot save -h` and keep a note of the mandatory global options.

Since our ETCD database is TLS-Enabled, the following options are mandatory:

`--cacert` verify certificates of TLS-enabled secure servers using this CA bundle

`--cert` identify secure client using this TLS certificate file

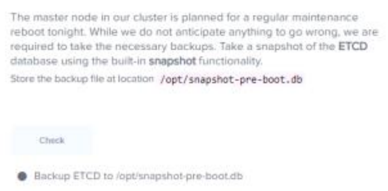
`--endpoints=[127.0.0.1:2379]` This is the default as ETCD is running on master node and exposed on localhost 2379.

`--key` identify secure client using this TLS key file

Similarly use the help option for **snapshot restore** to see all available options for restoring the backup.

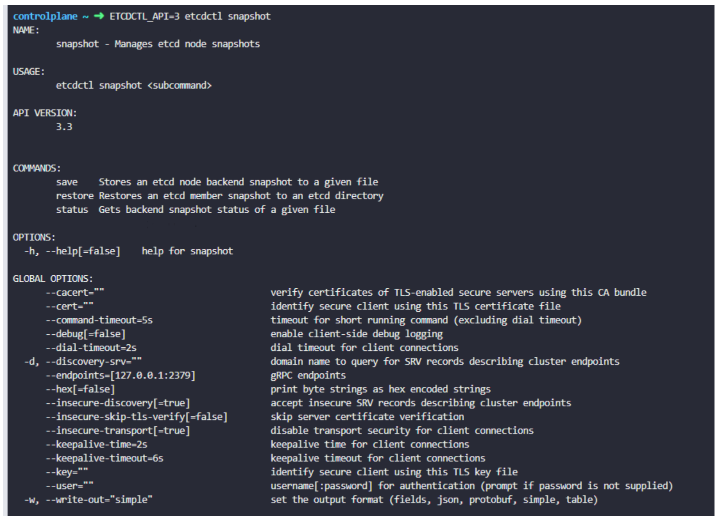
`etcdctl snapshot restore -h`

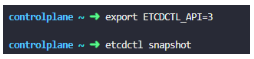
For a detailed explanation on how to make use of the etcdctl command line tool and work with the -h flags, check out the solution video for the Backup and Restore Lab.

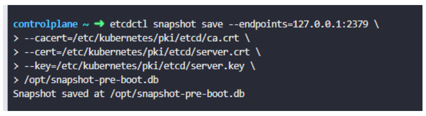

The master node in our cluster is planned for a regular maintenance reboot tonight. While we do not anticipate anything to go wrong, we are required to take the necessary backups. Take a snapshot of the **ETCD** database using the built-in **snapshot** functionality.

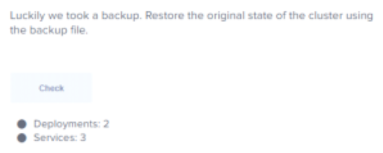
Store the backup file at location `/opt/snapshot-pre-boot.db`

● Backup ETCD to /opt/snapshot-pre-boot.db

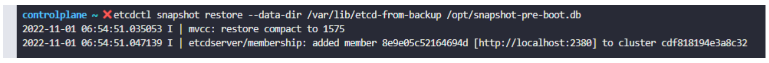

```
controlplane ~ ➜ ETCDCTL_API=3 etcdctl snapshot
NAME:
        snapshot - Manages etcd node snapshots

USAGE:
        etcdctl snapshot <subcommand>

API VERSION:
        3.3


COMMANDS:
        save    Stores an etcd node backend snapshot to a given file
        restore Restores an etcd member snapshot to an etcd directory
        status  Gets backend snapshot status of a given file

OPTIONS:
  -h, --help[=false]    help for snapshot

GLOBAL OPTIONS:
      --cacert=""                               verify certificates of TLS-enabled secure servers using this CA bundle
      --cert=""                                 identify secure client using this TLS certificate file
      --command-timeout=5s                      timeout for short running command (excluding dial timeout)
      --debug[=false]                           enable client-side debug logging
      --dial-timeout=2s                         dial timeout for client connections
  -d, --discovery-srv=""                        domain name to query for SRV records describing cluster endpoints
      --endpoints=[127.0.0.1:2379]              gRPC endpoints
      --hex[=false]                             print byte strings as hex encoded strings
      --insecure-discovery[=true]               accept insecure SRV records describing cluster endpoints
      --insecure-skip-tls-verify[=false]        skip server certificate verification
      --insecure-transport[=true]               disable transport security for client connections
      --keepalive-time=2s                       keepalive time for client connections
      --keepalive-timeout=6s                    keepalive timeout for client connections
      --key=""                                  identify secure client using this TLS key file
      --user=""                                 username[:password] for authentication (prompt if password is not supplied)
  -w, --write-out="simple"                      set the output format (fields, json, protobuf, simple, table)
```

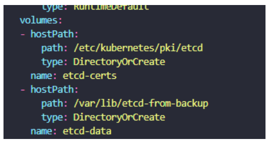

```
controlplane ~ ➜ export ETCDCTL_API=3

controlplane ~ ➜ etcdctl snapshot
```


```
controlplane ~ ➜ etcdctl snapshot save --endpoints=127.0.0.1:2379 \
> --cacert=/etc/kubernetes/pki/etcd/ca.crt \
> --cert=/etc/kubernetes/pki/etcd/server.crt \
> --key=/etc/kubernetes/pki/etcd/server.key \
> /opt/snapshot-pre-boot.db
Snapshot saved at /opt/snapshot-pre-boot.db
```


Luckily we took a backup. Restore the original state of the cluster using the backup file.

● Deployments: 2
● Services: 3


```
controlplane ~ ✗etcdctl snapshot restore --data-dir /var/lib/etcd-from-backup /opt/snapshot-pre-boot.db
2022-11-01 06:54:51.035053 I | mvcc: restore compact to 1575
2022-11-01 06:54:51.047139 I | etcdserver/membership: added member 8e9e05c52164694d [http://localhost:2380] to cluster cdf818194e3a8c32
```


```
      type: RuntimeDefault
  volumes:
  - hostPath:
      path: /etc/kubernetes/pki/etcd
      type: DirectoryOrCreate
    name: etcd-certs
  - hostPath:
      path: /var/lib/etcd-from-backup
      type: DirectoryOrCreate
    name: etcd-data
```

References

https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#backing-up-an-etcd-cluster

https://github.com/etcd-io/website/blob/main/content/en/docs/v3.5/op-guide/recovery.md

https://www.youtube.com/watch?v=qRPNuT080Hk