Backing Up and Restoring Kubernetes Clusters

41m 21s left

Open Cloud Environment
100% Setup completed
Average setup time: 2m 31s

Open Code Environment
100% Setup completed

Credentials

Account ID
45141212[illegible] Copy

Username
student Copy

Password
Ca1_HcGbv Copy

Region
US West 2 Copy

PEM Download PPK Download

Bastion Public Ip
35.92.49. Copy

Cluster SSH
ssh ubunt[illegible]

Introduction

The state information of a Kubernetes cluster is stored in etcd. You can back up a Kubernetes cluster and restore it to an earlier state by using the snapshot and restore functionality of etcd. You will explore this functionality in this lab step. You will use a command-line client named etcdctl to interact with the Kubernetes ectd key-value store. Rather than install etcdctl on the host machine, which is a viable option, you will use pods and containers to perform the backup and restore operations.

Although not a focus of this lab step, in practice you should also back up the cluster's certificate authority key (/etc/kubernetes/pki/ca.key) and certificate (/etc/kubernetes/pki/ca.crt). You should do that after the control-plane is first initialized with `kubeadm init`.

Instructions

1. SSH into the control-plane node:

Copy code

```
# SSH into control-plane node
ssh control-plane -oStrictHostKeyChecking=no
```

2. Create a deployment of the Nginx application with two replicas:

Copy code

```
# Create Nginx application
kubectl create deployment nginx --image=nginx
kubectl scale deployment nginx --replicas=2
```

```
deployment.apps/nginx created
```

Support

The deployment and the following service are simply created so that you can [illegible] ation at the end of

Lab Steps

Need help? Contact our support team

Copy code

```
kubectl expose deployment nginx --type=ClusterIP --port=80 --tar
```

```
service/web exposed
```

4. Send a HTTP request to the web service:

Copy code

```
# Get the Cluster IP of the service
service_ip=$(kubectl get service web -o jsonpath='{.spec.cluster
# Use curl to send an HTTP request to the service
curl $service_ip
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

The Nginx server response verifies that everything is working as expected.

Note: If you receive a failed to connect message, the calico node may still be provisioning.

5. Before proceeding to back up the cluster, ensure all pods are ready:

Copy code

```
kubectl get pods --all-namespaces
```

```
kube-system             calico-node-6ht7w                                                1/1
kube-system             calico-node-bhstl                                                1/1
kube-system             coredns-5dd5756b68-9mffl                                         1/1
kube-system             coredns-5dd5756b68-z589h                                         1/1
kube-system             ebs-csi-controller-6567c45dd7-7cnx6                              5/6
kube-system             ebs-csi-controller-6567c45dd7-8qg67                              6/6
kube-system             ebs-csi-node-5q5pt                                               3/3
kube-system             ebs-csi-node-gr5pm                                               3/3
kube-system             ebs-csi-node-gtn8s                                               3/3
kube-system             etcd-ip-10-0-0-100.us-west-2.compute.internal                    1/1
kube-system             kube-apiserver-ip-10-0-0-100.us-west-2.compute.internal          1/1
kube-system             kube-controller-manager-ip-10-0-0-100.us-west-2.compute.internal 1/1
kube-system             kube-proxy-9n96w                                                 1/1
kube-system             kube-proxy-hmn7j                                                 1/1
kube-system             kube-proxy-w4nmx                                                 1/1
kube-system             kube-scheduler-ip-10-0-0-100.us-west-2.compute.internal          1/1
kube-system             metrics-server-78b5bddcd8-pw49k                                  1/1
kubernetes-dashboard    dashboard-metrics-scraper-fc485cc4c-h9vr4                        1/1
kubernetes-dashboard    kubernetes-dashboard-67d4b797d9-c6dhr                            1/1
```

Note: Please wait for all pods to reach the ready state before moving to the next step.

6. Create a management namespace:

Copy code

```
kubectl create namespace management
```

```
namespace/management created
```

7. Create a job that creates a pod, and issues the `etcdctl snapshot save` command to back up the cluster:

Copy code

```
cat <<EOF | kubectl create -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: backup
  namespace: management
spec:
  template:
    spec:
      containers:
      # Use etcdctl snapshot save to create a snapshot in the /
      - command:
        - /bin/sh
        args:
        - -ec
        - etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --ce
        # The same image used by the etcd pod
        image: registry.k8s.io/etcd:3.5.9-0
        name: etcdctl
        env:
        # Set the etcdctl API version to 3 (to match the versic
        - name: ETCDCTL_API
          value: '3'
        volumeMounts:
        - mountPath: /etc/kubernetes/pki/etcd
          name: etcd-certs
```



```
ort is accessible
```

```
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/control-plane
                operator: Exists
      restartPolicy: OnFailure
      tolerations:
      # tolerate the control-plane's NoSchedule taint to allow
      - effect: NoSchedule
        operator: Exists
      volumes:
      # Volume storing the etcd PKI keys and certificates
      - hostPath:
          path: /etc/kubernetes/pki/etcd
          type: DirectoryOrCreate
        name: etcd-certs
      # A volume to store the backup snapshot
      - hostPath:
          path: /snapshots
          type: DirectoryOrCreate
        name: snapshots
EOF
```

You could also create a CronJob Kubernetes resource instead of a one-off `Job` to periodically perform the backup operation. A `Job` is sufficient for this lab. Read through the `Job` manifest, and use the comments to help understand what it does.

The `etcdctl` command (see `spec.template.spec.containers.args`) requires the certificate authority certificate, a client key, and a client certificate to encrypt the etcd traffic. `kubeadm` configures etcd to listen to HTTPS only as a security best practice. The `snapshot save` command creates a snapshot of the entire key-value store at the given location (`/snapshots/backup.db`).

8. List the contents of the /snapshots directory:

Copy code

```
ls /snapshots
```

```
backup.db
```

The etcd snapshot saved by the pod is present. You will now cause the control-plane to fail and remove the data files of the etcd key-value store to simulate a substantial cluster failure.

9. To ensure the calico nodes can properly restore enter the following command:

Copy code

```
system IP_AUTODETE
```

Copy code

```
sudo systemctl stop kubelet.service
```

The kubelet will automatically try to restart the etcd pod if it detects that it has been deleted. You need to stop the kubelet to prevent this.

11. Delete the etcd data files persisted to disk:

Copy code

```
sudo rm -rf /var/lib/etcd/member
```

The etcd pod mounts /var/lib/etcd to persist its data to disk.

12. Use a container to restore the /var/lib/etcd data from the backup snapshot:

Copy code

```
sudo podman run --rm \
 -v '/snapshots:/snapshots' \
 -v '/var/lib/etcd:/var/lib/etcd' \
 -e ETCDCTL_API=3 \
 'registry.k8s.io/etcd:3.5.9-0' \
 /bin/sh -c "etcdctl snapshot restore --data-dir /var/lib/etcd
/snapshots/backup.db"
```

You need to directly use a container instead of creating a pod in Kubernetes because Kubernetes will not function with the kubelet and etcd offline. The kubelet will recreate the etcd pod from the static pod manifest in /etc/kubernetes/manifests/etcd.yaml. The `etcdctl snapshot restore` command performs the restore operation.

13. Start the kubelet:

Copy code

```
sudo systemctl start kubelet
```

The kubelet automatically recreates the missing etcd pod containers. The pod will use the restored data files created from the backup, and Kubernetes will

Copy code

```
kubectl get pods
```

```
NAME                                READY   STATUS
nginx-8f458dc5b-92j8k               1/1     Running
nginx-8f458dc5b-p8hcg               1/1     Running
```

Note: It may take a minute for the pods to restore. Re-enter the command as needed.

15. Confirm the web service works:

Copy code

```
service_ip=$(kubectl get service web -o jsonpath='{.spec.cluster
curl $service_ip
```

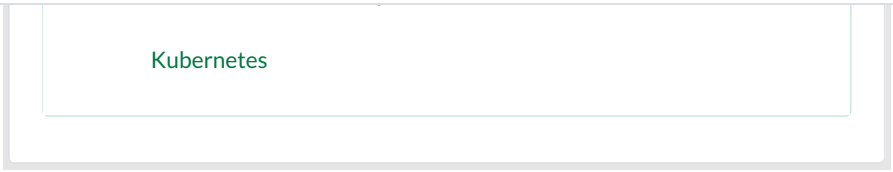
```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Note: It may take a minute for the pods to restore. Re-enter the command as needed.

Summary

In this lab step, you performed backup and restore operations of Kubernetes underlying etcd data store.

VALIDATION CHECKS



Did you like this step?

End Lab Back Next Step