Exploring a Kubernetes Cluster with kubectl

You are working for BeeBox, a company that provides regular shipments of bees to customers. The company is in the process of building a Kubernetes-based infrastructure for some of their software.

You have several work tickets that will need to be addressed by using kubectl to interact with the Kubernetes cluster. You will need to collect some information from the cluster and save that information in some files for later review. You will also need to make some changes to the cluster.

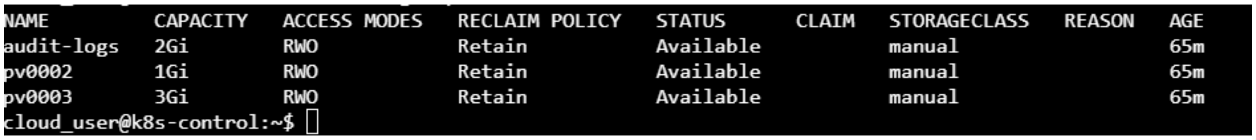
*This lab is already build out for us

ssh cloud_user@<PUBLIC_IP_ADDRESS>

1. First command we would use for the persistent volumes

kubectl get pv

Figure 1-1



```
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
audit-logs   2Gi        RWO            Retain           Available           manual                  65m
pv0002       1Gi        RWO            Retain           Available           manual                  65m
pv0003       3Gi        RWO            Retain           Available           manual                  65m
cloud_user@k8s-control:~$ 
```

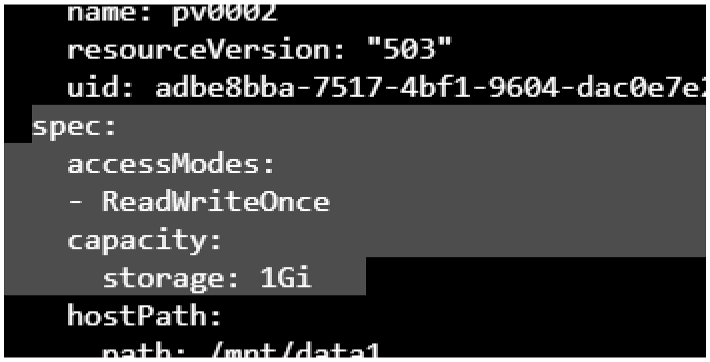
So we have 3 PV, but they are not being sorted by capacity

2. We need to sort by a particular field, so we need to determine the name of the field I want to sort by in yaml format

kubectl get pv -o yaml

Since we are trying to collect information for customers we can start off by looking at the storage. So look for capacity.

Figure 1-2



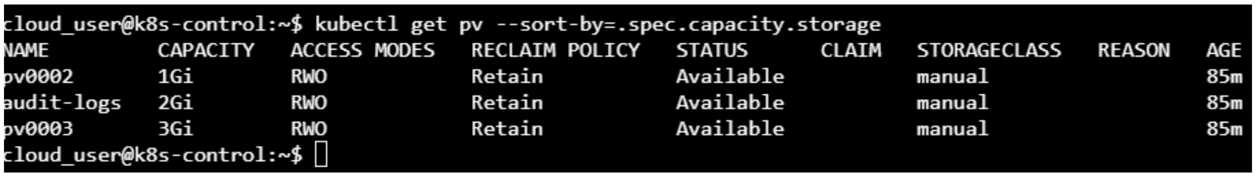
```
    name: pv0002
    resourceVersion: "503"
    uid: adbe8bba-7517-4bf1-9604-dac0e7e
  spec:
    accessModes:
    - ReadWriteOnce
    capacity:
      storage: 1Gi
    hostPath:
      path: /mnt/data1
```

This is the field we want to sort by

3. To sort a field with the capacity listed above

kubectl get pv --sort-by=.spec.capacity.storage

Figure 1-3



```
cloud_user@k8s-control:~$ kubectl get pv --sort-by=.spec.capacity.storage
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv0002       1Gi        RWO            Retain           Available           manual                  85m
audit-logs   2Gi        RWO            Retain           Available           manual                  85m
pv0003       3Gi        RWO            Retain           Available           manual                  85m
cloud_user@k8s-control:~$
```

You can see now it is being sorted by the lowest capacity by the largest

4. So we need save this output to a file. So we have to rerun that command and redirect it to a txt file.

kubectl get pv --sort-by=.spec.capacity.storage > /home/cloud_user/pv_list.txt

5. To verify that the file has been created you would use the cat command

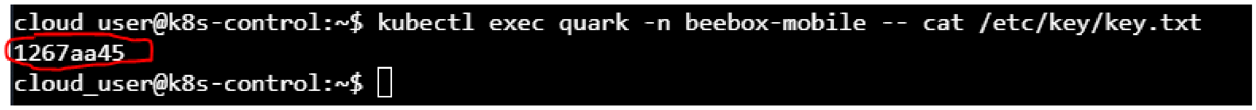
cat pv_list.txt

Our next objective is to run a command inside the cork pod and retrieve the data that's inside a file inside that pod's file system and save it to another file on the host file system

6. To go in the cork pod and retrieve data that is inside the file.

kubectl exec quark -n beebox-mobile -- cat /etc/key/key.txt

Figure 1-4



```
cloud_user@k8s-control:~$ kubectl exec quark -n beebox-mobile -- cat /etc/key/key.txt
1267aa45
cloud_user@k8s-control:~$
```

Once we get that output, we need to store it in the appropriate file on the host.

7. To store an output to a file

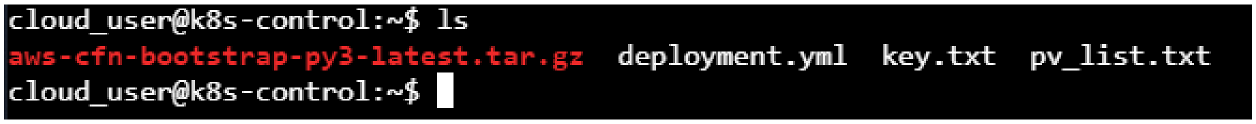
kubectl exec quark -n beebox-mobile -- cat /etc/key/key.txt > /home/cloud_user/key.txt

8. Once that is completed verify by using the cat command

cat key.txt

Next task we need to complete is to create a deployment using this deployment.yml spec file that already exist here on the server. (see below)

Figure 1-5



```
cloud_user@k8s-control:~$ ls
aws-cfn-bootstrap-py3-latest.tar.gz  deployment.yml  key.txt  pv_list.txt
cloud_user@k8s-control:~$
```

9. To create a deployment using the deployment spec found using the kubectl apply -f command

kubectl apply -f /home/cloud_user/deployment.yml

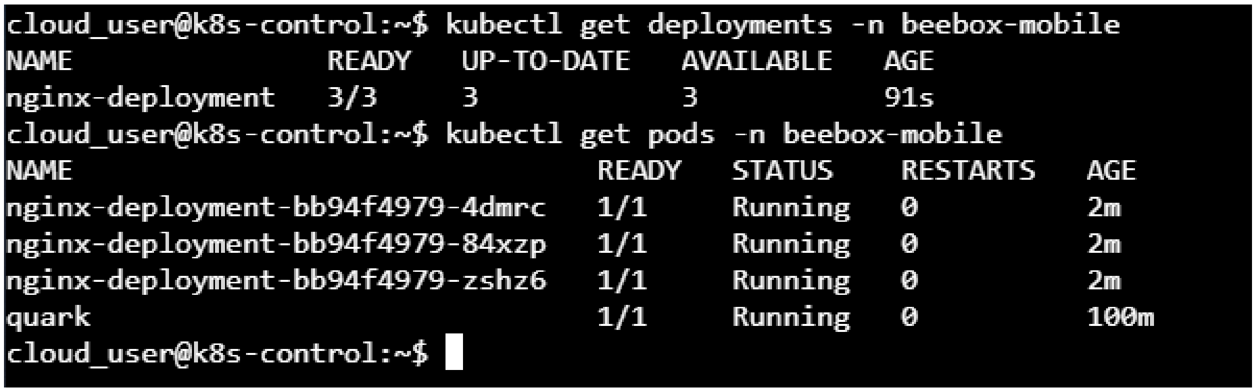
10. We can view this deployment if we look into the right namespace

kubectl get deployments -n beebox-mobile

11. Looking into the pods to see if they are running too

kubectl get pods -n beebox-mobile

Figure 1-6



```
cloud_user@k8s-control:~$ kubectl get deployments -n beebox-mobile
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           91s
cloud_user@k8s-control:~$ kubectl get pods -n beebox-mobile
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-bb94f4979-4dmrc   1/1     Running   0          2m
nginx-deployment-bb94f4979-84xzp   1/1     Running   0          2m
nginx-deployment-bb94f4979-zshz6   1/1     Running   0          2m
quark                              1/1     Running   0          100m
cloud_user@k8s-control:~$
```

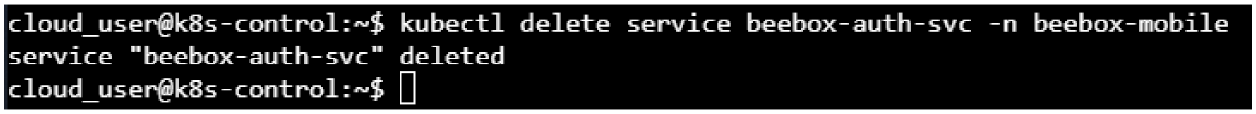
So we have successfully created that deployment

Our final task is to delete a service called beebox-auth-svc.

12. To delete the service named above

kubectl delete service beebox-auth-svc -n beebox-mobile

Figure 1-7



```
cloud_user@k8s-control:~$ kubectl delete service beebox-auth-svc -n beebox-mobile
service "beebox-auth-svc" deleted
cloud_user@k8s-control:~$
```

We have completed the lab