Exercise 6.1: RESTful API Access

Overview

We will continue to explore ways of accessing the control plane of our cluster. In the security chapter we will discuss there are several authentication methods, one of which is use of a `Bearer token` We will work with one then deploy a local proxy server for application-level access to the Kubernetes API.

We will use the **curl** command to make API requests to the cluster, in an insecure manner. Once we know the IP address and port, then the token we can retrieve cluster data in a RESTful manner. By default most of the information is restricted, but changes to authentication policy could allow more access.

1. First we need to know the IP and port of a node running a replica of the API server. The cp system will typically have one running. Use **kubectl config view** to get overall cluster configuration, and find the server entry. This will give us both the IP and the port.

```
student@cp:~$ kubectl config view
```

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://k8scp:6443
  name: kubernetes
<output_omitted>
```

2. Next we need to find the bearer token. This is part of a default token. Look at a list of tokens, first all on the cluster, then just those in the default namespace. There will be a `secret` for each of the controllers of the cluster.

```
student@cp:~$ kubectl get secrets --all-namespaces
```

```
NAMESPACE        NAME                   TYPE                                 ...
default          default-token-jdqp7    kubernetes.io/service-account-token...
kube-node-lease  default-token-j67mt    kubernetes.io/service-account-token...
kube-public      default-token-b2prn    kubernetes.io/service-account-token...
kube-system      attachdetach-controller-token-ckwvh kubernetes.io/servic...
kube-system      bootstrap-signer-token-wpx66 kubernetes.io/service-accou...
<output_omitted>
```

```
student@cp:~$ kubectl get secrets
```

```
NAME                 TYPE                                  DATA   AGE
default-token-jdqp7  kubernetes.io/service-account-token   3      23h
```

3. Look at the details of the secret. We will need the `token:` information from the output.

```
student@cp:~$ kubectl describe secret default-token-jdqp7
```

```
Name:         default-token-jdqp7
Namespace:    default
Labels:       <none>
<output_omitted>
token:      eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVz
L3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3Bh
```

```
Y2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubm
<output_omitted>
```

4. Using your mouse to cut and paste, or **cut**, or **awk** to save the data, from the first character `eyJh` to the last, to a variable named `token`. Your token data will be different. Also note the caret is a regex anchor, which may not copy and paste from a PDF properly and need to be replaced by hand.

```
student@cp:˜$ export token=$(kubectl describe \
   secret default-token-jdqp7 |grep ^token |cut -f7 -d ' ')
```

5. Test to see if you can get basic API information from your cluster. We will pass it the server name and port, the token and use the **-k** option to avoid using a cert.

```
student@cp:˜$ curl https://k8scp:6443/apis --header "Authorization: Bearer $token" -k
```

```
{
  "kind": "APIGroupList",
  "apiVersion": "v1",
  "groups": [
    {
      "name": "apiregistration.k8s.io",
      "versions": [
        {
          "groupVersion": "apiregistration.k8s.io/v1",
          "version": "v1"
<output_omitted>
```

6. Try the same command, but look at API v1. Note that the path has changed to `api`.

```
student@cp:˜$ curl https://k8scp:6443/api/v1 --header "Authorization: Bearer $token" -k
```

```
<output_omitted>
```

7. Now try to get a list of namespaces. This should return an error. It shows our request is being seen as `system`serviceaccount:, which does not have the `RBAC` authorization to list all namespaces in the cluster.

```
student@cp:˜$ curl \
     https://k8scp:6443/api/v1/namespaces --header "Authorization: Bearer $token" -k
```

```
<output_omitted>
  "message": "namespaces is forbidden: User \"system:serviceaccount:default...
<output_omitted>
```

8. Pods can also make use of included certificates to use the API. The certificates are automatically made available to a pod under the `/var/run/secrets/kubernetes.io/serviceaccount/`. We will deploy a simple Pod and view the resources. If you view the `token` file you will find it is the same value we put into the `$token` variable. The **-i** will request a **-t** terminal session of the `busybox` container. Once you exit the container will not restart and the pod will show as completed.

```
student@cp:˜$ kubectl run -i -t busybox --image=busybox --restart=Never
```

Inside container

```
# ls /var/run/secrets/kubernetes.io/serviceaccount/
ca.crt namespace token
# exit
```

9. Clean up by deleting the `busybox` container.

```
student@cp:~$ kubectl delete pod busybox
```

```
pod "busybox" deleted
```