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. The creation of token secrets by Kubernetes is no longer automatic in recent releases. It then falls on the user to design them as desired. Use the command shown below to create the token.

```
student@cp:~$ export token=$(kubectl create token default)
```

3. 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>
```

4. 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>
```

5. Now try to get a list of namespaces. This should return an error. It shows our request is being seen as systemserviceaccount:, 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>
```

6. 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
```

7. Clean up by deleting the busybox container.

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

```
  pod "busybox" deleted
```