Lab 3: Kubernetes API

The Kubernetes API is the central management interface and the front-end for the Kubernetes control plane. It exposes a RESTful interface that is consumed by all components within the cluster, as well as by external users and tools like `kubectl`, to communicate with the cluster.

Every operation you perform in Kubernetes, from deploying applications and managing services to scaling resources and querying the cluster's state, is ultimately executed through API calls. This makes the API server the heart of any Kubernetes cluster, acting as the single point of entry for all cluster operations.

Understanding how to interact with the Kubernetes API, both directly and indirectly, is crucial for effective Kubernetes administration. It provides the foundation for managing, troubleshooting, and observing your cluster's behavior and resources.

Perform the tasks below as student user on your master node:

Exploring API Calls

01

Use strace to see what kubectl is doing

Notice the several openat functions referencing the local .kube/cache directory.

```
strace kubectl get endpoints
```

02

Go to the .kube/cache/discovery directory and explore the files there

```
cd /home/student/.kube/cache/discovery/
cd master_6443
ls
tree
```

03

Display the apps/v1/serverresources.json

Take note of the shortnames.

```
cat apps/v1/serverresources.json | jq
```

04

Check resources using the kubectl command

Use -o json|yaml to see more details.

```
kubectl api-resources
```

RESTful API Access

There are several authentication methods. We will be using "Bearer token" here, and deploy local proxy for application-level access to the Kubernetes API.

1

Generate token for the default serviceaccount

```
export token=$(kubectl create token default)
```

2

List the apis group

We will use insecure access to avoid using a cert.

```
curl https://master:6443/apis --header "Authorization: Bearer $token" -k
```

3

Try again but try using API v1

This will result in failure with a forbidden message. This shows that the default serviceaccount doesn't have RBAC authorization to list all namespaces.

```
curl https://master:6443/api/v1/namespaces --header "Authorization: Bearer $token" -k
```

Using Proxy

Another way to interact with Kubernetes API is via a proxy which can be done from a node or pod sidecar.

Step 1: Learn how to use kubectl proxy

```
kubectl proxy -h
```

Step 2: Run the command to proxy all of the Kubernetes API in the background

```
kubectl proxy --api-prefix=/ &
[1] 5020
Starting to serve on 127.0.0.1:8001
```

Step 3: Test the api by using the proxy

```
curl http://127.0.0.1:8001/api/
curl http://127.0.0.1:8001/api/v1/namespaces
```

What failed previously works now. This is used to troubleshoot problems by narrowing down the problem to authentication and authorization issues.

Step 4: Stop the proxy

```
kill %1
```

Using TLS Access

Display the current config using kubectl

The --raw option displays hidden contents. The contents are exactly the same as your .kube/config file.

```
kubectl config view
```

Extract the three certificates and get the API URL

```
grep auth .kube/config | awk '{print $2}' | base64 -d > ca.pem
grep client-cert .kube/config | awk '{print $2}' | base64 -d > cert.pem
grep key .kube/config | awk '{print $2}' | base64 -d > key.pem
kubectl config view | grep server
```

Use a secure connection to retrieve the list of pods from the cluster

```
curl --cert cert.pem --key key.pem --cacert ca.pem master:6443 https://master:6443/api/v1/pods
```

Create a pod in the default namespace by using the 3s-web.json file

```
curl --cert cert.pem --key key.pem --cacert ca.pem master:6443
https://master:6443/api/v1/namespaces/default/pods -XPOST -H'Content-Type: application/json' -
d@3s-web.json
```

Check if the pod is created

```
kubectl get pods
```

Cleanup

```
kubectl delete po/web
kubectl get po
```

Lab Complete! You have successfully explored the Kubernetes API using multiple access methods including strace, RESTful API with Bearer tokens, proxy, and TLS certificates.