Exercise 5.2: Explore API Calls

1. One way to view what a command does on your behalf is to use **strace**. In this case, we will look for the current endpoints, or targets of our API calls. Install the tool, if not present.

```
student@cp:~$ sudo apt-get install -y strace
```

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

```
NAME         ENDPOINTS          AGE
kubernetes   10.128.0.3:6443    3h
```

2. Run this command again, preceded by **strace**. You will get a lot of output. Near the end you will note several **openat** functions to a local directory, /home/student/.kube/cache/discovery/k8scp_6443. If you cannot find the lines, you may want to redirect all output to a file and grep for them. This information is cached, so you may see some differences should you run the command multiple times. As well your IP address may be different.

```
student@cp:~$ strace kubectl get endpoints
```

```
execve("/usr/bin/kubectl", ["kubectl", "get", "endpoints"], [/*....
....
openat(AT_FDCWD, "/home/student/.kube/cache/discovery/k8scp_6443..
<output_omitted>
```

3. Change to the parent directory and explore. Your endpoint IP will be different, so replace the following with one suited to your system.

```
student@cp:~$ cd /home/student/.kube/cache/discovery/
```

```
student@cp:~/.kube/cache/discovery$ ls
```

```
k8scp_6443
```

```
student@cp:~/.kube/cache/discovery$ cd k8scp_6443/
```

4. View the contents. You will find there are directories with various configuration information for kubernetes.

```
student@cp:~/.kube/cache/discovery/k8scp_6443$ ls
```

```
admissionregistration.k8s.io  certificates.k8s.io           node.k8s.io
apiextensions.k8s.io          coordination.k8s.io           policy
apiregistration.k8s.io        crd.projectcalico.org         rbac.authorization.k8s.io
apps                          discovery.k8s.io              scheduling.k8s.io
authentication.k8s.io         events.k8s.io                 servergroups.json
authorization.k8s.io          extensions                    storage.k8s.io
autoscaling                   flowcontrol.apiserver.k8s.io  v1
batch                         networking.k8s.io
```

5. Use the find command to list out the subfiles. The prompt has been modified to look better on this page.

```
student@cp:./k8scp_6443$ find .
```

```
.
./storage.k8s.io
./storage.k8s.io/v1beta1
./storage.k8s.io/v1beta1/serverresources.json
./storage.k8s.io/v1
```

```
./storage.k8s.io/v1/serverresources.json
./rbac.authorization.k8s.io
<output_omitted>
```

6. View the objects available in version 1 of the API. For each object, or kind:, you can view the verbs or actions for that object, such as create seen in the following example. Note the prompt has been truncated for the command to fit on one line. Some are HTTP verbs, such as `GET`, others are product specific options, not standard HTTP verbs. The command may be **python**, depending on what version is installed.

```
student@cp:.$ python3 -m json.tool v1/serverresources.json
```

JSON **serverresources.json**

```
{
    "apiVersion": "v1",
    "groupVersion": "v1",
    "kind": "APIResourceList",
    "resources": [
        {
            "kind": "Binding",
            "name": "bindings",
            "namespaced": true,
            "singularName": "",
            "verbs": [
                "create"
            ]
        },
<output_omitted>
```

7. Some of the objects have `shortNames`, which makes using them on the command line much easier. Locate the `shortName` for endpoints.

```
student@cp:.$ python3 -m json.tool v1/serverresources.json | less
```

JSON **serverresources.json**

```
....
{
"kind": "Endpoints",
"name": "endpoints",
"namespaced": true,
"shortNames": [
      "ep"
],
"singularName": "",
"verbs": [
   "create",
   "delete",
....
```

8. Use the `shortName` to view the endpoints. It should match the output from the previous command.

```
student@cp:.$ kubectl get ep
```

```
NAME         ENDPOINTS          AGE
kubernetes   10.128.0.3:6443    3h
```

9. We can see there are 37 objects in version 1 file.

```
student@cp:.$ python3 -m json.tool v1/serverresources.json | grep kind
```

```
      "kind": "APIResourceList",
             "kind": "Binding",
             "kind": "ComponentStatus",
             "kind": "ConfigMap",
             "kind": "Endpoints",
             "kind": "Event",
   <output_omitted>
```

10. Looking at another file we find nine more.

```
student@cp:$ python3 -m json.tool apps/v1/serverresources.json | grep kind
```

```
      "kind": "APIResourceList",
             "kind": "ControllerRevision",
             "kind": "DaemonSet",
             "kind": "DaemonSet",
             "kind": "Deployment",
   <output_omitted>
```

11. Delete the `curlpod` to recoup system resources.

```
student@cp:$ kubectl delete po curlpod
```

```
   pod "curlpod" deleted
```

12. Take a look around the other files in this directory as time permits.