Passing Configuration Data to a Kubernetes Container

You are working for BeeBox, a company that provides regular shipments of bees to customers. The company is working on deploying some applications to a Kubernetes cluster.

One of these applications is a simple Nginx web server. This server is used as part of a secure backend application, and the company would like it to be configured to use HTTP basic authentication.

This will require an htpasswd file as well as a custom Nginx config file. In order to deploy this Nginx server to the cluster with good configuration practices, you will need to load the custom Nginx configuration from a ConfigMap (this already exists) and use a Secret to store the htpasswd data.

Create a Pod with a container running the nginx:1.19.1 image. Supply a custom Nginx configuration using a ConfigMap, and populate an htpasswd file using a Secret.

htpasswd is already installed on the server, and you can generate an htpasswd file like so:

htpasswd -c .htpasswd user

A pod called busybox already exists in the cluster, which you can use to contact your Nginx pod and test your setup.

Generate an htpasswd File and Store It as a Secret

Our first objective is the generate an htpassword file and store it as a Kubernetes secret

1. Setting up htpassword as Kubernetes secret

htpasswd -c .htpasswd cawingo

^ after typing this, it will require you to set a password

2. Verify what is in the htpassword

ls -la

Figure 1-1

```
cloud_user@k8s-control:~$ ls -la
total 628
drwxr-xr-x 7 cloud_user cloud_user   4096 Nov 13 21:10 .
drwxr-xr-x 4 root       root         4096 Jul  7  2020 ..
drwx------ 3 cloud_user cloud_user   4096 Feb  1  2022 .ansible
-rw-r--r-- 1 cloud_user cloud_user     53 May 24 20:15 .bash_history
-rw-r--r-- 1 cloud_user cloud_user   3106 Jul 10  2020 .bashrc
drwx------ 2 cloud_user cloud_user   4096 Nov 13 21:06 .cache
drwx------ 3 cloud_user cloud_user   4096 Dec 16  2021 .config
-rw-rw-r-- 1 cloud_user cloud_user     43 Nov 13 21:11 .htpasswd
drwxr-xr-x 2 root       root         4096 Nov 13 20:04 .kube
-rw-r--r-- 1 cloud_user cloud_user    161 Jul 10  2020 .profile
drwx------ 2 cloud_user cloud_user   4096 Nov 13 20:03 .ssh
-rw-r--r-- 1 cloud_user cloud_user      0 Aug 20  2020 .sudo_as_admin_successful
-rw------- 1 cloud_user cloud_user   1163 May 24 19:30 .viminfo
-rw-r--r-- 1 root       root       591853 Jan 10  2022 aws-cfn-bootstrap-py3-latest.tar.gz
cloud_user@k8s-control:~$
```

3. To view what is inside that htpassword file

```
cat .htpasswd
```

4. Create a secret containing the htpasswd data

```
kubectl create secret generic nginx-htpasswd --from-file .htpasswd
```

5. So since we went ahead and created that secret password. Lets remove the .htpasswd file for security reasons since it is in plain text.

```
rm .htpasswd
```

Create the Nginx Pod

6. Now we are ready to create our pod

```
vi pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.19.1
    ports:
    - containerPort: 80
    volumeMounts:
    - name: config-volume
      mountPath: /etc/nginx
    - name: htpasswd-volume
      mountPath: /etc/nginx/conf
```

```
 volumes:
 - name: config-volume
   configMap:
     name: nginx-config
 - name: htpasswd-volume
   secret:
     secretName: nginx-htpasswd
```

^To copy and paste
Set vi to 'paste' mode by hitting :, and then typing set paste (ENTER). Then switch back to INSERT mode by hitting i.

Figure 1-2

```
piVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.19.1
    ports:
    - containerPort: 80
    volumeMounts:
    - name: config-volume
      mountPath: /etc/nginx
    - name: htpasswd-volume
      mountPath: /etc/nginx/conf
  volumes:
  - name: config-volume
    configMap:
      name: nginx-config
  - name: htpasswd-volume
    secret:
      secretName: nginx-htpasswd
```

7. Once that is complete, lets go ahead and look at the ConfigMap before we create the pod with the apply command

```
kubectl get cm
```

Figure 1-4

```
cloud_user@k8s-control:~$ kubectl get cm
NAME               DATA   AGE
kube-root-ca.crt   1      16m
nginx-config       1      15m
cloud_user@k8s-control:~$
```

^ Above you can see the nginx-config

8. We then want to describe the ConfigMap on that nginx config

kubectl describe cm nginx-config

^ This command is just for showing the data inside the CM. And since we mounted our ConfigMap as a volume mount, inside that mouth path directory. There is going to be a file in our container called nginx.conf (Just basic nginx configuration)

So essentially we're going to be loading an nginx config file into our container using this process of mounting the ConfigMap as a volume mount in our pod definition.

9. So now that we looked that the day and so on. We now have to apply the pod.yml file we just did in step 6

kubectl apply -f pod.yml

When back into the .yml file and double checked some stuff and in return got this error below (Figure 1-5)

Figure 1-5

```
cloud_user@k8s-control:~$ kubectl apply -f pod.yml
error: error validating "pod.yml": error validating data: apiVersion not set; if you choose to
cloud_user@k8s-control:~$ ^C
```

10. Looking back at the "apiVersion" in the .yml file I forgot to input the letter **a** in api. So it was created now
11. So now we check the status of the pod. That way we can see the status and the IP address

kubectl get pods -o wide

Figure 1-6

```
cloud_user@k8s-control:~$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE     IP               NODE          NOMINATED NODE   READINESS GATES
busybox   1/1     Running   0          49m     192.168.194.65   k8s-worker1   <none>           <none>
nginx     1/1     Running   0          3m14s   192.168.194.67   k8s-worker1   <none>           <none>
```

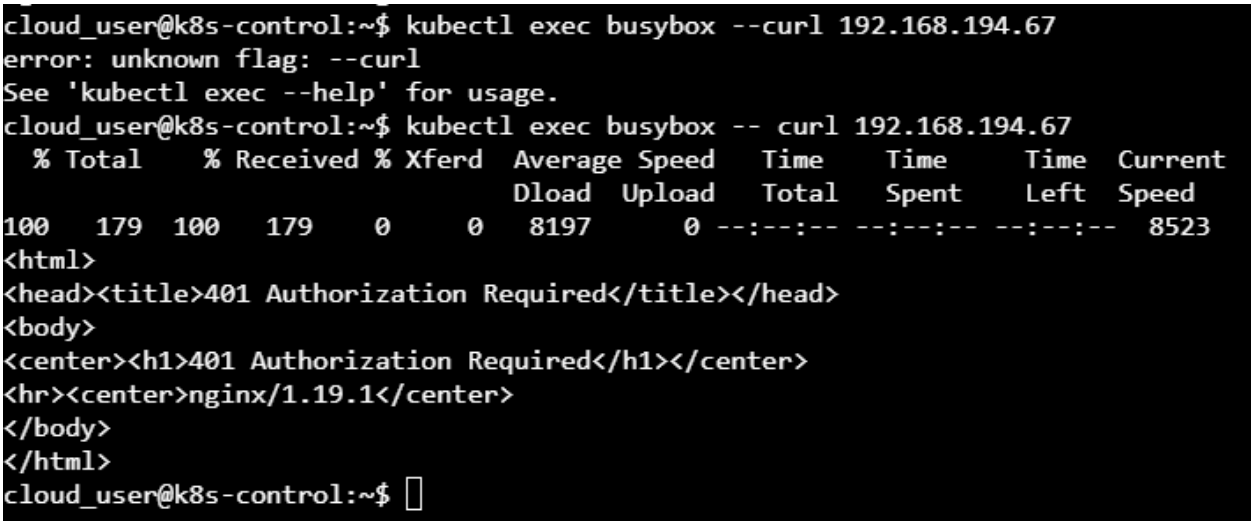
So it looks like it is running but we still want to verify that. So we need to actually make a request to this nginx.

Since we have another pod existing in this, we can use this pod to reach out to the nginx pod and test.

12. To test one pod to another using the IP address

kubectl exec busybox – curl 192.168.194.67 (The nginx IP address)

Figure 1-7



^ That is good that we are getting that 401 Authorization Required. Because the .htpasswd and nginx config setup is working as it should because we didn't supply the username and password

13. Test one pod to another using the IP address with authenticating

kubectl exec busybox -- curl -u cawingo:test 192.168.194.67

^ the cawingo:test is the user associated with the htpasswd we did earlier in step 1

14. Received a bash error saying bash: (password I inputted: event not found

Figure 1-8

```
cloud_user@k8s-control:~$ kubectl exec busybox -- curl -u cawingo:test 192.168.194.67
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   291k      0 --:--:-- --:--:-- --:--:--  597k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
cloud_user@k8s-control:~$
```

So we have completed the task by creating a Pod with a container running the nginx:1.19.1 image. Supply a custom Nginx configuration using a ConfigMap, and populate an htpasswd file using a Secret.