



Exercise 6.2: Create and consume Secrets

Secrets are consumed in a manner similar to ConfigMaps, covered in an earlier lab. While at-rest encryption is just now enabled, historically a secret was just base64 encoded. There are three types of encryption which can be configured.

1. Begin by generating an encoded password.

```
student@ckad-1:~/app2$ echo LFTr@1n | base64
TEZUckAxbgo=
```

2. Create a YAML file for the object with an API object kind set to Secret. Use the encoded key as a password parameter.

```
student@ckad-1:~/app2$ vim secret.yaml
```

YAML

secret.yaml

```
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: lfsecret
5 data:
6   password: TEZUckAxbgo=
```

3. Ingest the new object into the cluster.

```
student@ckad-1:~/app2$ kubectl create -f secret.yaml
secret/lfsecret created
```

4. Edit secondapp YAML file to use the secret as a volume mounted under `/mysqlpassword`. `volumeMounts`: lines up with the container name: and `volumes`: lines up with `containers`: Note the pod will restart when the sleep command finishes every 3600 seconds, or every hour.

```
student@ckad-1:~/app2$ vim second.yaml
```

YAML

second.yaml

```
1 ....
2     runAsUser: 2000
3     allowPrivilegeEscalation: false
4     capabilities:
5       add: ["NET_ADMIN", "SYS_TIME"]
6     volumeMounts:                                     #<-- Add this and six following lines
7       - name: mysql
8         mountPath: /mysqlpassword
9     volumes:
10    - name: mysql
11      secret:
12        secretName: lfsecret
```

```
student@ckad-1:~/app2$ kubectl delete pod secondapp
pod "secondapp" deleted

student@ckad-1:~/app2$ kubectl create -f second.yaml
pod/secondapp created
```

5. Verify the pod is running, then check if the password is mounted where expected. We will find that the password is available in its clear-text, decoded state.

```
student@ckad-1:~/app2$ kubectl get pod secondapp

NAME        READY   STATUS    RESTARTS   AGE
secondapp   1/1     Running   0           34s

student@ckad-1:~/app2$ kubectl exec -ti secondapp -- /bin/sh
```



On Container

```
/ $ cat /mysqlpassword/password
LFTTr@1n
```

6. View the location of the directory. Note it is a symbolic link to `../data` which is also a symbolic link to another directory. After taking a look at the filesystem within the container, exit back to the node.



On Container

```
/ $ cd /mysqlpassword/

/mysqlpassword $ ls
password

/mysqlpassword $ ls -al
total 4
drwxrwxrwt  3 root    root      100 Apr 11 07:24 .
drwxr-xr-x 21 root    root      4096 Apr 11 22:30 ..
drwxr-xr-x  2 root    root        60 Apr 11 07:24 ..4984_11_04_07_24_47.831222818
lrwxrwxrwx  1 root    root        31 Apr 11 07:24 ..data -> ..4984_11_04_07_24_47.831222818
lrwxrwxrwx  1 root    root        15 Apr 11 07:24 password -> ..data/password

/mysqlpassword $ exit
```