Secrets and K8s

Task 1

1. Create and verify a secret.

1.1 Create a secret with the name `mysecret` with the key `password` and value `mypassword`.

```
kubectl create secret generic mysecret --from-literal=username=myusername --
from-literal=password=mypassword
```

```
PS C:\Users\wezza> kubectl create secret generic mysecret --from-literal=username=myusername --from-literal=password=myp
assword
secret/mysecret created
PS C:\Users\wezza>
```

1.2 Get the secret and decode the password.

```
kubectl get secret mysecret -o json
```


```
developer@xp /m/d/I/3/2/d/S/k8s> kubectl get secret mysecret -o json
{
    "apiVersion": "v1",
    "data": {
        "password": "bXlwYXNzd29yZA==",
        "username": "bXl1c2VybmFtZQ=="
    },
    "kind": "Secret",
    "metadata": {
        "creationTimestamp": "2024-04-17T06:10:30Z",
        "name": "mysecret",
        "namespace": "default",
        "resourceVersion": "5342",
        "uid": "baf55c88-905a-4e4b-b808-337bc131b76a"
    },
    "type": "Opaque"
}
```


1.3 Describe the secret.

```
kubectl describe secret mysecret
```

```
developer@xp /m/d/I/3/2/d/S/k8s> kubectl get secret mysecret -o json
{
    "apiVersion": "v1",
    "data": {
        "password": "bXlwYXNzd29yZA==",
        "username": "bXl1c2VybmFtZQ=="
    },
    "kind": "Secret",
    "metadata": {
        "creationTimestamp": "2024-04-17T06:10:30Z",
        "name": "mysecret",
        "namespace": "default",
        "resourceVersion": "5342",
        "uid": "baf55c88-905a-4e4b-b808-337bc131b76a"
    },
    "type": "Opaque"
}
```

1.4 Decode the secret.

- The secret is encoded in base64, so we need to decode it.

For Linux:

```
kubectl get secret mysecret -o jsonpath='{.data.password}' | base64 --decode
```

```
developer@xp /m/d/I/3/2/d/S/k8s> kubectl get secret mysecret -o jsonpath='{.data.password}' | base64 -
-decode
mypassword
```

2. Manage secrets with Helm.

2.1 Install Helm secrets plugin.

```
helm plugin install https://github.com/jkroepke/helm-secrets
helm secrets patch windows
helm secrets upgrade name . -f secrets.yaml
```

2.2 Create a `secrets.yaml` file in the `templates` folder.

```
apiVersion: v1
kind: Secret
metadata:
  name: credentials
  labels:
    app: python-app-helm
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
type: Opaque
```

```
  data:
    password: { { .Values.password | quote } }
```

2.3 Create a main `secrets.yaml`

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: bXlwYXNzd29yZA==
```

2.4 Encrypt the secret.

2.4.1 Create a key pair.

```
gpg --full-generate-key
```

2.4.2 Encrypt the secret using the public key.

```
sudo sops -p "C3BAA339FB44D6F215874015D514EC137B0012F4" -e secrets-raw.yaml >
secrets.yaml

rm secrets-raw.yaml
```

2.5 Deploy the Helm chart with the encrypted secret

```
helm secrets install app-python-helm ./app-python-helm/ -n default -f
./secrets.yaml
```

2.6 Verify the secret inside the pod.

```
kubectl exec app-python-helm-0 -- printenv | grep MY_PASSWORD
```

```
MY_PASSWORD=mypassword
```

Task 2

1. Install Vault Using Helm Chart:

```
helm repo add hashicorp https://helm.releases.hashicorp.com
helm repo update
helm install vault hashicorp/vault --set "server.dev.enabled=true"
```

```
developer@xp:/mnt/d/Innopolis/3/2/devops/S24-core-course-labs/k8s$ helm install vault hashicorp/vault
--set "server.dev.enabled=true"
NAME: vault
LAST DEPLOYED: Wed Apr 17 10:04:31 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Thank you for installing HashiCorp Vault!

Now that you have deployed Vault, you should look over the docs on using
Vault with Kubernetes available here:

https://developer.hashicorp.com/vault/docs


Your release is named vault. To learn more about the release, try:

  $ helm status vault
  $ helm get manifest vault
```

2. Follow the Tutorial with Your Helm Chart:

2.1 Access the Vault pod.

```
kubectl exec -it vault-0 -- /bin/sh
```

2.2 Get the pods

```
kubectl get pod
```

```
developer@xp:/mnt/d/Innopolis/3/2/devops/S24-core-course-labs/k8s$ kubectl get pod
NAME                                    READY   STATUS              RESTARTS      AGE
post-install-sleep-job-qdjfn            0/1     Completed           0             74m
pre-install-sleep-job-hfpln             0/1     Completed           0             74m
python-helm-helm-python-59c4d5fdb6-4r24f   1/1     Running             1 (98s ago)   90m
vault-0                                 0/1     ContainerCreating   0             62s
vault-agent-injector-dbfc5cd77-hnkjp    0/1     ContainerCreating   0             63s
developer@xp:/mnt/d/Innopolis/3/2/devops/S24-core-course-labs/k8s$
```

2.3 Set Secret in Vault.

```
>$ kubectl exec -it vault-0 -- /bin/sh
/ $ vault secrets enable -path=internal kv-v2
Success! Enabled the kv-v2 secrets engine at: internal/
/ $ vault kv put internal/server/config password="mypassword"
======= Secret Path =======
internal/data/server/config

======= Metadata =======
Key                Value
---                -----
created_time       2024-04-17T09:55:05.726565855Z
custom_metadata    <nil>
deletion_time      n/a
```

```
destroyed          false
version            1
/ $ vault kv get internal/server/config
======= Secret Path =======
internal/data/server/config

======= Metadata =======
Key                Value
---                -----
created_time       2024-04-17T09:55:05.726565855Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
password    mypassword
/ $ vault auth enable kubernetes
Success! Enabled kubernetes auth method at: kubernetes/
/ $ vault write auth/kubernetes/config \
>       kubernetes_host="https://$KUBERNETES_PORT_443_TCP_ADDR:443"
Success! Data written to: auth/kubernetes/config
/ $ vault policy write internal-app - <<EOF
> path "internal/data/database/config" {
>    capabilities = ["read"]
> }
> EOF
Success! Uploaded policy: internal-app
/ $ vault write auth/kubernetes/role/internal-app \
>       bound_service_account_names=internal-app \
>       bound_service_account_namespaces=default \
>       policies=internal-app \
>       ttl=24h
Success! Data written to: auth/kubernetes/role/internal-app
/ $ exit
```

2.4 Check the secret in the pod.

```
>$ kubectl exec -it python-helm-helm-python-59c4d5fdb6-4r24f -- /bin/sh
/app cat /vault/secrets/config.txt
data: map[password:mypassword]
metadata: map[created_time:2024-04-17T09:55:05.726565855Z custom_metadata:<nil>
deletion_time: destroyed:false version:1]

/app df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          55G   50G  2.0G  97% /
tmpfs            64M     0   64M   0% /dev
tmpfs           3.8G     0  3.8G   0% /sys/fs/cgroup
```

```
tmpfs           7.5G  4.0K  7.5G   1% /vault/secrets
/dev/nvme0n1p5   55G   50G  2.0G  97% /etc/hosts
shm              64M     0   64M   0% /dev/shm
tmpfs           7.5G   12K  7.5G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs           3.8G     0  3.8G   0% /proc/asound
tmpfs           3.8G     0  3.8G   0% /proc/acpi
tmpfs           3.8G     0  3.8G   0% /proc/scsi
tmpfs           3.8G     0  3.8G   0% /sys/firmware
```

Bouns Task

```
>$ kubectl exec python-helm-helm-python-59c4d5fdb6-4r24f -- printenv | grep -e
'RELEASE_NAME' -e 'MY_PASS' -e 'SLEEP_TIME'

RELEASE_NAME=app-python-helm
SLEEP_TIME=3
MY_PASSWORD=mypassword
```

```
>$ kubectl exec javascript-helm-helm-javascript-cr8bdc844-s62p5 -- printenv |
grep -e 'RELEASE_NAME' -e 'IMAGE_TAG'

RELEASE_NAME=app-javascript-helm
IMAGE_TAG=latest
```