Namespace: secure
https://kubernetes.io/docs/concepts/configuration/secret/
https://kubernetes.io/docs/concepts/workloads/pods/
https://kubernetes.io/docs/concepts/storage/volumes/

Creating a Secret of type kubernetes.io/basic-auth
Consuming the Secret in a Pod as Volume
Verifying the correct runtime behavior

Creating a Secret
Create a Secret of type kubernetes.io/basic-auth named api-basic-auth in the namespace secure.

Provide the key-value pairs username=serviceapp and password=j6PcbDtdhnizmaa.

The Secret object can be created using either the imperative or declarative approach.

Imperative Approach
To create the Secret with an imperative command, use the --type CLI option:

$ kubectl create secret generic api-basic-auth --type=kubernetes.io/basic-auth
--from-literal=username=serviceapp --from-literal=password=j6PcbDtdhnizmaa -n secure
secret/api-basic-auth created

kubernetes.io/basic-auth is a type of Secret in Kubernetes that is specifically designed for storing basic authentication credentials. It is commonly used to store username and password combinations that can be used for authenticating with external systems or services.

When creating a Secret of type kubernetes.io/basic-auth, you provide the username and password as key-value pairs using the --from-literal flag. The Secret is then created with these credentials securely encoded.

The kubectl create secret command you provided creates a Kubernetes Secret named api-basic-auth in the secure namespace. This Secret is of type kubernetes.io/basic-auth and contains two key-value pairs:

username with the value serviceapp
password with the value j6PcbDtdhnizmaa
This Secret can be used to store and manage basic authentication credentials for your application within Kubernetes.

Declarative Approach
To create the Secret declaratively, start by defining a YAML manifest in the file secret.yaml:

```
apiVersion: v1
kind: Secret
metadata:
  name: api-basic-auth
  namespace: secure
type: kubernetes.io/basic-auth
stringData:
  username: serviceapp
  password: j6PcbDtdhnizmaa
```

Execute the following command to create the Secret:

```
$ kubectl apply -f secret.yaml
secret/api-basic-auth created
```

```
root@controlplane:~$
root@controlplane:~$ kubectl create namespace secure
namespace/secure created
root@controlplane:~$ ^[[200~kubectl create secret generic api-basic-auth --type=kub
dhnizmaa -n secure~^C
root@controlplane:~$ kubectl create secret generic api-basic-auth --type=kubernetes
a -n secure
secret/api-basic-auth created
root@controlplane:~$ kubectl get secret api-basic-auth -o yaml -n secure
apiVersion: v1
data:
  password: ajZQY2JEdGRobml6bWFh
  username: c2VydmljZWFwcA==
kind: Secret
metadata:
  creationTimestamp: "2023-06-20T08:13:00Z"
  name: api-basic-auth
  namespace: secure
  resourceVersion: "1153"
  uid: c030443f-2c66-45aa-ae7f-5dc21860ce45
type: kubernetes.io/basic-auth
```

Inspecting the Secret Data

Upon creation, the Secret automatically base64-encoded the values. Rendering the YAML representation of the live object reveals the behavior:

```
$ kubectl get secret api-basic-auth -o yaml -n secure
```

Consuming the Secret

Create a Pod named server-app with the image nginx:1.23.3 in the namespace secure and consume the Secret as Volume at the mount path /var/data.

Open an interactive shell to the Pod and find the relevant Secret files mounted by the containe

Create the YAML manifest of the Pod in the namespace with the following imperative command:

$ kubectl run server-app --image=nginx:1.23.3 --restart=Never -n secure --dry-run=client -o yaml > pod.yaml

Edit the pod.yaml file with vim or nano and add the Volume and the mount path to the container.

```
pod/server-app created
root@controlplane:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: server-app
  name: server-app
  namespace: secure
spec:
  containers:
  - image: nginx:1.23.3
    name: nginx
    volumeMounts:
    - name: auth-volume
      mountPath: /var/data
      readOnly: true
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
  volumes:
  - name: auth-volume
    secret:
      secretName: api-basic-auth
status: {}
root@controlplane:~$ kubectl get pod
No resources found in default namespace.
root@controlplane:~$ kubectl get pod -n secure
NAME         READY   STATUS    RESTARTS   AGE
server-app   1/1     Running   0          52s
root@controlplane:~$
```

The YAML manifest you provided describes a Kubernetes Pod named server-app in the secure namespace. The Pod is configured to run an Nginx container using the nginx:1.23.3 image.

Here's a breakdown of the Pod manifest:

Metadata:

The Pod has labels with the key-value pair run: server-app.
The name of the Pod is server-app.
The Pod is created in the secure namespace.
Spec:

Containers:

The Pod has one container named nginx using the nginx:1.23.3 image.
The container has a volume mount specified:
The mount is named auth-volume.
The mount path is set to /var/data within the container.
The mount is read-only.
The container has no specified resource limits.
DNS Policy:

The DNS policy is set to ClusterFirst, which means the Pod will use the cluster's DNS for DNS resolution.
Restart Policy:

The restart policy is set to Never, indicating that the Pod will not be restarted automatically if it fails or terminates.
Volumes:

The Pod has one volume specified:
The volume is named auth-volume.
The volume is of type secret.
The secret referenced is api-basic-auth, which is the Secret you created earlier.
Status:

The status field is empty and will be populated with information about the Pod's status once it is created and running.
This Pod mounts the api-basic-auth Secret as a read-only volume at the path /var/data within the Nginx container. You can use this volume to provide the basic authentication credentials (username and password) to the Nginx container for authentication purposes.

Wait until the Pod transitions into the Running status. Verify that the files for the Secret keys username and password became available at the mount path /var/data and contain the correct value:

```
root@controlplane:~$ kubectl exec server-app -n secure -- cat /var/data
cat: /var/data: Is a directory
command terminated with exit code 1
root@controlplane:~$ kubectl exec server-app -n secure -- cat /var/data/username
serviceapproot@controlplane:~$ kubectl exec server-app -n secure -- cat /var/data/password
j6PcbDtdhnizmaaroot@controlplane:~$
```