Web-MySQL Application

app.py

```python
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                            user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```


```python
mysql.connector.connect(host='mysql', database='mysql',
                        user='root', password='paswrd')
```

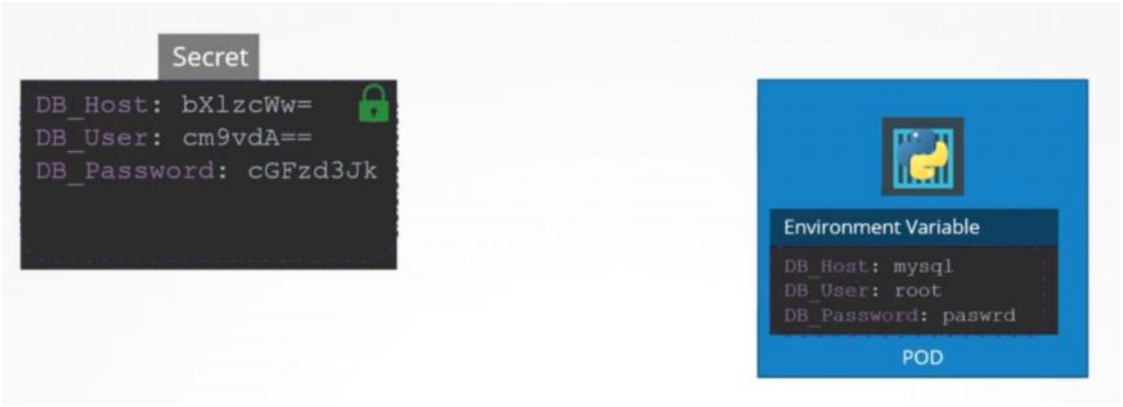
config-map.yaml

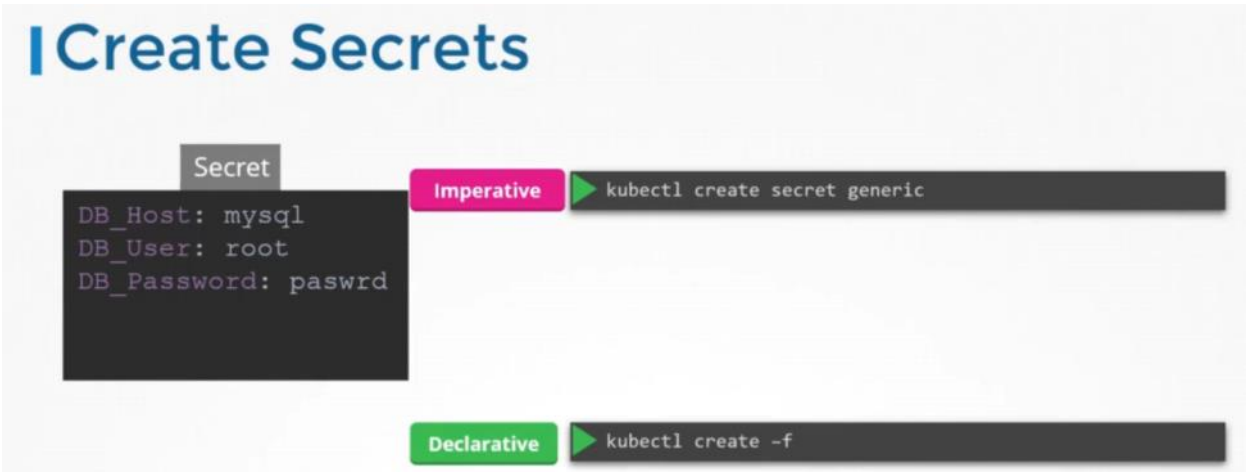
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswrd
```

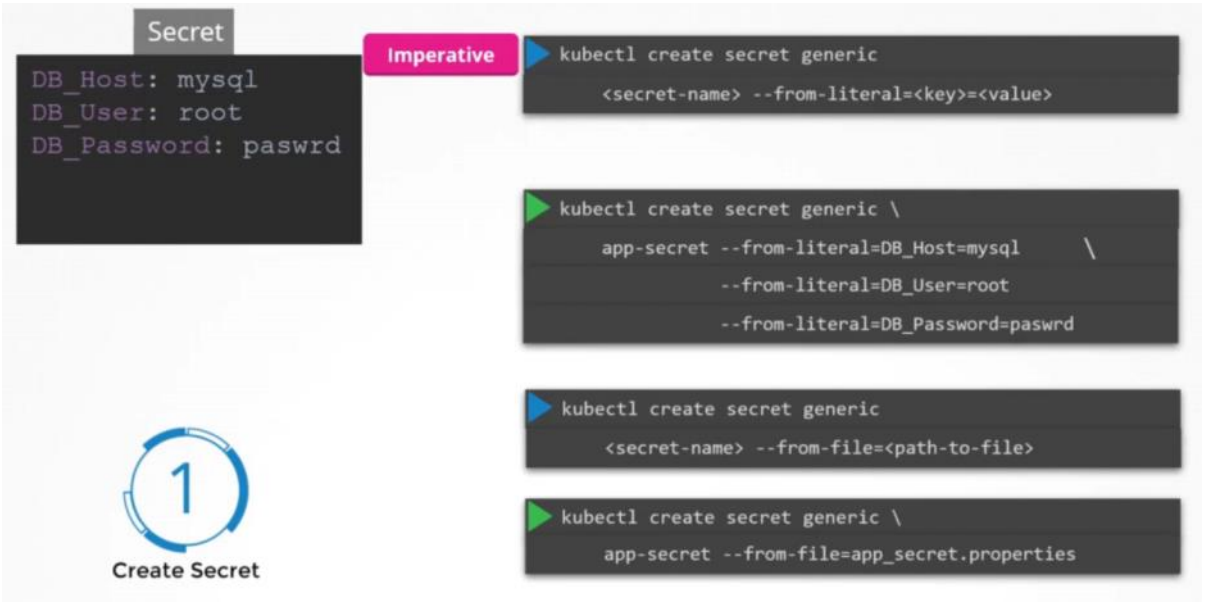
Secret

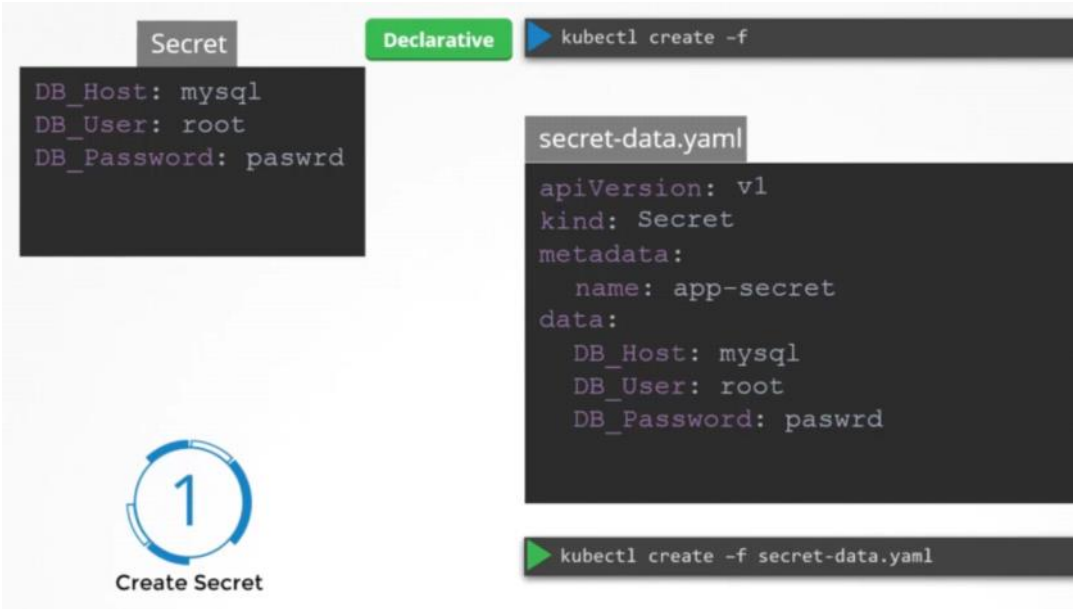
Secret

```yaml
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```


Secret
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
Environment Variable
DB_Host: mysql
DB_User: root
DB_Password: paswrd
POD


Create Secrets
Secret
DB_Host: mysql
DB_User: root
DB_Password: paswrd
Imperative
kubectl create secret generic
Declarative
kubectl create -f

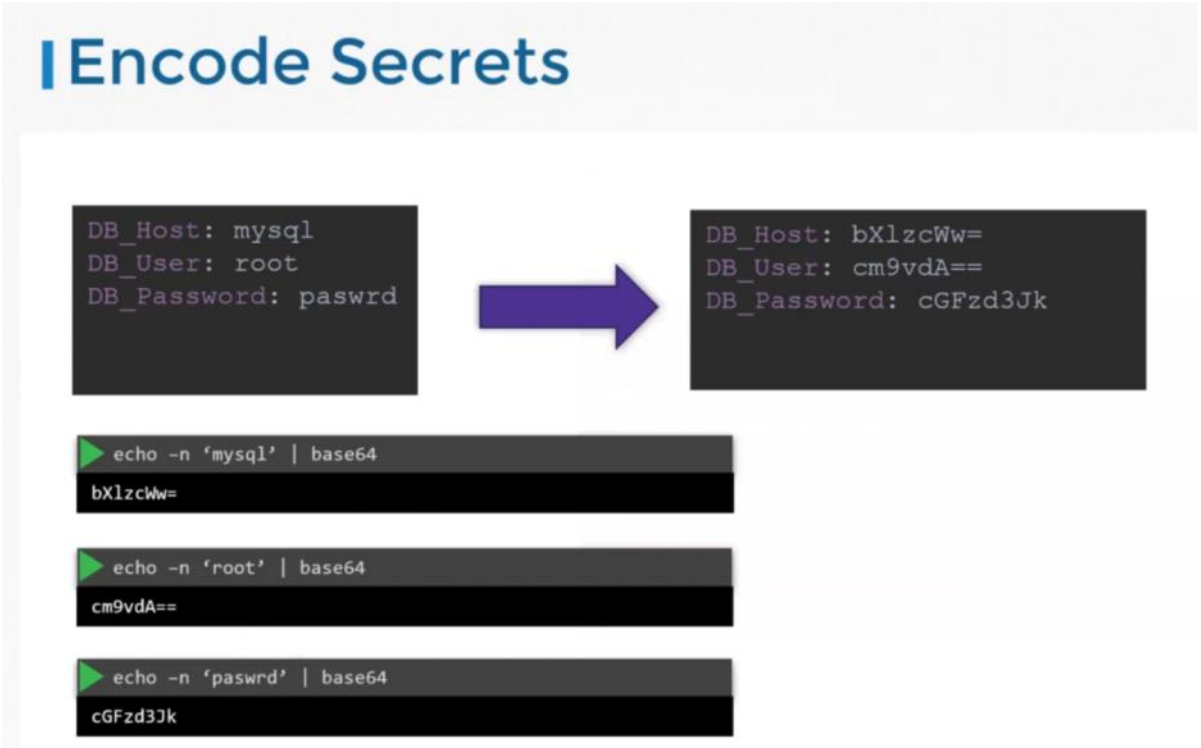

Secret
DB_Host: mysql
DB_User: root
DB_Password: paswrd
Imperative
kubectl create secret generic
<secret-name> --from-literal=<key>=<value>
kubectl create secret generic \
app-secret --from-literal=DB_Host=mysql \
--from-literal=DB_User=root
--from-literal=DB_Password=paswrd
kubectl create secret generic
<secret-name> --from-file=<path-to-file>
kubectl create secret generic \
app-secret --from-file=app_secret.properties
1
Create Secret



Not SAFE

```
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswrd
```

```
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```



Decode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'bXlzcWw=' | base64 --decode
mysql
```

```
echo -n 'cm9vdA==' | base64 --decode
root
```

```
echo -n 'cGFzd3Jk' | base64 --decode
paswrd
```

View Secrets

```
kubectl get secrets
NAME                  TYPE                                  DATA      AGE
app-secret            Opaque                                3         10m
default-token-mvtkv   kubernetes.io/service-account-token   3         2h
```

```
kubectl describe secrets
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      10 bytes
DB_Password:  6 bytes
DB_User:      4 bytes
```

```
kubectl get secret app-secret -o yaml
apiVersion: v1
data:
  DB_Host: bXlzcWw=
  DB_Password: cGFzd3Jk
  DB_User: cm9vdA==
kind: Secret
metadata:
  creationTimestamp: 2018-10-18T10:01:12Z
  labels:
    name: app-secret
  name: app-secret
  namespace: default
uid: be96e989-d2bc-11e8-a545-080027931072
type: Opaque
```

Secrets in Pods

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - secretRef:
          name: app-secret
```

secret-data.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```


```
kubectl create -f pod-definition.yaml
```

Secrets in Pods

```yaml
envFrom:
  - secretRef:
      name: app-config
```

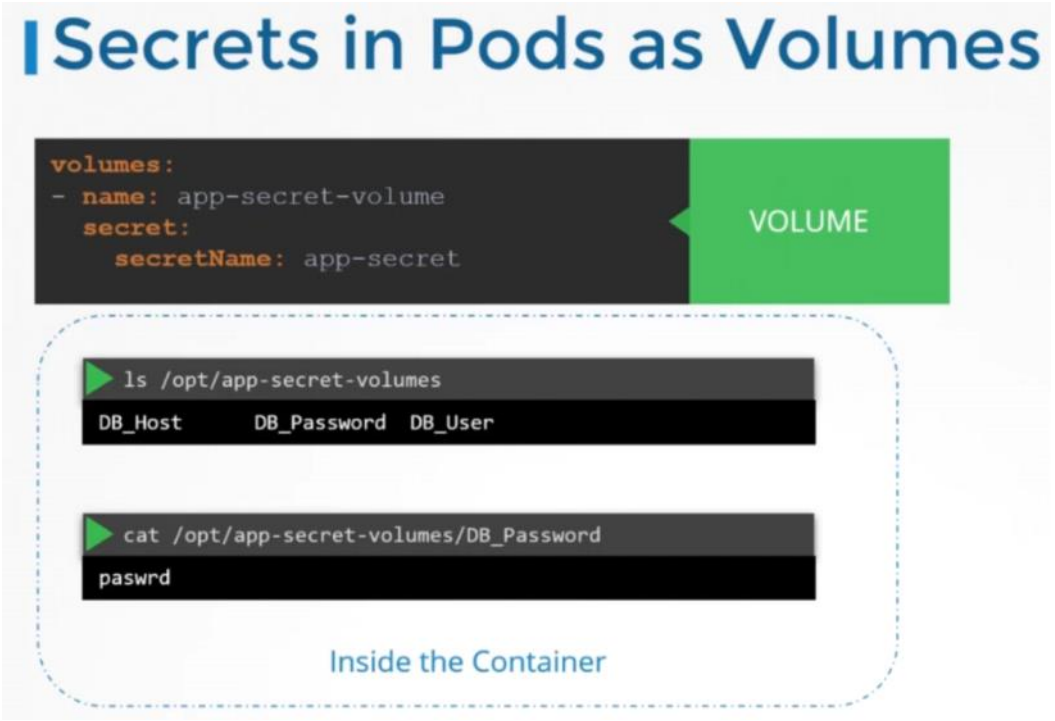
ENV

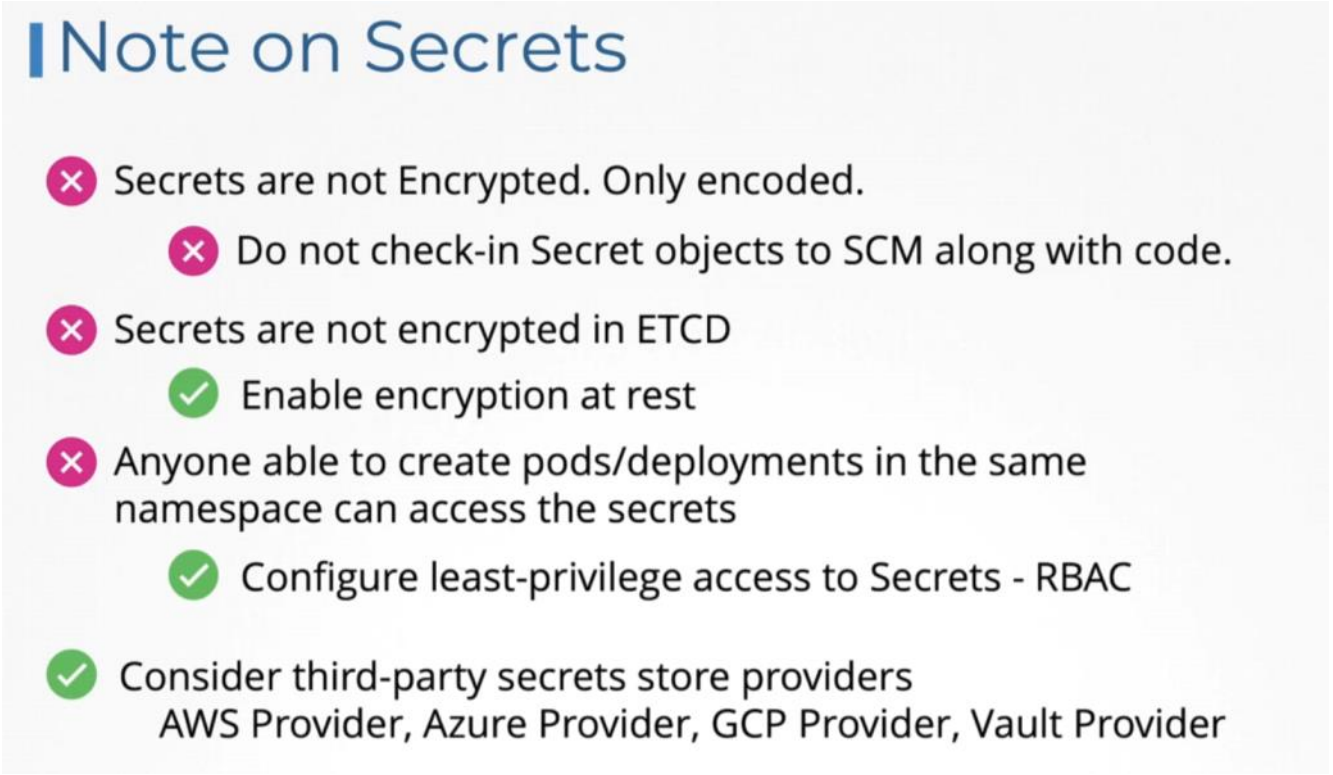
SINGLE ENV

```yaml
env:
  - name: DB_Password
    valueFrom:
      secretKeyRef:
        name: app-secret
        key: DB_Password
```

```yaml
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME





A note about Secrets!
Remember that secrets encode data in base64 format. Anyone with the base64 encoded secret can easily decode it. As such the secrets can be considered as not very safe.
The concept of safety of the Secrets is a bit confusing in Kubernetes.
The kubernetes documentation page and a lot of blogs out there refer to

secrets as a "safer option" to store sensitive data. They are safer than storing in plain text as they reduce the risk of accidentally exposing passwords and other sensitive data. In my opinion it's not the secret itself that is safe, it is the practices around it.
Secrets are not encrypted, so it is not safer in that sense. However, some best practices around using secrets make it safer. As in best practices like:

- Not checking-in secret object definition files to source code repositories.
- Enabling Encryption at Rest for Secrets so they are stored encrypted in ETCD.

Also the way kubernetes handles secrets. Such as:

- A secret is only sent to a node if a pod on that node requires it.
- Kubelet stores the secret into a tmpfs so that the secret is not written to disk storage.
- Once the Pod that depends on the secret is deleted, kubelet will delete its local copy of the secret data as well.

Read about the protections and risks of using secrets here

Having said that, there are other better ways of handling sensitive data like passwords in Kubernetes, such as using tools like Helm Secrets, HashiCorp Vault. I hope to make a lecture on these in the future.