ThoughtWorks®

CONFIGMAPS & SECRETS

Rise of the Containers Workshop

Are we really going to hardcode configuration in our app?

What are our options?

Config

- Command Line Arguments
- Properties file
- Environment variables

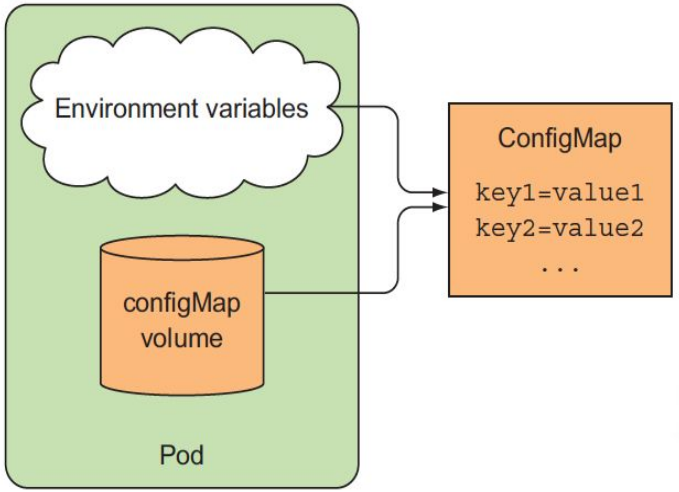
```
kind: Pod
spec:
 containers:
 - image: luksa/fortune:env
   env:
   - name: INTERVAL
     value: "30"
   name: html-generator
...

...
```

Adding a single variable to the environment variable list

Config Maps

- Represents the configuration of your application
- Stores key-value pairs
- Can be created from directories, files and literals
- Can be injected as environment variables
- Can be mounted as volumes
- Like the container's command and arguments, the list of environment variables also **cannot be updated** after the pod is created.



Creating config maps

Config Name

Config Source Type

```
$ kubectl create configmap special-config --from-literal=name=JamesBond
$ kubectl create configmap special-config --from-file=name=./name.txt
```

Config Value

Let's create a config map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
data:
  name: Jon Snow
```

Spawn a pod that uses the config

```
kind: Pod
apiVersion: v1
metadata:
  name: nameapp
  labels:
     type: nameapp
spec:
  containers:
   - name: nameapp
     image:
ankitaluthra1/k8s-config-sample:1.0
     ports:
     - containerPort: 8080
     env:
     - name: NAME
       valueFrom:
         configMapKeyRef:
           name: special-config
           key: name
```

Config Maps

```
metadata:
  name: fortune-env-from-configmap
spec:
  containers:
  - image: luksa/fortune:env
    env:
    - name: INTERVAL
      valueFrom:
        configMapKeyRef:
          name: fortune-config
          key: sleep-interval
...
```

You're setting the environment variable called INTERVAL.

Instead of setting a fixed value, you're initializing it from a ConfigMap key.

The name of the ConfigMap you're referencing

You're setting the variable to whatever is stored under this key in the ConfigMap.

Expose the pod

```
$ kubectl expose pod nameapp \
--port=8080 \
--type=NodePort
```

Exercise time!

Use a env mongo URL in metadata service

P.S. Configure metadata service to use env variable as follows:

```
spring.data.mongodb.uri=${MONGO_URL}
```

ThoughtWorks®

Example: Mount the config

```
kind: Pod
apiVersion: v1
metadata:
  name: nameapp
  labels:
     type: nameapp
spec:
  volumes:
  -  name: config-volume
     configMap:
       name: special-config
  containers:
  -  name: nameapp
     image:
ankitaluthra1/k8s-config-sample:1.0
     ports:
     - containerPort: 8080
     volumeMounts:
     - name: config-volume
       mountPath: /etc/config
```

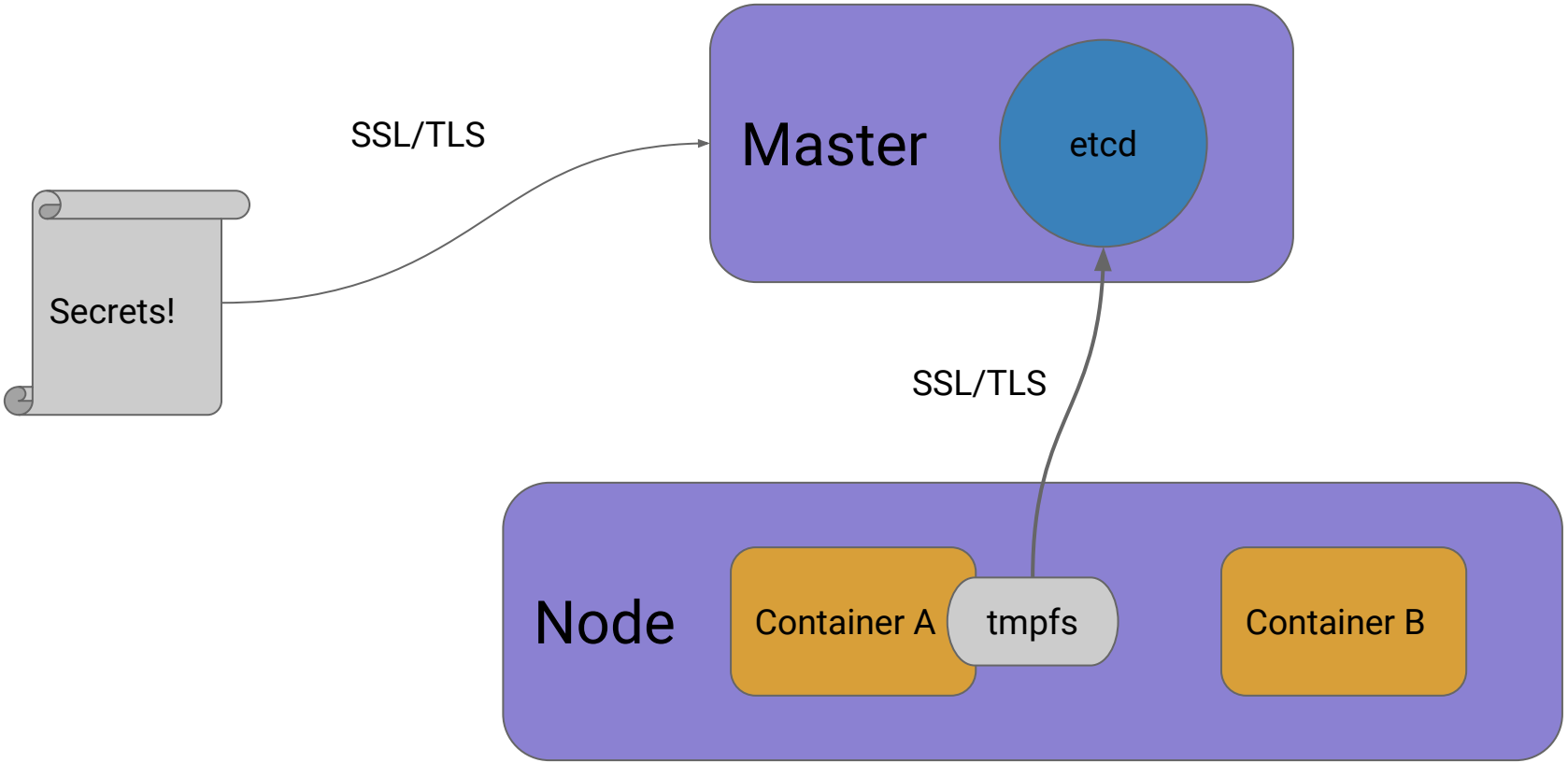
Is it OK to use passwords and keys in config maps? If not, how then?

K8s Secrets

- Store private and sensitive information
- Uses encryption at rest (v1.7 onwards)
- (Very) Similar to config map
- Can be mounted as volume or as env variables

```
kind: Secret
```

How Secrets Work



Creating secrets

Secret Name

Secret Source Type

Base64 encoded

```
$ kubectl create secret generic mysecret --from-literal=password=Ym9uZAo=
```

```
$ kubectl create secret generic mysecret --from-file=password=./password.txt
```

Secret Value

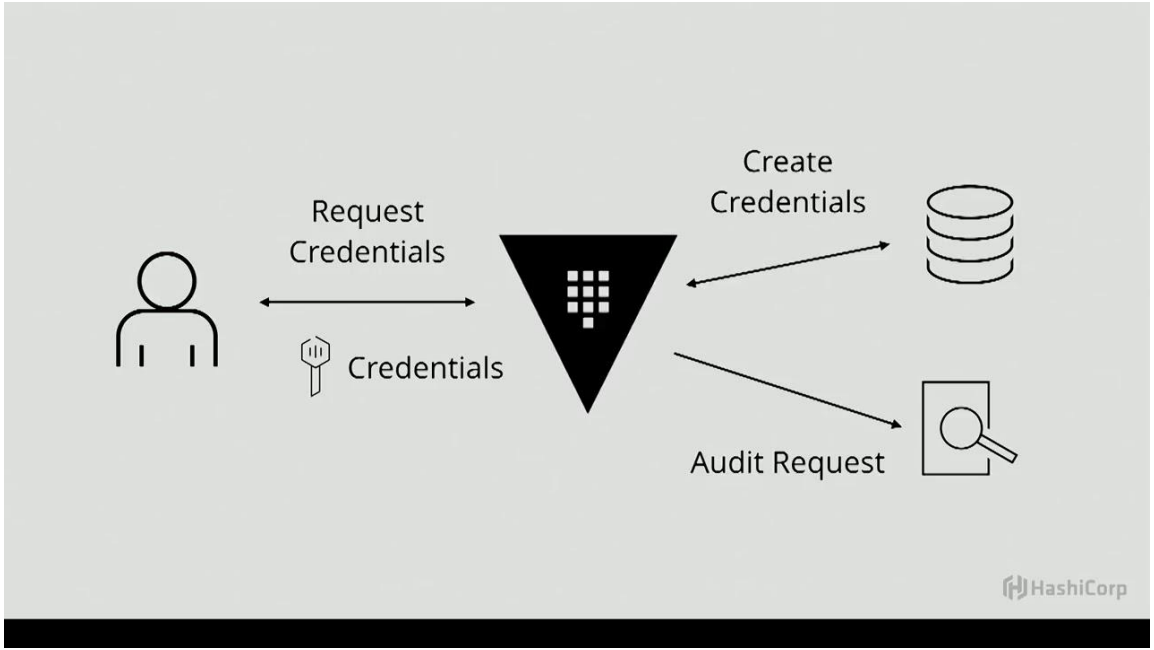
```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
Type: Opaque
data:
  password: Ym9uZAo=
```

*Access **Secrets** from a Pod*

```
kind: Pod
apiVersion: v1
metadata:
  name: nameapp
  labels:
     type: nameapp
spec:
  containers:
   - name: nameapp
     image:
ankitaluthra1/k8s-config-sample:1.0
     ports:
     - containerPort: 3000
     env:
     - name: NAME
       valueFrom:
         secretKeyRef:
           name: mysecret
           key: password
```

Is this enough?

Dynamic Secrets



Exercise time!

Use a secret mongo URL in metadata service

P.S. Configure metadata service to use env variable as follows:

```
spring.data.mongodb.uri=${MONGO_URL}
```

ThoughtWorks®