III. Config with ConfigMaps and Secrets

Relevant Documentation

- Twelve-Factor App: III. Config
- ConfigMaps
- Secrets

Lesson Reference

Switch to the source code directory:

```
cd ~/content-designing-applications-for-kubernetes
```

Optional: If you want to start with a clean slate, check out the start tag for this lesson. However, you could lose any local changes you have made if you do this.

```
git reset --hard HEAD

git checkout III-config-start
```

Note: You will need a Docker hub account to follow along with this lesson. Create an account and take note of your Docker Hub username and password.

Now we're ready to get our app running in the cluster.

Change the server code so configuration data can be passed in:

```
vi src/server/index.js
```

Locate the following lines:

```
// Setup MongoDb backing database
const MongoClient = require('mongodb').MongoClient
// MongoDB credentials
const username = encodeURIComponent("uloe_user");
const password = encodeURIComponent("ILoveTheList");
// MongoDB connection info
const mongoPort = 27017;
const mongoHost = 'localhost';
// MongoDB connection string
const mongoURI = `mongodb://${username}:${password}@localhost:27017/uloe`;
const mongoURISanitized = `mongodb://${username}:****@${mongoHost}:${mongoPort}/uloe`;
console.log("MongoDB connection string %s", mongoURISanitized);
```

Change them so the username, password, hostname, and port can be optionally passed in using environment variables:

```
// Setup MongoDb backing database
const MongoClient = require('mongodb').MongoClient
// MongoDB credentials
const username = encodeURIComponent(process.env.MONGODB_USER || "uloe_user");
const password = encodeURIComponent(process.env.MONGODB_PASSWORD || "ILoveTheList");
// MongoDB connection info
const mongoPort = process.env.MONGODB_PORT || 27017;
const mongoHost = process.env.MONGODB_HOST || 'localhost';
// MongoDB connection string
const mongoURI = `mongodb://${username}:${password}@${mongoHost}:${mongoPort}/uloe`;
const mongoURISanitized = `mongodb://${username}:****@${mongoHost}:${mongoPort}/uloe`;
console.log("MongoDB connection string %s", mongoURISanitized);
```

Rebuild the Docker image. Note that you should include your Docker Hub username in the image tag so you can push it to Docker Hub:

```
docker build -t <YOUR_DOCKER_HUB_USERNAME>/uloe-server:0.0.1 --target server .
```

Log in to your Docker Hub account from the server. Supply your Docker Hub username and password when prompted:

```
docker login
```

Push the image to Docker Hub:

```
docker push <YOUR_DOCKER_HUB_USERNAME>/uloe-server:0.0.1
```

Create a production Namespace:

```
kubectl create namespace production
```

Get base64-encoded strings for the MongoDB username and password:

```
echo -n uloe_user | base64

echo -n ILoveTheList | base64
```

Create a ConfigMap and Secret to configure the backing service connection information for the app:

```
vi uloe-config.yml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: uloe-config
data:
  mongodb.host: "uloe-mongodb"
  mongodb.port: "27017"

---

apiVersion: v1
kind: Secret
metadata:
  name: uloe-secure-config
type: Opaque
data:
  mongodb.username: dWxvZV91c2Vy
  mongodb.password: SUxvdmVUaGVMaXN0
```

```
kubectl apply -f uloe-config.yml -n production
```

Create a temporary Pod to test the configuration setup:

```
vi test-pod.yml
```

Note that you need to supply your Docker Hub username as part of the image name in this file:

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - name: uloe-server
    image: <YOUR_DOCKER_HUB_USERNAME>/uloe-server:0.0.1
    ports:
    - name: web
      containerPort: 3001
      protocol: TCP
    env:
    - name: MONGODB_HOST
      valueFrom:
        configMapKeyRef:
          name: uloe-config
          key: mongodb.host
    - name: MONGODB_PORT
      valueFrom:
        configMapKeyRef:
          name: uloe-config
          key: mongodb.port
    - name: MONGODB_USER
      valueFrom:
        secretKeyRef:
          name: uloe-secure-config
          key: mongodb.username
    - name: MONGODB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: uloe-secure-config
          key: mongodb.password
```

```
kubectl apply -f test-pod.yml -n production
```

Check the logs to verify the config data is being passed to the container:

```
kubectl logs test-pod -n production
```

Clean up the test pod:

```
kubectl delete pod test-pod -n production --force
```