9: Dynamic Application Configuration¶

9.1: Lab Goals¶

In this lab, you will be updating your gowebapp container image to accept configuration at runtime. We will create a configmap for configuration and a secret for the mysql password. The deployment will be updated to use the new image, configmaps and secrets instead of having them preconfigured in the docker image.

Let's get your lab environment setup by automatically performing necessary steps from previous labs in this course. We've automated it for you with the following command:

```
eval "$BOOTSTRAP_COMMAND"
```

9.2: Build new Docker image for your frontend application¶

Step 1: Update Dockerfile for your frontend application¶

```
cd $HOME/gowebapp/gowebapp
```

Update the Dockerfile in this directory for the frontend Go application. Use vi or any preferred text editor. The template below provides a starting point for updating the contents of this file.

Note

Replace TODO comments with the appropriate commands.

Remove the line where the configuration file is copied to the image.

Dockerfile-gowebapp

```
FROM ubuntu:impish

# TODO --- add an environment variable declaration for a default DB_PASSWORD
# of "mydefaultpassword"
# https://docs.docker.com/engine/reference/builder/#env

COPY ./code /opt/gowebapp
# TODO --- remove the following line. We no longer want to include the
configuration
# file in the image.
COPY ./config /opt/gowebapp/config

# TODO --- add a volume declaration for the container configuration path we
want to
# mount at runtime from the host file system: /opt/gowebapp/config
# https://docs.docker.com/engine/reference/builder/#volume

EXPOSE 8080
USER 1000

WORKDIR /opt/gowebapp/
ENTRYPOINT ["/opt/gowebapp/gowebapp"]
```

Step 2: Build updated gowebapp Docker image locally¶

Build the gowebapp image locally. Make sure to include "." at the end. Notice the new version label.

```
docker build -t gowebapp:v2 .
```

9.3: Publish New Image¶

We built the second version of gowebapp from the last section and tested it locally. Now we can tag and push gowebapp:v2 to our Docker registry.

Step 1: update tag for gowebapp and publish to registry¶

```
docker tag gowebapp:v2 $REGISTRY_HOST/gowebapp:v2
docker push $REGISTRY_HOST/gowebapp:v2
```

9.4: Create a Secret¶

Step 1: create secret for the MySQL password¶

It’s not a good idea to hard-code sensitive information in configurations. Let’s create a secret for the MySQL database password, so that we can reference it from configurations. Additionally, let’s override the default password we placed and used inside the configuration file and Docker image.

```
kubectl create secret generic mysql --from-literal=password=mypassword

kubectl describe secret mysql
```

Step 2: update MySQL StatefulSet to incorporate the secret¶

Update gowebapp-mysql-sts.yaml If you need help, please see https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets-as-environment-variable

```
cd $HOME/gowebapp/gowebapp-mysql
```

Note

Replace TODO comments with the appropriate commands

`gowebapp-mysql-sts.yaml`

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: gowebapp-mysql
  labels:
    app: gowebapp-mysql
    tier: backend
spec:
  serviceName: gowebapp-mysql
  replicas: 1
  selector:
    matchLabels:
      app: gowebapp-mysql
      tier: backend
  template:
    metadata:
      labels:
        app: gowebapp-mysql
        tier: backend
    spec:
      securityContext:
        fsGroup: 1000
      containers:
        - name: gowebapp-mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  #TODO: Set secret name to 'mysql'
                  #TODO: Set key to 'password'
          image: gowebapp-mysql:v1
          ports:
            - containerPort: 3306
          livenessProbe:
            tcpSocket:
              port: 3306
            initialDelaySeconds: 20
            periodSeconds: 5
            timeoutSeconds: 1
          readinessProbe:
            exec:
              #TODO: replace "mypassword" in command line below to be
${MYSQL_ROOT_PASSWORD} and
              #      also update the line to this new format. Unfortunately we
need to move to
              #      a different format due to
https://github.com/kubernetes/kubernetes/issues/40846
              command:
                [
                  "bash",
                  "-c",
                  "mysql -uroot -pmypassword -e 'use gowebapp; select count(*)
from user'",
                ]
            initialDelaySeconds: 25
            periodSeconds: 10
            timeoutSeconds: 1
          volumeMounts:
            - name: mysql-pv
              mountPath: /var/lib/mysql
  volumeClaimTemplates:
    - metadata:
        name: mysql-pv
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 5Gi
```

We are using a custom image that we created in a previous lab. Therefore we need to add the registry server to the `image:` line in the YAML so that Kubernetes knows which registry to pull the image from. Otherwise it will try to find the image on the public/default configured registry server.

```
sed -i s/"image: gowebapp"/"image: $REGISTRY_HOST\/gowebapp"/g gowebapp-mysql-sts.yaml
```

9.5: Perform upgrade¶

Step 1: apply new StatefulSet¶

Start the upgrade

```
kubectl apply -f gowebapp-mysql-sts.yaml
```

Verify that the deployment was successful. The gowebapp-mysql pod’s status should be ‘Running’

```
kubectl get pods -l 'app=gowebapp-mysql'
```

9.6: Create ConfigMap¶

Step 1: create ConfigMap for gowebapp’s config.json file¶

```
kubectl create configmap gowebapp
--from-file=webapp-config-json=$HOME/gowebapp/gowebapp/config/config.json

kubectl describe configmap gowebapp
```

Notice that the entire file contents from config.json are stored under the key webapp-config-json

Step 2: update gowebapp deployment to utilize ConfigMap¶

Update gowebapp-deployment.yaml. If you need help, please see https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#add-configmap-data-to-a-specific-path-in-the-volume

```
cd $HOME/gowebapp/gowebapp
```

Note

Replace TODO comments with the appropriate commands

`gowebapp-deployment.yaml`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gowebapp
  labels:
    app: gowebapp
    tier: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gowebapp
      tier: frontend
  template:
    metadata:
      labels:
        app: gowebapp
        tier: frontend
    spec:
      containers:
        - name: gowebapp
          env:
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  #TODO: Set secret name to 'mysql'
                  #TODO: Set key name to 'password'
          # TODO: change image to v2
          image: gowebapp:v1
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: "256Mi"
              cpu: "250m"
            limits:
              memory: "512Mi"
              cpu: "500m"
          livenessProbe:
            tcpSocket:
              port: 8080
            periodSeconds: 5
            timeoutSeconds: 1
          readinessProbe:
            httpGet:
              path: /
              port: 8080
            periodSeconds: 5
            timeoutSeconds: 1
          volumeMounts:
            -  #TODO: Set volume name to 'config-volume'
              #TODO: Set mountPath to
'/opt/gowebapp/config'
      volumes:
        - #TODO: define volume name: config-volume
          configMap:
            #TODO: Set ConfigMap name to 'gowebapp'
            items:
              - key: webapp-config-json
                path: config.json
```

We are using a custom image that we created in the previous lab. Therefore we need to addd the registry server to the `image:` line in the YAML so that Kubernetes knows which registry to pull the image from. Otherwise it will try to find the image on the public/default configured registry server.

```
sed -i s/"image: gowebapp"/"image: $REGISTRY_HOST\/gowebapp"/g
gowebapp-deployment.yaml
```

9.7: Perform rolling upgrade¶

Step 1: apply new deployment¶

Start the rolling upgrade

```
kubectl apply -f gowebapp-deployment.yaml
```

Verify that rollout was successful. The two gowebapp pods' status should be 'Running'

```
kubectl get pods -l 'app=gowebapp'
```

9.8: Access your application¶

Step 1: access your application¶

You should be able to access the application by running the following command in the terminal and then clicking the URL it produces.

```
echo "http://$SESSION_NAME-gowebapp-k8s.$INGRESS_DOMAIN"
```

You can now test the application. Since we added persistent storage in the last lab, your username, password, and notes should still be valid and present.