Lab:
Running Kubernetes Jobs

Verify Minikube Is Running

- From a codespaces terminal, check if Minikube is still running with:
  `minikube status`
- If it is stopped, start it again with:
  `minikube start`

Introduction

- Before the case study events app can use the database, it must be initialized
  - Build the table schema, etc.
- This is a good use of a Kubernetes Job
- The code for the database initializer has already been written for you
  - It was downloaded when the git repo was pulled earlier in the course
  - In this lab, you will put it in a Docker container, store it in the container registry, and deploy it as a Kubernetes Job

Introduction (continued)

- Switch to the folder containing the database initializer code:

```
cd /workspaces/eventsapp/database-initializer
ls
```

- In this folder is a simple Node.js app that initializes the database
  - And a Dockerfile
- Feel free to investigate the `server.js` file and the Dockerfile

Containerize the DB Initializer

- Run the following commands to create the Docker container:

```
docker login -u your-docker-hub-id
```

  - Provide your password

```
docker build -t your-docker-hub-id/events-job:v1.0 .
```

```
docker push your-docker-hub-id/events-job:v1.0
```

Viewing MariaDB Properties

- When Helm installed MariaDB, it also created:
  - A secret storing the database root password
    - We will discuss secrets later
  - And a ClusterIP service
- Run the following commands to view these objects:

```
kubectl get secrets
kubectl get service
```

Create the db_init_job.yaml

- In the kubernetes-config folder, create a new file named db_init_job.yaml
  - Copy the yaml on this slide and paste it into the file
  - Replace your-dockerhub-id with your docker hub ID
  - Notice the environment variables being passed to the job
    - No need to change any other values

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: db-initializer
spec:
 template:
   spec:
     containers:
     - image: your-dockerhub-id/events-job:v1.0
       name: db-init-job
       imagePullPolicy: "Always"
       env:
       - name: DBHOST
         value: "database-server-mariadb"
       - name: DBUSER
         value: "root"
       - name: DBPASSWORD
         valueFrom:
           secretKeyRef:
             name: database-server-mariadb
             key: mariadb-root-password
     restartPolicy: Never
 backoffLimit: 4
```

Running the Job

- Run the following command to deploy the job:
  `kubectl apply -f db_init_job.yaml`
- Watch the job until it has completed once:
  `kubectl get jobs -w`
  - Press **CTRL+C** when done
- Get the name of the pod created by the job and view the logs:
  `kubectl get pods`
  `kubectl logs DB-INITIALIZER-POD-NAME-HERE`
  - You should see similar output to shown here:

```
$ kubectl logs db-initializer-7tjtm
1
Connected!
Database created
Tables created
Records added
8:07:22 PM
Exiting
$
```

If You Have More Time

- If you have time, experiment with a CronJob
  - Create a `cronjob.yaml` file and paste in the following yaml:
  - This will create a CronJob that runs once a minute
  - Apply the yaml file and watch the pods that are created every minute
  - View the logs for the pods

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
 name: hello
spec:
 schedule: "*/1 * * * *"
 jobTemplate:
   spec:
     template:
       spec:
         containers:
         - name: hello
           image: busybox
           args:
           - /bin/sh
           - -c
           - date; echo Hello from the Kubernetes cluster
         restartPolicy: OnFailure
```

Clean Up

- Delete the CronJob if you created it

```
kubectl delete cronjob hello
```

Syncing the Changes to Git

- Commit these changes to your Git repository
  - On the left side, click the **Source Control** button 
  - Be sure ALL changes are staged by clicking in the + button
  - Type a commit message of: **`Added Events app start code`** and click the **Commit** button
  - Press the **Sync Changes** button and press **OK** to push the changes
- The code has now been saved to your **eventsapp** Git repository created earlier

Success

- **Congratulations**! You have successfully used Kubernetes Jobs
  - Created a job to perform database initialization required for the case study
  - Created a CronJob to perform scheduled work