Deploying a Java Application to IBM Cloud Private

In this lab, you deploy a Java application in containers to IBM Cloud Private. The application you will deploy is one of the component microservices of a reference application called Light Blue Compute. Light Blue Compute is a simplified version of the Blue Compute reference application offered on the IBM Cloud Architecture web site. The application stores and maintains images and descriptions for the vintage computer products in an online store. Its data is stored in a MySQL database, which is also deployed in a container.

In this lab, you will be deploying the application in containers in a manual way, by compiling the code, packaging it in a container, and deploying that container to the Kubernetes cluster in IBM Cloud Private. Doing it this way will help you to understand the various steps that are carried out by the more automated methods of deployment you will be working with later.

This exercise assumes that the IBM Cloud Private command line interface plugin has been installed, and that you are generally familiar with your IBM Cloud Private lab environment.

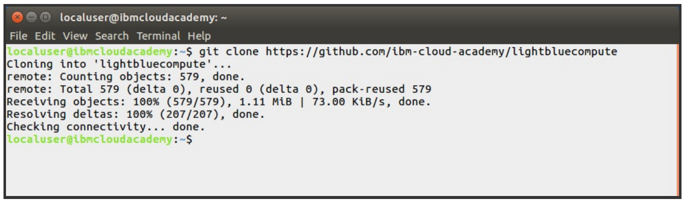
Note: In this exercise and others, there are various text files to be edited and configured. The command instructions in the lab assume the use of the `vi` editor. While `vi` is the prevalent historical text editor for Unix and Linux systems, it does have a learning curve and not all of our students will be familiar with it. If you are not comfortable with the `vi` editor, an alternative is to use `gedit`. You can use the `Files` application in the Application launcher to navigate to the file, then right-click to start the `gedit` editor.

Exercise 1: Deploying the MySQL Container

In this exercise, you create a container for a MySQL database. You deploy that container to your IBM Cloud Private cluster. You log in to the container and run a script to populate the MySQL database with the application data. Finally, you test to see that the data was successfully loaded.

1. In your terminal window, clone the git repository that contains the code for the application you will be deploying.

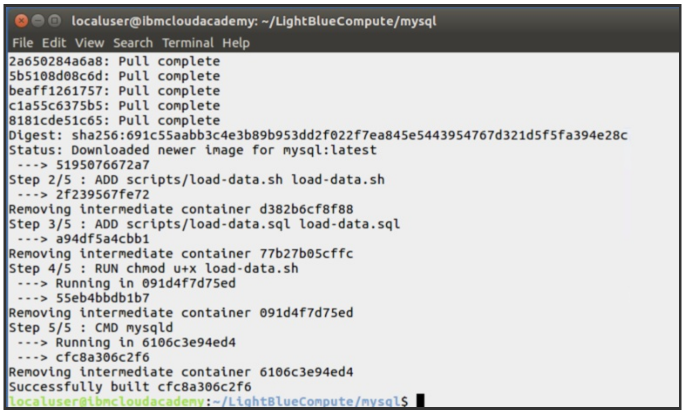
   ```
   git clone https://github.com/ibm-cloud-academy/LightBlueCompute
   ```


localuser@ibmcloudacademy: ~
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~$ git clone https://github.com/ibm-cloud-academy/lightbluecompute
Cloning into 'lightbluecompute'...
remote: Counting objects: 579, done.
remote: Total 579 (delta 0), reused 0 (delta 0), pack-reused 579
Receiving objects: 100% (579/579), 1.11 MiB | 73.00 KiB/s, done.
Resolving deltas: 100% (207/207), done.
Checking connectivity... done.
localuser@ibmcloudacademy:~$
```

2. Use the mysql folder contents and the commands below to build a docker image named `mysql`. The docker image is built using instructions in a file called `Dockerfile` in the `mysql` subdirectory. Edit the `Dockerfile` to see the commands if you haven't seen a Dockerfile before. When you go to build your container image, don't forget the dot (.) at the end of the `docker build` command.

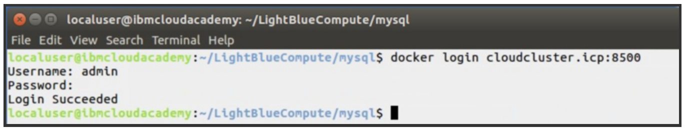
   ```
   cd ~/LightBlueCompute/mysql
   docker build –t mysql .
   ```

   
   localuser@ibmcloudacademy: ~/LightBlueCompute/mysql
   File Edit View Search Terminal Help

   ```
   2a650284a6a8: Pull complete
   5b5108d08c6d: Pull complete
   beaff1261757: Pull complete
   c1a55c6375b5: Pull complete
   8181cde51c65: Pull complete
   Digest: sha256:691c55aabb3c4e3b89b953dd2f022f7ea845e5443954767d321d5f5fa394e28c
   Status: Downloaded newer image for mysql:latest
    ---> 5195076672a7
   Step 2/5 : ADD scripts/load-data.sh load-data.sh
    ---> 2f239567fe72
   Removing intermediate container d382b6cf8f88
   Step 3/5 : ADD scripts/load-data.sql load-data.sql
    ---> a94df5a4cbb1
   Removing intermediate container 77b27b05cffc
   Step 4/5 : RUN chmod u+x load-data.sh
    ---> Running in 091d4f7d75ed
    ---> 55eb4bbdb1b7
   Removing intermediate container 091d4f7d75ed
   Step 5/5 : CMD mysqld
    ---> Running in 6106c3e94ed4
    ---> cfc8a306c2f6
   Removing intermediate container 6106c3e94ed4
   Successfully built cfc8a306c2f6
   localuser@ibmcloudacademy:~/LightBlueCompute/mysql$
   ```

3. Log in to the IBM Cloud Private image registry for docker. This provides the local docker engine credentials to push images to the IBM Cloud Private image registry. Use Username `admin` and Password `passw0rd`. All communications with the IBM Cloud Private image registry use port 8500.

   ```
   docker login cloudcluster.icp:8500
   ```

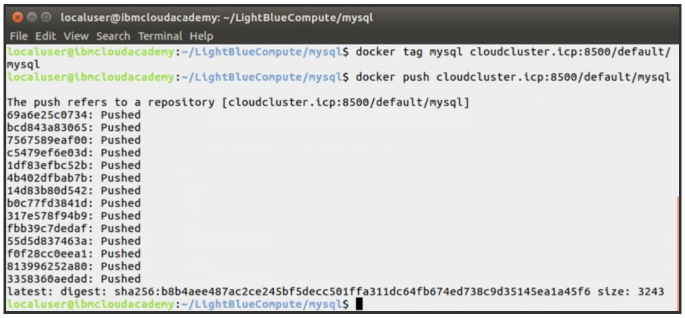
   
   localuser@ibmcloudacademy: ~/LightBlueCompute/mysql
   File Edit View Search Terminal Help

   ```
   localuser@ibmcloudacademy:~/LightBlueCompute/mysql$ docker login cloudcluster.icp:8500
   Username: admin
   Password:
   Login Succeeded
   localuser@ibmcloudacademy:~/LightBlueCompute/mysql$
   ```

4. Tag and push the docker image to your IBM Cloud Private image registry. The `docker tag` command creates an alias name for the container image that associates it with the image repository and

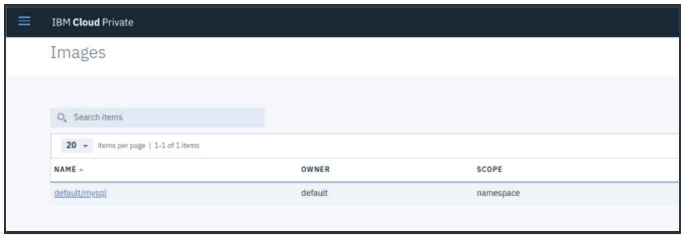
namespace that you want to push it to. In this example we will use the `default` namespace that comes out of the box with IBM Cloud Private. You can also create other namespaces if you wish. Namespaces allow you to enforce access control to the images in your cluster.

```
docker tag mysql cloudcluster.icp:8500/default/mysql
docker push cloudcluster.icp:8500/default/mysql
```


localuser@ibmcloudacademy: ~/LightBlueCompute/mysql
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~/LightBlueCompute/mysql$ docker tag mysql cloudcluster.icp:8500/default/
mysql
localuser@ibmcloudacademy:~/LightBlueCompute/mysql$ docker push cloudcluster.icp:8500/default/mysql

The push refers to a repository [cloudcluster.icp:8500/default/mysql]
69a6e25c0734: Pushed
bcd843a83065: Pushed
7567589eaf00: Pushed
c5479ef6e03d: Pushed
1df83efbc52b: Pushed
4b402dfbab7b: Pushed
14d83b80d542: Pushed
b0c77fd3841d: Pushed
317e578f94b9: Pushed
fbb39c7dedaf: Pushed
55d5d837463a: Pushed
f0f28cc0eea1: Pushed
813996252a80: Pushed
3358360aedad: Pushed
latest: digest: sha256:b8b4aee487ac2ce245bf5decc501ffa311dc64fb674ed738c9d35145ea1a45f6 size: 3243
localuser@ibmcloudacademy:~/LightBlueCompute/mysql$ 
```

5. Now that the image has been pushed to the IBM Cloud Private image registry, you should be able to see it in the User Interface. Go back to your browser session for IBM Cloud Private. In the menu on the top left, select `Menu > Catalog > Images`. You should see your image listed.



NAME	OWNER	SCOPE
default/mysql	default	namespace

6. Deploy the mysql container to Kubernetes. This step uses a Kubernetes deployment yaml file. Using vi or the editor of your choice, edit the mysql.yml file in the kubernetes subdirectory

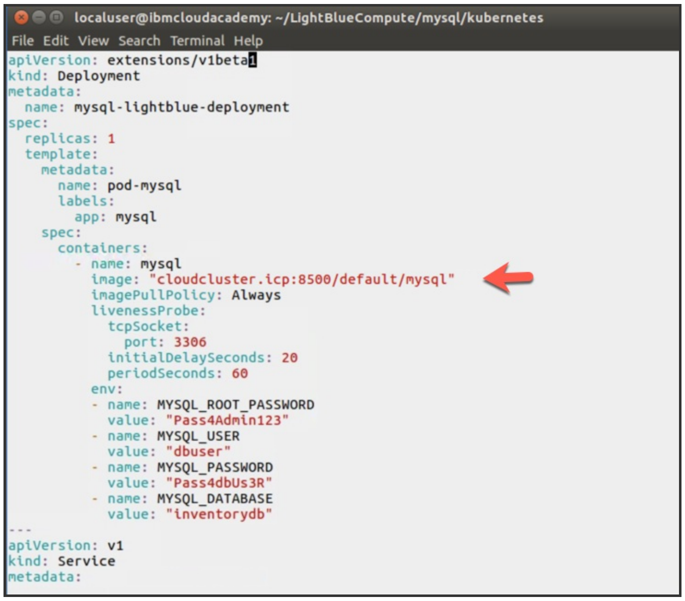
   ```
   cd ~/LightBlueCompute/mysql/kubernetes
   vi mysql.yml
   ```

   Make the following modifications.

   - Set your image source under `spec > template > spec > containers > image`. Make it match the tagged image name you just pushed to the IBM Cloud Private image registry.
   - Answer the following questions:

- How is the mysql health checked? ______________________________
- What port will the mysql pod be contacted on? ______________
- What is the database userID and password used? ____

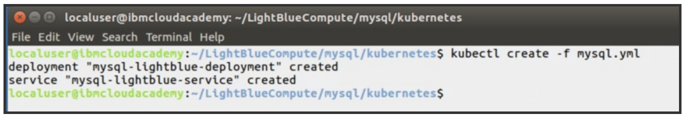
- Save the yml file (`:wq <Enter>`)


localuser@ibmcloudacademy: ~/LightBlueCompute/mysql/kubernetes
File Edit View Search Terminal Help

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: mysql-lightblue-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: pod-mysql
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: "cloudcluster.icp:8500/default/mysql"
          imagePullPolicy: Always
          livenessProbe:
            tcpSocket:
              port: 3306
            initialDelaySeconds: 20
            periodSeconds: 60
          env:
          - name: MYSQL_ROOT_PASSWORD
            value: "Pass4Admin123"
          - name: MYSQL_USER
            value: "dbuser"
          - name: MYSQL_PASSWORD
            value: "Pass4dbUs3R"
          - name: MYSQL_DATABASE
            value: "inventorydb"
---
apiVersion: v1
kind: Service
metadata:
```

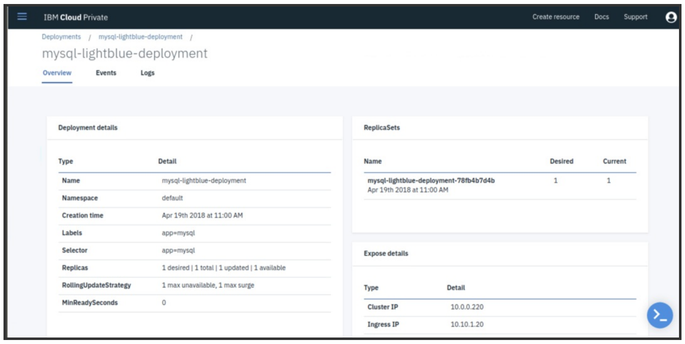
Deploy the yaml file using the following command:

```
kubectl create –f mysql.yml
```


localuser@ibmcloudacademy: ~/LightBlueCompute/mysql/kubernetes
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ kubectl create -f mysql.yml
deployment "mysql-lightblue-deployment" created
service "mysql-lightblue-service" created
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$
```

7. Now that you have created the deployment, and also created a Kubernetes service, as configured in the `mysql.yml` file, you should be able to see them in your IBM Cloud Private web user interface. Go to your browser session. Select `Menu > Workloads > Deployments`. You should see `mysql–lightblue–deployment` on the top of the list. Click that link to see more detailed information about this deployment.



mysql-lightblue-deployment

Deployment details

Type	Detail
Name	mysql-lightblue-deployment
Namespace	default
Creation time	Apr 19th 2018 at 11:00 AM
Labels	app=mysql
Selector	app=mysql
Replicas	1 desired | 1 total | 1 updated | 1 available
RollingUpdateStrategy	1 max unavailable, 1 max surge
MinReadySeconds	0

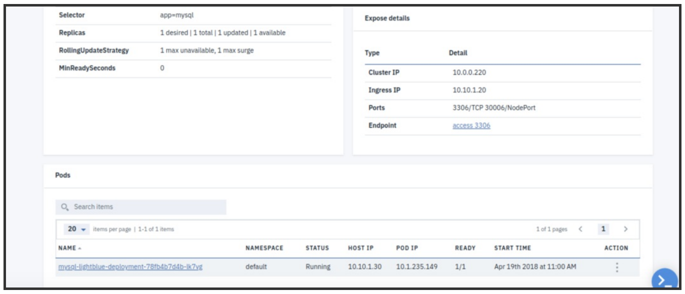
ReplicaSets

Name	Desired	Current
mysql-lightblue-deployment-78fb4b7d4b Apr 19th 2018 at 11:00 AM	1	1

Expose details

Type	Detail
Cluster IP	10.0.0.220
Ingress IP	10.10.1.20

Scroll down to see more information about the pod(s) the deployment has been deployed to.



Selector	app=mysql
Replicas	1 desired | 1 total | 1 updated | 1 available
RollingUpdateStrategy	1 max unavailable, 1 max surge
MinReadySeconds	0

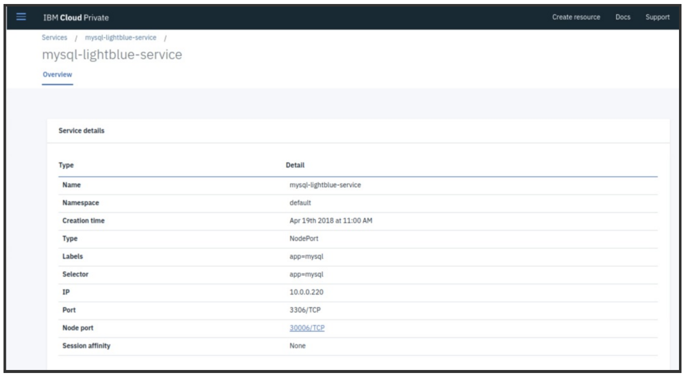
Expose details

Type	Detail
Cluster IP	10.0.0.220
Ingress IP	10.10.1.20
Ports	3306/TCP 30006/NodePort
Endpoint	access 3306

Pods

NAME	NAMESPACE	STATUS	HOST IP	POD IP	READY	START TIME	ACTION
mysql-lightblue-deployment-78fb4b7d4b-lk7vg	default	Running	10.10.1.30	10.1.235.149	1/1	Apr 19th 2018 at 11:00 AM	⋮

8. You can also display details of the Kubernetes service that provides access to your MySQL database. In your IBM Cloud Private browser web interface, select `Menu > Network Access > Services`. You should see `mysql-lightblue-service` on top of the list. Click that link to see detailed information about your service.



mysql-lightblue-service

Service details

Type	Detail
Name	mysql-lightblue-service
Namespace	default
Creation time	Apr 19th 2018 at 11:00 AM
Type	NodePort
Labels	app=mysql
Selector	app=mysql
IP	10.0.0.220
Port	3306/TCP
Node port	30006/TCP
Session affinity	None

9. Populate the table in the MySQL database

- Get the pod name. The pod name is based on the deployment name. To get the pod name, run the following command (all one line).

```
kubectl describe pod | grep mysql-lightblue-deployment | grep Name
```

- Open a shell session to the pod using the following command. Replace with the pod name returned by the previous command.

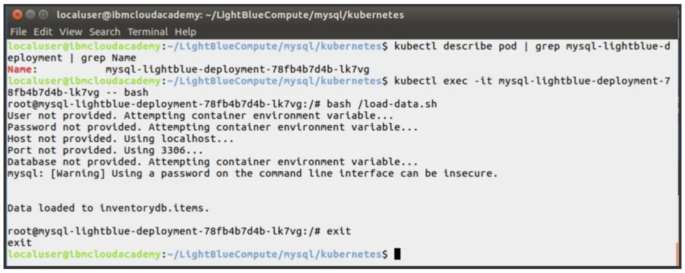
  `kubectl exec -it <podname> -- bash`

- Run the load-data.sh script, which populates the database:

  `bash /load-data.sh`

- Exit from the shell session:

  `exit`



localuser@ibmcloudacademy: ~/LightBlueCompute/mysql/kubernetes
File Edit View Search Terminal Help

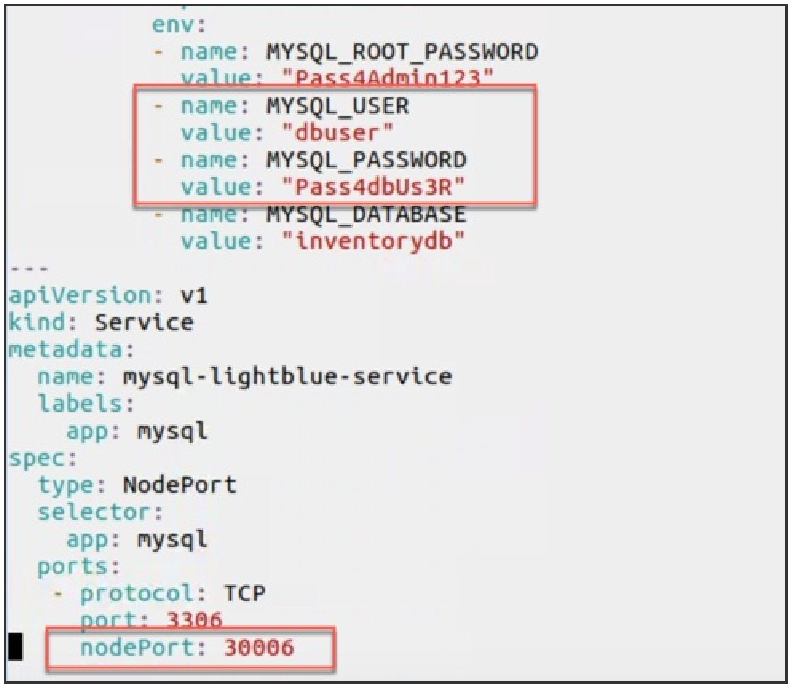
```
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ kubectl describe pod | grep mysql-lightblue-d
eployment | grep Name
Name:           mysql-lightblue-deployment-78fb4b7d4b-lk7vg
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ kubectl exec -it mysql-lightblue-deployment-7
8fb4b7d4b-lk7vg -- bash
root@mysql-lightblue-deployment-78fb4b7d4b-lk7vg:/# bash /load-data.sh
User not provided. Attempting container environment variable...
Password not provided. Attempting container environment variable...
Host not provided. Using localhost...
Port not provided. Using 3306...
Database not provided. Attempting container environment variable...
mysql: [Warning] Using a password on the command line interface can be insecure.


Data loaded to inventorydb.items.

root@mysql-lightblue-deployment-78fb4b7d4b-lk7vg:/# exit
exit
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ 
```

10. Verify that the data is loaded and accessible. To do this, you will use the mysql client that is installed on your VM. In the mysql command below, substitute the mysql user, password and the nodePort found in the `mysql.yml` file.

The `<worker_publicIP>` is the public IP of your worker node. If you can't remember this value, use the command `bx pr workers cloudcluster` to retrieve it. You can use any of the worker nodes' IP addresses in your command.

The `mysql` command below does not tolerate spaces between the parameter and the value for some parameters. Therefore it is best to not use a space between any parameters and their values.

`mysql -u<mysqluser> -p<mysqlpassword> -h<worker_publicIP> -P<nodePort>`



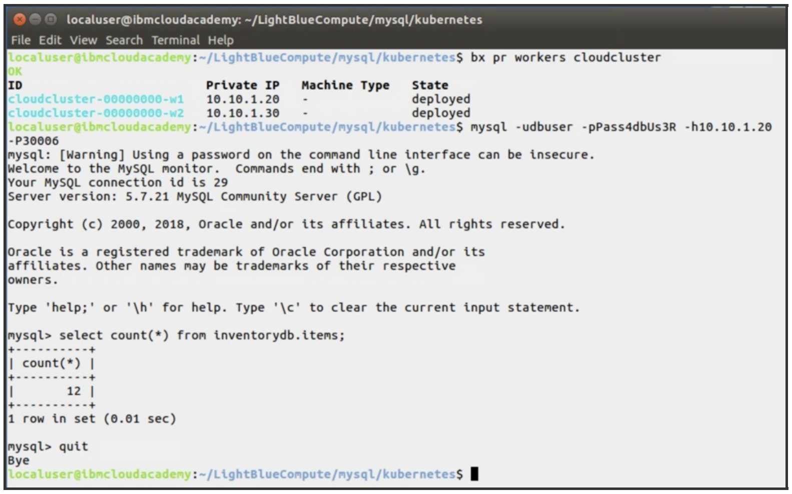
```
          env:
          - name: MYSQL_ROOT_PASSWORD
            value: "Pass4Admin123"
          - name: MYSQL_USER
            value: "dbuser"
          - name: MYSQL_PASSWORD
            value: "Pass4dbUs3R"
            name: MYSQL_DATABASE
            value: "inventorydb"
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-lightblue-service
  labels:
    app: mysql
spec:
  type: NodePort
  selector:
    app: mysql
  ports:
   - protocol: TCP
     port: 3306
     nodePort: 30006
```

Once connected, you can verify that the data has been loaded by running the following command:

`select count(*) from inventorydb.items;`

It should return a count of 12.

Type `quit` to exit.


localuser@ibmcloudacademy: ~/LightBlueCompute/mysql/kubernetes
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ bx pr workers cloudcluster
OK
ID                           Private IP   Machine Type   State
cloudcluster-00000000-w1     10.10.1.20   -              deployed
cloudcluster-00000000-w2     10.10.1.30   -              deployed
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ mysql -udbuser -pPass4dbUs3R -h10.10.1.20
-P30006
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select count(*) from inventorydb.items;
+----------+
| count(*) |
+----------+
|       12 |
+----------+
1 row in set (0.01 sec)

mysql> quit
Bye
localuser@ibmcloudacademy:~/LightBlueCompute/mysql/kubernetes$ 
```

Exercise 2: Deploying the Catalog Application

1. Explore the application.

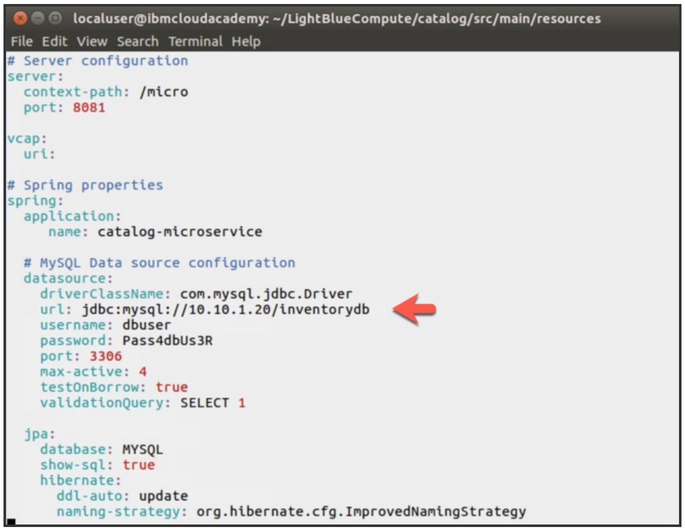
   `cd ~/LightBlueCompute/catalog`

   Check the existing application configuration in the application.yml file (`src/main/resources/application.yml`). What are the correct variable values for:

   - mysql URL: `____________________________________`
   - mysql port: `____________________________________`
   - mysql username: `____________________________________`
   - mysql password: `____________________________________`
   - application port: `____________________________________`

   These values should match the values for the mysql container deployment you just did in the last section. Hint: The mysql URL should include the Worker IP address.

2. Modify the application.yml file to match the configuration of your mysql deployment.

   
   localuser@ibmcloudacademy: ~/LightBlueCompute/catalog/src/main/resources
File Edit View Search Terminal Help

```
# Server configuration
server:
  context-path: /micro
  port: 8081

vcap:
  uri:

# Spring properties
spring:
  application:
     name: catalog-microservice

  # MySQL Data source configuration
  datasource:
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://10.10.1.20/inventorydb
    username: dbuser
    password: Pass4dbUs3R
    port: 3306
    max-active: 4
    testOnBorrow: true
    validationQuery: SELECT 1

  jpa:
    database: MYSQL
    show-sql: true
    hibernate:
      ddl-auto: update
      naming-strategy: org.hibernate.cfg.ImprovedNamingStrategy
```

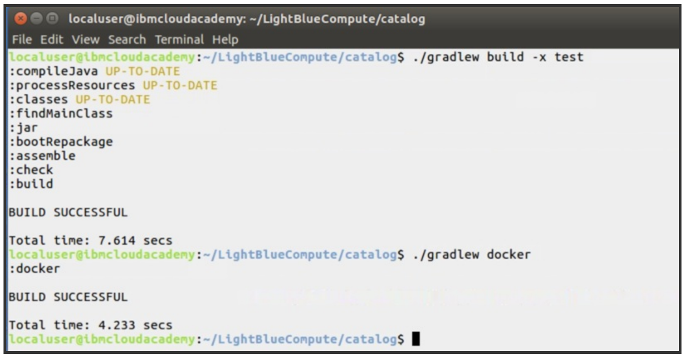
   Note: While it is good practice to have the application.yml file match the target environment for the application, when you are deploying the application to kubernetes, as you will be, some of the parameters in this file will be superceded by parameters in a yaml file for the kubernetes service (in this example the file will be called `catalog.yml`) which you will be configuring in a future step. The configuration change you just made to the mysql URL parameter in this example is therefore optional.

3. Look at the CatalogController.java file. (`src/main/java/catalog/CatalogController.java`). Check the paths that the application will respond on. There is nothing for you to modify in this file. Looking at it is just for your own interest.

```
| Path                 | Method       |
|----------------------|--------------|
|                      |              |
|                      |              |
|-------------------------------------|
```

4. Use the following commands to build the executable and copy the application jar file into the docker directory.

```
cd ~/LightBlueCompute/catalog
./gradlew build -x test
./gradlew docker
```

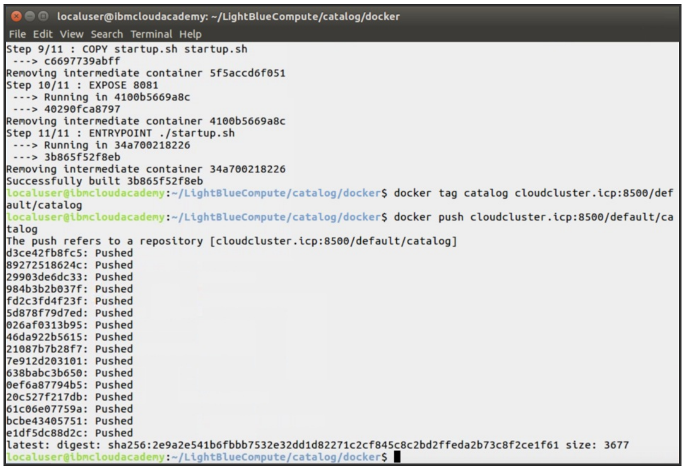

localuser@ibmcloudacademy: ~/LightBlueCompute/catalog
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~/LightBlueCompute/catalog$ ./gradlew build -x test
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:findMainClass
:jar
:bootRepackage
:assemble
:check
:build

BUILD SUCCESSFUL

Total time: 7.614 secs
localuser@ibmcloudacademy:~/LightBlueCompute/catalog$ ./gradlew docker
:docker

BUILD SUCCESSFUL

Total time: 4.233 secs
localuser@ibmcloudacademy:~/LightBlueCompute/catalog$
```

It is likely that your output will be more lengthy than that shown above. If newer versions of dependencies are available, they will be downloaded and this will lengthen the output. As long as you have a `BUILD SUCCESSFUL` message at the end, you are OK.

5. Build and upload the docker image. Don't forget the dot (.) at the end of the `docker build` command.

```
cd docker
docker build -t catalog .
docker tag catalog cloudcluster.icp:8500/default/catalog
docker push cloudcluster.icp:8500/default/catalog
```



localuser@ibmcloudacademy: ~/LightBlueCompute/catalog/docker
File Edit View Search Terminal Help

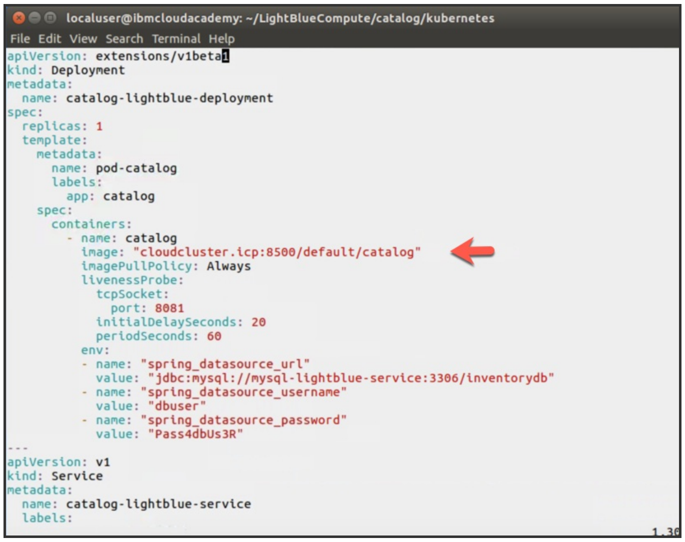
```
Step 9/11 : COPY startup.sh startup.sh
 ---> c6697739abff
Removing intermediate container 5f5accd6f051
Step 10/11 : EXPOSE 8081
 ---> Running in 4100b5669a8c
 ---> 40290fca8797
Removing intermediate container 4100b5669a8c
Step 11/11 : ENTRYPOINT ./startup.sh
 ---> Running in 34a700218226
 ---> 3b865f52f8eb
Removing intermediate container 34a700218226
Successfully built 3b865f52f8eb
localuser@ibmcloudacademy:~/LightBlueCompute/catalog/docker$ docker tag catalog cloudcluster.icp:8500/def
ault/catalog
localuser@ibmcloudacademy:~/LightBlueCompute/catalog/docker$ docker push cloudcluster.icp:8500/default/ca
talog
The push refers to a repository [cloudcluster.icp:8500/default/catalog]
d3ce42fb8fc5: Pushed
89272518624c: Pushed
29903de6dc33: Pushed
984b3b2b037f: Pushed
fd2c3fd4f23f: Pushed
5d878f79d7ed: Pushed
026af0313b95: Pushed
46da922b5615: Pushed
21087b7b28f7: Pushed
7e912d203101: Pushed
638babc3b650: Pushed
0ef6a87794b5: Pushed
20c527f217db: Pushed
61c06e07759a: Pushed
bcbe43405751: Pushed
e1df5dc88d2c: Pushed
latest: digest: sha256:2e9a2e541b6fbbb7532e32dd1d82271c2cf845c8c2bd2ffeda2b73c8f2ce1f61 size: 3677
localuser@ibmcloudacademy:~/LightBlueCompute/catalog/docker$ 
```

6. Using vi or the editor of your choice, edit the `catalog.yml` file in the kubernetes subdirectory. Modify the file so that the definition matches the name of your container image in the IBM Cloud Private registry.

   ```
   cd ~/LightBlueCompute/catalog/kubernetes
   vi catalog.yml
   ```

   Answer the following questions:

   - How does Kubernetes test this pod's health? `______________`
   - How does the catalog application connect to the mysql instance? `________________________`
   - What port is the catalog application exposed at? `_____`
   - What is the service name for the catalog application? `________________________`



localuser@ibmcloudacademy: ~/LightBlueCompute/catalog/kubernetes
File Edit View Search Terminal Help

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: catalog-lightblue-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: pod-catalog
      labels:
        app: catalog
    spec:
      containers:
        - name: catalog
          image: "cloudcluster.icp:8500/default/catalog"
          imagePullPolicy: Always
          livenessProbe:
            tcpSocket:
              port: 8081
            initialDelaySeconds: 20
            periodSeconds: 60
          env:
          - name: "spring_datasource_url"
            value: "jdbc:mysql://mysql-lightblue-service:3306/inventorydb"
          - name: "spring_datasource_username"
            value: "dbuser"
          - name: "spring_datasource_password"
            value: "Pass4dbUs3R"
---
apiVersion: v1
kind: Service
metadata:
  name: catalog-lightblue-service
  labels:
                                                                                   1,30
```

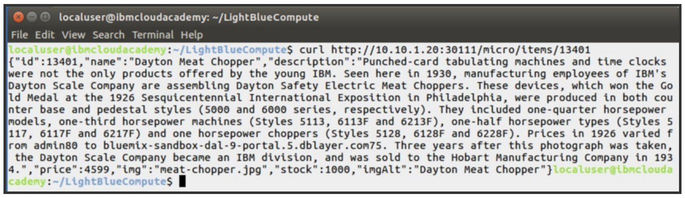
Deploy the yaml file using the command:

```
kubectl create -f catalog.yml
```

7. Now that you have sent the catalog deployment and service to your IBM Cloud Private cluster, you can visit the same pages in the web interface that you did for the mysql deployment to see the details.

8. Test the application. The following command should return the description of one of the products in the catalog.It uses the worker node IP Address and the Node Port of the catalog application service.

```
curl http://10.10.1.20:30111/micro/items/13401
```



localuser@ibmcloudacademy: ~/LightBlueCompute
File Edit View Search Terminal Help

```
localuser@ibmcloudacademy:~/LightBlueCompute$ curl http://10.10.1.20:30111/micro/items/13401
{"id":13401,"name":"Dayton Meat Chopper","description":"Punched-card tabulating machines and time clocks
were not the only products offered by the young IBM. Seen here in 1930, manufacturing employees of IBM's
Dayton Scale Company are assembling Dayton Safety Electric Meat Choppers. These devices, which won the Go
ld Medal at the 1926 Sesquicentennial International Exposition in Philadelphia, were produced in both cou
nter base and pedestal styles (5000 and 6000 series, respectively). They included one-quarter horsepower
models, one-third horsepower machines (Styles 5113, 6113F and 6213F), one-half horsepower types (Styles 5
117, 6117F and 6217F) and one horsepower choppers (Styles 5128, 6128F and 6228F). Prices in 1926 varied f
rom admin80 to bluemix-sandbox-dal-9-portal.5.dblayer.com75. Three years after this photograph was taken,
 the Dayton Scale Company became an IBM division, and was sold to the Hobart Manufacturing Company in 193
4.","price":4599,"img":"meat-chopper.jpg","stock":1000,"imgAlt":"Dayton Meat Chopper"}localuser@ibmclouda
cademy:~/LightBlueCompute$ █
```

If you see output of a product description like the example above, you can successfully deployed your application and completed this lab exercise.

END OF EXERCISE