Guided Learning

Deploy Microservices Applications with Kubernetes to IBM Cloud

Deploy a microservices application

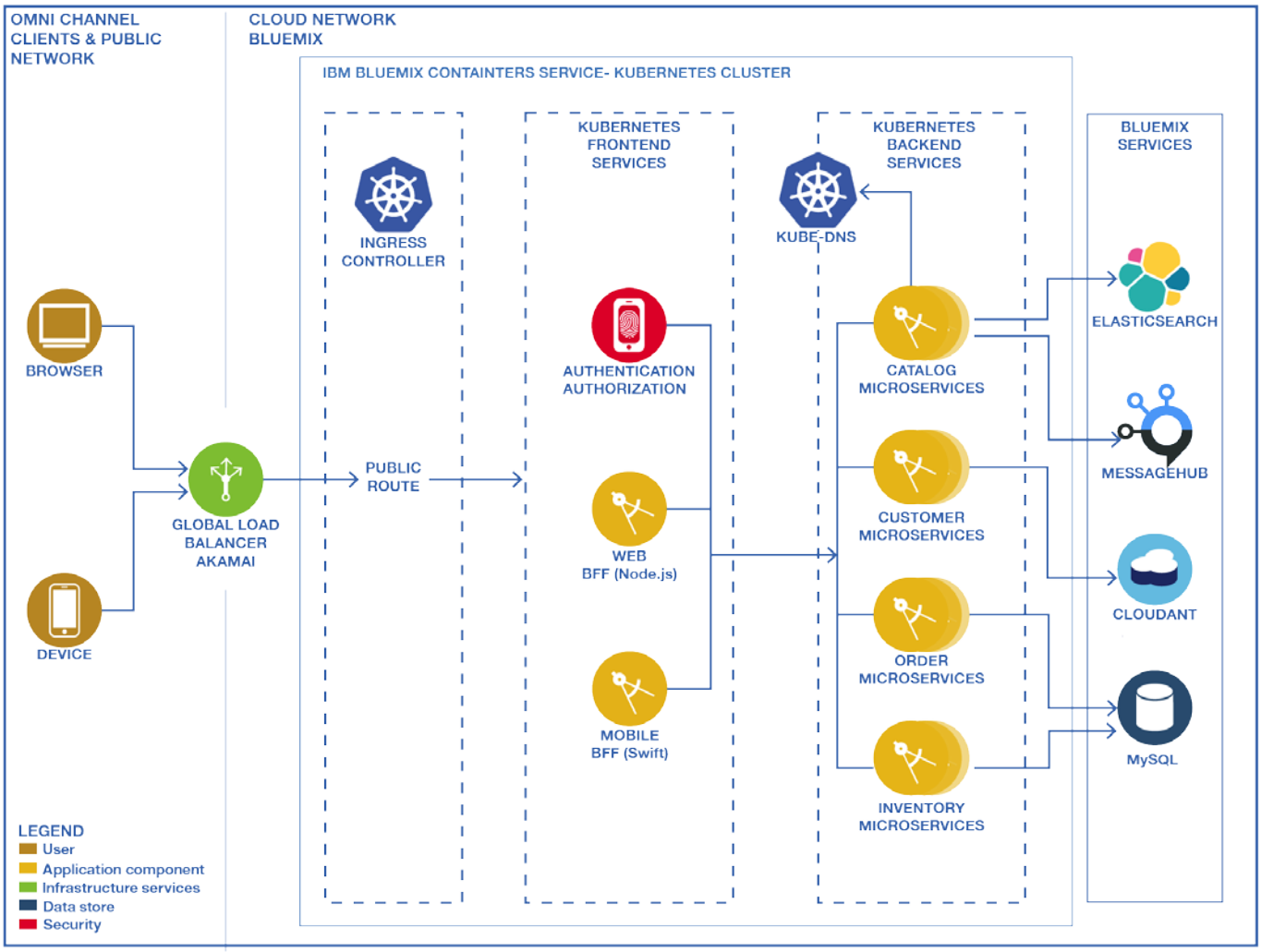
January 2018

IBM Training

January 2018

Deploy a microservices application

In this exercise, you deploy a sample microservices application to your cluster. The application is based on an IBM reference architecture application, and has several parts. For the purpose of this course, you only install and test certain parts of the application. This exercise takes about 2 hours to complete.



The application is a simple storefront shopping application that displays a catalog of antique computing devices, where users can search and buy products. It has both web and mobile interfaces: the mobile app and web app both rely on separate BFF (back-end for front-end) services to interact with the back-end data. This application is for demonstration only, and not all functions of the application are enabled at this time.

For more information about this sample application, see https://github.com/ibm-cloud-architecture/refarch-cloudnative-kubernetes.

You download this sample application from a repository on **github.com**. If you have not already done so, you must install **git** (as described in a previous activity) to download the application.

Important

- You must complete the previous activities before proceeding with this one.
- You must be logged in to IBM Cloud (`bx login`).
- You must be logged in to the IBM Cloud Container Registry CLI (`bx cr login`)
- You must set the context of your session to your cluster (`bx cs cluster-config <cluster-name>`)
- You also need the following information to complete this activity:
  - Your private image registry namespace (`bx cr namespaces`)
  - Your cluster name (`bx cs clusters`)
  - Worker node public IP address (`bx cs workers <cluster-name>`)

Task overview

1. Download the sample application.
2. Create the Cloudant database service.
3. Build the **customer** image.
4. Deploy the **customer** container.
5. Build the **mysql** image.
6. Deploy the **mysql** container.
7. Build the **catalog** image.
8. Deploy the **catalog** container.
9. Edit the **webapp `default.json`** configuration file.
10. Build the **webapp** image.
11. Deploy the **webapp** container.

Deploy a microservices application

Task 1. Download the sample application.

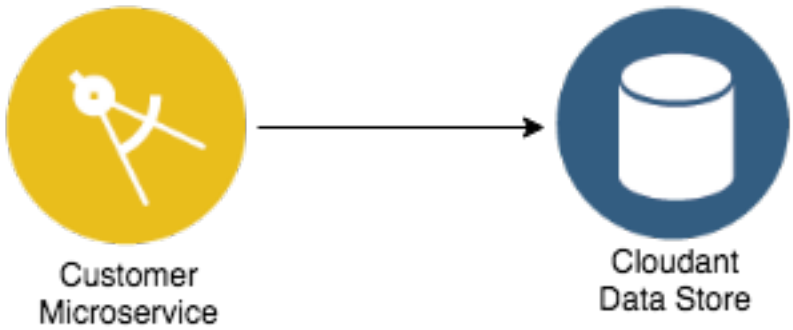
In a command prompt or terminal, run the following command:

```
$ git clone https://github.com/ibm-learning-skills-dev/LearningKubeApp.git
```

This command downloads a copy of the repository to your local directory.

Task 2. Create the Cloudant database service.

The **customer** microservice uses a Cloudant database, which must be configured before deploying the microservice.



Before proceeding with this task, make sure that you are logged in to IBM Cloud and the IBM Container Registry CLI, and set the context of your session to your cluster, as described in a previous exercise.

1. In a command prompt or terminal, run the following command to create the Cloudant database service:

   ```
   bx service create cloudantNoSQLDB Lite <cloudantdb-name>
   ```

   Where <*cloudantdb-name*> is a name of your choosing:

   - Cloudant service name: _____

   Remember this name. You will need to specify this value with subsequent commands.

Troubleshooting:

This command might fail if you did not target an organization and a space when you logged in to IBM Cloud. If this is the case, you can run **`bx target –o <org> -s <space>`** to set the target to your organization and space, and try running the above command again.

2. Run the following command to define a credential for the Cloudant service:

```
bx service key-create <cloudantdb-name> cred
```

Specify your Cloudant service name with the command.

3. Run the following command to get the secret information for the credential:

```
bx service key-show <cloudantdb-name> cred
```

Specify your Cloudant service name with the command.

4. Record the following information from the output of the command, for future reference:
   - Hostname: _____
   - Username: _____
   - Password: _____

Tip: You can cut and paste this information into a text file, if you prefer. You can also view the credential details in **IBM Cloud dashboard > *service-name* > Service credentials**.

---

5. Run the following command to bind the Cloudant service with your cluster:

```
bx cs cluster-service-bind <cluster-name> default <cloudantdb-name>
```

Specify your cluster name for <*cluster-name*> and your Cloudant service name for <*cloudantdb-name*>.

6. Record the secret name from the output of the command. You will need this information later.
   - Secret name: _____
7. Run the following command to view the secret details:

```
kubectl describe secret <secret-name>
```

Where <*secret-name*> is the secret name that you recorded above.

Task 3. Build the customer image.

1. In a command prompt or terminal, change to the directory, **LearningKubeApp/customer**.
2. Run the following commands to prepare Docker:

```
$ ./gradlew build -x test
$ ./gradlew docker
```

3. Change to the directory **LearningKubeApp/customer/docker**, and run the following commands to build and upload the Docker image to your private image registry in IBM Cloud:

   ```
   docker build -t customer .
   docker tag customer registry.<region>.bluemix.net/<namespace>/customer
   docker push registry.<region>.bluemix.net/<namespace>/customer
   ```

   Specify your region for <*region*>, and your private image registry namespace for <*namespace*>. For example, if your region is **US South**, and your namespace is **`learningkubens`**, enter:

   ```
   $ docker build -t customer .
   $ docker tag customer registry.ng.bluemix.net/learningkubens/customer
   $ docker push registry.ng.bluemix.net/learningkubens/customer
   ```

4. To verify that the image was added successfully, run the following command:

   ```
   $ bx cr images
   ```

Task 4. Deploy the customer container.

Before deploying the **customer** container, you must change some of the information that is specified in the deployment configuration. You can use **vi**, if available, or the text editor of your choice to edit the file. The file is in **LearningKubeApp/customer/kubernetes/customer.yml** of the repository that you cloned from github.

1. Open the **`customer.yml`** file for editing.
2. Make the following changes:
   - Change the **`image`** name to **"registry.<*region*>.bluemix.net/<*namespace*>/customer:latest"**. Specify your region for **<*region*>**, and your private image registry namespace for **<*namespace*>**.
   - Change the **`secretName`** to the **<*secret-name*>** that you obtained previously.

```
spec:
  containers:
    - name: customer
      image: "registry.ng.bluemix.net/<namespace>/customer:latest"
      imagePullPolicy: Always
      volumeMounts:
      - mountPath: /var/run/secrets/binding-refarch-cloudantdb
        name: binding-refarch-cloudantdb
      livenessProbe:
        tcpSocket:
          port: 8080
        initialDelaySeconds: 20
        periodSeconds: 60
  volumes:
  - name: binding-refarch-cloudantdb
    secret:
      defaultMode: 420
      secretName: "binding-<cloudant-instance>"
```

3. Note the value for **`nodePort`** at the bottom of the file (**`30110`**). This is how the application is accessed from outside the cluster.
4. Save and exit the file.
5. In a command prompt or terminal, change to the directory of the configuration file and run the following command to deploy the configuration:

   ```
   $ kubectl create -f customer.yml
   ```

6. Test the application by running the following command:

   ```
   curl -X POST -H "Content-Type: application/json" -H "Accept: application/json" -d '{"username": "foo", "password": "bar", "firstName": "foo", "lastName": "bar", "email": "foo@bar.com"}' -i http://<worker-IP>:<port>/micro/customer
   ```

   Specify your worker node public IP address for *<worker-IP>*, and port **30110** for *<port>*.

7. Run the following command to retrieve the customer record:

   ```
   curl http://<worker-IP>:<port>/micro/customer/search?username=foo
   ```

   Again, specify your worker node public IP address for *<worker-IP>*, and port **30110** for *<port>*.

Note: You can also use the Kubernetes dashboard to check the status of the pod.

8. Run the following command to start the **kube-proxy**:

   $ **kubectl proxy**

   And then, open a web browser to **http://localhost:8001/ui**. When you are finished with the Kubernetes dashboard, you can close the browser, and press **Ctrl-C** in the command prompt or terminal to end the process.

Task 5. Build the mysql image.

1. In a command prompt or terminal, change to the directory, **LearningKubeApp/mysql**.
2. Run the following commands to prepare Docker:

   $ **./gradlew build -x test**

   $ **./gradlew docker**

3. Change to the directory **LearningKubeApp/mysql/docker**, and run the following commands to build and upload the Docker image to your private image registry in IBM Cloud:

   ```
   docker build -t mysql .
   docker tag mysql registry.<region>.bluemix.net/<namespace>/mysql
   docker push registry.<region>.bluemix.net/<namespace>/mysql
   ```

   Specify your region for <*region*>, and your private image registry namespace for <*namespace*>. For example, if your region is **US South**, and your namespace is **learningkubens**, enter:

   $ **docker build -t mysql .**

   $ **docker tag mysql registry.ng.bluemix.net/learningkubens/mysql**

   $ **docker push registry.ng.bluemix.net/learningkubens/mysql**

4. To verify that the image was added successfully, run the following command:

   $ **bx cr images**

Task 6. Deploy the mysql container.

Before deploying the **mysql** container, you must change some of the information that is specified in the deployment configuration for it. You can use **vi**, if available, or the text editor of your choice to edit the file. The file is in

LearningKubeApp/mysql/kubernetes/mysql.yml of the repository that you cloned from github.

1. Open the **`mysql.yml`** file for editing.
2. Change the image name to **`"registry.<region>.bluemix.net/<namespace>/mysql:latest"`**. Specify your region for ***`<region>`***, and your private image registry namespace for ***`<namespace>`***.
3. Note the **env** section in the file that defines environment variables for the MySQL database:

```
env:
- name: MYSQL_ROOT_PASSWORD
  value: "Pass4Admin123"
- name: MYSQL_USER
  value: "dbuser"
- name: MYSQL_PASSWORD
  value: "Pass4dbUs3R"
- name: MYSQL_DATABASE
  value: "inventorydb"
```

   This is how the MySQL database is accessed.
4. Also note the value for **`nodePort`** at the bottom of the file (**`30006`**). This is how the application is accessed from outside the cluster.
5. Save and exit the file.
6. In a command prompt or terminal, change to the directory of the configuration file and run the following command to deploy the configuration:

   $ **`kubectl create -f mysql.yml`**
7. Run the following command to get the pod name:

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

   The pod name is displayed in the output.
8. Run the following command to open a shell session with the pod:

   `kubectl exec -it <pod-name> -- bash`

   Specify the pod name that you obtained for *`<pod-name>`*.
9. Run the following script to populate the table:

```
# bash /load-data.sh
```

10. Exit the shell session:

```
# exit
```

11. To verify that the script ran successfully, run the following command to start the mysql client:

```
mysql –u <mysql-user> -p<mysql-password> -h <worker-IP> -P <port>
```

Specify the MySQL user name and password that you obtained from the **mysql.yml** file, and your worker node public IP address and port where appropriate with the command. *Do not* include a space between the **–p** option and the password value. For example:

```
$ mysql –u dbuser -pPass4dbUs3R -h 123.456.78.9 -P 30006
```

Remember to substitute your worker node IP address for the one shown above!

12. To verify the data, run the following command:

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

The output looks similar to the following:

```
+----------+
| count(*) |
+----------+
|       12 |
+----------+
1 row in set (0.06 sec)
```

13. Exit the **mysql** session:

```
mysql> exit
```

Task 7. Build the catalog image.

1. In a command prompt or terminal, change to the directory, **LearningKubeApp/catalog**.
2. Run the following commands to prepare Docker:

```
$ ./gradlew build -x test
$ ./gradlew docker
```

1. Change to the directory **LearningKubeApp/catalog/docker**, and run the following commands to build and upload the Docker image to your private image registry in IBM Cloud:

```
docker build -t catalog .
docker tag catalog registry.<region>.bluemix.net/<namespace>/catalog
docker push registry.<region>.bluemix.net/<namespace>/catalog
```

Specify your region for *<region>,* and your private image registry namespace for *<namespace>*. For example, if your region is **US South**, and your namespace is **learningkubens**, enter:

```
$ docker build -t catalog .
$ docker tag catalog registry.ng.bluemix.net/learningkubens/catalog
$ docker push registry.ng.bluemix.net/learningkubens/catalog
```

2. To verify that the image was added successfully, run the following command:

```
$ bx cr images
```

Task 8. Deploy the catalog container.

Before deploying the **catalog** container, you must change some of the information that is specified in the deployment configuration. You can use **vi**, if available, or the text editor of your choice to edit the file. The file is in **LearningKubeApp/catalog/kubernetes/catalog.yml** of the repository that you cloned from github.

1. Open the **catalog.yml** file for editing.
2. Change the **image** name to **"registry.<*region*>.bluemix.net/<*namespace*>/catalog:latest".** Specify your region for ***<region>,*** and your private image registry namespace for ***<namespace>***.
3. Note the port value that is specified for **nodePort** (**30111**).
4. Save and exit the file.
5. In a command prompt or terminal, change to the directory of the configuration file and run the following command to deploy the configuration:

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

6. Run the following command to test the app:

```
curl http://<worker-IP>:<port>/micro/items/13401
```

Specify your worker node public IP address for *<worker-IP>,* and **30111** for *<port>.* The output of the command shows information about item #13401 in the catalog.

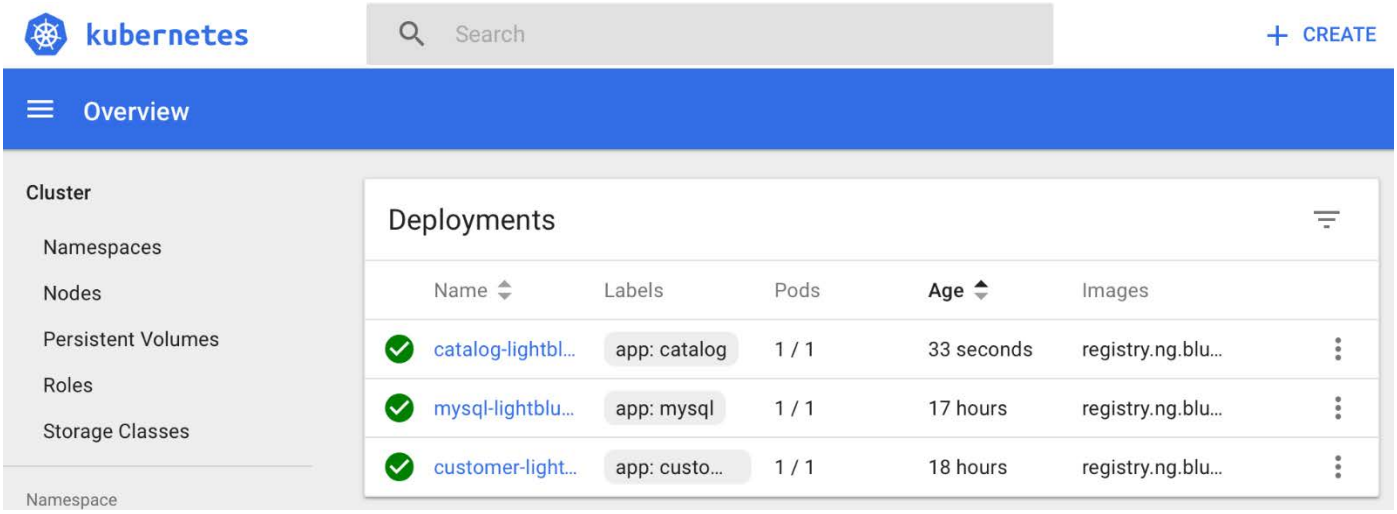
Note: You can also use the Kubernetes dashboard to check the status of the pod.

---

7. Run the following command to start the kube-proxy:

   $ **`kubectl proxy`**

   And then, open a web browser to **`http://localhost:8001/ui`**.



   At this point, you see three deployments. When you are finished with the Kubernetes dashboard, you can close the browser, and press **Ctrl-C** in the command prompt or terminal to end the process.

Task 9. Edit the **`default.json`** configuration file.

Before building the **webapp** image, you must make some changes to the file **LearningKubeApp/web-app-lite/config/default.json**.

1. Open the **`default.json`** file for editing.
2. Make the following changes:

- Change the **`"cataloghost"`** and **`"customerhost"`** values to correspond with the service definitions in the deployment configurations for those containers.
  - Change **`"cataloghost"`** to **`"catalog-lightblue-service:8081"`**.
  - Change **`"customerhost"`** to **`"customer-lightblue-service:8080"`**.
- Change the value for **`base_path`** in the **`catalog`** and **`customer`** sections from **`"/api"`** to **`"/micro".`**

```
"APIs": {
  "catalog": {
    "host": "cataloghost",
    "base_path": "/api",
    "require": [
        "client_id"
    ]
  },
  "order": {
    "host": "orderhost",
    "base_path": "/api",
    "require": [
      "client_id",
      "oauth"
    ]
  },
  "customer": {
    "host": "customerhost",
    "base_path": "/api",
    "require": [
      "client_id"
```

Service definition from **`catalog.yml`**:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: catalog-lightblue-service
  labels:
    app: catalog
spec:
  type: NodePort
  selector:
    app: catalog
  ports:
   - protocol: TCP
     port: 8081
     nodePort: 30111
```

Service definition from **`customer.yml`**:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: customer-lightblue-service
  labels:
    app: customer
spec:
  type: NodePort
  selector:
    app: customer
  ports:
   - protocol: TCP
     port: 8080
     nodePort: 30110
```

The APIs section in the **`default.json`** file should now look similar to this:

```
"APIs": {
  "catalog": {
    "host": "catalog-lightblue-service:8081",
    "base_path": "/micro",
    "require": [
        "client_id"
    ]
  },
  "order": {
    "host": "orderhost",
    "base_path": "/api",
    "require": [
      "client_id",
      "oauth"
    ]
  },
  "customer": {
    "host": "customer-lightblue-service:8080",
    "base_path": "/micro",
    "require": [
      "client_id"
```

3. Save and exit the file.

Task 10. Build the webapp image.

1. In a command prompt or terminal, change to the directory, **LearningKubeApp/web-app-lite**.
2. Run the following commands to prepare Docker:

   $ **`./gradlew build -x test`**

   $ **`./gradlew docker`**

3. In the directory **LearningKubeApp/web-app-lite/** directory, run the following commands to build and upload the Docker image to your private image registry in IBM Cloud:

   ```
   docker build -t webapp .
   docker tag webapp registry.<region>.bluemix.net/<namespace>/webapp
   docker push registry.<region>.bluemix.net/<namespace>/webapp
   ```

   Specify your region for *<region>,* and your private image registry namespace for *<namespace>*. For example, if your region is **US South**, and your namespace is **learningkubens**, enter:

   ```
   $ docker build -t webapp .
   $ docker tag webapp registry.ng.bluemix.net/learningkubens/webapp
   $ docker push registry.ng.bluemix.net/learningkubens/webapp
   ```

Note: If you are using a free trial account on IBM Cloud, you might exceed your quota at this point. You can work around this limitation by deleting one of the older images that you created earlier in the course (it is not needed to proceed with the remaining tasks).

4. To delete an image, run the following command:

   ```
   bx cr image-rm <image-name>
   ```

   Specify an image name for `<image-name>`. You can get a list of image names by running the command:

   ```
   $ bx cr images
   ```

   For example, to delete the **customer** image, run the command:

   ```
   $ bx cr image-rm registry.ng.bluemix.net/learningkubens/customer
   ```

   Remember to go back to the previous step to build the **webapp** image again.

Task 11. Deploy the web application.

Before deploying the **webapp** container, you must change some of the information that is specified in the deployment configuration. You can use **vi**, if available, or the text editor of your choice to edit the file. The file is in **LearningKubeApp/web-app-lite/kubernetes/webapp.yml** of the repository that you cloned or downloaded from github.

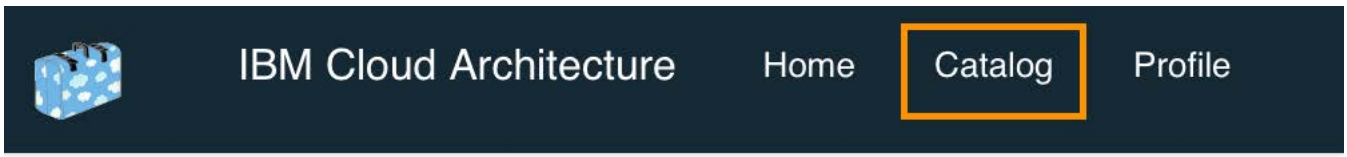
1. Change the **image** name to **"registry.<*region*>.bluemix.net/<*namespace*>/webapp:latest".**

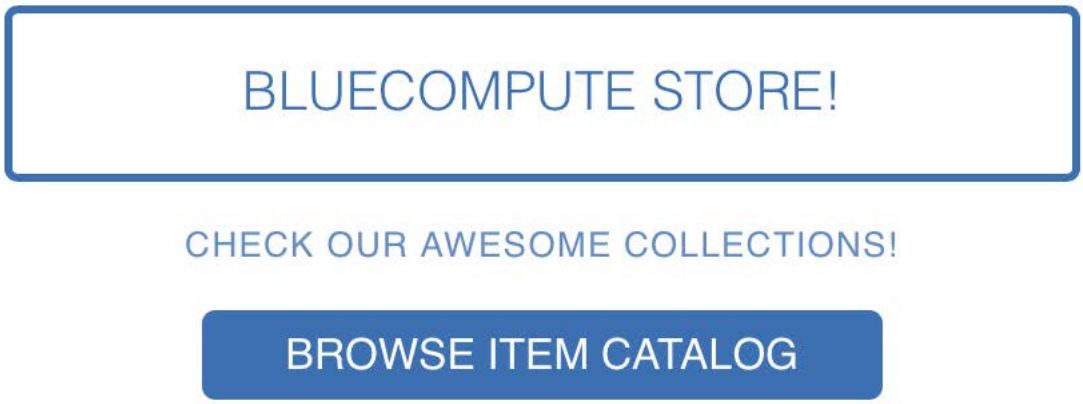
Specify your region for ***<region>***, and your private image registry namespace for ***<namespace>***.

2. Note the port value that is specified for **`nodePort`** (**`30130`**).
3. Save and exit the file.
4. In a command prompt or terminal, change to the directory of the configuration file and run the following command to deploy the configuration:

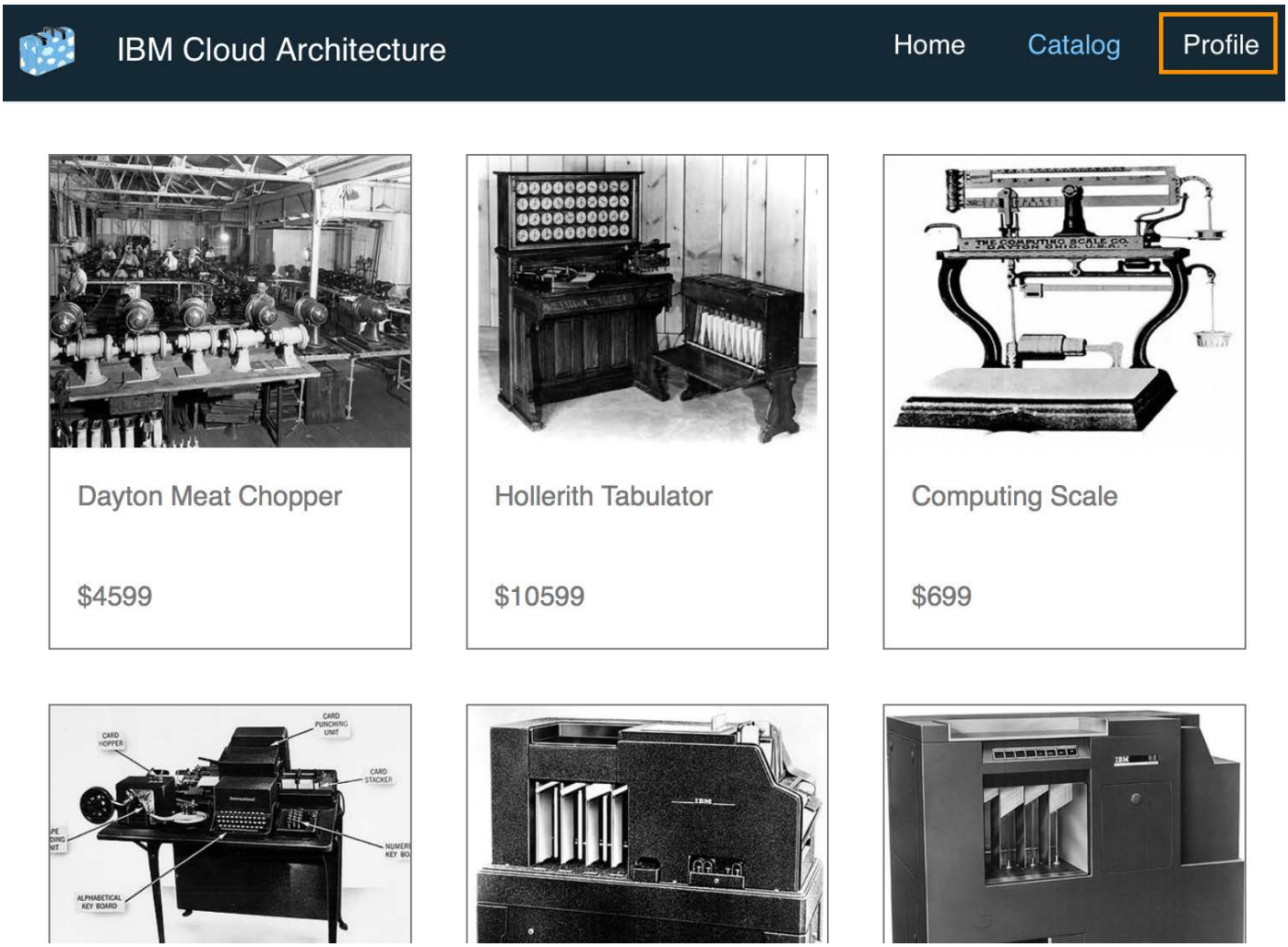
   ```
   $ kubectl create -f webapp.yml
   ```

5. To test the application, in a web browser, go to `http://<worker-IP>:<port>/`, where `<worker-IP>` is your worker node public IP address, and the port is the **nodePort** for **webapp** (**30130**).

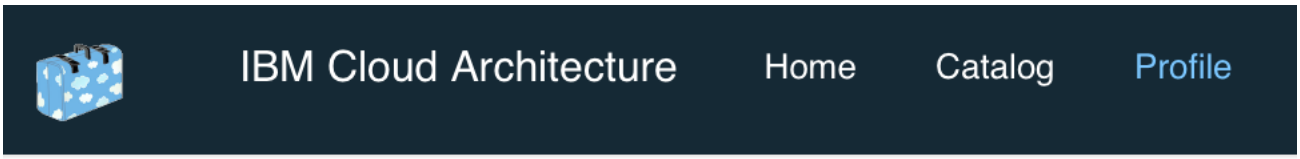




6. Click **Catalog**.



7. Click **Profile**.



Please review your profile

First Name: foo

Last Name: bar

Email: foo@bar.com

Username: foo

Note: For the **customer** application, the search for a customer called “foo bar” is hard-coded, as the application does not yet have a security and login component configured.

End of exercise.