Containerize and Deploy the Application

Estimated time needed: 90 mins

You have made good progress in your project thus far!

- Your GiftLink Application is fully functional
- The front end is completely functional and integrates with the back end
- Your team is happy!

The company wants to use containers to manage and deploy the application. Furthermore, the management wants a hybrid cloud strategy, where some applications and services reside on a private cloud and others on a public cloud.

To provide a more robust development experience, you will use both Kubernetes and IBM Code Engine. So, let's containerize your application now.

Prerequisites

Please ensure that the giftlink-frontend and giftlink-backend are complete and the code has been pushed into the git repository.

> **Note**: Before starting the lab, ensure you deleted any previously persisting sessions to avoid issues while running this lab.

To delete previous sessions run the following command:

```
kubectl get deployments | grep sn-labs-$USERNAME
```

If you see that the `mongodb` and/or `giftapp` deployment already exists, delete it using these commands:

```
kubectl delete deployment mongodb
```

```
kubectl delete deployment giftapp
```

Now run this command:

```
ibmcloud cr images
```

If you see any `mongodb` and/or `giftapp` images, please delete them using:

```
ibmcloud cr image-rm us.icr.io/sn-labs-$USERNAME/mongodb:latest && docker rmi us.icr.io/sn-labs-$USERNAME/mongodb:latest
```

```
ibmcloud cr image-rm us.icr.io/sn-labs-$USERNAME/giftapp:latest && docker rmi us.icr.io/sn-labs-$USERNAME/giftapp:latest
```

If you do not remember your namesapce, you can get it by using either of the below commands:

- `oc project`
- `ibmcloud cr namespaces` (please use the one that is of the format `sn-labs-$USERNAME`)
- Sign out of SN Labs and clear your browser cache and cookies.
- Start the lab again and proceed.

Set Up the MongoDB Server

1. Open a new terminal and ensure you are in the `/home/project` directory.
2. You will first deploy MongoDB on Kubernetes. Create a file name `deploymongo.yml` and add the code below.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo:latest
        ports:
        - containerPort: 27017
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: NodePort
  ports:
  - port: 27017
    targetPort: 27017
    nodePort: 30007
  selector:
    app: mongodb
```

3. Create the deployment using the following command and your deployment file:

```
kubectl apply -f deploymongo.yml
```

Note: If you get an error like “port already allocated”, please change nodePort to `30008`.

4. Use the following command to check if your Mongo service is ready:

```
kubectl get deployments
```

Set Up the Back-end Server

1. Clone your repository into which you pushed all your code changes.
2. Change to the `giftlink-backend` folder.
3. Export your SN Labs namespace and print it on the console as shown below below:

```
MY_NAMESPACE=$(ibmcloud cr namespaces | grep sn-labs-)
echo $MY_NAMESPACE
```

4. Create a `Dockerfile` by running the following command:

```
touch Dockerfile
```

5. Open the `Dockerfile` in the editor and write the code to containerize the server application.

▼ Click here for a sample

```
FROM node:18.12.1-bullseye-slim
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3060
CMD ["node", "app.js"]
```

6. In your .env file configure the server to point to `mongodb://mongodb-service:27017` as per the deployment service name in kubernetes. Also add your `JWT_SECRET`.

▼ You can use the below sample code

```
MONGO_URL=mongodb://mongodb-service:27017
JWT_SECRET=mysecret
```

Note: If you have added a username and password for your MongoDB, that needs to be added as well.

Now you will build your image and push it to the IBM Cloud Image Registry (ICR). You need to do the same here and then refer to this image in your Kubernetes deployment file.

7. Perform a docker build with the `Dockerfile` in the current directory.

```
docker build . -t us.icr.io/$MY_NAMESPACE/giftapp
```

8. Next, push the image to the container registry:

```
docker push us.icr.io/$MY_NAMESPACE/giftapp
```

9. Create a new file `deployment.yml` and paste the following content in it.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: giftapp
  labels:
    app: giftapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: giftapp
  template:
    metadata:
      labels:
        app: giftapp
    spec:
      containers:
      - name: giftapp-container
        image: <the-name-of-your-image>
        ports:
        - containerPort: 3060
      imagePullSecrets:
      - name: icr
---
apiVersion: v1
kind: Service
metadata:
  name: gift-app-service
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 3060
    protocol: TCP
  selector:
    app: giftapp
```

Make sure you replace `<the-name-of-your-image>` with the actual image name.

10. Create the deployment using the following command and your deployment file:

    ```
    kubectl apply -f deployment.yml
    ```

11. Normally, we would add a service to our deployment; however, you will use port-forwarding in this environment to see the running application.

    ```
    kubectl port-forward deployment.apps/giftapp 3060:3060
    ```

    Note: After executing the deployment command, allow time for the pods to start. If you encounter errors, wait and try the command again. If problems persist, use `kubectl describe pod <pod-name>` for detailed information about the pod's state and issues..

12. Open a new terminal.

13. Check to see if your services are ready with the following command:

    ```
    kubectl get deployments
    ```

14. Take a screenshot of the terminal showing the MongoDB service and the back-end server running and save it as `backend_deployment.png` or `backend_deployment.jpg`.

15. Click the `Gifts Server` button below or click the `Skills Network` button on the right.

- It will open the "Skills Network Toolbox".
- Click `OTHER`, then `Launch Application`. From there, you enter the port: `3060`
- Append `/api/gifts` to the end of the URL

Launch to see it running on backend. It should retrieve all the gifts as a JSON object.

Gifts Server

16. Copy the base URL in your browser (without the `/api/gifts` at the end of the URL) into a text editor to save it for later use. *You will need this URL to point to your server app in the React client.*

Set Up the Front-end Server

1. In the project home directory (`fullstack-capstone-project`) create a new directory named `giftwebsite`.

    ```
    mkdir giftwebsite
    ```

2. Change to the directory and run `npm init` to initialize a new Node.js project.

3. Run the following command to install `express` and `cors`.

```
npm install express cors
```

4. Create a new file named `index.js`.

5. Open `index.js` in the editor and paste the following code in it.

```
const express = require('express');
const path = require('path');
const app = express();
app.use(express.static(path.join(__dirname, 'build')));
app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'build', 'index.html'));
});
app.get('/app', function (req, res) {
  res.sendFile(path.join(__dirname, 'build', 'index.html'));
});
app.listen(9000);
```

6. Change to your front-end directory.

```
cd ../giftlink-frontend
```

7. Open the `.env` file in the editor. Edit the `REACT_APP_BACKEND_URL` to point to the server URL you copied into the text editor in the previous task.

8. Install all the necessary packages with `npm install`.

9. Edit the `scripts` in the package.json file so it includes the following immediately after `build`.

```
"postbuild": "cp -r build ../giftwebsite/",
```

10. Run the following command to build the client. This will build the client and add the client files to `giftwebsite` directory.

```
npm run build
```

Configure and Containerize the REACT Server

1. Go back to the `giftwebsite` directory. Create a `Dockerfile` for containerizing the client side application.

▼ Click here for sample

```
# Use Node.js LTS version
FROM node:18-slim
# Set working directory
WORKDIR /usr/src/app
# Copy package.json and package-lock.json files
COPY package*.json ./
# Install dependencies
RUN npm install --production
# Copy application code
COPY . .
# Expose port (if needed)
EXPOSE 9000
# Run the application
CMD ["node", "index.js"]
```

2. Export your SN Labs namespace and print it to the console using this command:

```
MY_NAMESPACE=$(ibmcloud cr namespaces | grep sn-labs-)
echo $MY_NAMESPACE
```

3. Perform a Docker build with the `Dockerfile` in the current directory.

```
docker build . -t us.icr.io/$MY_NAMESPACE/giftwebsite
```

4. Push the Docker image to the container by running the following command:

```
docker push us.icr.io/$MY_NAMESPACE/giftwebsite
```

5. Next, run the image in the container registry:

```
docker run -p 9000:9000 us.icr.io/$MY_NAMESPACE/giftwebsite
```

6. Select the `Gifts Website` button below or click the `Skills Network` button on the right to open the "Skills Network Toolbox".

- Select `OTHER`,

- Select `Launch Application`.
- Enter `9000` as the port
- Suffix the URL with `/home.html`
- Launch to see the running website.

Gifts Website

7. Access all the endpoints and test your website.

In the next part of this task, you will start the Code Engine and deploy your front end.

Start the Code Engine and Deploy the Front End

1. Start Code Engine by creating a project.

---

2. The Code Engine environment takes a while to prepare. You will see the progress status indicated in the set up panel.

---

3. Once the Code Engine set up is complete, you can see that it is active. Click on `Code Engine CLI` to begin the pre-configured CLI in the terminal below.

---

4. You will observe that the preconfigured CLI startup and the home directory is set to the current directory. As a part of the preconfiguration, the project has been set up and Kubeconfig is set up. The details are shown on the terminal.

---

5. Deploy the `giftwebsite` application on Code Engine.

```
ibmcloud ce application create --name giftwebsite --image us.icr.io/${SN_ICR_NAMESPACE}/giftwebsite --registry-secret icr-secret --port 9000
```

This will take a while.

6. Connect to the generated URL to access the website and check if the deployment is successful.

Submission

- Copy the publicly available URL for the application running in the Kubernetes cluster for submission.
- Copy the publicly available URL for the website running in the Code Engine for submission.
- Take the following screenshots with the deployed application.

IMPORTANT Ensure that the address bar is visible in all the screenshots and that the address bar is the same as the URL copied above and submitted.

1. Take a screenshot of the landing page and save it as `deployed_landingpage.png`.
2. Click on `Get Started` and go to the main page showing the gifts. Take a screeenshot and save it as `deployed_mainpage.png`.
3. Register a new user, by entering the details. Take a screeenshot before submission and save it as `registration.png`.
4. Show the home page with the logged in username on top. Take a screeenshot and save it as `deployed_loggedin.png`.
5. Click on any gift to see the details of the gift. Take a screeenshot and save it as `deployed_gift_detail.png`.
6. Click on any `Search` and search for `Older` gifts in `Living room` category. Take a screeenshot and save it as `deployed_search_gift.png`.

Author

Lavanya T S

Other Contributors

Upkar Lidder