Containerize your application

Estimated time needed: 60 mins

You have made good progress in your assignment thus far! Your Django application is fully functional, and your team is happy. However, your boss has a new request. The company is looking at using containers to manage and deploy the application. Furthermore, the management is interested in using the 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 are asked to look at Kubernetes. So, let's containerize your application now.

> **Note**: Before starting the lab, please click "Click to expand!" and follow the steps to check and delete previously persisting sessions to avoid any issues while running the lab.

▼ Click to expand!

- Please run the below command:

```
1
```

```
kubectl get deployments
```

Copied!

- If you see that the `dealership` deployment already exists, please delete it using:

```
1
```

```
kubectl delete deployment dealership
```

Copied!

- Please run the below command:

```
1
```

```
ibmcloud cr images
```

Copied!

- If there is any `dealership` image, please delete it using:

```
1
```

```
ibmcloud cr image-rm us.icr.io/<your sn labs namespace>/dealership:latest && docker rmi us.icr.io/<your sn labs namespace>/dealership:latest
```

Copied!

Please enter your SN Labs namespace in place of `<your sn labs namespace>`

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`)

- Please sign out of SN Labs and clear your browser cache and cookies.
- Please start the lab again and proceed as below.

Setup the environment

1. If the lab environment is reset, clone your git repo.
2. Open a new terminal, change to the `server/database` directory, and run the Mongo Express server as done in the previous week. Please refer to this documentation if required.
3. If the sentiment analyzer microservice deployed on Code Engine is unavailable, redeploy it and update the URL wherever required. Please refer to this documentation if required.
4. Open another new terminal. Change to the `server/frontend` directory. Build the front end by running the following commands.

```
1
2
```

```
npm install
npm run build
```

Copied! Executed!

Add Dockerfile

1. Create a `Dockerfile` in the `server` directory. The file should have the following steps listed:

    1. Add a base image.
    2. Add the requirements.txt file.
    3. Install and update Python.
    4. Change the working directory.
    5. Expose port.
    6. Run the command to start the application.

Here is an example file to get you started:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
FROM python:3.12.0-slim-bookworm

ENV PYTHONBUFFERED 1
ENV PYTHONWRITEBYTECODE 1

ENV APP=/app

# Change the workdir.
WORKDIR $APP

# Install the requirements
COPY requirements.txt $APP

RUN pip3 install -r requirements.txt

# Copy the rest of the files
COPY . $APP

EXPOSE 8000

RUN chmod +x /app/entrypoint.sh

ENTRYPOINT ["/bin/bash","/app/entrypoint.sh"]

CMD ["gunicorn", "--bind", ":8000", "--workers", "3", "djangoproj.wsgi"]
```

Copied!

Note: Please ensure that the contents of the Dockerfile are indented as above.

2. Notice that the second-to-last command in the Dockerfile refers to `entrypoint.sh`. Create this file in the `server` directory. This file should have the following content:

```
1
2
3
4
5
6
7
8
```

```
#!/bin/sh

# Make migrations and migrate the database.
echo "Making migrations and migrating the database. "
```

```
5. python manage.py makemigrations --noinput
6. python manage.py migrate --noinput
7. python manage.py collectstatic --noinput
8. exec "$@"
```

Copied!

Build and push image to container registry

You must recall how to 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.

1. Please export your SN Labs namespace and print it on the console as below:

```
1. 1
2. 2
```

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

Copied! Executed!

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

```
1. 1
```

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

Copied! Executed!

Next, push the image to the container registry:

```
1. 1
```

```
1. docker push us.icr.io/$MY_NAMESPACE/dealership
```

Copied!

Add deployment artifacts

Create a `deployment.yaml` file in the `server` directory to create the deployment and the service. It should look something like this:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
```

```
1. apiVersion: apps/v1
2. kind: Deployment
3. metadata:
4.   labels:
5.     run: dealership
6.   name: dealership
7. spec:
8.   replicas: 1
9.   selector:
10.    matchLabels:
11.      run: dealership
12.  strategy:
13.    rollingUpdate:
```

```
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: dealership
    spec:
      containers:
      - image: us.icr.io/your-name-space/dealership:latest
        imagePullPolicy: Always
        name: dealership
        ports:
        - containerPort: 8000
          protocol: TCP
      restartPolicy: Always
```

Copied!

Please enter your SN Labs namespace in place of `your-name-space` in the above file.

Deploy the application

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

```
1
```

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

Copied!

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

```
1
```

```
kubectl port-forward deployment.apps/dealership 8000:8000
```

Copied!

> Note: If you see any errors, please wait for some time and run the command again.

Click `Dealership Application` button below or click the `Skills Network` button on the right; it will open the "Skills Network Toolbox". Then click `OTHER`, then `Launch Application`. From there, you should be able to enter the port as `8000` and launch to see the running application, log in, view the dealers, review the dealers, and logout.

Dealership Application

Submission

1. Copy the publicly available URL for the application running in the lab Kubernetes cluster for submission.
2. Please take the following screenshots with the deployed application. 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 homepage with the login screen and save it as `deployed_landingpage.png`.
    2. Login with a credential you created earlier. This would also be copied into the deployment as it is an SQLite file. Take a screenshot of the homepage with the logged in user screen and save it as `deployed_loggedin.png`.
    3. Click on any dealer to show the dealer details along with reviews. Take a screenshot and save it as `deployed_dealer_detail.png`.
    4. Add a review for any one of the dealers. Take a screenshot showing the review and the sentiment added to the details page, and save it as `deployed_add_review.png`.

Author(s)

Lavanya T S

Other Contributor(s)

Upkar Lidder

Priya