Containerize your application

Estimated time needed: 60 mins

You have made good progress in your assignment thus far! Your Django application is running on IBM Cloud, and your team is happy. However, your boss has a new ask. 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 on "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!

Add Dockerfile

Create a `Dockerfile` in the root directory. The file will have the following steps listed:

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

Here is an example file to get you started:

```
 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
```

```
1.     FROM python:3.8.2
2.
3.     ENV PYTHONBUFFERED 1
4.     ENV PYTHONWRITEBYTECODE 1
5.
6.     RUN apt-get update \
7.         && apt-get install -y netcat
8.
9.     ENV APP=/app
10.
11.    # Change the workdir.
12.    WORKDIR $APP
13.
14.    # Install the requirements
15.    COPY requirements.txt $APP
16.
17.    RUN pip3 install -r requirements.txt
18.
19.    # Copy the rest of the files
20.    COPY . $APP
21.
22.    EXPOSE 8000
23.
24.    RUN chmod +x /app/entrypoint.sh
25.
26.    ENTRYPOINT ["/bin/bash","/app/entrypoint.sh"]
27.
28.    CMD ["gunicorn", "--bind", ":8000", "--workers", "3", "djangobackend.wsgi"]
```

Copied!

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

Notice that the the second to last command in Dockerfile refers to `entrypoint.sh`. This file should have the following content:

```
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
```

```
1.     #!/bin/sh
2.
3.     if [ "$DATABASE" = "postgres" ]; then
```

```
        echo "Waiting for postgres..."

        while ! nc -z $DATABASE_HOST $DATABASE_PORT; do
        sleep 0.1
        done

        echo "PostgreSQL started"
    fi

    # Make migrations and migrate the database.
    echo "Making migrations and migrating the database. "
    python manage.py makemigrations main --noinput
    python manage.py migrate --noinput
    exec "$@"
```

Copied!

Please use the below command to make `entrypoint.sh` executable.

```
1
```

```
chmod +x ./entrypoint.sh
```

Copied!

Push built image to container registry

If you remember from the previous course in this certification, you were asked to build your image and push to IBM Cloud Image Registry (ICR). You need to do the same here and then refer to this image in your Kubernetes deployment file.

Please export your SN labs namespace and print it on the console, as below:

```
1
2
```

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

Copied!

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

```
1
```

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

Copied!

Next, push the image to the container registry:

```
1
```

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

Copied!

Add deployment artifacts

Create `deployment.yaml` file to create the deployment and the service. It should look something like:

```
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
26
27
28
29
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: dealership
  name: dealership
spec:
  replicas: 1
  selector:
    matchLabels:
      run: dealership
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: dealership
    spec:
      containers:
      - image: us.icr.io/$MY_NAMESPACE/dealership:latest
        imagePullPolicy: Always
        name: dealership
        ports:
        - containerPort: 8000
          protocol: TCP
      restartPolicy: Always
```

Copied!

Please enter your SN labs namespace in place of `$MY_NAMESPACE` in the above file.

Deploy the application

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

```
1. 1

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

Copied!

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

```
1. 1

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

Copied!

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

Clicki on 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.

You will get an error from the home page. Add `/djangoapp` at the end of the URL to see your application.

Submission

Submit the publicly available URL for the application running in the lab Kubernetes cluster.

Author(s)

Upkar Lidder

Other Contributor(s)

Upkar Lidder

Priya

Changelog

Date	Version	Changed by	Change Description
2021-01-28	1.0	Upkar Lidder	Created new instructions for Capstone project
2022-05-05	1.1	K Sundararajan	Updated 'port-forward' command
2022-05-24	1.2	K Sundararajan	Updated Lab instructions
2022-09-01	1.3	K Sundararajan	Updated `Launch Application` instructions as per the new Theia IDE