Final Project

Objectives

In this lab, you will:

- Build and deploy a simple Guestbook application
- Autoscale the Guestbook application using Horizontal Pod Autoscaler
- Perform Rolling Updates and Rollbacks

Project Overview

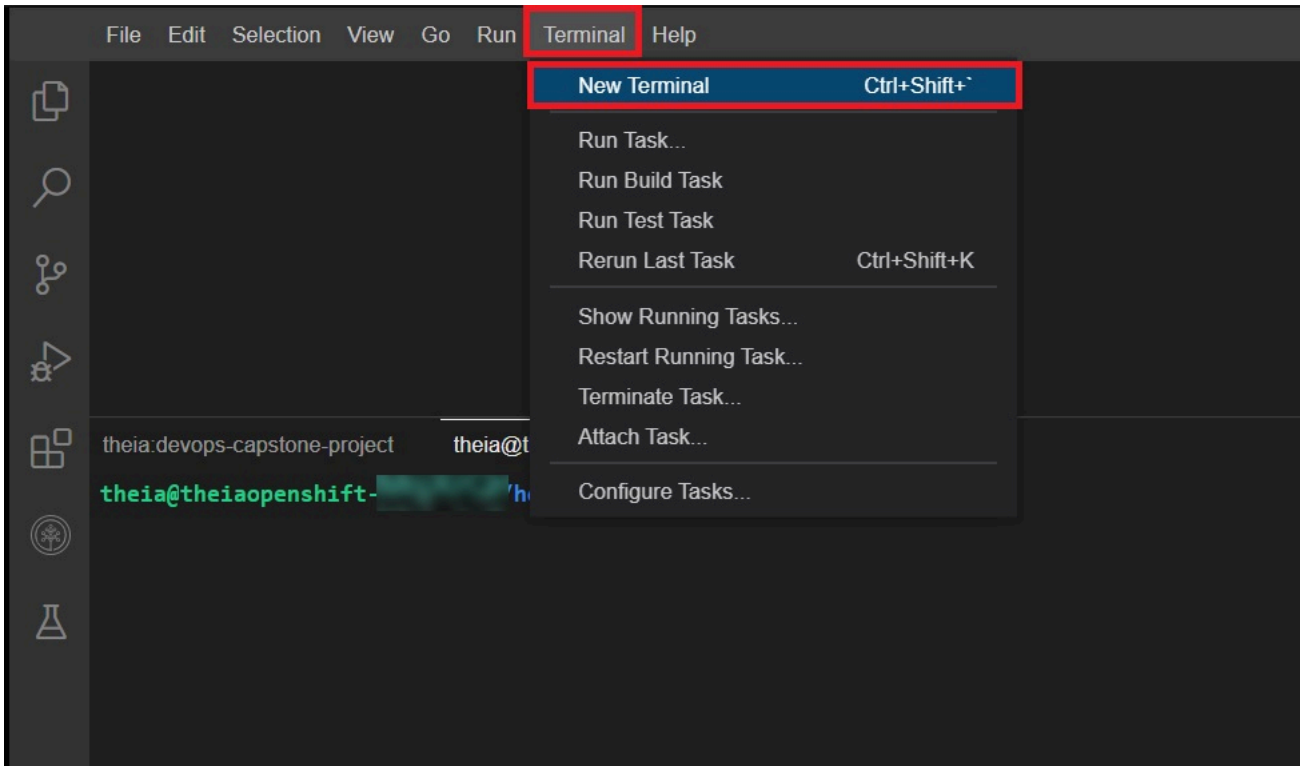
Guestbook application

Guestbook is a simple web application that we will build and deploy with Docker and Kubernetes. The application consists of a web front end which will have a text input where you can enter any text and submit. For all of these we will create Kubernetes Deployments and Pods. Then we will apply Horizontal Pod Scaling to the Guestbook application and finally work on Rolling Updates and Rollbacks.

Verify the environment and command line tools

1. If a terminal is not already open, open a terminal window by using the menu in the editor: `Terminal > New Terminal`.

   Note: Please wait for some time for the terminal prompt to appear.



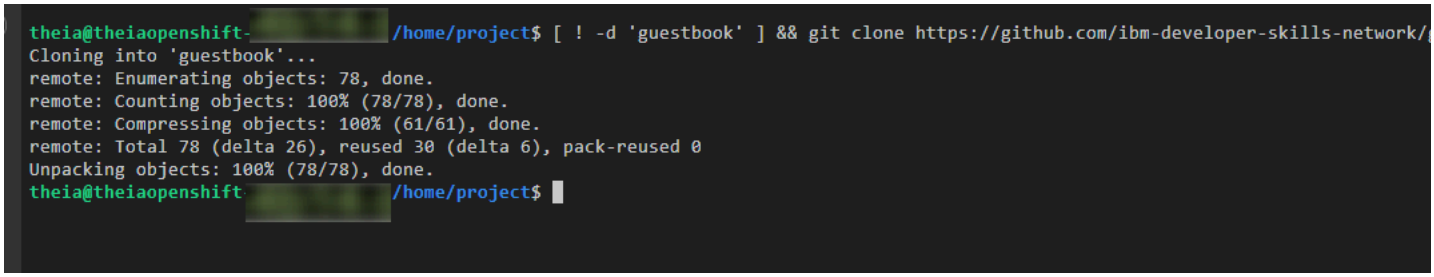
2. Change to your project folder.

   Note: If you are already on the `/home/project` folder, please skip this step.

```
cd /home/project
```

3. Clone the git repository that contains the artifacts needed for this lab.

```
[ ! -d 'guestbook' ] && git clone https://github.com/ibm-developer-skills-network/guestbook
```



4. Change to the directory for this lab.

   ```
   cd guestbook
   ```

5. List the contents of this directory to see the artifacts for this lab.

   ```
   ls
   ```

Build the guestbook app

To begin, we will build and deploy the web front end for the guestbook app.

1. Change to the `v1/guestbook` directory.

   ```
   cd v1/guestbook
   ```

2. Dockerfile incorporates a more advanced strategy called multi-stage builds, so feel free to read more about that here.

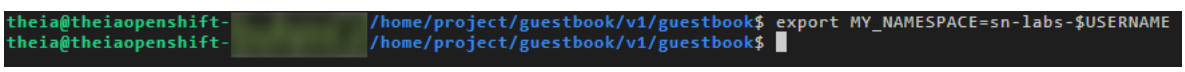
Complete the Dockerfile with the necessary Docker commands to build and push your image. The path to this file is `guestbook/v1/guestbook/Dockerfile`.

▶ Hint!

   Take a screenshot of the completed dockerfile and save it as **Dockerfile.png**.

3. Export your namespace as an environment variable so that it can be used in subsequent commands.

   ```
   export MY_NAMESPACE=sn-labs-$USERNAME
   ```



4. Build the guestbook app using the Docker Build command.

▶ Hint!

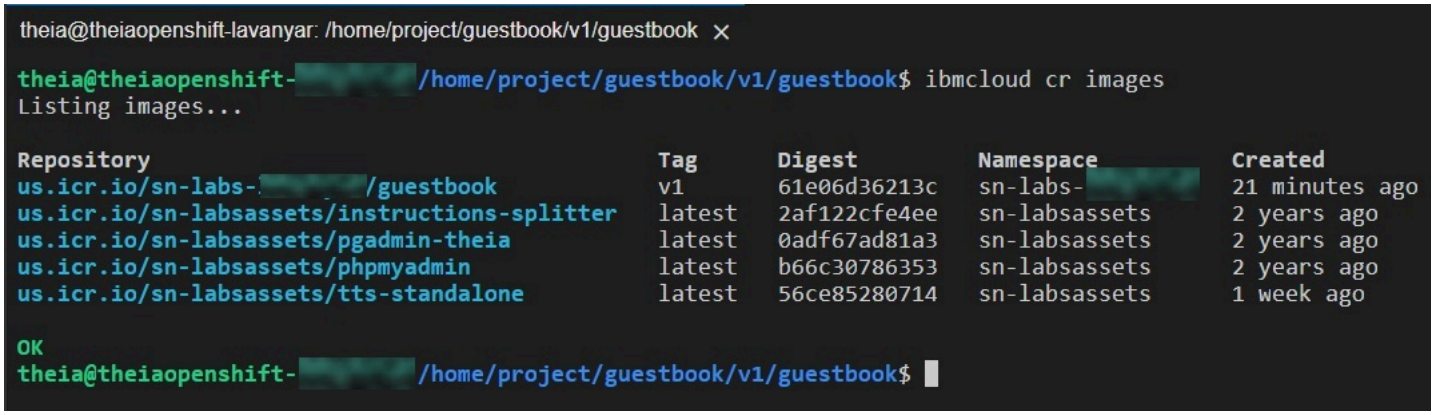
5. Push the image to IBM Cloud Container Registry.

▶ Hint!

Note: If you have tried this lab earlier, there might be a possibility that the previous session is still persistent. In such a case, you will see a **'Layer already Exists'** message instead of the **'Pushed'** message in the above output. We recommend you to proceed with the next steps of the lab.

6. Verify that the image was pushed successfully.

```
ibmcloud cr images
```



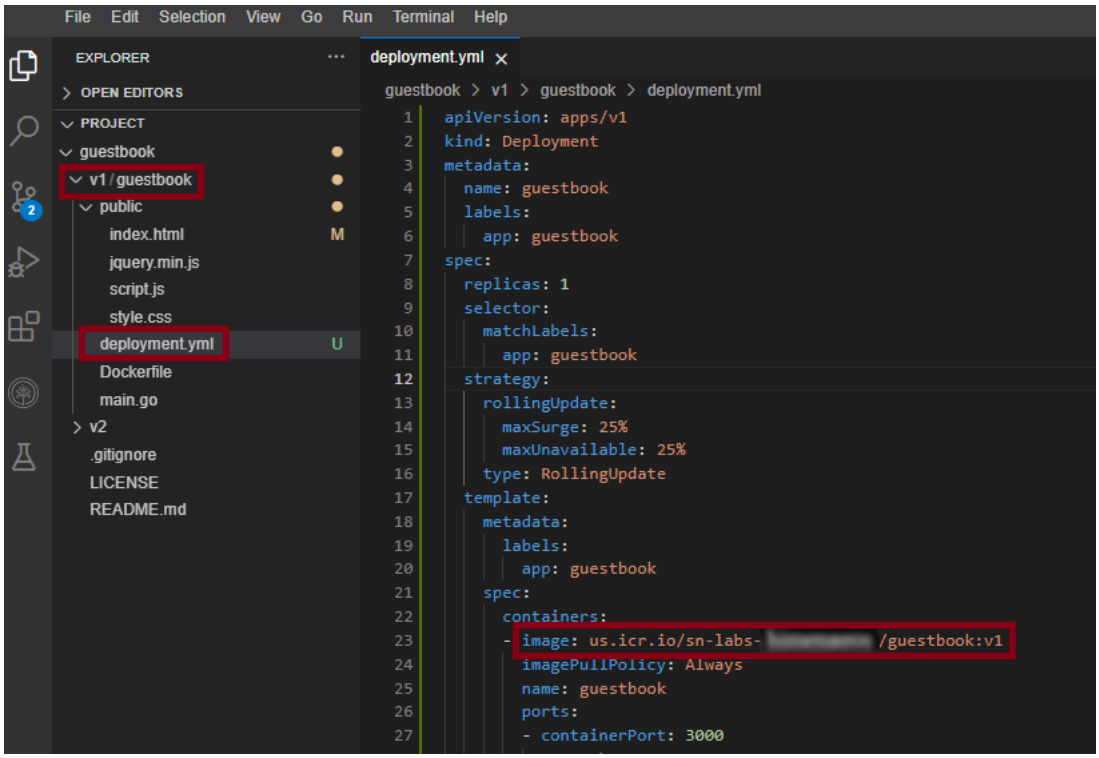
Take a screenshot of the output of Step 6 and save it as a .jpg or .png with the filename `crimages.png`. You will be prompted to upload the screenshot in the Peer Assignment.

7. Open the `deployment.yml` file in the `v1/guestbook` directory & view the code for the deployment of the application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  replicas: 1
  selector:
    matchLabels:
      app: guestbook
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: guestbook
    spec:
      containers:
      - image: us.icr.io/<your sn labs namespace>/guestbook:v1
        imagePullPolicy: Always
        name: guestbook
        ports:
        - containerPort: 3000
          name: http
        resources:
          limits:
            cpu: 50m
          requests:
            cpu: 20m
```

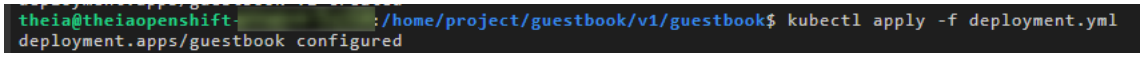
Note: Replace `<your sn labs namespace>` with your SN labs namespace. To check your SN labs namespace, please run the command `ibmcloud cr namespaces`

- It should look as below:



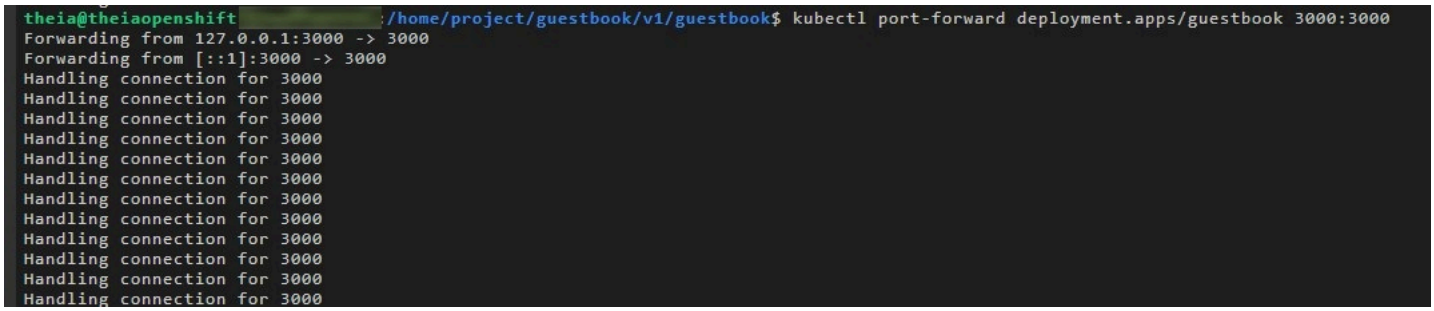
8. Apply the deployment using:

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

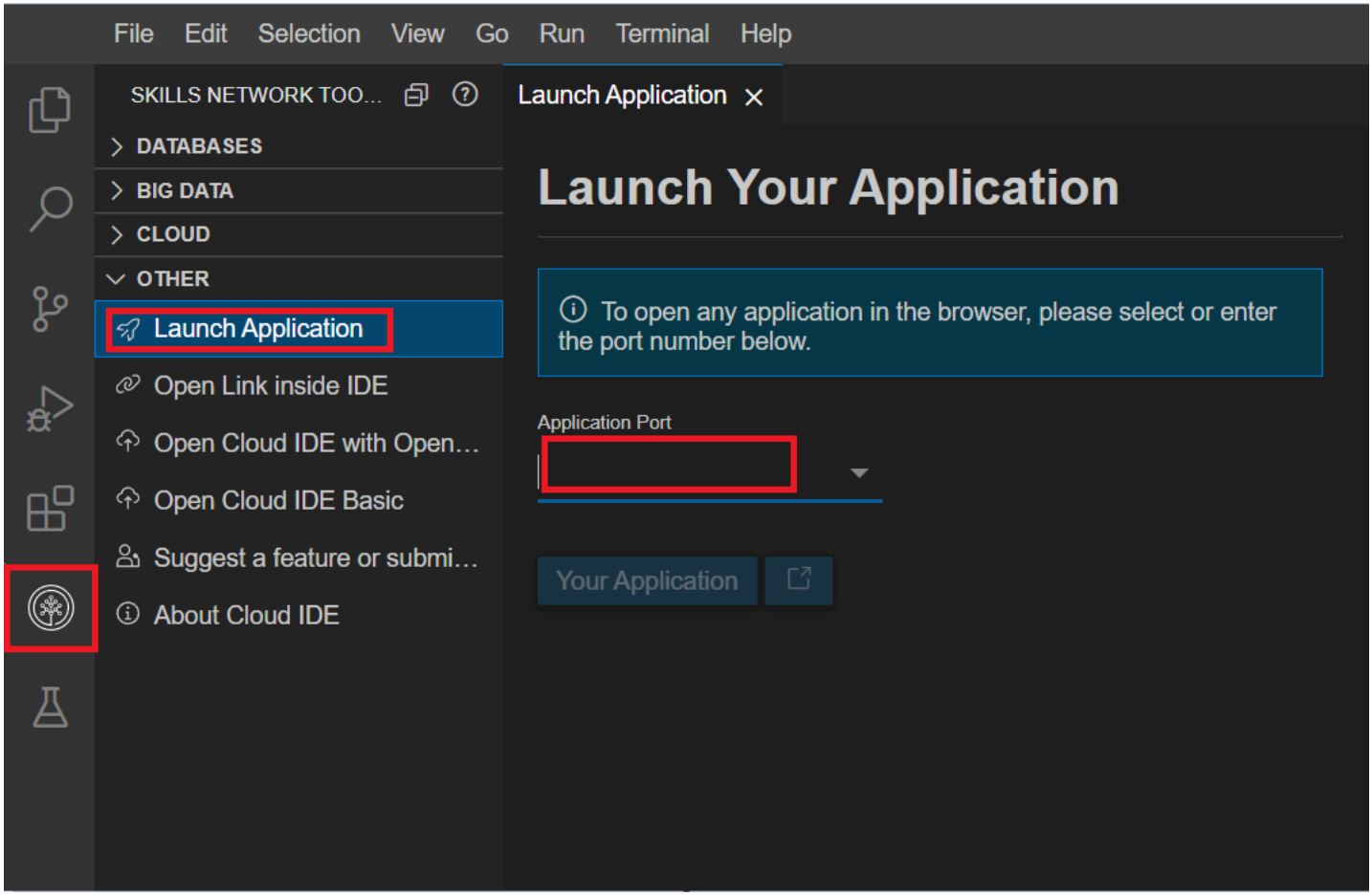


9. Open a New Terminal and enter the below command to view your application:

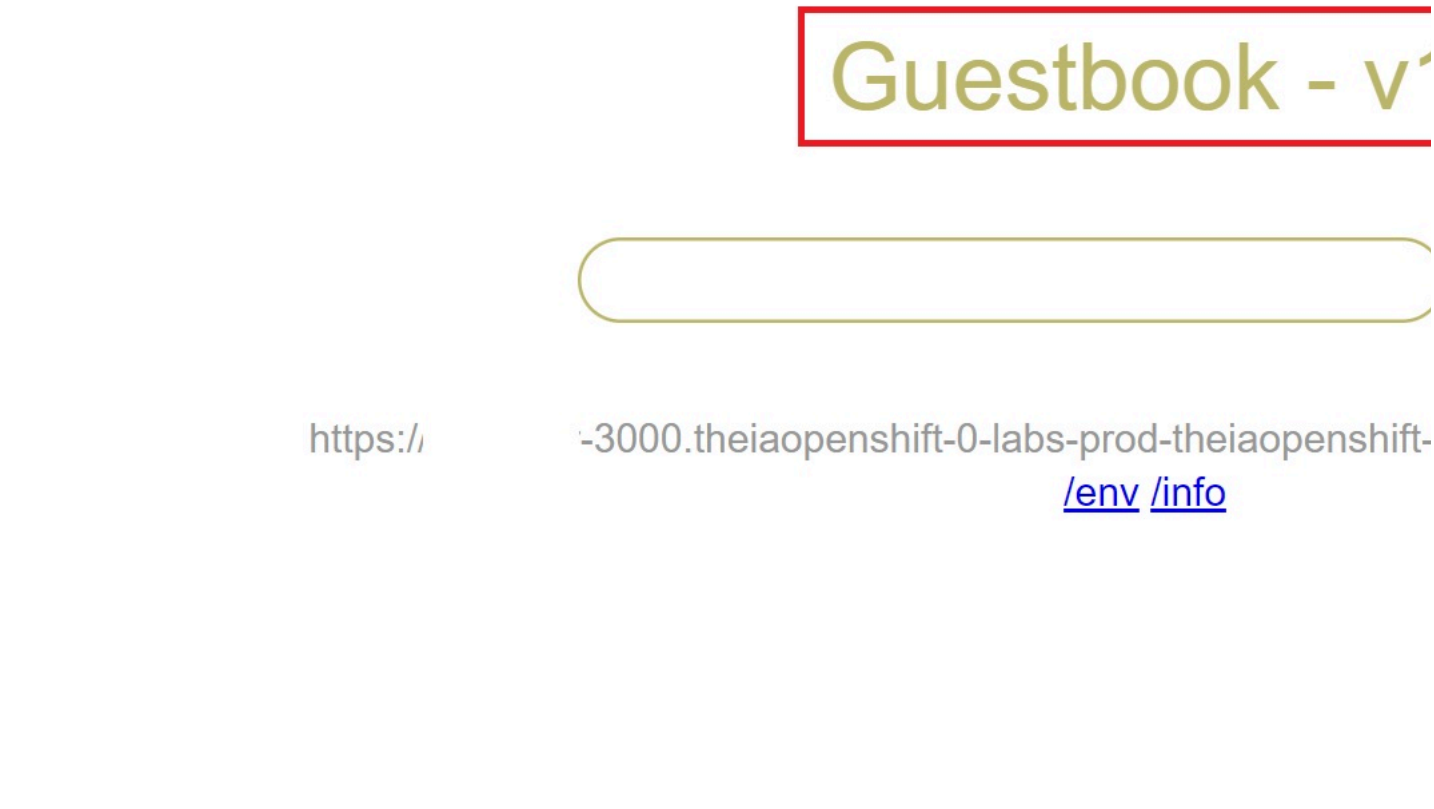
```
kubectl port-forward deployment.apps/guestbook 3000:3000
```



10. Launch your application on port 3000. Click on the Skills Network button on the right, it will open the **"Skills Network Toolbox"**. Then click the **Other** then **Launch Application**. From there you should be able to enter the port and launch.



11. Now you should be able to see your running application. Please copy the app URL which will be given as below:



Take a screenshot of your deployed application and save it as a .jpg or .png with the filename `app.png`. You will be prompted to upload the screenshot in the Peer Assignment.

12. Try out the guestbook by putting in a few entries. You should see them appear above the input box after you hit **Submit**.

Autoscale the Guestbook application using Horizontal Pod Autoscaler

1. Autoscale the Guestbook deployment using `kubectl autoscale deployment`

▶ Hint!

2. You can check the current status of the newly-made HorizontalPodAutoscaler, by running:

```
kubectl get hpa guestbook
```

The current replicas is 0 as there is no load on the server.

Take a screenshot of your Horizontal Pod Autoscaler and save it as a .jpg or .png with the filename `hpa.png`. You will be prompted to upload the screenshot in the Peer Assignment.

3. Open another new terminal and enter the below command to generate load on the app to observe the autoscaling (Please ensure your port-forward command is running. In case you have stopped your application, please run the port-forward command to re-run the application at port 3000.)

```
kubectl run -i --tty load-generator --rm --image=busybox:1.36.0 --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- <your app URL>;
```

- Please replace your app URL in the `<your app URL>` part of the above command.

  Note: Use the same copied URL which you obtained in step 11 of the previous task.

The command will be as below:

```
theia@theiaopenshift:/home/project$ kubectl run -i --tty load-generator-7 --rm --image=busybox:1.28 --restart=Never -- /
0.theiaopenshift-2-labs-prod-theiaopenshift-4-tor01.proxy.cognitiveclass.ai/; done"
```

Note: In case you get a `Load generator already exists` error, please suffix a number after `load-generator` eg. `load-generator-1`, `load-generator-2`.

- You will keep getting an output similar as below which will indicate the increasing load on the app:

```
    </div>

    <div id="guestbook-entries">
      <link href="https://afeld.github.io/emoji-css/emoji.css" rel="stylesheet">
      <p>Waiting for database connection... <i class='em em-boat'></i></p>

    </div>

    <div>
      <form id="guestbook-form">
        <input autocomplete="off" id="guestbook-entry-content" type="text">
        <a href="#" id="guestbook-submit">Submit</a>
      </form>
    </div>

    <div>
      <p><h2 id="guestbook-host-address"></h2></p>
      <p><a href="env">/env</a>
      <a href="info">/info</a></p>
    </div>
    <script src="jquery.min.js"></script>
    <script src="script.js"></script>
  </body>
</html>
```

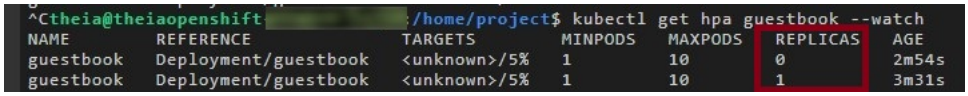
Note: Continue further commands in the 1st terminal

4. Run the below command to observe the replicas increase in accordance with the autoscaling:

```
kubectl get hpa guestbook --watch
```

```
theia@theiaopenshift:/home/project$ kubectl get hpa guestbook --watch
NAME        REFERENCE              TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
guestbook   Deployment/guestbook   <unknown>/5%    1         10        0          51s
```

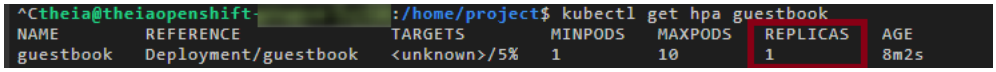
5. Run the above command again after 5-10 minutes and you will see an increase in the number of replicas which shows that your application has been autoscaled.



Take a screenshot of your Autoscaler details and save it as a .jpg or .png with the filename `hpa2.png`. You will be prompted to upload the screenshot in the Peer Assignment.

6. Run the below command to observe the details of the horizontal pod autoscaler:

```
kubectl get hpa guestbook
```



- Please close the other terminals where load generator and port-forward commands are running.

Perform Rolling Updates and Rollbacks on the Guestbook application

Note: Please run all the commands in the 1st terminal unless mentioned to use a new terminal.

1. Please update the title and header in `index.html` to any other suitable title and header like **<Your name> Guestbook - v2** & **Guestbook - v2**.

▶ Hint!

2. Run the below command to build and push your updated app image:

▶ Hint!

Take a screenshot of your updated image and save it as a .jpg or .png with the filename `upguestbook.png`. You will be prompted to upload the screenshot in the Peer Assignment.

3. Update the values of the CPU in the `deployment.yml` to **cpu: 5m** and **cpu: 2m** as below:

▶ Hint!

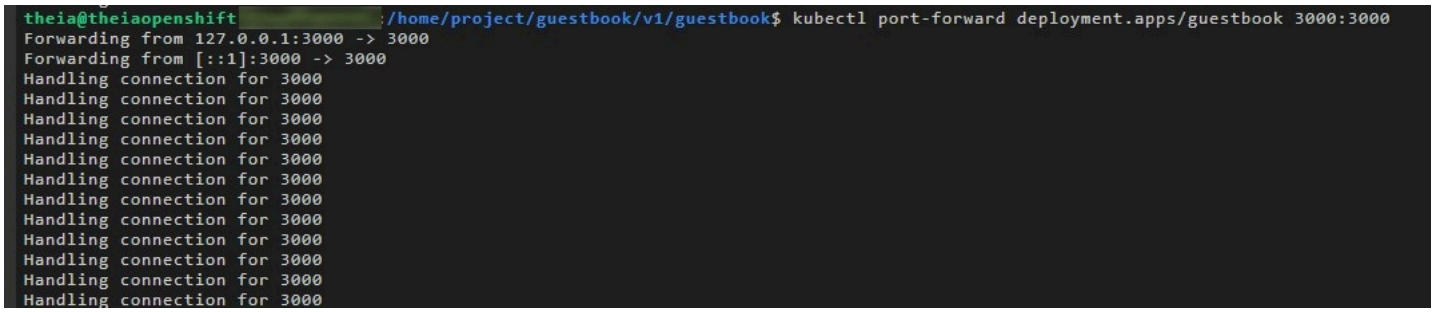
4. Apply the changes to the `deployment.yml` file.

▶ Hint!

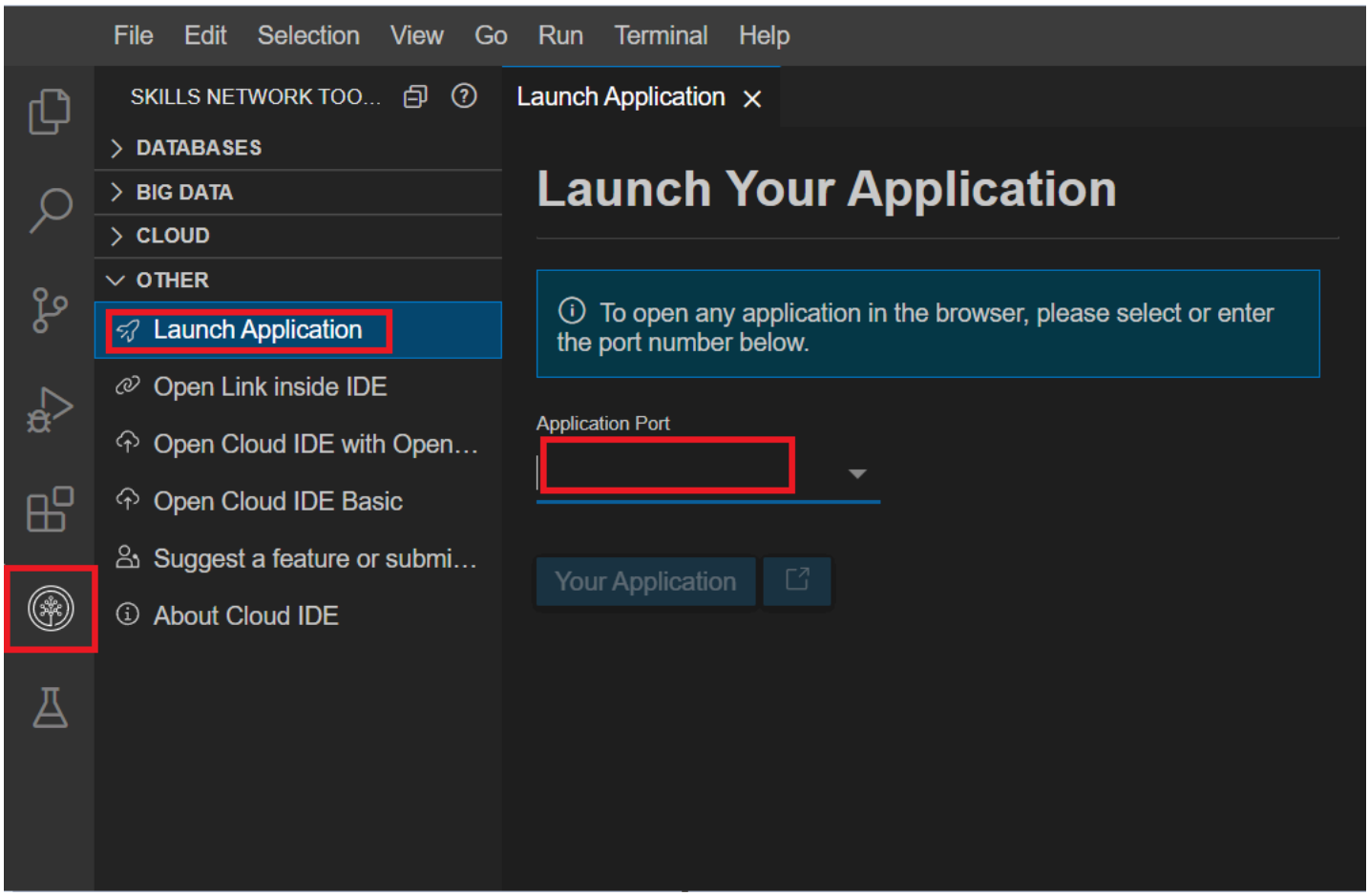
Take a screenshot of the details of the output of Step 4 and save it as a .jpg or .png with the filename `deployment.png`. You will be prompted to upload the screenshot in the Peer Assignment.

5. Run the port-forward command again to start the app:

```
kubectl port-forward deployment.apps/guestbook 3000:3000
```



6. Launch your application on port 3000. Click on the Skills Network button on the right, it will open the **"Skills Network Toolbox"**. Then click the **Other** then **Launch Application**. From there you should be able to enter the port and launch.



7. You will notice the updated app content as below:

Take a screenshot of your updated application and save it as a .jpg or .png with the filename `up-app.png`. You will be prompted to upload the screenshot in the Peer Assignment.

Note: Please stop the application before running the next steps.

8. Run the below command to see the history of deployment rollouts:

```
kubectl rollout history deployment/guestbook
```

```
theia@theiaopenshift-                /home/project/guestbook/v1/guestbook$ kubectl rollout history deployment/guestbook
deployment.apps/guestbook
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

9. Run the below command to see the details of Revision of the deployment rollout:

```
kubectl rollout history deployments guestbook --revision=2
```

```
theia@theiaopenshift-                /home/project/guestbook/v1/guestbook$ kubectl rollout history deployments guestbook --revision=2
deployment.apps/guestbook with revision #2
Pod Template:
  Labels:       app=guestbook
        pod-template-hash=6bdb7c74c6
  Containers:
   guestbook:
    Image:      us.icr.io/sn-labs-                /guestbook:v1
    Port:       3000/TCP
    Host Port:  0/TCP
    Limits:
      cpu:      5m
    Requests:
      cpu:      2m
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

Take a screenshot of the details of the correct Revision and save it as a .jpg or .png with the filename `rev.png`. You will be prompted to upload the screenshot in the Peer Assignment.

10. Run the below command to get the replica sets and observe the deployment which is being used now:

```
kubectl get rs
```

```
theia@theiaopenshift-                /home/project/guestbook/v1/guestbook$ kubectl get rs
NAME                   DESIRED   CURRENT   READY   AGE
guestbook-5997448ccb   0         0         0       17m
guestbook-6bdb7c74c6   1         1         1       105s
```

11. Run the below command to undo the deploymnent and set it to Revision 1:

```
kubectl rollout undo deployment/guestbook --to-revision=1
```

```
theia@theiaopenshift                 :/home/project/guestbook/v1/guestbook$ kubectl rollout undo deployment/guestbook --to-revision=1
deployment.apps/guestbook rolled back
```

12. Run the below command to get the replica sets after the Rollout has been undone. The deployment being used would have changed as below:

```
kubectl get rs
```

```
theia@theiaopenshift                 /home/project/guestbook/v1/guestbook$ kubectl get rs
NAME                   DESIRED   CURRENT   READY   AGE
guestbook-5997448ccb   1         1         1       19m
guestbook-6bdb7c74c6   0         0         0       3m40s
```

Take a screenshot of the output of Step 9 and save it as a .jpg or .png with the filename `rs.png`. You will be prompted to upload the screenshot in the Peer Assignment.

Congratulations! You have completed the final project for this course. Do not log out of the lab environment (you can close the browser though) or delete any of the artifacts created during the lab, as these will be needed for the next lab,`Optional: Deploy Guestbook App from the OpenShift Internal Registry`.