
Exercise 6. Deploying an application on Kubernetes

Estimated time

01:30

Overview

In this exercise, you build a containerized application and deploy it to IBM Cloud Kubernetes Service.

Objectives

After completing this exercise, you should be able to:

- Create a containerized Node.js application and build it on IBM Cloud Container Registry.
- Explain how the container security analysis capability of Vulnerability Advisor can identify the security vulnerabilities by scanning an image.
- Create a deployment and scale it.
- Expose your application on the internet.

Introduction

In this exercise, you create configuration files to deploy an application to the IBM Cloud Kubernetes Service.

To make your app more resilient, you can determine the number of instances of the app in your deployment and let Kubernetes automatically create a replica set for you. If one of the pods becomes unresponsive, the pod is re-created automatically.

Also, you expose your app to a port that can be accessed through all your worker nodes public IPs.

Requirements

- The Kubernetes CLI must be installed.
- The IBM Cloud Dev plug-in must be installed.
- The IBM Cloud Kubernetes Service plug-in must be installed.
- The IBM Cloud Container Registry plug-in must be installed.
- Successfully complete Exercise 5.

Exercise instructions

In this exercise, you complete the following tasks:

- ___ 1. Log in to your IBM Cloud account, connect to your cluster, and view the number of worker nodes.
- ___ 2. Create a container and build it on IBM Cloud Container Registry.
- ___ 3. Create a deployment and scale it.
- ___ 4. Expose the app to the internet
- ___ 5. Clean up the environment

Part 1: Logging in to your IBM Cloud account, connecting to your cluster, and viewing the number of worker nodes

You completed these steps in Exercise 5. If you did not completed Exercise 5 yet, you must complete it before you continue with this exercise.

Part 2: Creating a container and building it on IBM Cloud Container Registry

IBM Cloud offers many starter kits to help you get started with coding quickly. In this tutorial, you complete the steps to create a Node.js application and then build it into IBM Cloud Container Registry.

Also, you can review the Vulnerability Advisor report for details about any vulnerable packages, insecure containers, or app settings.

- ___ 1. Target a region by running the following command

```
ibmcloud target -r <region>
```



Note

A region is a specific geographical location where you can deploy apps, services, and other IBM Cloud resources. IBM Cloud regions differ from IBM Cloud Kubernetes Service regions.

You must choose the same region in which your organization and space are.

You can find where your space is by running the following command:

```
ibmcloud account orgs
```

The following screen shows the output of the command.

```
~/Box Sync $ ibmcloud account orgs
Getting orgs in all regions as darkmong0111@gmail.com...
Retrieving current account...
OK
```

Name	Region	Account owner	Account ID	Status
darkmong0111@gmail.com	us-south	darkmong0111@gmail.com	12061a956fe34fa8ab89939b905e7f40	active

1+1=2 Example

```
ibmcloud target -r us-south
```

-
- __ 2. To target an organization and a space, run the following command:

```
ibmcloud target --cf
```



Note

Targeting an organization and a space is required for Cloud Foundry services and also to use Cloud App Service starter kits.

You can interactively choose the org and space by running the `ibmcloud target --cf` command. If you have only one space, the org and space are automatically selected by the command.

-
- __ 3. To create a starter app from scratch, create a Node.js simple web application by using the Cloud App Service starter kit. The starter application prints a simple text on the browser.



Note

IBM Cloud App Service starter kits are pre-configured and integrated.

A starter kit sets up a starter application so that you can get started quickly with creating a cloud native application. Each starter app is generated with configurations to connect to the added services, files for deploying to Cloud Foundry, Kubernetes or a DevOps pipeline. It includes capabilities to monitor the health of your application. They are also generated to reflect the architecture pattern that you choose.

Complete the following steps to create your app:

- __ a. At the Command Prompt, change to the directory that you want to use for your app. Your app will be saved into this current directory.
- __ b. Run the `ibmcloud dev create` command.
- __ c. Select 1. Blank App, which creates an application with no predefined architecture pattern. It is just a simple application.
- __ d. Select 4. Node to create a Node.js application.

Enter a name for your app. For example, `hello-student`.

- __ e. Type `n` to the question “Do you want to select a service to add to this application?”

You can bind IBM Cloud services to the application in this step, but for this exercise, you do not need to do so.

- __ f. Select 4. No DevOps, with manual deployment. If you select another option, it creates a DevOps Pipeline to deploy to Cloud Foundry or IBM Cloud Kubernetes Service. In this exercise, you deploy the application manually.

```

=====
Select from the following DevOps toolchain and target runtime environment
options:

```

1. IBM DevOps, deploy to Knative-based Kubernetes containers
2. IBM DevOps, deploy to Helm-based Kubernetes containers
3. IBM DevOps, deploy to Cloud Foundry buildpacks
4. No DevOps, with manual deployment

```

=====
? Enter selection number:> 4_

```

- ___ 4. Go to the created application by running `cd hello-student`.
- ___ 5. Edit the `index.js` file in the sample application by opening `/public/index.js`. Go to line 11, and replace `Congratulations! By Welcome!` and then save the file.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>IBM Cloud Web Starter</title>
5    <style>#flex-header,body,main ul{-webkit-box-direction:normal}body{background-color:#fff;font-family:IBM Plex Sans;display:-webk
6    <meta name="viewport" content="width=device-width, initial-scale=1">
7  </head>
8  <body>
9    <header id="flex-header">
10     <div class="cloud-header"></div>
11     <h1>Welcome!</h1>
12     <h2>You are currently running a Node.js app built for the IBM Cloud.</h2>
13   </header>
14   <main>
15     <ul>
16       <li>
17         <div class="right-arrow"></div>
18         <div>
19           <a target="_blank" rel="noopener" href="https://cloud.ibm.com/developer/appservice/dashboard?env_id=ibm33Ayp13Aus-
20         </div>
21       </li>
22       <li>
23         <div class="right-arrow"></div>
24         <div>
25           <a target="_blank" rel="noopener" href="https://slack-invite-ibm-cloud-tech.mybluemix.net/">Ask questions on Slack
26         </div>
27       </li>
28       <li>
29         <div class="right-arrow"></div>
30         <div>
31           <a target="_blank" rel="noopener" href="https://www.ibm.com/cloud/cli">Install IBM Cloud Developer Tools</a>
32         </div>
33       </li>
34     </ul>

```

- ___ 6. Create a namespace in the IBM Cloud Container Registry by running the following command:

```
ibmcloud cr namespace-add <namespace name>
```

1+1=2 Example

```
ibmcloud cr namespace-add cr-studentxx
```

The name space in the registry must be unique. Replace “xx” in the example by your initials or a unique identifier.

- ___ 7. Build an image by using the Dockerfile in the IBM Cloud Container Registry and give it a tag. A Dockerfile is a text document that contains all the commands that a user might call from the command line to assemble an image. Run the following command:

```
ibmcloud cr build -t <region domain>/<namespace>/<application name>:<tag>
<directory>
```

Example

```
ibmcloud cr build -t us.icr.io/cr-studentxx/hello-student:v1 .
```

Do not miss “.” at the end. It indicates that the current directory is the location of your build context, which contains your Dockerfile and prerequisite files.

The following table shows the supported IBM Cloud Container Registry local regions.

IBM Cloud Container Registry region	Domain name
ap-north	jp.icr.io
ap-south	au.icr.io
eu-central	de.icr.io
uk-south	uk.icr.io
us-south	us.icr.io



Information

The Dockerfile must be defined to build an image. When you create the sample application by running the `ibmcloud dev create` command, a Dockerfile is created automatically. Look at the specifics of your Dockerfile, which is shown in the following figure.

```

1  FROM node:8-stretch
2
3  # Change working directory
4  WORKDIR "/app"
5
6  # Update packages and install dependency packages for services
7  RUN apt-get update \
8    && apt-get dist-upgrade -y \
9    && apt-get clean \
10   && echo 'Finished installing dependencies'
11
12 # Install npm production packages
13 COPY package.json /app/
14 RUN cd /app; npm install --production
15
16 COPY . /app
17
18 ENV NODE_ENV production
19 ENV PORT 3000
20
21 EXPOSE 3000
22
23 CMD ["npm", "start"]

```

- Line 1 `FROM node:8-stretch`: This line determines the base image on which you base your image. In this example, you are creating an image from the Node.js 8-stretch version of the image.
- Line 3 `#Change working directory`: This line is a comment. You can add comments to the Dockerfile by adding `#` at the beginning of the line.
- Line 4 `WORKDIR "/app"`: This line specifies where to place the application code inside the image, which becomes the working directory for your application.
- Lines 7 – 10: The `RUN` command installs your application and its required packages. In this example, it updates the OS image and prints the `'Finished installing dependencies'` message.
- Line 13: The `COPY` command copies the files or directories from the host to the Docker image.
- Line 14: This `RUN` command installs an NPM package of the application inside the image.

- Line 16: The **COPY** command copies the files from the local source to the image. In this example, the files in the current directory are copied to the location `/app` within the image.
- Lines 18 – 19: The **ENV** instruction sets the environment variable and the value is in the environment of all “descendant” Dockerfile commands.
- Line 21: The **EXPOSE** instruction is used to specify the network port on which the Docker container listens at run time.

The **EXPOSE** instruction does not publish the port. It functions as a type of documentation between the person who builds the image and the person who runs the container about which ports are intended to be published. To publish the port when running the container, use the `-p` flag on Docker to publish and map one or more ports, or the `-P` flag to publish all available ports and map them to high-order ports.

- Line 23: You can use **CMD** to specify the command to run when the Docker container starts. In this case, “`npm start`” runs when the container starts.

-
- ___ 8. Verify that the image is created. The Vulnerability Advisor is automatically run and indicates the issues with the image as shown in the SECURITY STATUS column.

Run the following command:

```
ibmcloud cr images
```

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student $ ibmcloud cr images
Listing images...

REPOSITORY                                TAG    DIGEST          NAMESPACE    CREATED          SIZE    SECURITY STATUS
us.icr.io/cr-student/hello-student      v1     5330d20f987e    cr-student   26 minutes ago  431 MB  3 Issues
OK
```



Troubleshooting

If you see issues in the SECURITY STATUS column, to learn more details about these issues, run the following command:

```
ibmcloud cr va <region domain>/<namespace>/<application name>:<tag>
```

Example: `ibmcloud cr va us.icr.io/cr-studentxx/hello-student:v1`


```

~/Box_Sync/data_2019/RESIDENCY/Ex6/hello-student $ ibmcloud cr wa us.icr.io/cr-student/hello-student:v1
Checking security issues for 'us.icr.io/cr-student/hello-student:v1'...

Image 'us.icr.io/cr-student/hello-student:v1' was last scanned on Sat May 4 05:57:47 UTC 2019
The scan results show that 3 ISSUES were found for the image.

Configuration Issues Found

```

Configuration Issue ID	Policy Status	Security Practice	How to Resolve
application_configuration:mysql.ssl-ca	Active	A setting in /etc/mysql/my.cnf that specifies the Certificate Authority (CA) certificate.	ssl-ca is not specified in /etc/mysql/my.cnf
application_configuration:mysql.ssl-cert	Active	A setting in /etc/mysql/my.cnf that specifies the server public key certificate. This certificate can be sent to the client and authenticated against its CA certificate.	ssl-cert is not specified in /etc/mysql/my.cnf
application_configuration:mysql.ssl-key	Active	A setting in /etc/mysql/my.cnf that identifies the server private key.	ssl-key is not specified in /etc/mysql/my.cnf

```

OK

```

In this case, all issues are related to SSL. You can ignore these issues in this exercise.

Vulnerability Advisor checks the security status of container images.

When you add an image to a namespace, the image is automatically scanned by Vulnerability Advisor to detect security issues and potential vulnerabilities. If any security issues are found, instructions are provided to help fix the reported vulnerability.

Part 3: Creating a deployment and scaling it

Deployments are used to manage pods, which include containerized instances of an app. You can run an application by creating a Kubernetes deployment, and you can describe a deployment in a YAML file.

When you create a deployment, a Kubernetes pod is created for each container that you defined in the deployment.

Complete the following steps:

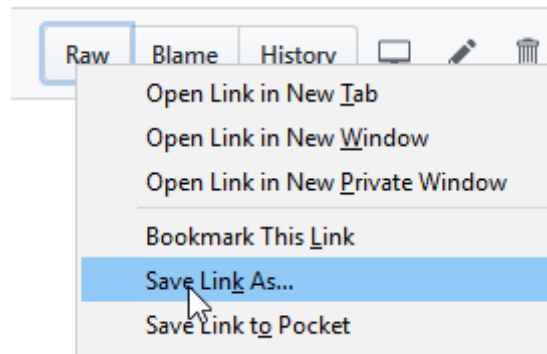
- ___ 1. Create a folder that is named **yaml** to store the configuration files by running the following commands:


```
mkdir yaml
cd yaml
```
- ___ 2. Create a deployment file (`01hello-student-deployment.yaml`). Download the file from the following link and replace the image tag by the one that you created in Part 2:

<https://github.com/IBM-SkillsAcademy/Cloud-Application-Developer/blob/master/CloudAppDev/Ex6/01hello-student-deployment.yaml>

**Hint**

To download the file from the Git repo, right-click **Raw** and select **Save Link As** as shown in the following figure.



Save the file in the **yaml** directory that you previously created.

The following code snippet shows the `01hello-student-deployment.yaml` file in this example. A pod with the following specifications is deployed:

- Your deployment name is `hello-student-deployment`.
- When you bind the deployment to the service, you use a selector. Selectors are key value pairs. In this exercise, the selector is `app : hello-student`.
- The pod has one replica with `100Mi` memory and `125m` CPU.
- The application is exposed on port `3000`.

- The target image is "us.icr.io/cr-student/hello-student:v1".

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-student-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-student
  template:
    metadata:
      labels:
        app: hello-student
    spec:
      containers:
        - name: hello-student
          image: us.icr.io/cr-student/hello-student:v1
          resources:
            limits:
              memory: "100Mi"
              cpu: "125m"
            requests:
              memory: "100Mi"
              cpu: "125m"
          ports:
            - containerPort: 3000

```

___ 3. Modify some of the properties of 01hello-student-deployment.yaml:

- ___ a. Increase the upper limits of the memory and CPU to allow vertical scalability of the containers inside the pods. Under limits, change the memory to 200Mi and the CPU to 250m.
- ___ b. Change image to the image tag that you created in Part 2.
- ___ c. To set up high availability for your deployments, add two more replicas, which enable horizontal scalability. Change replicas: 1 to replicas: 3.

Your deployment file now looks like the following example.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-student-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-student
  template:
    metadata:
      labels:
        app: hello-student
    spec:
      containers:
      - name: hello-student
        image: us.icr.io/cr-studentxx/hello-student:v1
        resources:
          limits:
            memory: "200Mi"
            cpu: "250m"
          requests:
            memory: "100Mi"
            cpu: "125m"
        ports:
        - containerPort: 3000

```

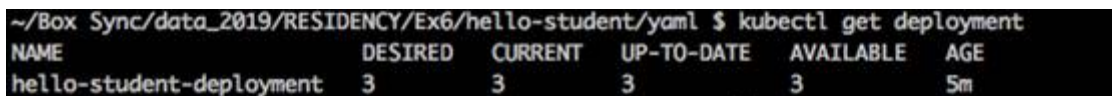
- ___ 4. Apply the deployment to your Kubernetes cluster by running the following command:

```
kubectl apply -f 01hello-student-deployment.yaml
```

- ___ 5. Retrieve all the deployments in your cluster and observe the number of replicas by running the following command:

```
kubectl get deployment
```

The following figure shows the output.



NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
hello-student-deployment	3	3	3	3	5m

- ___ 6. Retrieve all the pods in your cluster by running the following command. Your app is deployed three times in each pod.

```
kubectl get pods
```

The following figure shows the output.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-student-deployment-7d799d48c5-7l6xl	1/1	Running	0	7m
hello-student-deployment-7d799d48c5-l8sqc	1/1	Running	0	7m
hello-student-deployment-7d799d48c5-xsf9v	1/1	Running	0	7m

7. Try Kubernetes self-healing by deleting all the pods. After the pods are deleted, Kubernetes self-heals by always ensuring that three replicas are deployed.
- a. Run the `kubectl delete pods --all` command to delete all the pods. The following figure shows the output.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl delete pods --all
pod "hello-student-deployment-7d799d48c5-7l6xl" deleted
pod "hello-student-deployment-7d799d48c5-l8sqc" deleted
pod "hello-student-deployment-7d799d48c5-xsf9v" deleted
```

- b. To retrieve all pods in your cluster again, run the following command:

```
kubectl get pods
```

As shown in the following figure, you can see that the statuses of some containers are Terminating and some are ContainerCreating.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-student-deployment-7d799d48c5-24h9s	1/1	Terminating	0	47s
hello-student-deployment-7d799d48c5-4zn2r	1/1	Terminating	0	46s
hello-student-deployment-7d799d48c5-9d9x9	1/1	Terminating	0	46s
hello-student-deployment-7d799d48c5-chrdr	0/1	ContainerCreating	0	4s
hello-student-deployment-7d799d48c5-hpcf9	0/1	ContainerCreating	0	4s
hello-student-deployment-7d799d48c5-wzppr	0/1	Pending	0	3s



Troubleshooting

If you cannot see the `ContainerCreating` status, it is because containers are healing before you can run the `kubectl get pods` command. Delete again and check the pods' statuses faster.

After a few seconds, run the `kubectl get pods` command again. The following figure shows that three pods are running as deployed.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-student-deployment-7d799d48c5-chrdr	1/1	Running	0	4m
hello-student-deployment-7d799d48c5-hpcf9	1/1	Running	0	4m
hello-student-deployment-7d799d48c5-wzppr	1/1	Running	0	4m

**Note**

After the pods are terminated for any reason, Kubernetes automatically self-heals by creating pods.

___ 8. Scale in and out the deployment by changing the number of replicas:

- ___ a. Find the deployment name by running the `kubectl get deployment` command. The following figure shows the output.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl get deployment
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
hello-student-deployment	3	3	3	3	5m

- ___ b. Run the following command:

```
kubectl scale deployment <deployment name> --replicas=4
```

**Example**

```
kubectl scale deployment hello-student-deployment --replicas=4
```

- ___ c. Check the status of pods by running the following command. The following figure shows that there are four pods.

```
kubectl get pods
```

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-student-deployment-7d799d48c5-chrdr	1/1	Running	0	25m
hello-student-deployment-7d799d48c5-ghpzw	1/1	Running	0	4s
hello-student-deployment-7d799d48c5-hpcf9	1/1	Running	0	25m
hello-student-deployment-7d799d48c5-wzppr	1/1	Running	0	25m

- ___ 9. Run a command to create a Horizontal Pod Autoscaler that maintains between 4 and 10 replicas of the Pods to achieve an average CPU utilization across all Pods of 80%.

```
kubectl autoscale deployment <deployment name> --cpu-percent=80 --min=4
--max=10
```

**Example**

```
kubectl autoscale deployment hello-student-deployment --cpu-percent=80 --min=4
--max=10
```

Part 4: Exposing the app over the internet

Kubernetes supports four basic types of network services: ClusterIP, NodePort, LoadBalancer, and Ingress. ClusterIP services make your apps accessible internally to allow communication among

pods in your cluster only. The NodePort, LoadBalancer, and Ingress services make your apps externally accessible from the public internet or a private network.

In this example, you expose the service by using NodePort.



Note

When you expose apps with the NodePort service, a NodePort of 30000 - 32767 and an internal cluster IP address is assigned to the service. To access the service from outside the cluster, you use the public or private IP address of any worker node and the NodePort in the format `<IP_address>:<nodeport>`.

Complete the following steps:

1. Download the `02hello-student-service.yaml` file from the Git repo at the following link and save it in the **yaml** directory:

<https://github.com/IBM-SkillsAcademy/Cloud-Application-Developer/blob/master/CloudAppDev/Ex6/02hello-student-service.yaml>

A service with the following specifications is deployed, as shown in the following example:

- The service name is `hello-student-service`.
- The service is bound to the pods that have the label `app: x`.
- The application is exposed on port 3000.

```
apiVersion: v1
kind: Service
metadata:
  name: hello-student-service
spec:
  selector:
    app: x
  ports:
    - name: http
      protocol: TCP
      port: 3000
  type: NodePort
```

- ___ 2. Update app: x to app: hello-student to match the selector that is defined in the deployment. The final deployment file looks like the following example.

```
apiVersion: v1
kind: Service
metadata:
  name: hello-student-service
spec:
  selector:
    app: hello-student
  ports:
    - name: http
      protocol: TCP
      port: 3000
      type: NodePort
```

- ___ 3. Apply the deployment file by running the following command:

```
kubectl apply -f 02hello-student-service.yaml
```

- ___ 4. Retrieve the details about your services by running the following command:

```
kubectl get services
```

The following figure shows the output.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student $ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
hello-student-service	NodePort	172.21.200.155	<none>	3000:30456/TCP	1m
kubernetes	ClusterIP	172.21.0.1	<none>	443/TCP	18d

Your service is exposed on a specific port number, which you can find in the PORT(S) column. In the previous screen, it is 30456, which means any requests to your worker nodes on this port are directed to your application.

- ___ 5. Retrieve your worker node public IP by running the following command:

```
ibmcloud ks workers mycluster
```

The following figure shows the output.

```
~/Box Sync/data_2019/RESIDENCY/Ex6/hello-student/yaml $ ibmcloud ks workers mycluster
```

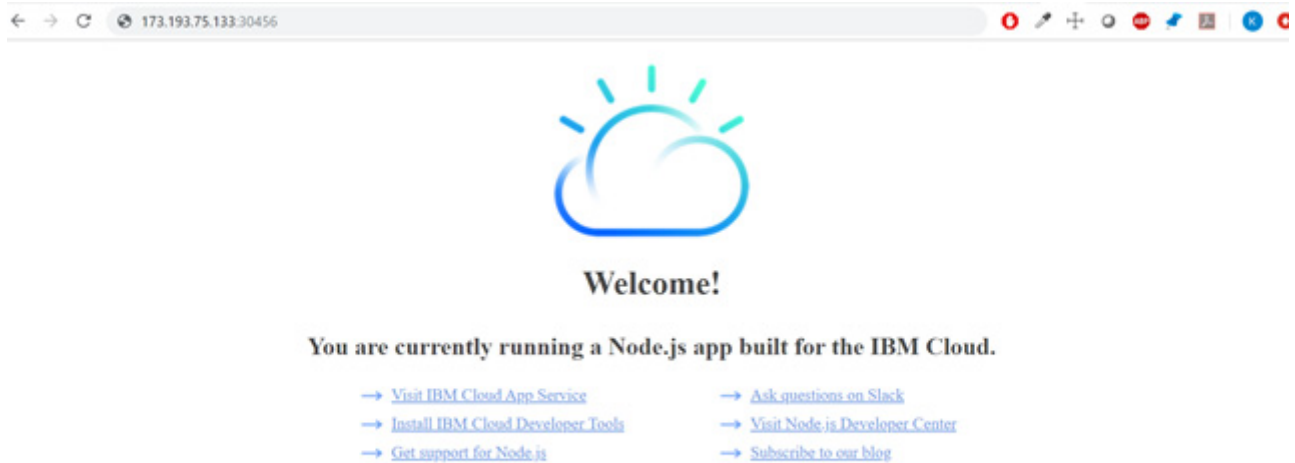
OK	ID	Public IP	Private IP	Machine Type	State	Status	Zone	Version
	kube-hou02-paf7cb9bfddc7c455688e17077cea231a3-w1	173.193.75.133	10.44.12.8	free	normal	Ready	hou02	1.12.7_1548*

- ___ 6. Access your application on the browser through the public IP of your worker node that you retrieved in Step 5 and the port that you retrieved in Step 4:

<Public IP>:<Port>

1+1=2 Example

173.193.75.133:30456



Part 5: Cleaning up the environment

In this part, you delete the Kubernetes cluster that you created on IBM Cloud.

- ___ 1. Open IBM Cloud dashboard.
- ___ 2. Expand **Kubernetes Clusters** and click the three dots next to the cluster name as shown in the figure

Name ▲	Group	Location	Status	Tags
Q Filter by name or IP address...	Filter by group or org...	Filter...	Q Filter...	Filter...
> Devices (0)				
▼ Kubernetes Clusters (1)				
mycluster	Default	Dallas	● Normal	-- ...
> Cloud Foundry Apps (0)				
> Cloud Foundry Services (0)				
> Services (0)				
> Storage (0)				
> Cloud Foundry Enterprise Environments (0)				
> Apps (7)				

- ___ 3. Click **Delete**.
- ___ 4. At the Delete resource window, enter the name of your cluster to confirm that you want to delete it, and click **Delete** as shown in the figure.

Delete resource

When you delete 'mycluster', all worker nodes, apps, and containers are permanently deleted. This action cannot be undone. Before you proceed, make sure to back up all required data and configuration files.

Type 'mycluster' to confirm

mycluster

Cancel

Delete

- ___ 5. Delete all namespaces from IBM Cloud registry by accessing this URL
<https://cloud.ibm.com/kubernetes/registry/main/start>
- ___ 6. Click **Namespaces**. Select your namespace, click the three dots to the right and select **Delete namespace**, as shown in the following figure.

The screenshot shows the IBM Cloud Kubernetes Registry console. The left sidebar has a 'Kubernetes' header and a menu with 'Overview', 'Clusters', 'Registry', and 'Helm Catalog'. The main content area is titled 'Registry' with a 'LOCATION Dallas' dropdown. It features a 'Quick Start' section with icons for 'Namespaces', 'Repositories', and 'Images'. Below this, the 'Namespaces' section is active, showing a table with one namespace: 'cr-studentbox'. A 'Delete namespace' modal is open, displaying the table and a 'Delete namespace' button.

Name	Repository Count	Image Count
cr-studentbox	1	1

End of exercise

Exercise review and wrap-up

In this exercise, you created a Node.js application by using IBM Cloud App Service starter kits and deployed the application to the IBM Cloud Kubernetes Service with three replicas to ensure high availability.

Also, you learned how self-healing works, and you exposed your application to the public by using the NodePort service.

Troubleshooting

This section provides some basic troubleshooting techniques to fix errors that you may encounter while you perform this exercise.

Storage quota exceeded

If you encounter the error that is shown in the following figure, delete one or more images.

```

[18495ddff4]: Preparing
[182b00c71e]: Preparing
[185b1a8b49]: Preparing
[183ea5415f]: Preparing
[1890f129c3]: Preparing
[1877893bad]: Preparing
[186aad86d]: Preparing
[182fe51680]: Preparing
[18c4b192e8]: Preparing
[18c551a0da]: Preparing
[1831157b81]: Preparing
[18d801397d]: Preparing
[1868cde90b]: Preparing
[18FAILED9a]: Waiting g
Status: denied: You have exceeded your storage quota. Delete one or more images, or review your storage quota and pricing plan. For more information, see https://ibm.biz/BdjFWL, Code: 1

```

- ___ 1. List the available images by running this command:

```
ibmcloud cr image-list
```

```

C:\Users\NorhanKhaled\Downloads\hello-student-772019>ibmcloud cr image-list
Listing images...

REPOSITORY                                TAG    DIGEST                NAMESPACE    CREATED        SIZE    SECURITY STATUS
us.icr.io/cr-student/hello-student      v1     b658d76c2f50         cr-student   46 minutes ago 422 MB  3 Issues
us.icr.io/cr-student12/hello-student    v1     7dc6570b1c6c         cr-student12 1 month ago   416 MB  4 Issues
OK

```

- ___ 2. Remove the image by running this command:

```
ibmcloud cr image-rm <REPOSITORY>:<TAG>
```

```

C:\Users\NorhanKhaled\Downloads\hello-student-772019>ibmcloud cr image-rm us.icr.io/cr-student/hello-student:v1
Deleting image 'us.icr.io/cr-student/hello-student:v1'...

Successfully deleted image 'sha256:b658d76c2f50f895d600f6e73dabc4b023039328808fdd5449912657332e0f30'
OK

```

Requested namespace already exists

If you encounter the error that is shown in the following figure, remove the existing namespace or create a new namespace with a unique name.

```
C:\Users\NorhanKhaled\Downloads\hello-student-772019\yaml>ibmcloud cr namespace-add cr-student
Adding namespace 'cr-student'...

The requested namespace is already owned by your account.

OK
```

To list all the namespaces own by your account, run the command

```
ibmcloud cr namespace-list
```

```
C:\Users\NorhanKhaled\Downloads\hello-student-772019\yaml>ibmcloud cr namespace-list
Listing namespaces for account 'Student91's Account' in registry 'us.icr.io'...

Namespace
cr-student
cr-student12
cr-studentaa

OK
```

To delete a namespace, run the command:

```
ibmcloud cr namespace-rm <namespace>
```

Enter **y** for Deleting the namespace will also delete all images and trust information in the namespace. [y/N]>, as shown in the following figure.

```
C:\Users\NorhanKhaled\Downloads\hello-student-772019\yaml> ibmcloud cr namespace-rm cr-student

Are you sure you want to delete namespace 'cr-student'? Deleting the namespace will also delete all images and trust information in the namespace. [y/N]> y
Deleting namespace 'cr-student'...

Successfully deleted namespace 'cr-student'

OK
```

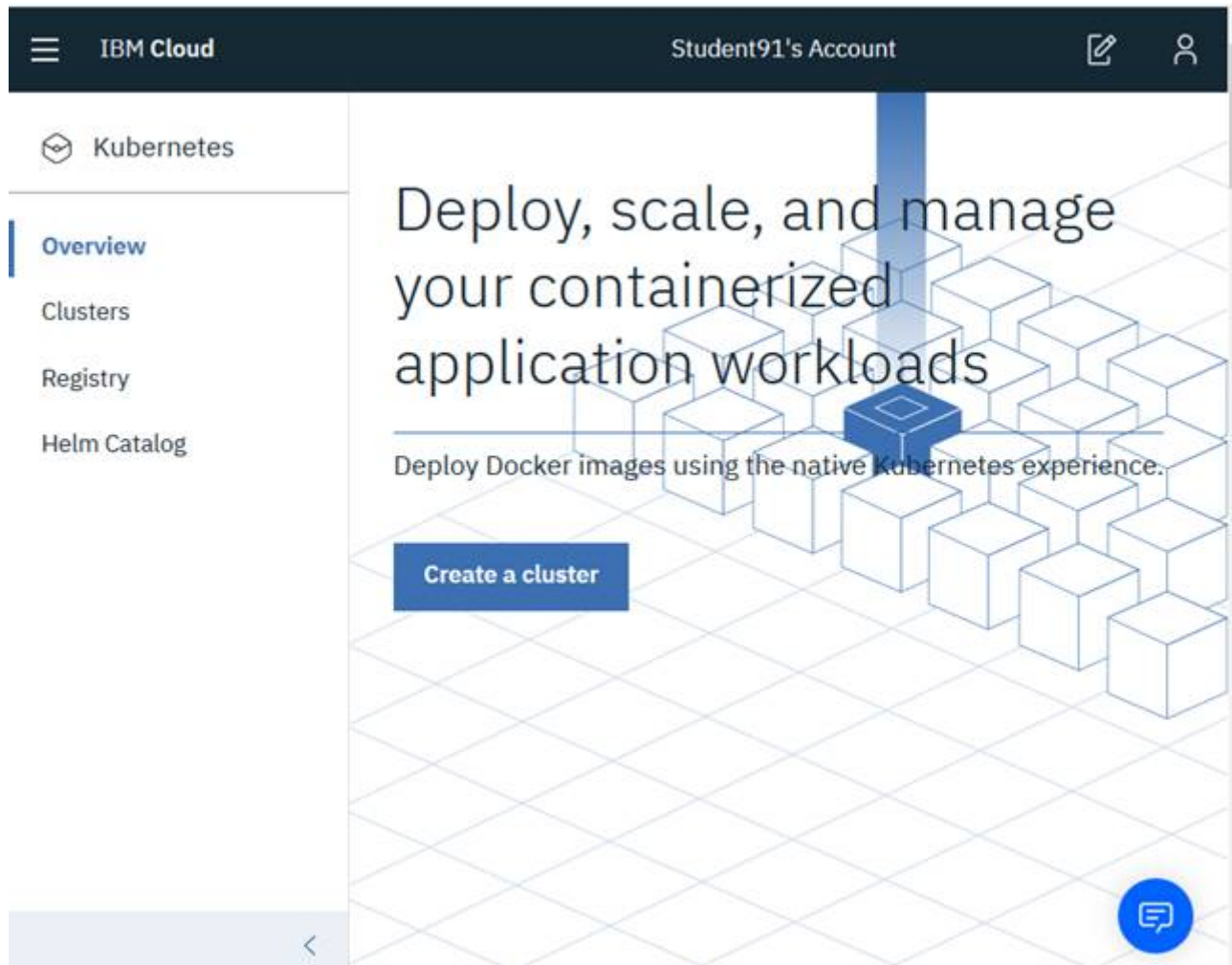
Error validating data with kubectl apply

If you receive the error that is shown in the following figure when you run the `kubectl apply` command, update the `kubectl` version in your workstation to the latest stable version by following the steps in Exercise 0, section **Installing the Kubernetes CLI**.

```
error: error validating "02hello-student-service.yaml": error validating data: the server could not find the requested resource; if you choose to ignore these errors, turn validation off with --validate=false
```

Accessing the Kubernetes dashboard

You can view, create, and delete namespaces, clusters, repositories and images through the Kubernetes dashboard at <https://cloud.ibm.com/kubernetes/overview> as shown in the following figure.



For more information see *[Troubleshooting clusters](https://cloud.ibm.com/docs/containers?topic=containers-cs_troubleshoot)* at https://cloud.ibm.com/docs/containers?topic=containers-cs_troubleshoot