

Build Smart on Kubernetes Learning Journey

Introduction to Red Hat OpenShift on IBM Cloud

Table of Contents

Getting Started	1
Introduction	1
Prerequisites.....	2
Using the lab guide	2
Acknowledgements.....	3
Exercise 1	4
Red Hat OpenShift on IBM Cloud Basics	4
Exercise 2	6
Deploying an application	6
Exercise 3	11
Using the command line interface (CLI)	11
Exercise 4	16
Logging and events.....	16
Exercise 5	19
Metrics and Dashboards	19
Exercise 6	23
Scaling the application.....	23
Next Steps.....	30
Next steps.....	30

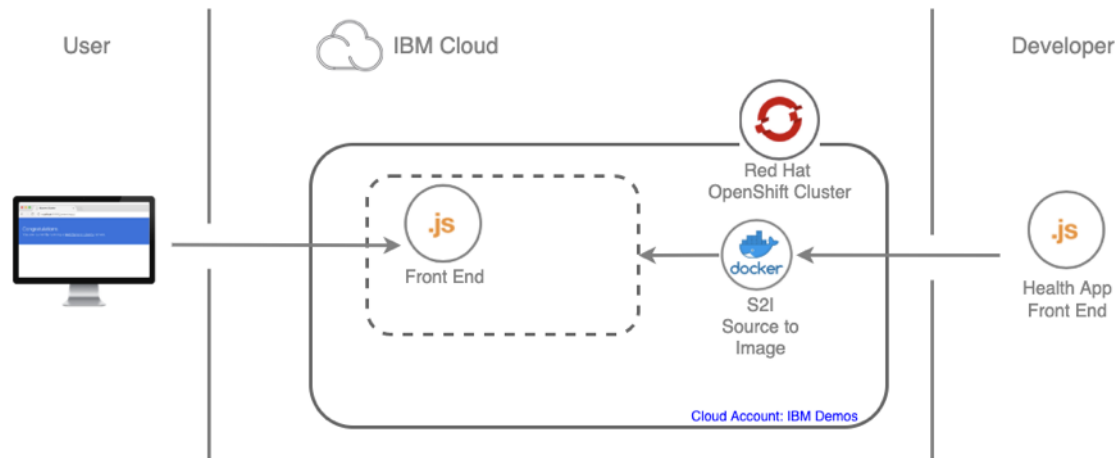
Getting Started

Introduction

Red Hat OpenShift on IBM Cloud provides fully managed OpenShift clusters running on the highly scalable and reliable IBM Cloud platform. This managed environment allows you to focus on developing and managing your applications while IBM handles the infrastructure.

In this lab, learn the basics of Red Hat OpenShift on IBM Cloud using real world scenarios of deploying a sample health care application, including using the web console and command-

line interfaces (CLI). The diagram below depicts the architecture of the solution that is deployed in this lab.



In this lab, you will:

- Deploy an application
- Use the OpenShift command line interface
- Explore logs and events that are generated
- Explore metrics and dashboards available
- Scale an application

In Lab 2, you deploy a backend that utilizes a cloud-based database; and in Lab 3, you explore using LogDNA to gain insights into the application.

Note: Labs 1 and 2 build upon each other and should be completed in a single session. Lab 3 can be performed by itself with the exception of a couple of optional steps to view logs from Lab 1 and Lab 2.

Prerequisites

You will need the following to complete the exercises in this lab:

- Familiarity with Kubernetes
- An IBM Cloud user ID

Using the lab guide


Getting started

- Use the URL provided by the instructors, eg.
URL: <https://buildsmart.mybluemix.net/>
- On the form enter the provided Key: eg. oslab and your IBMid


Welcome to an IBM Cloud lab

Use this form to get access to a lab environment

Lab Key (provided by the host)

 mylab

Your IBMid

 myID@mail.com

☒ I agree to the [terms and conditions](#)

Submit

[Here is a FAQ for users.](#)

You will need to acknowledge the cluster assignment by clicking a link in an email you will be sent once you submit this form. You are being added to a development account on IBM Cloud for the duration of the workshop which requires your acceptance. Please check your email.

You need a cloudshell you can use <https://shell.cloud.ibm.com/>. It should be attached to the IBMid, that you created for the workshop.

- Browse to the OpenShift web console.
- From the dropdown menu in the upper right of the page, click "Copy Login" Command.
- Click the "Display Token" link.
- Copy the "Log in with this token" command line.
- Paste the copied command in your terminal.
- Browse to the oauth token request page and follow the instructions on the page.

Sample command:

```
oc login --server https://c109-e.us-east.containers.cloud.ibm.com:31411 -u apikey -p 75ecc28297050904b4fedac2be174595
```

Acknowledgements

Acknowledgements

This core of this lab has been used in other training courses and environments. The content has matured over time and has been updated based upon new releases of Red Hat OpenShift on IBM Cloud. Contributors include:

- Spencer Krum

- JJ Asghar
 - Tim Robinson
 - Mofi Rahman
 - Sai Vennam
 - Steve Martinelli
 - Ram Vennam
 - Remko De Knikker
 - Alex Parker
 - Lionel Mace
 - Marisa Lopez de Silanes Ruiz
-

Exercise 1

Red Hat OpenShift on IBM Cloud Basics

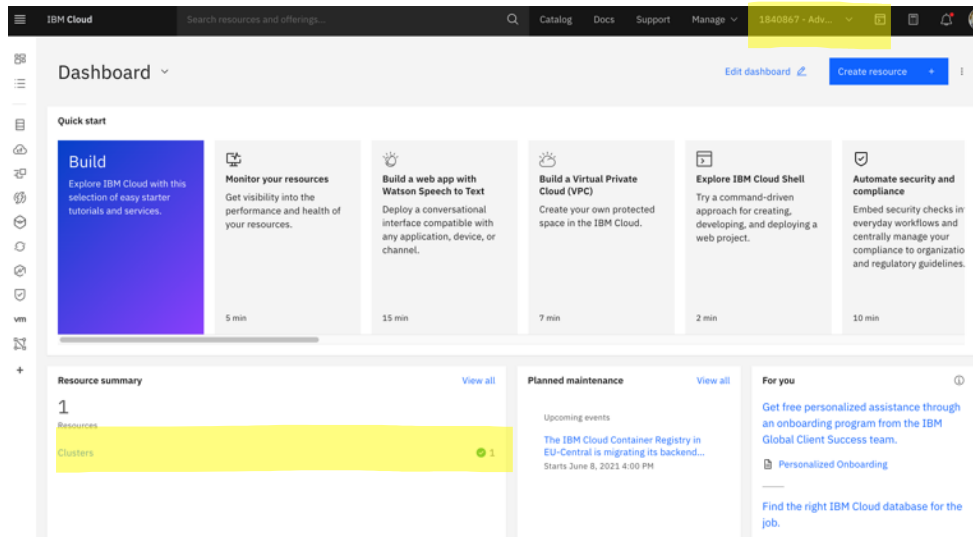
Exercise 1

In this exercise, learn how to access the IBM Cloud portal and the OpenShift web console. An OpenShift cluster has already been provisioned and temporarily assigned to you as part of the lab provisioning process.

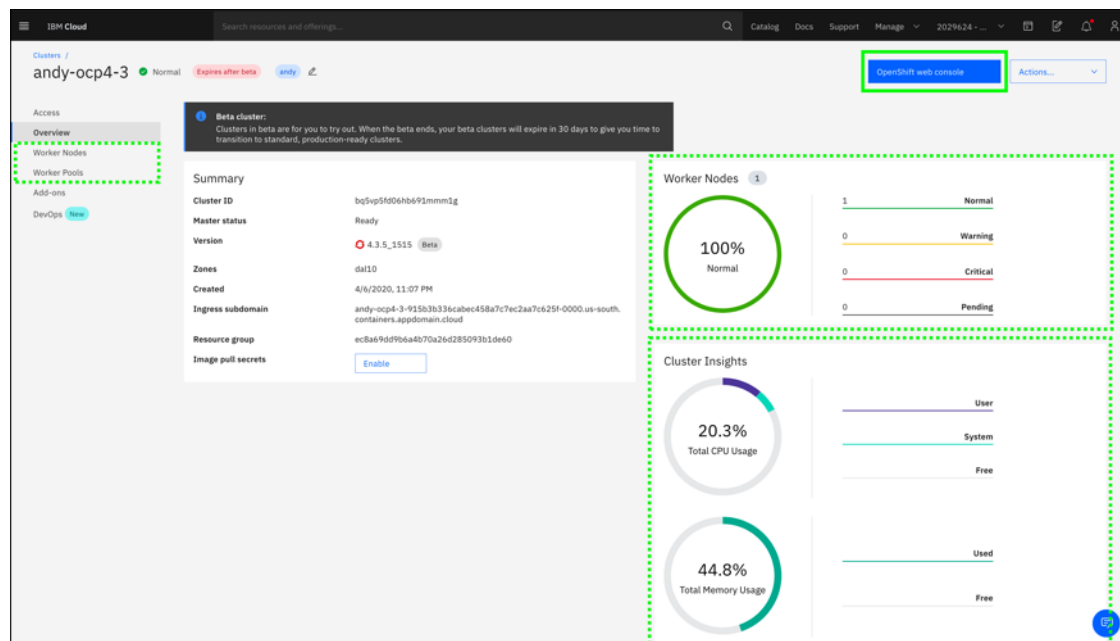
Note: The sample output shown in the lab guide may be slightly different than what you see in the output when you issue the commands for your cluster.

Access the OpenShift web console

1. Launch the IBM Cloud portal: <https://cloud.ibm.com> and ensure the xxxxxx – Advwork account is selected. Navigate to the Clusters and open the cluster for the the lab.



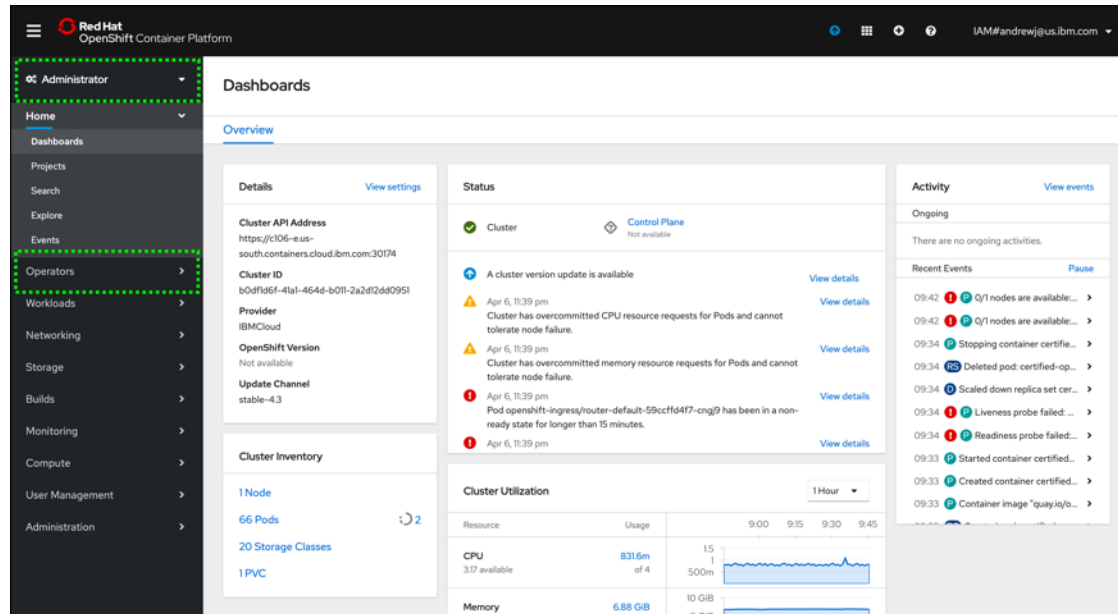
2. **Note:** The IBM Cloud Portal and OpenShift web console will be opened in a new browser tab. You need to switch between this tab and the new tabs to accomplish many of the lab tasks. You may want to open the new tabs in new windows and display both browser windows at the same time. You may need to disable pop-up blockers if you do not see the new tabs.



3. On the IBM Cloud portal page, notice details about the OpenShift cluster that has been provisioned, including:
 - Number of Worker Nodes and their status (you should see two worker nodes in a “normal” state)
 - Links to manage worker nodes and worker pools
 - Link to open the “OpenShift web console”

Note: In this lab environment, your IBM Cloud User ID has been given limited permissions in an IBM Advwork account. While you can explore these options in the IBM Cloud portal, you will not be able to add, remove, or modify worker nodes or pools. To learn more about these processes, check out the videos and product tours here: [Red Hat OpenShift on IBM Cloud Demos](#).

4. Next, explore the **OpenShift web console**.



5. In the OpenShift web console, notice the following:
- The different perspectives available in the OpenShift web console: Administrator and Developer
 - Capabilities of each perspective and dashboards available for each
6. Take a few minutes to explore the OpenShift web console. Try finding the following:
- Number of nodes in the cluster inventory dashboard
 - Standard services deployed in the default project
 - Operators installed

End Exercise 1 __

Exercise 2

Deploying an application

Exercise 2

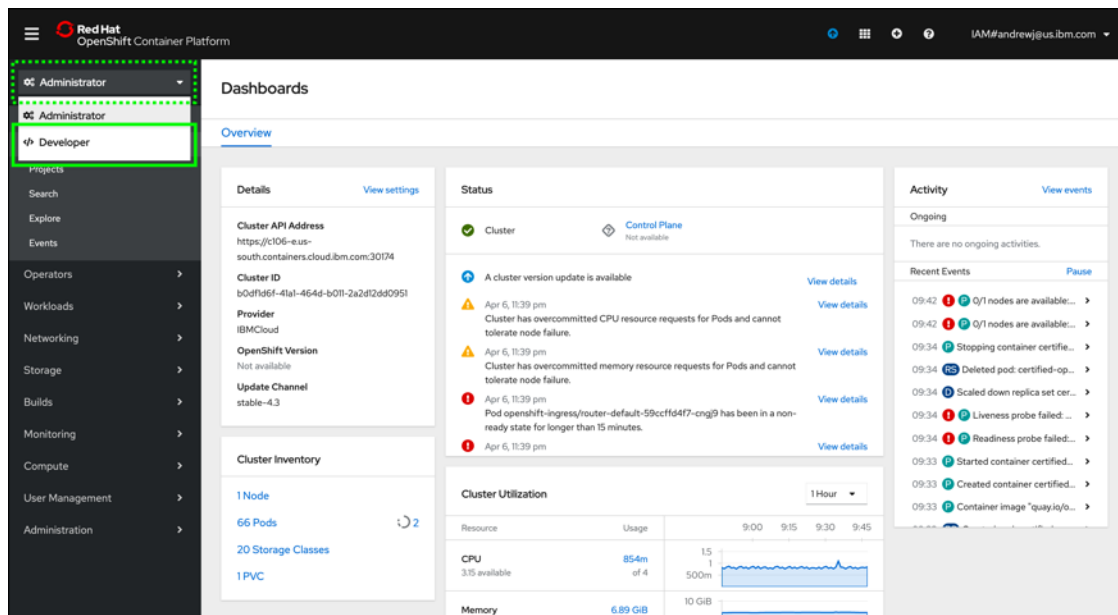
In this exercise, deploy a simple Node.js Express application - "Example Health". Example Health is a simple UI for a patient health records system. We will use this example to demonstrate key OpenShift features throughout this workshop.

Deploy Example Health

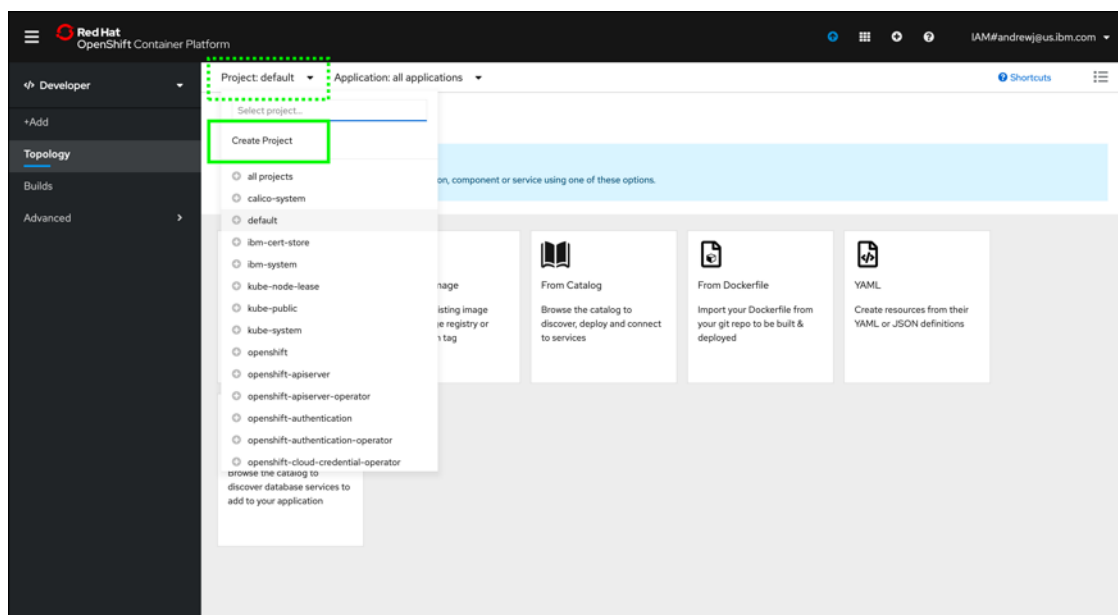
1. Launch the **OpenShift web console**, if not already running.

You should still have a browser tab open with the OpenShift web console from the previous exercise. We will use the web console for this exercise. If you closed that tab, you can access your OpenShift web console on via the “OpenShift web console” button at the top of your cluster page on the IBM Cloud page (see step 2 in exercise 1).

2. Use the **Developer** perspective. If you are currently in the Administrator perspective, switch to the Developer perspective.



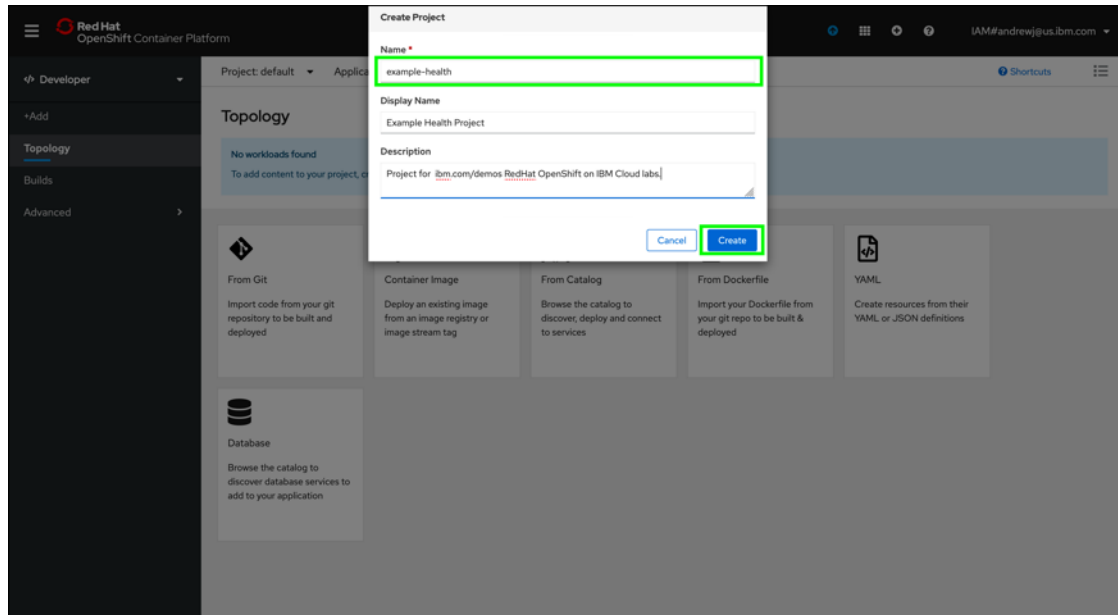
3. While on the **Topology** page, Select the **Project** pulldown and click **Create Project**.



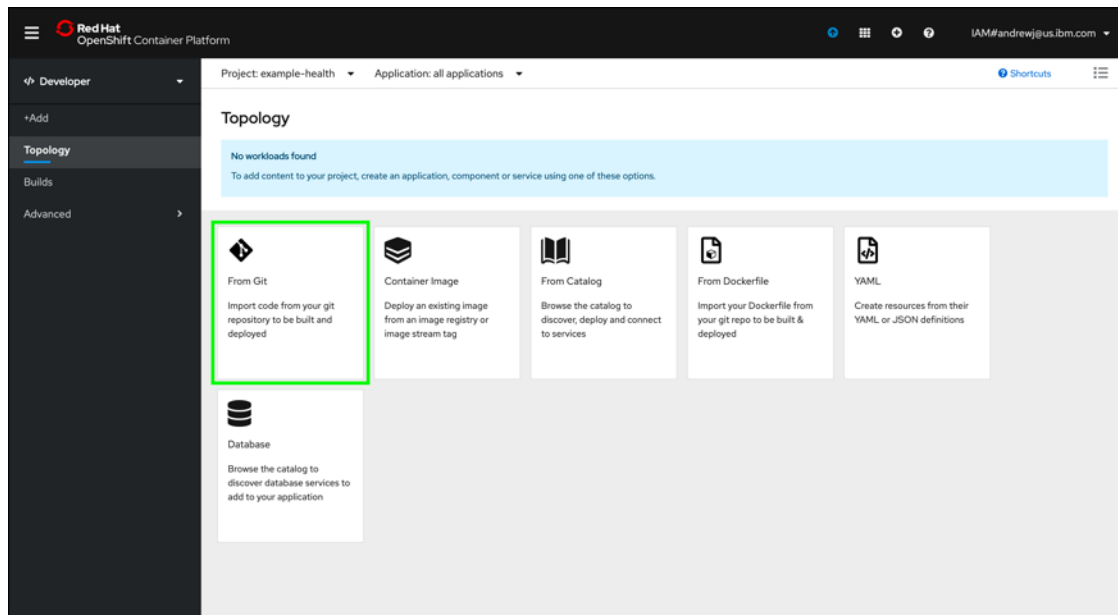
4. Enter “example-health” as the new project’s **Name**.

example-health

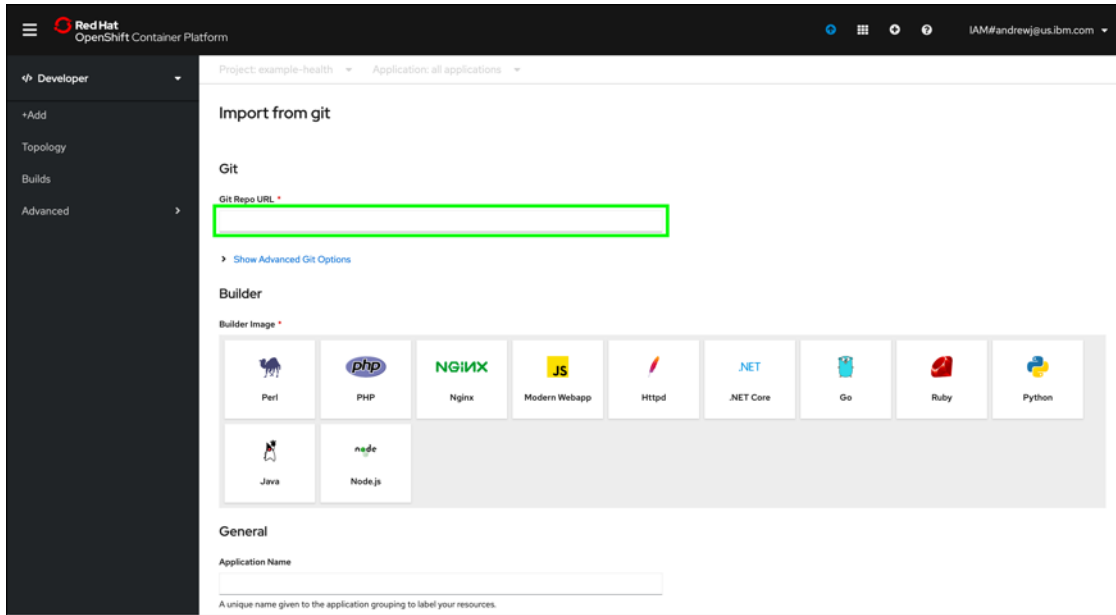
Optionally specify a **Display Name** and **Description**.



5. You should see a view that looks like this.



6. Let’s deploy the application by selecting the **From Git** tile.



7. In the **Git Repo URL field**, enter: <https://github.com/IBM/ibm-dte-openlab-samples>.

<https://github.com/IBM/ibm-dte-openlab-samples>

8. Press the **tab** key to validate the repository path.

Note: If you receive an error regarding invalid repository, make sure you do not have any trailing spaces in the repo field. You can ignore the "Unable to detect the builder image" message, we will fix that in the next step.

9. Click **Show Advanced Git Options**.
10. Enter **/Red Hat OpenShift on IBM Cloud/node-s2i-openshift-master/** in the **Context Dir** field.

[Red Hat OpenShift on IBM Cloud/node-s2i-openshift-master/](#)

Note: there is a leading "/" already in the entry field. If you copy from the lab guide, click in the field and do a paste or control v, the correct value will be: "/Red Hat OpenShift on IBM Cloud/node-s2i-openshift-master/".

11. Scroll down and select the **Node.js** tile under **Builder Image**.

12. Scroll down and change **Application Name** to **patient-ui** and **Name** to **node-s-2-i-openshift**.

```
patient-ui  
node-s-2-i-openshift
```

Note: These values are used later in this lab as well as Lab 2. If you change these names, remember the values and make appropriate changes in the subsequent steps.

13. Click **Create** at the bottom of the window to build and deploy the application.

Your application is being deployed. This takes a few minutes. While it is being deployed, you see the state icon of your application change:

14. Once the application has been completely deployed, the bottom icon changes to a green check mark.

Note: During the build process, OpenShift will register 2 or 3 image pull errors. The application should successfully deploy within a couple of minutes. If it does not, review the logs and events as described in subsequent exercises in this lab, or delete the application from your topology and deploy it again making sure to specify the information exactly as described in this lab guide.

15. Click the **Node.js** label in the graphic. A panel like below is displayed.

This panel displays the details about your Pods, Builds, Services and Routes. Note that Build #1 should be complete. If it is not yet complete, wait until it changes to the complete state (green check mark) before continuing.

- Pods: the Node.js application containers
- Builds: auto-generated build of Docker image from your Node.js source
- Services: access to pods and listening ports

- Routes: exposed service route using LoadBalancer provided by IBM Cloud network

16. Click the URL under **Routes** or the link icon in the graphic to open your application.

A new window or tab should open in your browser for your application.

17. **Sign In** to your application. You can enter any strings for username and password, for instance test:test because the app is running in demo mode.

The application is now up and running and you have successfully logged in.

Congrats! You've deployed a Node.js app to OpenShift Container Platform using a Source to Image (S2I) method. You can learn more about the Source to Image (S2I) method [here](#).

You have completed the following:

- Deployed the "Example Health" Node.js application directly from GitHub into your cluster
- Used the "Source to Image" strategy provided by OpenShift

End Exercise 2

Exercise 3

Using the command line interface (CLI)

Exercise 3

In this exercise, learn how to access your cluster using the OpenShift command-line interface (CLI).

You need a ccloudshell you can use <https://shell.cloud.ibm.com/>. It should be attached to the IBMid, that you created for the workshop.

You will be using the lab terminal to access your cluster. The terminal is running on an IBM Cloud virtual server instance (VSI). The IBM Cloud CLI and Red Hat OpenShift CLI have already been installed for you.

Note: The sample output shown in the lab guide may be slightly different than what you see in the output when you issue the commands for your cluster.

Red Hat OpenShift utilizes the **oc** command to extend the capabilities of the **kubectl** command. The **kubectl** command is provided as well to support existing workflows and scripts. You can learn more about the **oc** and **kubectl** differences [here](#).

Normally, to get the login command, you would need to:

- Browse to the OpenShift web console.
- From the dropdown menu in the upper right of the page, click Copy Login Command.
- Click the "Display Token" link.
- Copy the "Log in with this token" command line.

```
oc login --token=sha256~ycT1j9l47XDmtkHDz_A70-_UCGsddMYeK5RTaTAJq-8 --
server=https://c117-e.us-south.containers.cloud.ibm.com:30035
```

- Paste the copied command in your terminal.
- Browse to the oauth token request page, and follow the instructions on the page.

You should see a success message similar to this:

```
Logged into "https://c106-e.us-south.containers.cloud.ibm.com:30174" as
"IAM#andrewj@us.ibm.com" using the token provided.
```

```
You have access to 58 projects, the list has been suppressed. You can list
all projects with 'oc projects'
```

```
Using project "default".
```

2. Validate access to your cluster by viewing the nodes in the cluster:

```
oc get node
```

Sample Output:

NAME	STATUS	ROLES	AGE	VERSION
10.171.76.77	Ready	master,worker	37h	v1.16.2

3. Execute the command below to view services, deployments, and pods:

```
oc get svc,deploy,po --all-namespaces
```

Sample Output:

NAME	STATUS	ROLES	AGE	VERSION
10.171.76.77	Ready	master,worker	37h	v1.16.2

```
container-lab$ oc get svc,deploy,po --all-namespaces
```

NAMESPACE	NAME	
TYPE	CLUSTER-IP	EXTERNAL-IP

PORT(S)		AGE	
calico-system			service/calico-typha
ClusterIP	172.21.108.58	<none>	
5473/TCP		37h	
default			service/kubernetes
ClusterIP	172.21.0.1	<none>	
443/TCP		38h	
default			service/openshift
ExternalName	<none>	kubernetes.default.svc.cluster.local	<none>
37h			
default			service/openshift-
apiserver		ClusterIP	172.21.146.181 <none>
...			

****Note:**** Some commands have large amounts of data in their output. You can scroll up and down in the terminal window and clear it using the “clear” command

4. Execute the command below to clear the terminal screen:

```
clear
```

5. Execute the command below to view all OpenShift projects:

```
oc get projects
```

Sample Output:

NAME	DISPLAY NAME
STATUS	
calico-system	
Active	
default	
Active	
example-health	Example Health
Project Active	
ibm-cert-store	
Active	
ibm-system	
Active	
kube-node-lease	
Active	
kube-public	
Active	
kube-system	
Active	
openshift	
Active	
openshift-apiserver	
Active	
openshift-apiserver-operator	
Active	
...	

6. Set default project to **example-health**:

By setting the default project, any subsequent command that is project specific will be performed against that project.

```
oc project example-health
```

7. Retrieve the public route to access your Example Health application:

```
oc get routes
```

Sample output:

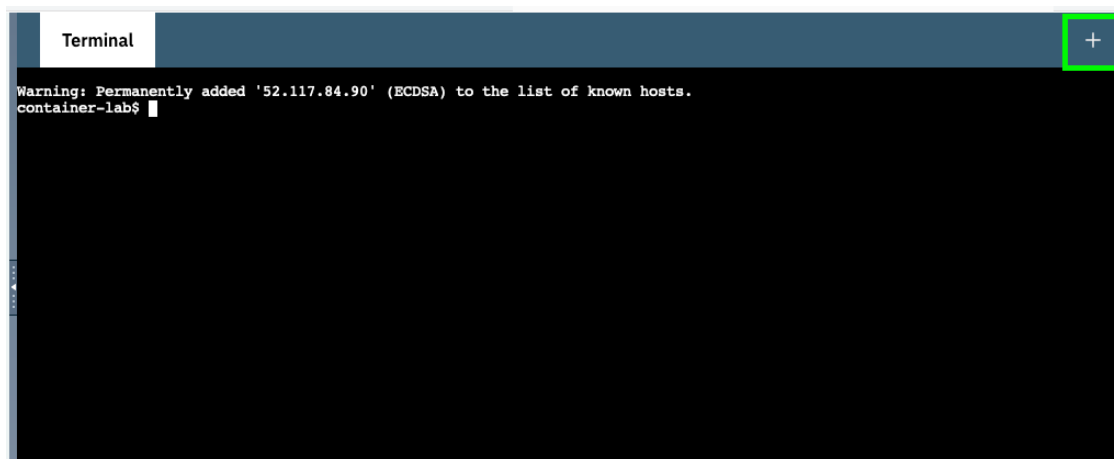
NAME	HOST/PORT
PATH SERVICES	PORT TERMINATION WILDCARD
node-s-2-i-openshift	node-s-2-i-openshift-example-health.andy-ocp4-3-915b3b336cabec458a7c7ec2aa7c625f-0000.us-south.containers.appdomain.cloud
node-s-2-i-openshift	8080-tcp None

We will use the host name of the output above in a minute.

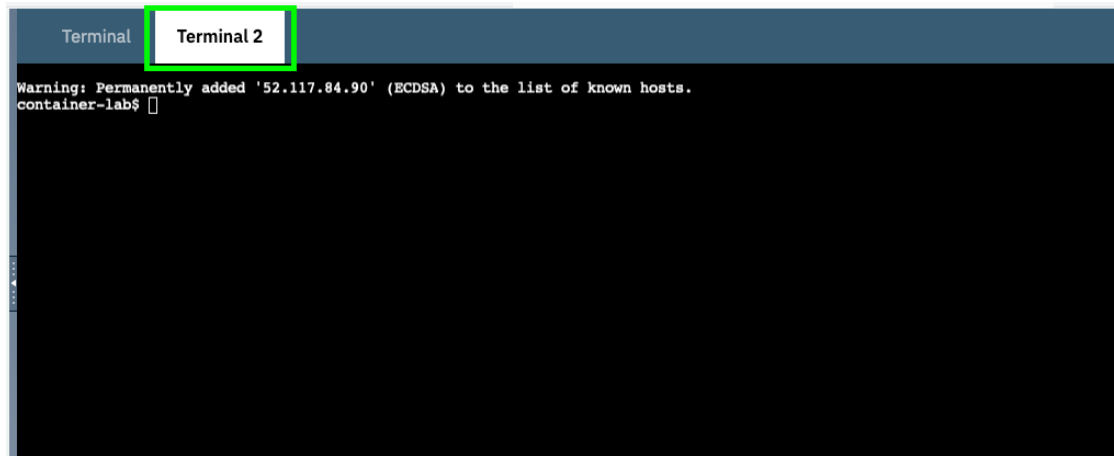
In the next exercise, you will look at the logs and events for our cluster and application. Since you just deployed the application and only accessed it once, there isn't much log data. In order to generate traffic to our application and thus create log data, we need to execute a script to access the application repeatedly. We will refer to this as the "traffic script" in future exercises.

First, since this script needs to remain running for the remainder of the lab, create a second terminal window by clicking the plus sign on the terminal title bar.

8. Click the + icon on the terminal screen.



A tab with a new terminal is opened. Switch between terminals by clicking on the appropriate tab.



9. In this new terminal, run the script to generate traffic.

Leave this script running for the next exercises and labs.

```
APP_ROUTE=`oc get routes --no-headers|cut -d' ' -f4`  
while sleep 1; do curl -s http://$APP_ROUTE/info; done
```

The output will be similar to:

```
container-lab$ while sleep 1; do curl -s http://node-s-2-i-openshift-  
example-health.andy-ocp4-3-915b3b336cabec458a7c7ec2aa7c625f-0000.us-  
south.containers.appdomain.cloud/info; done  
{  
  "personal": {  
    "name": "Ralph DAlmeida",  
    "age": 38,  
    "gender": "male",  
    "street": "34 Main Street",  
    "city": "Toronto",  
    "zipcode": "M5H 1T1"  
  },  
  "medications": [ "Metoprolol", "ACE inhibitors", "Vitamin D" ],  
  "appointments": [ "2018-01-15 1:00 - Dentist", "2018-02-14 4:00 - Internal Medicine", "2018-09-30 8:00 - Pediatrics" ]  
}  
{  
  "personal": {  
    "name": "Ralph DAlmeida",  
    "age": 38,  
    "gender": "male",  
    "street": "34 Main Street",  
    "city": "Toronto",  
    "zipcode": "M5H 1T1"  
  },  
  "medications": [ "Metoprolol", "ACE inhibitors", "Vitamin D" ],  
  "appointments": [ "2018-01-15 1:00 - Dentist", "2018-02-14 4:00 - Internal Medicine", "2018-09-30 8:00 - Pediatrics" ]  
}  
{  
  "personal": {  
    "name": "Ralph DAlmeida",  
    "age": 38,  
    "gender": "male",  
    "street": "34 Main Street",  
    "city": "Toronto",  
    "zipcode": "M5H 1T1"  
  },  
  "medications": [ "Metoprolol", "ACE inhibitors", "Vitamin D" ],  
  "appointments": [ "2018-01-15 1:00 - Dentist", "2018-02-14 4:00 - Internal Medicine", "2018-09-30 8:00 - Pediatrics" ]  
}  
{  
  "personal": {  
    "name": "Ralph DAlmeida",  
    "age": 38,  
    "gender": "male",  
    "street": "34 Main Street",  
    "city": "Toronto",  
    "zipcode": "M5H 1T1"  
  },  
  "medications": [ "Metoprolol", "ACE inhibitors", "Vitamin D" ],  
  "appointments": [ "2018-01-15 1:00 - Dentist", "2018-02-14 4:00 - Internal Medicine", "2018-09-30 8:00 - Pediatrics" ]  
}  
{  
  "personal": {  
    "name": "Ralph DAlmeida",  
    "age": 38,  
    "gender": "male",  
    "street": "34 Main Street",  
    "city": "Toronto",  
    "zipcode": "M5H 1T1"  
  },  
  "medications": [ "Metoprolol", "ACE inhibitors", "Vitamin D" ],  
  "appointments": [ "2018-01-15 1:00 - Dentist", "2018-02-14 4:00 - Internal Medicine", "2018-09-30 8:00 - Pediatrics" ]  
}
```

```

D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
{"personal":{"name":"Ralph DAlmeida","age":38,"gender":"male","street":"34
Main Street","city":"Toronto","zipcode":"M5H
1T1"},"medications":["Metoprolol","ACE inhibitors","Vitamin
D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
{"personal":{"name":"Ralph DAlmeida","age":38,"gender":"male","street":"34
Main Street","city":"Toronto","zipcode":"M5H
1T1"},"medications":["Metoprolol","ACE inhibitors","Vitamin
D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
{"personal":{"name":"Ralph DAlmeida","age":38,"gender":"male","street":"34
Main Street","city":"Toronto","zipcode":"M5H
1T1"},"medications":["Metoprolol","ACE inhibitors","Vitamin
D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
{"personal":{"name":"Ralph DAlmeida","age":38,"gender":"male","street":"34
Main Street","city":"Toronto","zipcode":"M5H
1T1"},"medications":["Metoprolol","ACE inhibitors","Vitamin
D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
{"personal":{"name":"Ralph DAlmeida","age":38,"gender":"male","street":"34
Main Street","city":"Toronto","zipcode":"M5H
1T1"},"medications":["Metoprolol","ACE inhibitors","Vitamin
D"],"appointments":["2018-01-15 1:00 - Dentist","2018-02-14 4:00 - Internal
Medicine","2018-09-30 8:00 - Pediatrics"]}]
...

```

Leave this infinite loop running in the second terminal.

End Exercise 3 __

Exercise 4

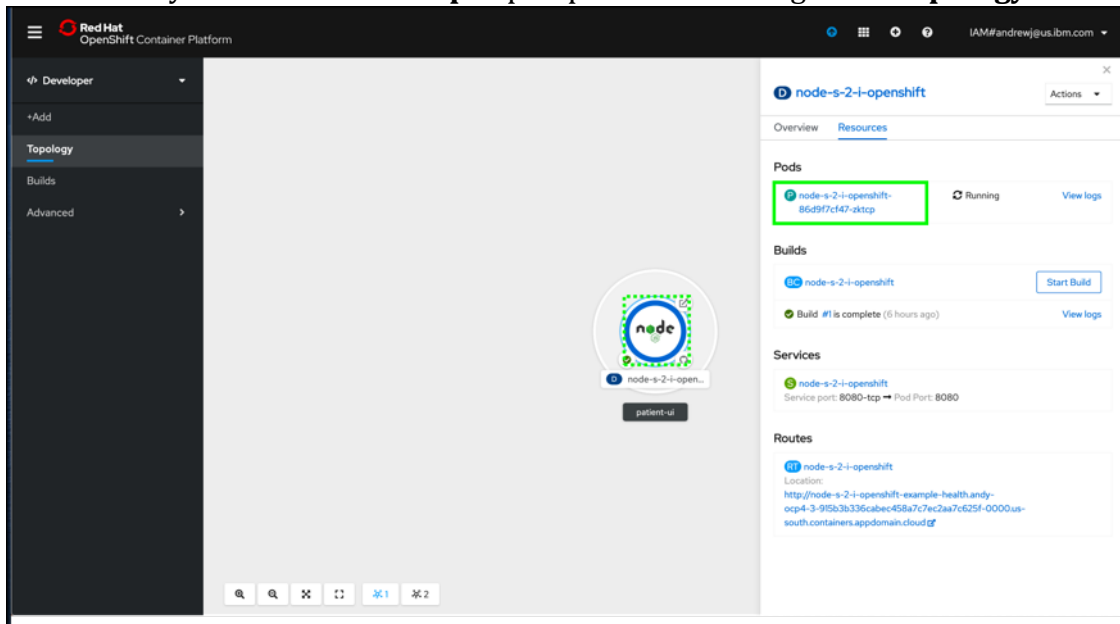
Logging and events

Exercise 4

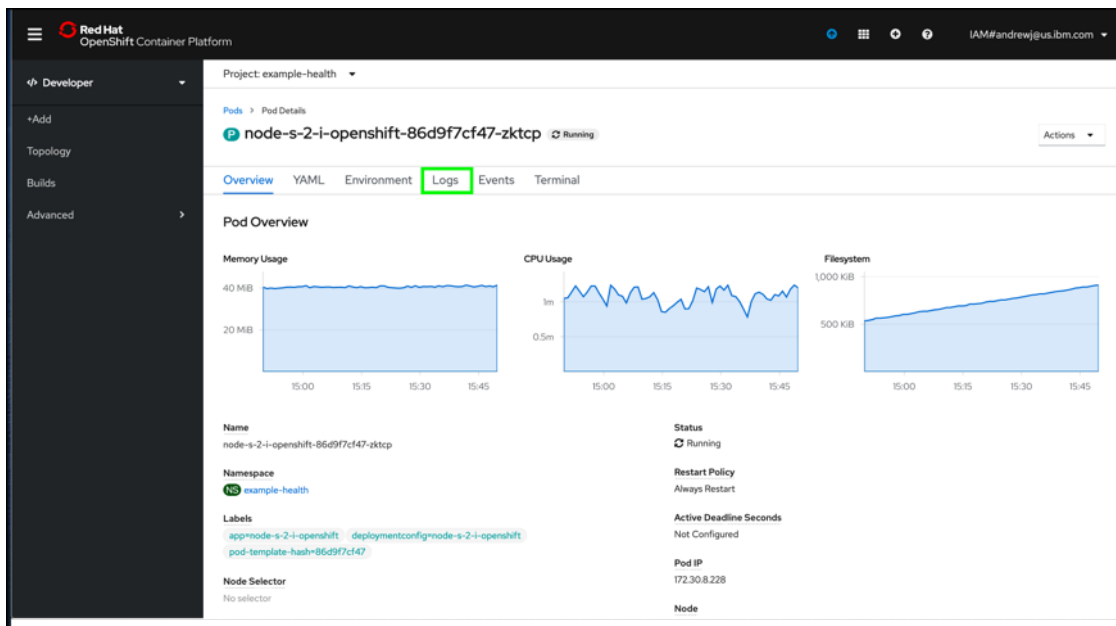
In this exercise, explore the out-of-the-box logging and monitoring capabilities that are offered in OpenShift.

Since we only created one pod, seeing our logs will be straight forward.

1. Ensure that you're in the **Developer** perspective and navigate to **Topology**.

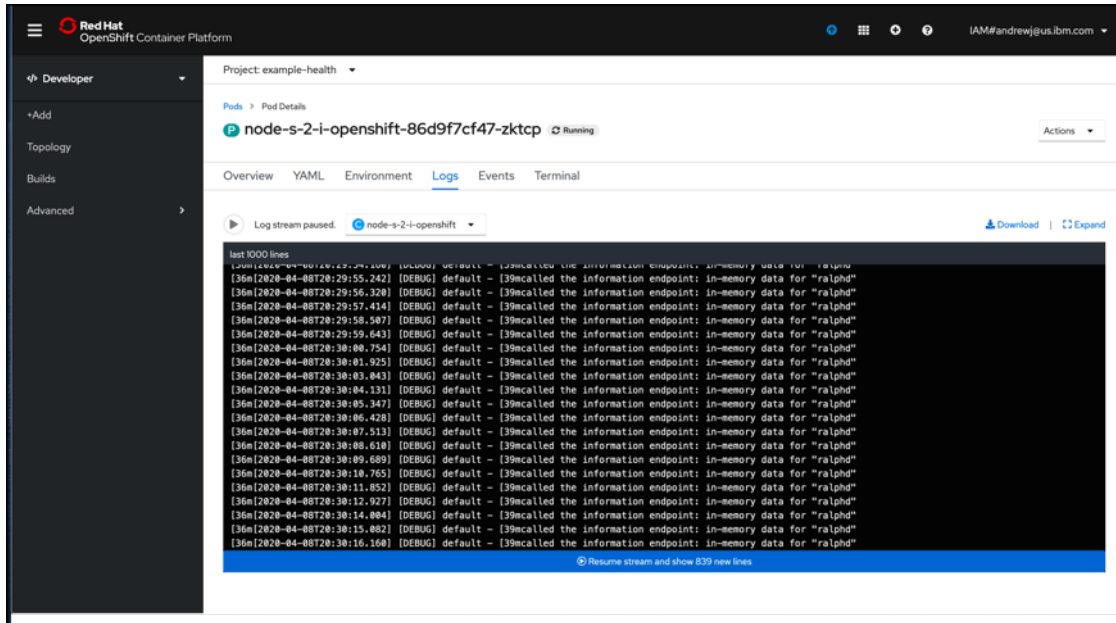


2. Open the details panel for the application and click the link under **Pods**.



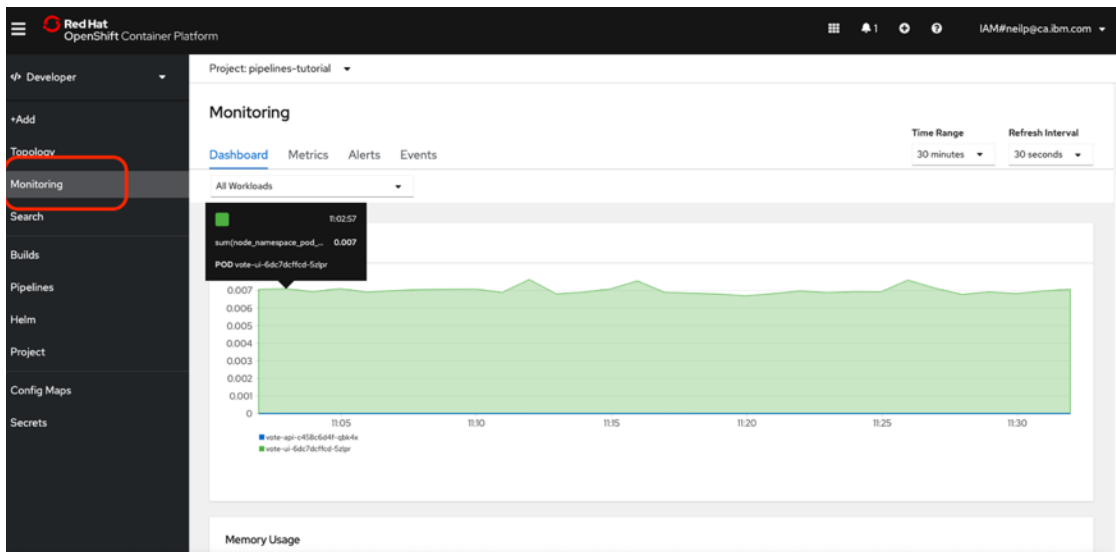
3. Click the **Logs** tab.

The **Logs** page streams information logged from your running application. If you're still generating traffic from the previous exercise, you should see log messages appearing for every request being made.



When deploying new apps, making configuration changes, or simply inspecting the state of your cluster, the project-scoped events dashboard gives can provide additional insights.

4. Access the Dashboard now by going to the **Monitoring > Events** on the left side menu.



The **Events** view is useful for identifying the timeline of events and finding potential error messages. When tracking the state of a new rollout, managing existing assets, or even something simple like exposing a route, the Events view is critical in identifying the timeline of activity. This becomes even more useful when considering that multiple operators may be working against a single cluster. You will learn more about operators in Lab 2. You can filter on the view on different types and categories of events.

You'll want to refer to the Events view throughout the lab. Almost all actions taken in OpenShift will result in an event being fired. As it is updated in real-time, it's a great way to track changes to state.

End Exercise 4 ____

Exercise 5

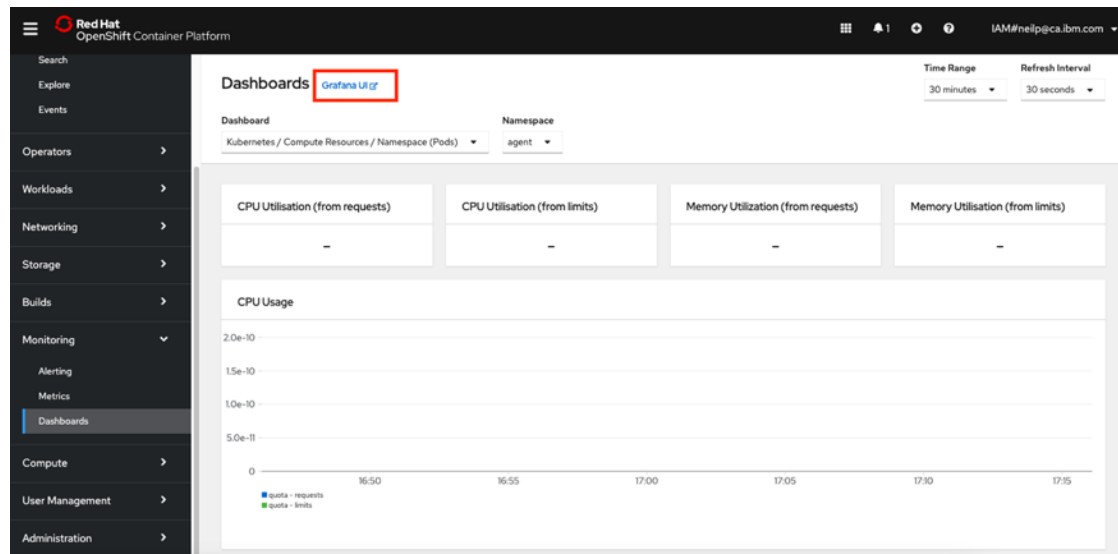
Metrics and Dashboards

Exercise 5

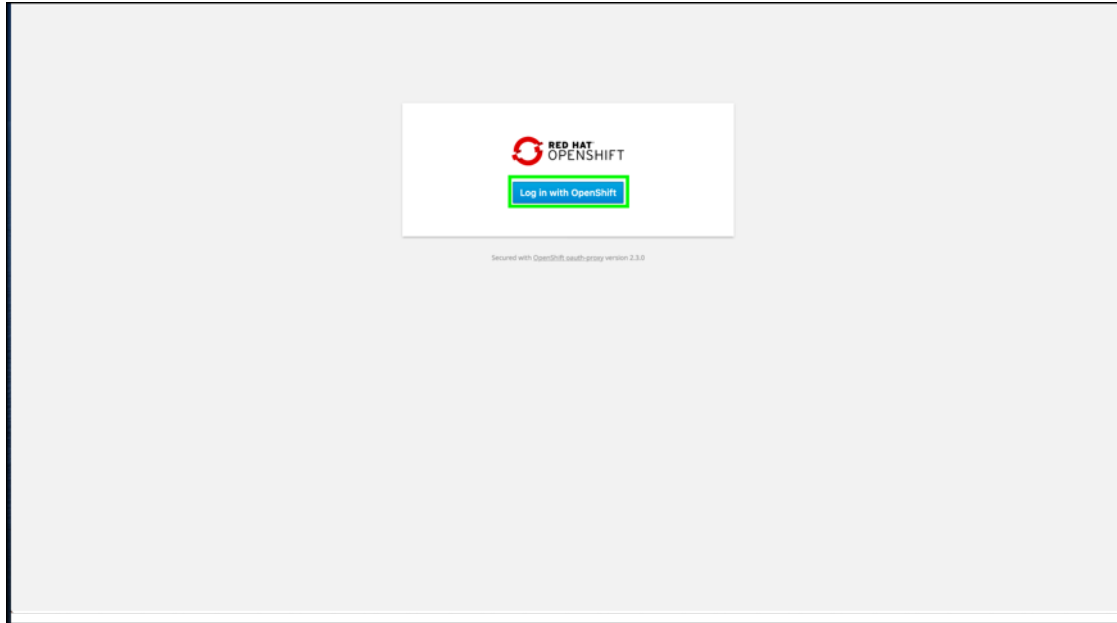
In this exercise, explore the third-party monitoring and metrics dashboards that are installed for free with OpenShift.

Red Hat OpenShift on IBM Cloud comes with Grafana preinstalled, which is a great tool to help you monitor the health of your applications and systems. You can learn more about Grafana [here](#).

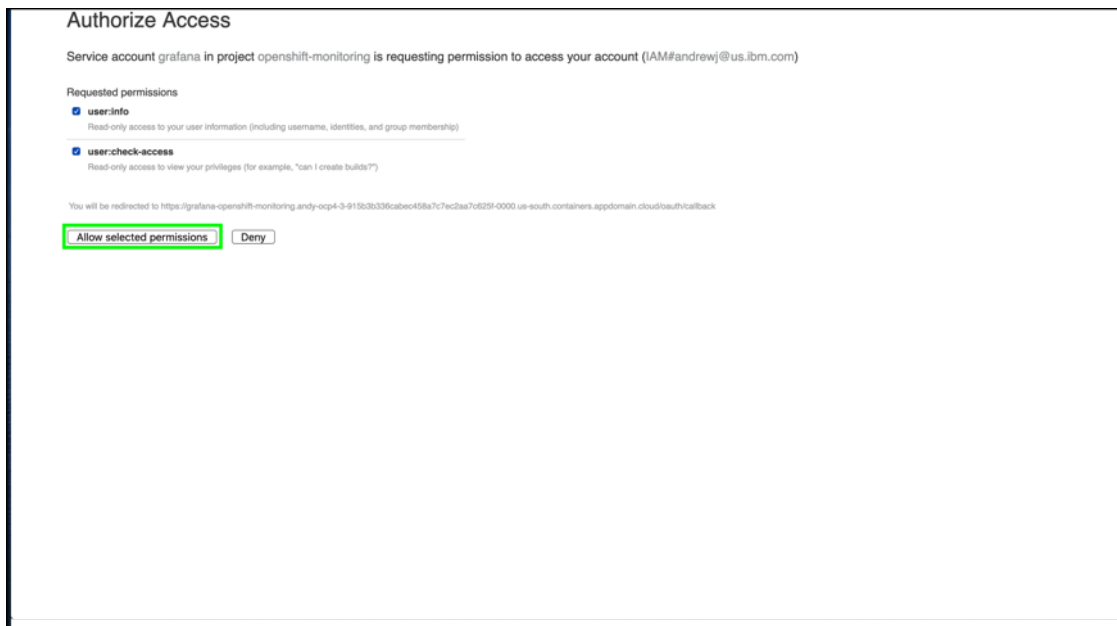
1. Switch to the **Administrator perspective**.
2. Navigate to **Monitoring > Dashboards** in the left bar.



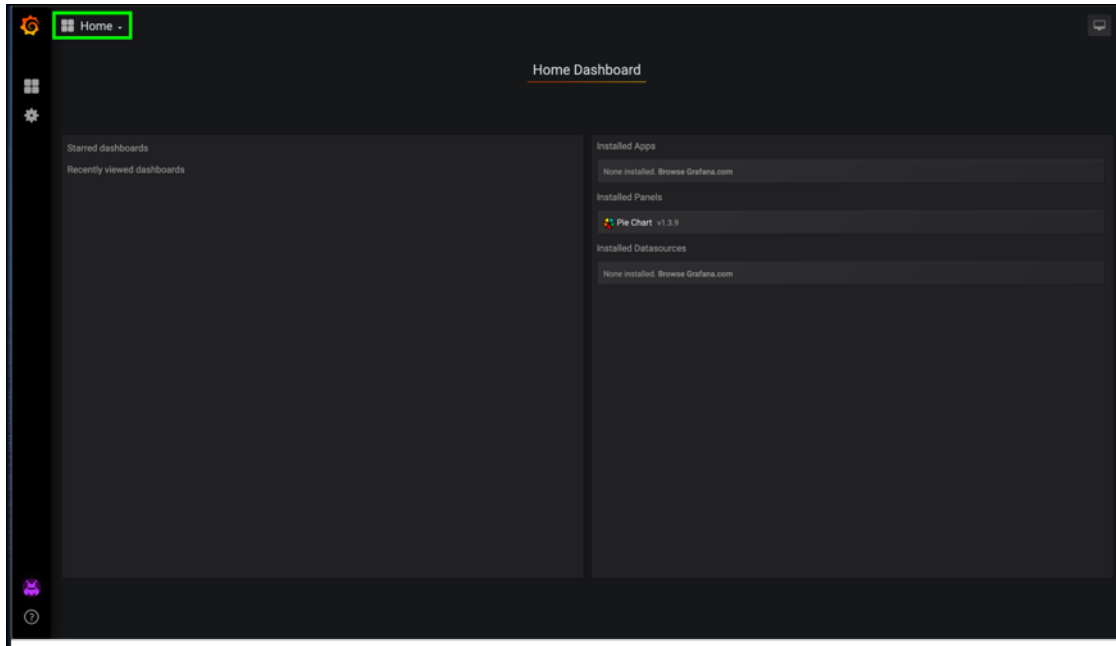
3. Click **Grafana UI**.



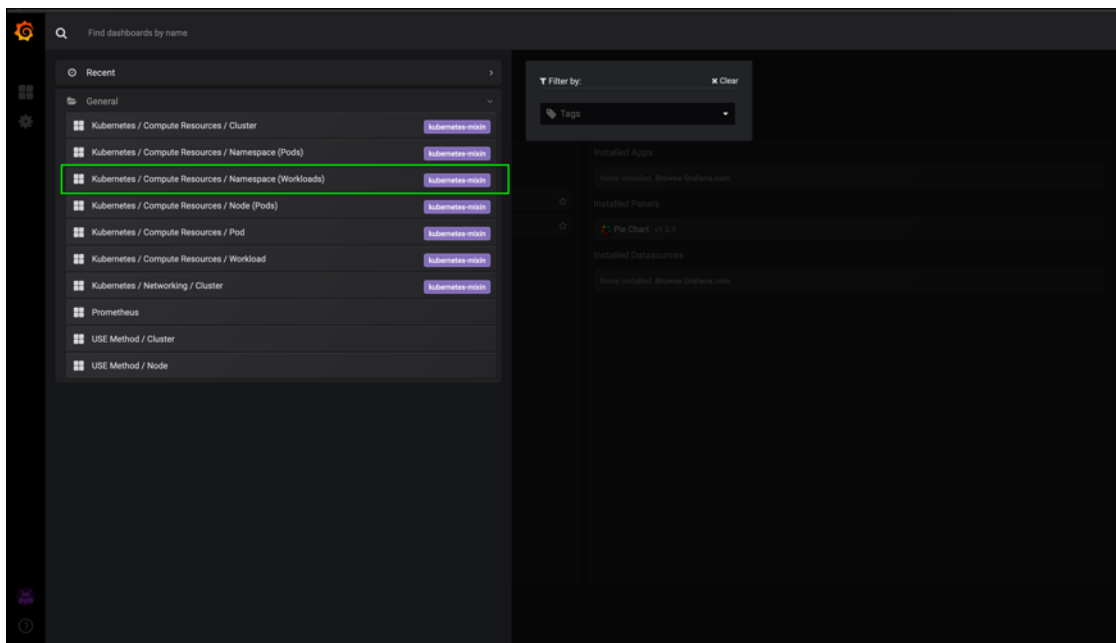
4. Click **Login with OpenShift**.



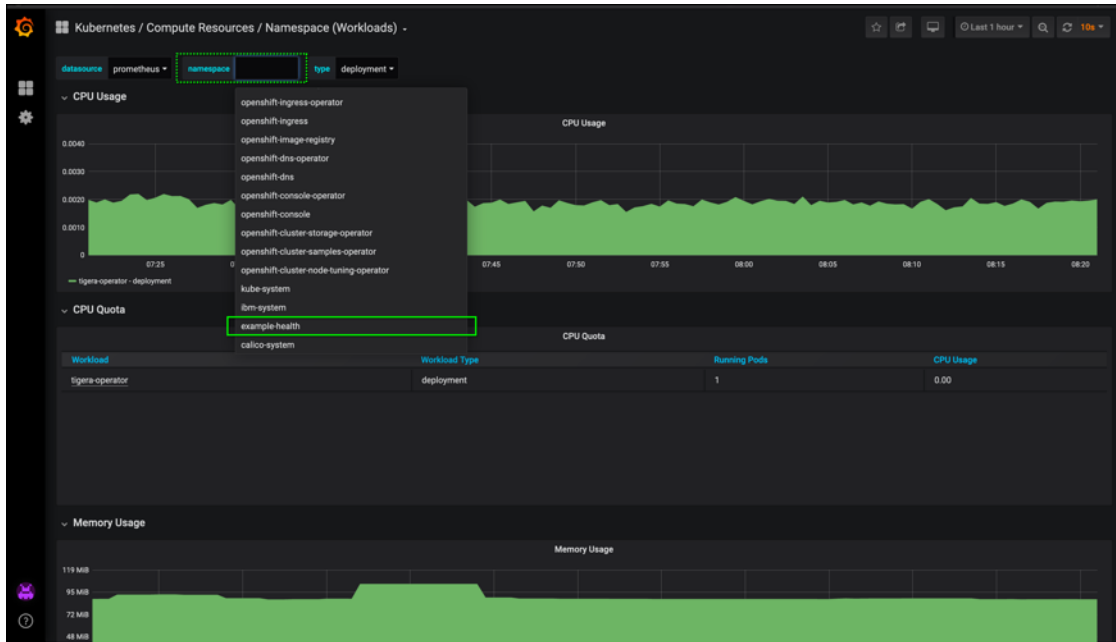
5. Click **Allow selected permissions**.
6. This will allow the Grafana service account access to the openshift-monitoring project in your cluster.



6. Click **Home** on the Grafana landing page.



7. Select the **Kubernetes / Compute Resources / Namespace (Workloads)** view from the drop-down menu.



8. Switch to the **example-health** namespace.

You should be able to see the CPU and Memory usage for your application. In production environments, this is helpful for identifying the average amount of CPU or Memory your application uses, especially as it can fluctuate throughout the day. We'll use this information in the next exercise to set up auto-scaling for our pods.

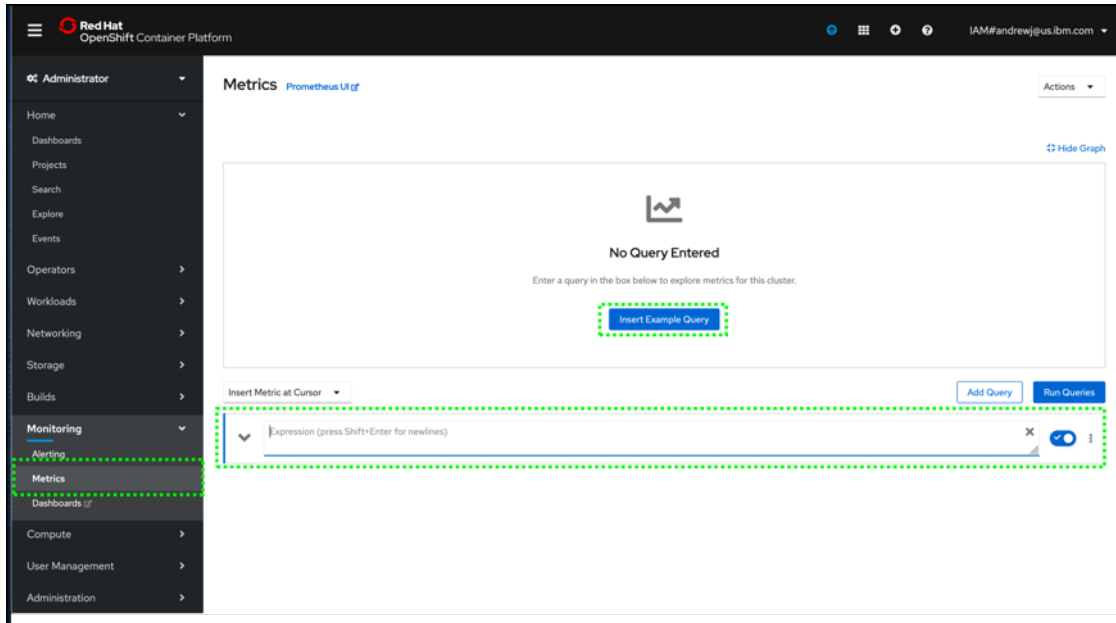
Take note of the CPU and Memory usage of for the application. If your traffic script is still running from Exercise 3, you should observe fairly steady CPU and memory usage.

OpenShift also utilizes Prometheus and Alert Manager. You can learn more about these here:

Prometheus - provides monitoring of cluster components and ships with a set of alerts to notify the cluster administrator about problems

Alertmanager - an extension of Prometheus focused on managing alerts

OpenShift provides a web interface to Prometheus, which enables you to run Prometheus Query Language (PromQL) queries and examine the metrics visualized on a plot. This functionality provides an extensive overview of the cluster state and enables you to troubleshoot problems. Look around and try the **Insert Example Query** by using the **Insert Metric at Cursor** pull-down menu.



End Exercise 5 ____

Exercise 6

Scaling the application

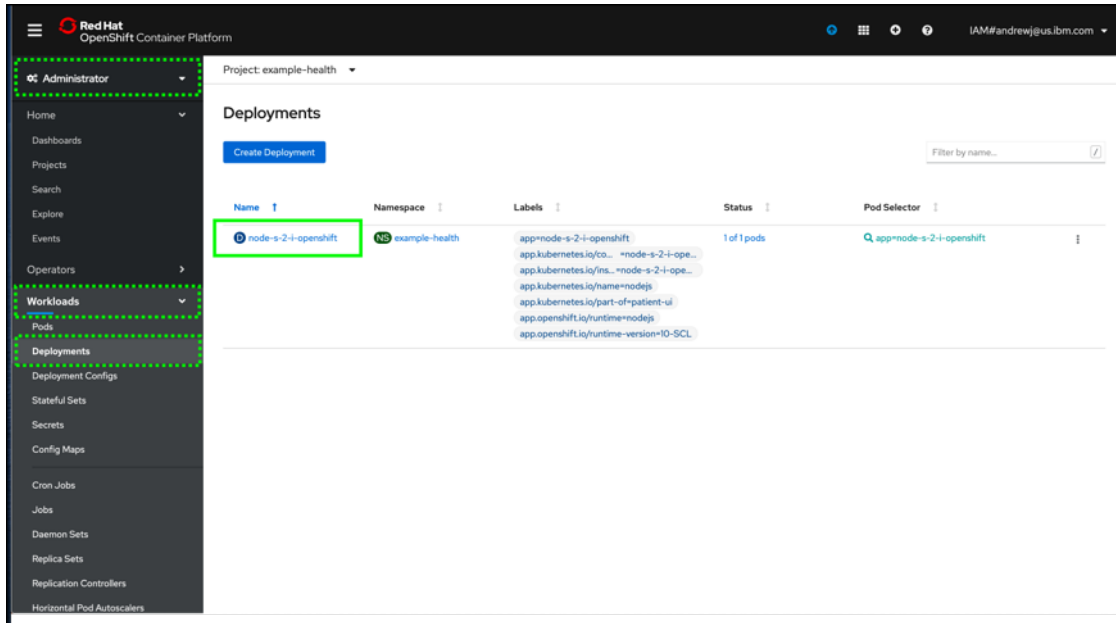
Exercise 6

In this exercise, we'll leverage the metrics we've observed in the previous exercises to automatically scale our front-end application in response to load.

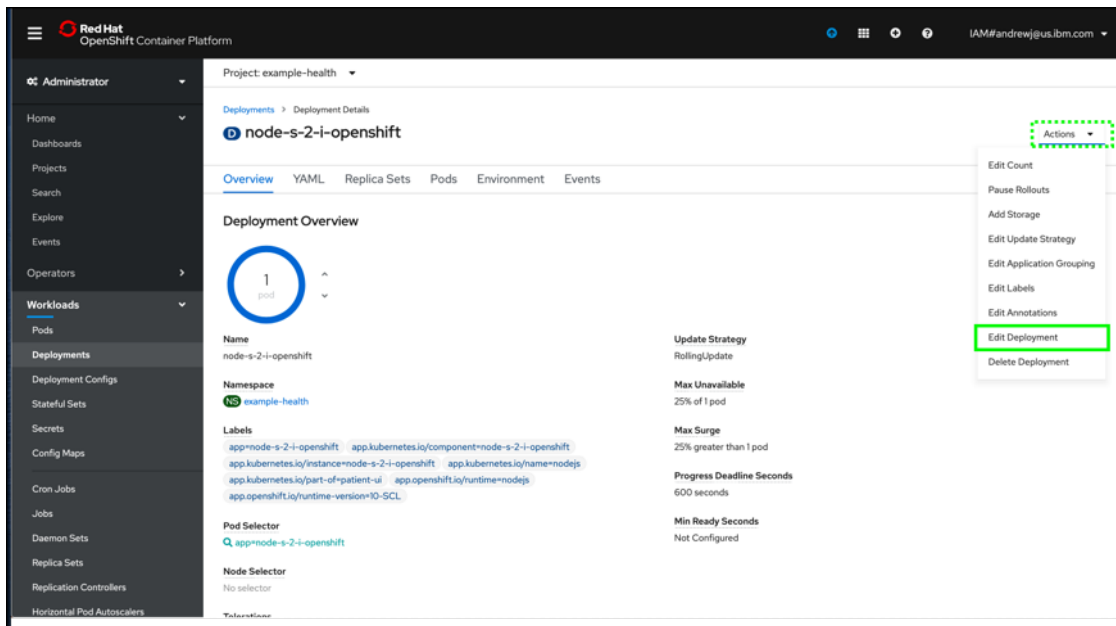
Before we can set up autoscaling for our pods, we first need to set resource limits on the pods running in our cluster. Limits allow you to choose the minimum and maximum CPU and memory usage for a pod. You can learn more about resources and limits [here](#).

Grafana showed you that your application was consuming anywhere between “.002” to “.02” cores. This translates to 2-20 “millicores”. That seems like a good range for our CPU request, but to be safe, let's bump the higher-end up to 30 millicores. In addition, Grafana showed that the app consumes about 25-35 MB of RAM. Set the following resource limits for your deployment now.

1. In the **Administrator** perspective, navigate to **Workloads > Deployments**.
2. Select the **patient-ui** deployment by clicking the **node-s-2-i-openshift** deployment link.



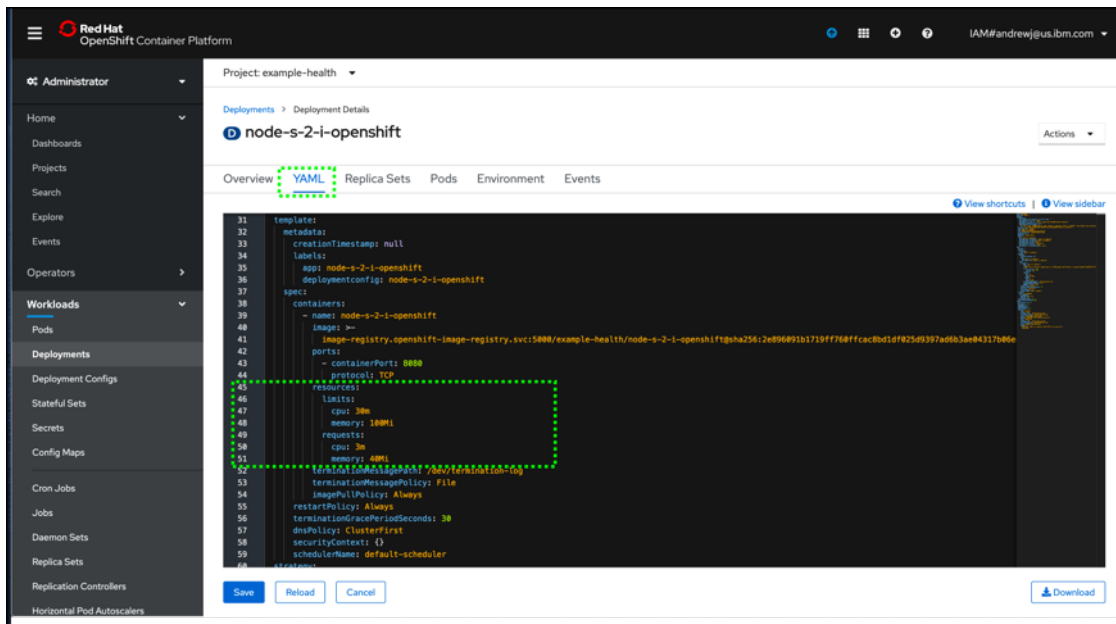
3. Edit the deployment by selecting **Actions > Edit Deployment**.



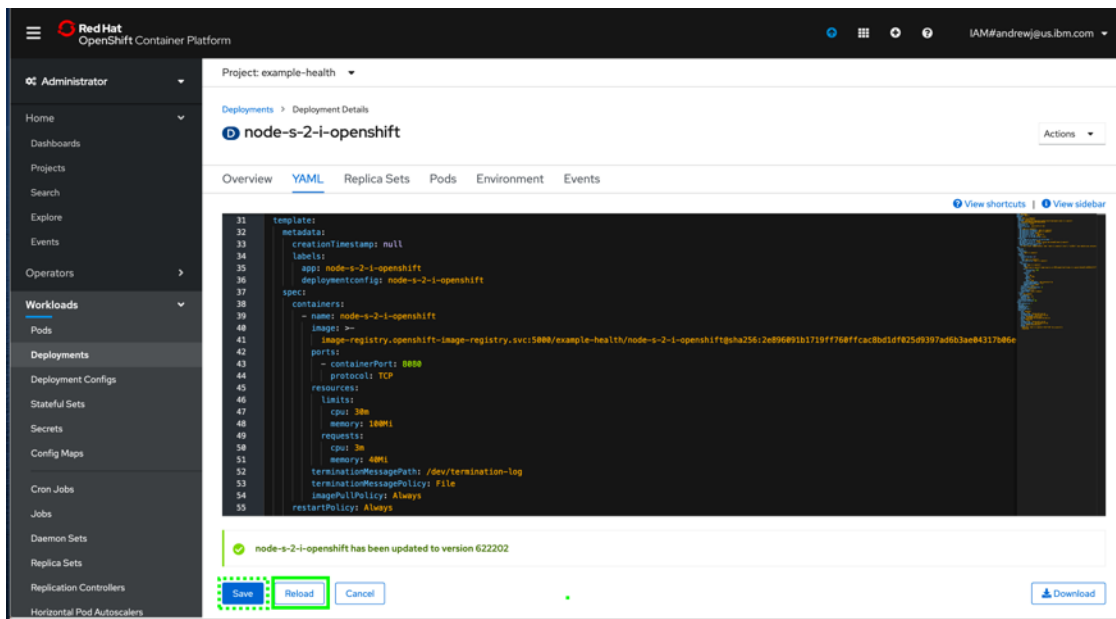
4. In the YAML editor, at approximately line 45 notice the **resources** section under **template > spec > containers**. Replace the resources {} with:

```
resources:
  limits:
    cpu: 30m
    memory: 100Mi
  requests:
    cpu: 3m
    memory: 40Mi
```

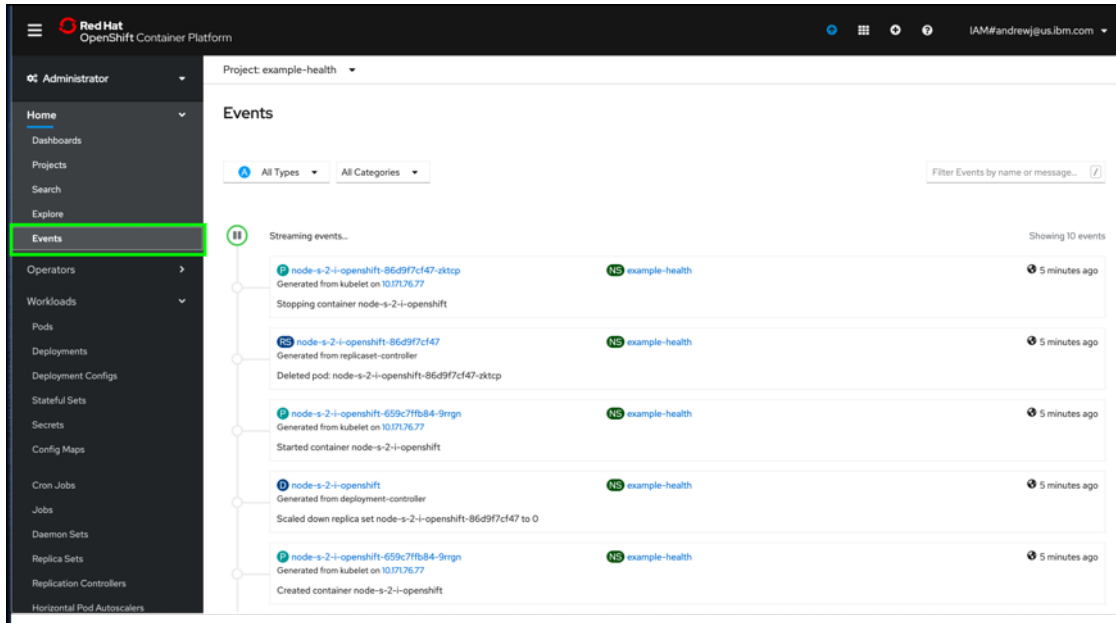

Note: YAML uses strict indentation. Be sure the spacing is correct.



5. Save and Reload to see the new version.



6. Verify that the replication controller has been changed by navigating to Events by observing the events related to the **node-s-2-i-openshift** deployment stopping and starting:



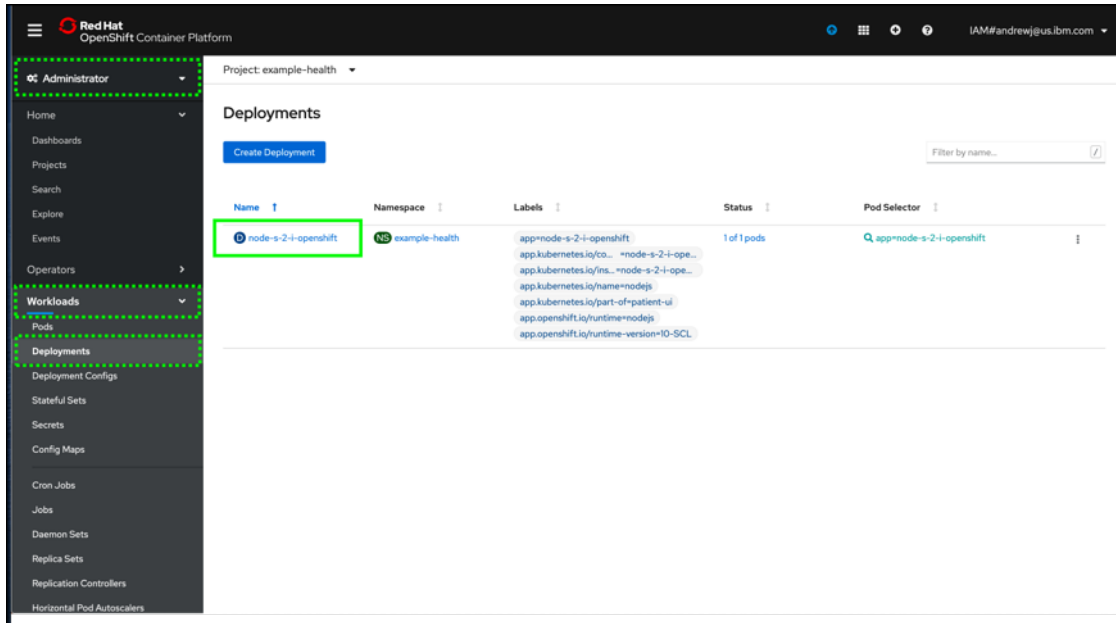
Enable Autoscaler

Now that we have set resource limits, let's enable Autoscaler.

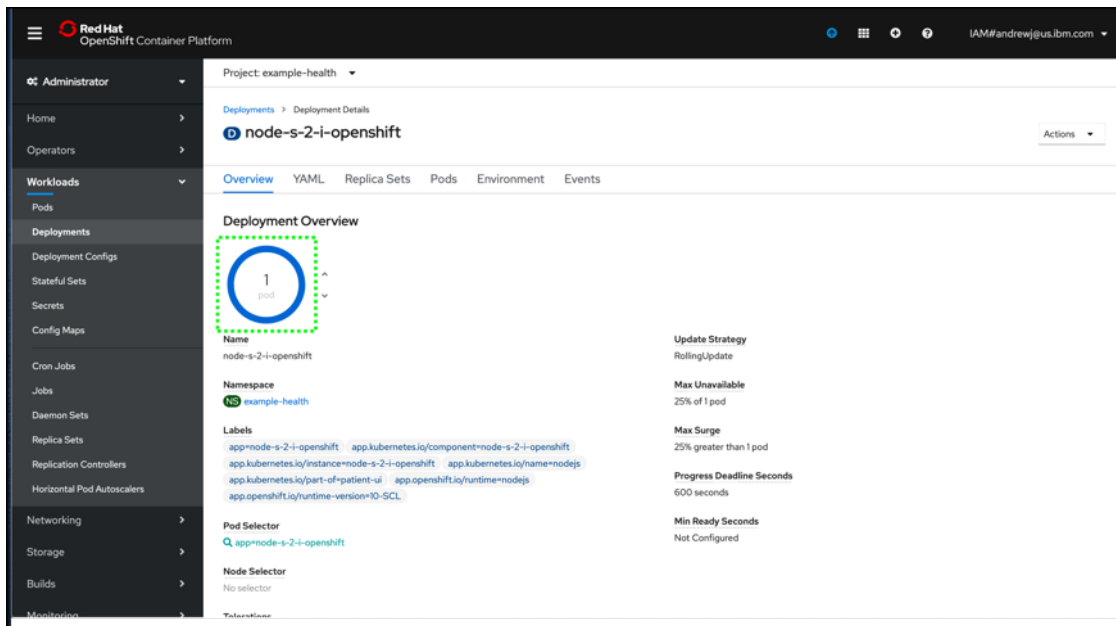
By default, Autoscaler allows you to scale based on CPU or Memory. The UI allows you to configure CPU only (for now). Pods are balanced between the minimum and maximum number of pods that you specify. With Autoscaler, pods are automatically created or deleted to ensure that the average CPU usage of the pods is below the CPU request target as defined. In general, you probably want to start scaling up when you exceed 50% of the CPU usage of a pod. In our case, let's make it 1% to test Autoscaler since we are generating minimal load. You can learn more Autoscaling [here](#).

Before we add the autoscaler, verify that there is currently only 1 pod running for our example-health application.

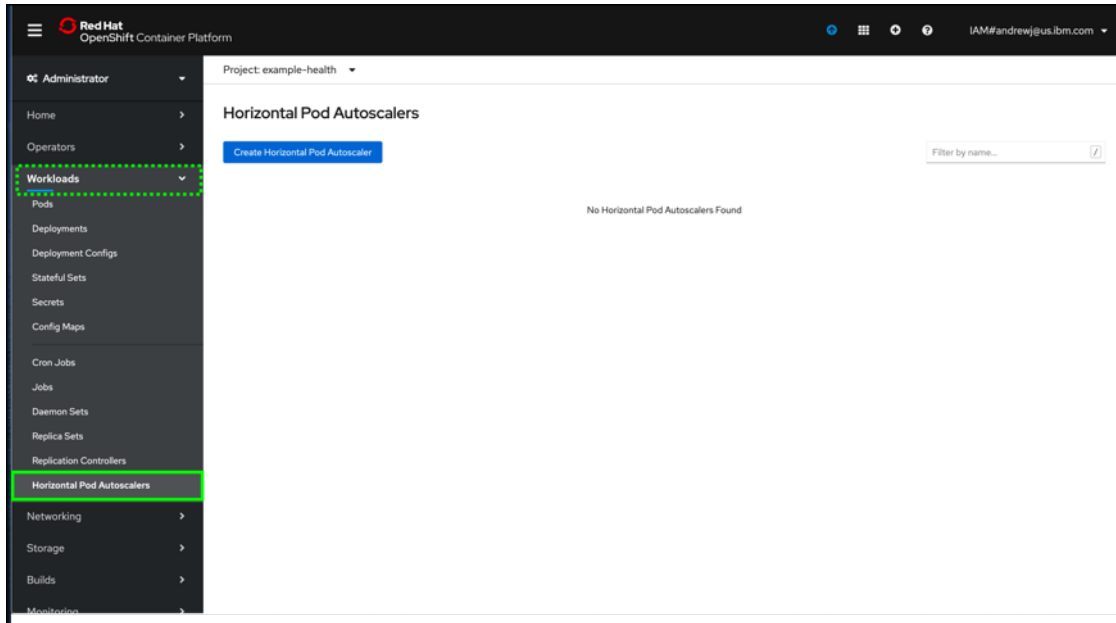
7. Click the **node-s-2-i-openshift** deployment in the **Workloads > Deployments** page.



8. Verify there is 1 pod running.



9. Navigate to **Workloads > Horizontal Pod Autoscalers**.

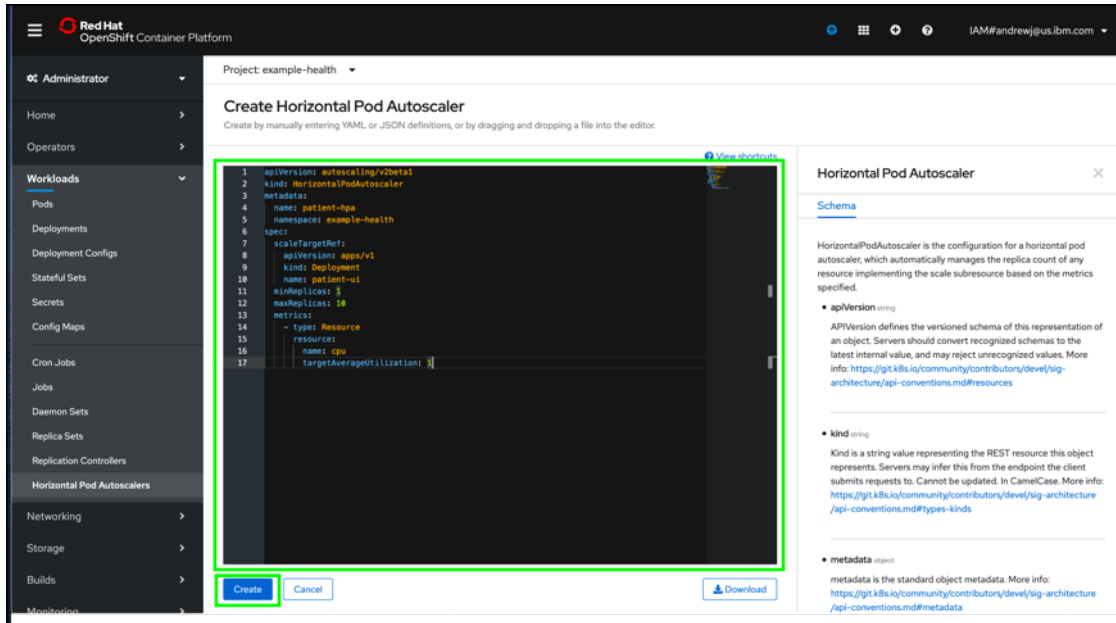


10. Click **Create Horizontal Pod Autoscaler**.

11. Replace the entire default YAML with the following new YAML.

Take note of the values we are setting for minReplicas, MaxReplicas, and targetAverageUtilization for CPU. The value for targetAverageUtilization is artificially low, but will allow us to see the scaling go into effect provided our traffic script is still running from **Exercise 3**.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: patient-hpa
  namespace: example-health
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: node-s-2-i-openshift
  minReplicas: 1
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        targetAverageUtilization: 1
```



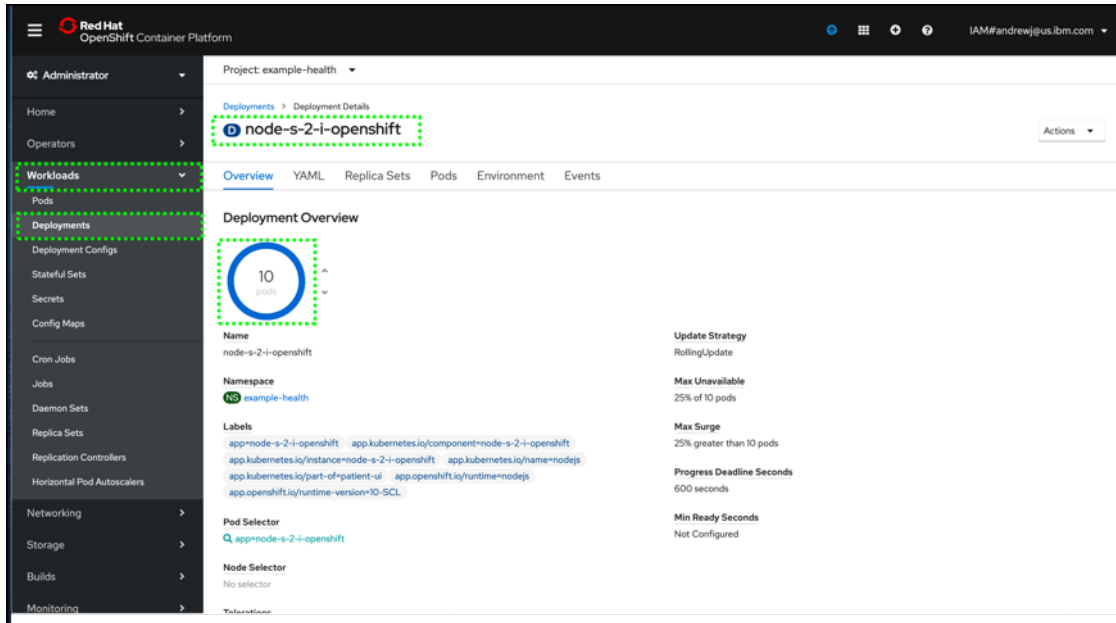
Note: If you changed the **Name** field of your deployment when you deployed the patient-ui app, change the **name:** attribute in the YAML to match. This example uses **node-s-2-i-openshift**.

12. Click **Create**.

13. Verify Autoscaler is working by returning to the **node-s-2-i-openshift** deployment in the **Workloads > Deployments** page and selecting the **node-s-2-i-openshift** link.

After a few minutes, you should see the number of pods increase.

Note: It will take a few minutes for metrics to come and for the scaling to begin. You may see error events in the Autoscaler Events log initially. If, after 5 minutes, the scaling has not occurred, verify your traffic script from **Exercise 3** is still running and there are no other errors with Autoscaler.



If you're not running the traffic script from **Exercise 3**, the number of pods will stay at 1.

That's it! You now have a highly available and automatically scaled front-end Node.js application. OpenShift is automatically scaling your application pods since the CPU usage of the pods greatly exceeded 1% of the resource limit of 30 millicores.

End Exercise 6

Next Steps

Next steps

Congratulations! You have completed the introductory lab for Red Hat OpenShift on IBM Cloud.

In this lab, you learned how to:

- Deploy an application
- Use the OpenShift command line interface
- Locate logs and events that are generated by your application deployment
- Find metrics and dashboards regarding your application
- Configure auto scaling of your application