

Introduction to Red Hat OpenShift



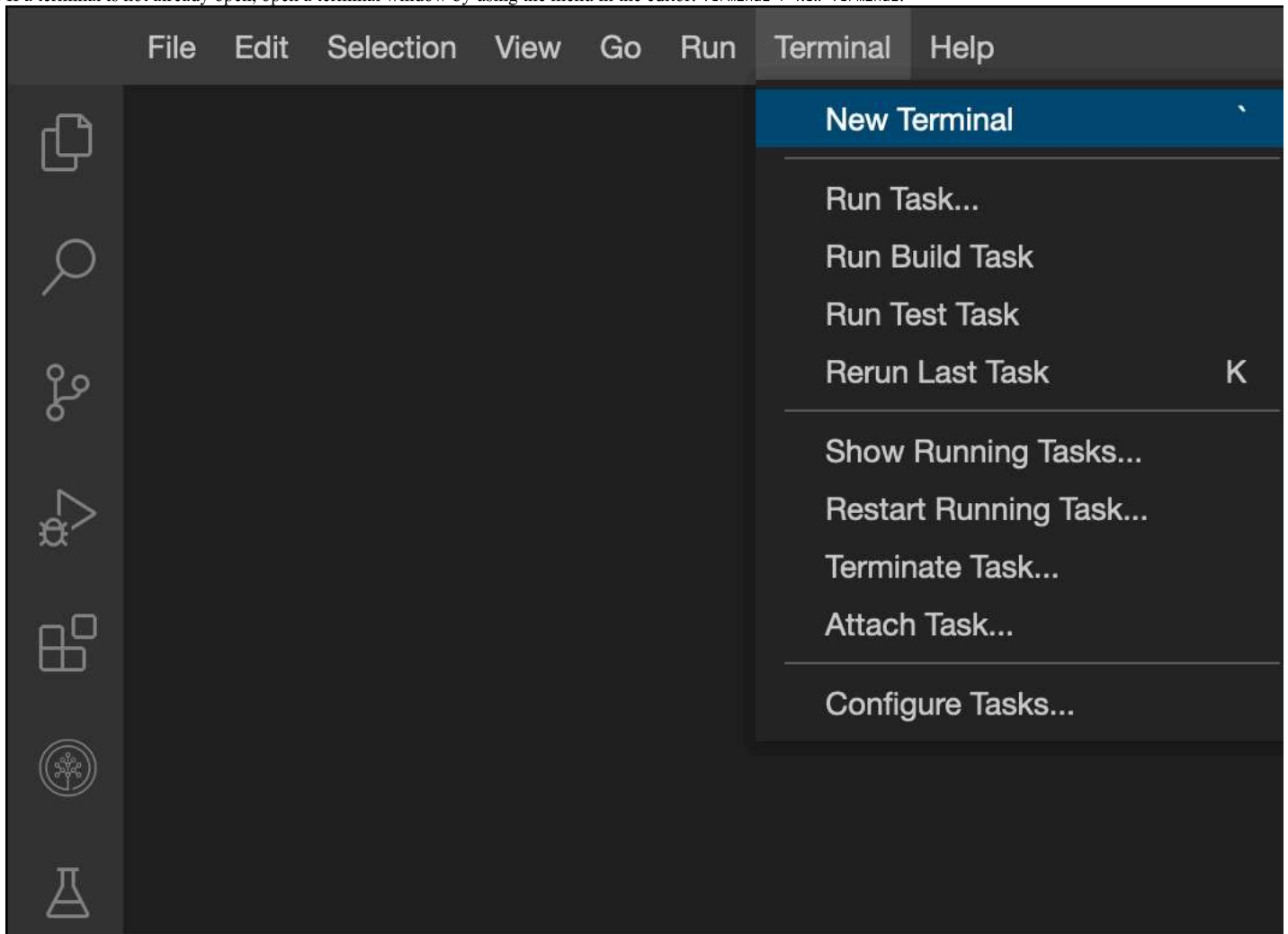
Objectives

In this lab, you will:

- Use the oc CLI (OpenShift command line interface)
- Use the OpenShift web console
- Build and deploy an application using s2i ('Source-to-image' build strategy)
- Inspect a BuildConfig and an ImageStream
- Autoscale the application

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. Verify that oc CLI is installed.

```
oc version
```

```
theia@theiaopenshift-...:/home/project$ oc version
Client Version: 4.9.0
Kubernetes Version: v1.21.8+ee73ea2
```

You should see output similar to this, although the versions may be different.

3. Change to your project folder.

NOTE: If you are already on home/project please skip this step

```
cd /home/project
```

4. Clone the git repository that contains the artifacts needed for this lab, if it doesn't already exist.

```
[ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.git
```

```
theia@theiaopenshift-: /home/project$ [ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.git
Cloning into 'CC201'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 20 (delta 6), reused 19 (delta 6), pack-reused 0
Unpacking objects: 100% (20/20), done.
```

Use the oc CLI

OpenShift projects are Kubernetes namespaces with additional administrative functions. Therefore, projects also provide isolation within an OpenShift cluster. You already have access to one project in an OpenShift cluster, and oc is already set to target that cluster and project.

Let's look at some basic oc commands. Recall that oc comes with a copy of kubectl, so all the kubectl commands can be run with oc.

1. List the Pods in this namespace.

```
oc get pods
```

```
theia@theiaopenshift-: /home/project$ oc get pods
NAME                                READY   STATUS    RESTARTS   AGE
openshift-web-console-995896df-vz2tp 2/2     Running   0           4h1m
```

You will likely see a few Pods that are part of the environment. You don't need to worry about these.

2. In addition to Kubernetes objects, you can get OpenShift specific objects.

```
oc get buildconfigs
```

```
theia@theiaopenshift-: /home/project$ oc get buildconfigs
No resources found in sn-labs- namespace.
```

Because you haven't created a BuildConfig yet, this will not return any resources.

3. View the OpenShift project that is currently in use.

```
oc project
```

```
theia@theiaopenshift-: /home/project$ oc project
Using project "sn-labs-" from context named "-context" on server "https://c109-e.us-east.containers.cloud.ibm.com:30807".
theia@theiaopenshift-: /home/project$
```

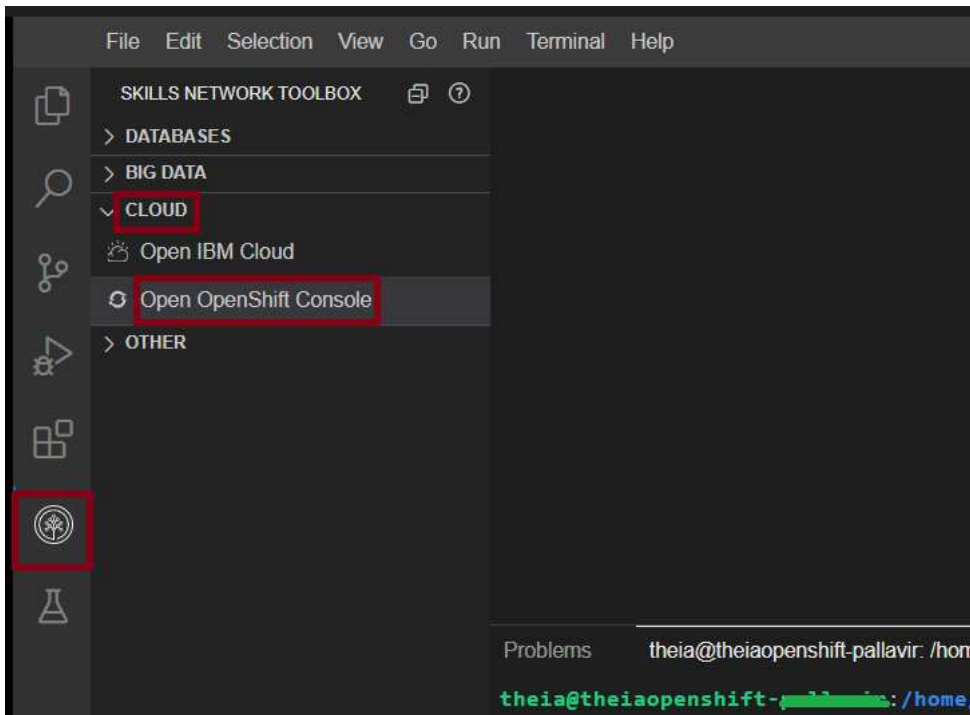
This project is specific to you and provides isolation within the cluster so that you can deploy your own applications.

Use the OpenShift web console

In addition to the CLI, OpenShift provides an intuitive web console. This is a useful and powerful feature because it enables you to deploy applications, view resources, monitor applications and view logs, and much more right in the console.

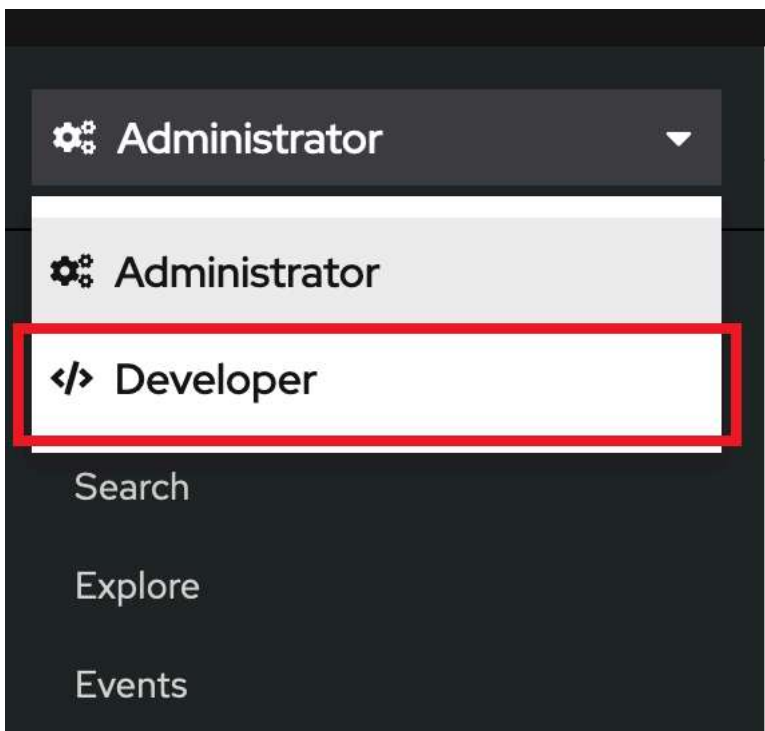
Let's open up the console and have a look around.

1. To open openshift web console, click on the Skills Network button on the right, it will open the **Skills Network Toolbox** Then click the **Cloud** then **Open OpenShift console** as shown in the following image.



It can take a few minutes to become available after opening the lab environment, so if you get an error, wait a minute and try again.

2. The console should open to the project details for the project you have been assigned. Take a look at all the information OpenShift provides you in an intuitive, visual manner. Click through the Dashboard, Overview, and other tabs for this project to see additional information. You should see inventory on the resources that currently exist in this project, the YAML that defines this project, and much more.
3. Familiarize yourself with the items in the left navigation menu. You can see Operators, many different Kubernetes objects, and some OpenShift-specific objects, all of which we have talked about in this course. There won't yet be many instances of these objects, but they will fill up once we deploy our application.
4. Notice the word “Administrator” at the top left. This indicates that you are in the Administrator perspective. There is also a Developer perspective. Each perspective provides workflows specific to that persona. **Switch to the Developer perspective** to begin deploying an application. (If it says “Developer” already, don't change it.)



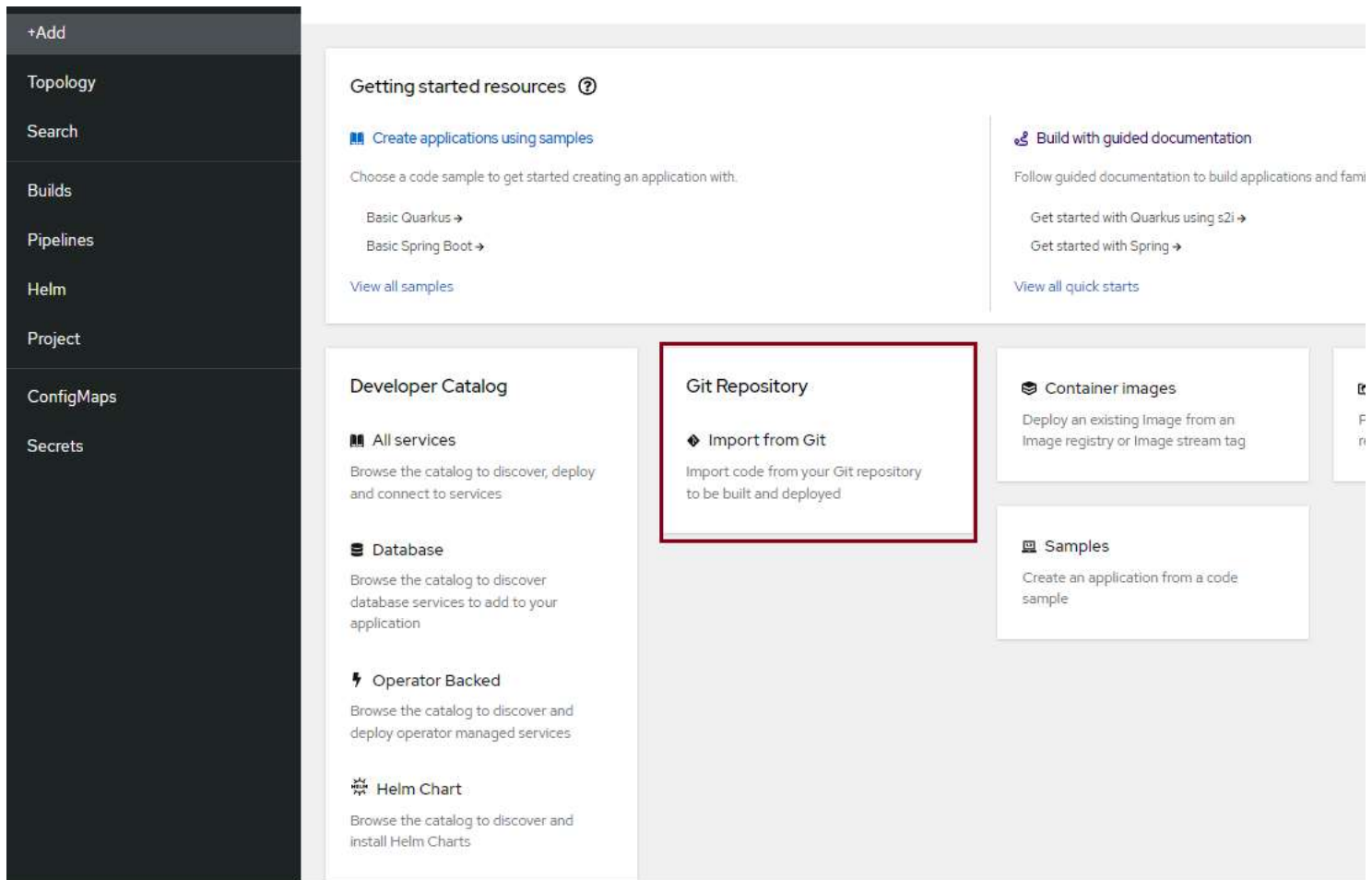
Deploy an application in the web console

The Developer perspective provides workflows specific to developer use cases, such as the ability to create and deploy applications. Let's start here! You are likely in the “Topology” view, which provides a visual representation of applications. If not, switch to it to take a look.

1. Let us add a new application to this project. There are several ways to add a new application in Openshift.

2. Click the **+Add** button to add a new application.

3. Select **Git Repository (Import from Git)** among the options.



4. You will be redirected to **Import from Git** window. OpenShift will deploy an application using only one input from you: the application source.

5. In the **Git Repo URL** box, paste the sample one mentioned below.

`https://github.com/sclorg/nodejs-ex.git`

In the Builder section, scroll down to see the various builder images. We shall be using the Node.js image for our application. Ensure that this image has been selected.

Skills Network OpenShift Lab

Developer

+Add

Topology

Monitoring

Search

Builds

Pipelines

Helm

Project

Config Maps

Secrets

Project: sn-labs-

Import from Git

Git

Git Repo URL *

https://github.com/sclorg/nodejs-ex.git

Validated


Show Advanced Git Options


Builder


Builder Image


Builder image(s) detected.


Recommended builder images are represented by ★ icon.

Perl

PHP


Nginx

Httpd

.NET

Builder Image Version *

IST 14-ubi7

 Node.js 14 (UBI 7)
BUILDER NODEJS

Build and run Node.js 14 applications on UBI 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-nodejs-container/blob/master/14/README.md>.

Sample repository: <https://github.com/sclorg/nodejs-ex.git>

General

Application

Select an application for your grouping or no application group to not use an application grouping.

Name *

nodejs-ex-git

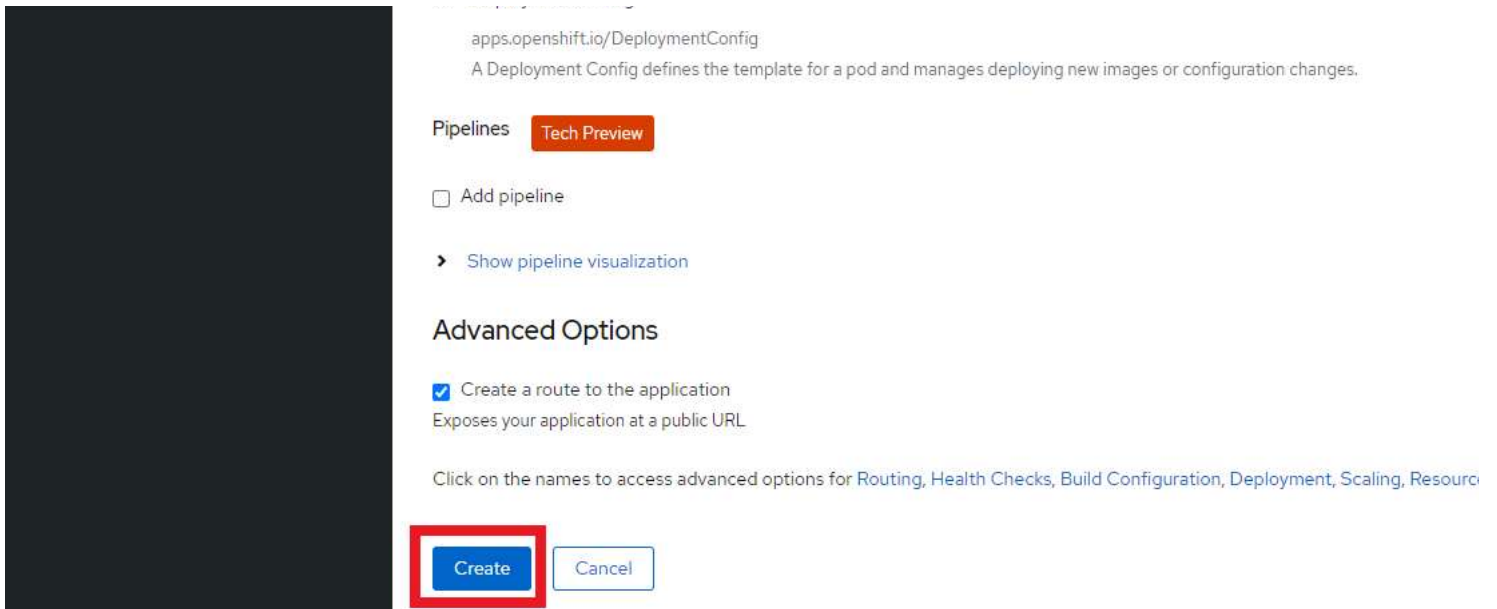
A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

☒ Deployment
apps/Deployment
A Deployment enables declarative updates for Pods and ReplicaSets.

☐ Deployment Config



apps.openshift.io/DeploymentConfig

A Deployment Config defines the template for a pod and manages deploying new images or configuration changes.

Pipelines **Tech Preview**

☐ Add pipeline

› [Show pipeline visualization](#)

Advanced Options

☒ Create a route to the application
Exposes your application at a public URL

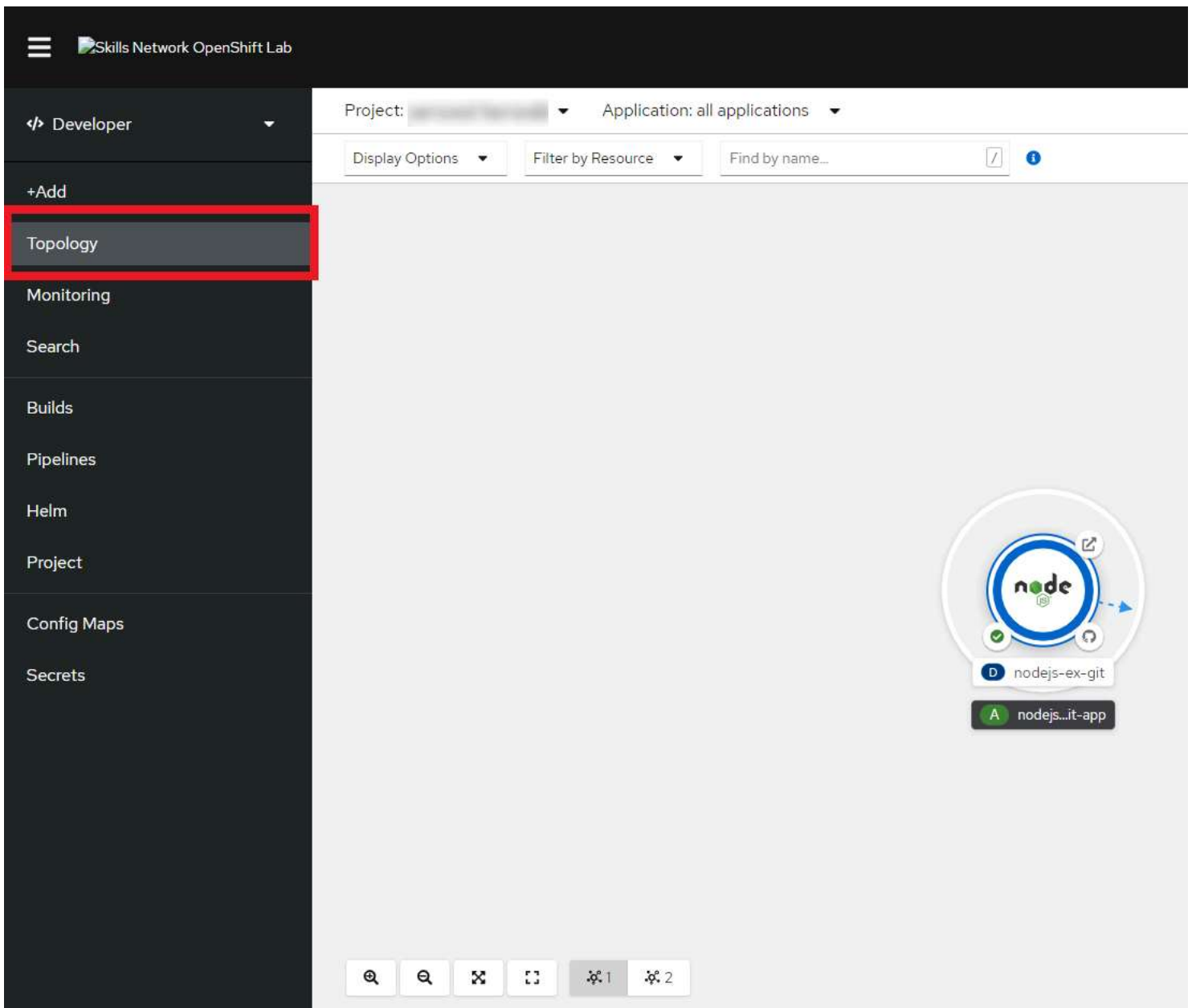
Click on the names to access advanced options for [Routing](#), [Health Checks](#), [Build Configuration](#), [Deployment](#), [Scaling](#), [Resource](#)

Create Cancel

6. Keep the rest of the default options as they already are. Then scroll down and click **Create**.

In the Topology view, you should now see your newly created application.

NOTE: It will take several minutes for the application to appear. Refresh the browser if within 3 minutes, you don't see any application.



View application in the web console

The Topology view provides quick links to a lot of important parts of an application:

- The outer circle gets the information on the application.
- The inner circle with the Node.js logo gives information about the Deployment.
- The GitHub icon is used to access the code repository.
- The check mark shows the most recent build (you will see circular arrows if the build is in progress).
- The arrow coming out of a box can be used to view the application in the browser if the application is externally available.

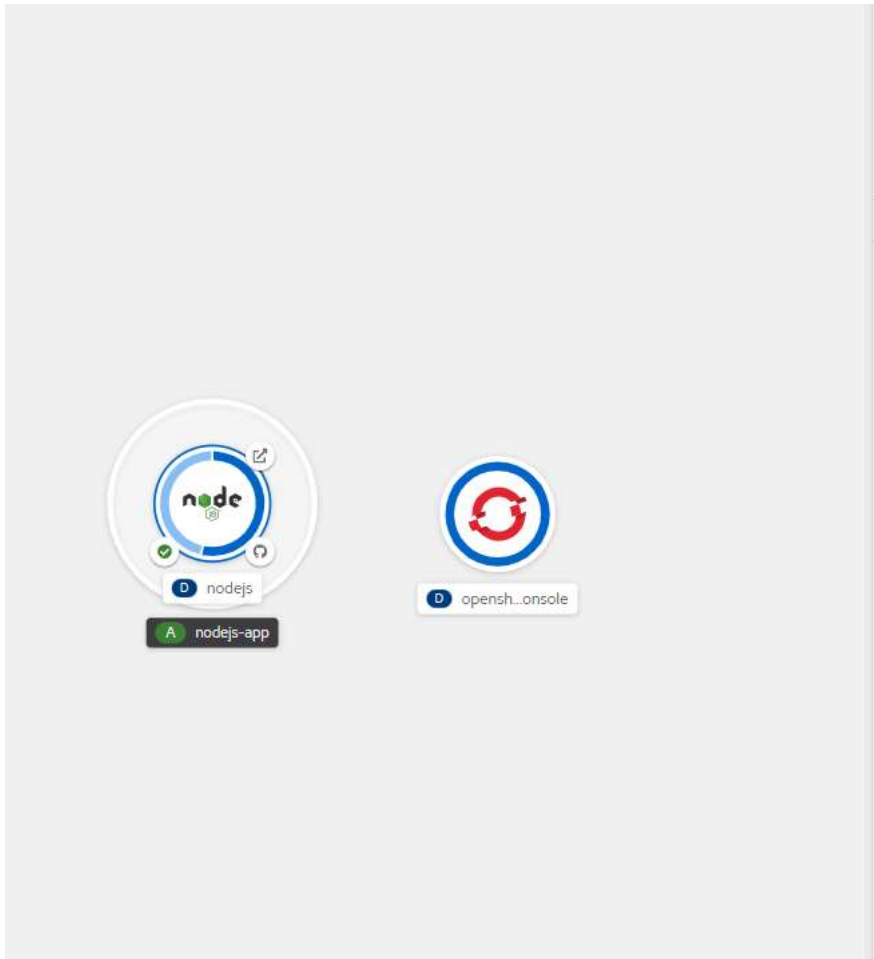
Let's try some specific steps:

1. Click the inner circle with the Node.js logo to bring up information on the Deployment and observe the four resources associated with this Deployment: a Pod that runs the containerized application; a Build that uses the s2i strategy to build the application into a container image; a Service that exposes the application as a network service; and a Route that provides an externally reachable hostname.

The screenshot shows the Skills Network OpenShift Lab interface. On the left is a dark sidebar with a menu. The top of the sidebar has a hamburger menu icon and the text "Skills Network OpenShift Lab". Below this, the "Developer" menu is expanded, showing options: "+Add", "Topology" (highlighted), "Monitoring", "Search", "Builds", "Pipelines", "Helm", "Project", "Config Maps", and "Secrets". The main area of the interface has a header with "Project:" and "Application: all applications". Below this are three filters: "Display Options", "Filter by Resource", and "Find by name...". The main content area is mostly empty, but on the right side, there is a red rectangular box highlighting a Node.js application icon. The icon is a blue circle with "node" in the center, surrounded by a white ring. Below the icon are two labels: "D nodejs-ex-git" and "A nodejs...it-app". At the bottom of the interface, there are several icons for search, zoom, and other functions.

Note: Please wait for status of the pod to change to 'Running' and for the Build to complete.

2. Click **View logs** on the line that says **Build #1**.



The dashboard shows a cluster with two nodes. The 'nodejs' node is running a 'nodejs-app' service. The 'openshift-onsole' node is also running. The 'nodejs' service is exposed on port 8080. The 'nodejs' route is available at the specified URL.

nodejs

Actions

Health Checks ×

Container nodejs does not have health checks to ensure your application is running correctly. [Add Health Checks](#)

Details

Resources

Monitoring

Pods

nodejs-776fbd8bd-wplg5

Running

[View logs](#)

Builds

nodejs

[Start Build](#)

Build #1 is complete (a few seconds ago)

[View logs](#)

Services

nodejs

Service port: 8080-tcp → Pod Port: 8080

Routes

nodejs

Location:
<http://nodejs-sn-labs-...prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud>

3. Read the logs to see a few key completed steps. The repository is cloned, a Dockerfile is generated, an image is built, and the image is pushed to the internal registry.

Skills Network OpenShift Lab

Developer

+Add

Topology

Monitoring

Search

Builds

Pipelines

Helm

Project

Config Maps

Secrets

Project

Builds > Build Details

nodejs-ex-git-1 Complete

Details YAML Environment **Logs** Events

Log stream ended.

89 lines

```
Cloning "https://github.com/sclorg/nodejs-ex.git" ...
Commit: b27dd9e9ffe53f76a1407d4fe357cf64c8a0ac6f (chore: remove the reference to node 10 and
Author: Lucas Holmquist <lholmqui@redhat.com>
Date: Thu Oct 21 13:29:44 2021 -0400
Caching blobs under "/var/cache/blobs".
Getting image source signatures
Copying blob sha256:8a4cee2d3973a8b9ccb73fc982adbefef274e95cb2548098e755b1df847aca0de
Copying blob sha256:71f6d04e5352b855df99a734fa3df8b4ce5c1e73583756a38dae7f0365d48f43
Copying blob sha256:ad62d8acaeb8e10bb459e0fb98054b6cd0769fe4d0485daf504967c8ffccd2df
Copying blob sha256:a7f628200d73a511e8ca006262c54054fd4fb3d4c6260bc0b7a770e402891ef8
Copying config sha256:ac1cc3129e2aab2b495b178f05076d357869f0dab48753cf8593bd24e1640dc8
Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-registry.svc:5000/openshift/r
Adding transient rw bind mount for /run/secrets/rhsm
Adding transient rw bind mount for /run/secrets/etc-pki-entitlement
Adding transient rw bind mount for /run/secrets/redhat.repo
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/openshift/nodejs@sha256:0442577b0599cbd
STEP 2: LABEL "io.openshift.build.commit.author"="Lucas Holmquist <lholmqui@redhat.com>" "io.op
STEP 3: ENV OPENSHIFT_BUILD_NAME="nodejs-ex-git-1" OPENSHIFT_BUILD_NAMESPACE="sn-labs-samaahs"
STEP 4: USER root
STEP 5: COPY upload/src /tmp/src
```

4. Click the **Details** tab for this Build.

5. And then click the link under **Owner** (at the very bottom) that says BC (Build Config).

The screenshot shows the OpenShift Skills Network interface. On the left is a dark sidebar with navigation options: Developer, +Add, Topology, Monitoring, Search, Builds, Pipelines, Helm, Project, Config Maps, and Secrets. The main area displays the 'Details' tab for a build configuration named 'nodejs-ex-git-1' in the 'sn-labs' namespace. The 'Details' tab is highlighted with a red box. Below the tab are sections for Labels, Annotations, Triggered By, Started, Created At, and Owner. The 'Owner' section is also highlighted with a red box. The 'Labels' section shows several labels including 'app=nodejs-ex-git', 'app.kubernetes.io/part-of=nodejs-ex-git-app', 'app.kubernetes.io/instance=nodejs-ex-git', 'openshift.io/build-config.name=nodejs-ex-git', 'app.kubernetes.io/component=nodejs-ex-git', 'openshift.io/build.start-policy=Serial', 'buildconfig=nodejs-ex-git', 'app.openshift.io/runtime=nodejs', 'app.kubernetes.io/name=nodejs', and 'app.openshift.io/runtime-version=14-ubi7'. The 'Annotations' section shows '3 Annotations'. The 'Triggered By' section shows 'Image change'. The 'Started' section shows 'Apr 11, 4:17 pm'. The 'Created At' section shows 'Apr 11, 4:17 pm'. The 'Owner' section shows 'BC nodejs-ex-git'.

Skills Network OpenShift Lab

Developer

+Add

Topology

Monitoring

Search

Builds

Pipelines

Helm

Project

Config Maps

Secrets

Project: [redacted]

Details YAML Environment Logs Events

Build Details

Name
nodejs-ex-git-1

Namespace
NS sn-labs- [redacted]

Labels Edit

- app=nodejs-ex-git
- app.kubernetes.io/part-of=nodejs-ex-git-app
- app.kubernetes.io/instance=nodejs-ex-git
- openshift.io/build-config.name=nodejs-ex-git
- app.kubernetes.io/component=nodejs-ex-git
- openshift.io/build.start-policy=Serial
- buildconfig=nodejs-ex-git
- app.openshift.io/runtime=nodejs
- app.kubernetes.io/name=nodejs
- app.openshift.io/runtime-version=14-ubi7

Annotations
3 Annotations

Triggered By
Image change

Started
Apr 11, 4:17 pm

Created At
Apr 11, 4:17 pm

Owner
BC nodejs-ex-git

6. If you look at the **Details** and **YAML** tabs, you'll see many concepts that we talked about in this module: triggers, build strategy, webhooks, and more.

Details

YAML

Environment

Logs

Events

```
1 kind: Build
2 apiVersion: build.openshift.io/v1
3 metadata:
4   annotations:
5     openshift.io/build-config.name: nodejs
6     openshift.io/build.number: '1'
7     openshift.io/build.pod-name: nodejs-1-build
8   resourceVersion: '334028934'
9   name: nodejs-1
10  uid: efb13c7a-6803-488c-b58a-a1e2cd554401
11  creationTimestamp: '2022-03-31T08:57:35Z'
12  generation: 2
13  namespace: sn-labs
14  ownerReferences:
15    - apiVersion: build.openshift.io/v1
16      kind: BuildConfig
17      name: nodejs
18      uid: 6dc83d68-d0da-46fe-a6c6-331dfa841fc7
19      controller: true
20  labels:
21    app: nodejs
22    app.kubernetes.io/part-of: nodejs-app
23    app.kubernetes.io/instance: nodejs
24    openshift.io/build-config.name: nodejs
25    app.kubernetes.io/component: nodejs
26    openshift.io/build.start-policy: Serial
27    buildconfig: nodejs
28    app.openshift.io/runtime: nodejs
29    app.kubernetes.io/name: nodejs
30    app.openshift.io/runtime-version: 14-ubi7
31  spec:
32    nodeSelector: null
33    output:
34      to:
35        kind: ImageStreamTag
36        name: 'nodejs:latest'
37    pushSecret:
38      name: builder-dockercfg-j9s2b
```

7. On the **Details** tab, click the link under **Output To** that says IST (ImageStreamTag).


The screenshot shows the OpenShift Skills Network Lab interface. On the left is a dark sidebar with navigation options: Developer, +Add, Topology, Monitoring, Search, Builds (highlighted), Pipelines, Helm, Project, Config Maps, and Secrets. The main content area is titled 'Project: [redacted]' and shows 'Build Configs > Build Config Details'. The build config is named 'nodejs-ex-git' (BC). A red box highlights the 'Details' tab in the top navigation bar, which also includes 'YAML', 'Builds', 'Environment', and 'Events'. Below the tabs, the 'Build Config Details' section shows the following information:

- Name:** nodejs-ex-git
- Namespace:** NS sn-labs-samaahs
- Labels:** app=nodejs-ex-git, app.kubernetes.io/component=nodejs-ex-git, app.kubernetes.io/instance=nodejs-ex-git, app.kubernetes.io/name=nodejs, app.kubernetes.io/part-of=nodejs-ex-git-app, app.openshift.io/runtime=nodejs, app.openshift.io/runtime-version=14-ubi7. An 'Edit' link is visible.
- Annotations:** 3 Annotations (with an edit icon).
- Created At:** Apr 11, 4:17 pm
- Owner:** No owner

At the bottom of the main content area, the 'Webhooks' section is visible. A large red bracket on the right side of the page indicates the area from the Labels section down to the Webhooks section.

8. You can now see the ImageStreamTag that was created as an output of the build. Click the **History** tab to see the image in the internal registry to which this ImageStreamTag points.

Image Streams > nodejs-ex-git > Image Stream Tag Details

 nodejs-ex-git:latest

Details

YAML

History

Apr 11, 4:19 pm

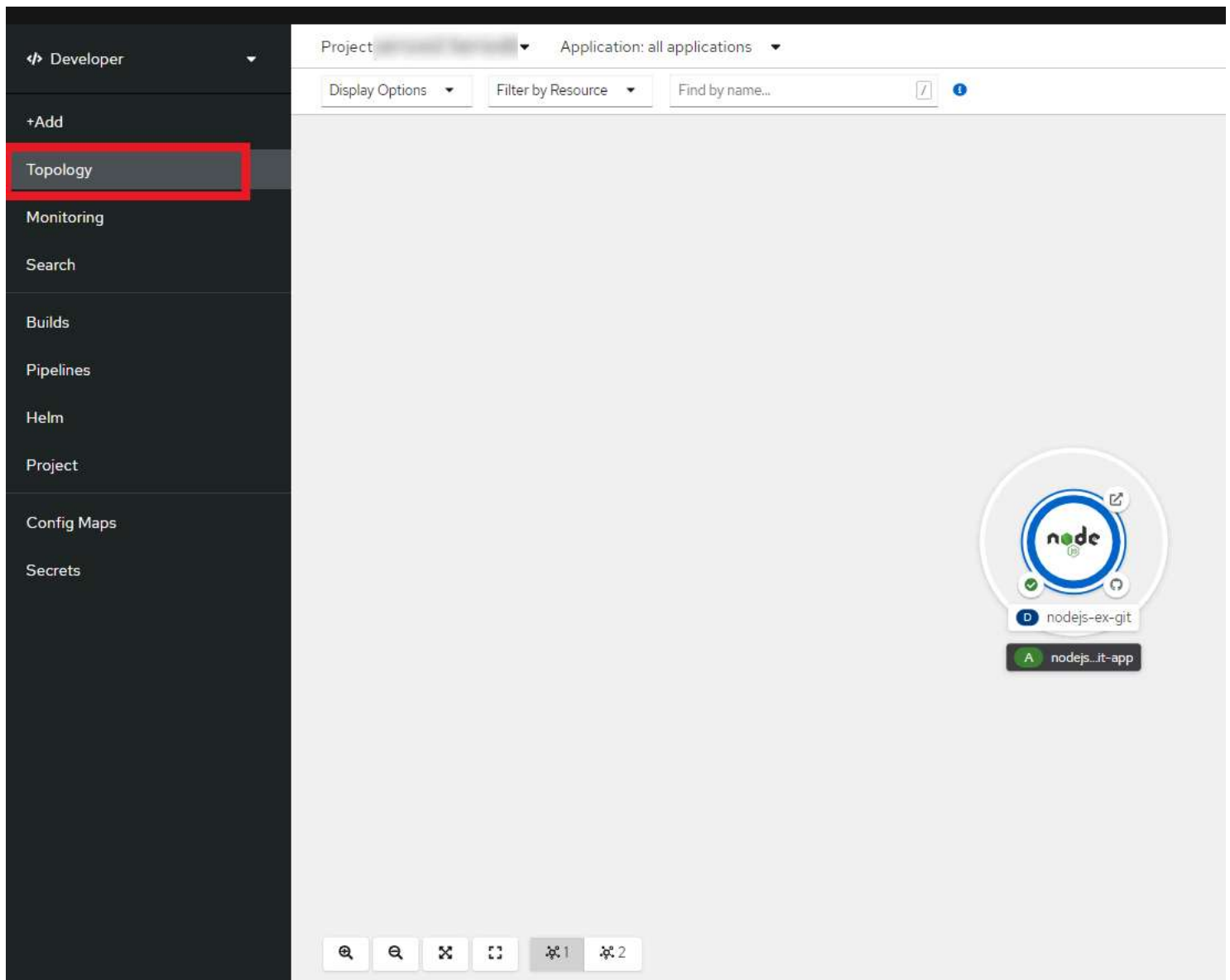
 nodejs-ex-git:latest

from image-registry.openshift-image-registry.svc:5000/sn-labs-: /nodejs-ex-git

sha256:39bf8ad2306e9a755a75abace734e466c238b6f985f68732e9b859a215f0ac01

9. Return to the Topology view and click on your Deployment info. Click the Route that OpenShift automatically created for you. This will open the application in the browser.

Note: Please note down this URL as it will be used in the next section



Autoscaling the nodejs-ex-git application

Now that the `nodejs-ex-git` app is successfully up and running, let's set up a horizontal pod autoscaler (HPA) so that it can handle any load that comes its way. Make sure to keep the `nodejs-ex-git` app open in a browser tab so that it continues to make requests and consume resources so that it can be successfully autoscaled.

First, we need to set resource requests and limits for the containers that will run. If a container requests a resource like CPU or memory, Kubernetes will only schedule it on a node that can give it that resource. On the other hand, limits prevent a container from consuming more than a certain amount of a resource.

In this case, we're going to request 3 millicores of CPU and 40 MB of RAM. We'll limit the containers to 30 millicores and 100 MB. These numbers are contrived in order to ensure that the app scales.

1. From the Topology view, click the `nodejs-ex-git` Deployment. Then click Actions > Edit Deployment.

The screenshot shows the OpenShift console interface. On the left, a large application icon for 'nodejs-ex-git' is highlighted with a red box. Below it, a smaller icon for 'opensh...onsole' is visible. The right sidebar displays the 'nodejs-ex-git' application details. The 'Actions' menu is open, showing options like 'Edit Application Grouping', 'Edit Pod Count', 'Pause Rollouts', 'Add Health Checks', 'Add Horizontal Pod Autoscaler', 'Add Storage', 'Edit Update Strategy', 'Edit nodejs-ex-git', 'Edit Labels', 'Edit Annotations', 'Edit Deployment' (highlighted with a red box), and 'Delete Deployment'. The 'Resources' tab is selected in the sidebar, showing 'Pods', 'Builds', and 'Services'.

2. In the `template.spec.containers` section, find `resources: {}`. Replace that with the following text. Make sure the spacing is correct as YAML uses strict indentation.

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


Deployments > Deployment Details

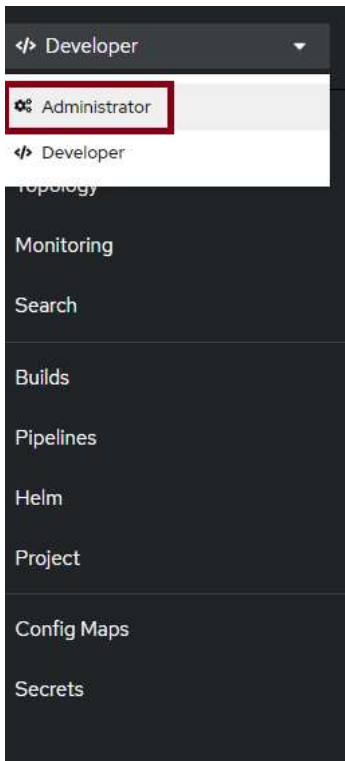
D nodejs-ex-gitDetails YAML Replica Sets Pods Environment Events

```
140   creationTimestamp: null
141   labels:
142     app: nodejs-ex-git
143     deploymentconfig: nodejs-ex-git
144   spec:
145     containers:
146     - name: nodejs-ex-git
147       image: >-
148         image-registry.openshift-image-registry.svc:5000/sn-labs-  
/nodejs-ex-git@sha256
149       ports:
150       - containerPort: 8080
151         protocol: TCP
152       resources:
153         limits:
154           cpu: 30m
155           memory: 100Mi
156         requests:
157           cpu: 3m
158           memory: 40Mi
159       terminationMessagePath: /dev/termination-log
160       terminationMessagePolicy: File
161       imagePullPolicy: Always
162     restartPolicy: Always
163     terminationGracePeriodSeconds: 30
164     dnsPolicy: ClusterFirst
165     securityContext: {}
166     schedulerName: default-scheduler
167   strategy:
168     type: RollingUpdate
```

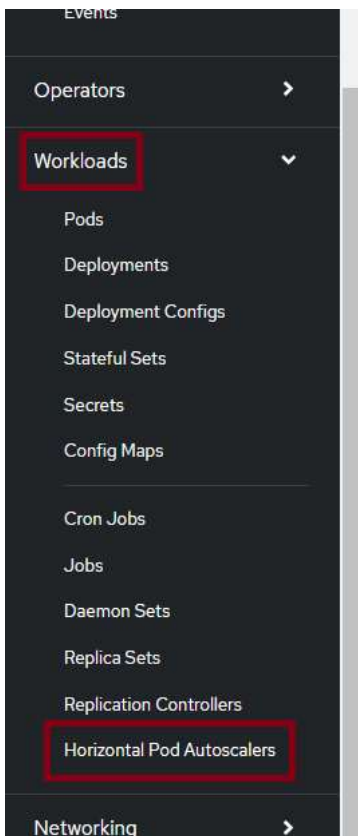

3. Click Save.

```
140   creationTimestamp: null
141   labels:
142     app: nodejs-ex-git
143     deploymentconfig: nodejs-ex-git
144   spec:
145     containers:
146     - name: nodejs-ex-git
147       image: >-
148         image-registry.openshift-image-registry.svc:5000/sn-labs-  
/nodejs-ex-git@sha256
149       ports:
150       - containerPort: 8080
151         protocol: TCP
152       resources:
153         limits:
154           cpu: 30m
155           memory: 100Mi
156         requests:
157           cpu: 3m
158           memory: 40Mi
159       terminationMessagePath: /dev/termination-log
160       terminationMessagePolicy: File
161       imagePullPolicy: Always
162     restartPolicy: Always
163     terminationGracePeriodSeconds: 30
164     dnsPolicy: ClusterFirst
165     securityContext: {}
166     schedulerName: default-scheduler
167   strategy:
168     type: RollingUpdate
```


4. Switch to the Administrator perspective.



5. Select Workloads > Horizontal Pod Autoscalers



6. Click Create Horizontal Pod Autoscaler

Horizontal Pod Autoscalers

No Horizontal Pod Autoscalers Found

7. Paste the following YAML into the editor

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: nodejs-ex-git-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nodejs-ex-git
  minReplicas: 1
  maxReplicas: 3
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 10
```

```
1  apiVersion: autoscaling/v2
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: nodejs-ex-git-hpa
5  spec:
6    scaleTargetRef:
7      apiVersion: apps/v1
8      kind: Deployment
9      name: nodejs-ex-git
10   minReplicas: 1
11   maxReplicas: 3
12   metrics:
13     - type: Resource
14       resource:
15         name: cpu
16         target:
17           type: Utilization
18           averageUtilization: 10
```

This HPA indicates that we're going to scale based on CPU usage. Generally you want to scale when your CPU utilization is in the 50-90% range. For this example, we're going to use 10% so that the app is more likely to need scaling. The minReplicas and maxReplicas fields indicate that the Deployment should have between one and three replicas at any given time depending on load.

8. Click Create

```

1  apiVersion: autoscaling/v2
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: nodejs-ex-git-hpa
5  spec:
6    scaleTargetRef:
7      apiVersion: apps/v1
8      kind: Deployment
9      name: nodejs-ex-git
10   minReplicas: 1
11   maxReplicas: 3
12   metrics:
13     - type: Resource
14       resource:
15         name: cpu
16         target:
17           type: Utilization
18           averageUtilization: 10

```

Create

Cancel

Download

9. Run the below command on the terminal in Theia to increase the load on the nodejs-ex-git and view the Autoscaling:

```
for i in `seq 1000`; do curl -L <your app URL>; done
```

Note: Replace <your app URL> with the URL that you obtained in Step 9 of the previous section.

```

^C
theia@theiaopenshift-7d206ba9ae5c-0000.us-east.containers.appdomain.cloud: /home/project$ for i in `seq 1000`; do curl -L http://nodejs-ex-git-sn-labs-7d206ba9ae5c-0000.us-east.containers.appdomain.cloud; done

```

The command will keep giving an output as below indicating successful load generation:

```

<h3>Command Line</h3>
<p>With the <a href="http://docs.okd.io/latest/cli_reference/overview.html">OpenShift command line interface</a> (CLI), you can manage projects from a terminal.</p>

<h2>Development Resources</h2>
<ul>
<li><a href="http://docs.okd.io/latest/welcome/index.html">OpenShift Documentation</a></li>
<li><a href="https://github.com/openshift/origin">OpenShift Origin GitHub</a></li>
<li><a href="https://github.com/openshift/source-to-image">Source To Image GitHub</a></li>
<li><a href="http://docs.okd.io/latest/using_images/s2i_images/nodejs.html">Getting Started with Node.js on OpenShift</a></li>
<li><a href="http://stackoverflow.com/questions/tagged/openshift">Stack Overflow questions for OpenShift</a></li>
<li><a href="http://git-scm.com/documentation">Git documentation</a></li>
</ul>

<h2>Request information</h2>
<p>Page view count:

<span class="code" id="count-value">No database configured</span>
</p>
</section>
</div>

<footer>
<div class="logo"><a href="https://www.openshift.com/"></a></div>
</footer>
</section>
</body>
</html>

```

Note: You can also verify autoscaling by directly executing your app URL in your browser. Add some fruit names along with their quantities on the application UI, and click 'Save'. Although the added fruits may not appear on the UI, this action will trigger a load change, causing the pods to autoscale to 3 after some time.

10. Click on nodejs-ex-git under Scale Target.

Horizontal Pod Autoscalers

Name ▾	Search by name...	
Name ↑	Labels ↓	Scale Target ↓
HPA nodejs-ex-git-hpa	No labels	D nodejs-ex-git
		Min Pods ↓
		1

11. If you wait, you'll see both Current Replicas and Desired Replicas become three. This is because the HPA detected sufficient load to trigger a scale up to the maximum number of Pods, which is three. You can also view the Last Scale Time as well as the current and target CPU utilization. The target is obviously 1% since that's what we set it to. Note that it can take a few minutes to trigger the scale up.

[Deployments](#) > [Deployment details](#)

D nodejs-ex-git

[Details](#) [YAML](#) [ReplicaSets](#) [Pods](#) [Environment](#) [Events](#)

Deployment details



Name

nodejs-ex-git

Namespace

NS

Labels

app=nodejs-ex-git app.kubernetes.io/component=nodejs-ex-git app.kubernetes.io/instance=nodejs-ex-git
app.kubernetes.io/name=nodejs-ex-git app.kubernetes.io/part-of=nodejs-ex-git-app app.openshift.io/runtime=nodejs
app.openshift.io/runtime-version=16-ubi8

[Edit](#)

Update strategy

RollingUpdate

Max unavailable

25% of 3 pods

Max surge

25% greater than 3 pods

Progress deadline seconds

600 seconds

Wow! OpenShift did some pretty incredible work on your behalf. All it needed was a code repository and it was able to build the code into a container image, push that image to a registry, create a Deployment that references that image, and also expose the application to the internet with a hostname.

Congratulations! You have completed the lab for the fourth module of this course.

© IBM Corporation. All rights reserved.