

Font: Helvetica Ne...
Step 4 of 6



Introduction to Red Hat OpenShift

Objectives

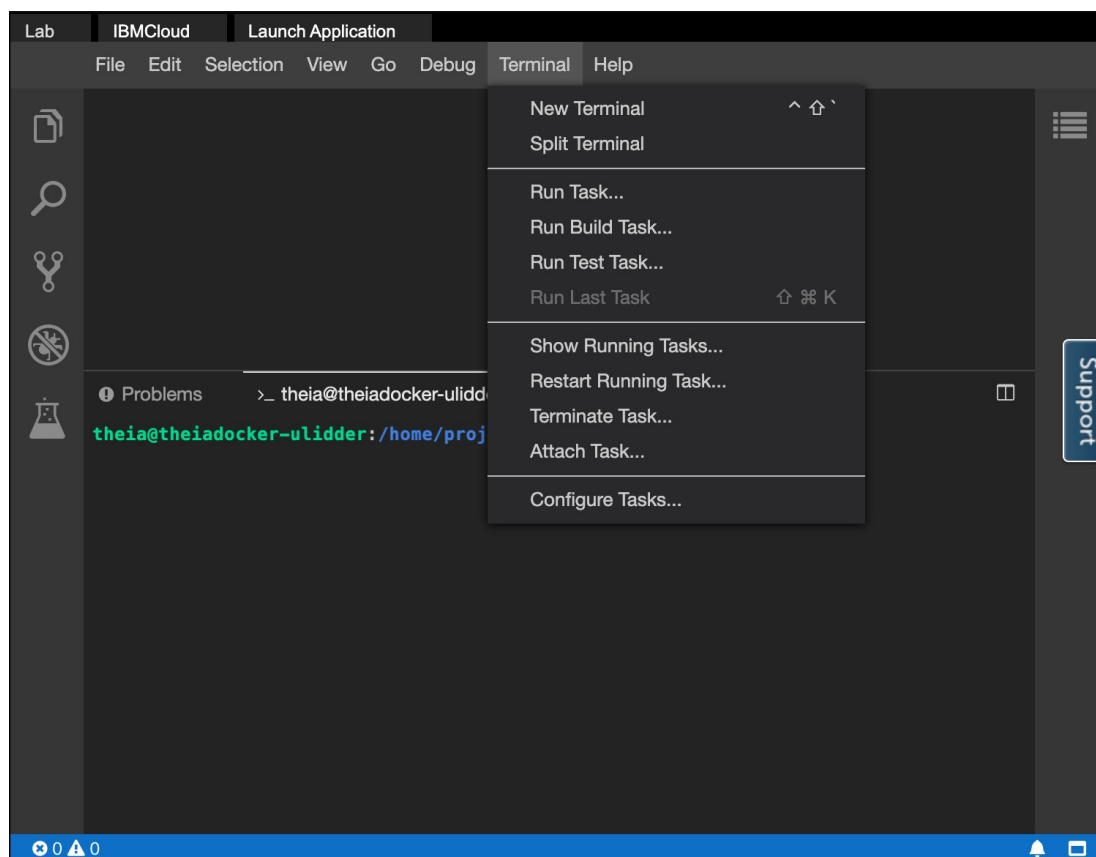
In this lab, you will:

- 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

Note: Kindly click on the OpenShift console tab and check if you have any existing project. If yes, then follow this [link](#) to delete the same first to avoid any issues with the lab. Otherwise, you are all set to perform the lab.

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

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

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

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
```

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

```
theia@theiaopenshift-: ~/:$ oc get pods
NAME                                READY   STATUS    RESTARTS   AGE
openshift-web-console-995896df-v22tp 2/2     Running   0           4h1m
```

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

```
oc get buildconfigs
```

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

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

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

```
oc project
```

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

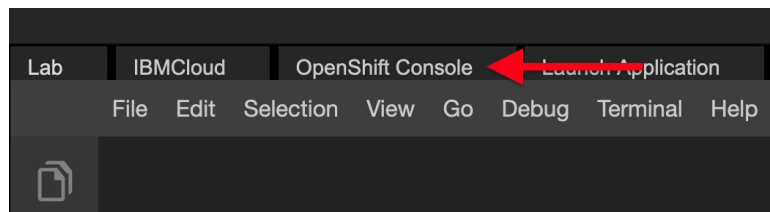
```
theia@theiaopenshift-: ~/:$ oc project
Using project "sn-labs-" from context named "sn-labs-oc" on server "https://c109-e.us-east.containers.cloud.ibm.com:30807"
theia@theiaopenshift-: ~/:$
```

Use the OpenShift web console

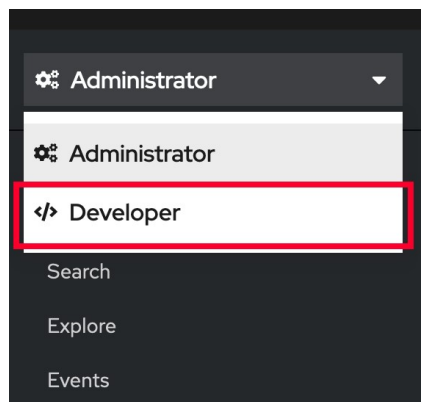
In addition to the CLI, OpenShift provides an intuitive web console. This is a useful and powerful feature because it enables you to deploy applications, view resources, monitor applications and view logs, and much more right in the console.

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

1. Open the OpenShift web console using the link at the top of the lab environment. 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 **From Git** among the options.

The screenshot shows the OpenShift Developer console interface. On the left is a dark sidebar with navigation links: Developer, +Add (highlighted with a red box), Topology, Monitoring, Search, Builds, Pipelines, Helm, Project, Config Maps, and Secrets. The main area is titled 'Add' and contains a grid of options to create an application. The 'From Git' option, which involves importing code from a Git repository, is highlighted with a red box. Other options include Quick Starts, Samples, Container Image, From Dockerfile, YAML, From Catalog, Database, Operator Backed, Helm Chart, and Pipeline.

Project: sn-
Add
Select a way to create an application, component or service from one of the options.

Quick Starts
Deploying an application with a pipeline
Getting started with a sample
Adding health checks to your sample application
See all Quick Starts →

From Git
Import code from your Git repository to be built and deployed

Container Image
Deploy an existing image from an image registry or image stream tag

From Dockerfile
Import your Dockerfile from your Git repository to be built and deployed

YAML
Create resources from their YAML or JSON definitions

From Catalog
Browse the catalog to discover, deploy and connect to services

Database
Browse the catalog to discover database services to add to your application

Operator Backed
Browse the catalog to discover and deploy operator managed services

Helm Chart
Browse the catalog to discover and install Helm Charts

Pipeline
Create a Tekton Pipeline to automate delivery of your application

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

Project: sn-labs-**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 Go Ruby Python Java **Node.js**

Builder Image Version *

14-ubi7

node 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 Limits and Labels.

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.

The screenshot shows the OpenShift web console interface. On the left, a dark sidebar contains navigation options: Developer, +Add, Topology (highlighted with a red box), Monitoring, Search, Builds, Pipelines, Helm, Project, Config Maps, and Secrets. The main content area displays a topology diagram for the 'nodejs-ex-git' application. It features two circular nodes: an outer circle labeled 'nodejs-ex-git' and an inner circle labeled 'openshift-onsole'. Below the outer circle, there are two smaller icons: a green checkmark labeled 'nodejs-ex-git' and a green 'A' labeled 'nodejs...it-app'. The right sidebar provides detailed information for the selected application, including a 'Health Checks' section with a warning message, a 'Pods' section showing a running pod, a 'Builds' section with a 'Start Build' button and a completed build, a 'Services' section showing a service port, and a 'Routes' section showing a route.

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.

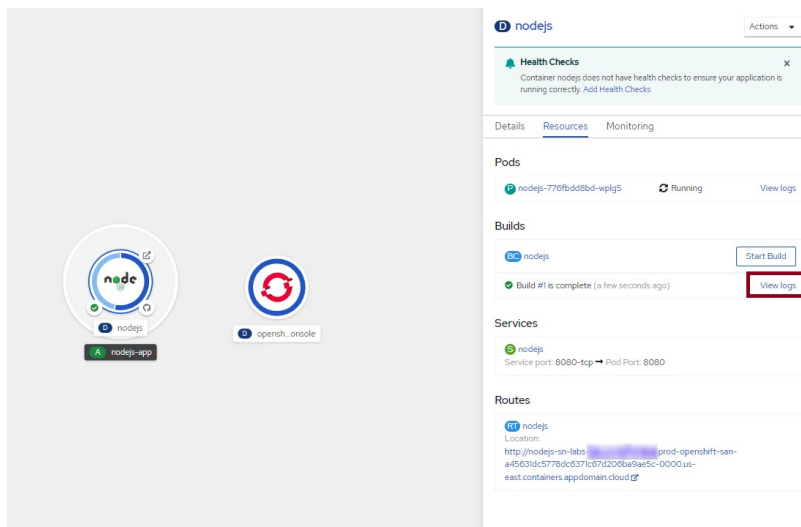
Let's try some specific steps:

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.

This screenshot is similar to the first one, but the 'nodejs-ex-git' node in the topology diagram is highlighted with a red box. The right sidebar remains the same, showing details for the 'nodejs-ex-git' application.

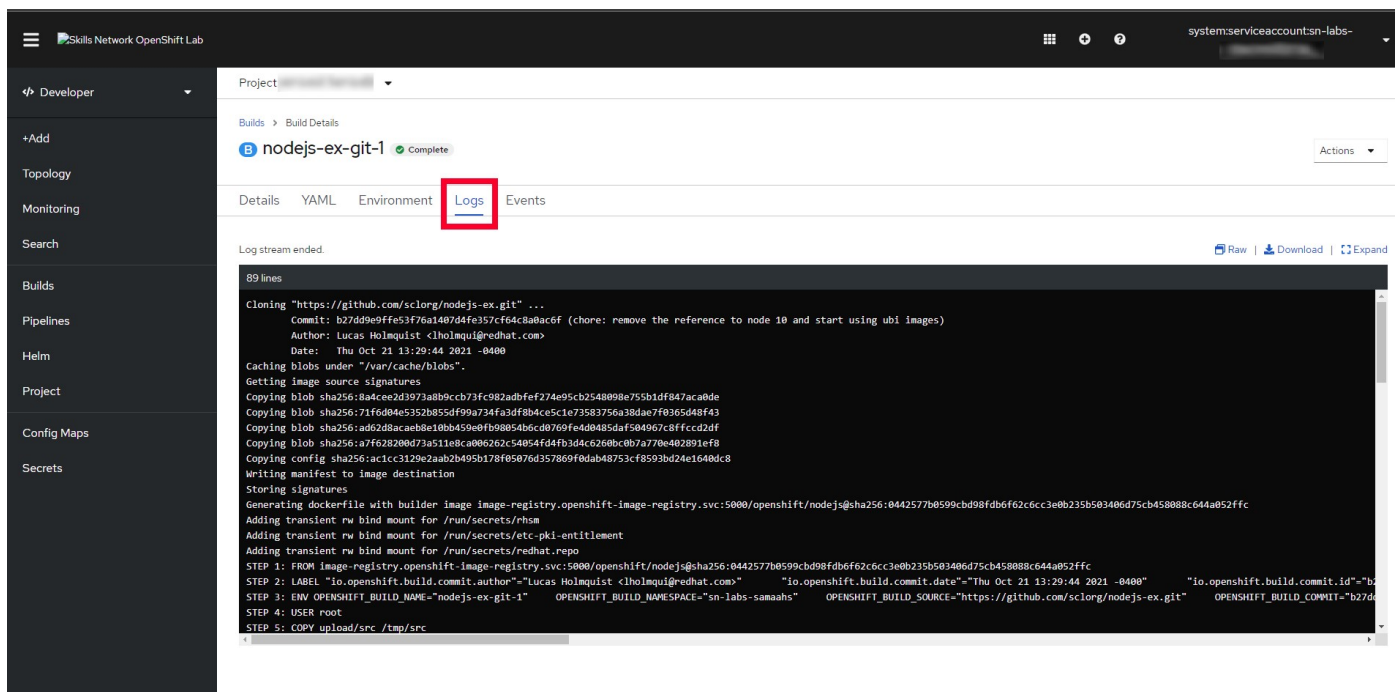
Note: Please wait for status of the pod to change to 'Running' and for the Build to complete.

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



The screenshot shows the OpenShift console interface for a 'nodejs' build configuration. On the left, there are icons for 'nodejs' and 'nodejs-app'. The main panel displays the 'nodejs' configuration with tabs for 'Details', 'Resources', and 'Monitoring'. The 'Builds' section shows a build is complete, and the 'Routes' section shows the application is accessible via a route.

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.



The screenshot shows the OpenShift console interface for a 'nodejs-ex-git-1' build configuration. The 'Logs' tab is selected, showing the build process logs. The logs include the following steps:

```
Cloning "https://github.com/sclorg/nodejs-ex.git" ...
Commit: b27dd9e9f53f76a1407d4fe357cf64c8a0ac6f (chore: remove the reference to node 10 and start using ubi images)
Author: Lucas Holmquist <lholmquist@redhat.com>
Date: Thu Oct 21 13:29:44 2021 -0400
Caching blobs under "/var/cache/blobs".
Getting image source signatures
Copying blob sha256:8a4cee2d3973a8b9ccb73fc982adbef274e95cb2548098e755b1df847aca0de
Copying blob sha256:71f6d84e532b855df99a734fa3df8b4ce5c1e73583756a38dae7f8365d48f43
Copying blob sha256:ade2d8acae8e10bb459e0fb98054b6cd0709fe4d0485daf504967c8ffcd2df
Copying blob sha256:a7f628200d73a511e8ca00e262c54054fd4fb3d4c6260bc0b7a776e482891ef8
Copying config sha256:ac1cc3129e2aab2b495b178f05076d357869f8dab48753cf8593bd24e1640dc8
Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-registry.svc:5000/openshift/nodejs@sha256:0442577b0599cbd98fdb6f62c6cc3e0b235b503406d75cb458088c644a052ffc
Adding transient rw bind mount for /run/secrets/ehsm
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@sha256:0442577b0599cbd98fdb6f62c6cc3e0b235b503406d75cb458088c644a052ffc
STEP 2: LABEL "io.openshift.build.commit.author"="Lucas Holmquist <lholmquist@redhat.com>" "io.openshift.build.commit.date"="Thu Oct 21 13:29:44 2021 -0400" "io.openshift.build.commit.id"="b27dd9e9f53f76a1407d4fe357cf64c8a0ac6f"
STEP 3: ENV OPENSHIFT_BUILD_NAME="nodejs-ex-git-1" OPENSHIFT_BUILD_NAMESPACE="sn-labs-samaahs" OPENSHIFT_BUILD_SOURCE="https://github.com/sclorg/nodejs-ex.git" OPENSHIFT_BUILD_COMMIT="b27dd9e9f53f76a1407d4fe357cf64c8a0ac6f"
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 web console interface. On the left is a 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 named 'nodejs-ex-git-1' in the 'sn-labs' namespace. The 'Build Details' section includes fields for Name, Namespace, Labels, Annotations, Triggered By, Started, and Created At. The 'Owner' field is highlighted with a red box and shows a user icon and the text 'nodejs-ex-git'. To the right, the 'Status' is 'Complete', the 'Type' is 'Source', and the 'Git Repository' is 'https://github.com/sclorg/nodejs-ex-git'. The 'Git Commit' is 'b27d9e' by Lucas Holmquist. The 'Context Dir' is '/'. The 'Build From' section shows the image registry and SHA. The 'Output To' section shows 'nodejs-ex-git:latest'. The 'Push Secret' is 'builder-dockercfg-c9ztr'.

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.

The screenshot shows the 'YAML' tab of the build configuration. The YAML content is as follows:

```
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: '534828934'
9   name: nodejs-1
10  uid: efb33c7a-6803-488c-b58a-ale2cd54401
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: 6dc83068-0b0a-46fe-ab6c-331d4a841fc7
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-19a2b
```

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

Project: [Project Name]

Build Configs > Build Config Details

nodesjs-ex-git

Actions

Details | YAML | Builds | Environment | Events

Build Config Details

Name nodesjs-ex-git	Type Source
Namespace sn-labs-samaahs	Git Repository https://github.com/sclog/nodesjs-ex-git
Labels app=nodesjs-ex-git, app.kubernetes.io/component=nodesjs-ex-git, app.kubernetes.io/instance=nodesjs-ex-git, app.kubernetes.io/name=nodesjs, app.kubernetes.io/part-of=nodesjs-ex-git-app, app.openshift.io/runtime=nodesjs, app.openshift.io/runtime-version=14-ubi7	Context Dir /
Annotations 3 Annotations	Build From nodesjs14-ubi7
Created At Apr 11, 4:17 pm	Output To nodesjs-ex-git:latest
Owner No owner	Run Policy Serial
	Triggers Generic, GitHub, ImageChange, ConfigChange

Webhooks

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 > nodesjs-ex-git > Image Stream Tag Details

nodesjs-ex-git:latest

Actions

Details | YAML | **History**

Apr 11, 4:19 pm

nodesjs-ex-git:latest

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

sha256:39bf8ad2306e9a755a75abace734e466c238b6f985f68732e9b859a2f5f0ac01

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.

Project: [Project Name] Application: all applications

Display Options | Filter by Resource | Find by name...

Topology | Monitoring | Search | Builds | Pipelines | Helm | Project | Config Maps | Secrets

nodesjs-ex-git

Health Checks

Container nodesjs-ex-git does not have health checks to ensure your application is running correctly. Add Health Checks

Pods

nodesjs-ex-git-9cd4c6b4d-nth58 Running View logs

Builds

nodesjs-ex-git Start Build

Build #1 is complete (24 minutes ago) View logs

Services

nodesjs-ex-git Service port: 8080-tcp → Pod Port: 8080

Routes

nodesjs-ex-git Location: http://nodesjs-ex-git-...-prod-openshift-sa...-a4563dc5778dc637c67d206ba9ae5c-0000us-east.containers.appdomain.cloud

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.

Welcome to your Node.js application on OpenShift

How to use this example application

For instructions on how to use this application with OpenShift, start by reading the [Developer Guide](#).

Deploying code changes

The source code for this application is available to be forked from the [OpenShift GitHub repository](#). You can configure a webhook in your repository to make OpenShift automatically start a build whenever you push your code:

1. From the Web Console homepage, navigate to your project
2. Click on Browse > Builds
3. Click the link with your BuildConfig name
4. Click the Configuration tab
5. Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field
6. Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook
7. Paste your webhook URL provided by OpenShift in the "Payload URL" field
8. Change the "Content type" to 'application/json'
9. Leave the defaults for the remaining fields — that's it!

After you save your webhook, if you refresh your settings page you can see the status of the ping that GitHub sent to OpenShift to verify it can reach the server.

Note: adding a webhook requires your OpenShift server to be reachable from GitHub.

Working in your local Git repository

If you forked the application from the OpenShift GitHub example, you'll need to manually clone the repository to your local system. Copy the application's source code Git URL and then run:

```
$ git clone <git_url> <directory_to_create>

# Within your project directory
# Commit your changes and push to OpenShift

$ git commit -a -m 'Some commit message'
$ git push
```

After pushing changes, you'll need to manually trigger a build if you did not setup a webhook as described above.

Managing your application

Documentation on how to manage your application from the Web Console or Command Line is available at the [Developer Guide](#).

Web Console

You can use the Web Console to view the state of your application components and launch new builds.

Command Line

With the [OpenShift command line interface](#) (CLI), you can create applications and manage projects from a terminal.

Development Resources

- [OpenShift Documentation](#)
- [OpenShift Origin GitHub](#)
- [Source To Image GitHub](#)
- [Getting Started with Node.js on OpenShift](#)
- [Stack Overflow questions for OpenShift](#)
- [Git documentation](#)

Request information

Page view count: No database configured

Built on



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

Note: Please delete your project from OpenShift Console & SN labs environment before signing out to ensure that further labs requiring the use of OpenShift console run correctly. To do the same, click on this [link](#)

Changelog

Date	Version	Changed by	Change Description
2022-04-08	1.1	Samaah Sarang	Updated Lab instructions & images
2022-04-13	1.2	Samaah Sarang	Updated Lab instructions & images
2022-04-14	1.3	K Sundararajan	Updated Lab instructions & images
2022-04-19	1.4	K Sundararajan	Updated Lab instructions

© IBM Corporation 2022. All rights reserved.

[Previous](#)

[Continue](#)