Introduction to Red Hat OpenShift
红帽 OpenShift 简介

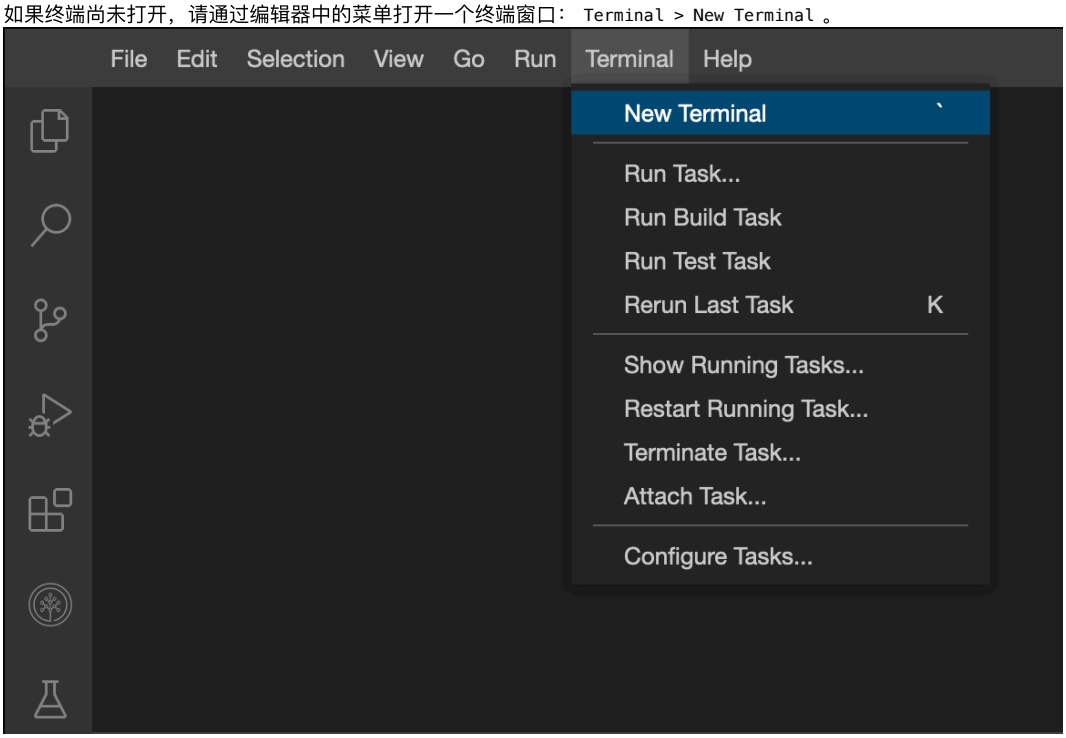
Objectives 目标

In this lab, you will:
在本实验中，您将：

- Use the `oc` CLI (OpenShift command line interface)
  使用 `oc` 命令行界面（OpenShift 命令行界面）
- Use the OpenShift web console
  使用 OpenShift 网络控制台
- Build and deploy an application using s2i ('Source-to-image' build strategy)
  使用 s2i（'源到镜像' 构建策略）构建和部署应用程序
- Inspect a BuildConfig and an ImageStream
  检查 BuildConfig 和 ImageStream
- Autoscale the application
  自动调整应用程序大小

Verify the environment and command line tools
验证环境和命令行工具

1. If a terminal is not already open, open a terminal window by using the menu in the editor: `Terminal > New Terminal`.
   如果终端尚未打开，请通过编辑器中的菜单打开一个终端窗口： `Terminal > New Terminal` 。



   Note: Please wait for some time for the terminal prompt to appear.
   注意：请等待一段时间，以便终端提示符出现。

2. Verify that `oc` CLI is installed.
   验证 `oc` CLI 是否已安装。

```
oc version
```

```
theia@theiaopenshift-[redacted]:/home/project$ oc version
Client Version: 4.13.7
Kustomize Version: v4.5.7
Kubernetes Version: v1.27.16+03a907c
```

You should see output similar to this, although the versions may be different.
您应该看到类似于以下内容的输出，尽管版本可能不同。

3. Change to your project folder.
   切换到你的项目文件夹。

   NOTE: If you are already on `home/project` please skip this step
   注意：如果你已经在 `home/project` ，请跳过此步骤

   ```
   cd /home/project
   ```

4. Clone the git repository that contains the artifacts needed for this lab, if it doesn't already exist.
   克隆包含本实验所需工件的 git 仓库，如果它还不存在的话。

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

```
theia@theiaopenshift-[redacted]:/home/project$ [ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.gi
t
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 使用 oc 命令行界面

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.
OpenShift 项目是具有附加管理功能的 Kubernetes 命名空间。因此，项目也在 OpenShift 集群内提供隔离。您已经可以访问 OpenShift 集群中的一个项目，并且 `oc` 已经设置为针对该集群和项目。

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`.
让我们来看一些基本的 `oc` 命令。回想一下 `oc` 带有 `kubectl` 的副本，因此所有 `kubectl` 命令都可以使用 `oc` 运行。

1. List the Pods in this namespace.
   列出此命名空间中的 Pod。

   ```
   oc get pods
   ```

```
theia@theiaopenshift-[redacted]:/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.
您可能会看到几个属于环境的一部分的 Pod。您不需要担心这些。

2. In addition to Kubernetes objects, you can get OpenShift specific objects.
   除了 Kubernetes 对象外，您还可以获取 OpenShift 特定对象。

   ```
   oc get buildconfigs
   ```

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

Because you haven't created a BuildConfig yet, this will not return any resources.
由于您尚未创建 BuildConfig，因此这将不会返回任何资源。

3. View the OpenShift project that is currently in use.
   查看当前正在使用的 OpenShift 项目。

   ```
   oc project
   ```

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

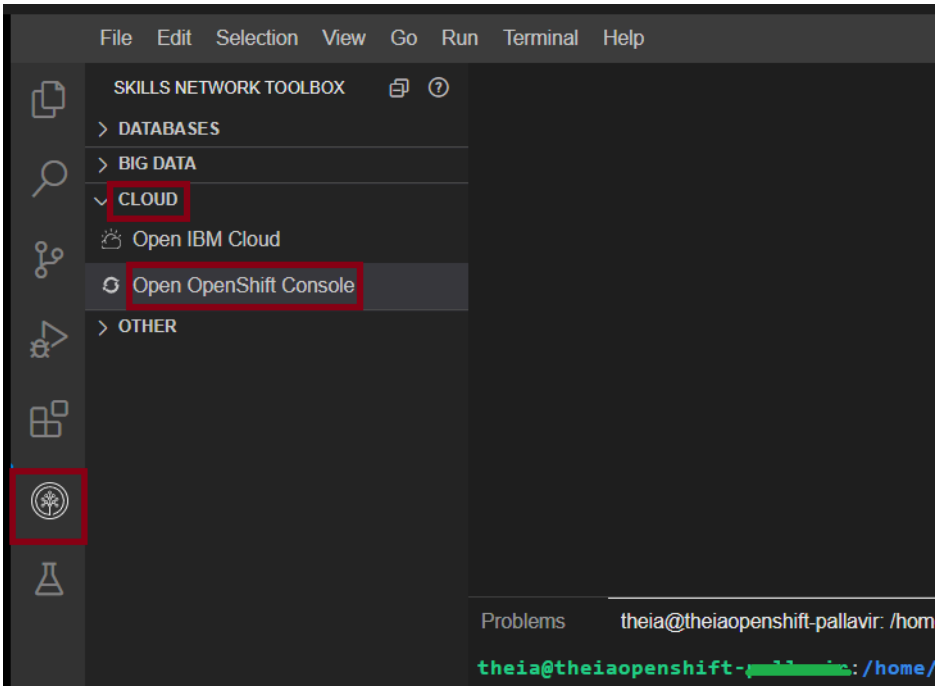
This project is specific to you and provides isolation within the cluster so that you can deploy your own applications.
这个项目是专门为您提供的，并在集群中提供隔离，以便您可以部署自己的应用程序。

Use the OpenShift web console
使用 OpenShift 网络控制台

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.
除了命令行界面，OpenShift 还提供了一个直观的网页控制台。这是一个有用且强大的功能，因为它使您能够在控制台中直接部署应用程序、查看资源、监控应用程序和查看日志等。

Let's open up the console and have a look around.
让我们打开控制台并四处看看。

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.
   要打开 OpenShift 网页控制台，请点击右侧的 Skills Network 按钮，它将打开 Skills Network 工具箱。然后点击云，再点击 Open OpenShift 控制台，如下所示图像所示。

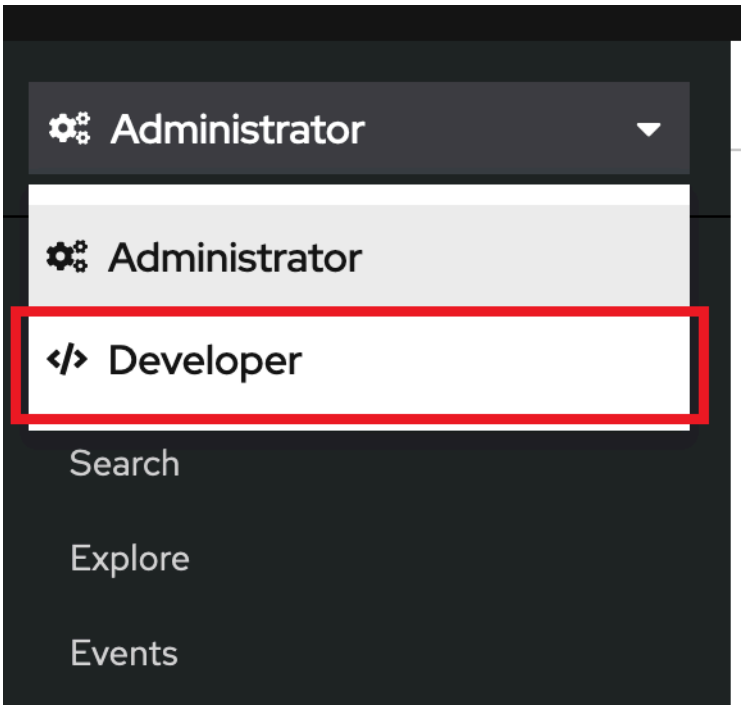


It can take a few minutes to become available after opening the lab environment, so if you get an error, wait a minute and try again.
在打开实验室环境后，可能需要几分钟才能可用，所以如果您遇到错误，请等待一分钟再尝试。

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.
控制台应打开到您已分配的项目详情页面。查看 OpenShift 以直观、可视化方式提供的所有信息。点击此项目的仪表板、概览和其他标签以查看更多信息。您应看到当前项目中存在的资源清单、定义此项目的 YAML 文件以及更多内容。

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.
熟悉左侧导航菜单中的项目。您可以看到运营商、多种不同的 Kubernetes 对象以及一些 OpenShift 特定的对象，我们在这门课程中已经讨论过这些内容。目前这些对象还没有很多实例，但一旦我们部署应用程序，它们就会充满内容。

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.)
注意左上角的"管理员"字样。这表示您处于管理员视角。还有一个开发者视角。每个视角都提供针对该角色的特定工作流程。切换到开发者视角开始部署应用程序。（如果已经显示"开发者"，则无需更改。）



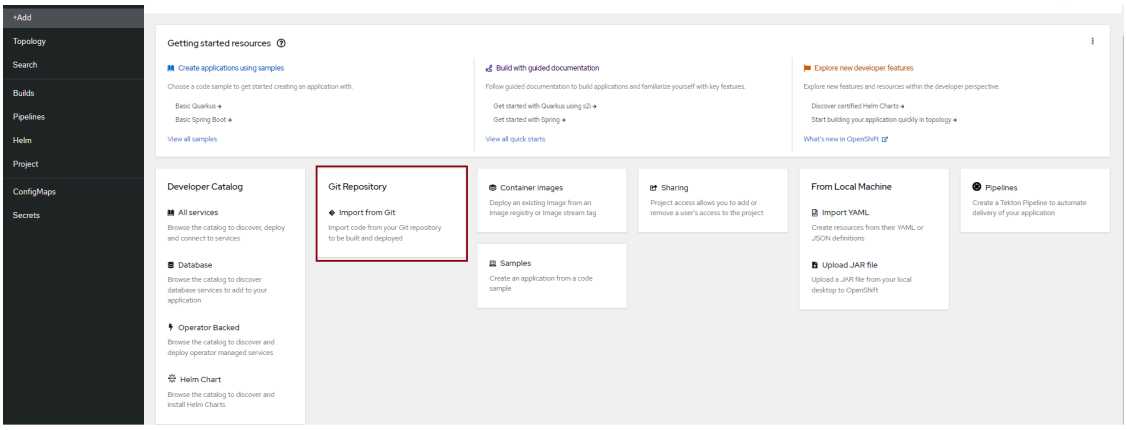
Deploy an application in the web console
在 Web 控制台部署应用程序

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.
开发者视角提供了针对开发者使用场景的工作流程，例如创建和部署应用程序的能力。让我们从这里开始！您可能处于"拓扑"视图，该视图提供了应用程序的视觉表示。如果不是，请切换到它查看。

1. Let us add a new application to this project. There are several ways to add a new application in Openshift.
让我们向此项目添加一个新的应用程序。在 OpenShift 中添加新应用程序有几种方法。

2. Click the **+Add** button to add a new application.
点击+添加按钮以添加新应用程序。

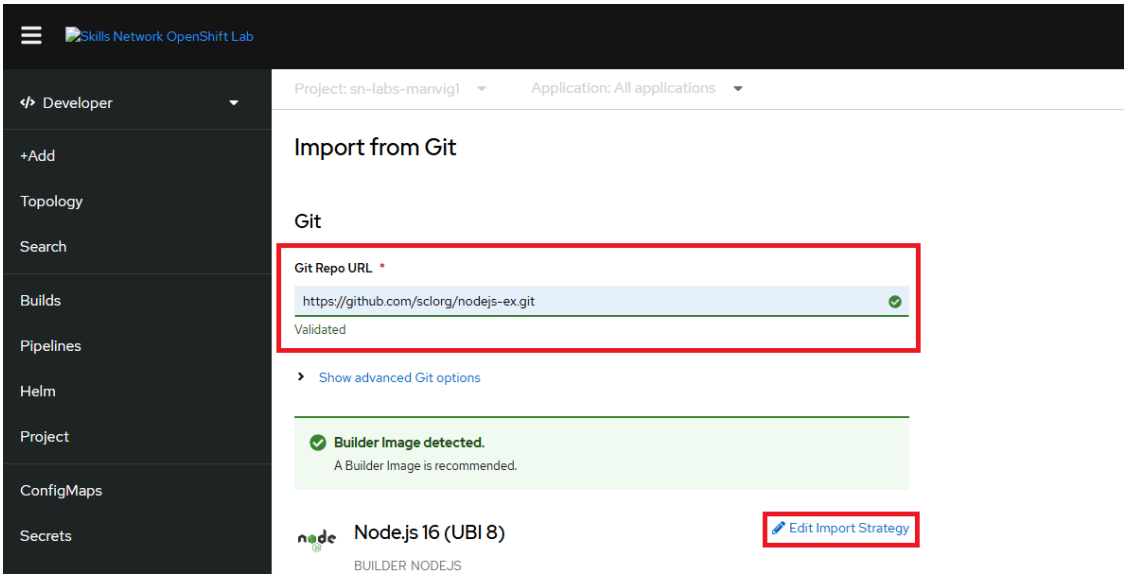
3. Select **Git Repository (Import from Git)** among the options.
在选项中选择"Git 仓库（从 Git 导入）"。



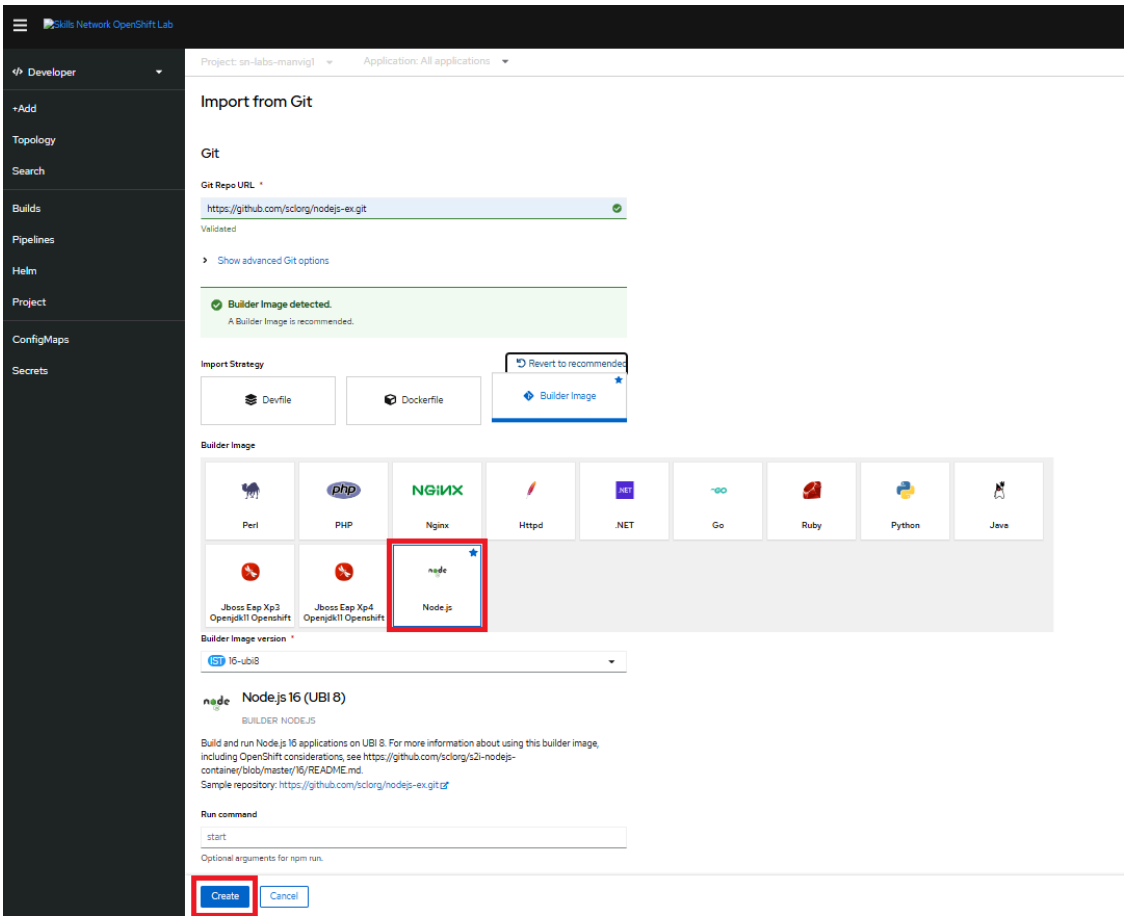
4. You will be redirected to **Import from Git** window. OpenShift will deploy an application using only one input from you: the application source.
   您将被重定向到从 Git 导入窗口。OpenShift 将使用您的一个输入来部署应用程序：应用程序源。

5. In the **Git Repo URL** box, paste the sample one mentioned below.
   在 Git 仓库 URL 框中，粘贴下方提到的示例。

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



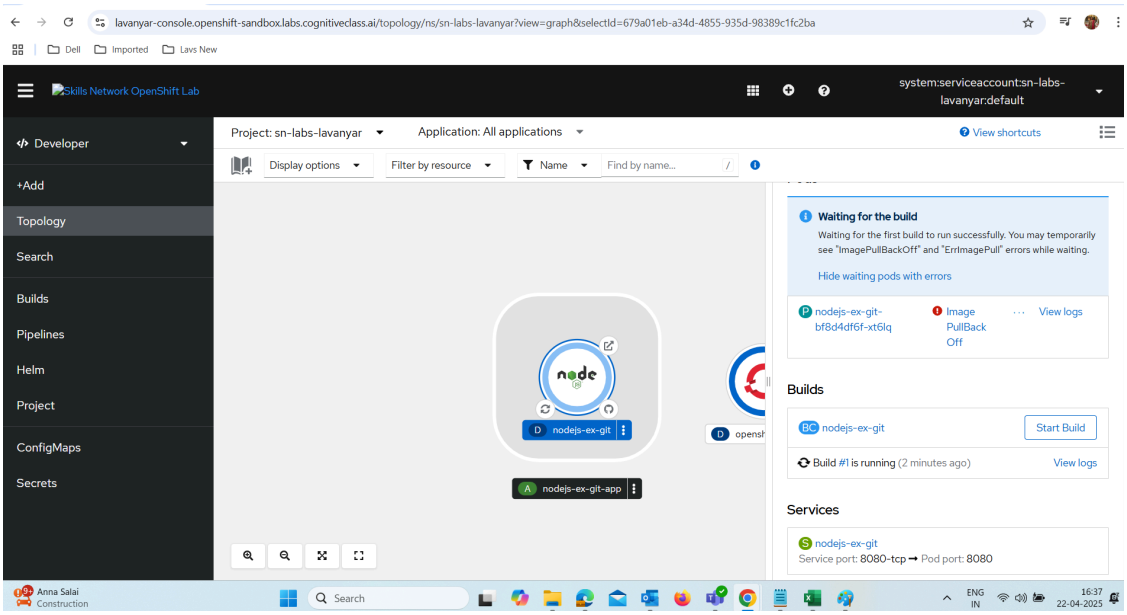
Note: Click on the Edit Import Strategy to see the various builder images. We shall be using the Node.js image for our application. Ensure that this image has been selected.
注意：点击编辑导入策略以查看各种构建器镜像。我们将为我们的应用程序使用 Node.js 镜像。确保已选中此镜像。



6. Keep the rest of the default options as they already are. Then scroll down and click **Create**.
   保持其余默认选项不变。然后向下滚动并点击创建。

In the Topology view, you should now see your newly created application.
在拓扑视图中，您现在应该能够看到您新创建的应用程序。

NOTE: It will take several minutes for the application to appear. Refresh the browser if within 3 minutes, you don't see any application.
注意：应用程序出现需要几分钟时间。如果在 3 分钟内仍未看到任何应用程序，请刷新浏览器。



Note: Please wait for the first build to run successfully. You may temporarily see **ImagePullBackOff** and **ErrImagePull** errors while waiting.
注意：请等待首次构建成功运行。在等待期间，您可能会暂时看到 ImagePullBackOff 和 ErrImagePull 错误。

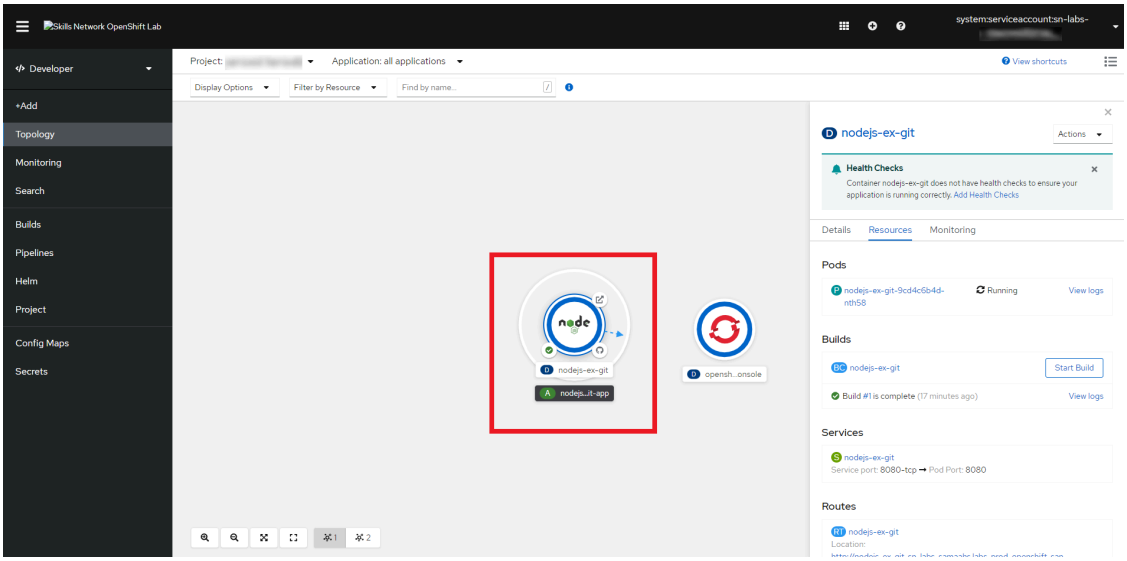
View application in the web console
在网页控制台查看应用程序

The Topology view provides quick links to a lot of important parts of an application:
拓扑视图提供了快速链接到应用程序的许多重要部分：

- The outer circle gets the information on the application.
  外圈获取应用程序的信息。
- The inner circle with the Node.js logo gives information about the Deployment.
  带有 Node.js 标志的内圈提供有关部署的信息。
- The GitHub icon is used to access the code repository.
  GitHub 图标用于访问代码仓库。
- The check mark shows the most recent build (you will see circular arrows if the build is in progress).
  对勾标记表示最新的构建（如果构建正在进行中，您将看到循环箭头）。
- The arrow coming out of a box can be used to view the application in the browser if the application is externally available.
  从方框中出来的箭头可用于在浏览器中查看应用程序，如果应用程序是外部可用的。

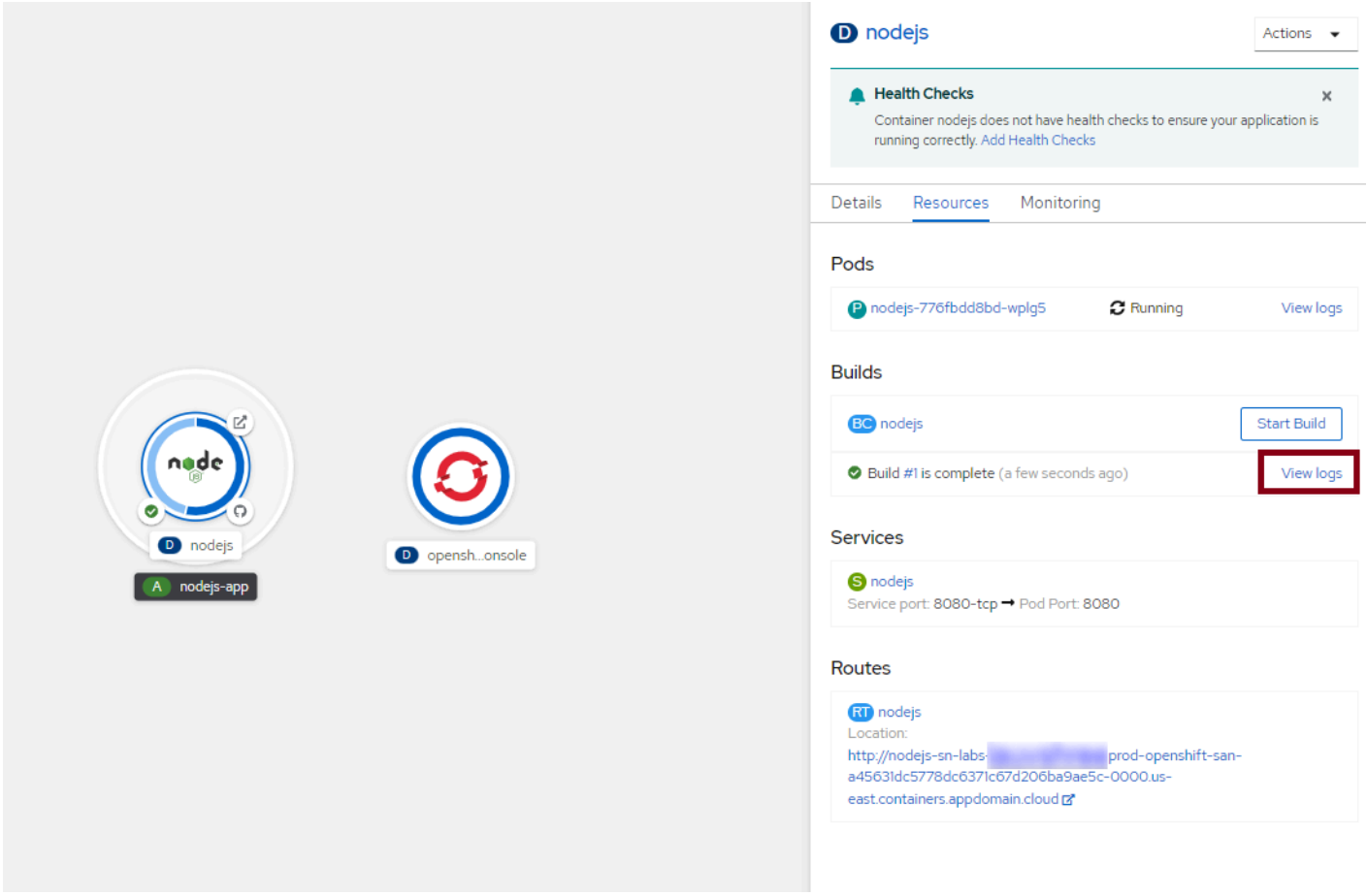
Let's try some specific steps:
让我们尝试一些具体步骤：

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.
   点击带有 Node.js 标志的内圆，以显示有关部署的信息并观察与此部署相关的四个资源：一个运行容器化应用程序的 Pod；一个使用 s2i 策略将应用程序构建为容器镜像的 Build；一个将应用程序作为网络服务暴露的服务；以及一个提供外部可访问主机名的路由。

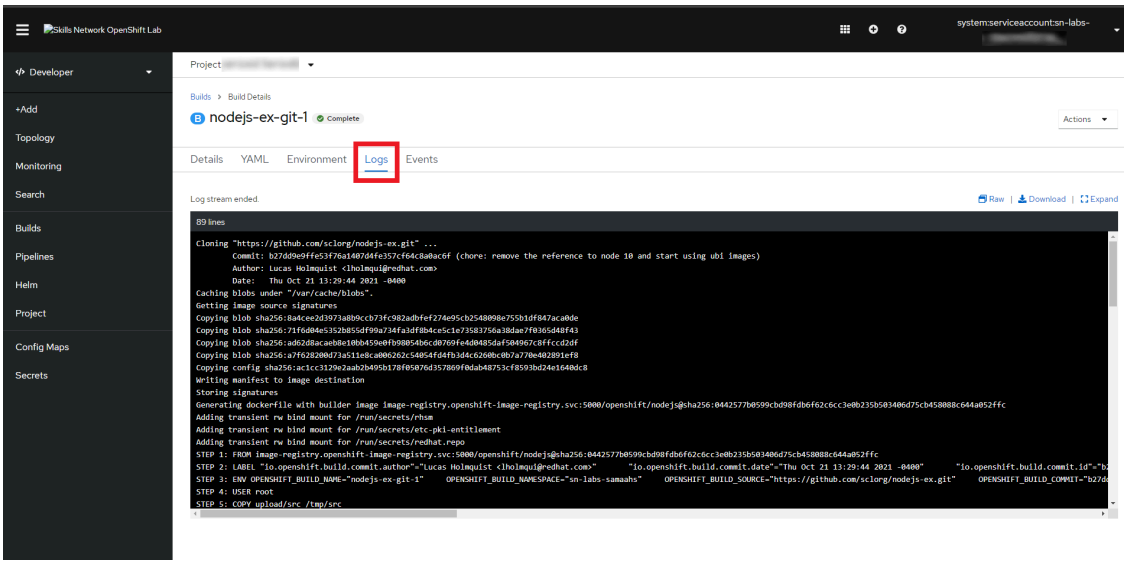


Note: Please wait for status of the pod to change to 'Running' and for the Build to complete.
注意：请等待 Pod 状态变为 'Running'，并等待构建完成。

2. Click **View logs** on the line that says **Build #1**.
点击显示 Build #1 的那一行上的 "查看日志"。

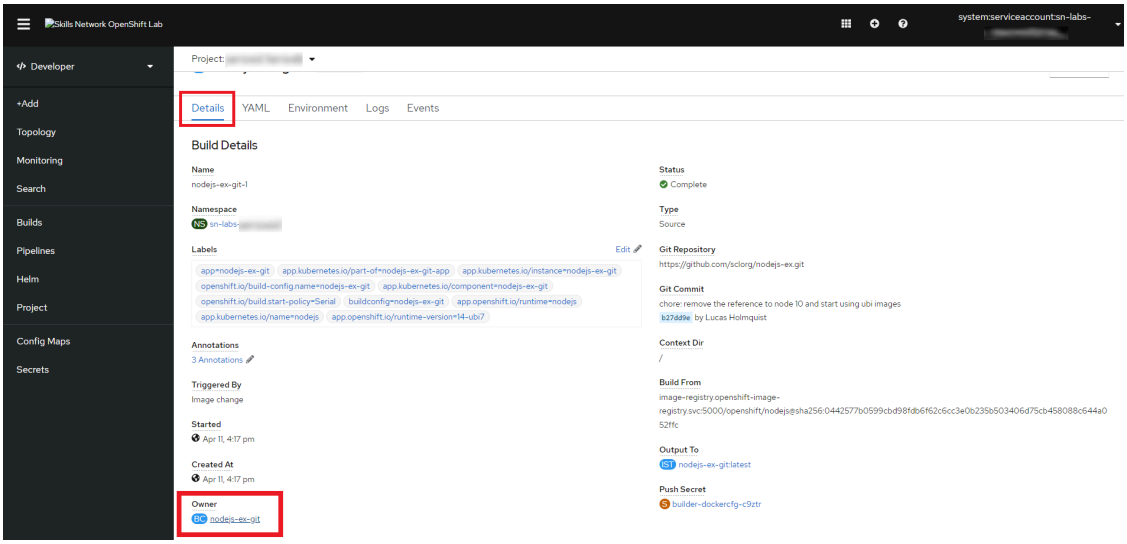


3. Read the logs to see a few key completed steps. The repository is cloned, a Dockerfile is generated, an image is built, and the image is pushed to the internal registry.
阅读日志以查看几个关键完成的步骤。代码库被克隆，生成了一个 Dockerfile，构建了一个镜像，并将镜像推送到内部注册中心。



4. Click the **Details** tab for this Build.
点击此构建的 "详情" 标签。

5. And then click the link under **Owner** (at the very bottom) that says BC (Build Config).
然后点击 Owner 下方的链接（在最底部），该链接标有 BC（Build Config）。

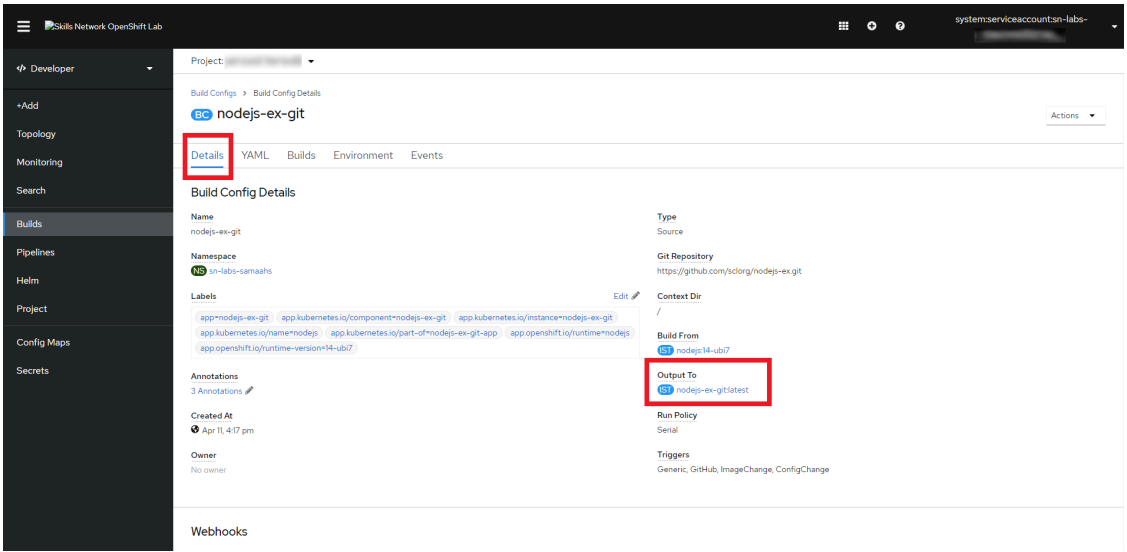


6. If you look at the **Details** and **YAML** tabs, you'll see many concepts that we talked about in this module: triggers, build strategy, webhooks, and more.
如果你查看 Details 和 YAML 标签页，你会看到我们在这个模块中讨论过的许多概念：触发器、构建策略、webhooks 等。

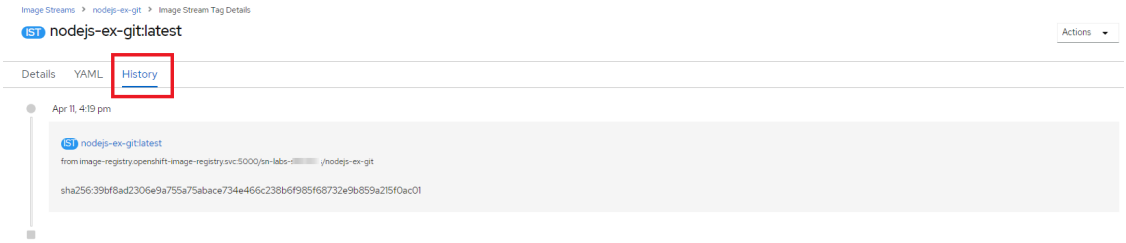



```yaml
kind: Build
apiVersion: build.openshift.io/v1
metadata:
  annotations:
    openshift.io/build-config.name: nodejs
    openshift.io/build.number: '1'
    openshift.io/build.pod-name: nodejs-1-build
  resourceVersion: '334028934'
  name: nodejs-1
  uid: efb13c7a-6803-488c-b58a-a1e2cd554401
  creationTimestamp: '2022-03-31T08:57:35Z'
  generation: 2
  namespace: sn-labs-[redacted]
  ownerReferences:
    - apiVersion: build.openshift.io/v1
      kind: BuildConfig
      name: nodejs
      uid: 6dc83d68-d0da-46fe-a6c6-331dfa841fc7
      controller: true
  labels:
    app: nodejs
    app.kubernetes.io/part-of: nodejs-app
    app.kubernetes.io/instance: nodejs
    openshift.io/build-config.name: nodejs
    app.kubernetes.io/component: nodejs
    openshift.io/build.start-policy: Serial
    buildconfig: nodejs
    app.openshift.io/runtime: nodejs
    app.kubernetes.io/name: nodejs
    app.openshift.io/runtime-version: 14-ubi7
spec:
  nodeSelector: null
  output:
    to:
      kind: ImageStreamTag
      name: 'nodejs:latest'
    pushSecret:
      name: builder-dockercfg-j9s2b
```

7. On the **Details** tab, click the link under **Output To** that says IST (ImageStreamTag).
在 Details 标签页中，点击 Output To 下方的链接，该链接标有 IST（ImageStreamTag）。

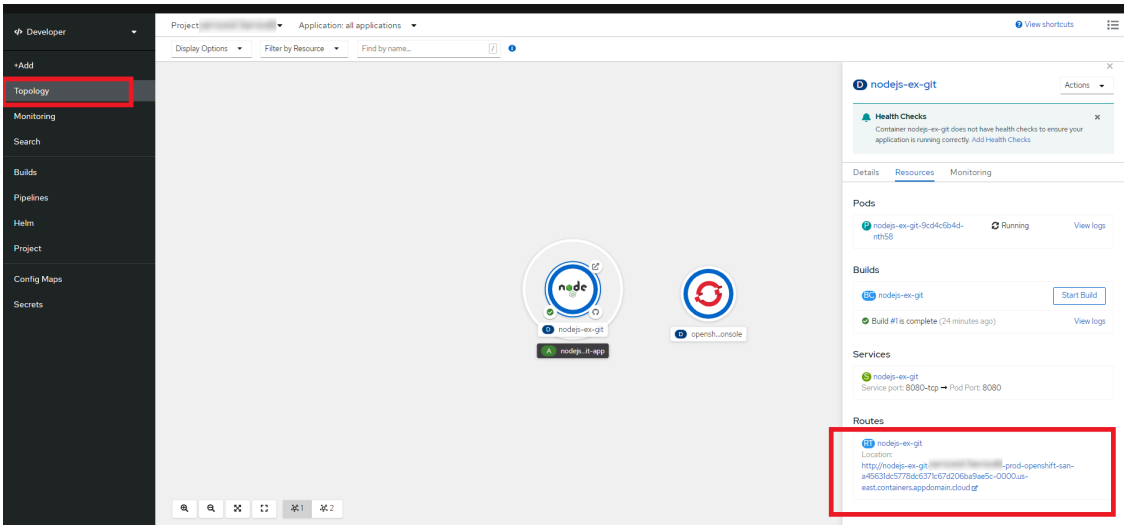


8. You can now see the ImageStreamTag that was created as an output of the build. Click the **History** tab to see the image in the internal registry to which this ImageStreamTag points.
现在你可以看到作为构建输出的 ImageStreamTag。点击 History 标签页，查看该 ImageStreamTag 指向的内部注册表中的镜像。



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.
返回拓扑视图，点击您的部署信息。点击 OpenShift 自动为您创建的路由。这将使用浏览器打开应用程序。

   Note: Please note down this URL as it will be used in the next section
   注意：请记下此 URL，因为它将在下一节中使用。



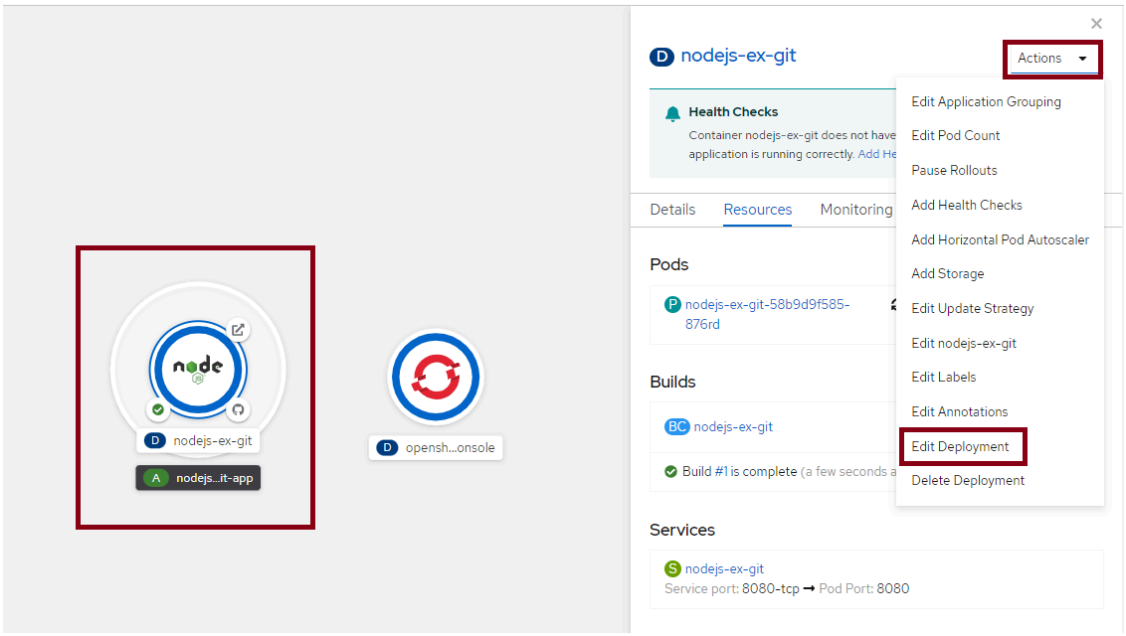
Autoscaling the `nodejs-ex-git` application
自动扩展 `nodejs-ex-git` 应用程序

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.
现在 `nodejs-ex-git` 应用程序已经成功运行，让我们设置一个水平 pod 自动缩放器（HPA），以便它可以处理任何到来的负载。确保在浏览器标签中保持 `nodejs-ex-git` 应用程序打开，以便它继续发起请求并消耗资源，以便能够成功自动缩放。

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.
首先，我们需要为将要运行的容器设置资源请求和限制。如果一个容器请求像 CPU 或内存这样的资源，Kubernetes 将只在能够提供该资源的节点上调度它。另一方面，限制可以防止容器消耗超过一定数量的资源。

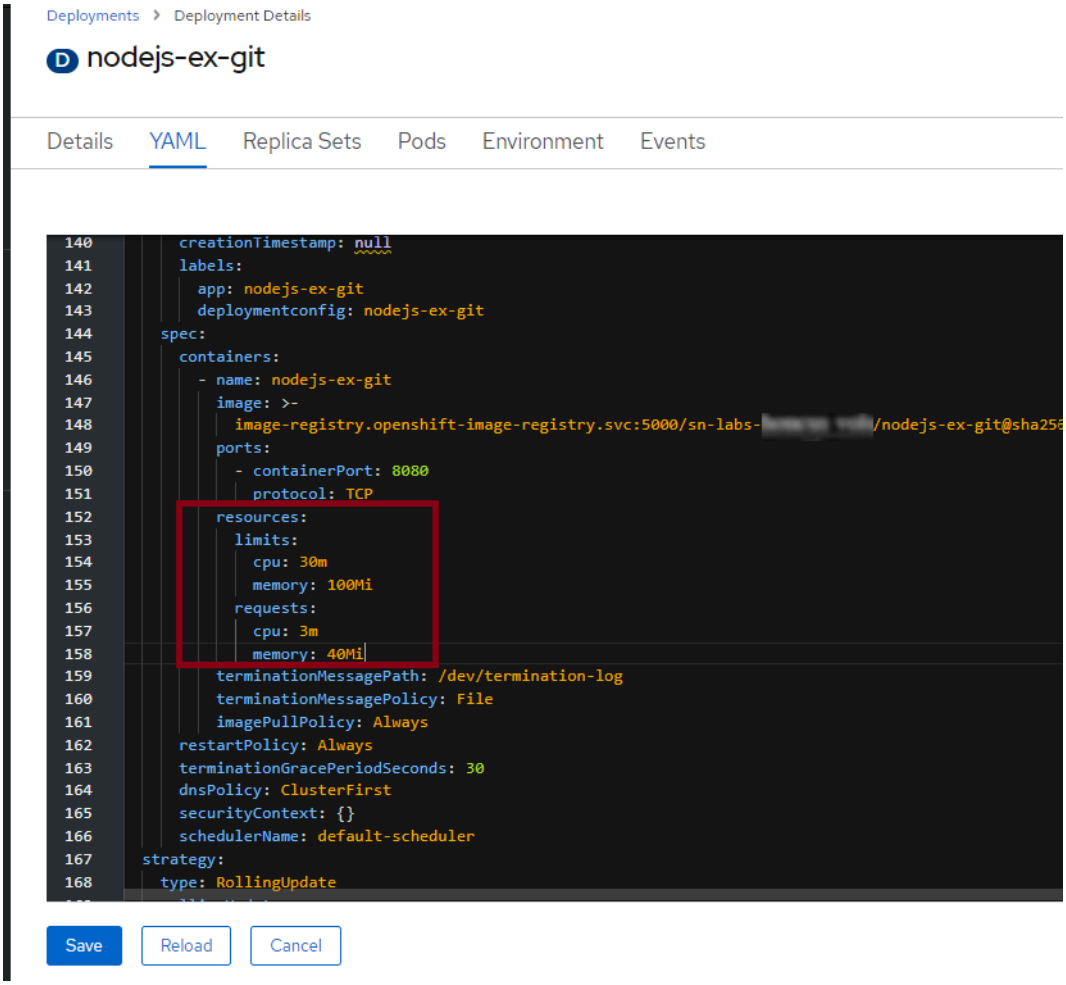
In this case, we're going to request 3 millicores of CPU and 40 MB of RAM. We'll limit the containers to 30 millicores and 100 MB. These numbers are contrived in order to ensure that the app scales.
在这种情况下，我们将请求 3 毫核的 CPU 和 40 MB 的 RAM。我们将容器的限制设置为 30 毫核和 100 MB。这些数字是虚构的，以确保应用程序能够缩放。

1. From the Topology view, click the `nodejs-ex-git` Deployment. Then click Actions > Edit Deployment.
从拓扑视图，点击 `nodejs-ex-git` 部署。然后点击操作 > 编辑部署。



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.
在template.spec.containers部分查找资源：{}。将其替换为以下文本。确保空格正确，因为 YAML 使用严格的缩进。

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

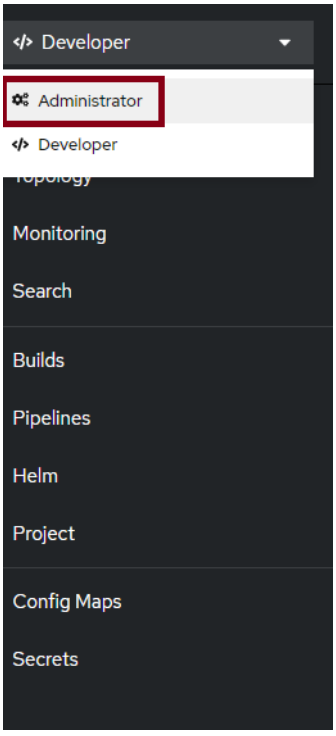


3. Click Save. 点击保存。

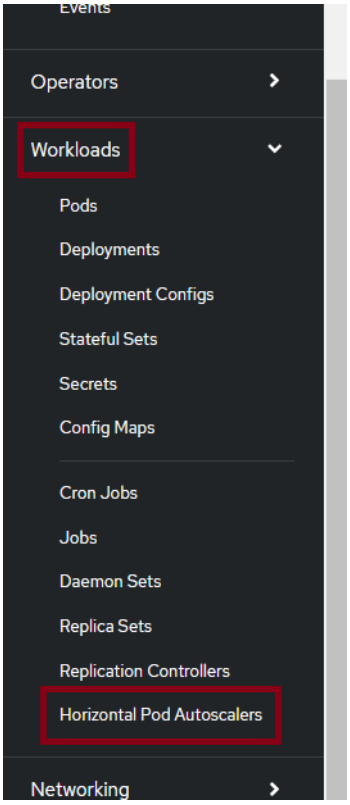
```
    creationTimestamp: null
    labels:
      app: nodejs-ex-git
      deploymentconfig: nodejs-ex-git
  spec:
    containers:
      - name: nodejs-ex-git
        image: >-
          image-registry.openshift-image-registry.svc:5000/sn-labs-
        ports:
          - containerPort: 8080
            protocol: TCP
        resources:
          limits:
            cpu: 30m
            memory: 100Mi
          requests:
            cpu: 3m
            memory: 40Mi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        imagePullPolicy: Always
    restartPolicy: Always
    terminationGracePeriodSeconds: 30
    dnsPolicy: ClusterFirst
    securityContext: {}
    schedulerName: default-scheduler
strategy:
  type: RollingUpdate
```

Save Reload Cancel

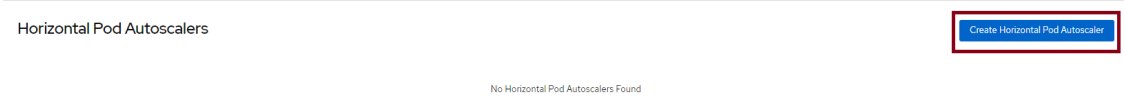
4. Switch to the Administrator perspective.
   切换到管理员视角。



5. Select Workloads > Horizontal Pod Autoscalers
   选择工作负载 > 水平 Pod 自动缩放器



6. Click Create Horizontal Pod Autoscaler
   点击创建水平 Pod 自动缩放器



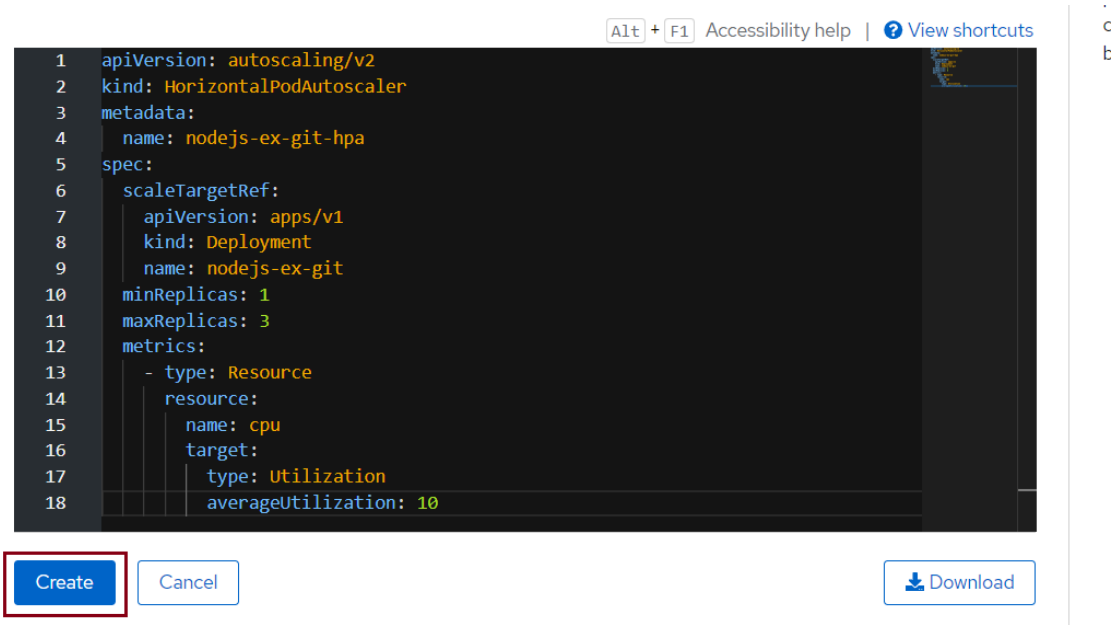
7. Paste the following YAML into the editor
   将以下 YAML 粘贴到编辑器中

```
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
```

```yaml
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
```

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.
此 HPA 表示我们将基于 CPU 使用情况进行缩放。通常，当您的 CPU 利用率在 50–90%范围内时，您希望进行缩放。在这个例子中，我们将使用 10%，这样应用程序更有可能需要缩放。minReplicas 和 maxReplicas 字段表示，根据负载，部署在任何给定时间应具有 1 到 3 个副本。

8. Click Create 点击创建



9. Run the below command on the terminal in Theia to increase the load on the `nodejs-ex-git` and view the Autoscaling:
在 Theia 终端中运行以下命令以增加 `nodejs-ex-git` 的负载并查看自动扩展：

```
for i in $(seq 1000); do curl -s -L <your app URL> & done
```

Note: Replace `<your app URL>` with the URL that you obtained in Step 9 of the previous section.
注意：请将 `<your app URL>` 替换为上一节第 9 步中获得的 URL。

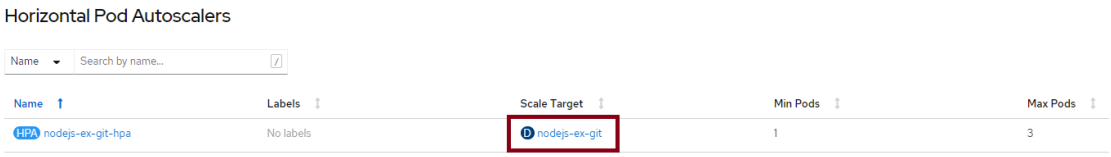
```
theia@theiaopenshift-nikeshkr:/home/project$ for i in $(seq 1000); do curl -s -L https:
labs-nikeshkr.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.con
loud/ & done
```

The command will keep giving an output as below indicating successful load generation:
该命令将不断输出如下内容，指示负载生成成功：

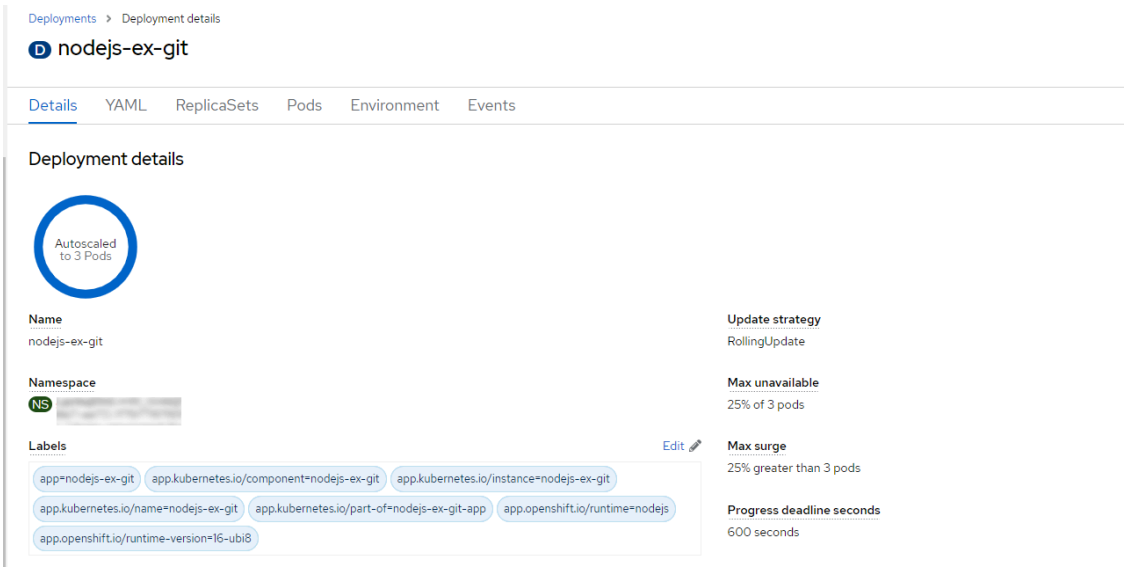


Note: You can also verify autoscaling by directly executing your app URL in your browser. Add some fruit names along with their quantities on the application UI, and click 'Save'. Although the added fruits may not appear on the UI, this action will trigger a load change, causing the pods to autoscale to 3 after some time.
注意：您也可以通过在浏览器中直接执行您的应用 URL 来验证自动扩展。在应用 UI 中添加一些水果名称及其数量，然后点击“保存”。尽管添加的水果可能不会在 UI 中显示，但此操作将触发负载变化，导致一段时间后 pods 自动扩展到 3 个。

10. Click on `nodejs-ex-git` under `Scale Target`.
    点击 `Scale Target` 下的 `nodejs-ex-git` 。



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.
    如果你等待，你会看到 Current Replicas 和 Desired Replicas 都变成三个。这是因为 HPA 检测到足够的负载来触发扩展到最大 Pod 数量，即三个。你也可以查看 Last Scale Time 以及当前和目标的 CPU 利用率。目标是显然 1%，因为我们就是设置成这个值。请注意，触发扩展可能需要几分钟。



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.
哇！OpenShift 为你做了相当出色的工作。它只需要一个代码仓库，就能够将代码构建成容器镜像，将镜像推送到注册中心，创建一个引用该镜像的 Deployment，并且还通过主机名将应用程序暴露到互联网上。

Congratulations! You have completed the lab for the fourth module of this course.
恭喜！你已经完成了本课程第四模块的实验。

© IBM Corporation. All rights reserved.