Table of Contents

Resource Exploration Lab

Scenario

In this lab you deliver a demonstration of deploying an application in Red Hat® OpenShift® Container Platform. You portray a developer who wants to deploy a Node.JS example application from a Git repository and review the different OpenShift resources created for that application.

You take **Actions** in the OpenShift Container Platform interface, and **Explain**: activities, product features, resources, and concepts.

Goals

- Understand the web console perspectives: Developer and Administrator
- Create a project
- Build and deploy an application from a Git repository
- Become familiar with the web console's Topology view
- Associate resources as part of an application grouping
- Understand the Administrator view of deployed resources

---

1. Provision Lab Environment

If you previously set up an environment for this class, you can skip this section and go directly to the Demonstrate Authentication to OpenShift section.

In this section, you provision the lab environment to provide access to all of the components required to perform the labs, including access to the shared OpenShift cource cluster. The lab environment is a shared cloud-based environment, so that you can access it over the Internet from anywhere. However, do not expect performance to match a dedicated environment.

You need to request **two** catalog items for this course! Make sure you request both the **OPENTLC OpenShift 4.6 Shared Access** and **OpenShift 4 Student VM** catalog items.

1. Go to the OPENTLC lab portal (https://labs.opentlc.com/) and use your OPENTLC credentials to log in.

If you do not remember your password, go to the OPENTLC Account Management page (https://www.opentlc.com/account/) to reset your password.

2. You will need to request two catalog items.

The first catalog item grants you access to a shared OpenShift Cluster: .. Navigate to **Services → Catalogs → All Services → OPENTLC OpenShift 4 Labs**. .. On the left, select **OPENTLC OpenShift 4.6 Shared Access**. .. On the right, click **Order**. .. On the bottom right, click **Submit**.

3. The second catalog item creates a Red Hat Enterprise Linux Virtual Machine with all necessary tools pre-configured. This way you have a well-defined environment to run your lab commands in.
   a. Navigate to **Services → Catalogs → All Services → OPENTLC OpenShift 4 Labs**.
   b. On the left, select **OpenShift 4 Student VM**.
   c. On the right, click **Order**.
   d. From the **OpenShift Release** dropdown select **4.6**
   e. From the **Region** dropdown select a region that is close to you.
   f. On the bottom right, click **Submit**.

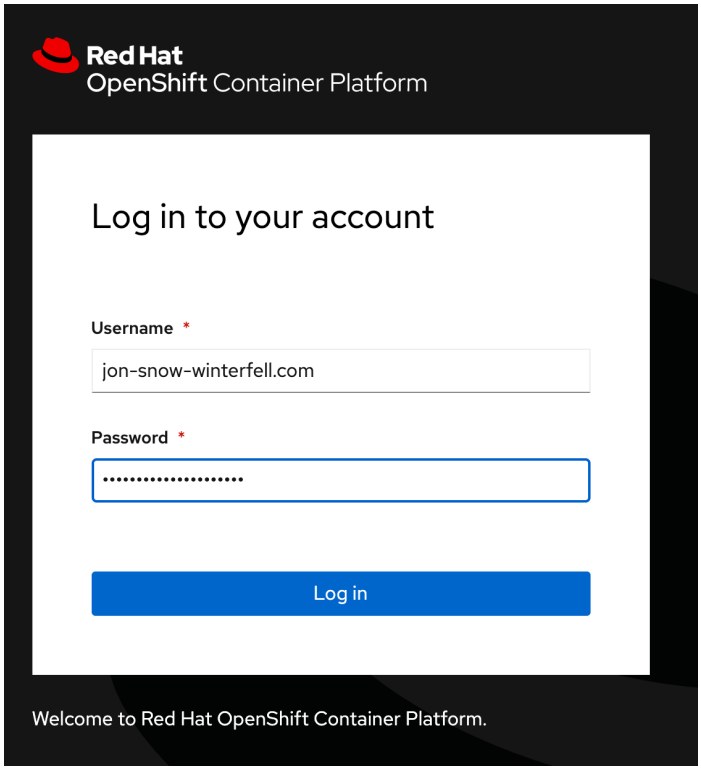
Do not select **App Control → Start** after ordering a catalog item.

   - After a few minutes, expect to receive two emails with instructions on how to connect to the environment.

4. While you are waiting for the environment to build, read the email carefully and specifically note the following:
   - One email includes a URL for the OpenShift web console similar to **https://console-openshift-console.apps.shared-na46.openshift.opentlc.com/dashboards (https://console-openshift-console.apps.shared-na46.openshift.opentlc.com/dashboards)**.
   - The other e-mail includes the SSH command and password to connect to the Student VM

2. Demonstrate Authentication to OpenShift

1. **Action**: Log in to the OpenShift Container Platform web console using your login credentials.



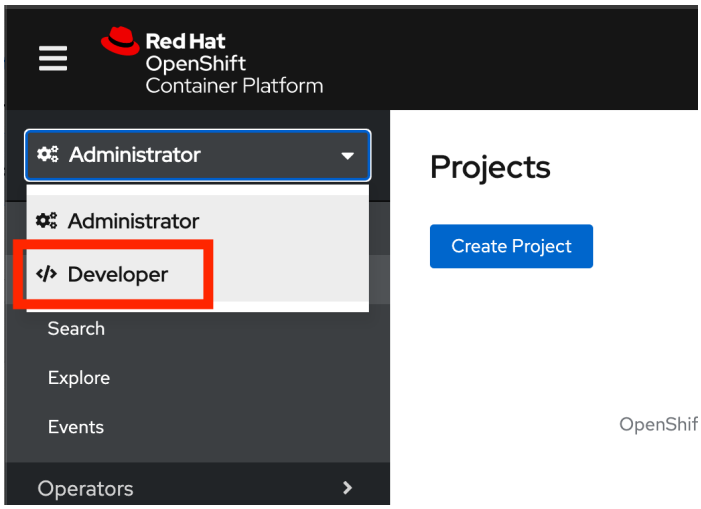
   - **Explain**: You are taken to the **Projects** page. The default view for the OpenShift Container Platform web console is the Administrator perspective.

---

3. Demonstrate the Developer Perspective

Explain: The OpenShift Container Platform web console provides two perspectives: the Administrator perspective and the Developer perspective. The Developer perspective provides workflows specific to developer use cases.

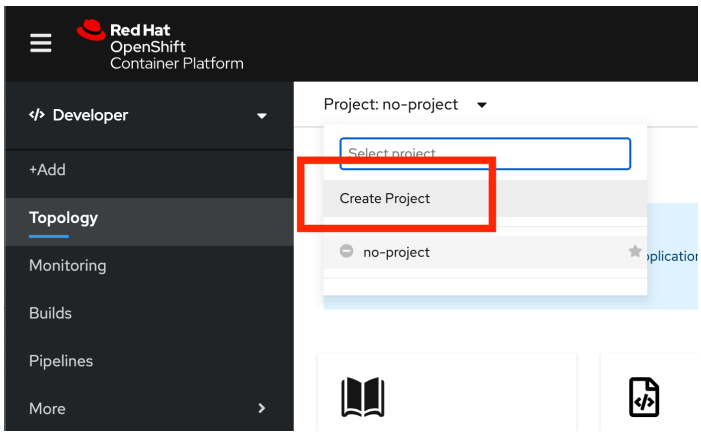
1. **Action**: Use the perspective switcher to switch to the Developer perspective. The Topology view with options to create an application is displayed.



3.1. Demonstrate Project Creation

Explain: A project allows a community of users to organize and manage their content in isolation from other communities. Projects are OpenShift extensions to Kubernetes namespaces, with additional features to enable user self-provisioning. For the vast majority of purposes, projects and namespaces are interchangeable.

1. **Action**: Click the **Project** drop-down menu to see a list of all available projects. Select **Create Project**.

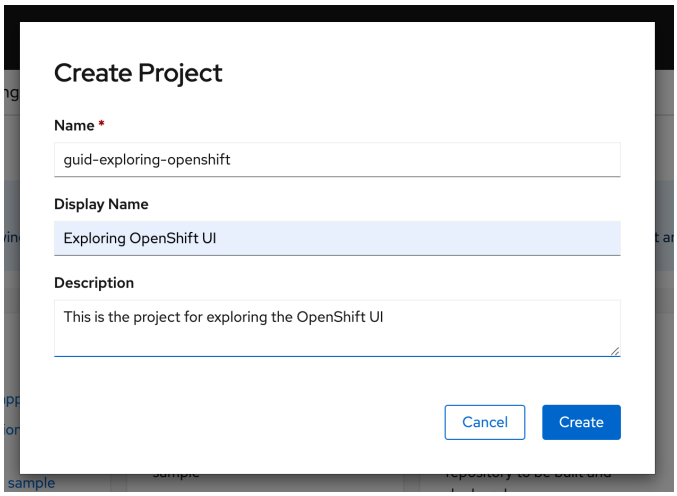


   - **Explain**: Different projects can have different user permissions and quotas attached to them.

2. **Action**: Fill in the **Name**, **Display Name**, and **Description** fields as follows:
   - **Name**: `guid-exploring-openshift`

Make sure you replace `guid` with a unique identifier, such as your client's name or the four-character identifier you received from OPENTLC. Project names must be unique in OpenShift.

- **Display Name**: `Exploring OpenShift UI`
- **Description**:
  `This is the project for exploring the OpenShift UI`

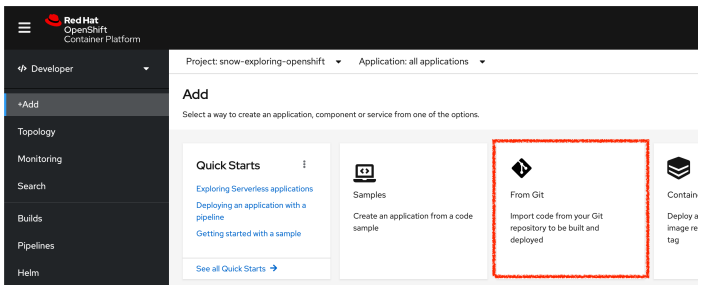


3.2. Demonstrate Application Deployment

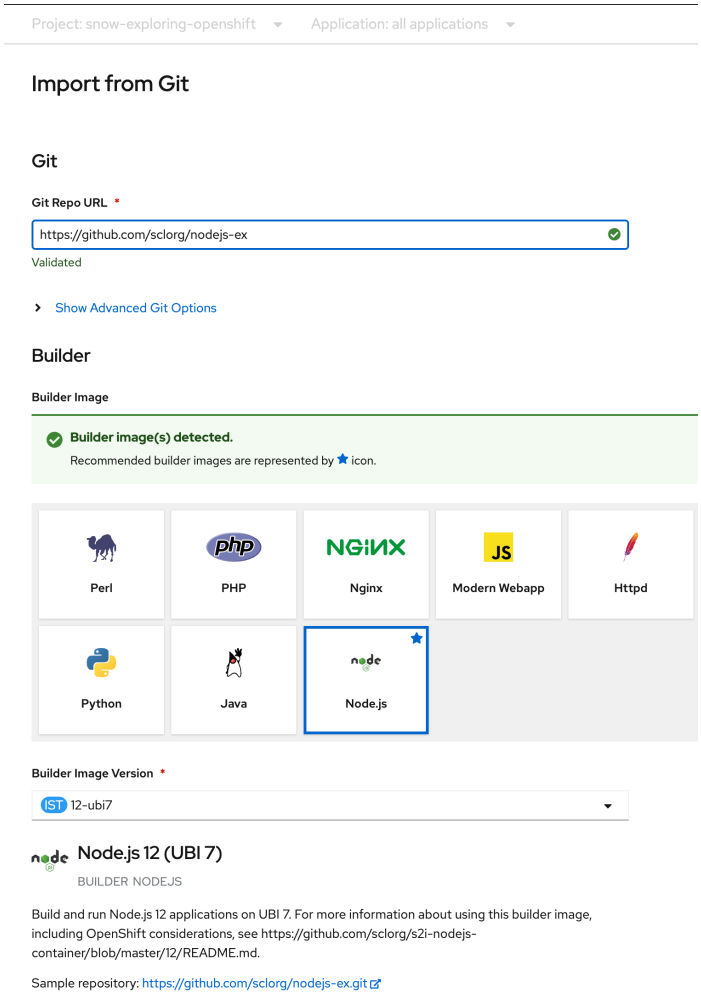
Explain: The Developer perspective in the web console provides options from the Add view to create applications and associated services and deploy them on OpenShift Container Platform.

Explain: The following procedure walks you through the **Import from git** option in the Developer perspective to create an application. Create, build, and deploy an application on OpenShift Container Platform using an existing codebase in GitHub as follows:

1. **Action**: In the Add view, click **From Git** to see the **Import from git** form.



2. **Action**: In the Git section, enter the Git repository URL for the codebase you want to use to create an application. For example, enter the URL of this sample Node.js application https://github.com/sclorg/nodejs-ex (https://github.com/sclorg/nodejs-ex). The URL is then validated.



- **Explain**: In the Builder section, after the URL is validated, an appropriate builder image is detected, indicated by a star, and automatically selected. For the `https://github.com/sclorg/nodejs-ex` Git URL, the Node.js builder image is selected by default. If required, you can change the version using the Builder Image Version drop-down list.

3. **Action**: In the General section, do the following:

   a. Enter a unique name in the **Application** field for the application grouping, for example, "nodejs-ex-app". Ensure that the application name is unique in a namespace.

   b. Enter a unique name in the **Name** field to identify the resources created for this application. This is automatically populated based on the Git repository URL.

> The resource name must be unique in a namespace. Modify the resource name if you get an error.

4. **Action**: In the **Resources** section, select the default resource: **Deployment**.

- **Explain**: Deployment creates an application in plain Kubernetes style. "Deployment Config" creates an OpenShift style application. "Knative Service" creates a microservice.

The Knative Service option is displayed in the Import from git form only if the Serverless Operator is installed in your cluster. For further details, refer to documentation on installing OpenShift Serverless.

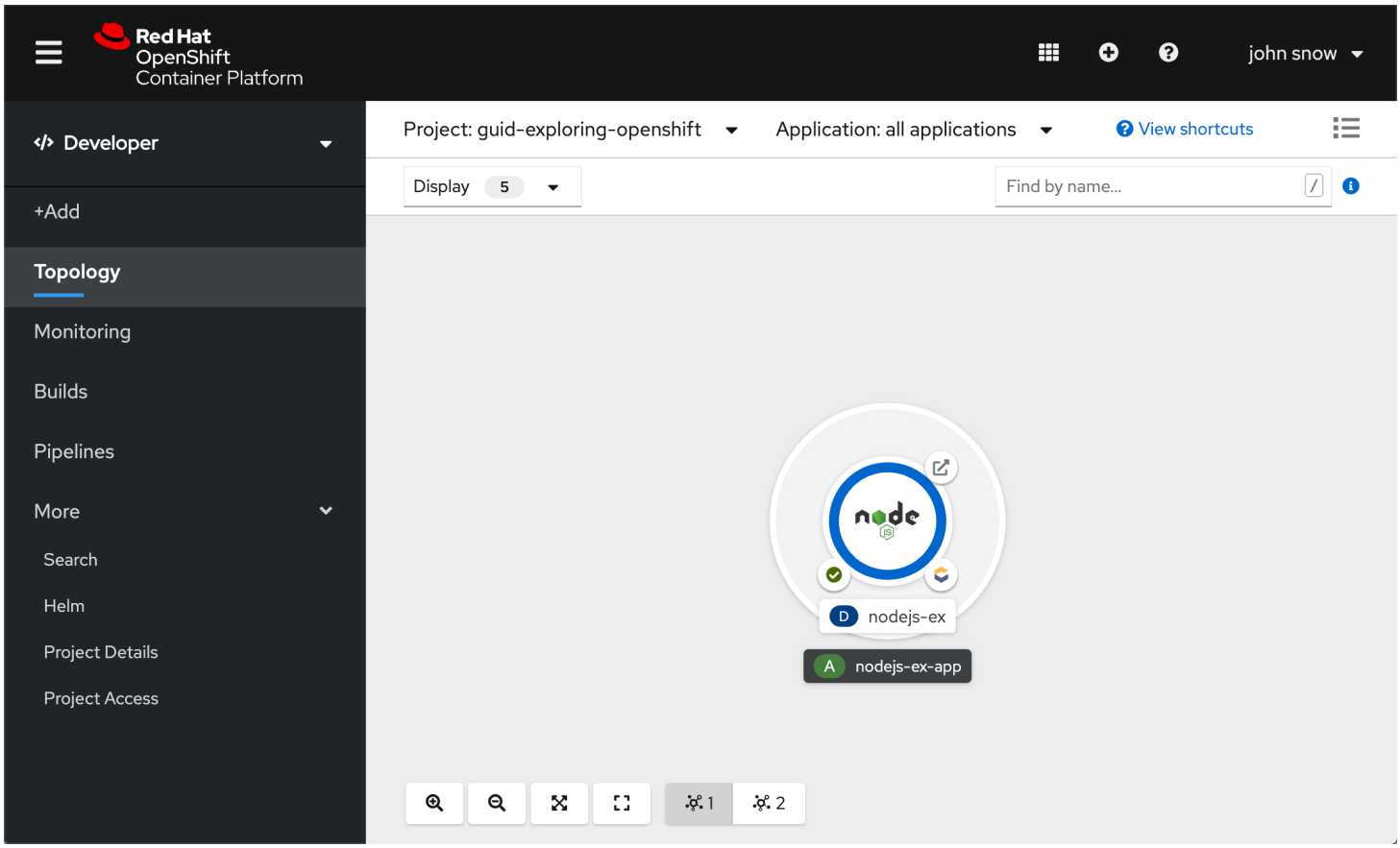
- **Explain**: In the **Advanced Options** section, **Create a route to the application** is selected by default so that you can access your application using a publicly available URL. You can clear the check box if you do not want to expose your application on a public route.

5. **Action**: Click **Create** to create the application and see its build status in the Topology view.

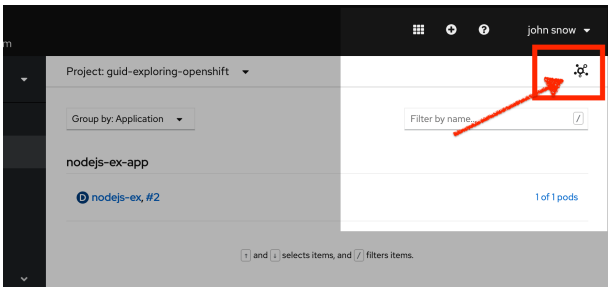
The Topology view might show "1 Error." This is OK. Wait for the build to complete and the Deployment #2 to finish. You can still expect your application to build and deploy successfully.

3.3. Demonstrate Topology View

Explain: The Topology view in the Developer perspective of the web console provides a visual representation of all of the applications within a project, their build status, and the components and services associated with them.



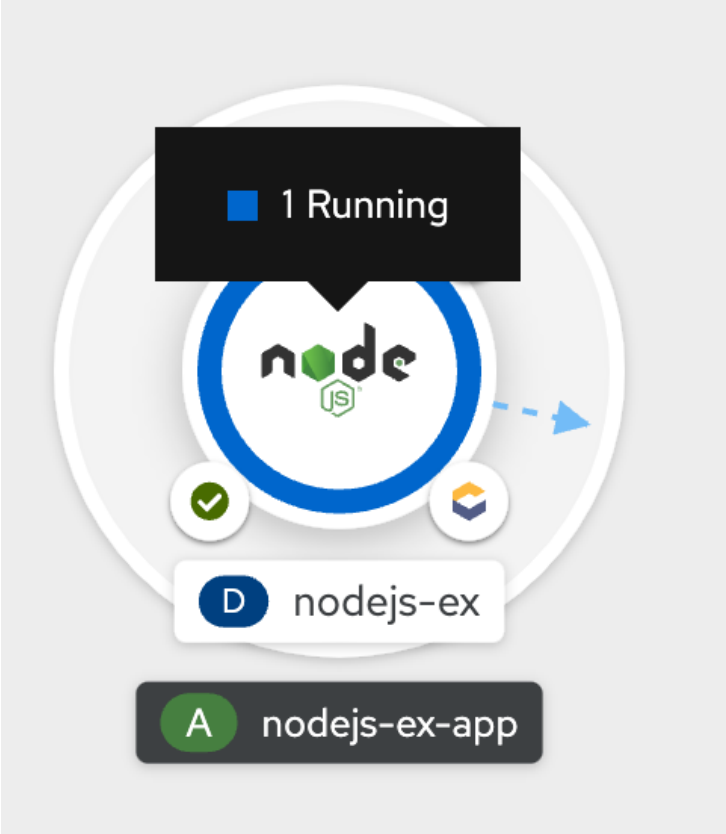
Action: If you do not see a graphical representation, click the Topology View icon on the top right of the web console.



Explain: You can navigate to the Topology view using the left navigation panel in the Developer perspective. After you create an application, you are directed automatically to the Topology view where you can see the status of the application Pods, quickly access the application on a public URL, access the source code to modify it, and see the status of your last build. You can zoom in and out to see more details for a particular application.

Explain: The status or phase of the Pod is indicated by different colors and tooltips as Running (■), Not Ready (■), Warning (■), Failed (■), Pending (■), Succeeded (■), Terminating(■), or Unknown(■). For more information about Pod status, see the Kubernetes documentation.

Explain: After you create an application and an image is deployed, the status is shown as Pending. After the application is built, it is displayed as Running.



- Point out that the application resource name (nodejs-ex) is appended with indicators for the different types of resource objects as follows:
  - **D**: Deployment
  - **DC**: Deployment Config
  - **SS**: StatefulSet
  - **DS**: Daemonset

Point out that in addition to OpenShift Deployment Configs, Kubernetes *Deployments* are also supported. Kubernetes Deployments share many of the features available in Deployment Configs, and are the default deployment resource object from OpenShift Container Platform 4.5.

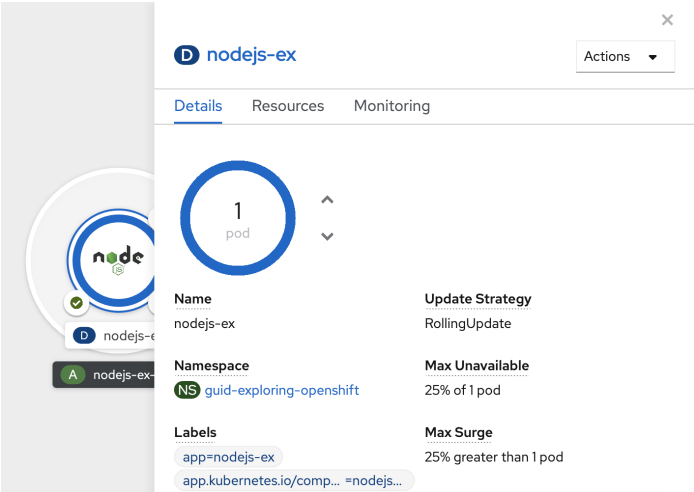
3.4. Demonstrate Interacting with the Overview Panel

Explain: There is an Overview panel that can open to access many features of the Deployment.

1. **Action**: Click the **D nodejs-ex** to open the overview panel.

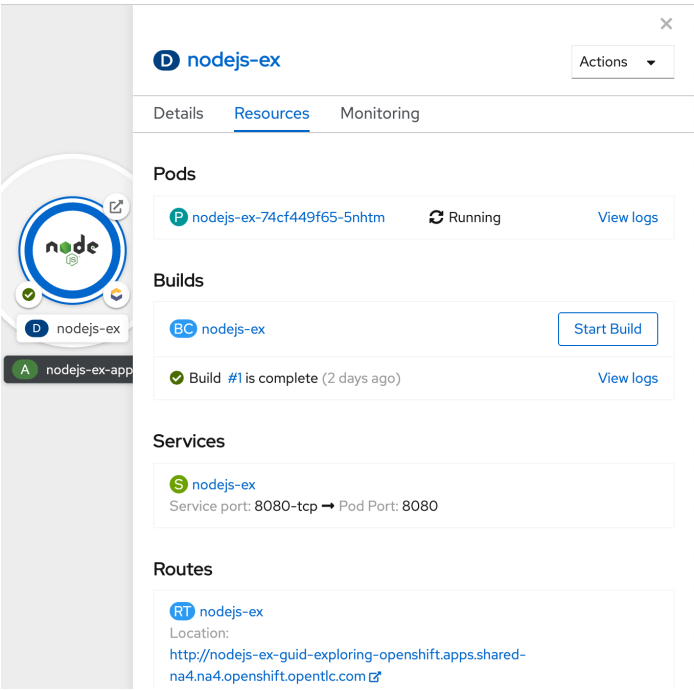
- **Explain**: Point out that the details of the nodejs-ex Deployment are available here. This includes Details, Resources, and Monitoring.

2. **Action**: Click the **Details** tab of the overview panel.



- **Explain**: Point out the following aspects:
  - The Pod replica count can be managed here. You can scale your Pods using the up and down arrows to increase or decrease the number of instances of the application manually. For serverless applications, the Pods are automatically scaled down to zero when idle and scaled up depending on the channel traffic.
  - The presence of various aspects of the Pod configuration and the clickable contextual help. Check the Labels, Annotations, and Status of the application.

3. **Action**: Click the **Resources** tab.



- **Explain**: Point out the following aspects:
  - Several **Resources** were created by the Deployment, and their status is indicated here.
  - These are just some of the resources associated with the Deployment.
    - **Pods** are the basic units of OpenShift applications, with access to their logs.
    - **Builds** were created to compile the source code and package it into *images*. See their status, access logs, and start a new build if needed.
    - **Services** were created for your Pod, and assigned ports are listed.
    - **Routes** were created to permit external access to the Pods, and the URL to access them is listed.

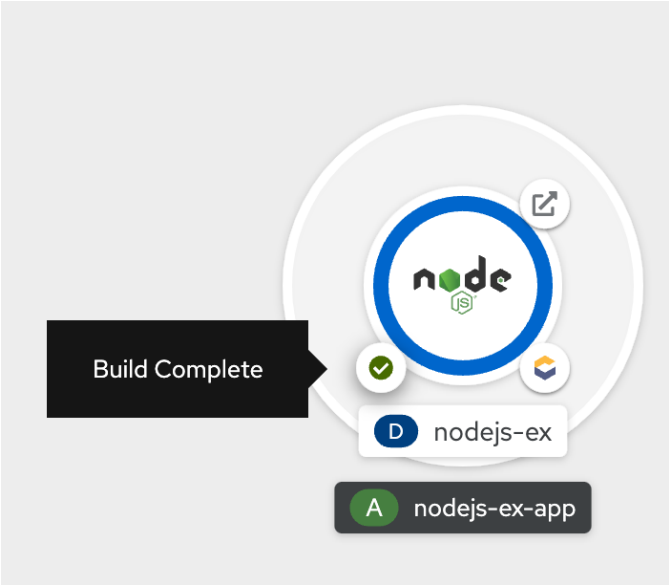
4. **Action**: Click the **Monitoring** tab of the review panel.
   - **Explain**: Point out the various **Events** and **Metrics** information that appears on this page.
5. **Action**: Close the review panel by clicking the **X** or anywhere in the topology view field.

For serverless applications, the Resources tab provides information on the revision, routes, and the configurations used for that component.

3.5. Demonstrate Interacting with the Application and Components

Explain: The Topology view in the Developer perspective of the web console provides the following options to interact with the application and the components.

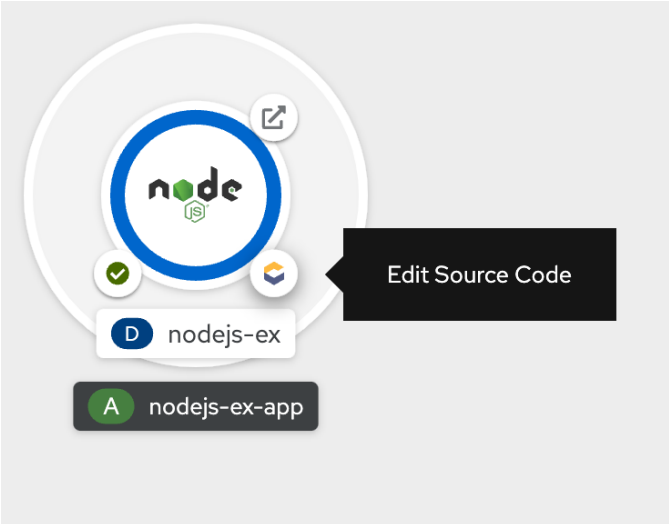
1. **Action**: Hover your cursor over the lower left icon on the Pod to see the name of the latest build and its status.
   - Point out the Pod's build status indicator:



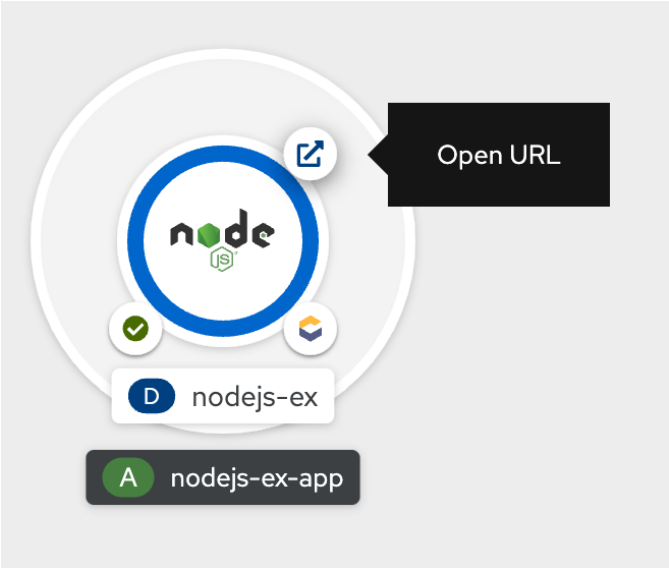
- **Explain**: The status of the application build is indicated as New, Pending, Running, Completed, Failed, or Canceled.

2. **Action**: Hover your cursor over the lower right icon on the Pod to see access to the source code.
   - **Explain**: You can open the source code in Code Ready Workspaces and edit the application code.

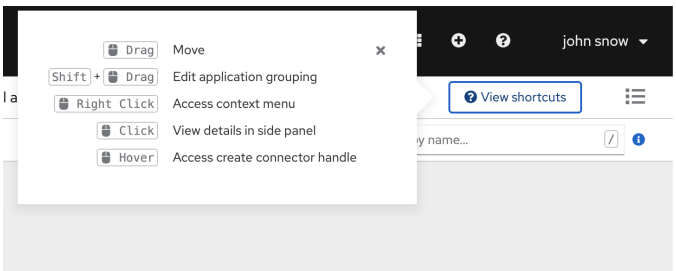
> This feature is available only when you create applications using the From Git, From Catalog, and From Dockerfile options.



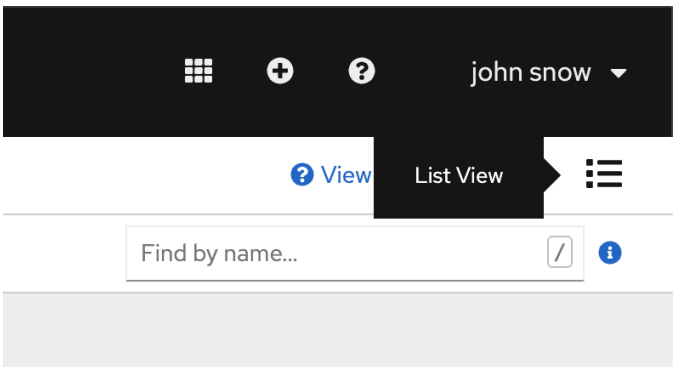
3. **Action**: Hover your cursor over the upper right icon to see the public URL.
   - Point out that the application URL is available:



4. **Action**: Use the Shortcuts menu listed on the upper-right of the screen to navigate components in the Topology view.
    - **Explain**: To make it easier to view many applications in the Topology view, there are keyboard and mouse shortcuts.



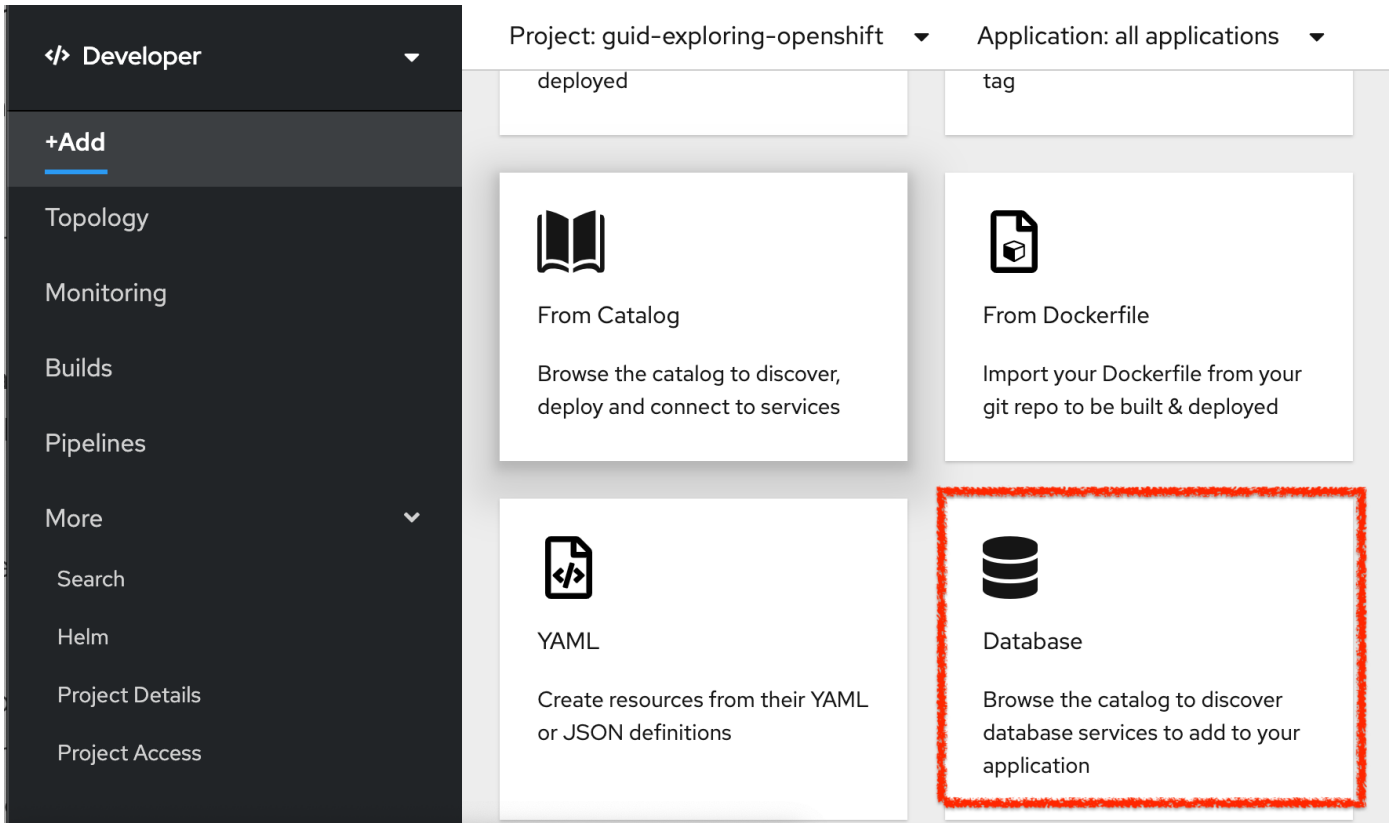
5. **Action**: Use the List View icon to see a list of all your applications and use the Topology View icon to switch back to the Topology view.
    - **Explain**: Sometimes you may want to see all your applications and their resources in list view.



3.6. Demonstrate Grouping Multiple Components within an Application

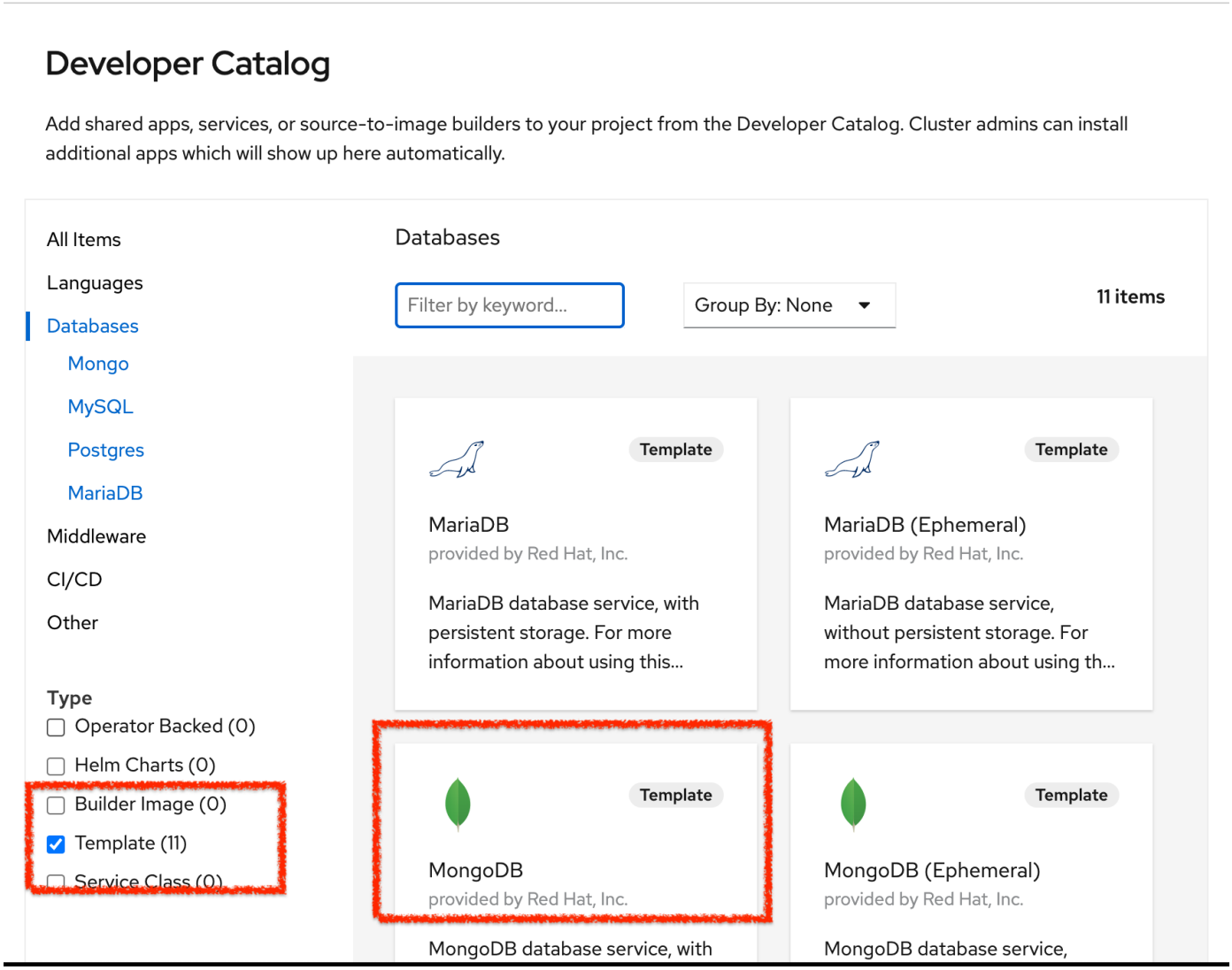
Explain: You can use the Add page to add multiple components or services to your project and use the Topology page to group applications and resources within an application group. The following procedure adds a MongoDB database service to an existing application with a Node.js component.

1. **Action**: Navigate to the Add view and select the **Database** option to see the Developer Catalog.



   - **Explain**: The catalog has multiple options that you can add as components or services to your application.

2. **Action**: Select the **Type: Template** checkbox to show OpenShift Template-based catalog items.



a. **Action**: Click the **MongoDB** (not MongoDB Ephemeral) option to see the details for the service.

b. **Action**: Click **Instantiate Template** to see an automatically populated template with details for the MongoDB service.

c. **Action**: Scroll to the bottom of the page and click **Create** to create the service.

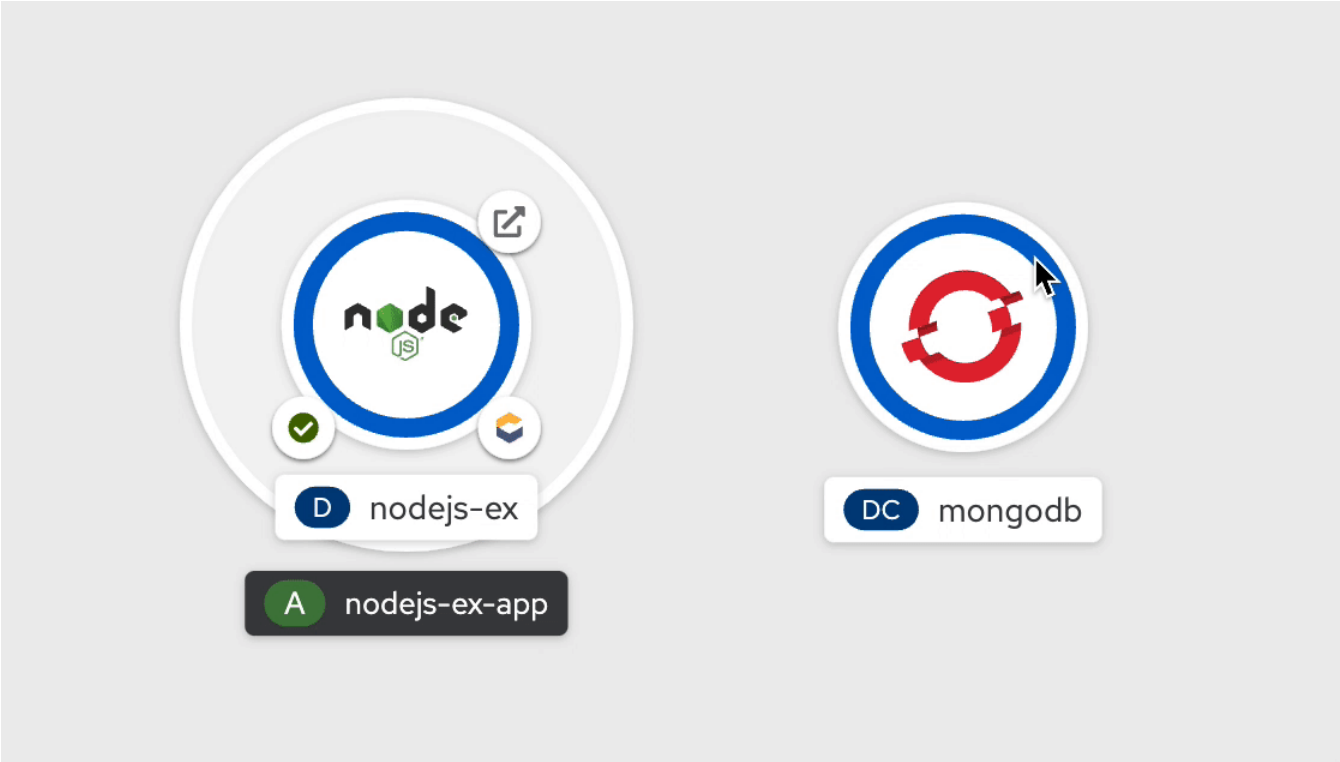
d. **Action**: You should be automatically directed to the Topology view to observe the deployment of the MongoDB.

   - Point out how the Pod circle changes color to indicate the status.

e. **Action**: Hover your cursor over the Pod to see the status.

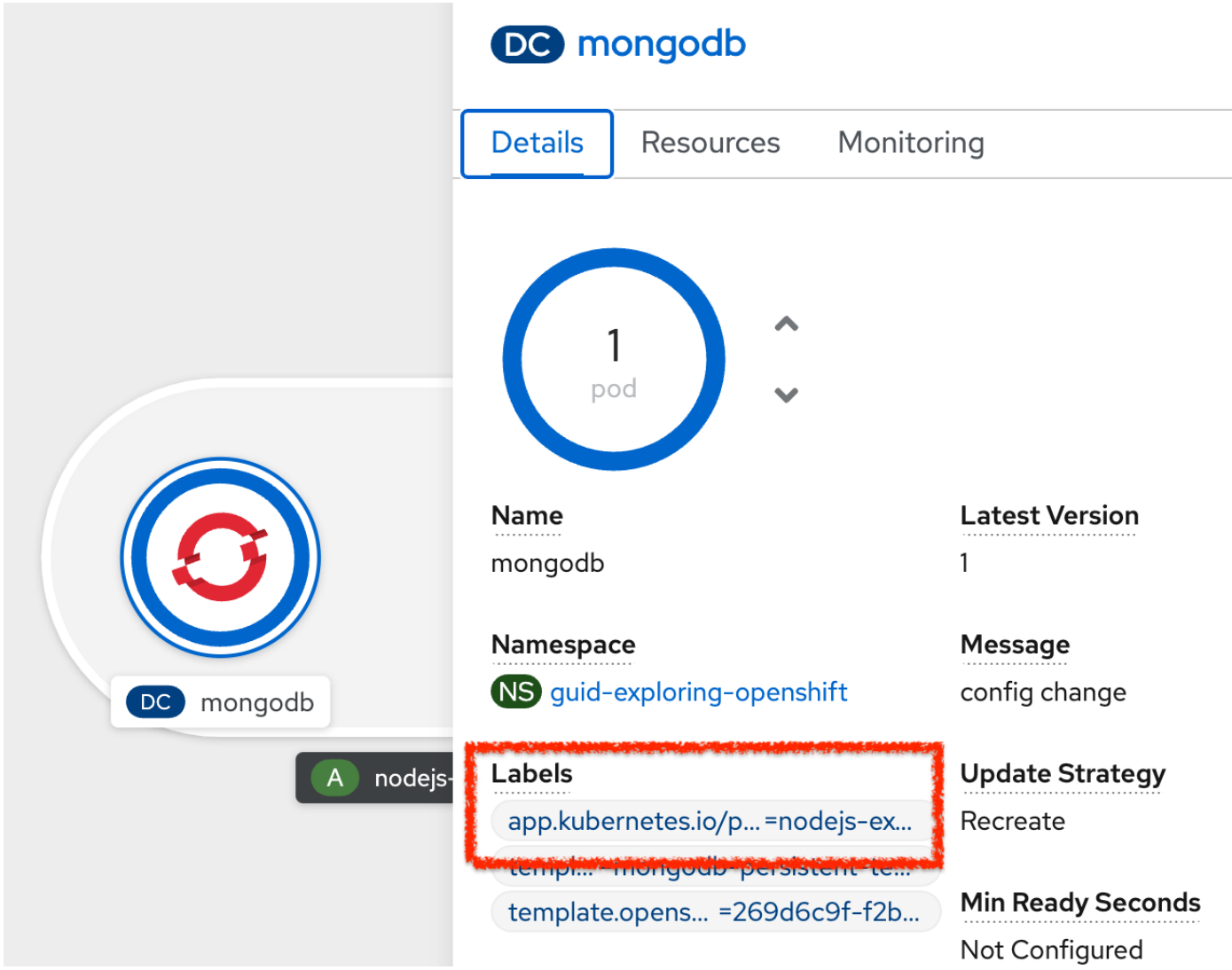
3. **Action**: On the left navigation panel, click **Topology** to see the MongoDB service deployed in your project.

   a. **Action**: To add the MongoDB service to the existing application group, hold **SHIFT**, select the `mongodb` Pod and drag it to the application; the MongoDB service is added to the existing application group.



- Point out that the application grouping (nodejs-ex-app) is labeled with an oval labeled **A**.
- **Explain**: Dragging a component while holding the SHIFT key and adding it to an application group automatically adds the required labels to the component.

4. **Action**: Click the MongoDB **DC mongodb** to open the Overview Panel, and click the **Details** tab.



- Point out that the label `app.kubernetes.io/part-of=nodejs-ex` is added to the Labels section.

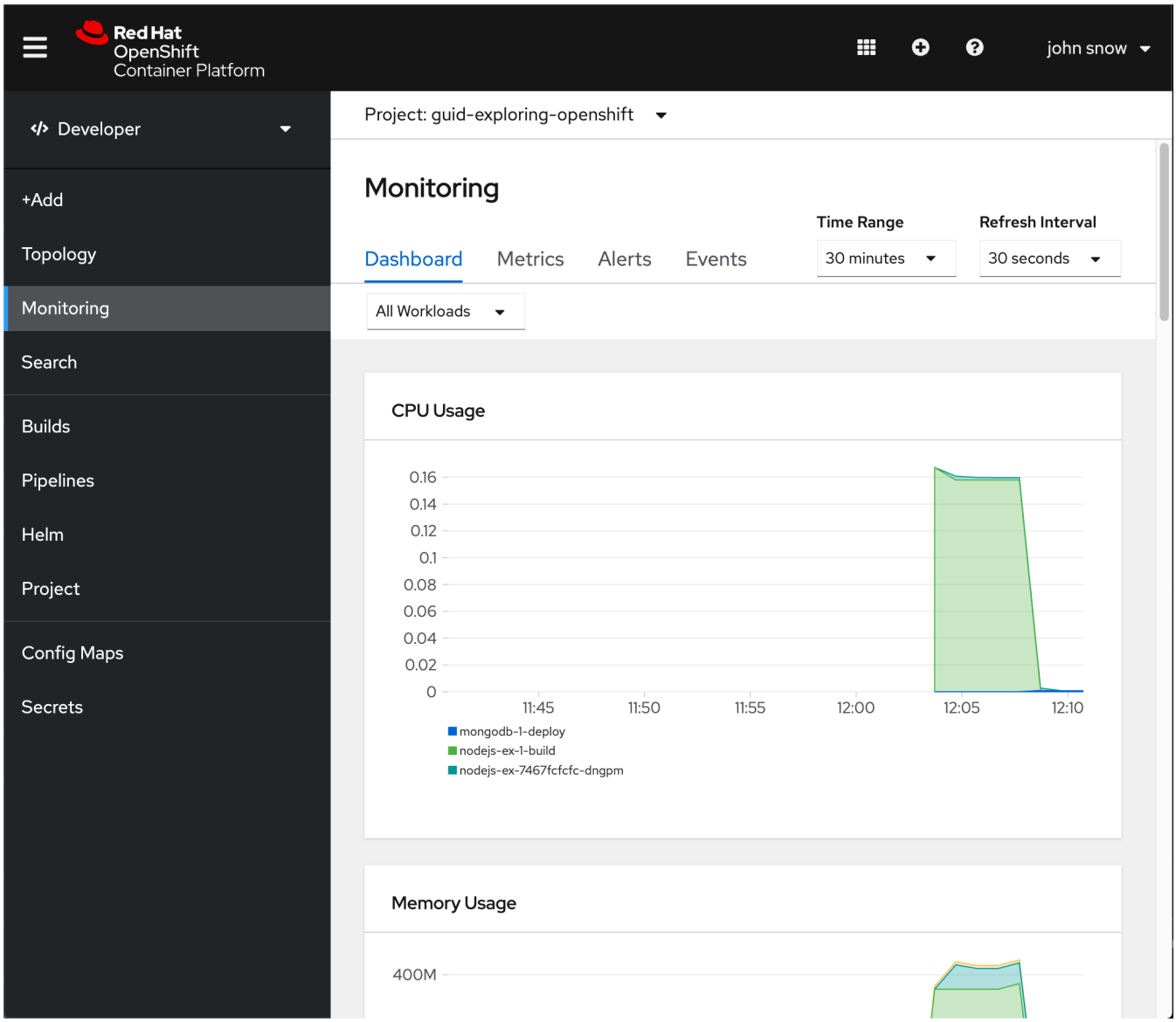
To integrate the app and the database, the Node.js application and the MongoDB database need to be configured elsewhere. Look out for Service Bindings based on Operators in future releases of OpenShift.

3.7. Demonstrate Exploring Monitoring Page in Developer Perspective

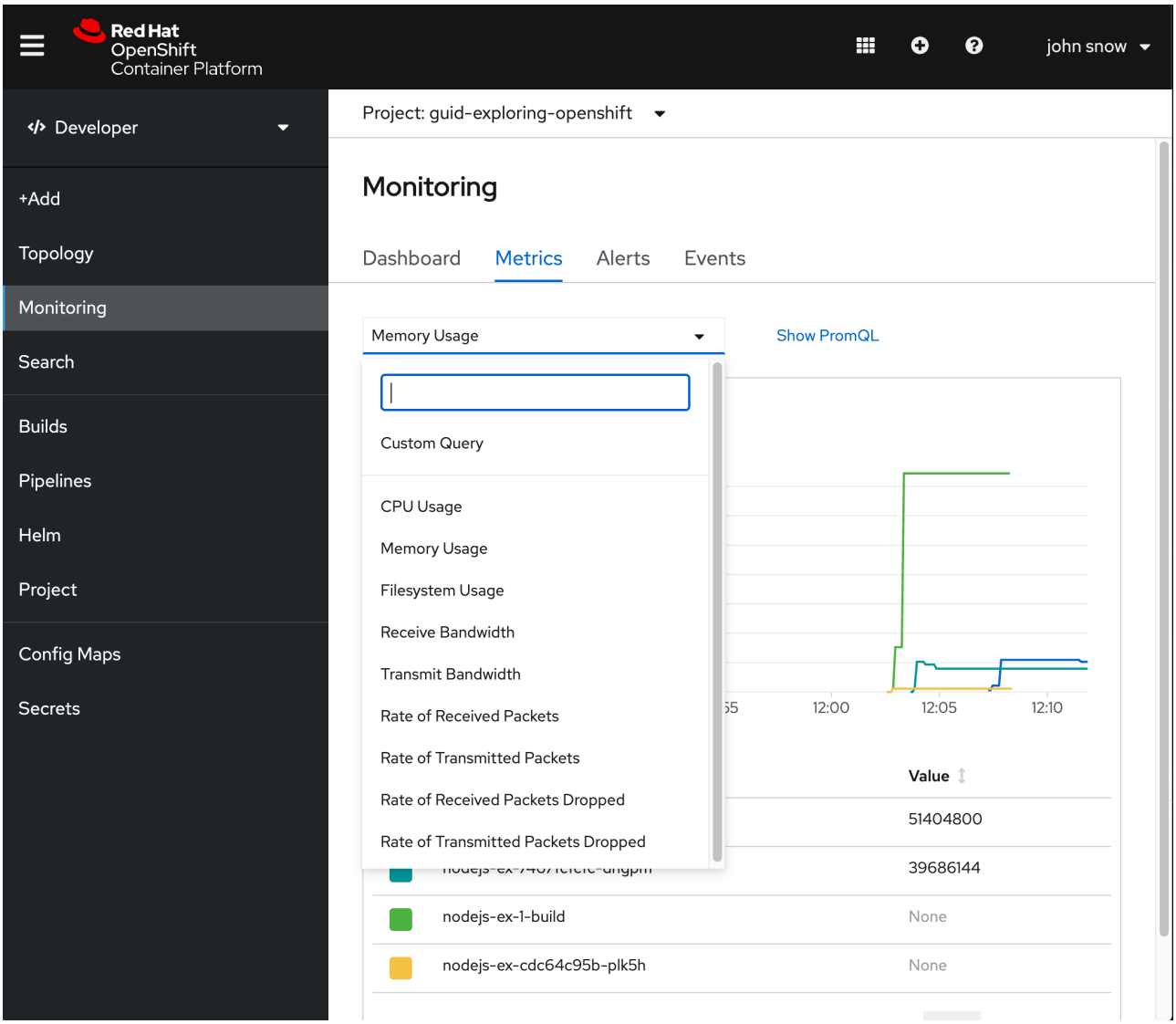
Explain: Red Hat has recently integrated monitoring features into the web console. Explore project-wide Metrics, Alerts, and Events here.

1. **Action**: On the left panel, click **Monitoring**:
   - Point out that these features are scoped only to the project selected above.
2. **Action**: Click the **Dashboard** tab:

- Point out the Dashboard, which shows combined metrics for the project.



3. **Action**: Click the **Metrics** tab:
   - Point out the Metrics tab, where custom graphs of Prometheus Metrics can be created.
     a. **Action**: Choose the **Memory Usage** query to see the memory usage of each pod in the namespace.



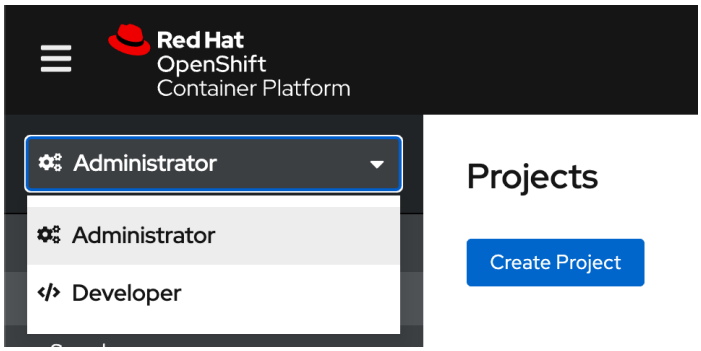
4. **Action**: Click on the **Alerts** tab:
    - Point out that no alerts have been defined yet.
5. **Action**: Click the **Events** tab:
    - Point out the Events tab, where events are reported as a stream, and may be filtered.

4. Demonstrate Deployments and Deployment Configs

4.1. Demonstrate Switching to the Administrator Perspective

Explain: Many advanced functions are only available through the Administrator Perspective.

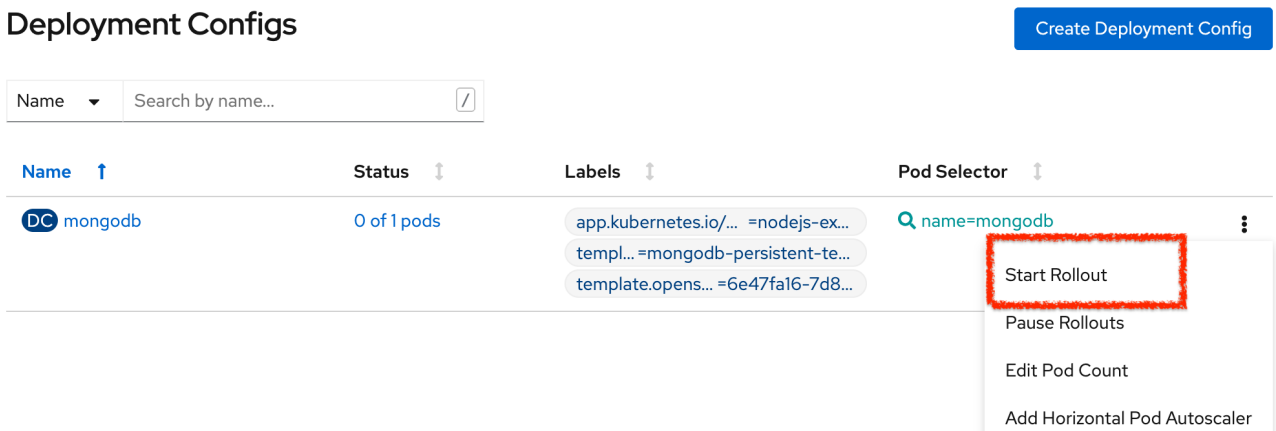
1. **Action**: From the Perspective drop-down box, select **Administrator**:



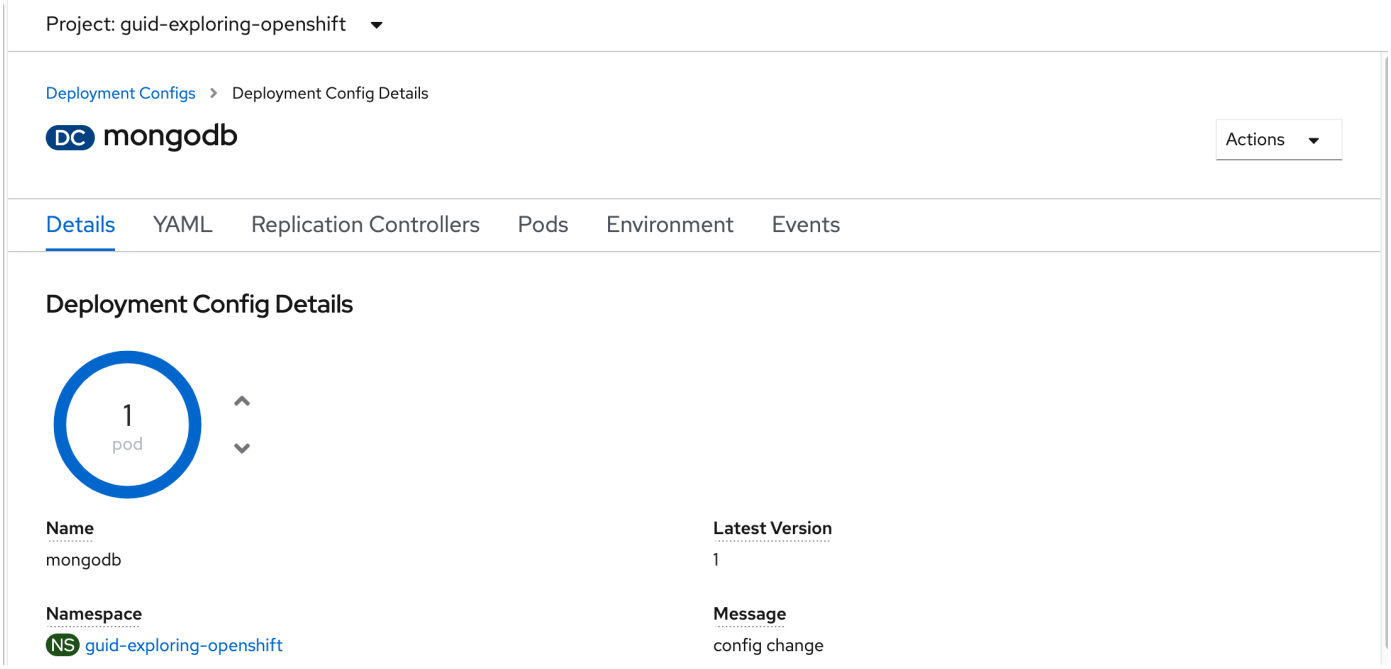
5. Demonstrate Deployments and Deployment Configs

Explain: *Deployments* and *Deployment Configs* in OpenShift Container Platform are API objects that provide two similar, but different, methods for fine-grained management over common user applications. They are composed of the following separate API objects: **A Deployment Config or a Deployment, either of which describes the desired state of a particular component of the application as a Pod template.** Deployment Configs involve one or more ReplicationControllers, which contain a point-in-time record of the state of a Deployment Config as a Pod template. Similarly, Deployments involve one or more ReplicaSets, a successor of ReplicationControllers. ** One or more Pods, which represent an instance of a particular version of an application.

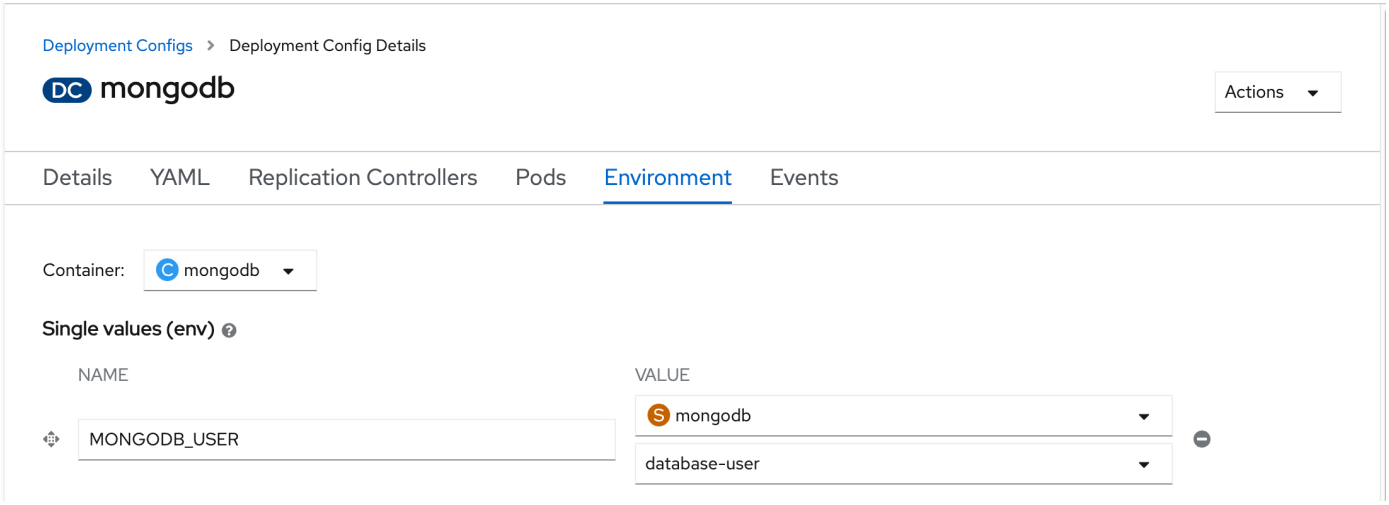
1. **Action**: From the left navigation, click **Workloads → Deployment Configs**
    - Point out that from the select menu on the right of each Deployment Config, you can start a new rollout of the Deployment Config desired.



2. **Action**: Click the `mongodb` Deployment Config from the list, and the **Deployment Config Details** page opens.

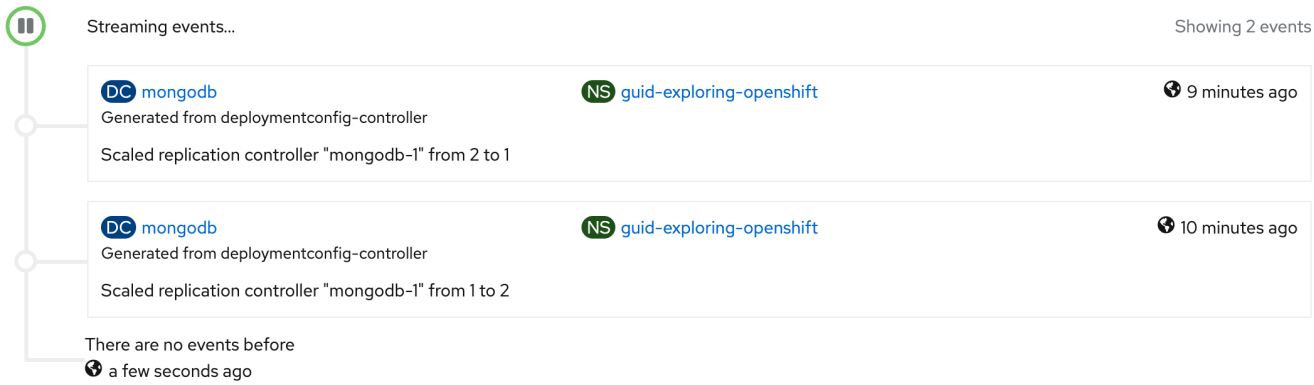


   - **Explain**: Point out the following:
     - The circle indicating status of the deployment of the Pods.
     - The *desired count* (aka replicas) of Pods can be easily increased or reduced from this panel by clicking the up and down arrows next to the circle.
     - You can see the following:
       - The **Latest Version** of the Deployment Config
       - The latest **Message** indicating why the current Deployment was started
       - The list of **Containers** further down the page, the images they started from, resource limits, and and the ports they listen on
       - The list of **Conditions**, which express the major events associated with this Deployment Config, including the `NewReplicationControllerAvailable` **Reason** and the `replication controller "mongodb-1" successfully rolled out` **Message**
     - You can also start a new deployment here by clicking the **Actions** menu on the upper right and selecting **Start Rollout**, and that this creates a new replication controller, as well.
3. **Action**: Select the **Environment** tab.



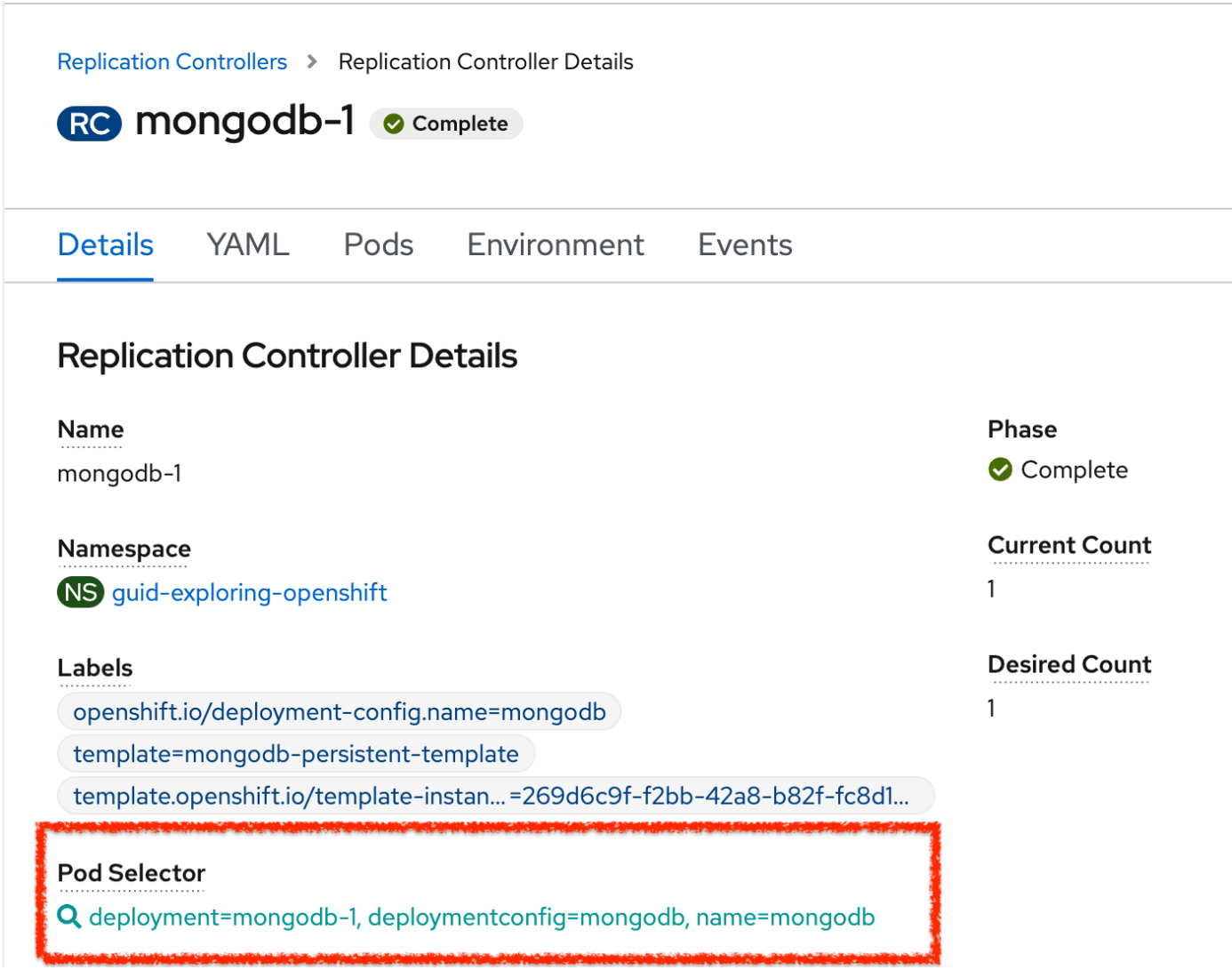
- **Explain**: The **Environment** tab displays the environment variables set for your Deployment.
  - Point out that the environment variables are used to set different parameters within your containers and Pods, such as user names, database service name, and more.
  - Point out that you can add your own environment variables and address them from your application's code.

4. **Action**: Select the **Events** tab.



- **Explain**: In the **Events** tab you can see the events related to your deployment.
  - Point out that the events list offers a useful way to see if something went wrong in a deployment or to trace back a chain of events.

5. **Action**: On the left, select **Workloads** → **Replication Controllers**.
   - **Explain**: The Deployment Configs create *replication controllers*.
6. **Action**: Click the name of the replication controller, **`mongodb-1`**, and the **Replication Controller Overview** page opens.



- **Explain**: Point out the following:
  - The `mongodb-1` replication controller has a Pod selector that denotes the version of the deployment and the name of the Deployment Config that created that deployment.
  - The replication controller manages the number of Pods actually rolled out, as requested by the Deployment Config.

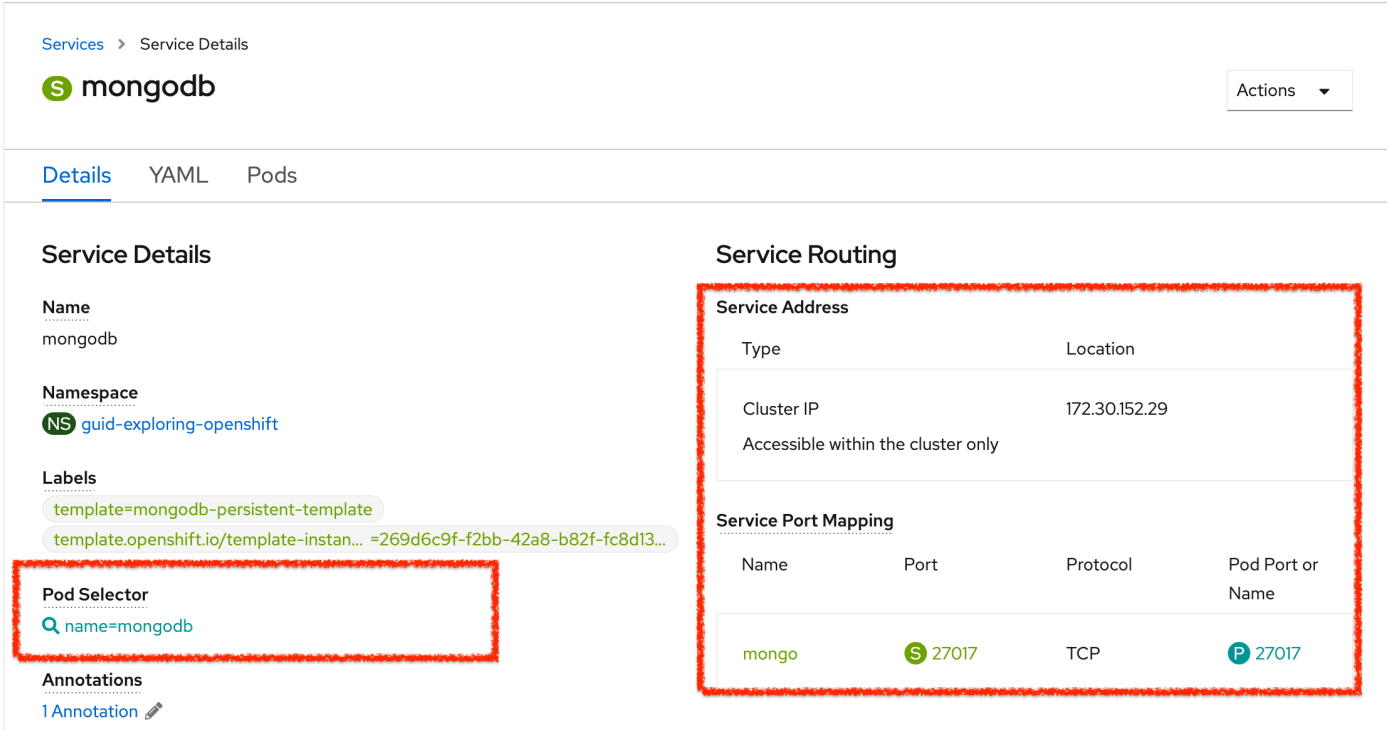
Explain the relationship between ReplicationControllers and ReplicaSets. It is similar to the relationship between OpenShift Deployment Configs and Kubernetes Deployments. All of them are supported by OpenShift.

5.1. Demonstrate Network Resources for Deployments

1. **Action**: From the menu on the left, select **Networking** → **Services**:
   - **Explain**: The project's **Services** page displays all of the services that currently exist in the project.

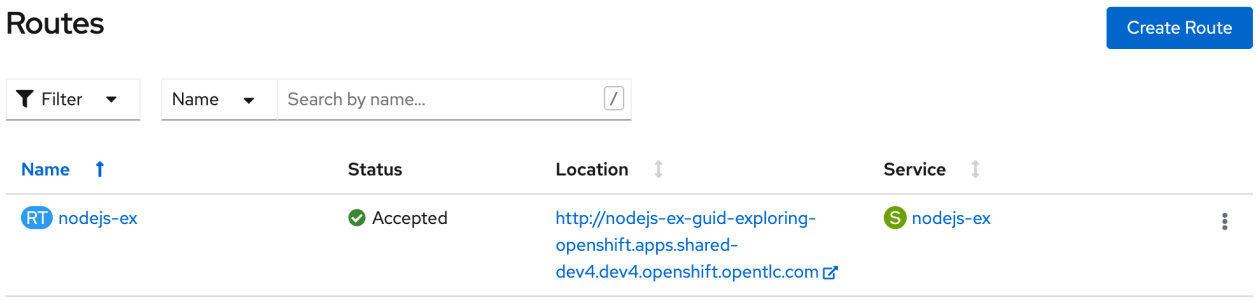
- Point out that **Location** is the permanent **internal** IP address and port that the service uses to represent its Pods.

2. **Action**: Click the **mongodb** service:



Point out that Service Address and Service Port Mapping that the service is listening on is the Location in the prior screen, but reveal the port of the Pod that the port of the services maps. Point out that the **Pod Selector** describes the labels that the Pods must have to be regarded as part of the service.

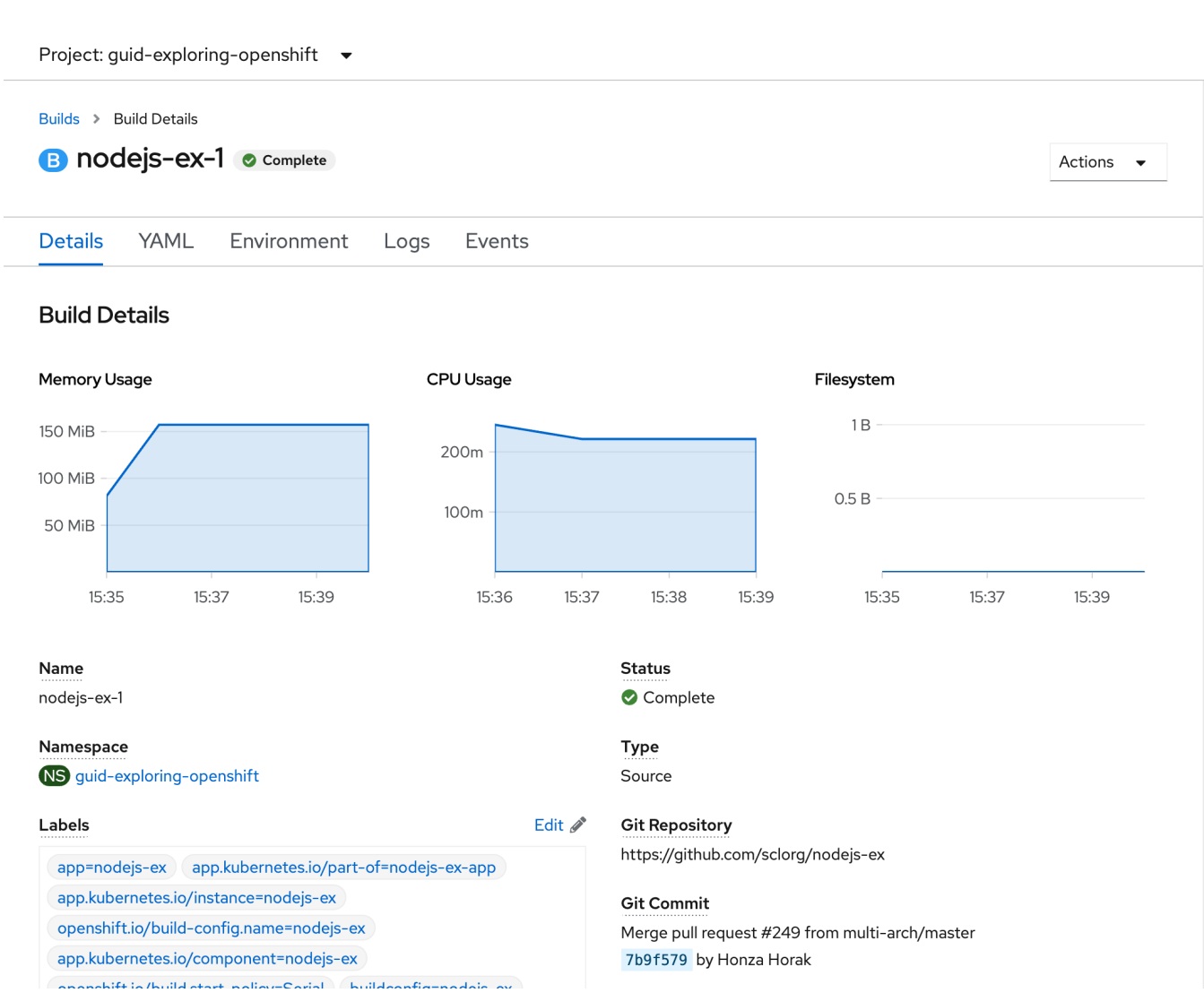
3. **Action**: From the menu on the left, select **Networking** → **Routes**:
   - **Explain**: The project's **Routes** page displays all of the routes that currently exist in the project.
     - Point out the list of routes, their locations, and the services they represent.
     - Point out the row of route information labeled **`nodejs-ex`**.
     - Point out that there is one route, **`nodejs-ex`**, and that it exposes one service, also called **`nodejs-ex`**.
     - Point out the URL that the **Route** exposes.



6. Demonstrate Exploring Build Pages

Explain: In this section, you look at the information available in the UI about your builds.

1. **Action**: From the menu on the left, select **Builds** → **Build Configs**:
    - **Explain**: The **Builds Configs** page displays all of the BuildConfigs for the project.
2. **Action**: Click the `nodejs-ex` BuildConfig from the list and the **Build Config Details** page opens.
    - Point out that you can start a new build by clicking the top right **Actions** menu and selecting **Start Build**.
    - Point out that you can see the **GIT REPOSITORY** where the source code of the build is stored.
3. **Action**: Select the **Builds** tab and the list of builds page opens.
    - **Explain**: The *project's* **Builds** page displays all of the builds for the project.
        - Point out that you can see the status and completion time for each build in the project.
4. **Action**: Select one of the builds in the list:
    - **Explain**: In the *application's* **Build Details** page you can see the following:



- Point out that you can view the configuration used for this build in the **Details** tab.
- Point out that the resource utilization statistics used by the build are displayed in graph form on this page if they are available.
- Point out that you can see the **Owner** of the build. In this case, the owner is the **`nodejs-ex`** BuildConfig, which you were looking at in the previous step.
- Explain that you can rebuild this build by clicking the **Actions** menu in the top right corner and selecting **Rebuild**.
- Point out that you can see the status of the build and the reason it was triggered.
- Point out that you can see the configuration for the build, including the base image that was used and the name of the output image.

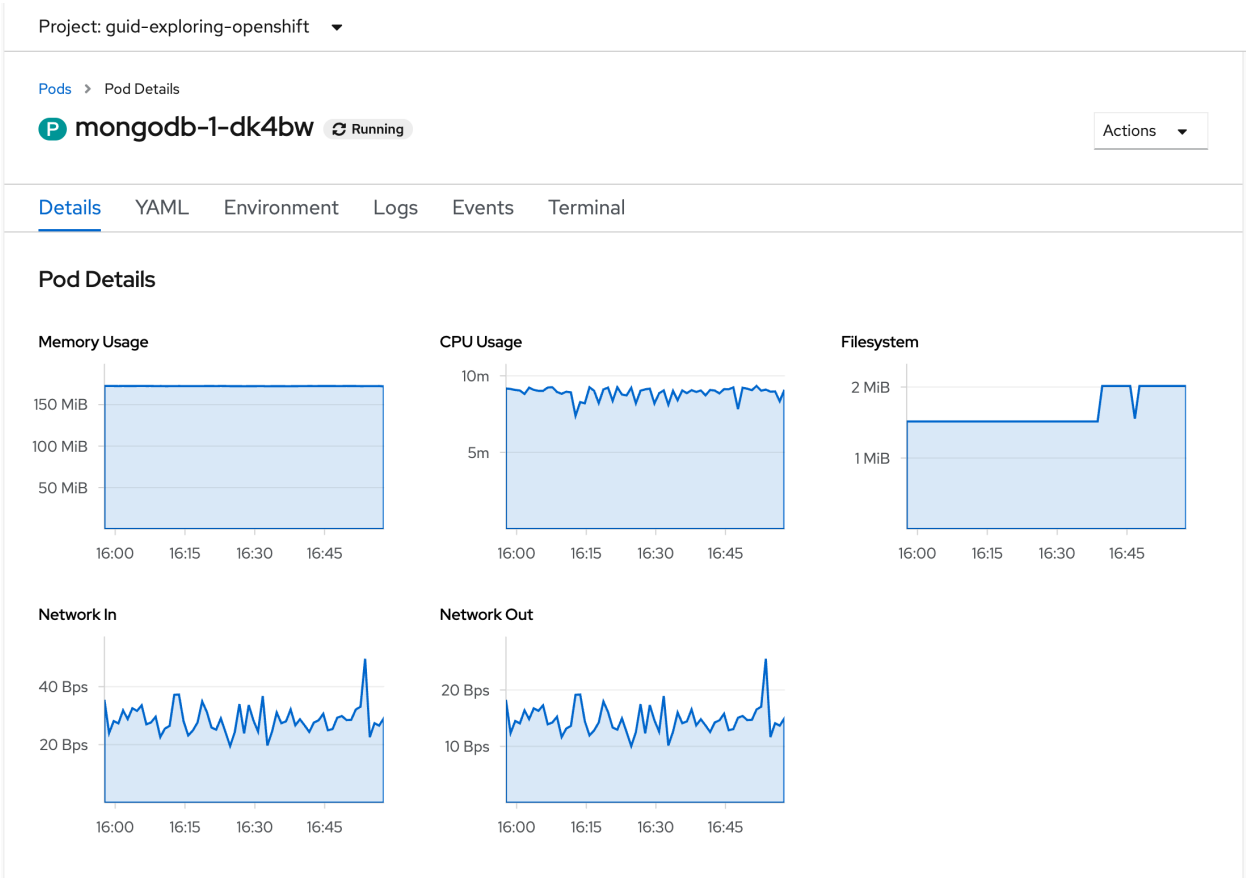
5. **Action**: Select the **Logs** tab.
    - **Explain**: Point out the following:

- The **Logs** tab displays the logs for the build.
- In other examples you may see code dependencies pulled down for the container and other configuration logs.
  - In this example the source code repository is cloned or downloaded to your container.
  - You can see that the completed image is automatically pushed into the integrated registry under your project name.

---

7. Demonstrate Exploring Pods

1. **Action**: From the menu on the left, select **Workloads** → **Pods**:
   - **Explain**: The project's **Pods** page displays all of the Pods that are currently running in the project.
     - Point out that you can see the container status: `Running`, `Pending`, etc.
     - Point out that the **Ready** column shows the true status of the application in the container based on readiness checks.
2. **Action**: When you are done, select your 'mongodb-XXXXX' Pod from the list.
   - **Explain**: The individual **Pod** page displays the following:
     - Point out in the **Pod Overview** section:
       - The charts for **Memory Usage**, **CPU Usage**, and **Filesystem**.



    - The Pod's status and the OpenShift node hosting it.
    - The information displayed in the **Containers** section, including the image name and container state.

3. **Action**: Click the container named `mongodb` and the **Container Details** page opens.
    - Point out the **Resource Requests, Resource Limits** for your Node.js application container.
    - Point out the **Readiness and Liveness Probes** for your PHP application container.
4. **Action**: Click **Back** on your browser to go to the **Pod Details** page.
    - Point out the information displayed in the **Volumes** section, including the name, type, and permissions of the volume.
    - Point out that at the moment there are two volumes mounted:
        - mongodb-data volume bound to PVC **mongodb-data** with Permissions Read/Write
        - default-token **Secret** volume is bound and it is **Read-only**.
5. **Action**: When you are done, select the **Logs** tab.

- **Explain**: The **Logs** tab displays the log for the Pod:
  - Point out the Pod's messages that are displayed here and note that you can follow the log as it updates.
  - Point out that you can pause and restart the log streaming as it updates.

6. **Action**: When you are done, select the **Terminal** tab.
   - **Explain**: The **Terminal** tab allows you to use a terminal inside any of the containers in the Pod.
     - Point out that you can run commands within the container for debugging and testing.
7. **Action**: When you are done, select the **Events** tab.
   - **Explain**: The **Events** tab displays events related to the Pod.
     - Point out that you can use this list to see if something went wrong in the Pod deployment or to trace back a chain of events.

---

8. Clean Up Environment

If you are not going to do any additional demonstrations, go to the home page in the OpenShift Container Platform web console and delete your project.

Build Version: 1.25R : Last updated 2021-10-29 14:30:31 EDT