Work with databases in the OpenShift web console

In this lesson, you will learn how to use a web console and create a database instance in the Developer Sandbox. You will also learn how to create a Node.js instance from source code and bind it with the database running in OpenShift.

What you need to know

In order to get full benefit from this lesson, you need to:

- Understand how Kubernetes is the foundational technology for OpenShift.
- Know how to access the Developer Sandbox via web browser.
- Know how to use the web console to add applications and components to OpenShift.
- Understand how environment variables provide information to an application at runtime.

What you will learn

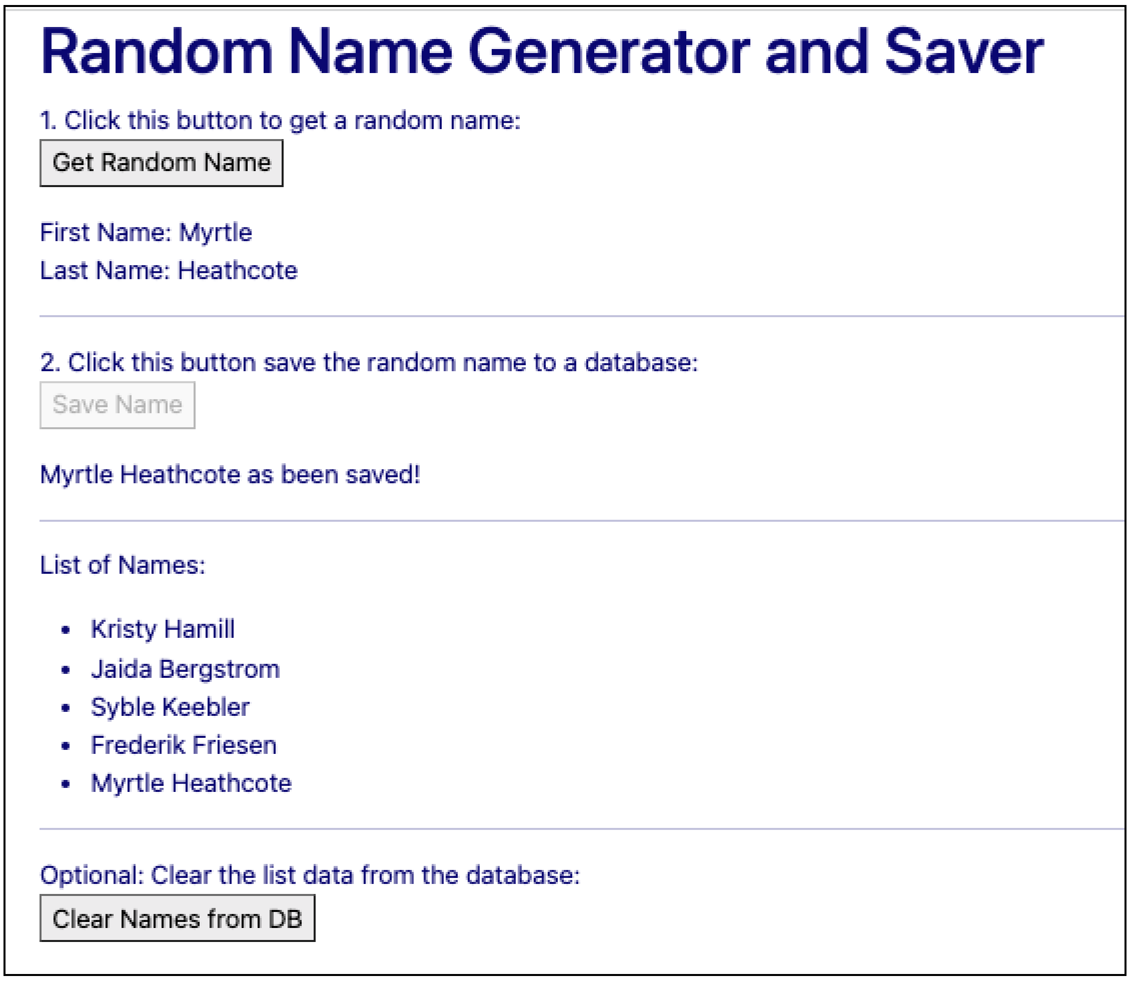
In this lesson, you will:

- Install a MongoDB ephemeral database template at the command line, making it available to the web console.
- Create an instance of a database in Developer Sandbox using the web console.
- Create an instance of a Node.js application from source code stored in GitHub using the web console.
- Bind the Node.js application to the database running in OpenShift.

Understand the demonstration scenario

The objective of this lesson is to demonstrate installing and running a Node.js application that is bound to a database that you will also install. Both the Node.js application and the database will run in the Developer Sandbox.

The Node.js application publishes a web page that allows users to choose a name at random then store that random name in a database to which the application is bound. Also, the application web page displays a list of names stored in the database (Figure 1).


Random Name Generator and Saver

1. Click this button to get a random name:

Get Random Name

First Name: Myrtle
Last Name: Heathcote

2. Click this button save the random name to a database:

Save Name

Myrtle Heathcote as been saved!

List of Names:

- Kristy Hamill
- Jaida Bergstrom
- Syble Keebler
- Frederik Friesen
- Myrtle Heathcote

Optional: Clear the list data from the database:

Clear Names from DB

Figure 1: The Node.js demonstration application allows users to get, store, and list random names.

The demonstration scenario uses a MongoDB database which you'll install using the web console. Also, you'll install the Node.js application in OpenShift using the install from source code feature available in the web console.

The Node.js application has a server-side API that stores and retrieves data stored in the MongoDB database. Also, the Node.js application publishes a web page that is bound to the server-side API. Users interact with the web page and the web page, in turn, interacts with the server-side API (Figure 2).

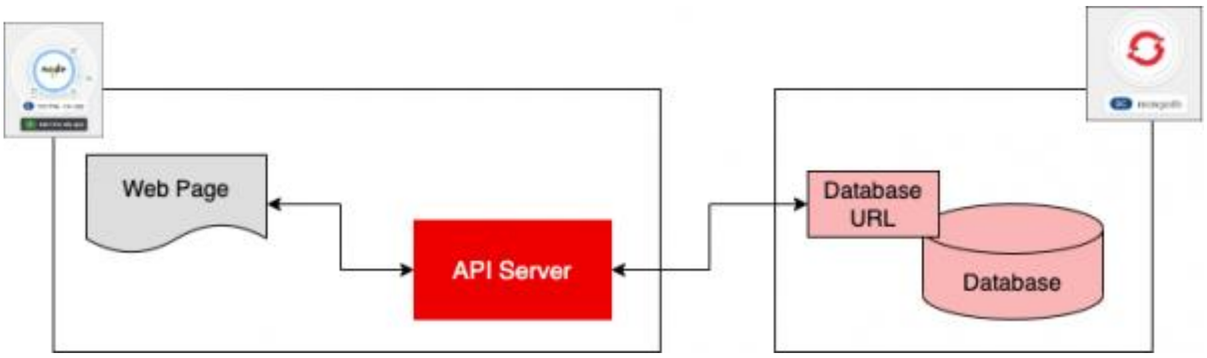


Figure 2: The demonstration project binds a Node.js application running in the Developer Sandbox to an instance of a database that's also running OpenShift.

The way that Node.js binds to the MongoDB database is by way of a connection URL that's created by OpenShift during the database installation process. The URL will contain the username and password information. You'll declare the username and password information when you install the MongoDB database.

The Node.js application references the MongoDB connection URL to create a connection to the database. The way that the Node.js application knows the connection URL is by way of an environment variable that's created as part of the process of installing the Node.js application in the Developer Sandbox.

The sections that follow demonstrate how to install the MongoDB database and the Node.js application along with the steps necessary to expose and consume the MongoDB connection URL.

Install the MongoDB ephemeral database template at the command line

Log in to your cluster at the command line. If you're unsure how to do this, you can find instructions here: Access your Developer Sandbox from the command line.

Run the following two commands to create the MongoDB ephemeral database template:

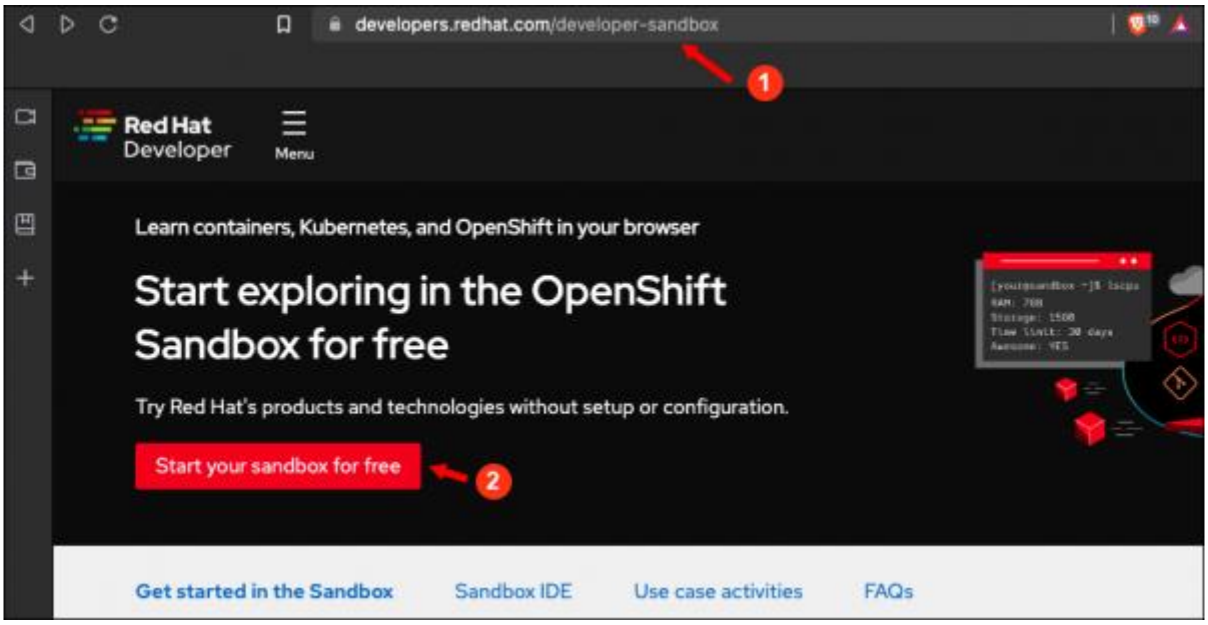
```
oc create -f https://raw.githubusercontent.com/openshift-labs/starter-guides/ocp-4.8/mongodb-template.yaml
```

```
oc import-image mongodb:3.6 --from=registry.access.redhat.com/rhscl/mongodb-36-rhel7 --confirm
```

Access the OpenShift web console using the Developer Sandbox

Take the following steps to access the OpenShift web console in the Developer Sandbox. You'll do the work of installing and configuring the MongoDB and Node.js instances using the web console.

1. Using your web browser, navigate here and select **Start your sandbox for free** (Figure 3).


developers.redhat.com/developer-sandbox
1
Red Hat Developer
Menu
2
Get started in the Sandbox
Sandbox IDE
Use case activities
FAQs

Learn containers, Kubernetes, and OpenShift in your browser

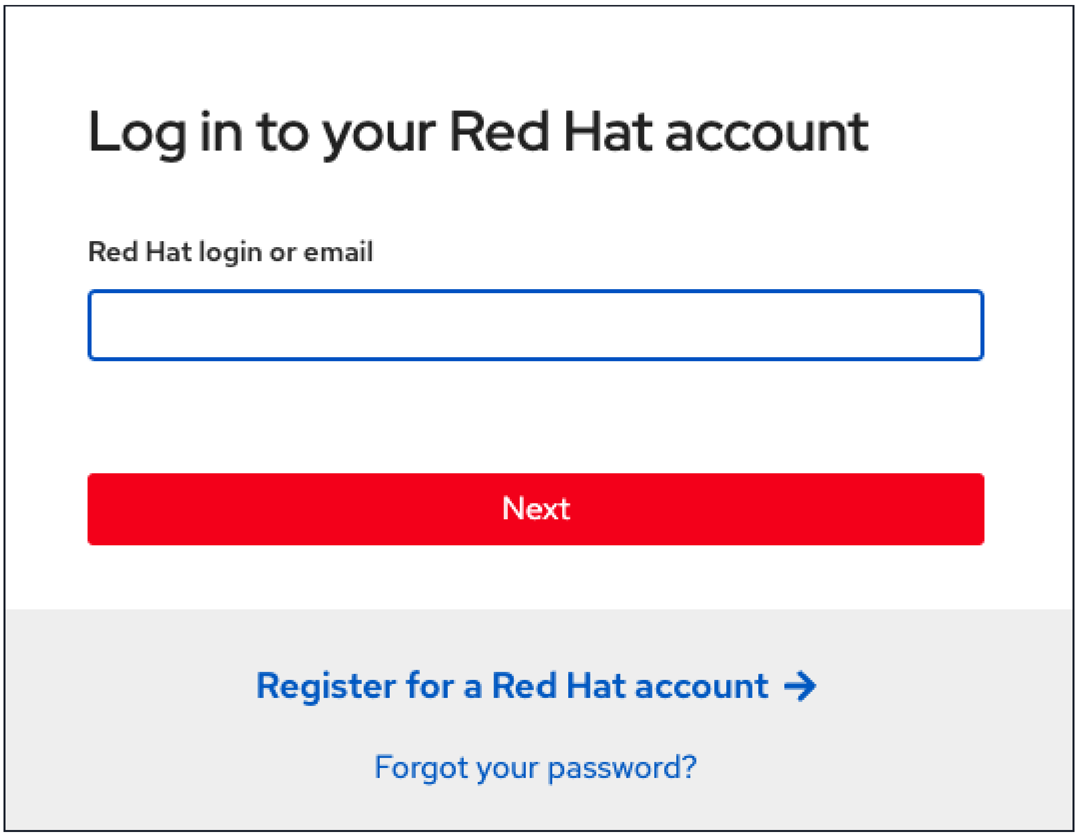
Start exploring in the OpenShift Sandbox for free

Try Red Hat's products and technologies without setup or configuration.

Start your sandbox for free

Figure 3: The entry point for access to the OpenShift Developer Sandbox.

2. Provide the username and password for your Red Hat account (Figure 4).



Log in to your Red Hat account

Red Hat login or email

Next

Register for a Red Hat account →

Forgot your password?

Figure 4: Log into the OpenShift web console with the username and password associate with your account on Red Hat.

Install a database using the OpenShift web console

This demonstration scenario requires that you install a MongoDB database in the Developer Sandbox. You will install the database by using a template listed in the web console's Developer Catalog:

1. Click **+Add** on the left side of the web console (Figure 5, Callout 1).
2. The Developer Catalog page will appear. Click **Database** (Figure 5, Callout 2).
3. The web page with database text blocks will appear. Click **MongoDB** (Figure 5, Callout 3).

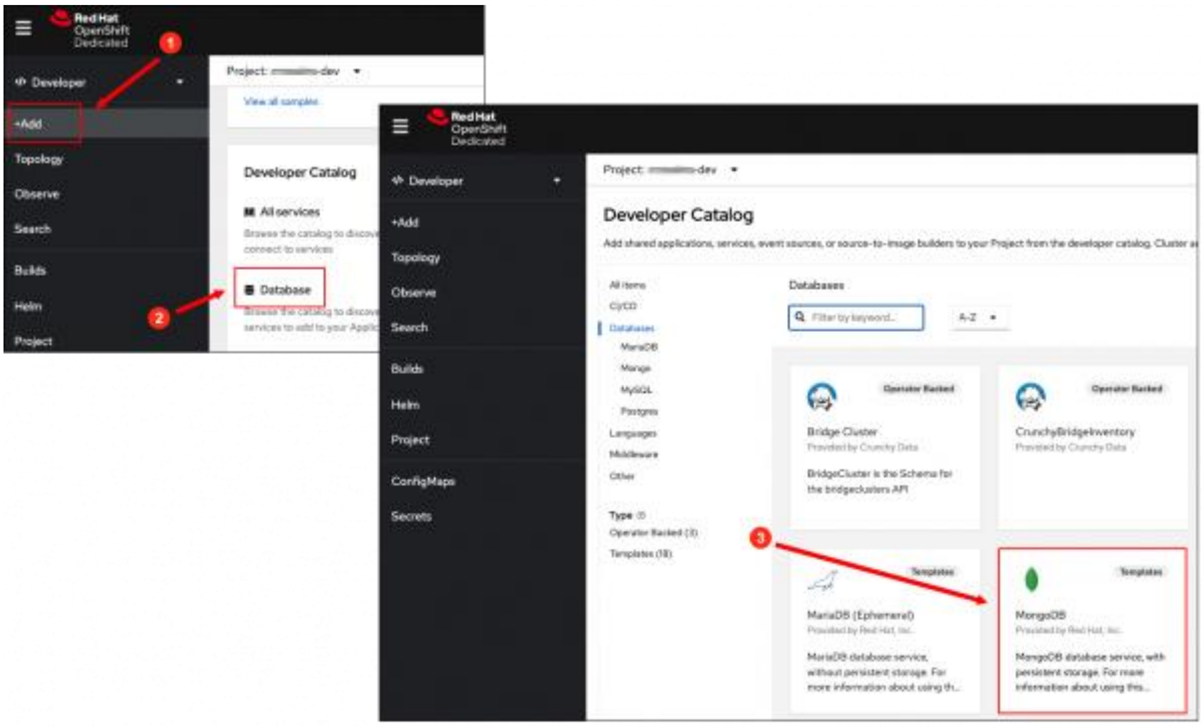
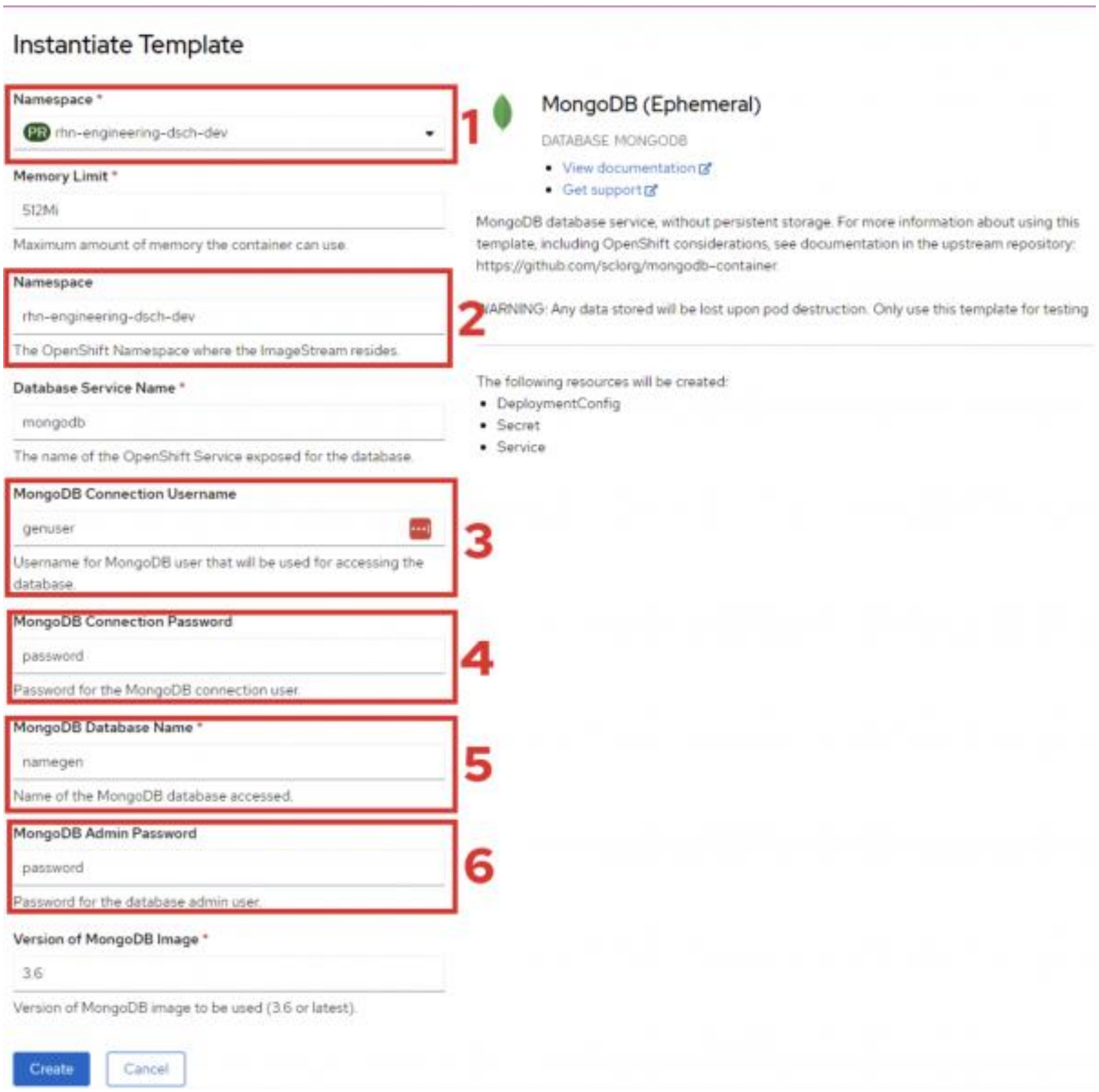


Figure 5: The web console’s Developer Catalog lists database templates available for installation.

4. The MongoDB template page will appear. Click **Instantiate Template** to expose the web page where you will enter configuration information for the MongoDB database.
5. Make note of (and copy) the Namespace in which the DB instance will be created (Figure 6, Callout 1). Enter this value as the Namespace where the template will look for the image (Figure 6, Callout 2).
6. Enter genuser in the **MongoDB Connection Username** text box (Figure 6, Callout 3).
7. Enter password in the **MongoDB Connection Password** text box (Figure 6, Callout 4).
8. Enter namegen in the **MongoDB Database Name** text box (Figure 6, Callout 5).
9. Enter password in the **MongoDB Admin Password** text box (Figure 6, Callout 6).
10. Click the **Create** button to begin the deployment.


Instantiate Template

Namespace *

PR rhn-engineering-dsch-dev

1

Memory Limit *

512Mi

Maximum amount of memory the container can use.

Namespace

rhn-engineering-dsch-dev

The OpenShift Namespace where the ImageStream resides.

2

Database Service Name *

mongodb

The name of the OpenShift Service exposed for the database.

MongoDB Connection Username

genuser

Username for MongoDB user that will be used for accessing the database.

3

MongoDB Connection Password

password

Password for the MongoDB connection user.

4

MongoDB Database Name *

namegen

Name of the MongoDB database accessed.

5

MongoDB Admin Password

password

Password for the database admin user.

6

Version of MongoDB Image *

3.6

Version of MongoDB image to be used (3.6 or latest).

MongoDB (Ephemeral)

DATABASE MONGODB

- View documentation
- Get support

MongoDB database service, without persistent storage. For more information about using this template, including OpenShift considerations, see documentation in the upstream repository: https://github.com/sclorg/mongodb-container.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing

The following resources will be created:

- DeploymentConfig
- Secret
- Service

Figure 6: Enter configuration information for the MongoDB instance in the the Instantiate Template web page.

Executing the steps shown above will create an instance of MongoDB in the Developer Sandbox. OpenShift will automatically implement the following connection URL:

mongodb://genuser:password@mongodb

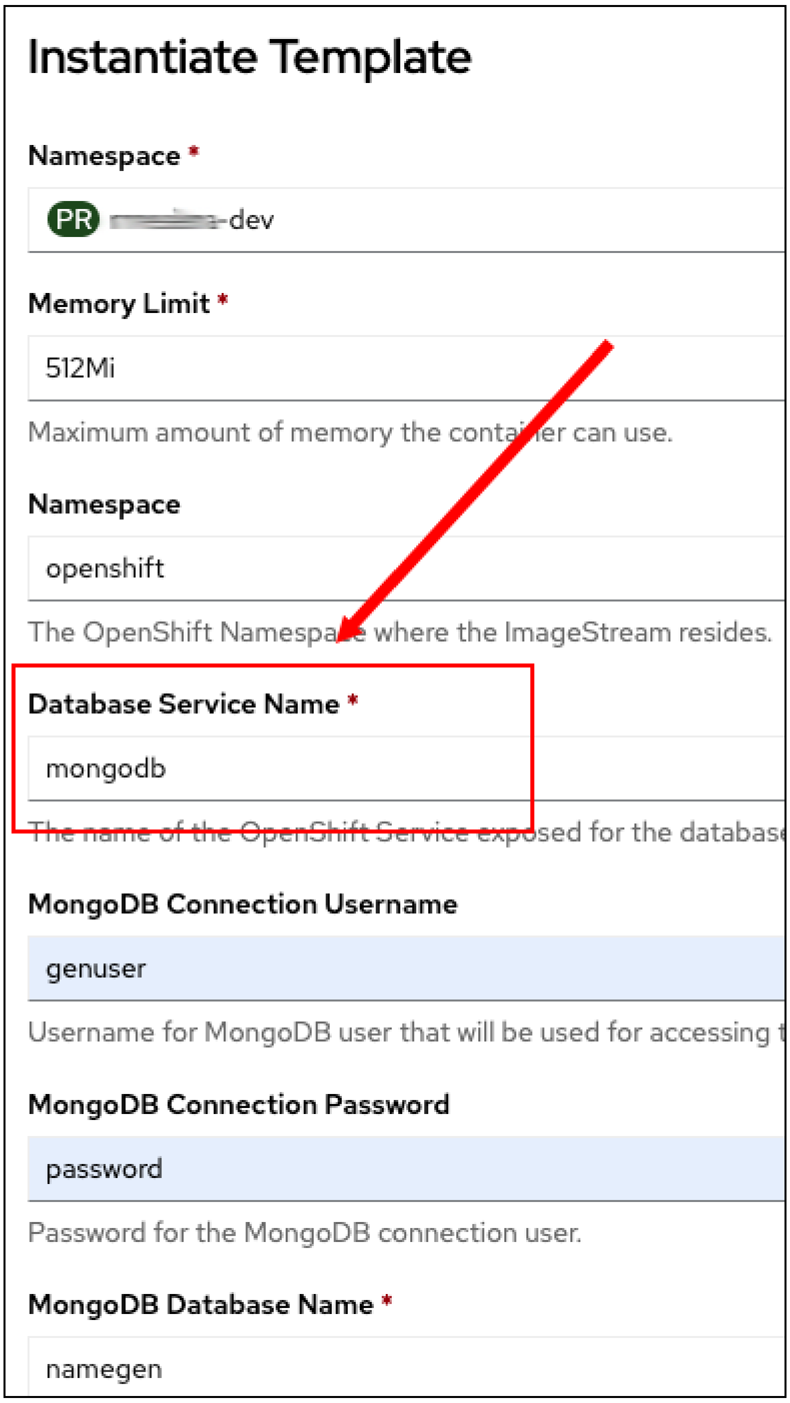
The connection URL is constructed according to the following naming pattern:

mongodb://<username>:<connection_password>@<service_name>

About the parameters:

- username is the username you defined earlier when you configured the MongoDB template.

- connection password is the password you defined earlier when you configured the MongoDB template.
- service name is the underlying Kubernetes service that is created as part of the database installation process. In this case, the service name is "**mongodb**", which was provided by default in the MongoDB Template configuration page (Figure 7).



Instantiate Template

Namespace *

PR [redacted]-dev

Memory Limit *

512Mi

Maximum amount of memory the container can use.

Namespace

openshift

The OpenShift Namespace where the ImageStream resides.

Database Service Name *

mongodb

The name of the OpenShift Service exposed for the databas

MongoDB Connection Username

genuser

Username for MongoDB user that will be used for accessing t

MongoDB Connection Password

password

Password for the MongoDB connection user.

MongoDB Database Name *

namegen

Figure 7: Database Service Name represents the MongoDB database in the OpenShift cluster.

After you create the MongoDB instance, it will appear in the **Topology** view (Figure 8).

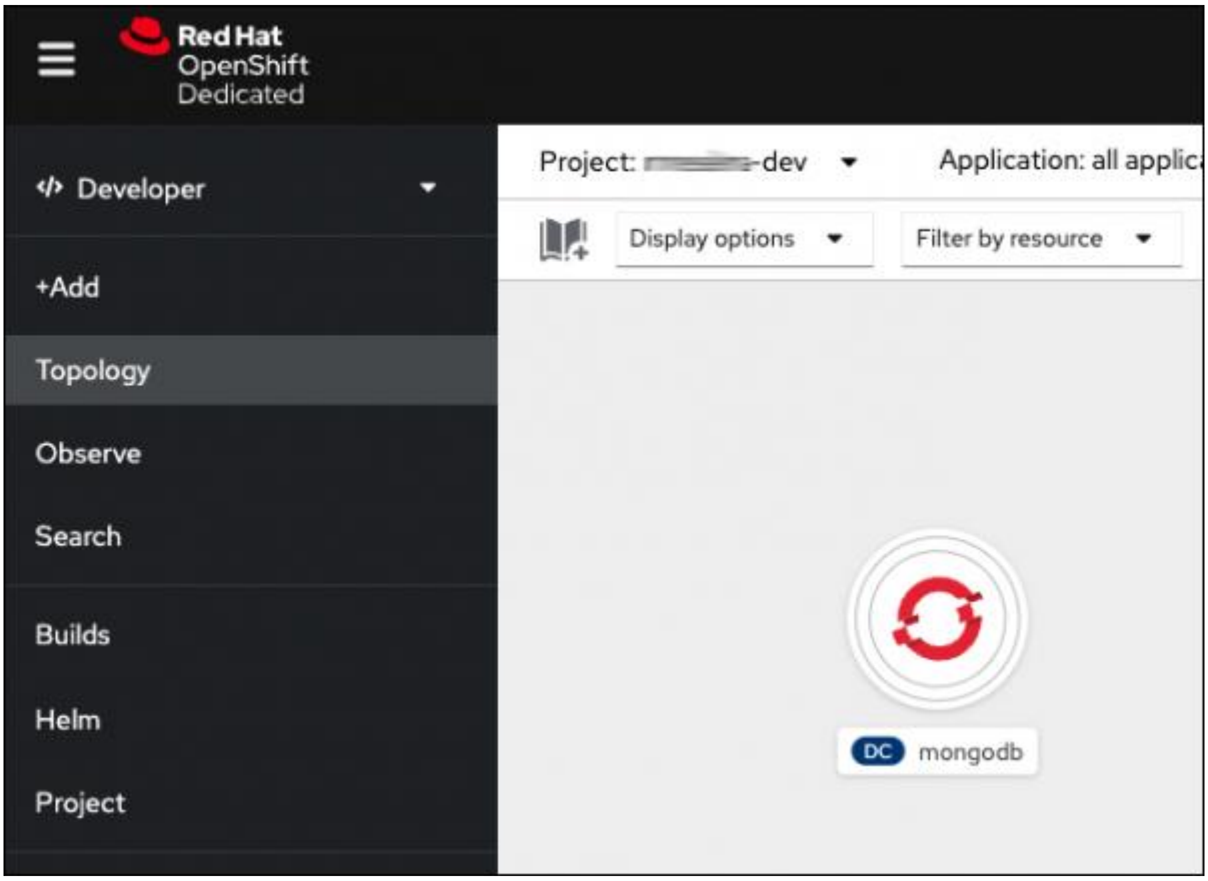


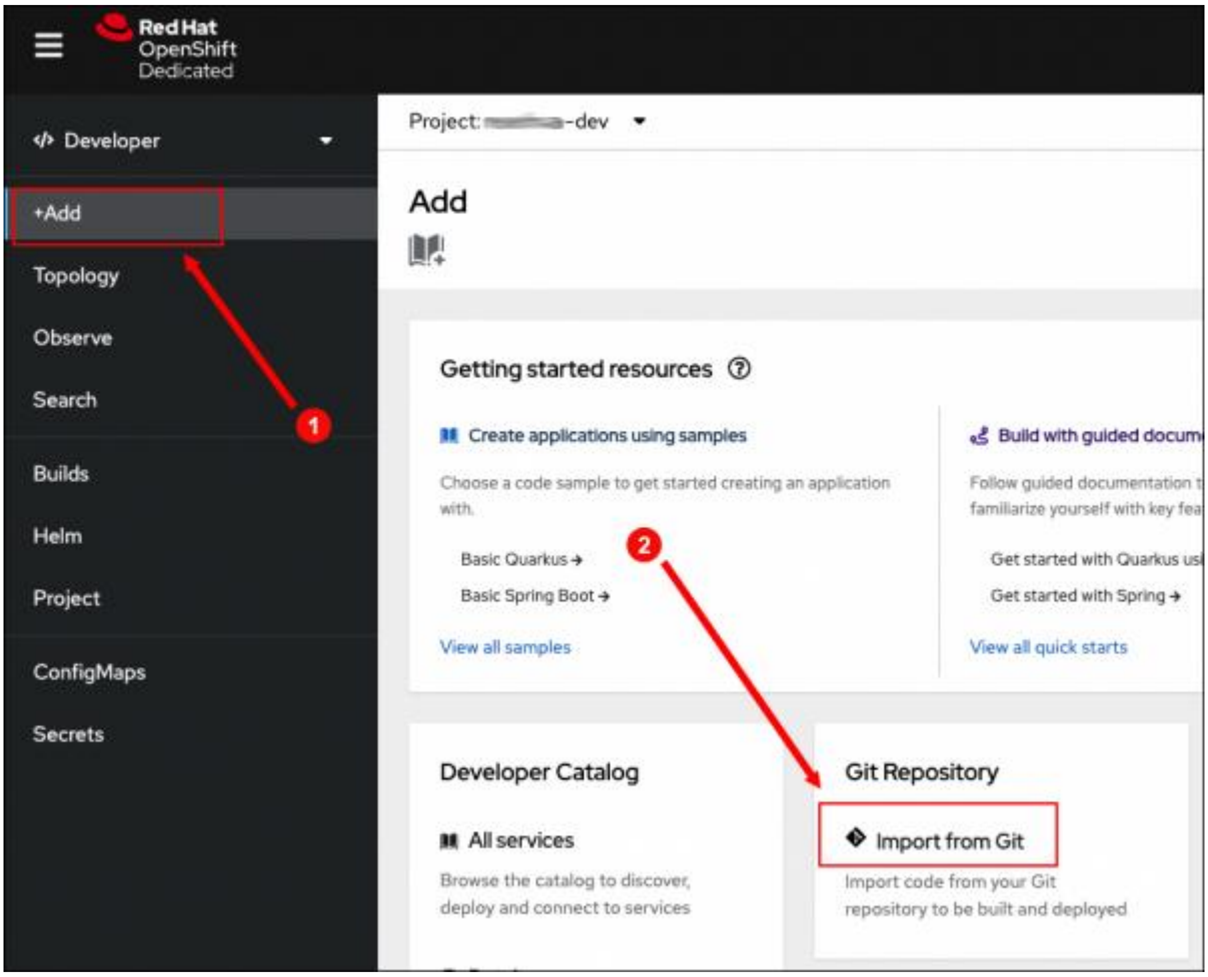
Figure 8: The MongoDB database will appear in the Topology view after it's installed.

Once the MongoDB database is installed and the connection URL is known, you're ready to install the Node.js application.

Install a Node.js application from GitHub using the web console

The following describes the steps required to install the Node.js application into the Developer Console. You will install the application using the **OpenShift Install from Git** feature.

1. Select **+Add** on the left side of the web console (Figure 9, Callout 1).
2. The Developer Catalog page will appear. Select the **Import from Git** text block (Figure 9, Callout 2).



Choose a code sample to get started creating an application with.

Follow guided documentation t familiarize yourself with key fea

Browse the catalog to discover, deploy and connect to services

Import code from your Git repository to be built and deployed

Figure 9: You can install an application in the Developer Sandbox from source code stored in a Git repository.

3. An Import from Git web page will appear when you click the text block (Figure 10, Callout 1). Enter the following URL into the Git Repository URL text box:

https://github.com/redhat-developer-demos/namegen

4. Scroll down the page until you come to the **Application name** text box (Figure 10, Callout 2) and enter the following text:

random-name-gen

5. Enter the following text (Figure 10, Callout 3) in the **Name** text box:

random-name-gen

6. Copy snippet.
7. Scroll further down the Import from Git web page to the section above the **Create** button. You will see an option called **Build configuration**. Select this option.
8. The web page will expand to display a section titled **Build configuration** (Figure 10, Callout 4). Enter the following text in the Name text box of the section labeled **Environmental variable** (build and runtime):

MONGODB_URL

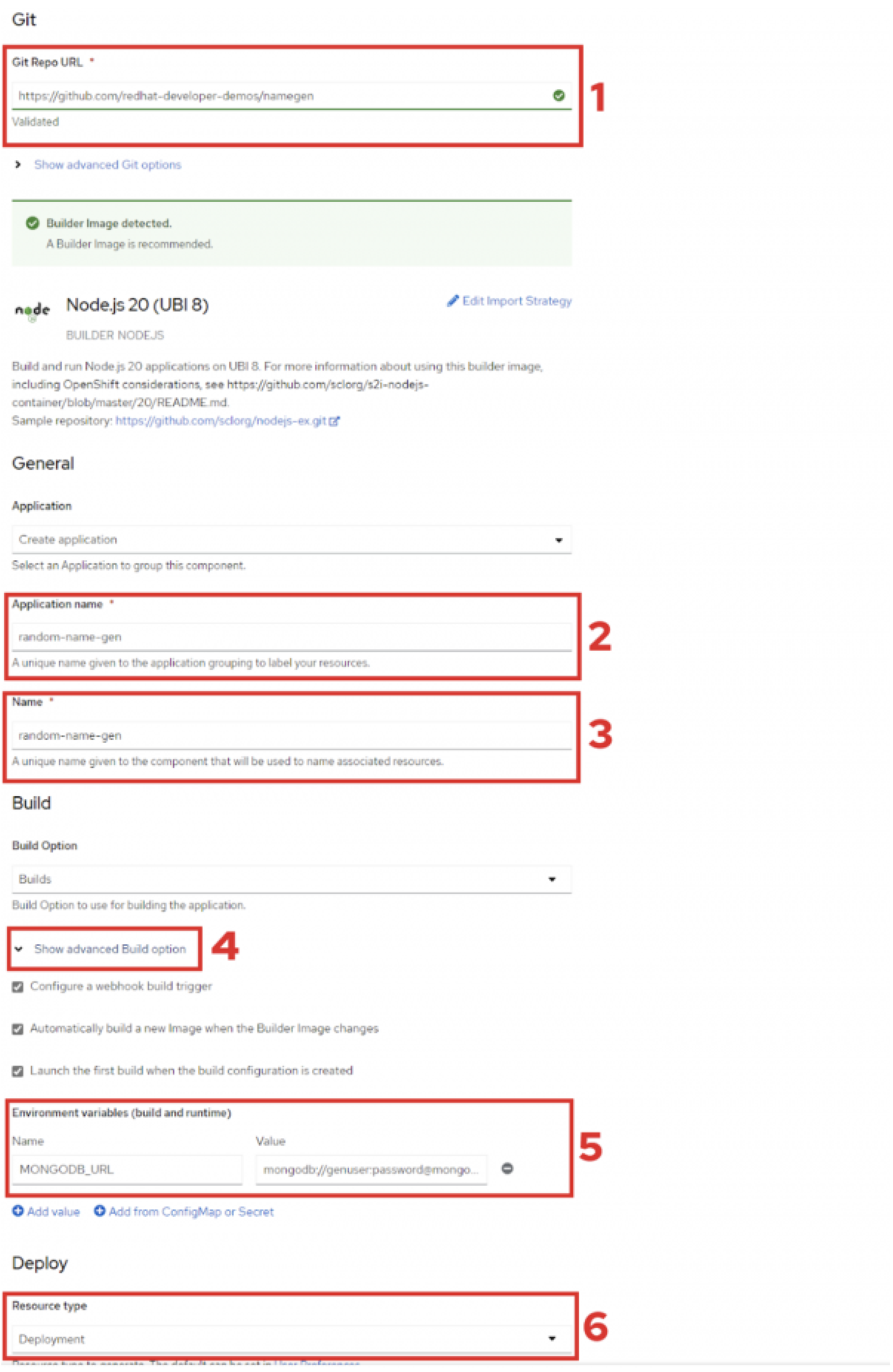
9. Enter the following text (Figure 10, Callout 5) in the **Value** field:

mongodb://genuser:password@mongodb

10. Notice that the value of the environment variable is the connection URL to the MongoDB instance you installed previously. The environment variable is the mechanism that provides the Node.js application with the information it needs to connect to the MongoDB database.

11. Select Deployment as the Resource type (Figure 10, Callout 6).

12. Select **Create**. This will start the process of installing the Node.js application from source code.



Git

Git Repo URL *

https://github.com/redhat-developer-demos/namegen

Validated

1

Show advanced Git options

Builder Image detected.

A Builder Image is recommended.

Node.js 20 (UBI 8)

Edit Import Strategy

BUILDER NODEJS

Build and run Node.js 20 applications on UBI 8. For more information about using this builder image, including OpenShift considerations, see https://github.com/sclorg/s2i-nodejs-container/blob/master/20/README.md.

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

General

Application

Create application

Select an Application to group this component.

Application name *

random-name-gen

A unique name given to the application grouping to label your resources.

2

Name *

random-name-gen

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

3

Build

Build Option

Builds

Build Option to use for building the application.

Show advanced Build option

4

- [x] Configure a webhook build trigger
- [x] Automatically build a new Image when the Builder Image changes
- [x] Launch the first build when the build configuration is created

Environment variables (build and runtime)

Name	Value
MONGODB_URL	mongodb://genuser:password@mongo...

5

Add value Add from ConfigMap or Secret

Deploy

Resource type

Deployment

6

Resource type to generate. The default can be set in User Preferences.

Figure 10: Carefully specifying the appropriate values gives you the unique application you desire.

13. Behind the scenes, OpenShift will automatically do the work of converting the source code into a Linux container that gets added to a Kubernetes pod that OpenShift also created. That pod represents the application in the Developer Sandbox. Also, OpenShift will create the Kubernetes service and route that will make the application accessible on the Internet.

14. The Node.js application will appear in the **Topology** view as a graphic with a light blue circle. The light blue circle means that the application is building (Figure 11).

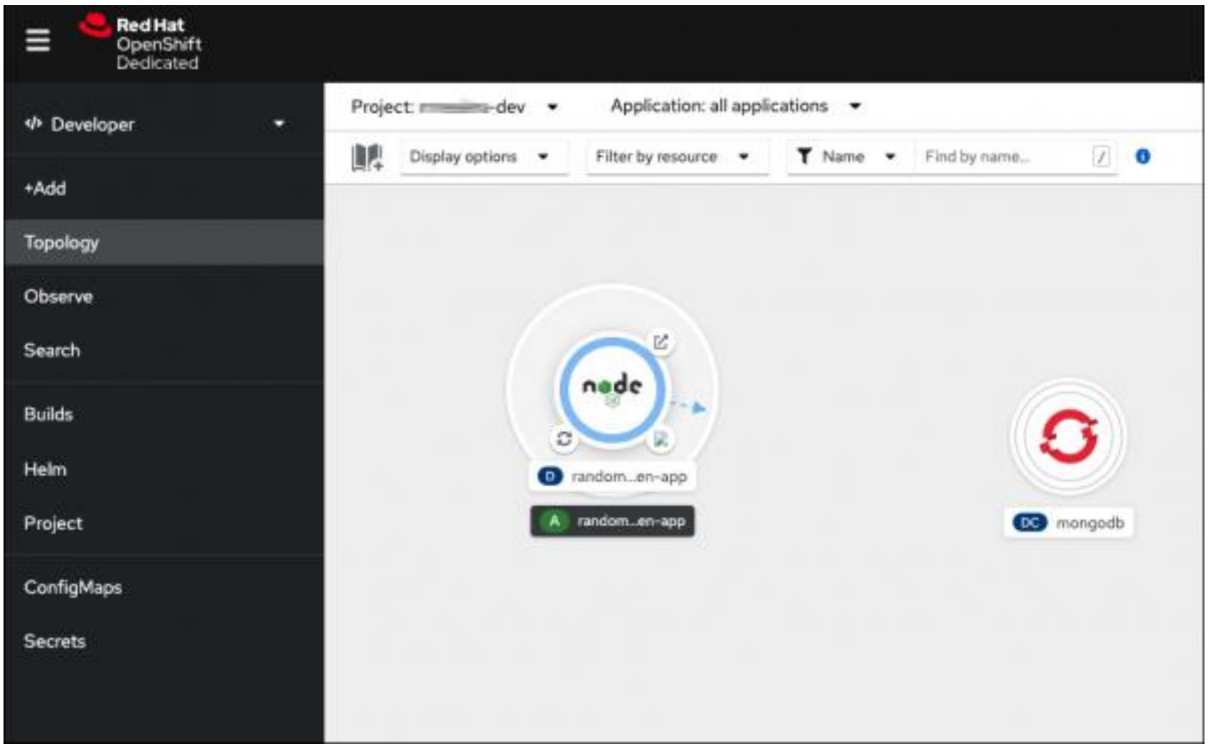


Figure 11: An application that is building will display a light blue circle in its graphic in Topology view.

15. When the application's build process is complete, the circle in the Node.js application graphic in **Topology** view will turn dark blue (Figure 12).

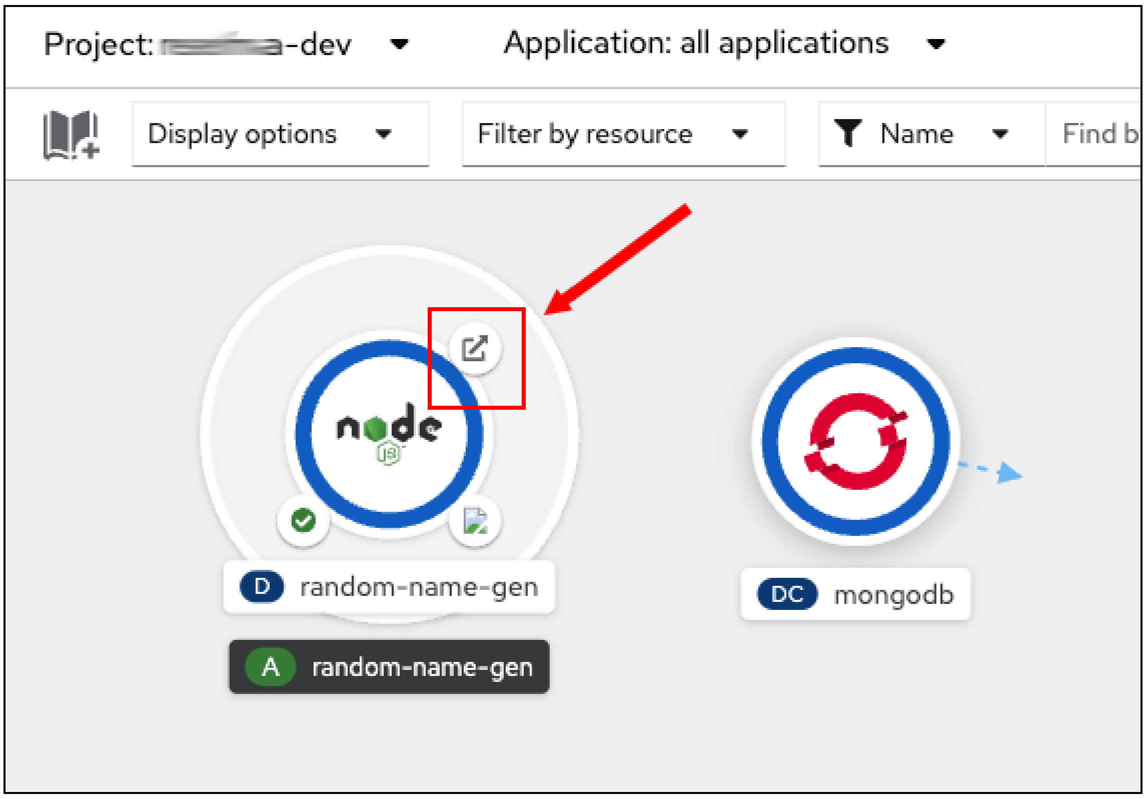
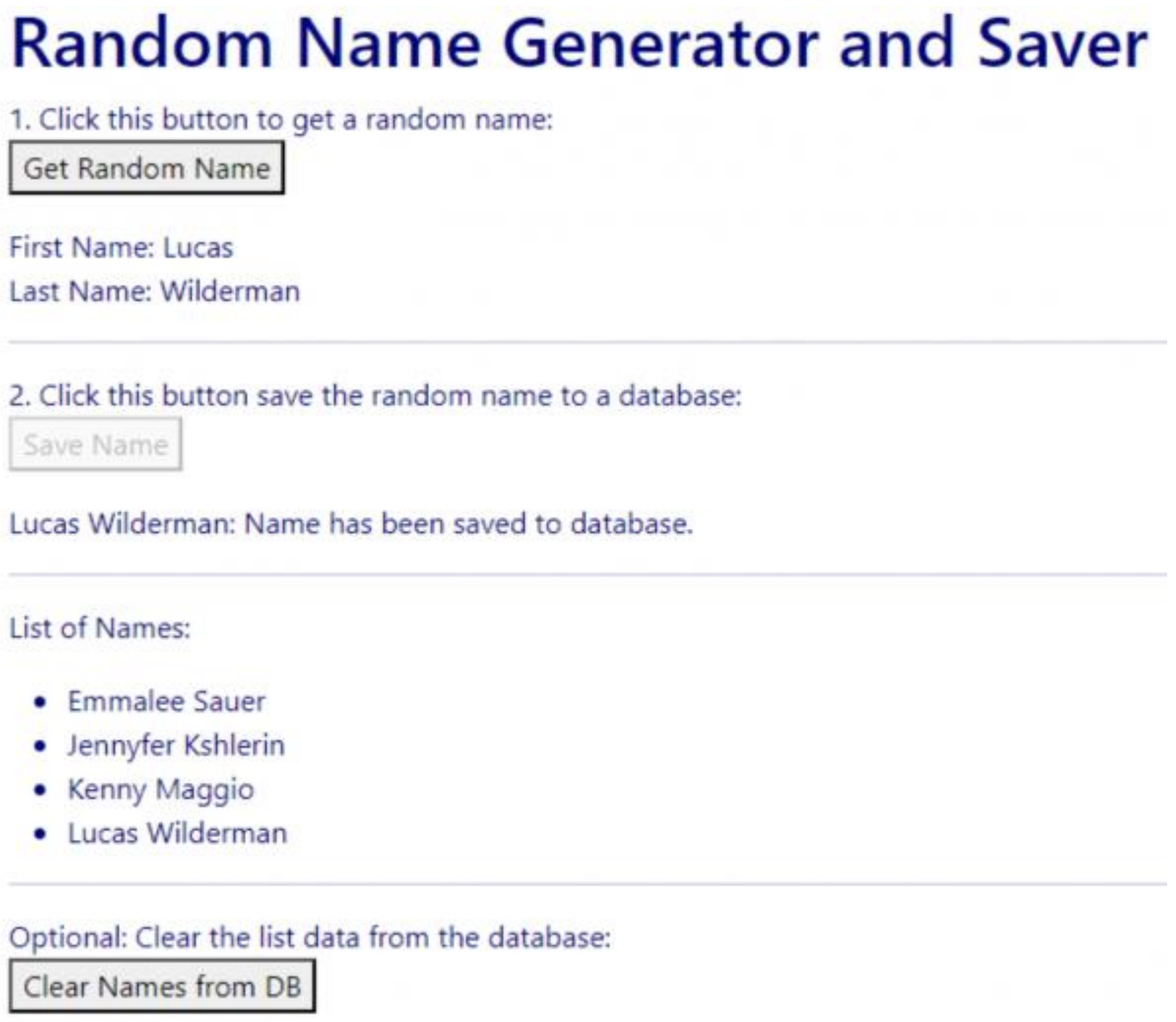


Figure 12: A built application will display a dark blue circle which means the application is up and running and accessible on the Internet by clicking the icon on the upper right of the graphic.

View the web application in a browser

Once the circle in the **Topology** view Node.js graphic turns dark blue, click the **icon** on the upper right corner of the Node.js graphic. Clicking the icon will open up a window in your browser that displays the Node.js application's web page (Figure 13).


Random Name Generator and Saver

1. Click this button to get a random name:

First Name: Lucas

Last Name: Wilderman

2. Click this button save the random name to a database:

Lucas Wilderman: Name has been saved to database.

List of Names:

- Emmalee Sauer
- Jennyfer Kshlerin
- Kenny Maggio
- Lucas Wilderman

Optional: Clear the list data from the database:

Figure 13: The Node.js demonstration application generates a random name on demand and then adds the name to a list of names stored in the MongoDB database.

The application is now up and running. You can exercise the application by selecting **Get Random Name**. An automatically generated random name will appear.

Also, the button labeled **Save Name** will be enabled. Click the **Save Name** button to add the newly created random name to List of Names.

Summary

In this lesson, you learned how to install an instance of MongoDB template in the OpenShift Catalog using the web console. You learned how to declare a database user, password, and default database during the installation process.

In addition, you learned how to use the Developer Sandbox web console to install an instance of a custom application from source code in GitHub. Also, you learned how to configure the application to bind to the MongoDB database by assigning a connection URL for the MongoDB instance to an environment variable that you declared during the installation process of the custom application.

Finally, after installing the MongoDB instance and custom application in the Developer Sandbox, you exercised the custom application from a browser window via its web interface.