Red Hat Fuse 7.1

Managing Fuse

Managing Fuse applications with Hawtio and Prometheus

Last Updated: 2020-08-07

Red Hat Fuse 7.1 Managing Fuse

Managing Fuse applications with Hawtio and Prometheus

Legal Notice

Abstract

When you deploy a Fuse application, you can use both the Fuse Console (Hawtio) and Prometheus to monitor and interact with Red Hat Fuse integrations.

Table of Contents

PREFACE

Red Hat Fuse provides two enterprise monitoring tools for viewing and managing Fuse integrations:

- The Fuse Console is a web-based console that you access from a browser to monitor and manage a running Fuse container. The Fuse Console is based on Hawtio open source software (http://hawt.io/).
- Prometheus (https://prometheus.io/docs/introduction/overview/) stores system and integration-level metrics for Fuse distributions. You can use a graphical analytics interface, such as Grafana, to view and analyze the stored historical data.

The audience for this guide is Red Hat Fuse administrators. This guide assumes that you are familiar with the Red Hat Fuse platform, Apache Camel, and the processing requirements for your organization.

When you deploy a Fuse application, you can use both the Fuse Console (Hawtio) and Prometheus to monitor and interact with Red Hat Fuse integrations.

Fuse Console

The Fuse Console provides a central interface to examine and manage the details of one or more deployed Fuse containers. You can also monitor Red Hat Fuse and system resources, perform updates, and start or stop services.

The Fuse Console is available when you install Red Hat Fuse standalone or use Fuse on OpenShift. The integrations that you can view and manage in the Fuse Console depend on the plugins that are running. Possible plugins include:

- Camel
- JMX
- Spring Boot
- OSGI
- Runtime
- Logs

The Fuse Console is a web-based console. For a list of supported browsers, go to https://access.redhat.com/articles/310603.

Prometheus

Prometheus is container-native software built for storing historical data and for monitoring large, scalable systems of which Fuse on OpenShift is a component. It gathers data over an extended time, rather than just for the currently running session.

PART I. ACCESSING THE FUSE CONSOLE

How you access the Fuse Console depends on your Fuse distribution:

- Fuse on OpenShift
- Fuse Standalone:
  - Spring Boot
  - Red Hat JBoss EAP
  - Apache Karaf

CHAPTER 1. ACCESSING THE FUSE CONSOLE ON OPENSHIFT

You can deploy the Fuse Console either from the OpenShift Console or from the command line.

NOTE

Security and user management for the Fuse Console is handled by OpenShift.

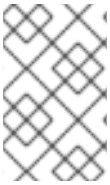
The Fuse Console templates configure end-to-end encryption by default so that your Fuse Console requests are secured end-to-end, from the browser to the in-cluster services.

Role-based access control (for users accessing the Fuse Console after it is deployed) is not yet available for Fuse on OpenShift.

1.1. BEFORE YOU BEGIN (CLUSTER-MODE SETUP)

If you want to deploy the Fuse Console in cluster mode on the OpenShift Container Platform environment, you need the cluster admin role and the cluster mode template. Run the following command:

```
oc adm policy add-cluster-role-to-user cluster-admin system:serviceaccount:openshift-infra:template-instance-controller
```



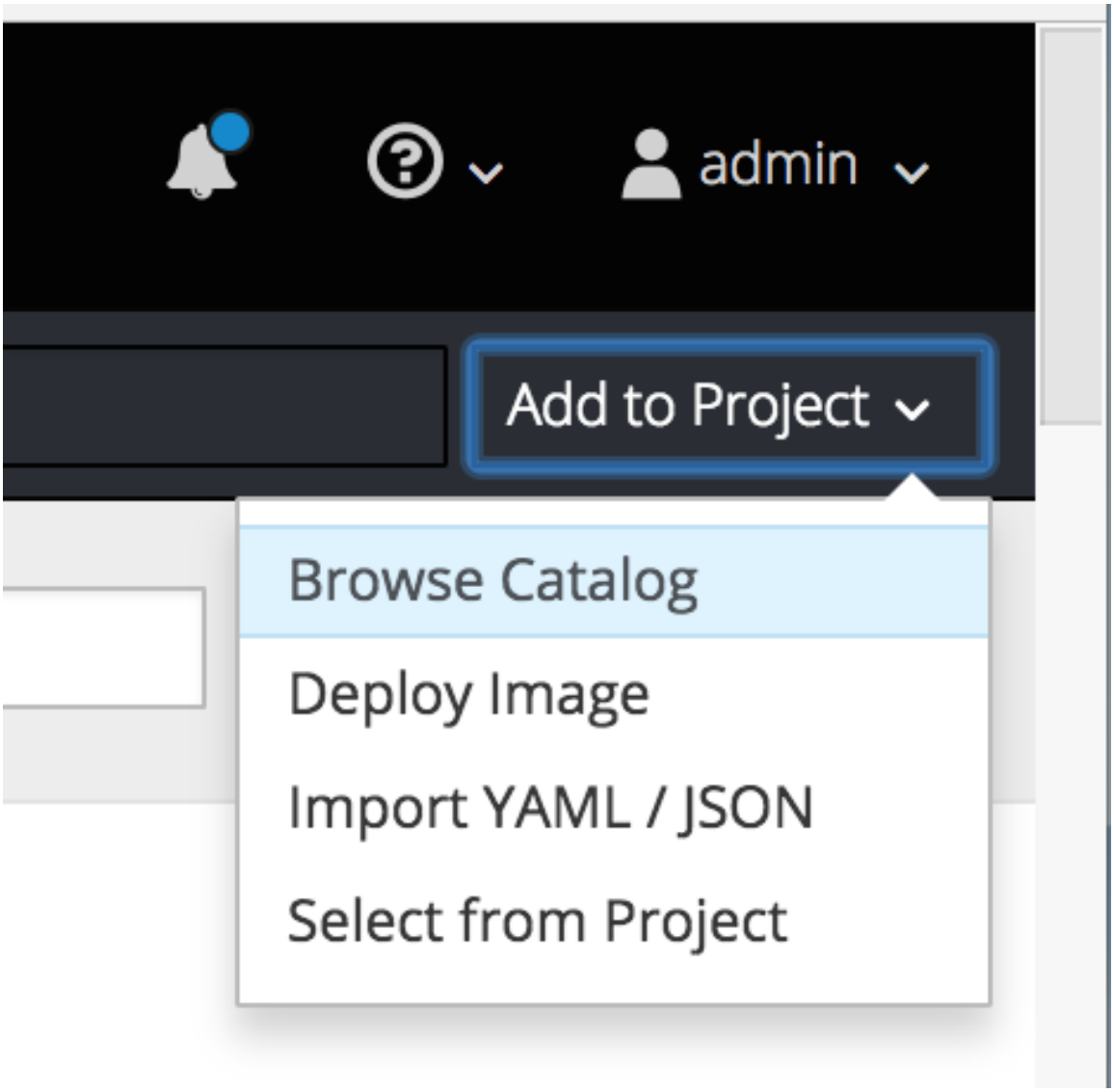
NOTE

The cluster mode template is only available, by default, on the latest version of the OpenShift Container Platform. It is not provided with the OpenShift Online default catalog.

1.2. DEPLOYING THE FUSE CONSOLE FROM THE OPENSHIFT CONSOLE

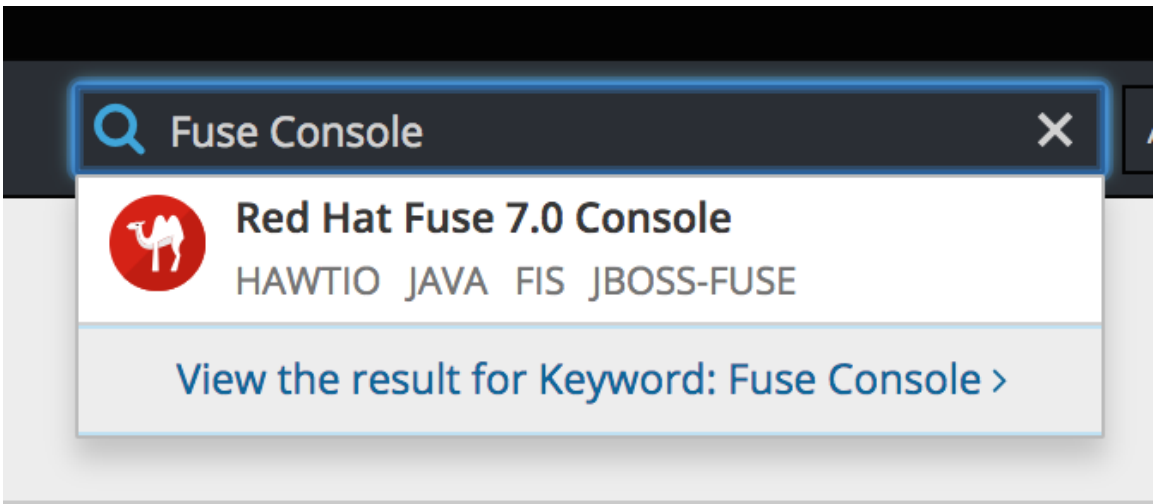
To deploy the Fuse Console on your OpenShift cluster, follow these steps:

1. In the OpenShift console, open an existing project or create a new project.
2. Add the Fuse Console to your OpenShift project:
    a. Select **Add to Project → Browse Catalog**.



The **Select an item to add to the current project** page opens.

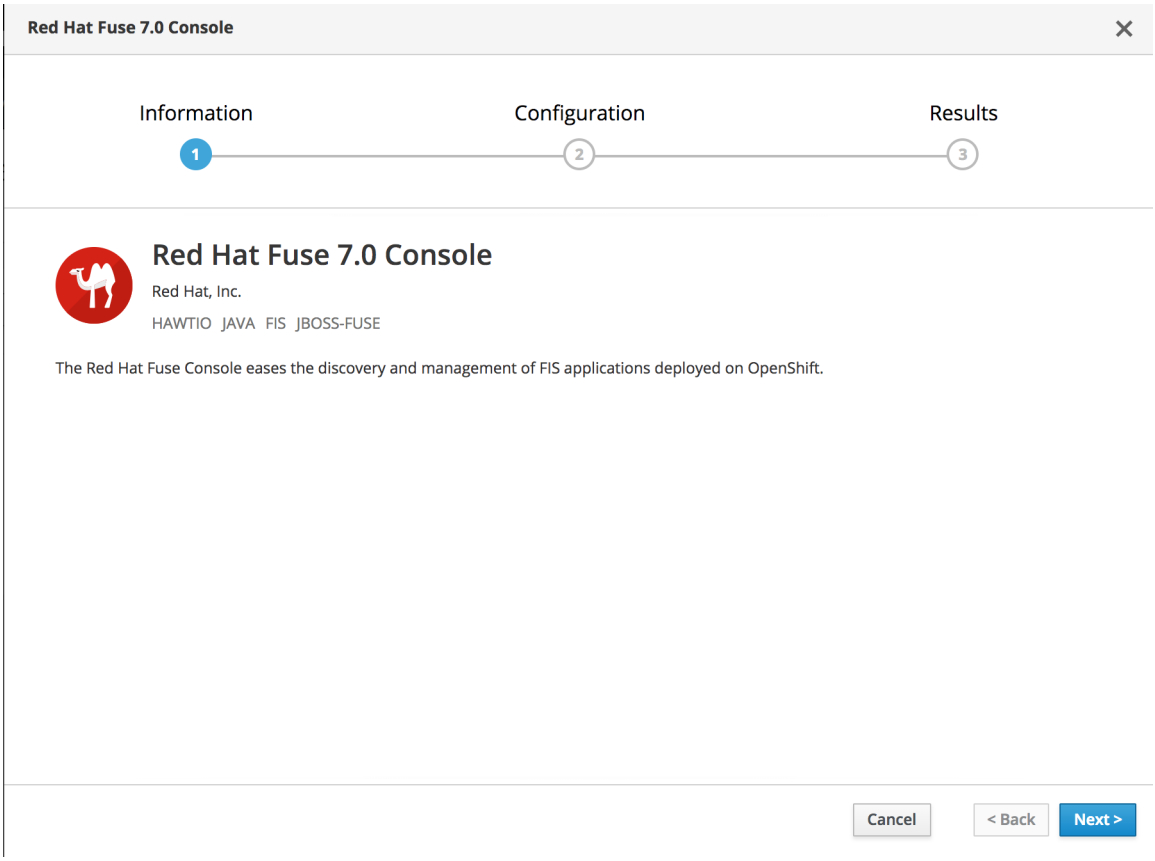
b. In the **Search** field, type **Fuse Console**.
The **Red Hat Fuse Console** item should appear as the search result.



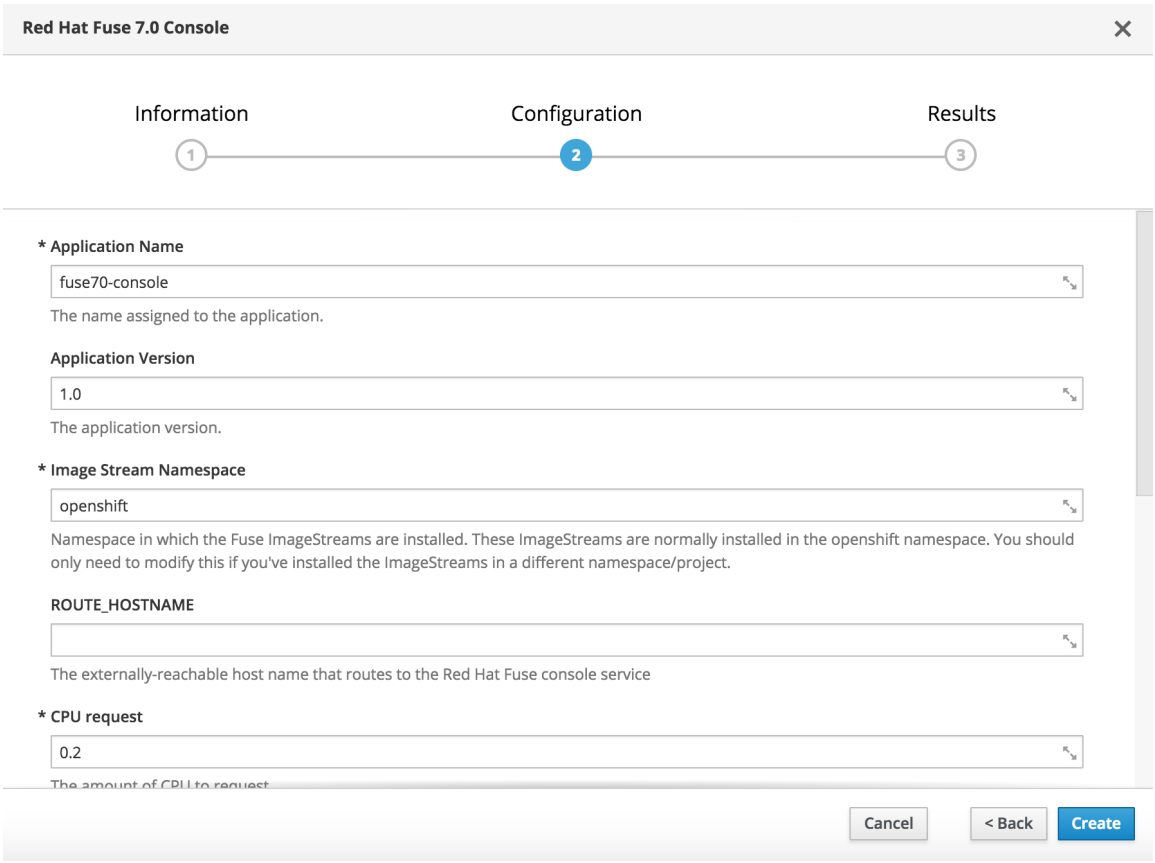
NOTE

If the **Red Hat Fuse Console** item does not appear as the search result, or if the item that appears is not the latest version, you can install the Fuse Console templates manually as described in the "Prepare the OpenShift server" section of the Fuse on OpenShift Guide.

c. Click the **Red Hat Fuse Console** item.
The **Red Hat Fuse Console** wizard opens.

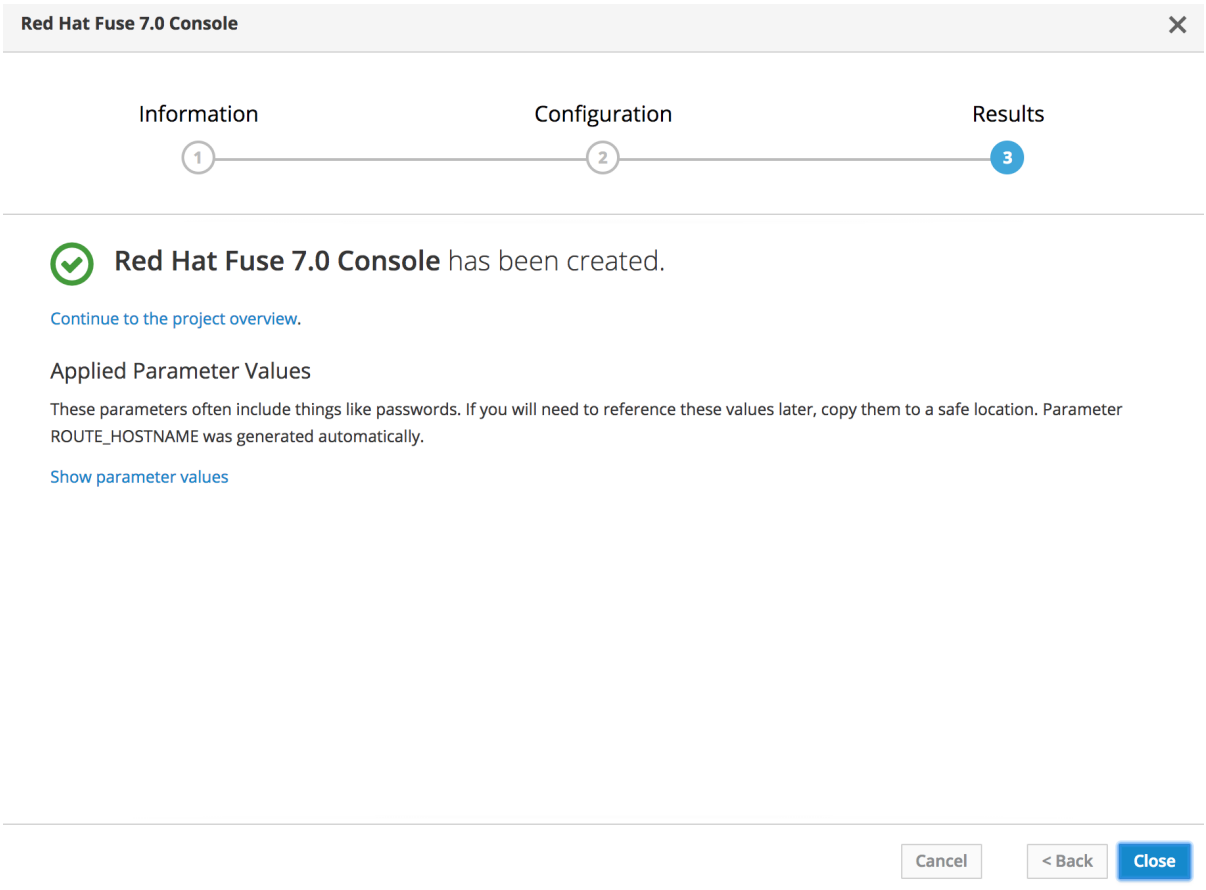


d. Click **Next**.

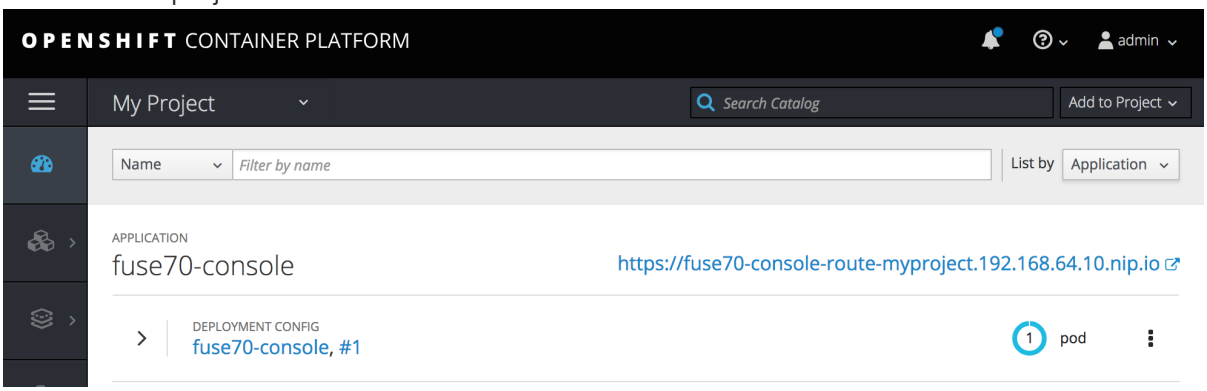


Optionally, you can change the default values of the configuration parameters.

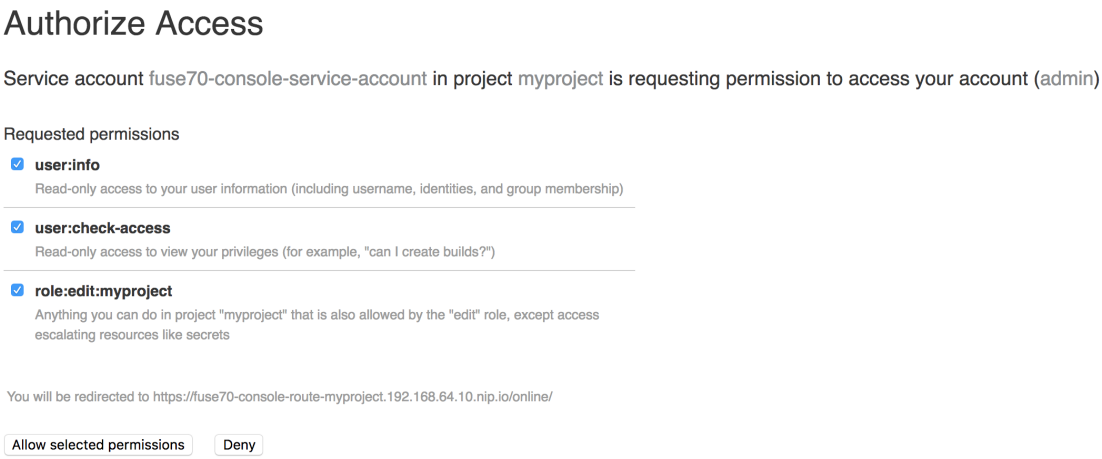
3. Click **Create**.
The **Results** page of the wizard indicates that the Red Hat Fuse Console has been created.



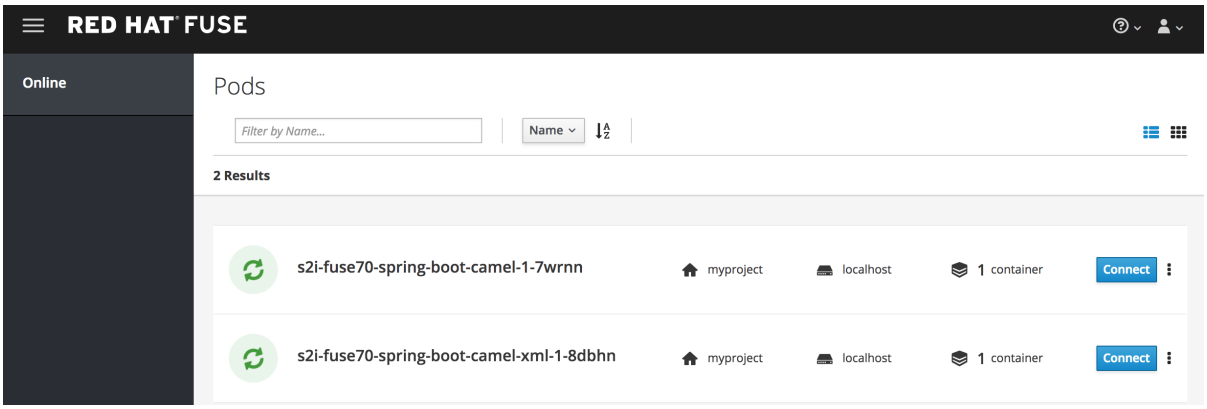
4. Click the **Continue to the project overview**link to verify that the Fuse Console application is added to the project.



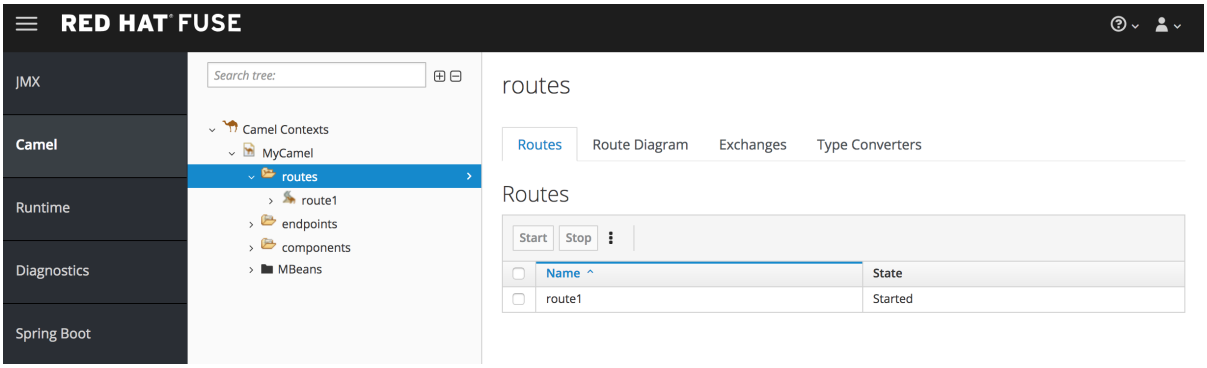
5. To open the Fuse Console, click the provided URL link and then log in.
An **Authorize Access** page opens in the browser listing the required permissions.



6. Click **Allow selected permissons**.
The Fuse Console opens in the browser and shows the Fuse pods running in the project, as shown for the example project named **myproject**:



7. Click **Connect** for the application that you want to view.
A new browser window opens showing the application in the Fuse Console.



1.3. DEPLOYING THE FUSE CONSOLE FROM THE COMMAND LINE

Table 1.1, "Fuse Console templates" describes the two OpenShift templates that you can use to access the Fuse Console from the command line, depending on the type of Fuse application deployment.

Table 1.1. Fuse Console templates

Type	Description
cluster	Use an OAuth client that requires the cluster-admin role to be created. The Fuse Console can discover and connect to Fuse applications deployed across multiple namespaces or projects.
namespace	Use a service account as OAuth client, which only requires the admin role in a project to be created. This restricts the Fuse Console access to this single project, and as such acts as a single tenant deployment.

Optionally, you can view a list of the template parameters by running the following command:

```
oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-710017-redhat-00006/fis-console-namespace-template.json
```

To deploy the Fuse Console from the command line:

1. Create a new application based on a Fuse Console template by running one of the following commands (where **myproject** is the name of your project):

   - For the Fuse Console **cluster** template, where **myhost** is the hostname to access the Fuse Console:

     ```
     oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-710017-redhat-00006/fis-console-cluster-template.json -p ROUTE_HOSTNAME=myhost
     ```

   - For the Fuse Console **namespace** template:

     ```
     oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-710017-redhat-00006/fis-console-namespace-template.json
     ```

NOTE

You can omit the route_hostname parameter for the **namespace** template because OpenShift automatically generates one.

2. Obtain the status of your deployment by running this command:

```
oc status
```

Here is an example response:

```
In project myproject on server https://192.168.64.12:8443
https://fuse-console.192.168.64.12.nip.io (redirects) (svc/fuse70-console-service)
dc/fuse70-console deploys openshift/jboss-fuse70-console:1.0
deployment #1 deployed 2 minutes ago - 1 pod
```

3. To access the Fuse Console from a browser, use the provided URL (for example, https://fuse-console.192.168.64.12.nip.io).

CHAPTER 2. ACCESSING THE FUSE CONSOLE FOR SPRING BOOT STANDALONE

To access the Fuse Console for a standalone Fuse Spring Boot distribution:

1. Add **hawtio-springboot** to your Fuse application's **pom.xml** file dependencies.

```
<dependency>
  <groupId>io.hawt</groupId>
  <artifactId>hawtio-springboot</artifactId>
</dependency>
```

   Note that you do not need to specify the version because it is provided by the Maven BOM.

2. Edit the **src/main/resources/application.properties** file:

   a. Set the following properties to false:

      - **endpoints.jolokia.sensitive**
      - **endpoints.hawtio.sensitive**
      - **hawtio.authenticationEnabled**

   b. Set the following properties to true:

      - **endpoints.hawtio.enabled**
      - **endpoints.jolokia.enabled**

        Your application.properties settings should like similar to the following example:

```
# ports

server.port=8080

management.port=10001


# enable management endpoints for healthchecks and hawtio

endpoints.enabled = false

endpoints.hawtio.enabled = true

endpoints.jolokia.enabled = true

endpoints.health.enabled = true

management.health.defaults.enabled=false

camel.health.enabled=false

camel.health.indicator.enabled=true

endpoints.jolokia.sensitive=false
```

```
endpoints.hawtio.sensitive=false

hawtio.authenticationEnabled=false
```

NOTE

By default, authentication for the Fuse Console on Spring Boot is disabled. Optionally, you can enable authentication by writing code specific to your Fuse Console distribution. Here is an example that you can use for guidance:

https://github.com/hawtio/hawtio/tree/master/examples/springboot-authentication

3. Run the Fuse application:

```
mvn spring-boot:run
```

4. To determine the port number for the Fuse Console URL, obtain the **management.port** value by looking at the value set in the **src/main/resources/application.properties** file. For example:

```
management.port   = 10001
```

5. To open the Fuse Console in a browser, use the following URL syntax where **nnnnn** is the value of the **management.port** property:
http://localhost:nnnnn/hawtio/index.html

   For example: **http://localhost:10001/hawtio/index.html**

CHAPTER 3. ACCESSING THE FUSE CONSOLE FOR JBOSS EAP

Before you can access the Fuse Console for Red Hat JBoss Enterprise Application Platform, you must install Fuse on the JBoss EAP container. For step-by-step instructions, see Installing on JBoss EAP.

To access the Fuse Console for a standalone JBoss EAP distribution:

1. Start Red Hat Fuse standalone with the following command:
   On Linux/Mac OS: **./bin/standalone.sh**

   On Windows: **./bin/standalone.bat**

2. In a web browser, enter the URL to connect to the Fuse Console. For example:
   http://localhost:8080/hawtio

3. In the login page, enter your user name and password and then click **Log In**.

By default, the Fuse Console shows the Home page. The left navigation tabs indicate the running plugins.

3.1. RESOLVING DELAYED FUSE CONSOLE LOADING

If the main Fuse Console page takes a long time to display in the browser, you might need to reduce the number and the size of the log files. You can use the **periodic-size-rotating-file-handler** to rotate the file when it reaches a maximum size (rotate-size) and maintains a number of files (max-backup-index). For details on how to use this handler, see the Red Hat JBoss Enterprise Application Platform product documentation.

CHAPTER 4. ACCESSING THE FUSE CONSOLE FOR KARAF

To access the Fuse Console for Apache Karaf standalone:

1. Install Fuse on the Karaf container. For step-by-step instructions, see Installing on Apache Karaf.
2. In the command line, navigate to the directory in which you installed Red Hat Fuse and run the following command to start Fuse standalone:

   ```
   ./bin/fuse
   ```

   The Karaf console starts and shows version information, the default Fuse Console URL, and a list of common commands.
3. In a browser, enter the URL to connect to the Fuse Console. For example:
   http://localhost:8181/hawtio
4. In the login page, enter your user name and password and then click **Log In**.

By default, the Fuse Console shows the Home page. The left navigation tabs indicate the running plugins.

4.1. SECURING FUSE CONTAINERS ON KARAF

You can implement the following security features to secure Fuse containers on Apache Karaf:

- Enable SSL/TLS security
- Configure user authorization

4.1.1. Enabling SSL/TLS security

SSL/TLS security is not enabled by default for the Fuse Console. It is recommended that you enable SSL/TLS security on the Fuse Console to protect username/password credentials from snooping. For detailed instructions on how to enable SSL/TLS security, see the Security Guide.

4.1.2. Controlling user access to the Fuse Console

NOTE

For this release, role-based access control for the Fuse Console is only enabled for Fuse on Karaf standalone.

The operations that an authenticated user are allowed to perform depend on the role (or roles) assigned to that user, as listed in Table 4.1, “Role-based access on Karaf standalone”.

Ensure that you have the necessary user role authorization to perform the Fuse Console operations that you want to perform.

To set a user role:

1. Open the Red Hat Fuse **etc/users.properties** file in an editor.

2. Add an entry for the user name, password, and role.
   For example, the following entry in the **etc/users.properties** file defines the admin user and grants the admin role.

   ```
   admin = secretpass,admin
   ```

3. Save the file.

Table 4.1. Role-based access on Karaf standalone

Operation	admin	manager	viewer
login/logout	Y	Y	Y
View Help topics	Y	Y	Y
Set user preferences	Y	Y	Y
Connect			
Discover and connect to remote integrations	Y	Y	Y
Discover and connect to local integrations	Y	Y	Y
Camel			
View all running Camel applications	Y	Y	Y
Start, suspend, resume, and delete Camel Contexts	Y	Y	N
Send messages	Y	Y	N
Add endpoints	Y	Y	N
View routes, route diagrams, and runtime statistics	Y	Y	Y
Start and stop routes	Y	Y	N
Delete routes	Y	Y	N
JMX			

Operation	admin	manager	viewer
Change attribute values	Y	Y	N
Select and view attributes in a time-based chart	Y	Y	Y
View operations	Y	Y	Y
OSGI			
View bundles, features, packages, services, servers, framework, and configurations	Y	Y	Y
Add and delete bundles	Y	Y	N
Add configurations	Y	Y	N
Install and uninstall features	Y	N	N
Runtime			
View system properties, metrics, and threads	Y	Y	Y
Logs			
Viewing logs	Y	Y	Y

For more information on role-based access control, see Deploying into Apache Karaf.

CHAPTER 5. DISABLING THE FUSE CONSOLE

How you disable the Fuse Console depends on your Fuse distribution.

5.1. KARAF

To disable the Fuse Console on Karaf so that it becomes inaccessible to all users without affecting any other component, follow these steps:

1. To determine the hawtio-web bundle ID, use the following command to list the Fuse bundles that the Fuse Console uses:
   osgi:list | grep hawtio

2. To stop the bundle, use the **osgi:stop** command. For example, if the **hawtio :: Web console** bundle has an ID of 246, type this command:
   osgi:stop 246

The bundle goes into the resolved state and you can no longer access the Fuse Console.

For more information about managing bundles, see the "Lifecycle Management" chapter of Deploying into Apache Karaf.

5.2. JBOSS EAP

To disable the Fuse Console on JBoss EAP, do one of the following:

- Remove the deployment file: **$EAP_HOME/standalone/deployments/hawtio-wildfly-xxxxx.war**
- Undeploy the Fuse Console by using the JBoss EAP admin console or command line interface.

5.3. SPRING BOOT

For Spring Boot, the Fuse Console is not enabled by default.

5.4. OPENSHIFT

For OpenShift, the Fuse Console is not enabled by default.

PART II. VIEWING AND MANAGING FUSE APPLICATIONS

You can view and manage Fuse applications as described in the following sections:

CHAPTER 6. VIEWING CONTAINERS AND APPLICATIONS (FUSE ON OPENSHIFT)

When you login to the Fuse Console for OpenShift, the Fuse Console home page shows the available containers.

To manage (create, edit, or delete) containers, use the OpenShift console.

To view Fuse applications on the OpenShift cluster, click the **Online** tab.

CHAPTER 7. CONNECTING TO REMOTE FUSE INTEGRATIONS (STANDALONE DISTRIBUTIONS)

The Fuse Console uses Jolokia, an agent-based approach to Java Management Extensions (JMX) that requires extra software (an agent) installed on the client. By default, Red Hat Fuse includes a jolokia agent.

With standalone Fuse Console distributions, you can connect to remote integrations that already have a jolokia agent (https://jolokia.org/) running inside them. If the process that you want to connect to does not have a jolokia agent inside, refer to the jolokia documentation (http://jolokia.org/agent.html).

7.1. UNLOCKING THE FUSE CONSOLE

By default, Jolokia for Fuse 7 standalone (on Apache Karaf} and JBoss EAP) is locked and the Fuse Console is not accessible remotely.

To unlock the Fuse Console for a hostname or IP address other than **locahost** or **127.0.0.1**, follow these steps:

1. Open the **jolokia-access.xml** file in an editor.
   On Karaf, the XML file is located in the **$KARAF_HOME/etc** folder.

   On JBoss EAP, it is located in the **$EAP_HOME/standalone/configuration** folder.

2. Register the hostnames or IP addresses for the Fuse integrations that you want to access with the Fuse console by adding them to the **<cors>** section.
   For example, to access hostname **0.0.0.3** from the Fuse Console, add the

   ```
   *<allow-origin>http://0.0.0.3:*</allow-origin>*
   ```

   line as shown:

   ```
   <!--

     Cross-Origin Resource Sharing (CORS) restrictions

     By default, only CORS access within localhost is allowed for maximum security.

     You can add trusted hostnames in the <cors> section to unlock CORS access from them.

     -->

     <cors>

       <!-- Allow cross origin access only within localhost -->

       <allow-origin>http*://localhost:*</allow-origin>

       <allow-origin>http*://127.0.0.1:*</allow-origin>

       <allow-origin>http://0.0.0.3:*</allow-origin>

       <!-- Whitelist the hostname patterns as <allow-origin> -->
   ```

```
    <!--
    <allow-origin>http*://*.example.com</allow-origin>
    <allow-origin>http*://*.example.com:*</allow-origin>
    -->
    <!-- Check for the proper origin on the server side to protect against CSRF -->
    <strict-checking />
  </cors>
```

3. Save the file.

7.2. RESTRICTING REMOTE ACCESS

Optionally, you can restrict remote access to the Fuse Console for specific hosts and IP addresses.

You can grant overall access based on the IP address of an HTTP client. To specify these restrictions:

In the **jolokia-access.xml** file, add or edit a **<remote>** section that contains one or more **<host>** elements. For the **<host>** element, you can specify an IP address, a host name, or a netmask given in CIDR format (for example, **10.0.0.0/16** for all clients coming from the 10.0 network).

The following example allows access from localhost and all clients whose IP addresses start with **10.0**. For all other IP addresses, access is denied.

```
<remote>
  <host>localhost</host>
  <host>10.0.0.0/16</host>
</remote>
```

For more details, see the Jolokia security documentation (https://jolokia.org/reference/html/security.html).

7.3. ALLOWING CONNECTIONS TO REMOTE FUSE INSTANCES

The Fuse Console's proxy servlet uses whitelist host protection, with which by default the Fuse Console can only connect to localhost. If you want to connect the Fuse Console to other remote Fuse instances, you need to configure the whitelist as follows:

- For Apache Karaf, make the following configuration changes in **etc/system.properties** file:

```
hawtio.proxyWhitelist = localhost, 127.0.0.1, myhost1, myhost2, myhost3
```

- For JBoss EAP, make the following configuration changes in the **standalone/configuration/standalone-*.xml** file:

```
 <property name=hawtio.proxyWhitelist" value="localhost, 127.0.0.1, myhost1, myhost2,
myhost3"/>
```

- For Spring Boot, configure the **hawtio.proxyWhitelist** system property in the **main()** method of your Spring Boot application:

```
System.setProperty("hawtio.proxyWhitelist", "localhost, 127.0.0.1, myhost1, myhost2, myhost3");
```

7.4. CONNECTING TO A REMOTE JOLOKIA AGENT

Before you begin, you need to know the connection details (host name, port, and path) of the remote Jolokia agent.

Here are the default connection URLs for the Jolokia agent depending on your Fuse distribution:

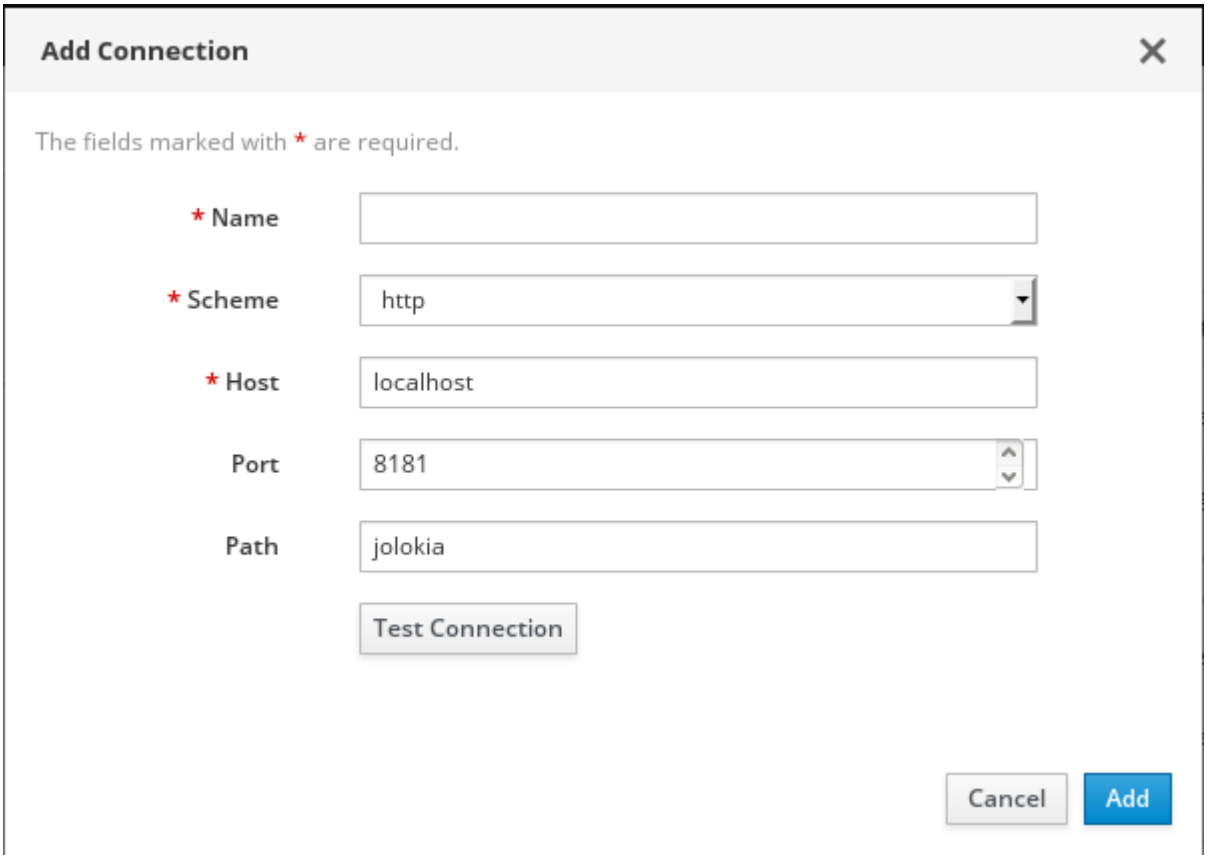
- Spring Boot: **http://<host>:8080/jolokia**
- Red Hat JBoss EAP: **http://<host>:8080/hawtio/jolokia**
- Fuse Karaf: **http://<host>:8181/hawtio/jolokia**

As a system administrator, you can change these defaults.

Typically, the URL to remotely connect to a Jolokia agent is the URL to open the Fuse Console plus **/jolokia**. For example, if the URL to open the Fuse Console is **http://<host>:1234/hawtio**, then the URL to remotely connect to it would probably be **http://<host>:1234/hawtio/jolokia**.

To connect to a remote Jolokia instance so that you can examine its JVM:

1. Click the **Connect** tab.
2. Click the **Remote** tab, and then **Add connection**.



3. Type the **Name**, **Scheme** (HTTP or HTTPS), and the **hostname**.

4. Click **Test Connection**.
5. Click **Add**.

NOTE

The Fuse Console automatically probes the local network interfaces other than localhost and 127.0.0.1 and adds them to the whitelist. Hence, you do not need to manually register the local machine's addresses to the whitelist.

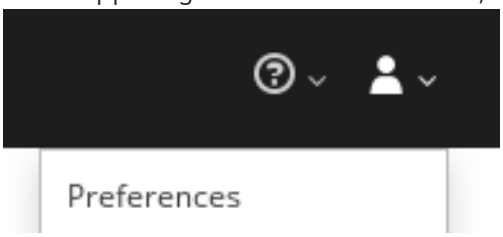
7.5. SETTING DATA MOVING PREFERENCES

You can change the following Jolokia preferences, for example, if you want to more frequently refresh data that displays in the Fuse Console. Note that increasing the frequency of data updates impacts networking traffic and increases the number of requests made to the server.

- **Update rate** - The period between polls to Jolokia to fetch JMX data (the default is 5 seconds).
- **Maximum depth** - The number of levels that Jolokia will marshal an object to JSON on the server side before returning (the default is 7).
- **Maximum collection size** - The maximum number of elements in an array that Jolokia marshals in a response (the default is 50,000).

To change the values of these settings:

1. In the upper right of the Fuse Console, click the user icon and then click **Preferences**.



2. Edit the options and then click **Close**.

7.6. VIEWING JVM RUNTIME INFORMATION

To view JVM runtime information, such as system properties, metrics, and threads, click the **Runtime** tab.

CHAPTER 8. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

In the Fuse Console's **Camel** tab, you view and manage Apache Camel contexts, routes, and dependencies.

8.1. OVERVIEW

The **Camel** tab is only available when you connect to a container that uses one or more Camel routes.

You can view the following details:

- A list of all running Camel contexts
- Detailed information of each Camel context such as Camel version number and runtime statics
- Lists of all routes in each Camel application and their runtime statistics
- Graphical representation of the running routes along with real time metrics

You can also interact with a Camel application by:

- Starting and suspending contexts
- Managing the lifecycle of all Camel applications and their routes, so you can restart, stop, pause, resume, etc.
- Live tracing and debugging of running routes
- Browsing and sending messages to Camel endpoints

8.2. INTERACTING WITH A CAMEL APPLICATION

To start, suspend, or delete a context:

1. In the **Camel** tab's tree view, click **Camel Contexts**.
2. Check the box next to one or more contexts in the list.
3. Click **Start** or **Suspend**.
4. To delete a context, you must first stop it. Then click the ellipse icon and select **Delete** from the dropdown menu.

NOTE

When you delete a context, you remove it from the deployed application.

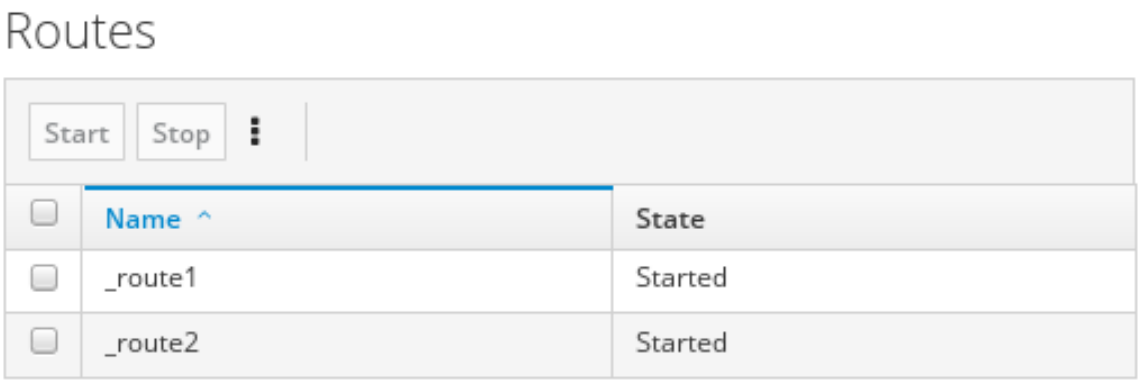
To view Camel application details:

1. In the **Camel** tab's tree view, click a Camel application.
2. To view a list of application attributes and values, click **Attributes**.

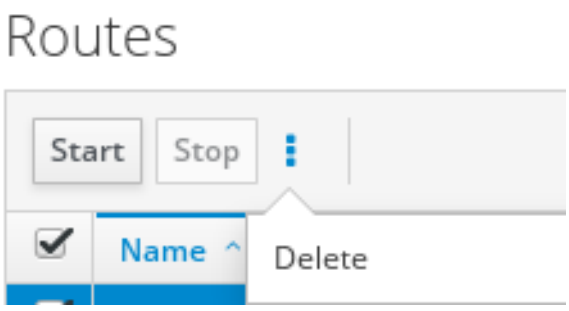
3. To view a graphical representation of the application attributes::
   a. Click **Chart**.
   b. Click **Edit** to select the attributes that you want to see in the chart.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. To view application endpoints, click **Endpoints**. You can filter the list by **URL**, **Route ID**, and **direction**.
6. Click **Type Converters** to view, enable, and disable statistics related to the Camel built-in type conversion mechanism, that is used to convert message bodies and message headers to different types.
7. Click **Operations** to view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath.

To interact with Camel routes:

1. In the **Camel** tab's tree view, click the application's routes folder to view a list of the routes:



2. To start, stop, or delete one or more routes:
   a. Check the box next to one or more routes in the list.
   b. Click **Start** or **Stop**.
   c. To delete a route, you must first stop it. Then click the ellipse icon and select **Delete** from the dropdown menu.



NOTE

- When you delete a route, you remove it from the deployed application.
- You can also select a specific route in the tree view and then click the upper-right menu to start, stop, or delete it.

3. To view a graphical diagram of the routes, click **Route Diagram**.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. To view endpoints, click **Endpoints**. You can filter the list by URL, Route ID, and direction.
6. Click **Type Converters** to view, enable, and disable statistics related to the Camel built-in type conversion mechanism, which is used to convert message bodies and message headers to different types.

To interact with a specific route:

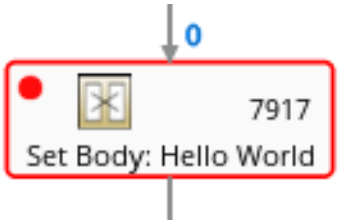
1. In the **Camel** tab's tree view, select a route.
2. To view a list of route attributes and values, click **Attributes**.
3. To view a graphical representation of the route attributes, click **Chart**. You can click **Edit** to select the attributes that you want to see in the chart.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. Click **Operations** to view and execute JMX operations on the route, such as dumping the route as XML or getting the route's Camel ID value.

To trace messages through a route:

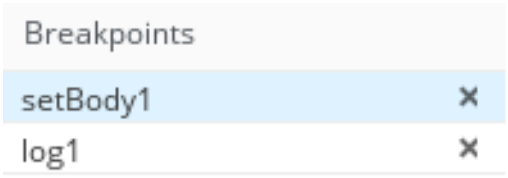
1. In the **Camel** tab's tree view, select a route.
2. Select **Trace**, and then click **Start tracing**.
3. To send messages to a route:
   a. In the **Camel** tab's tree view, open the context's endpoints folder and then select an endpoint.
   b. Click the **Send** subtab.
   c. Configure the message in JSON or XML format.
   d. Click **Send**.
4. Return to the route's **Trace** tab to view the flow of messages through the route.

To debug a route:

1. In the **Camel** tab's tree view, select a route.
2. Select **Debug**, and then click **Start debugging**.
3. To add a breakpoint, select a node in the diagram and then click **Add breakpoint**. A red dot appears in the node:



The node is added to the list of breakpoints:



4. Click the down arrow to step to the next node or the **Play** button to resume running the route.
5. Click the **Pause** button to suspend all threads for the route.
6. Click **Stop debugging** when you are done. All breakpoints are cleared.

CHAPTER 9. VIEWING AND MANAGING JMX DOMAINS AND MBEANS

Java Management Extensions (JMX) is a Java technology that allows you to manage resources (services, devices, and applications) dynamically at runtime. The resources are represented by objects called MBeans (for Managed Bean). You can manage and monitor resources as soon as they are created, implemented, or installed.

With the JMX plugin on the Fuse Console, you can view and manage JMX domains and MBeans. You can view MBean attributes, run commands, and create charts that show statistics for the MBeans.

The **JMX** tab provides a tree view of the active JMX domains and MBeans organized in folders. You can view details and execute commands on the MBeans.

To view and edit MBean attributes:

1. In the tree view, select an MBean.
2. Click the **Attributes** tab.
3. Click an attribute to see its details.

To perform operations:

1. In the tree view, select an MBean.
2. Click the **Operations** tab, expand one of the listed operations.
3. Click **Execute** to run the operation.

To view charts:

1. In the tree view, select an item.
2. Click the **Chart** tab.

CHAPTER 10. VIEWING AND MANAGING YOUR OSGI ENVIRONMENT (KARAF STANDALONE)

For Apache Karaf standalone distributions, you can view and manage the Red Hat Fuse OSGi environment. You can view and manage container bundles, features, and configurations, as well as Java packages and OSGi services.

The **OSGi** tab contains a series of subtabs with options for each container component:

Bundles

List of installed bundles. You can install and uninstall bundles, start and stop bundles, and edit bundle properties. You can also filter the list and toggle between list and grid view.

Features

List of available features. You can install and uninstall features or feature repositories, and drill down to view feature details.

Packages

List of installed Java packages. You can view package versions and associated bundles.

Services

List of running services. You can view service IDs, associated bundles and object classes.

Declarative Services

List of declarative OSGi services. You can view the service state and drill down to view service details. You can also activate and deactivate services.

Server

Detailed information about the local or remote host in read-only mode.

Framework

Configuration options for the container OSGi framework. You can set the framework start level and the initial bundle start level.

Configuration

List of configuration objects. You can view the state of each object and drill down to view or edit object details. You can also create a new configuration object.

10.1. VIEWING AND MONITORING THREAD STATE

To view and monitor the state of threads:

1. Click the **Runtime** tab and then the **Threads** subtab. The **Threads** page lists active threads and stack trace details for each thread. By default, the thread list shows all threads in descending ID order.

2. To sort the list by increasing ID, click the **ID** column label.

3. Optionally, filter the list by thread state (for example, **Blocked**) or by thread name.

4. To drill down to detailed information for a specific thread, such as the lock class name and full stack trace for that thread, in the **Actions** column, click **More**.

CHAPTER 11. VIEWING LOG ENTRIES

You can view log entries for Red Hat Fuse in the **Logs** tab. The **Logs** tab is available (for Fuse standalone on Karaf and Fuse standalone on JBoss EAP) when the Java application has the Log MBean available .

You can filter the list of logs to show specific log types, and drill down to each log entry to view detailed information about the log entry.

The **Logs** tab contains the following sections:

Action Bar

Options to filter the log entries section according to a text string or the logging level.

Log Entries

List view of the log entries. By default, the list shows log entries in ascending order. You can change the default sorting in the **User → Preferences → Server Logs** page. Click the log entry link to drill down to details about the log entry, such as the bundle name, thread, and the full message text.

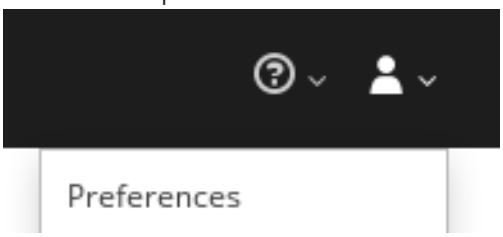
11.1. SETTING FUSE CONSOLE LOG ATTRIBUTES

You can customize the following Fuse Console browser settings for storing log messages:

- The number of log statements to keep in the Fuse Console (the default is 100).
- The global log level: **INFO** (the default), OFF, ERROR, WARN, and DEBUG.
- The child-level Hawtio messages to include, such as hawtio-oauth and hawtio-core-utils.

To change the defaults:

1. In the upper right corner of the Fuse Console, click the user icon and then click **Preferences** from the drop-down menu:



2. Edit the options and then click **Close**.

To reset the Console Logs settings to the default values, click **Reset → Reset settings**.

CHAPTER 12. CHANGING THE FUSE CONSOLE BRANDING

To change the branding name and image for the Fuse Console, you can edit the following files:

- The **hawtconfig.json** file in the Fuse Console's war file (**karaf-install-dir/system/io/hawt/hawtio-war/version/hawtio-war-version.war**):

```
{
  "branding": {
    "appName": "Hawtio Management Console",
    "appLogoUrl": "img/hawtio-logo.svg",
    "companyLogoUrl": "img/hawtio-logo.svg"
  },
  "about": {
    "title": "Hawtio Management Console",
    "productInfo": [],
    "additionalInfo": "",
    "copyright": "",
    "imgSrc": "img/hawtio-logo.svg"
  },
  "disabledRoutes": []
}
```

- The war file's style sheet (**.css**) file to modify other aspects of the Fuse Console UI.

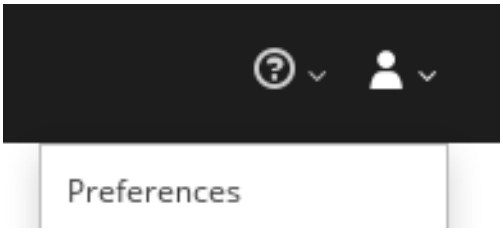
NOTE

If you have already run the Fuse Console in a web browser, the branding is stored in the browser's local storage. To use new branding settings, you must clear the browser's local storage.

CHAPTER 13. ENSURING THAT DATA DISPLAYS CORRECTLY IN THE FUSE CONSOLE

If the display of the queues and connections in the Fuse Console is missing queues, missing connections, or displaying inconsistent icons, adjust the Jolokia collection size parameter that specifies the maximum number of elements in an array that Jolokia marshals in a response.

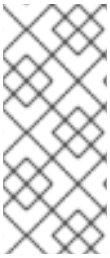
1. In the upper right corner of the Fuse Console, click the user icon and then click **Preferences**.



2. Increase the value of the **Maximum collection size** option (the default is 50,000).

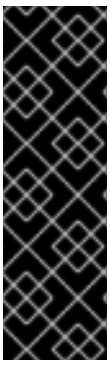
3. Click **Close**.

CHAPTER 14. ACCESSING PROMETHEUS



NOTE

For this release, Prometheus is only supported for Fuse on OpenShift. For documentation about Prometheus on OpenShift Container Platform, go to: https://access.redhat.com/documentation/en-us/openshift_container_platform/3.10/html-single/configuring_clusters/#openshift-prometheus



IMPORTANT

Prometheus on OpenShift is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information on Red Hat Technology Preview features support scope, see https://access.redhat.com/support/offerings/techpreview/.

You can use Prometheus to monitor and store Fuse on OpenShift data by exposing an endpoint with your Fuse application's data to Prometheus format. Prometheus stores the data so that you can use a graphical tool, such as Grafana, to visualize and run queries on the data.

You can use Prometheus to monitor Fuse applications on an installed and running single-node OpenShift cluster such as Minishift or the Red Hat Container Development Kit. A single-node OpenShift cluster provides you with access to a cloud environment that is similar to a production environment.

For information on installing and developing with Red Hat Fuse on OpenShift, see the Fuse on OpenShift Guide.

14.1. SETTING UP PROMETHEUS TO ACCESS FUSE APPLICATIONS ON MINISHIFT

Follow these steps to add Prometheus and Grafana to your Fuse on OpenShift project:

At the command line, follow these steps:

1. Start Minishift with the following options:

```
MINISHIFT_ENABLE_EXPERIMENTAL=y minishift start --ocp-tag v3.10.14 --extra-clusterup-flags "--enable=*,service-catalog"
```

2. At the prompt, login as the system admin user:

```
oc login -u system:admin
```

3. Download and install the Ansible Service Broker:

```
curl https://raw.githubusercontent.com/openshift/ansible-service-broker/release-1.2/apb/install.yaml | oc create -f -
```

4. Get a list of the installed projects:

```
oc get projects -w
```

This command returns a constantly updating list of projects:

```
NAME                        DISPLAY NAME   STATUS
automation-broker-apb                      Active
default                                    Active
kube-dns                                   Active
kube-proxy                                 Active
kube-public                                Active
kube-service-catalog                       Active
kube-system                                Active
myproject                   My Project     Active
openshift                                  Active
openshift-apiserver                        Active
openshift-controller-manager               Active
openshift-core-operators                   Active
openshift-infra                            Active
openshift-node                             Active
openshift-web-console                      Active
```

5. Monitor this list to see when the automation-broker is active. It can take some time (approximately two minutes) for **automation-broker-apb** to install the automation broker:

```
automation-broker            Active
```

6. When the list of projects indicates that the automation-broker is active, press **CTRL + C** to cancel the **oc get projects -w** command.

7. Run the following command:

```
oc export cm/broker-config -n automation-broker  | sed 's/sandbox_role: .*/sandbox_role:
\"admin\"/' | oc replace -f - cm/broker-config -n automation-broker
```

The command returns this message:

```
configmap "broker-config" replaced
```

It then takes some time to retrieve the Ansible Playbook catalog. (Now is a good time for you to take a coffee or tea break.)

NOTE

You can use the following command to monitor the status of the automation-broker:

```
oc get pods -n automation-broker -w
```

8. Wait several minutes and then run this command:

```
oc rollout latest dc/automation-broker -n automation-broker
```

If the following message is returned, you must wait longer and then rerun the oc rollout command:

```
Error from server (NotFound): deploymentconfigs.apps.openshift.io "automation-broker" not found
```

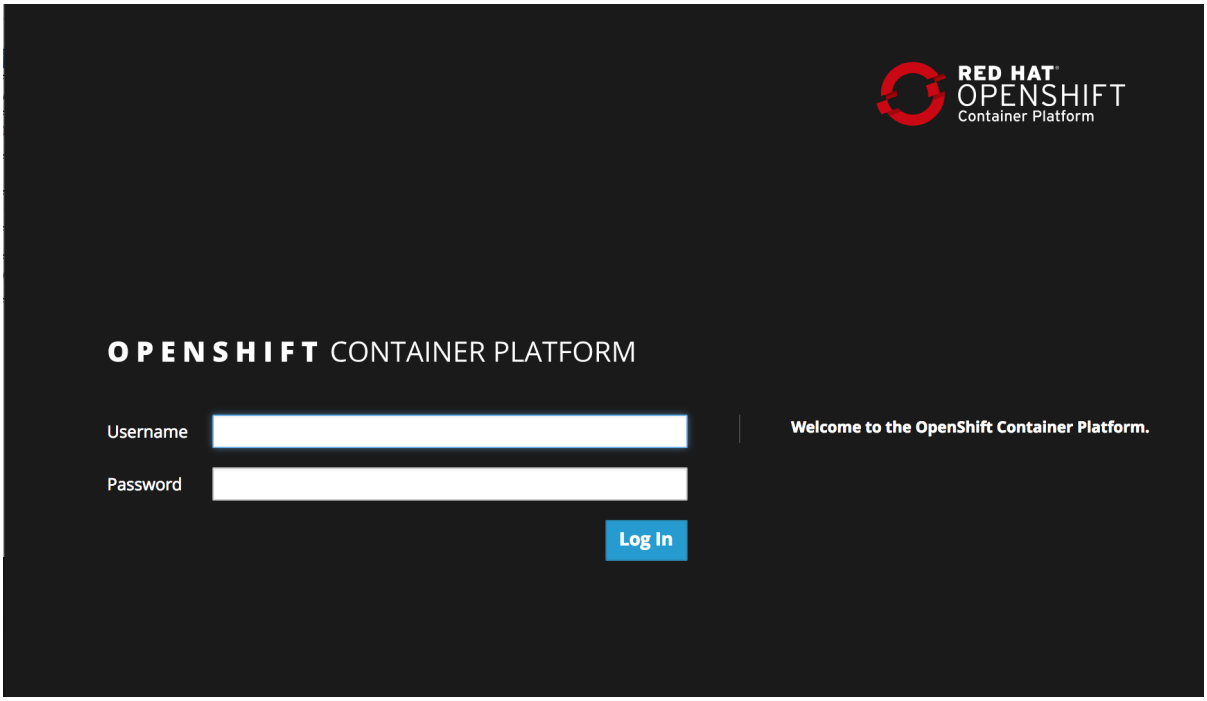
When the following message is returned by the **oc rollout** command, the command was successful and you can move to the next step:

```
error: #1 is already in progress (Running).
```

9. Open the OpenShift web console:

```
minishift console
```

Your web browser opens:



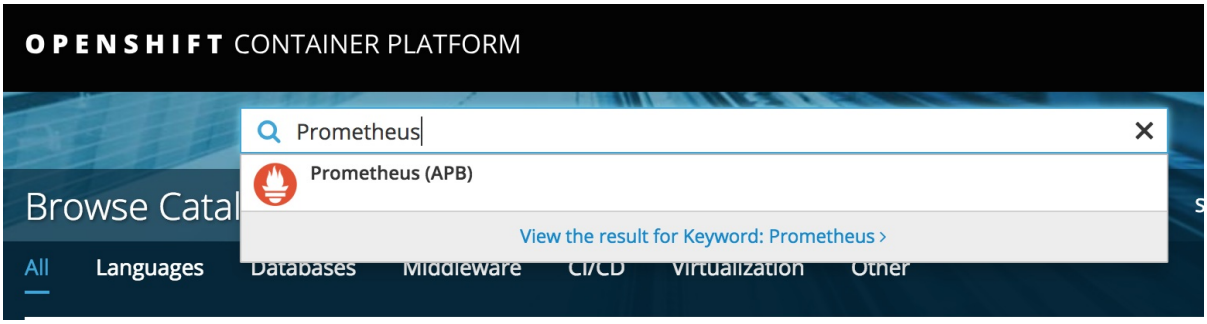
Follow these steps in the OpenShift web console:

1. Log into the web console with your developer credentials (for example, **developer**/**developer**).

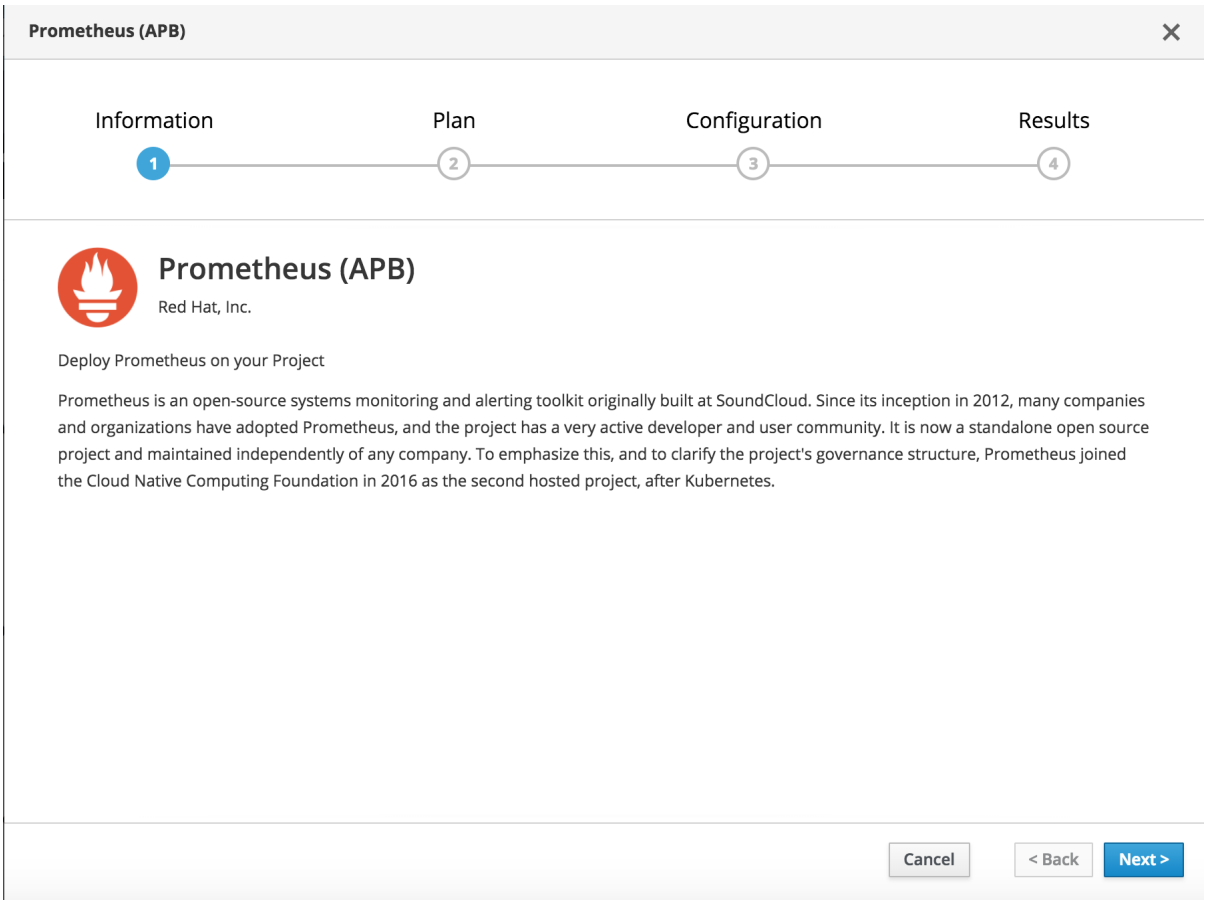
NOTE

If you are already logged into the web console, clear the web browser cache so that you can find the **Prometheus (APB)** item in the next step.

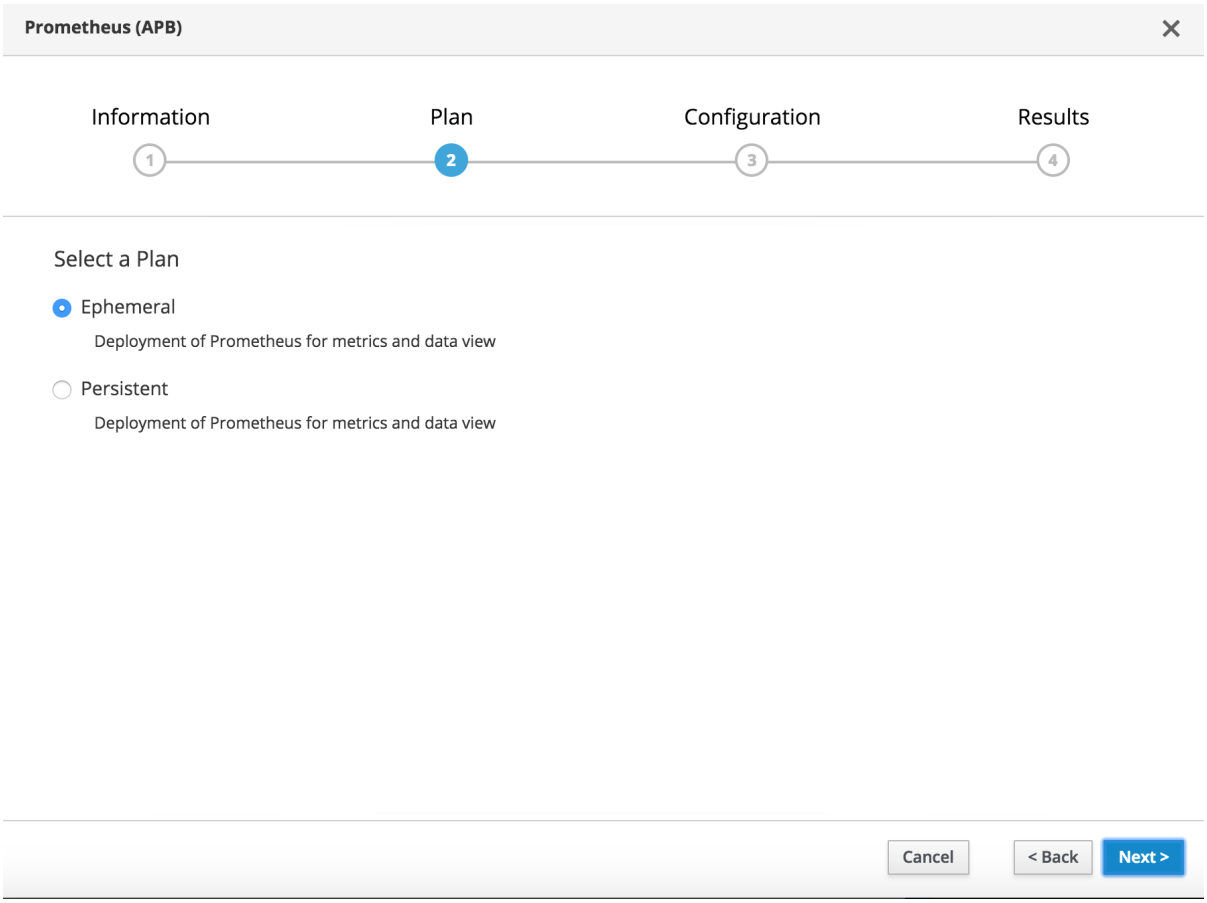
2. In the catalog search field, type **Prometheus**, and then select the **Prometheus (APB)** item:



The **Prometheus (APB)** wizard opens:

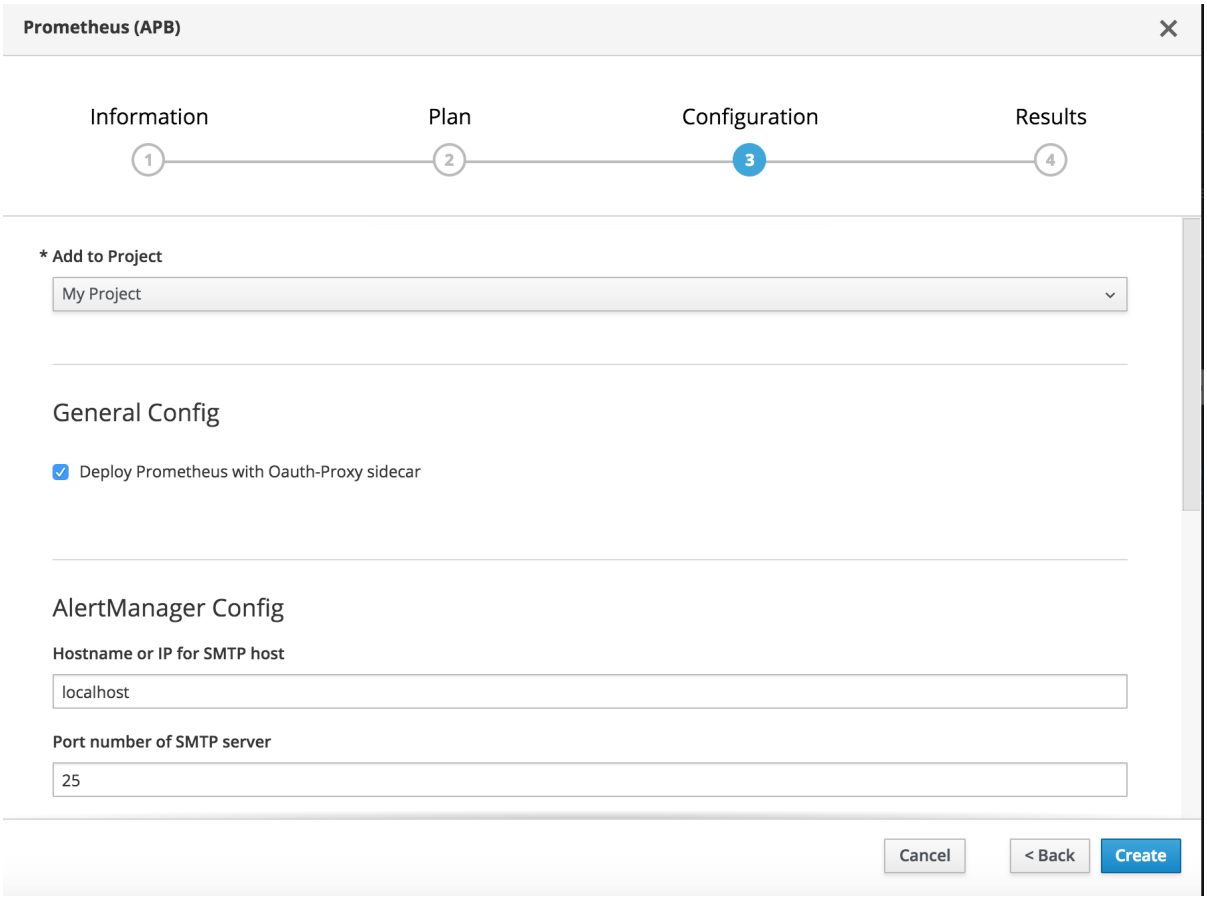


3. Click **Next**.
   Page 2 of the wizard opens:

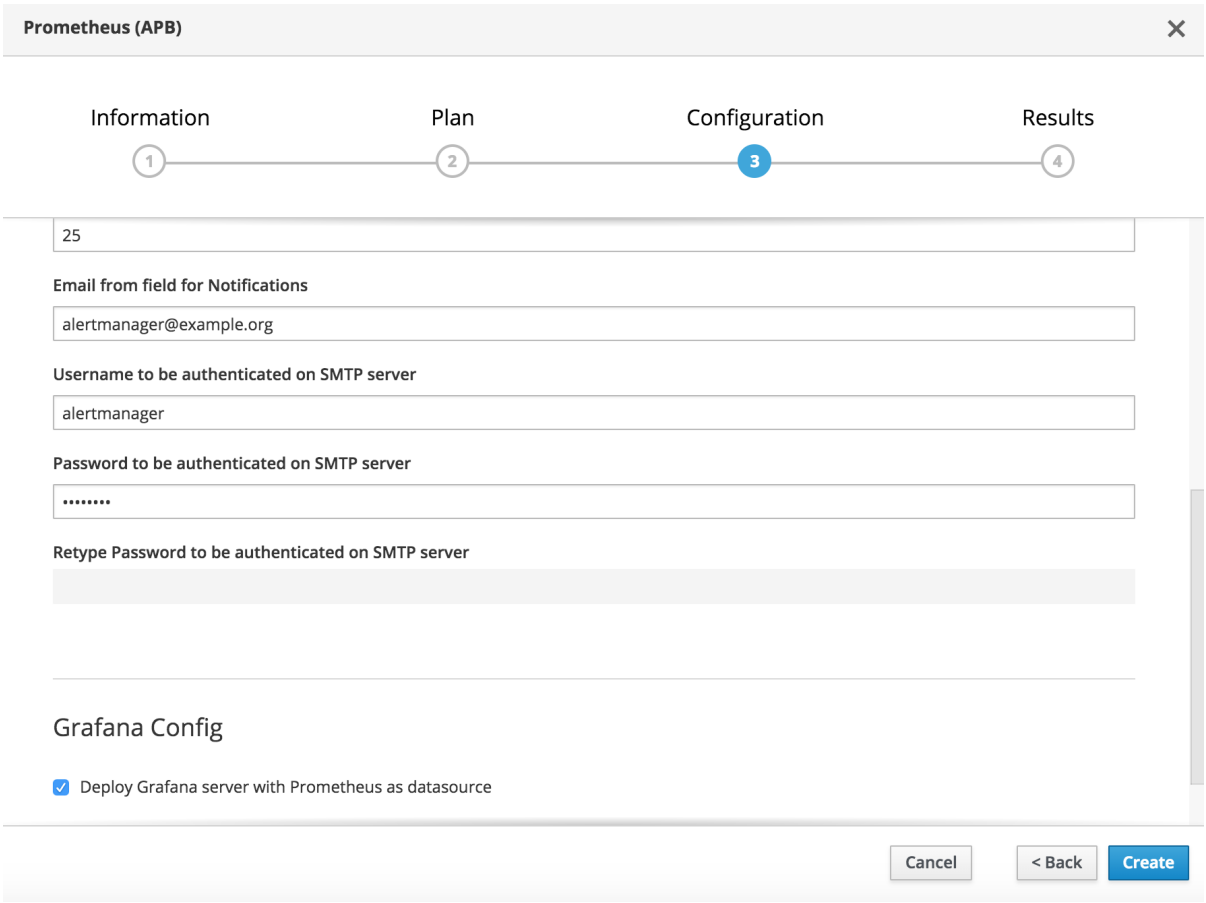


4. Select **Ephemeral** or **Persistent**. If you select **Ephemeral**, OpenShift stores the Prometheus metrics and data only for the duration of the current session. If you select **Persistent**, OpenShift writes the Prometheus metrics and data to disk and stores it until the Minishift VM is deleted.

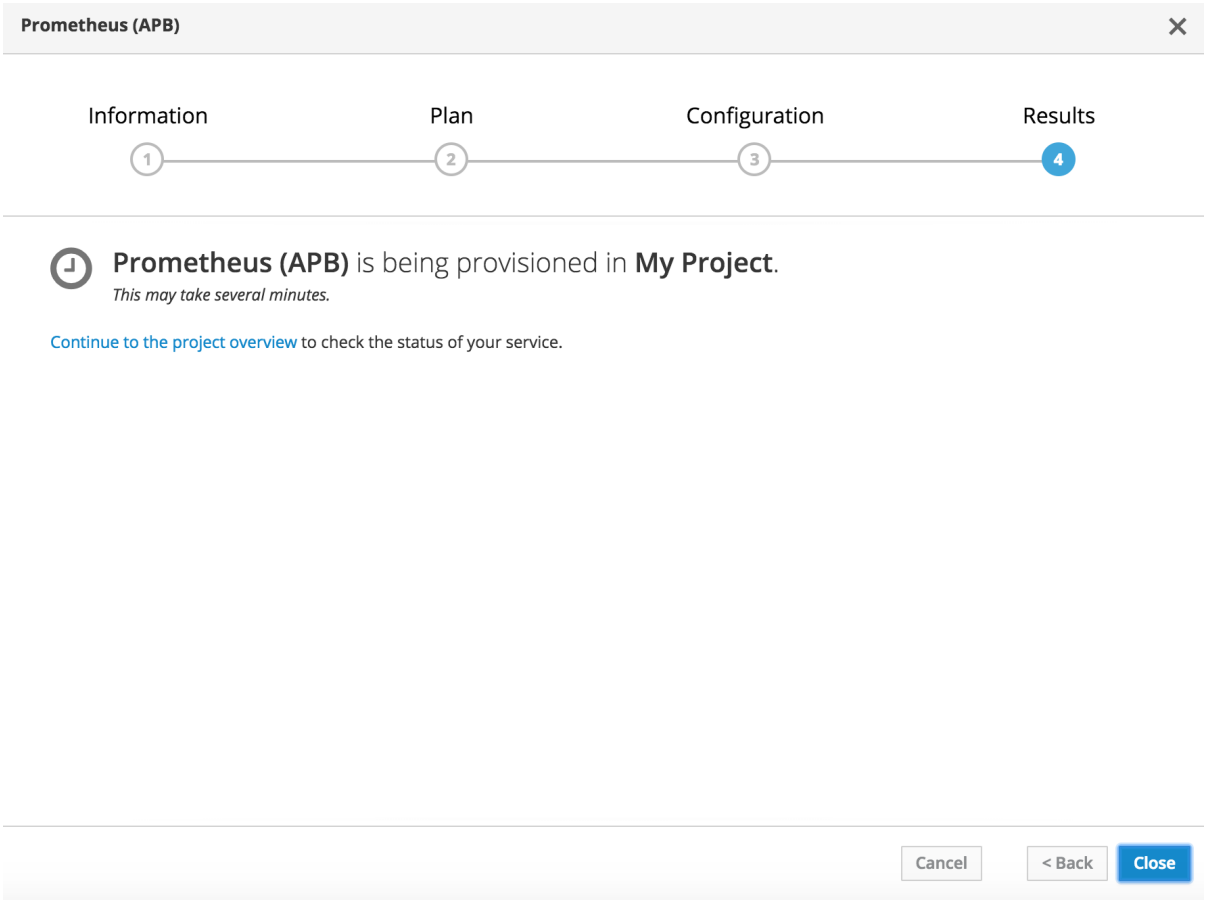
5. Click **Next**.
   Page 3 of the wizard opens:



6. Specify the project that contains the Fuse application.
7. If you want to require OpenShift login to access the Prometheus and Grafana dashboards, check the **Deploy Prometheus with Oauth-Proxy sidecar**option.
8. Scroll down to see the **Grafana config** section and then check the **Deploy Grafana server with Prometheus as datasource** option.



9. Click **Create**.

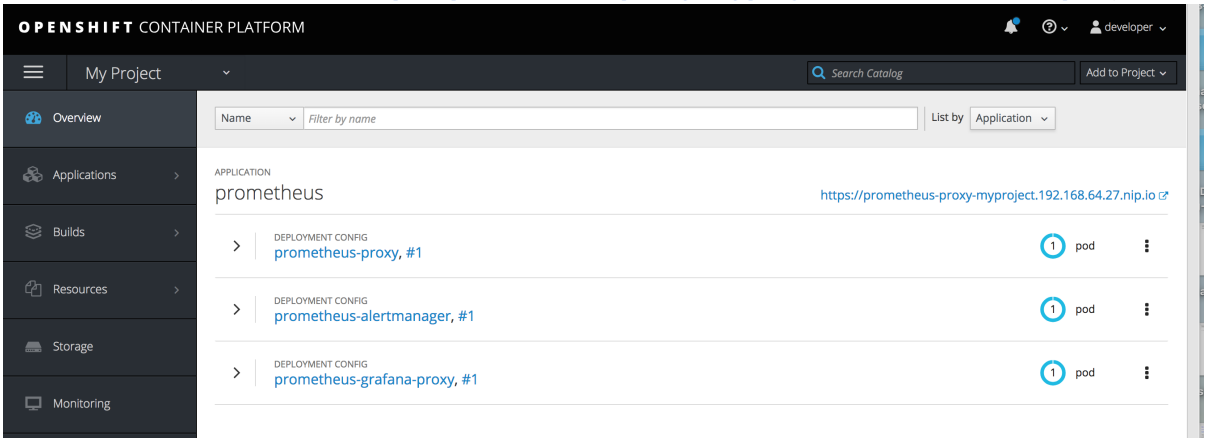


10. Click the **Continue to the project overview**link.

11. Wait until OpenShift provisions the Prometheus (APB) service.

To open the Prometheus dashboard:

1. When the Prometheus application is ready (as shown in the following screen capture), click the provided URL (for example, **https://prometheus-proxy-myproject.192.168.64.27.nip.io**):

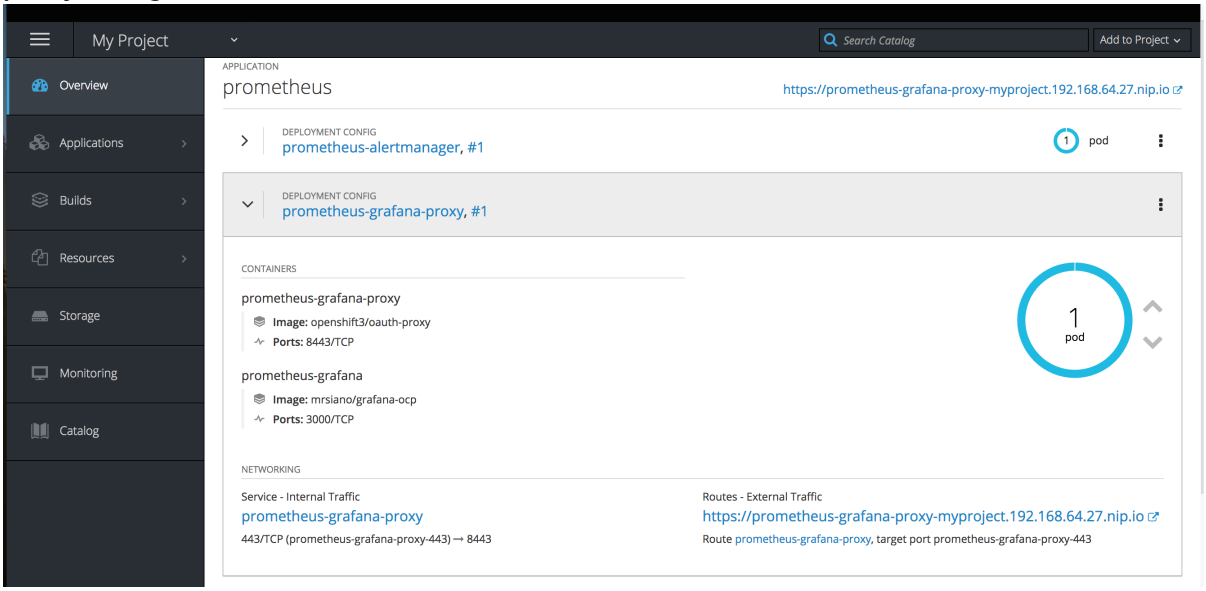


2. If you selected the **Deploy Prometheus with Oauth-Proxy sidecar**option, you must login using your OpenShift user name and password (for example, **developer**/**developer**).

3. In the **Authorize Access** page, leave the **user:info** and **user:check-access** options checked and click **Allow selected permissions**.
   The Prometheus dashboard opens.

To open the Grafana dashboard:

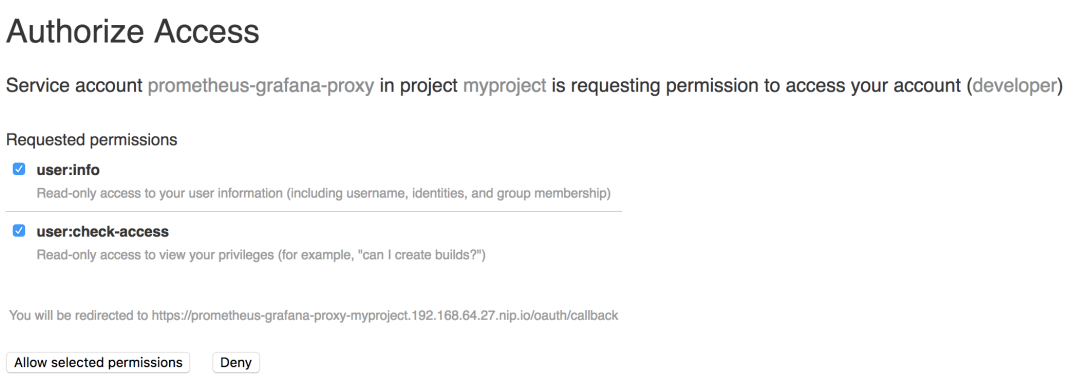
1. In the project **Overview** page of the OpenShift console, expand the **prometheus-grafana-proxy** configuration:



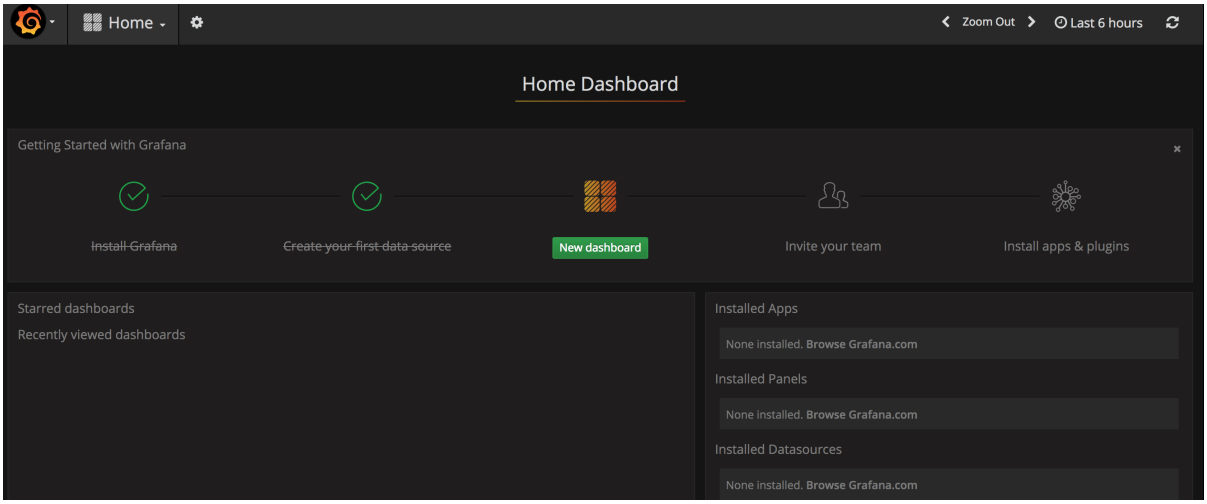
2. Click the provided URL (for example, **https://prometheus-grafana-proxy-myproject.192.168.64.27.nip.io/`**)

3. If you selected the **Deploy Prometheus with Oauth-Proxy sidecar**option, you must login using your OpenShift user name and password (for example, **developer/developer**).

4. In the **Authorize Access** page, leave the **user:info** and **user:check-access** options checked and click **Allow selected permissions**.



The Grafana Home Dashboard appears:



For information about getting started with Prometheus and Grafana, go to: https://prometheus.io/docs/prometheus/latest/getting_started/

14.2. CONFIGURING PROMETHEUS

You can set the following environment variables in OpenShift in order to configure your application's Prometheus agent:

Table 14.1. Prometheus Environment Variables

Environment Variable	Description	Default
AB_PROMETHEUS_HOST	The host address to bind.	**0.0.0.0**

Environment Variable	Description	Default
AB_PROMETHEUS_OFF	If set, disables the activation of Prometheus (echoes an empty value).	Prometheus is enabled.
AB_PROMETHEUS_PORT	The Port to use.	**9779**
AB_JMX_EXPORTER_CONFIG	Uses the file (including path) as the Prometheus configuration file.	The /opt/prometheus/prometheus-config.yml file with Camel metrics
AB_JMX_EXPORTER_OPTS	Additional options to append to the JMX exporter configuration.	Not applicable.

14.3. CONTROLLING THE METRICS THAT PROMETHEUS MONITORS AND COLLECTS

By default, Prometheus uses a configuration file (https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml) that includes all possible metrics exposed by Camel.

If you have custom metrics within your application that you want Prometheus to monitor and collect (for example, the number of orders that your application processes), you can use your own configuration file. Note that the metrics that you can identify are limited to those supplied in JMX.

To use a custom configuration file to expose JMX beans that are not covered by the default configuration, follow these steps:

1. Create a custom Prometheus configuration file. You can use the contents of the default file (**prometheus-config.yml** https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml) as a guide for the format.
   You can use any name for the custom configuration file, for example: **my-prometheus-config.yml**

2. Add your prometheus configuration file (for example, **my-prometheus-config.yml**) to your application's **src/main/fabric8-includes** directory.

3. Create a **src/main/fabric8/deployment.xml** file within your application and add an entry for the **AB_JMX_EXPORTER_CONFIG** environment variable with its value set to your configuration file. For example:

```
spec:
  template:
    spec:
      containers:
        -
          resources:
            requests:
              cpu: "0.2"
```

```
      limits:
        cpu: "1.0"
    env:
    - name: SPRING_APPLICATION_JSON
      value: '{"server":{"tomcat":{"max-threads":1}}}'
    - name: AB_JMX_EXPORTER_CONFIG
      value: "my-prometheus-config.yml"
```

This environment variable applies to your application at the pod level.

4. Rebuild and deploy your application.

14.4. GENERATING ALERTS

For an example of using Prometheus for OpenShift to generate alerts, see the Red Hat Cloud Forms *Monitoring, Alerts, and Reporting* guide:

https://access.redhat.com/documentation/en-us/red_hat_cloudforms/4.6/html/monitoring_alerts_and_reporting/integrating_prometheus_alerts