Monitoring Red Hat Integration
Abstract

Use Prometheus, an open-source monitoring system and time series database, to monitor Red Hat Integration.
# Table of Contents

**CHAPTER 1. ABOUT PROMETHEUS**  
3  

**CHAPTER 2. ACCESSING PROMETHEUS TO MONITOR FUSE APPLICATIONS ON OPENSSHIFT**  
2.1. SETTING UP PROMETHEUS  
4  
2.2. OPENSSHIFT ENVIRONMENT VARIABLES  
5  
2.3. CONTROLLING THE METRICS THAT PROMETHEUS MONITORS AND COLLECTS  
6  
2.4. GENERATING ALERTS  
7  

**CHAPTER 3. ACCESSING PROMETHEUS TO MONITOR FUSE ONLINE**  
8  
3.1. MONITORING FUSE ONLINE INFRASTRUCTURE COMPONENTS  
8  
3.2. MONITORING FUSE ONLINE INTEGRATION APPLICATIONS  
10  
3.3. CONFIGURING AN EXTERNAL PROMETHEUS INSTANCE TO MONITOR FUSE ONLINE APPLICATIONS  
12  
  
3.3.1. Configuring a Prometheus instance (with the Prometheus operator)  
12  
3.3.2. Configuring a Prometheus instance (without the Prometheus operator)  
13
CHAPTER 1. ABOUT PROMETHEUS

IMPORTANT

Prometheus is an open-source systems monitoring toolkit that you can use to monitor various aspects of Red Hat Integration products deployed in the Red Hat OpenShift environment. Red Hat support for Prometheus is limited to the setup and configuration recommendations provided in Red Hat product documentation.

Prometheus is container-native software built for storing historical data and for monitoring large, scalable systems, including Red Hat Integration. You can use Prometheus to monitor and store Fuse on OpenShift data by exposing endpoints to Prometheus format. Prometheus gathers data over an extended time, rather than just for the currently running session. 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 OpenShift

Fuse on OpenShift is the distribution of Fuse for running integration applications on OpenShift (supported on the Red Hat Enterprise Linux operating system). You can use Prometheus on an on-premise OpenShift cluster or on a single-node cluster, such as Minishift or the Red Hat Container Development Kit. For more information about Fuse on OpenShift, see Fuse on OpenShift Guide.

Fuse Online infrastructure components and Fuse Online integration applications

Fuse Online is the distribution of Fuse for non-expert integrators with a simplified workflow accessed through a browser based UI. For more information about Fuse Online, see Integrating Applications with Fuse Online.

NOTE

- Using Prometheus to monitor Fuse applications on OpenShift Online is not supported.

- Grafana is a community-supported feature. Deploying Grafana to monitor Red Hat Integration products is not supported with Red Hat production service level agreements (SLAs).

Additional resources

For information on installing and developing with Fuse on OpenShift, see the Fuse on OpenShift Guide.
CHAPTER 2. ACCESSING PROMETHEUS TO MONITOR FUSE APPLICATIONS ON OPENSIFT

2.1. SETTING UP PROMETHEUS

To set up Prometheus, install the Prometheus operator custom resource definition on the cluster and then add Prometheus to an OpenShift project that includes a Fuse application.

Prerequisites

- You have cluster admin access to the OpenShift cluster.
- You have prepared the OpenShift cluster by installing the Fuse on OpenShift images and templates as described in the Fuse on OpenShift Guide.
- You have created an OpenShift project on the cluster and added a Fuse application to it.

Procedure

1. Login to OpenShift with administrator permissions:

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

2. Install the custom resource definitions necessary for running the Prometheus operator, where `{templates-base-url}` is the location of the Fuse on OpenShift template files:

   ```bash
   $ oc create -f `{templates-base-url}`/fuse-prometheus-crd.yml
   ```

   The Prometheus operator is now available to any namespace on the cluster.

3. Install the Prometheus operator to your namespace by using the following command syntax:

   ```bash
   $ oc process -f `{templates-base-url}`/fuse-prometheus-operator.yml -p NAMESPACE=<YOUR NAMESPACE> | oc create -f -
   ```

   For example, use this command for a project (namespace) named `myproject`:

   ```bash
   oc process -f `{templates-base-url}`/fuse-prometheus-operator.yml -p NAMESPACE=myproject | oc create -f -
   ```

   **NOTE**

   The first time that you install the Prometheus operator into a namespace, it might take a few minutes for the Prometheus resource pods to start. Subsequently, if you install it to other namespaces on your cluster, the Prometheus resource pods start much faster.

4. Instruct the Prometheus operator to monitor the Fuse application in the project by using the following command syntax:

   ```bash
   $ oc process -f `{templates-base-url}`/fuse-servicemonitor.yml -p NAMESPACE=<YOUR NAMESPACE> -p FUSE_SERVICE_NAME=<YOUR FUSE SERVICE> | oc apply -f -
   ```
For example, use this command for an OpenShift project (namespace) named `myproject` that includes a Fuse application named `myfuseapp`:

```bash
oc process -f {${templates-base-url}}/fuse-servicemonitor.yml -p NAMESPACE=myproject -p FUSE_SERVICE_NAME=myfuseapp | oc apply -f -
```

5. To open the Prometheus dashboard:
   a. Login to the OpenShift console.
   b. Open the project to which you added Prometheus.
   c. In the left pane, select **Applications → Routes**.
   d. Click the Prometheus Hostname URL to open the Prometheus dashboard in a new browser tab or window.
   e. For information about getting started with Prometheus, go to: https://prometheus.io/docs/prometheus/latest/getting_started/

### 2.2. OPENSHIFT ENVIRONMENT VARIABLES

To configure your application’s Prometheus instance, you can set the OpenShift environment variables listed in Table 2.1, "Prometheus Environment Variables".
Table 2.1. Prometheus Environment Variables

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB_PROMETHEUS_HOST</td>
<td>The host address to bind</td>
<td>0.0.0.0</td>
</tr>
<tr>
<td>AB_PROMETHEUS_OFF</td>
<td>If set, disables the activation of Prometheus (echoes an empty value)</td>
<td>Prometheus is enabled.</td>
</tr>
<tr>
<td>AB_PROMETHEUS_PORT</td>
<td>The Port to use</td>
<td>9779</td>
</tr>
<tr>
<td>AB_JMX_EXPORTER_CONFIG</td>
<td>Uses the file (including path) as the Prometheus configuration file</td>
<td>The /opt/prometheus/prometheus-config.yml file with Camel metrics.</td>
</tr>
<tr>
<td>AB_JMX_EXPORTER_OPTS</td>
<td>Additional options to append to the JMX exporter configuration</td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>

Additional resources


2.3. CONTROLLING THE METRICS THAT PROMETHEUS MONITORS AND COLLECTS


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.

Procedure

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

1. Create a custom Prometheus configuration file. You can use the contents of the default file (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.

2.4. GENERATING ALERTS

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

CHAPTER 3. ACCESSING PROMETHEUS TO MONITOR FUSE ONLINE

You can use Prometheus to monitor Fuse Online infrastructure and Fuse Online applications as described in these sections:

- Section 3.1, "Monitoring Fuse Online infrastructure components"
- Section 3.2, "Monitoring Fuse Online integration applications"

You can configure an external Prometheus instance to monitor Fuse Online applications as described in this section:

- Section 3.3, “Configuring an external Prometheus instance to monitor Fuse Online applications”

3.1. MONITORING FUSE ONLINE INFRASTRUCTURE COMPONENTS

You can use Prometheus to monitor the metrics exposed by the following Fuse Online infrastructure components:

**Syndesis Server**

The `syndesis-server` component has been instrumented with Micrometer and exposes all of the JVM metrics Micrometer automatically by default. Additionally, `syndesis-server` exposes metrics about the REST API endpoints, such as request rate, error rate, and latency.

**Syndesis Meta**

The `syndesis-meta` component has been instrumented with Micrometer and exposes all of the JVM metrics Micrometer automatically by default. It also exposes metrics about its REST API endpoints.

**Syndesis DB**

Metrics for the Fuse Online Postgres database are exported using a third-party Prometheus exporter.

**Integrations**

The `integration` metrics are exported using the official JMX exporter, which exposes several JVM metrics by default. Additionally, integration metrics expose metrics that are specific to Apache Camel, such as message rate and error rate.

You can also use a Grafana dashboard to visualize the metrics gathered by Prometheus.

**Prerequisites**

- You have **cluster admin** access to the OpenShift cluster.
- Deploy Prometheus and Grafana with the Application Monitoring operator by following **these installation instructions**.

**NOTE**

Grafana is a community-supported feature. Deploying Grafana to monitor Red Hat Integration products is not supported with Red Hat production service level agreements (SLAs).

**Procedure**
1. In the Fuse Online namespace, set the `monitoring-key=middleware` label by using the following command:

```
oc label namespace <fuse-online-namespace> monitoring-key=middleware
```

2. Verify that your Fuse Online installation added the application monitoring configuration resources to the OpenShift cluster:

   a. In the OpenShift web console, go to the `application-monitoring` project and then open the `prometheus-route` URL.

   b. In the Prometheus console, go to the Status → Targets page.

      If a `Syndesis` target is listed, then Fuse Online is configured for monitoring and you can skip to Step 4.

      If a `Syndesis` target is not listed, continue to Step 3.

3. If the infrastructure resources are not on the OpenShift cluster, to enable Prometheus monitoring:

   a. Go to the Fuse Online namespace:

<table>
<thead>
<tr>
<th>OpenShift 4.x</th>
<th>OpenShift 3.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. In the OpenShift web console, go to the</td>
<td>i. Select Resources &gt; Other Resources.</td>
</tr>
<tr>
<td>Fuse Online (syndesis) project.</td>
<td>ii. From the dropdown menu, select the</td>
</tr>
<tr>
<td>ii. Select Catalog &gt; Installed Operators and</td>
<td>Syndesis resources type.</td>
</tr>
<tr>
<td>then click Fuse Online Operator.</td>
<td>iii. For the app resource, click Actions</td>
</tr>
<tr>
<td>iii. Click Syndesis CRD and then click app.</td>
<td>and then select Edit YAML to open the yaml</td>
</tr>
<tr>
<td>iv. Click Yaml to open the yaml file in the</td>
<td>file in the editor.</td>
</tr>
<tr>
<td>editor.</td>
<td></td>
</tr>
</tbody>
</table>

   b. Edit the `yaml` file to set the `Syndesis.Spec.Addons.Ops.Enable` value to `true` by adding the follow lines:

```
spec:
  addons:
    ops:
      enabled: "true"
```

c. Save the file.

d. Wait for the pod to restart.

**NOTE**

The infrastructure resources are not available immediately after you run the install commands. You might need to wait before you can see the Fuse Online (Syndesis) targets in the Prometheus Targets page.

4. To access Prometheus:
3.2. MONITORING FUSE ONLINE INTEGRATION APPLICATIONS

You can use the Application Monitoring operator to deploy Prometheus so that you can monitor Fuse Online integration applications. You can also use Grafana dashboards to visualize the metrics gathered by Prometheus.

Prerequisites

- You have **cluster admin** access to the OpenShift cluster.
- Deploy Prometheus and Grafana with the Application Monitoring operator by following these installation instructions.
NOTE

Grafana is a community-supported feature. Deploying Grafana to monitor Red Hat Integration products is not supported with Red Hat production service level agreements (SLAs).

Procedure

1. In the Fuse Online namespace, set the `monitoring-key=middleware` label by using the following command:

   ```bash
   oc label namespace <fuse-online-namespace> monitoring-key=middleware
   ```

2. Verify that your Fuse Online installation added the application monitoring configuration resources to the OpenShift cluster.
   a. In the OpenShift web console, go to the `application-monitoring` project and then open the `prometheus-route` URL.
   b. In the Prometheus console, go to the `Status > Targets` page.
      If a Syndesis target is listed, then Fuse Online has been configured for monitoring and you can skip to Step 4.
      If a Syndesis target is not listed, continue to Step 3.

3. If the infrastructure resources are not on the OpenShift cluster, to enable Prometheus monitoring:
   a. Go to the Fuse Online namespace:
      i. In the OpenShift web console, go to the Fuse Online (`syndesis`) project.
      ii. Select `Catalog > Installed Operators` and then click `Fuse Online Operator`.
      iii. Click `Syndesis CRD` and then click `app`.
      iv. Click `Yaml` to open the `.yaml` file in the editor.
   b. Edit the `.yaml` file to set the `Syndesis.Spec.Addons.Ops.Enable` value to `true` by adding the follow lines:

      ```yaml
      spec:
      addons:
      ops:
        enabled: "true"
      ```
   c. Save the file.
   d. Wait for the pod to restart.
NOTE

The infrastructure resources are not available immediately after you run the install commands. You might need to wait before you can see the Fuse Online (Syndesis) targets in the Prometheus Targets page.

4. To access Prometheus:
   a. In the OpenShift console for the project where the application monitoring operator is installed, open the list of routes.
   b. Next to the prometheus-route entry, click the hostname URL to open the Prometheus console.
   c. To view a list of the alert rules configured for Fuse Online infrastructure components, clicking the Alerts menu item.

5. To access a Grafana dashboard:
   a. In the OpenShift console for the project where the application monitoring operator is installed, open the list of routes.
   b. Next to the grafana-route entry, click the hostname URL to open the Grafana console.
   c. At the top of the Grafana console, click the dashboard selector and then select Integration - Camel.
      This dashboard displays the standard metrics exposed by Apache Camel integration applications.

3.3. CONFIGURING AN EXTERNAL PROMETHEUS INSTANCE TO MONITOR FUSE ONLINE APPLICATIONS

When you install Fuse Online on OpenShift Container Platform 3.11, a syndesis-prometheus instance is included by default. For detailed instructions on how to install Fuse Online on premise OpenShift Container Platform, see Integrating Applications with Fuse Online.

However, if you already have an existing external Prometheus instance, you can configure that external instance to also monitor Fuse Online Integration applications that are deployed on OpenShift Container Platform.

The steps depend on whether you used the Prometheus operator to install your external Prometheus instance.

- Section 3.3.1, “Configuring a Prometheus instance (with the Prometheus operator)”
- Section 3.3.2, “Configuring a Prometheus instance (without the Prometheus operator)”

3.3.1. Configuring a Prometheus instance (with the Prometheus operator)

If you installed a Prometheus instance by using the Prometheus operator, updating your external Prometheus configuration to monitor Fuse Online integration involves adding and then editing a service monitor.

Prerequisite

You installed Prometheus as described in Section 2.1, “Setting up Prometheus”.

Prerequisite

You installed Prometheus as described in Section 2.1, “Setting up Prometheus”.
Procedure

1. In a terminal window, add a service monitor to the namespace (project) in which you installed Prometheus:

   ```bash
   oc process -f {$templates-base-url}/fuse-servicemonitor.yml -p NAMESPACE=<YOUR-NAMESPACE> -p FUSE_SERVICE_NAME=fuseonline | oc create -f -
   ```

2. In the OpenShift console, open the project and then select Applications → Services.

3. Click the fuseonline service and then select Actions → Edit YAML.

4. In the Editor, change the selector section of the YAML file by replacing `app: fuseonline` with `syndesis.io/type: integration`.

5. Click Save.

The Prometheus operator updates its configuration to monitor all Fuse Online integrations.

You can now view Fuse Online integrations in the Prometheus instance. The following example shows a Prometheus instance that monitors two integrations:

![Prometheus Targets](image)

### 3.3.2. Configuring a Prometheus instance (without the Prometheus operator)

If you installed an external Prometheus instance without using the Prometheus operator, updating the configuration to monitor Fuse Online integration involves editing your Prometheus configuration file and updating the Prometheus pod configuration.

**Prerequisite**

You must have write access to the Prometheus configuration file (`prometheus-config.yml`). In the OpenShift web console, the configuration file is located in Resources → Config Maps.

**Procedure**

1. Edit your Prometheus configuration file (`prometheus-config.yml`) as follows:

   a. Set the scrape interval to 5 seconds:
global:
  scrape_interval:  5s
  evaluation_interval:  5s

b. Add a scrape config job named integration-pod with the Kubernetes service discovery configuration that configures it to scrape pods in the `${OPENShift_PROJECT}`, which is typically the syndesis namespace:

- job_name: integration-pods

  kubernetes_sd_configs:
  - role: pod
    namespaces:
      names:
        - ${OPENShift_PROJECT}

c. Add a relabel_configs section that:

- Only scrapes integration pods with the prometheus.io/scrape label set to true.
- Sets the metrics_path and address labels, that are used to scrape the JMX exporter in the integration pod, by using values from the prometheus.io/path and prometheus.io/port labels.
- Adds pod labels and annotations as Prometheus labels.
- Creates kubernetes_namespace and kubernetes_pod_name labels.

Here is an example relabel_configs section:

```
relabel_configs:
  - source_labels:
    [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
      action: keep
      regex: true
    - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
      action: replace
      target_label: __metrics_path__
      regex: (.+)
    - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
      action: replace
      regex: ([^:]+)(?:::(\d+))?(?::\d+)
      replacement: $1:$2
      target_label: __address__
      - action: labelmap
        regex: __meta_kubernetes_pod_label_(.+)
      - action: labelmap
        regex: __meta_kubernetes_pod_annotation_(syndesis.+)
    - source_labels: [__meta_kubernetes_namespace]
      action: replace
      target_label: kubernetes_namespace
    - source_labels: [__meta_kubernetes_pod_name]
      action: replace
      target_label: kubernetes_pod_name
```
d. Fuse Online integrations expose a large number of metrics from the JVM, Camel and CXF. To reduce the amount of storage needed for metrics, add the following `metric_relabel_configs` section to filter out metrics that are not displayed in the Fuse Online console.

```yaml
metric_relabel_configs:
  - source_labels: [__name__]
    regex: jmx_(.+)
    action: drop
  - source_labels: [__name__]
    regex: jvm_(.+)
    action: drop
  - source_labels: [__name__]
    regex: process_(.+)
    action: drop
  - source_labels: [type, __name__]
    separator: ':'
    regex: context:
      (org_apache_camel_ExchangesTotal|org_apache_camel_ExchangesFailed|io_syndesis.camel_StartTimestamp|io_syndesis.camel_LastExchangeCompletedTimestamp|io_syndesis.camel_LastExchangeFailureTimestamp)
    action: keep
```

**NOTE**

The last configuration line explicitly lists metrics to be added in Prometheus metrics store that are critical to the statistics shown in the Fuse Online web console. Your Prometheus instance must explicitly allow these metrics to be collected if other metrics are being filtered.

2. Update the Prometheus pod configuration to store 30 days worth of metric data as follows:

```bash
args:
  - --config.file=/etc/prometheus/prometheus.yml
  - --storage.tsdb.retention=30d
```