Red Hat Developer Hub 1.8

Monitoring and logging

Tracking performance and capturing insights with monitoring and logging tools in Red Hat Developer Hub

Last Updated: 2026-01-22

Red Hat Developer Hub 1.8 Monitoring and logging

Tracking performance and capturing insights with monitoring and logging tools in Red Hat Developer Hub

Abstract

As a Red Hat Developer Hub (RHDH) Operations or Project Manager, you can monitor performance and gather insights using Red Hat Developer Hub's monitoring and logging tools.

Table of Contents

CHAPTER 1. LOG LEVELS

Logging is an essential part of monitoring and debugging software applications. It provides insight into how the application is functioning at runtime and can help you detect and diagnose issues. By adjusting the log level, you can control the amount and type of information displayed in the logs, ranging from highly detailed diagnostic output to the most critical errors. With this flexibility, you can customize logging output to match your current requirements, whether during development, testing, or production.

You can choose from the following log levels, listed in order of decreasing verbosity:

- **debug**: Detailed information, typically useful only when troubleshooting.
- **info**: General information about the operation of the application. This is the default level.
- **warn**: Indicates potential issues or situations that might require attention.
- **error**: Indicates errors that have occurred but might not prevent the application from continuing.
- **critical**: Indicates critical errors that require immediate attention and are likely to prevent the application from functioning correctly.

You can control the verbosity of the logging by setting the log level. The log level determines the minimum severity level of events displayed in the console. For example, if the log level is set to 'info', events with a severity level of 'debug' are ignored.

To increase the log level, you can set the **LOG_LEVEL** environment variable to a higher severity level, such as 'warn' or 'error'. However, increasing the log level might not result in more output if the existing code primarily emits logs at lower severity levels, for example, 'debug' or 'info'. In such a case, adjust the logging statements within the code to use higher severity levels to see more output.

No additional steps are required beyond setting the **LOG_LEVEL** environment variable, but its effectiveness depends on the existing logging statements in the code.

CHAPTER 2. ENABLING OBSERVABILITY FOR RED HAT DEVELOPER HUB ON OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform, metrics are exposed through an HTTP service endpoint under the **/metrics** canonical name. You can create a **ServiceMonitor** custom resource (CR) to scrape metrics from a service endpoint in a user-defined project.

2.1. ENABLING METRICS MONITORING IN A RED HAT DEVELOPER HUB OPERATOR INSTALLATION ON AN OPENSHIFT CONTAINER PLATFORM CLUSTER

You can enable and view metrics for an Operator-installed Red Hat Developer Hub instance from the OpenShift Container Platform web console. Metrics are exposed through an HTTP service endpoint under the **/metrics** canonical name.

By setting the **spec.monitoring.enabled** field to **true** in your Red Hat Developer Hub custom resource (CR), you instruct the Operator to automatically create and manage the necessary **ServiceMonitor** to scrape metrics from the service endpoint.

Prerequisites

- Your OpenShift Container Platform cluster has monitoring for user-defined projects enabled.
- You have installed Red Hat Developer Hub on OpenShift Container Platform using the Red Hat Developer Hub Operator.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Use the **OpenShift CLI** (**oc**) to edit your existing Red Hat Developer Hub CR.

```
$ oc edit Backstage <instance-name>
```

2. In the CR, locate the **spec** field and add the **monitoring** configuration block.

```
spec:
  monitoring:
    enabled: true
```

3. Save the RHDH CR. The RHDH Operator detects the configuration and automatically creates the corresponding **ServiceMonitor** custom resource (CR).

NOTE

The Operator automatically configures the **ServiceMonitor** with the correct labels (**app.kubernetes.io/instance** and **app.kubernetes.io/name**) that match your Backstage CR. The **ServiceMonitor** will be named **metrics-<cr-name>**. No additional label configuration is required.

Verification

1. From the OpenShift Container Platform web console, select the **Observe** view.

2. Click the **Metrics** tab to view metrics for Red Hat Developer Hub pods.

3. From the OpenShift Container Platform web console, click **Project > Services** and verify the labels for **backstage-developer-hub**.

2.2. ENABLING METRICS MONITORING IN A HELM CHART INSTALLATION ON AN OPENSHIFT CONTAINER PLATFORM CLUSTER

You can enable and view metrics for a Red Hat Developer Hub Helm deployment from the OpenShift Container Platform web console. Metrics monitoring is enabled through configuration during a chart upgrade. After the upgrade, the Helm release generates the necessary **ServiceMonitor** resource.

Prerequisites

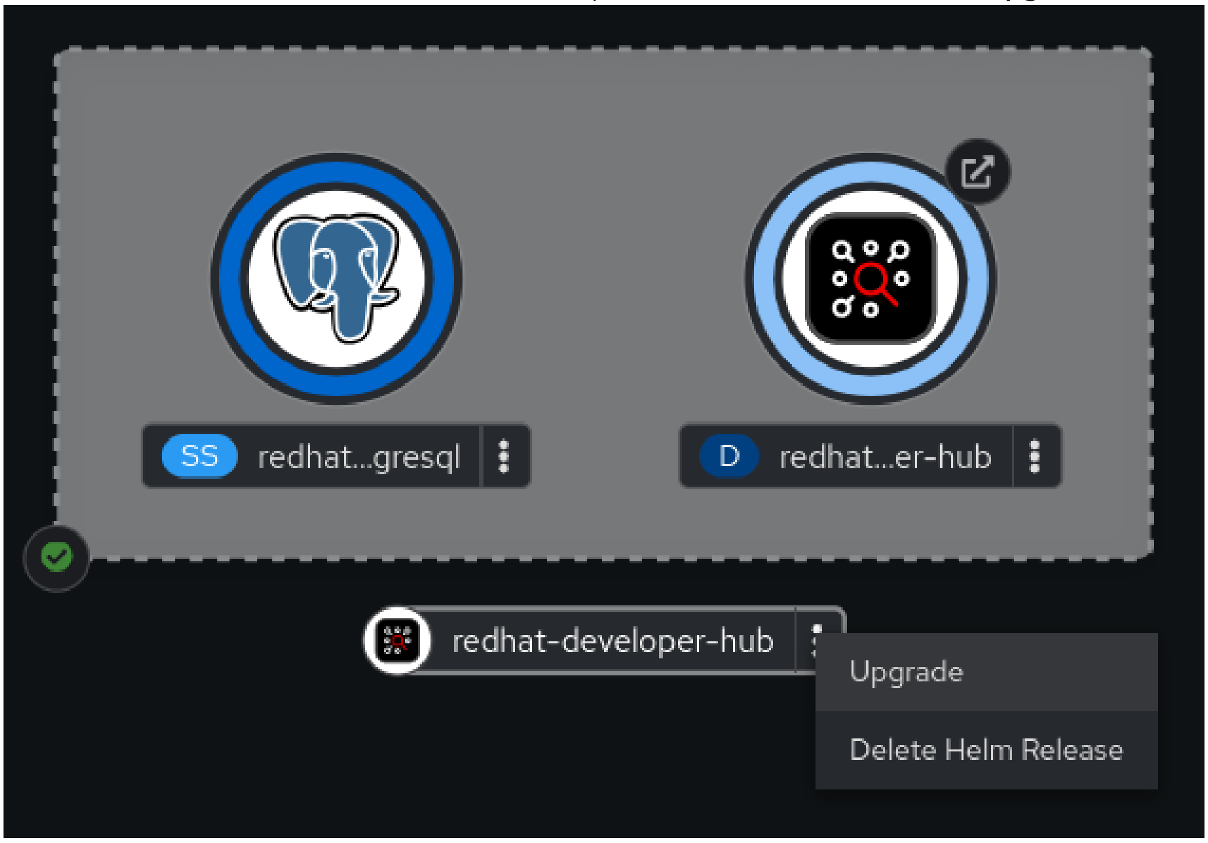
- Your OpenShift Container Platform cluster has monitoring for user-defined projects enabled.

- You have installed Red Hat Developer Hub on OpenShift Container Platform using the Helm chart.

Procedure

1. From the OpenShift Container Platform web console, select the **Topology** view.

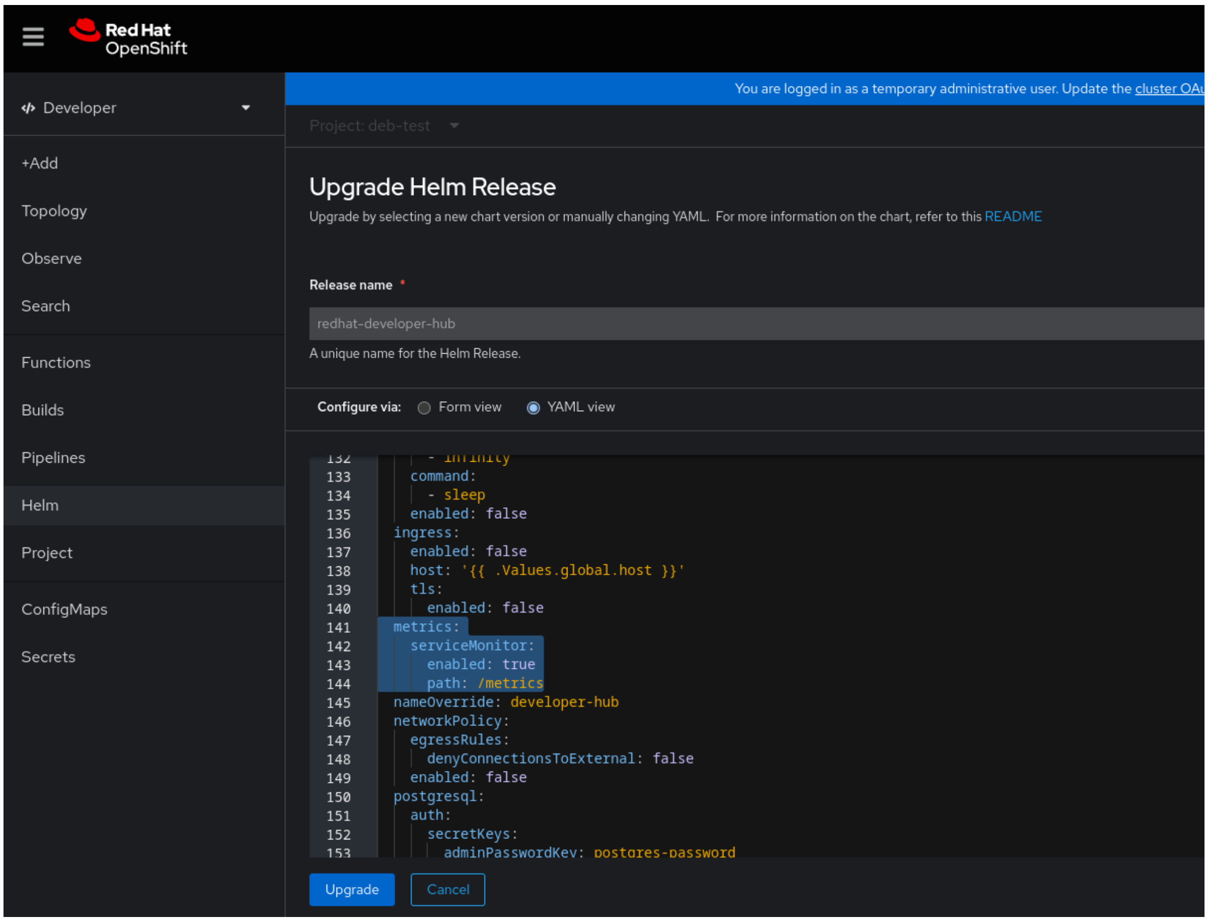
2. Click the overflow menu of the Red Hat Developer Hub Helm chart, and select **Upgrade**.



3. On the **Upgrade Helm Release** page, select the **YAML view** option in **Configure via**, then configure the **metrics** section in the YAML, as shown in the following example:

```
upstream:
```

```
# ...
  metrics:
    serviceMonitor:
      enabled: true
      path: /metrics
      port: http-metrics
# ...
```



You are logged in as a temporary administrative user. Update the cluster OAu

Upgrade Helm Release

Upgrade by selecting a new chart version or manually changing YAML. For more information on the chart, refer to this README

Release name *

redhat-developer-hub

A unique name for the Helm Release.

Configure via: ○ Form view [✓ YAML view]

```
      - infinity
    command:
      - sleep
    enabled: false
  ingress:
    enabled: false
    host: '{{ .Values.global.host }}'
    tls:
      enabled: false
  metrics:
    serviceMonitor:
      enabled: true
      path: /metrics
  nameOverride: developer-hub
  networkPolicy:
    egressRules:
      denyConnectionsToExternal: false
    enabled: false
  postgresql:
    auth:
      secretKeys:
        adminPasswordKey: postgres-password
```

4. Click **Upgrade**.

Verification

1. From the OpenShift Container Platform web console, select the **Observe** view.

2. Click the **Metrics** tab to view metrics for Red Hat Developer Hub pods.

Additional resources

- Configuring and using the monitoring stack in Red Hat OpenShift Container Platform

CHAPTER 3. MONITORING AND LOGGING RED HAT DEVELOPER HUB ON AMAZON WEB SERVICES (AWS)

You can configure Red Hat Developer Hub to use Amazon CloudWatch for real-time monitoring and Amazon Prometheus for comprehensive logging. This is convenient when hosting Developer Hub on Amazon Web Services (AWS) infrastructure.

3.1. MONITORING WITH AMAZON PROMETHEUS

You can configure Red Hat Developer Hub to use Amazon Prometheus for comprehensive logging. Amazon Prometheus extracts data from pods that have specific pod annotations.

3.1.1. Prerequisites

- You configured Prometheus for your Elastic Kubernetes Service (EKS) clusters .
- You created an Amazon managed service for the Prometheus workspace .
- You configured Prometheus to import the Developer Hub metrics .
- You ingested Prometheus metrics into the created workspace.

3.1.2. Configuring annotations for monitoring with Amazon Prometheus by using the Red Hat Developer Hub Operator

To enable logging to Amazon Prometheus, you can configure the required pod annotations by using the Red Hat Developer Hub Operator.

Procedure

1. As an administrator of the Red Hat Developer Hub Operator, edit the default configuration to add Prometheus annotations as follows:

```
# Update OPERATOR_NS accordingly
$ OPERATOR_NS=rhdh-operator
$ kubectl edit configmap backstage-default-config -n "${OPERATOR_NS}"
```

2. Find the **deployment.yaml** key in the config map and add the annotations to the **spec.template.metadata.annotations** field as follows:

```
deployment.yaml: |-
  apiVersion: apps/v1
  kind: Deployment
  # --- truncated ---
  spec:
    template:
      # --- truncated ---
      metadata:
        labels:
         rhdh.redhat.com/app:  # placeholder for 'backstage-<cr-name>'
        # --- truncated ---
        annotations:
          prometheus.io/scrape: 'true'
        prometheus.io/path: '/metrics'
        prometheus.io/port: '9464'
        prometheus.io/scheme: 'http'
  # --- truncated ---
```

3. Save your changes.

Verification

To verify if the scraping works:

1. Use **kubectl** to port-forward the Prometheus console to your local machine as follows:

   ```
   $ kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
   ```

2. Open your web browser and navigate to **http://localhost:9090** to access the Prometheus console.

3. Monitor relevant metrics, such as **process_cpu_user_seconds_total**.

3.1.3. Configuring annotations for monitoring with Amazon Prometheus by using the Red Hat Developer Hub Helm chart

To enable logging to Amazon Prometheus, you can configure the required pod annotations by using the Red Hat Developer Hub Helm chart.

Procedure

- To annotate the backstage pod for monitoring, update your **values.yaml** file as follows:

  ```
  upstream:
    backstage:
      # --- TRUNCATED ---
      podAnnotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/metrics'
        prometheus.io/port: '9464'
        prometheus.io/scheme: 'http'
  ```

Verification

To verify if the scraping works:

1. Use **kubectl** to port-forward the Prometheus console to your local machine as follows:

   ```
   $ kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
   ```

2. Open your web browser and navigate to **http://localhost:9090** to access the Prometheus console.

3. Monitor relevant metrics, such as **process_cpu_user_seconds_total**.

3.2. LOGGING WITH AMAZON CLOUDWATCH

Logging within the Red Hat Developer Hub relies on the Winston library.

3.2.1. Configuring the application log level by using the Red Hat Developer Hub Operator

You can configure the application log level by using the Red Hat Developer Hub Operator.

Procedure

- Modify the logging level by including the environment variable **LOG_LEVEL** in your custom resource as follows:

```
spec:
  # Other fields omitted
  application:
    extraEnvs:
      envs:
        - name: LOG_LEVEL
          value: debug
```

3.2.2. Configuring the application log level by using the Red Hat Developer Hub Helm chart

You can configure the application log level by using the Red Hat Developer Hub Helm chart.

Procedure

- Modify the logging level by adding the environment variable **LOG_LEVEL** to your Helm chart **values.yaml** file:

```
upstream:
  backstage:
    # --- Truncated ---
    extraEnvVars:
      - name: LOG_LEVEL
        value: debug
```

3.2.3. Retrieving logs from Amazon CloudWatch

Prerequisites

- CloudWatch Container Insights is used to capture logs and metrics for Amazon Elastic Kubernetes Service. For more information, see Logging for Amazon Elastic Kubernetes Service documentation.

- To capture the logs and metrics, install the Amazon CloudWatch Observability EKS add-on in your cluster. Following the setup of Container Insights, you can access container logs using Logs Insights or Live Tail views.

- CloudWatch names the log group where all container logs are consolidated in the following manner:

```
/aws/containerinsights/<cluster_name>/application
```

-

Procedure

- To retrieve logs from the Developer Hub instance, run a query such as:

```
fields @timestamp, @message, kubernetes.container_name
| filter kubernetes.container_name in ["install-dynamic-plugins", "backstage-backend"]
```

CHAPTER 4. MONITORING AND LOGGING WITH AZURE KUBERNETES SERVICES (AKS) IN RED HAT DEVELOPER HUB

Monitoring and logging are integral aspects of managing and maintaining Azure Kubernetes Services (AKS) in Red Hat Developer Hub. With features like Managed Prometheus Monitoring and Azure Monitor integration, administrators can efficiently monitor resource utilization, diagnose issues, and ensure the reliability of their containerized workloads.

4.1. ENABLING AZURE MONITOR METRICS

To enable managed Prometheus monitoring, use the **-enable-azure-monitor-metrics** option within either the **az aks create** or **az aks update** command, depending on whether you're creating a new cluster or updating an existing one, such as:

```
$ az aks create/update --resource-group <your-ResourceGroup> --name <your-Cluster> --enable-
azure-monitor-metrics
```

The previous command installs the metrics add-on, which gathers Prometheus metrics. Using the previous command, you can enable monitoring of Azure resources through both native Azure Monitor metrics. You can also view the results in the portal under **Monitoring → Insights**. For more information, see Monitor Azure resources with Azure Monitor .

Furthermore, metrics from both the Managed Prometheus service and Azure Monitor can be accessed through Azure Managed Grafana service. For more information, see Link a Grafana workspace section.

By default, Prometheus uses the minimum ingesting profile, which optimizes ingestion volume and sets default configurations for scrape frequency, targets, and metrics collected. The default settings can be customized through custom configuration. Azure offers various methods, including using different ConfigMaps, to provide scrape configuration and other metric add-on settings. For more information about default configuration, see Default Prometheus metrics configuration in Azure Monitor and Customize scraping of Prometheus metrics in Azure Monitor managed service for Prometheus documentation.

4.2. CONFIGURING ANNOTATIONS FOR MONITORING

You can configure the annotations for monitoring Red Hat Developer Hub specific metrics in both Helm deployment and Operator-backed deployment.

Helm deployment

To annotate the backstage pod for monitoring, update your **values.yaml** file as follows:

```
upstream:
  backstage:
    # --- TRUNCATED ---
    podAnnotations:
      prometheus.io/scrape: 'true'
      prometheus.io/path: '/metrics'
      prometheus.io/port: '9464'
      prometheus.io/scheme: 'http'
```

Operator-backed deployment

Procedure

1. As an administrator of the operator, edit the default configuration to add Prometheus annotations as follows:

```
# Update OPERATOR_NS accordingly
OPERATOR_NS=rhdh-operator
$ kubectl edit configmap backstage-default-config -n "${OPERATOR_NS}"
```

2. Find the **deployment.yaml** key in the ConfigMap and add the annotations to the **spec.template.metadata.annotations** field as follows:

```
deployment.yaml: |-
  apiVersion: apps/v1
  kind: Deployment
  # --- truncated ---
  spec:
    template:
      # --- truncated ---
      metadata:
        labels:
         rhdh.redhat.com/app:  # placeholder for 'backstage-<cr-name>'
        # --- truncated ---
        annotations:
          prometheus.io/scrape: 'true'
          prometheus.io/path: '/metrics'
          prometheus.io/port: '9464'
          prometheus.io/scheme: 'http'
  # --- truncated ---
```

3. Save your changes.

Verification

To verify if the scraping works, navigate to the corresponding Azure Monitor Workspace and view the metrics under **Monitoring → Metrics**.

4.3. VIEWING LOGS WITH AZURE KUBERNETES SERVICE (AKS)

You can access live data logs generated by Kubernetes objects and collect log data in Container Insights within AKS.

Prerequisites

- You have deployed Developer Hub on AKS.

For more information, see Installing Red Hat Developer Hub on Microsoft Azure Kubernetes Service .

Procedure

View live logs from your Developer Hub instance

1. Navigate to the Azure Portal.

2. Search for the resource group **<your-ResourceGroup>** and locate your AKS cluster **<your-Cluster>**.

3. Select **Kubernetes resources → Workloads**from the menu.

4. Select the **<your-rhdh-cr>-developer-hub** (in case of Helm Chart installation) or **<your-rhdh-cr>-backstage** (in case of Operator-backed installation) deployment.

5. Click **Live Logs** in the left menu.

6. Select the pod.

   NOTE

   There must be only single pod.

Live log data is collected and displayed.

View real-time log data from the Container Engine

1. Navigate to the Azure Portal.

2. Search for the resource group **<your-ResourceGroup>** and locate your AKS cluster **<your-Cluster>**.

3. Select **Monitoring → Insights** from the menu.

4. Go to the **Containers** tab.

5. Find the backend-backstage container and click it to view real-time log data as it's generated by the Container Engine.