



Red Hat build of Cryostat 2

Getting started with Cryostat

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Red Hat build of Cryostat is a Red Hat offering on OpenShift Container Platform. The Getting started with Cryostat guide provides an overview of this product and explains how to install the software and start using it.

Table of Contents

PREFACE	3
MAKING OPEN SOURCE MORE INCLUSIVE	4
CHAPTER 1. OVERVIEW OF CRYOSTAT	5
CHAPTER 2. INSTALLING CRYOSTAT ON RED HAT OPENSIFT BY USING A RED HAT BUILD OF CRYOSTAT OPERATOR	6
2.1. CREATING CRYOSTAT ON MULTIPLE NAMESPACES	10
CHAPTER 3. CONFIGURING JAVA APPLICATIONS	15
Cryostat agent	15
Remote Java Management Extensions (JMX) connections	17
Cryostat agent and JMX hybrid	17
3.1. CONFIGURING APPLICATIONS BY USING THE CRYOSTAT AGENT	17
3.2. CONFIGURING APPLICATIONS BY USING JMX CONNECTIONS	21
3.3. CONFIGURING APPLICATIONS BY USING THE CRYOSTAT AGENT AND JMX CONNECTIONS	24
CHAPTER 4. CREATING A JFR RECORDING WITH CRYOSTAT	27
4.1. CREATING A JFR RECORDING IN THE CRYOSTAT WEB CONSOLE	27
4.2. CREATING SNAPSHOTS FROM AN ACTIVE RECORDING	31
4.3. LABELS FOR JFR RECORDINGS	33
4.3.1. Adding labels to JFR recordings	33
4.3.2. Editing a label for your JFR recording	35

PREFACE

The Red Hat build of Cryostat is a container-native implementation of JDK Flight Recorder (JFR) that you can use to securely monitor the Java Virtual Machine (JVM) performance in workloads that run on an OpenShift Container Platform cluster. You can use Cryostat 2.4 to start, stop, retrieve, archive, import, and export JFR data for JVMs inside your containerized applications by using a web console or an HTTP API.

Depending on your use case, you can store and analyze your recordings directly on your Red Hat OpenShift cluster by using the built-in tools that Cryostat provides or you can export recordings to an external monitoring application to perform a more in-depth analysis of your recorded data.



IMPORTANT

Red Hat build of Cryostat is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in 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 about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

CHAPTER 1. OVERVIEW OF CRYOSTAT

Cryostat is a container-native Java application based on JDK Flight Recorder (JFR) that you can use to monitor Java Virtual Machine (JVM) performance for containerized workloads that run on a Red Hat OpenShift cluster.

You can deploy Cryostat in a container in a Red Hat OpenShift project that hosts your containerized Java applications. You can create JVM targets that correspond to the JVM instances that you use to run your containerized workload. You can connect Cryostat to the JVM targets to record and analyze data about heap and non-heap memory usage, thread count, garbage collection, and other performance metrics for each JVM target.

You can use the tools that are included with Cryostat to monitor the performance of your JVMs in real time, capture JDK Flight Recorder (JFR) recordings and snapshots, generate Automated Analysis reports, and visualize your recorded performance data by using a Grafana dashboard.

The Cryostat web console and HTTP API provides a way to analyze your JVM performance data inside the container without having to rely on an external monitoring application. However, you can also export your recordings from Cryostat into an external instance of JDK Mission Control (JMC) when you need to perform a deeper analysis of your data outside of a cluster environment.

Cryostat supports role-based access control (RBAC) as a standard feature of OpenShift Container Platform. You can configure different levels of authorization for each user role to ensure the privacy and integrity of your Flight Recording data.

You can install Cryostat inside a Red Hat OpenShift project by using Operator Lifecycle Manager (OLM).

You can also download the latest Cryostat component images from the Red Hat Ecosystem Catalog. The following container images exist for Cryostat 2.4 on the Red Hat Ecosystem Catalog:

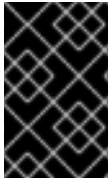
- Cryostat
- Red Hat build of Cryostat Operator
- Red Hat build of Cryostat Operator bundle
- Cryostat reports
- Cryostat Grafana dashboard
- JFR data source

Additional resources

- [Operator Lifecycle Manager \(OLM\) \(OpenShift Container Platform\)](#)
- [Container images \(Red Hat Ecosystem Catalog\)](#)

CHAPTER 2. INSTALLING CRYOSTAT ON RED HAT OPENSIFT BY USING A RED HAT BUILD OF CRYOSTAT OPERATOR

You can use the Operator Lifecycle Manager (OLM) to install the Red Hat build of Cryostat Operator in a project on your Red Hat OpenShift cluster. You can use the Red Hat build of Cryostat Operator to create single namespace or multi-namespace Cryostat instances. You can control these instances by using a GUI that is accessible from the Red Hat OpenShift web console.



IMPORTANT

If you need to upgrade your Red Hat build of Cryostat Operator subscription from Cryostat 2.0 to Cryostat 2.4, you must change the update channel from **stable-2.0** to **stable**.

Prerequisites

- Created an OpenShift Container Platform 4.11 or later cluster.
- Created a Red Hat OpenShift user account with permissions to install Red Hat build of Cryostat Operator in a project.
- Installed Operator Lifecycle Manager (OLM) on your cluster.
- Installed cert-manager with the cert-manager Operator for Red Hat OpenShift.
 - If you are using OpenShift Container Platform 4.11 or later, you can install the cert-manager Operator for Red Hat OpenShift. For more information, see [cert-manager Operator for Red Hat OpenShift \(OpenShift Container Platform\)](#).
- Logged in to Red Hat OpenShift by using the Red Hat OpenShift web console.

Procedure

1. In your browser, navigate to **Home > Projects** by using the web console.
2. Select the name of the project in which you want to install the Red Hat build of Cryostat Operator.
3. Install the Red Hat build of Cryostat Operator:
 - a. In the navigation menu of your web console, navigate to **Operators > OperatorHub**.
 - b. Select the **Red Hat build of Cryostat Operator** from the list. You can use the search box in the upper part of the screen to find the Red Hat build of Cryostat Operator.
 - c. To install the Red Hat build of Cryostat Operator in your project, click **Install**. The Red Hat OpenShift web console prompts you to create a Cryostat custom resource (CR).



NOTE

If you are installing a Cryostat instance that is enabled for multiple namespaces, in the **Installation mode** area, click the **All namespaces on the cluster (default)** radio button.

You can create the CR either manually or automatically. If you want to create the CR manually, see step 4. If you want to create the CR automatically, see step 5.

4. If you want to create the CR manually, complete the following steps:

- i. Navigate to **Operators > Installed Operators** by using the web console and select **Red Hat build of Cryostat Operator** from the list of installed operators:


Figure 2.1. Viewing the Red Hat build of Cryostat operator in the list of installed operators

Project: cryostat-test ▾

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#) or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

Name ▾ Search by name... /

Name	Managed Namespaces	Status	Last updated	Provided APIs
 Red Hat build of Cryostat 2.4.0 provided by Red Hat	All Namespaces	✔ Succeeded Up to date	🕒 Dec 5, 2023, 1:26 PM	Cluster Cryostat Cryostat

- ii. Click the **Details** tab.
- iii. To create a single-namespace Cryostat instance, go to the **Provided APIs** section. Then, under **Cryostat**, click **Create instance**.




NOTE

If you want to create a Cryostat instance that is enabled for multiple namespaces, in the **Provided APIs** section, select **Cluster Cryostat** and click **Create instance**. The **Cluster Cryostat** API has configuration options that control the deployment of the Cryostat application and its related components. For more information, see [Creating Cryostat on multiple namespaces](#).

Figure 2.2. Selecting the Cryostat API that is provided by the Red Hat build of Cryostat Operator


Project: cryostat-test ▾

[Installed Operators](#) > [Operator details](#)

 **Red Hat build of Cryostat**
2.4.0 provided by Red Hat Actions ▾


[Details](#) [YAML](#) [Subscription](#) [Events](#) [All instances](#) [Cluster Cryostat](#) [Cryostat](#)

Provided APIs

 **Cluster Cryostat**

ClusterCryostat allows you to install Cryostat for multiple namespaces or cluster-wide. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be...

[Create instance](#)

 **Cryostat**

Cryostat allows you to install Cryostat for a single namespace. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be created to instruct the...

[Create instance](#)

Provider
Red Hat

Created at
🕒 Nov 24, 2023, 9:42 PM

Links
Upstream Project
<https://github.com/cryostat/cryostat>

Website
<https://cryostat.io/>

Maintainers
The Cryostat Authors
cryostat-development@googlegroups.com

- iv. Click either the **Form view** radio button or the **YAML view** radio button. If you want to enter your information in the YAML configuration file, click **YAML view**.
- v. Specify a name for the instance of Crio that you want to create.
- vi. *Optional:* In the Labels field, specify a label or annotation for the Operand workload you want to deploy.
You can also specify additional configuration options for your deployment:

Figure 2.3. Creating an instance of Crio by using a form in the web console

Project: criostat-test

Create Crio

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: Form view YAML view

Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.

Name *
criostat-sample

Labels
app=frontend

Minimal Deployment *
 false
Deploy a pared-down Crio instance with no Grafana Dashboard or JFR Data Source.

Enable cert-manager Integration
 true
Use cert-manager to secure in-cluster communication between Crio components. Requires cert-manager to be installed.

Crio
provided by Red Hat

Crio allows you to install Crio for a single namespace. It contains configuration options for controlling the Deployment of the Crio application and its related components. A ClusterCrio or Crio instance must be created to instruct the operator to deploy the Crio application.

Alternatively, you can use a YAML template to create your instance and specify additional configuration options instead of using the form:

Figure 2.4. Creating an instance of Crio by using a YAML template in the web console

Project: criostat-test

Create Crio

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via: Form view YAML view

```

1 apiVersion: operator.crio.io/v1beta1
2 kind: Crio
3 metadata:
4   name: criostat-sample
5   namespace: criostat-test
6 spec:
7   storageOptions:
8     pvc:
9       annotations: {}
10      labels: {}
11     spec: {}
12   trustedCertSecrets: []
13   reportOptions:
14     replicas: 0
15   eventTemplates: []
16   enableCertManager: true
17   minimal: false
18

```

Crio
Schema

Crio allows you to install Crio for a single namespace. It contains configuration options for controlling the Deployment of the Crio application and its related components. A ClusterCrio or Crio instance must be created to instruct the operator to deploy the Crio application.

- apiVersion**
string
APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>
- kind**
string
Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- metadata**

Create **Cancel** **Download**

5. If you want to create the CR by using the automatic prompt option, follow the prompt's instructions and then complete the following steps:
 - i. Click either the **Form view** radio button or the **YAML view** radio button. If you want to enter your information in the YAML configuration file, click **YAML view**.
 - ii. Specify a name for the instance of Crio that you want to create.

- iii. *Optional*: In the Labels field, specify a label or annotation for the Operand workload you want to deploy.

You can also specify additional configuration options for your deployment:

Figure 2.5. Creating an instance of Cryostat by using a form in the web console

The screenshot shows the 'Create Cryostat' form in the web console. At the top, it says 'Project: cryostat-test'. Below that, the title is 'Create Cryostat' with a subtitle 'Create by completing the form. Default values may be provided by the Operator authors.' There are two tabs: 'Form view' (selected) and 'YAML view'. A blue note box states: 'Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.' The form has three main sections:

- Name**: A text input field containing 'cryostat-sample'.
- Labels**: A text area containing 'app=frontend'.
- Minimal Deployment**: A toggle switch set to 'false' with the text 'Deploy a pared-down Cryostat instance with no Grafana Dashboard or JFR Data Source.'
- Enable cert-manager Integration**: A toggle switch set to 'true' with the text 'Use cert-manager to secure in-cluster communication between Cryostat components. Requires cert-manager to be installed.'

 On the right side, there is a 'Cryostat' section with a Red Hat logo and text: 'Cryostat allows you to install Cryostat for a single namespace. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be created to instruct the operator to deploy the Cryostat application.'

Alternatively, you can use a YAML template to create your instance and specify additional configuration options instead of using the form:

Figure 2.6. Creating an instance of Cryostat by using a YAML template in the web console

The screenshot shows the 'Create Cryostat' form in the web console, switched to 'YAML view'. At the top, it says 'Project: cryostat-test'. Below that, the title is 'Create Cryostat' with a subtitle 'Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.' There are two tabs: 'Form view' and 'YAML view' (selected). A code editor displays the following YAML:


```

1 apiVersion: operator.cryostat.io/v1beta1
2 kind: Cryostat
3 metadata:
4   name: cryostat-sample
5   namespace: cryostat-test
6 specs:
7   storageOptions:
8     pvc:
9       annotations: {}
10      labels: {}
11      spec: {}
12   trustedCertSecrets: []
13   reportOptions:
14     replicas: 0
15   eventTemplates: []
16   enableCertManager: true
17   minimal: false
18
  
```

 Below the code editor are 'Create', 'Cancel', and 'Download' buttons. On the right side, there is a 'Cryostat' section with a 'Schema' link and text: 'Cryostat allows you to install Cryostat for a single namespace. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be created to instruct the operator to deploy the Cryostat application.' Below this, there is a list of schema properties:

- apiVersion**: string. APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>
- kind**: string. Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- metadata**

6. To start the creation process for your Cryostat instance, click **Create**.
You must wait for all resources of your Cryostat instance to be ready before you can access it.

Verification

1. In the navigation menu of the web console, click **Operators**, then click **Installed Operators**.
2. From the table of installed operators, select **Red Hat build of Cryostat Operator**.
3. Select the **Cryostat** tab.
Your Cryostat instance opens in the table of instances and lists the following conditions:

- **TLSSetupComplete** is set to **true**.
- **MainDeploymentAvailable** is set to **true**.
- Optional: If you enabled the reports generator service then **ReportsDeploymentAvailable** is shown and set to **true**.

Figure 2.7. Example of conditions set to True under the Status column for a Cryostat instance on OpenShift

Project: cryostat-test ▾

Installed Operators > Operator details

Red Hat build of Cryostat
2.4.0 provided by Red Hat

Actions ▾

Details YAML Subscription Events All instances Cluster Cryostat Cryostat

Cryostats Show operands in: All namespaces Current namespace only Create Cryostat

Name ▾ Search by name... /

Name	Kind	Namespace	Status	Labels	Last updated
cryostat-sample	Cryostat	cryostat-test	Conditions: TLSSetupComplete, MainDeploymentAvailable, MainDeploymentProgressing, ReportsDeploymentAvailable, ReportsDeploymentProgressing	No labels	Nov 24, 2023, 9:43 PM

4. *Optional:* Select your Cryostat instance from the **Cryostat** table. Go to the **Cryostat Conditions** table, where you can see more information for each condition.

Figure 2.8. Example of a Cryostat Conditions table that lists each condition and its criteria

Type	Status	Updated	Reason	Message
TLSSetupComplete	True	Nov 24, 2023, 9:43 PM	AllCertificatesReady	All certificates for Cryostat components are ready.
MainDeploymentAvailable	True	Dec 7, 2023, 5:00 AM	MinimumReplicasAvailable	Deployment has minimum availability.
MainDeploymentProgressing	True	Nov 24, 2023, 9:43 PM	NewReplicaSetAvailable	ReplicaSet "cryostat-sample-7dd96f95cd" has successfully progressed.
ReportsDeploymentAvailable	True	Nov 24, 2023, 9:43 PM	MinimumReplicasAvailable	Deployment has minimum availability.
ReportsDeploymentProgressing	True	Nov 24, 2023, 9:43 PM	NewReplicaSetAvailable	ReplicaSet "cryostat-sample-reports-71bbf6d95f" has successfully progressed.

Next Steps

- [Accessing Cryostat by using the web console](#)

2.1. CREATING CRYOSTAT ON MULTIPLE NAMESPACES

The Red Hat build of Cryostat Operator provides the **Cluster Cryostat** API, which you can use to create Cryostat instances that work across multiple namespaces.

Prerequisites

- Created an OpenShift Container Platform 4.11 or later cluster.
- Created a Red Hat OpenShift user account with permissions to install Red Hat build of Cryostat Operator in a project.
- Installed the Operator Lifecycle Manager (OLM) on your cluster.
- Installed cert-manager by using the cert-manager Operator for Red Hat OpenShift.
 - If you are using OpenShift Container Platform 4.11 or later, you can install the cert-manager Operator for Red Hat OpenShift. For more information, see [cert-manager Operator for](#)

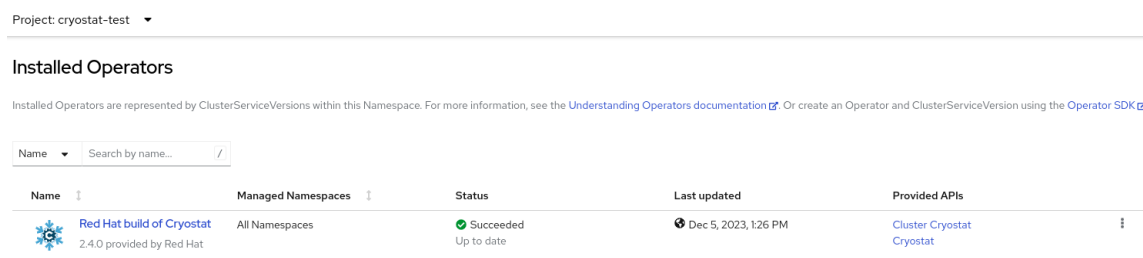
Operator for Red Hat OpenShift. For more information, see [cert-manager Operator for Red Hat OpenShift \(OpenShift Container Platform\)](#).

- Logged in to Red Hat OpenShift by using the Red Hat OpenShift web console.

Procedure

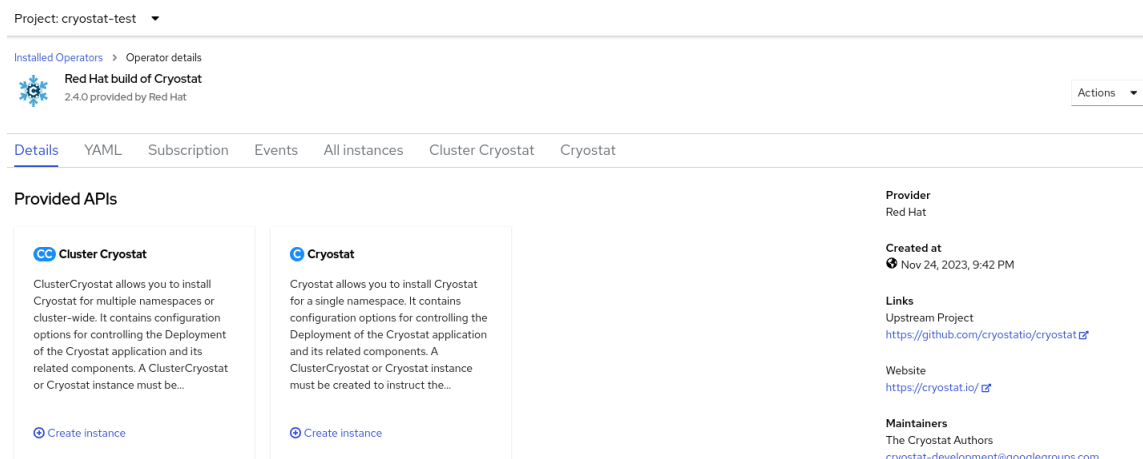
1. In your browser, navigate to **Home > Projects** by using the web console.
2. Select the name of the project where you want to install the Red Hat build of Cryostat Operator.
3. Install the Red Hat build of Cryostat Operator.
The Red Hat OpenShift web console prompts you to create a Cryostat custom resource (CR).
4. To create the CR, complete the following steps:
 - i. Navigate to **Operators > Installed Operators** by using the web console and select **Red Hat build of Cryostat Operator** from the list of installed operators:

Figure 2.9. Viewing the Red Hat build of Cryostat Operator in the list of installed operators



- ii. Click the **Details** tab.
- iii. To create a multi-namespace instance of Cryostat, go to the **Provided APIs** section. Then, under **Cluster Cryostat**, click **Create instance**.

Figure 2.10. Selecting the Cluster Cryostat API that is provided by the Red Hat build of Cryostat Operator



- iv. Click either the **Form view** radio button or the **YAML view** radio button. If you want to enter your information in the YAML configuration file, click **YAML view**.
- v. Specify a unique name for the Cluster Cryostat instance that you want to create.



NOTE

Ensure that the name you specify for your Cluster Cryostat instance is unique and does not conflict with the name of any single-namespace Cryostat instances that might already be created in the install namespace or target namespaces of the Cluster Cryostat instance.

- vi. *Optional:* In the Labels field, specify a label or annotation for the Operand workload you want to deploy.

You can also specify additional configuration options for your deployment:

Figure 2.11. Creating a Cluster Cryostat instance by using a form in the web console

Project: cryostat-test ▾

Create ClusterCryostat

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: Form view YAML view

Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.

Name *

Labels

Install Namespace *

Namespace where Cryostat should be installed. On multi-tenant clusters, we strongly suggest installing Cryostat into its own namespace.

Minimal Deployment *

 false

Deploy a pared-down Cryostat instance with no Grafana Dashboard or JFR Data Source.

Cluster Cryostat
provided by Red Hat

ClusterCryostat allows you to install Cryostat for multiple namespaces or cluster-wide. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be created to instruct the operator to deploy the Cryostat application.

- vii. In the **Install Namespace** field, select a namespace where you want to install this instance of Cryostat.

TIP

The Red Hat build of Cryostat Operator uses a larger set of permissions compared to the Cryostat application, and Cryostat might have more permissions than your target workloads. Therefore, for optimal security, install the Cryostat instance into a different namespace from where the Red Hat build of Cryostat Operator operator is installed and from where your target workloads are located.

- viii. In the **Target Namespaces** field, select namespaces whose workloads you want to permit this instance of Cryostat to access and work with. Optionally, you can select the same namespace where you installed Cryostat or you can choose a different namespace. To add additional namespaces, click **+Add Target Namespace**.



IMPORTANT

Users who can access the Cryostat instance have access to all target applications in any namespace that is visible to that Cryostat instance. Therefore, when you deploy a multi-namespace Cryostat instance, you must consider which namespaces to select for monitoring, which namespace to install Cryostat into, and which users you want to grant access to.

Alternatively, you can use a YAML template to create your instance and specify additional configuration options instead of using the form:

Figure 2.12. Creating a Cluster Cryostat instance by using a YAML template in the web console

Project: cryostat-test

Create ClusterCryostat

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via: Form view YAML view

```

1 apiVersion: operator.cryostat.io/v1beta1
2 kind: ClusterCryostat
3 metadata:
4   name: clustercryostat-sample
5 spec:
6   storageOptions:
7     pvc:
8       annotations: {}
9       labels: {}
10      spec: {}
11   trustedCertSecrets: []
12   reportOptions:
13     replicas: 0
14   eventTemplates: []
15   enableCertManager: true
16   minimal: false
17

```

ClusterCryostat Schema

ClusterCryostat allows you to install Cryostat for multiple namespaces or cluster-wide. It contains configuration options for controlling the Deployment of the Cryostat application and its related components. A ClusterCryostat or Cryostat instance must be created to instruct the operator to deploy the Cryostat application.

- apiVersion**
string
APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>
- kind**
string
Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- metadata**
object

Buttons: Create, Cancel, Download

5. Click **Create** to start the creation process for your Cryostat multi-namespace instance. You must wait for all resources of your Cluster Cryostat instance to be ready before you can access the instance.

Verification

1. In the navigation menu of the web console, navigate to **Operators > OperatorHub**.
2. From the table of installed operators, select **Red Hat build of Cryostat Operator**.
3. Click the **Cluster Cryostat** tab.
Your Cryostat instance opens in the table of instances and lists the following conditions:
 - **TLSSetupComplete** is set to **true**.
 - **MainDeploymentAvailable** is set to **true**.
 - Optional: If you enabled the reports generator service, **ReportsDeploymentAvailable** is shown and set to **true**.

Figure 2.13. Example of conditions set to True under the Status column for a Cluster Cryostat instance on OpenShift

Project: cryostat-test

Installed Operators > Operator details

Red Hat build of Cryostat
2.4.0 provided by Red Hat

Details | YAML | Subscription | Events | All instances | **Cluster Cryostat** | Cryostat

ClusterCryostats

Search by name... /

Name	Kind	Status	Labels	Last updated
clustercryostat-sample	ClusterCryostat	Conditions: TLSSetupComplete, MainDeploymentAvailable, MainDeploymentProgressing	No labels	Dec 7, 2023, 3:21 PM

Buttons: Create ClusterCryostat

4. *Optional:* From the **Cluster Cryostat** table, select your Cryostat instance. Go to the **Conditions** table to view more information for each condition.

Figure 2.14. Example of a CrioStat Conditions table that lists each condition and its criteria

CrioStat Conditions

Type	Status	Updated	Reason	Message
TLSSetupComplete	True	🕒 Nov 24, 2023, 9:43 PM	AllCertificatesReady	All certificates for CrioStat components are ready.
MainDeploymentAvailable	True	🕒 Dec 7, 2023, 5:00 AM	MinimumReplicasAvailable	Deployment has minimum availability.
MainDeploymentProgressing	True	🕒 Nov 24, 2023, 9:43 PM	NewReplicaSetAvailable	ReplicaSet "cryostat-sample-7dd96f95cd" has successfully progressed.
ReportsDeploymentAvailable	True	🕒 Nov 24, 2023, 9:43 PM	MinimumReplicasAvailable	Deployment has minimum availability.
ReportsDeploymentProgressing	True	🕒 Nov 24, 2023, 9:43 PM	NewReplicaSetAvailable	ReplicaSet "cryostat-sample-reports-7fbbfd95f" has successfully progressed.

Additional resources

- [Best practices for setting up CrioStat in different cluster configurations](#) (Red Hat Knowledgebase)
- [Accessing CrioStat by using the web console](#)

CHAPTER 3. CONFIGURING JAVA APPLICATIONS

To enable Cryostat to gather, store, and analyze Java Flight Recorder (JFR) data about target applications that run on Java Virtual Machine (JVM)s, you must configure the applications so that Cryostat can detect and connect to them.

You can configure the applications in any of the following ways:

- By using the Cryostat agent component for detection and connectivity, which is implemented as a Java Instrumentation Agent and acts as a plug-in for applications that run on the JVM
- By configuring the applications to allow Java Management Extensions (JMX) connections and using an OpenShift Service for detection and JMX for connectivity
- By using the Cryostat agent for detection and JMX for connectivity

Cryostat agent

From Red Hat build of Cryostat 2.4 onward, the Cryostat agent provides an HTTP API that the Cryostat server can use as an alternative to an application's JMX port. By attaching a properly configured Cryostat agent to the workload applications that you deploy, you can use the full Cryostat feature set without any need for the target applications to expose a JMX port.



NOTE

Before Red Hat build of Cryostat 2.4, the Cryostat agent provided a read-only HTTP API that supported a limited set of JFR operations only.

The Cryostat agent's HTTP API can offer the following benefits compared to a JMX port:

- Greater security due to the reduced API surface area
- Deployment flexibility due to the Cryostat agent's dual role as a Cryostat discovery plug-in

If the Cryostat agent detects that JMX is also configured on your application, the agent publishes itself to the Cryostat server with both agent HTTP API definitions and JMX URL definitions. In this situation, you can use whichever configuration option you prefer.

To use the Cryostat agent with your workload application, you must configure the application to pass the **-javaagent** JVM flag with the path to the Cryostat agent's JAR file (for example, **-javaagent:/deployment/app/lib/cryostat-agent.jar**). This setting enables your workload application's JVM to load and initialize the Cryostat agent at startup. Once the Cryostat agent's basic initialization is complete, your workload application's normal startup process begins as usual.

Options for including the agent's JAR file into your workload application

You can include the Cryostat agent's JAR file into the workload application in different ways:

- The simplest option is to add the JAR file to your application's dependencies in the **pom.xml** or **build.gradle** file. Your build tool (Maven or Gradle) downloads the JAR file for inclusion in your application build output.
- You can use a Maven plug-in, such as the **maven-dependency-plugin**, to provide more fine-grained control over the download and inclusion of the JAR file in the application build output.
- You can create a PersistentVolume storage volume that contains the JAR file. Then reconfigure your application's **Deployment/DeploymentConfig** to mount the PersistentVolume and use -

javaagent:/path/to/persistentvolume/cryostat-agent.jar. The exact way to accomplish this task depends on the type of PersistentVolume provider you enabled in your OpenShift cluster.

Once the Cryostat agent is successfully added to your application container and loaded, your application's **stdout** and **console** logs start displaying log messages from the Cryostat agent.

Agent configuration properties


You can specify configuration properties for the Cryostat agent in either of two ways:

- Use JVM system property flags on the application (for example, **-Dcryostat.agent.api.writes-enabled=true**).
- Use environment variables by making all letters upper case and replacing any punctuation with underscores (for example, **CRYOSTAT_AGENT_API_WRITES_ENABLED=true**).

You must configure the following properties to enable the Cryostat agent to operate successfully:

cryostat.agent.baseuri	This specifies the URL location of the Cryostat server back end that the Cryostat agent advertises itself to (that is, the internal OpenShift Service object) such as https://my-cryostat.my-namespace.svc.cluster.local .
cryostat.agent.callback	This specifies the URL location of the Cryostat agent instance or application itself. Cryostat uses this URL to perform health checks and request data from the agent. You can use the OpenShift/Kubernetes Downward API to determine this dynamically. For more information, see the Kubernetes Downward API documentation on status.podIP .

Depending on your setup requirements, you can also configure the following agent properties:

cryostat.agent.api.writes-enabled	<p>This indicates whether the Cryostat agent allows write operations. This is set to false by default. If you want the Cryostat agent to accept requests to start, stop, or delete JFR flight recordings, you must set this property to true.</p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>NOTE</p> <p>Even if this property is set to false, the agent can still fulfill requests to list flight recordings or download individual recording files.</p> </div> </div>
cryostat.agent.webserver.port	This specifies the HTTP port number that the agent uses to bind its HTTP API (by default, 9977). If this conflicts with an existing port that your application or another tooling agent uses, you must specify a different port number.

cryostat.agent.app.name	This specifies a label for identifying which application this Cryostat agent instance is attached to (by default, cryostat-agent). You could use the Downward API metadata.name or metadata.labels['app'] fields for this. For more information, see the Kubernetes Downward API documentation Kubernetes Downward API documentation.
--------------------------------	---

Remote Java Management Extensions (JMX) connections

JMX is a standard feature on a JVM with which you can monitor and manage target applications that run on the JVM. For Cryostat to use JMX, you must enable and configure JMX when you start the JVM because Cryostat requires the target applications to expose a JMX port.

Cryostat communicates with the target applications over this JMX port to start and stop JFR recordings and to pull JFR data over the network, enabling Cryostat to store and analyze this JFR data. Remote monitoring requires security to ensure that unauthorized persons cannot access application. Cryostat prompts you to enter your credentials before Cryostat can access any of the application's JFR recordings.

Cryostat agent and JMX hybrid

You can configure your target applications to use a hybrid approach where you use both the Cryostat agent and JMX. With this approach, you use the Cryostat agent to detect the target applications and JMX to expose the JFR data to Cryostat, which allows for more flexibility.

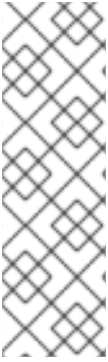
For example, you can use the agent to detect the applications without needing to depend on specific port numbers and also use the JMX connections to start and stop JFR flight recordings on demand.

3.1. CONFIGURING APPLICATIONS BY USING THE CRYOSTAT AGENT

You can use the Cryostat agent, implemented as a Java Instrumentation Agent, to configure target applications, so that Cryostat can detect the applications, collect data, and send the data to Cryostat for analysis. You can also optionally enable the Cryostat agent to accept requests from the Cryostat server to start, stop, and delete JFR recordings.

Red Hat build of Cryostat 2.4 distributes two different variations of the Cryostat agent's JAR file. Depending on your setup requirements, you can use either of the following types of agent JAR file:

- An all-in-one "shaded" JAR file that is self-contained and includes the agent code and all of its dependencies
This "shaded" JAR file provides the most convenient form of Cryostat agent to include in your existing applications, because you need to include only one additional agent JAR file. This is a common distribution pattern for similar agents and tools.
- A standard JAR file that contains the agent code without any dependencies
This type of JAR file is useful if you know that dependency conflicts exist between the agent and your workload applications. If you intend to apply your own strategies to provide the correct versions of each dependency to satisfy both the agent and your applications' requirements, you can use the standalone JAR file.



NOTE

Previous releases provided one distribution of the Cryostat agent, which was an all-in-one "shaded" JAR file. The following procedure describes how to install the "shaded" JAR file distribution of the Cryostat 2.4 agent.

As described in [Configuring Java applications: Cryostat agent](#), the Cryostat 2.4 agent supports different options for including the agent's JAR file into your workload applications. The following procedure describes how to add the "shaded" JAR file to your application's dependencies in the **pom.xml** or **build.gradle** file.

Prerequisites

- Logged in to your Cryostat web console.
- Installed JDK version 11 or later.

Procedure

1. Install the Cryostat agent. Choose one of the following options, depending on your application build:
 - Using Maven:
Update the application **pom.xml** file with the Cryostat agent JAR file information.

Example pom.xml

```
<project>
...
<repositories>
  <repository>
    <id>redhat-maven-repository</id>
    <url>https://maven.repository.redhat.com/earlyaccess/all/</url>
  </repository>
</repositories>
...
<build>
  <plugins>
    <plugin>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>3.3.0</version>
      <executions>
        <execution>
          <phase>prepare-package</phase>
          <goals>
            <goal>copy</goal>
          </goals>
          <configuration>
            <artifactItems>
              <artifactItem>
                <groupId>io.cryostat</groupId>
                <artifactId>cryostat-agent</artifactId>
                <version>0.3.0.redhat-00001</version>
                <classifier>shaded</classifier>
              </artifactItem>
            </artifactItems>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>
```

```

        <stripVersion>true</stripVersion>
    </configuration>
</execution>
</executions>
</plugin>
</plugins>
...
</build>
...
</project>

```

The next time you build your application, the Cryostat agent JAR file is available at **target/dependency/cryostat-agent-shaded.jar**.

- Using Gradle:
Update the **build.gradle** file.

Example build.gradle file

```

repositories {
    ...
    maven {
        url "https://maven.repository.redhat.com/earlyaccess/all/"
        credentials {
            username "myusername"
            password "mytoken"
        }
    }
}

```

How you package the agent JAR file into the application depends on the Gradle plug-ins that you use for the build. For example, if you are using the Jib plug-in, update the **build.gradle** file as follows:

Example build.gradle file

```

plugins {
    id 'java'
    id 'application'
    id 'com.google.cloud.tools.jib' version '3.3.1'
    id 'com.ryandens.javaagent-jib' version '0.5.0'
}
...
dependencies {
    ...
    javaagent 'io.cryostat:cryostat-agent:0.3.0.redhat-00001:shaded'
}

```

2. Update the Docker file. The following example uses the **JAVA_OPTS** environment variable to pass the relevant JVM information.

Example

```

...
COPY target/dependency/cryostat-agent.jar /deployments/app/

```

```
...
ENV JAVA_OPTS="-javaagent:/deployments/app/cryostat-agent-shaded.jar"
```

3. Rebuild the container image that is specific to your application.

```
docker build -t docker.io/myorg/myapp:latest -f src/main/docker/Dockerfile
```

4. To supply the JVM system properties or environment variables that you need to configure the Cryostat agent, push the updated image and then modify your application deployment.

Example

```
apiVersion: apps/v1
kind: Deployment
...
spec:
...
template:
...
spec:
  containers:
    - name: sample-app
      image: docker.io/myorg/myapp:latest
      env:
        - name: CRYOSTAT_AGENT_APP_NAME
          value: "myapp"
          # Replace this with the Kubernetes DNS record
          # for the Cryostat Service
        - name: CRYOSTAT_AGENT_BASEURI
          value: "http://cryostat.mynamespace.mycluster.svc:8181"
        - name: POD_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: CRYOSTAT_AGENT_CALLBACK
          value: "http://$(POD_IP):9977" 1
          # Replace "abcd1234" with a base64-encoded authentication token
        - name: CRYOSTAT_AGENT_AUTHORIZATION 2
          value: "Bearer abcd1234"
        - name: CRYOSTAT_AGENT_API_WRITES_ENABLED 3
          value: true
      ports:
        - containerPort: 9977
          protocol: TCP
      resources: {}
      restartPolicy: Always
  status: {}
```

- <1>: Port number **9977** is the default HTTP port that the agent exposes for the internal web server that services Cryostat requests. You can change this port number if it conflicts with your target application into which the agent is installed.
- <2>: The **CRYOSTAT_AGENT_AUTHORIZATION** value shows the credentials that the agent includes in the API requests to Cryostat to advertise its own presence or to push JFR data. You can also create a Kubernetes **Service Account** for this purpose and replace

abcd1234 with the base64-encoded authentication token associated with the service account.

- <3>: The **CRYOSTAT_AGENT_API_WRITES_ENABLED** variable is set to **false** by default. If you want the Cryostat agent to accept requests from the Cryostat server to start, stop, or delete JFR flight recordings, you must set this variable to **true**.

3.2. CONFIGURING APPLICATIONS BY USING JMX CONNECTIONS

For Cryostat to detect and communicate with your target Java applications, you can configure the applications to allow remote Java Management Extensions (JMX) connections.

Prerequisites

- Logged in to your Cryostat web console.
- Created a Cryostat instance in your project.

Procedure

1. To enable remote JMX connections, complete the following steps:
 - a. On your application, define the following Java system property:

```
-Dcom.sun.management.jmxremote.port=<port_num>
```



NOTE

To add the **-Dcom.sun.management.jmxremote.port=<port_num>** property without having to rebuild the target application, you can set the **JAVA_OPTS_APPEND** environment variable on the application. **JAVA_OPTS_APPEND** is an environment variable that is used by Red Hat Universal Base Images (UBI) only.

If you use Red Hat UBI to build application images, set the **JAVA_OPTS_APPEND** variable at build time in the application Docker file, or at runtime by running the following command:

```
oc set env deployment <name> JAVA_OPTS_APPEND="..."
```

If you do not use Red Hat UBI to build application images, refer to the documentation for your base image for information about how to add the Java system properties at build time or runtime.

- b. Specify that the application listens for remote JMX connections by allowing traffic to the application. Use an Red Hat OpenShift Service and specify the following values for its remote JMX port:

Example service.yaml

```
apiVersion: v1
kind: Service
...
```

```
spec:
  ports:
    - name: "jfr-jmx"
      port: 9091
      targetPort: 9091
  ...
```

2. Secure the remote JMX connections:

- a. Enable and configure authentication and SSL/TLS for the remote JMX connections in your application:

```
-Dcom.sun.management.jmxremote.port=<port_num>

# enable JMX authentication
-Dcom.sun.management.jmxremote.authenticate=true

# define users for JMX auth
-Dcom.sun.management.jmxremote.password.file=</path/to/jmxremote.password>

# set permissions for JMX users
-Dcom.sun.management.jmxremote.access.file=</path/to/jmxremote.access>

# enable JMX SSL
-Dcom.sun.management.jmxremote.ssl=true

# enable JMX registry SSL
-Dcom.sun.management.jmxremote.registry.ssl=true

# set your SSL keystore
-Djavax.net.ssl.keyStore=</path/to/keystore>

# set your SSL keystore password
-Djavax.net.ssl.keyStorePassword=<password>
```

- b. Configure Cryostat to trust the application TLS certificate. Create a secret for the application in the same namespace as your Cryostat application and configure Cryostat to refer to the secret. To create a secret for the certificate, run the following command:

```
oc create secret generic myapp-cert --from-file=tls.crt=/path/to/cert.pem
```



NOTE

The certificate must be in a **.pem** file format.

- c. When you create your Cryostat instance, add the secret to the list of trusted TLS certificates. For more information, see [Configuring TLS certificates](#).
- d. To allow your applications to verify that Cryostat is connecting to them by a means other than password authentication, enable TLS client authentication:

```
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true
-Djavax.net.ssl.trustStore=</path/to/truststore>
-Djavax.net.ssl.trustStorePassword=<password>
```

**NOTE**

TLS client authentication requires the cert-manager operator for Red Hat OpenShift.

- e. If you use TLS client authentication for remote JMX connections, the application truststore must contain a Cryostat certificate. The Cryostat operator cert-manager integration creates a self-signed certificate for the Cryostat deployment. This certificate is located in the **<cryostat>-tls** secret, where **<cryostat>** is the name of the Cryostat instance you created.

**NOTE**

The cert-manager Operator also places a Java keystore truststore in the secret.

To mount this truststore in your application deployment, run the following command, replacing "**<myapp>**" with the name of your application deployment and "**<cryostat>**" with the name of your Cryostat instance:

```
oc set volumes deploy <myapp> --add --name=truststore \
  --secret-name=<cryostat>-tls --sub-path=truststore.p12 \
  --mount-path=/var/run/secrets/<myapp>/truststore.p12
```

- f. The Cryostat operator generates the truststore password, which you can find in the **<cryostat>-keystore** secret. To mount this as an environment variable in your application deployment, run the following command:

```
oc set env deploy <myapp> --from='secret/<cryostat>-keystore'
```

- g. Configure the Java arguments for the container. Run the following commands:

```
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true
-Djavax.net.ssl.trustStore=/var/run/secrets/<myapp>/truststore.p12
-Djavax.net.ssl.trustStorePassword="$(KEYSTORE_PASS)"
```

**WARNING**

If you deployed Cryostat and your applications in a testing environment, you might want to configure the target applications without any JMX or TLS authentication. You can do so by using the following set of Java system properties, however, this configuration is not secure and not recommended.

```
-Dcom.sun.management.jmxremote.port=<port_num>
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

- [Configuring TLS certificates](#)
- [Storing and managing JMX credentials](#)

3.3. CONFIGURING APPLICATIONS BY USING THE CRYOSTAT AGENT AND JMX CONNECTIONS

You can configure target applications that run on Java Virtual Machines (JVM) to use a combination of the Cryostat agent and Java Management Extensions (JMX) connections to detect and communicate with the target applications.

You use the Cryostat agent to detect and communicate with the target application and use JMX to expose the Java Flight Recorder (JFR) data.

You must configure the Cryostat agent to communicate with Cryostat about itself and that the agent is reachable through JMX rather than through HTTP.

Prerequisites

- Logged in to your Cryostat web console.
- Created a Cryostat instance in your project.

Procedure

1. Install the Cryostat agent. For application builds that use Maven, update the application **pom.xml** file with the Cryostat agent JAR file information.

Example pom.xml file

```
<project>
...
<repositories>
  <repository>
    <id>redhat-maven-repository</id>
    <url>https://maven.repository.redhat.com/earlyaccess/all/</url>
  </repository>
</repositories>
...
<build>
  <plugins>
    <plugin>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>3.3.0</version>
      <executions>
        <execution>
          <phase>prepare-package</phase>
          <goals>
            <goal>copy</goal>
          </goals>
          <configuration>
            <artifactItems>
              <artifactItem>
                <groupId>io.cryostat</groupId>
                <artifactId>cryostat-agent</artifactId>
```

```

        <version>0.3.0.redhat-00001</version>
        <classifier>shaded</classifier>
      </artifactItem>
    </artifactItems>
    <stripVersion>true</stripVersion>
  </configuration>
</execution>
</executions>
</plugin>
</plugins>
...
</build>
...
</project>

```

2. Modify your application deployment:

Example

```

apiVersion: apps/v1
kind: Deployment
...
spec:
  ...
  template:
    ...
    spec:
      containers:
        - name: sample-app
          image: docker.io/myorg/myapp:latest
          env:
            - name: CRYOSTAT_AGENT_APP_NAME
              value: "myapp"
              # Replace this with the Kubernetes DNS record
              # for the Cryostat Service
            - name: CRYOSTAT_AGENT_BASEURI
              value: "http://cryostat.mynamespace.mycluster.svc:8181"
            - name: POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
            - name: CRYOSTAT_AGENT_CALLBACK
              value: "http://$(POD_IP):9977"
            - name: CRYOSTAT_AGENT_AUTHORIZATION
              value: "Bearer abcd1234"
              # Replace "abcd1234" with a base64-encoded authentication token
              # This environment variable is key to the Cryostat agent
              # and JMX "hybrid" setup.
              # Set the Cryostat agent to register itself with Cryostat
              # as reachable through JMX, rather than reachable through HTTP.
            - name: CRYOSTAT_AGENT_REGISTRATION_PREFER_JMX
              value: "true"
              # Configure the application to load the agent JAR file and
              # to enable JMX, so that the Cryostat agent can register
              # itself as reachable through JMX.
              # To configure authentication and SSL/TLS for the JMX

```

```
# connections, see <1>.
- name: JAVA_OPTS
  value: >-
    -javaagent:/deployments/app/cryostat-agent-shaded.jar
    -Dcom.sun.management.jmxremote.port=9091 1
  ports:
    - containerPort: 9977
      protocol: TCP
  resources: {}
  restartPolicy: Always
  status: {}
```

<1>: To configure authentication and SSL/TLS for the JMX connections and view further configuration options, see [Configuring applications by using JMX connections](#).

3. To enable Cryostat to detect the target application and connect to the Cryostat agent, configure an application **Service**:

Example

```
apiVersion: v1
kind: Service
...
spec:
  ports:
    - name: "jfr-jmx"
      port: 9091
      targetPort: 9091
    - name: "cryostat-agent"
      port: 9977
      targetPort: 9977
  ...
```

Additional resources

- [Configuring applications by using the Cryostat agent](#)
- [Configuring applications by using JMX connections](#)

CHAPTER 4. CREATING A JFR RECORDING WITH CRYOSTAT

With Cryostat, you can create a JDK Flight Recorder (JFR) recording that monitors the performance of your JVM in your containerized application. Additionally, you can take a snapshot of an active JFR recording to capture any collected data, up to a specific point in time, for your target JVM application.

4.1. CREATING A JFR RECORDING IN THE CRYOSTAT WEB CONSOLE

You can create a JFR recording that monitors the performance of your JVM located in your containerized application. After you create a JFR recording, you can start the JFR to capture real-time data for your JVM, such as heap and non-heap memory usage.

Prerequisites

- Installed Cryostat 2.4 on Red Hat OpenShift by using the OperatorHub option.
- Created a Cryostat instance in your Red Hat OpenShift project.
- Logged in to your Cryostat web console.
 - You can retrieve your Cryostat application's URL by using the Red Hat OpenShift web console.

Procedure

1. On the **Dashboard** panel for your Cryostat web console, select a target JVM from the **Target** list.



NOTE

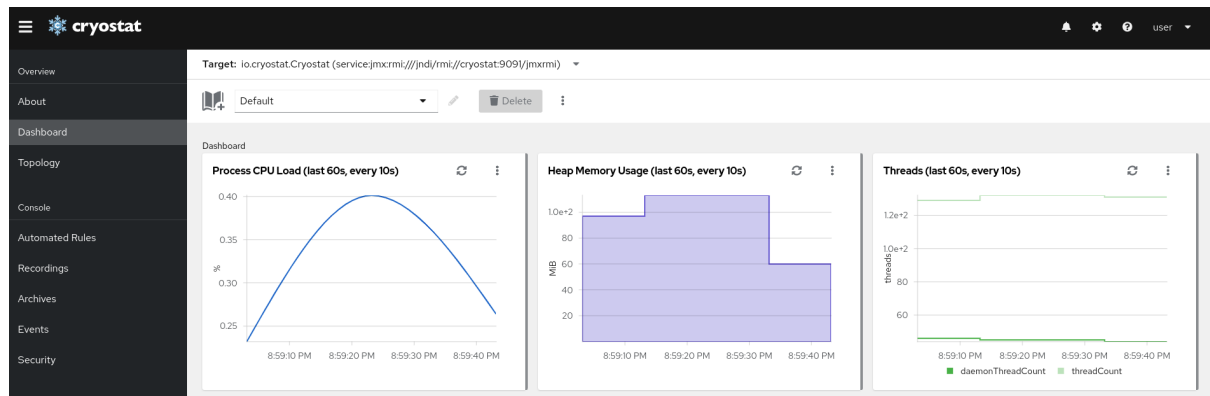
Depending on how you configured your target applications, your target JVMs might be using a JMX connection or an agent HTTP connection. For more information about configuring your target applications, see [Configuring Java applications](#).



IMPORTANT

If your target JVM is using an agent HTTP connection, ensure that you set the **cryostat.agent.api.writes-enabled** property to **true** when you configured your target application to load the Cryostat agent. Otherwise, the Cryostat agent cannot accept requests to start and stop JFR recordings.

Figure 4.1. Example of selecting a Target JVM for your Crystat instance



2. *Optional:* On the **Dashboard** panel, you can create a target JVM. From the **Target** list, click **Create Target**. The **Create Custom Target** window opens.
 - a. In the **Connection URL** field, enter the URL for your JVM's Java Management Extension (JMX) endpoint.
 - b. *Optional:* To test if the **Connection URL** that you specified is valid, click the **Click to test** sample node image. If there is an issue with the **Connection URL**, an error message is displayed that provides a description of the issue and guidance to troubleshoot.
 - c. *Optional:* In the **Alias** field, enter an alias for your JMX Service URL.
 - d. Click **Create**.

Figure 4.2. Create Custom Target window

The screenshot shows the 'Create Custom Target' dialog box. It has a title bar 'Topology > Create Custom Target'. The main content area includes:

- A note: 'Note: If the target requires authentication, use JMX Credential Options to provide credentials.'
- Connection URL** field: Contains 'service:jmx:rmi:///jndi/rmi://10-131-0-25.cryostat-test.pod:9091/jmxrmi'.
- JMX Service URL** field: Contains the same URL as above.
- Alias** field: An empty text input field.
- Connection Nickname** (same as Connection URL if not specified): An empty text input field.
- JMX Credential Options** section: A toggle switch is currently turned off. Below it, text reads: 'Creates credentials that Crystat uses to connect to target JVMs over JMX.'

 On the right side of the dialog, there is a sample node image (a red hat logo) with a 'Click to test' button. Below the image is a button labeled 'CT <Name>'. At the bottom of the dialog are 'Create' and 'Cancel' buttons.

3. From the navigation menu on the Crystat web console, click **Recordings**.
4. *Optional:* Depending on how you configured your target JVM, an **Authentication Required** dialog might open on your web console. In the **Authentication Required** dialog box, enter your **Username** and **Password**. To provide your credentials to the target JVM, click **Save**.

Figure 4.3. Example of a Cryostat Authentication Required window

**NOTE**

If the selected target JMX has Secure Socket Layer (SSL) certification enabled for JMX connections, you must add its certificate when prompted.

Cryostat encrypts and stores credentials for a target JVM application in a database that is stored on a persistent volume claim (PVC) on Red Hat OpenShift. See [Storing and managing credentials](#) (Using Cryostat to manage a JFR recording).

5. On the **Active Recordings** tab, click **Create**.

Figure 4.4. Example of creating an active recording

Target: quarkus-test-7c95cc5b-vjdzq (service:jmx:rmi:///jndi/rmi://10-128-2-17.cryostat-test.pod:9097/jmxrmi) ▾

Recordings

Active Recordings Archived Recordings

▼ Name Filter by name... Create Archive Edit Labels Stop Delete

<input type="checkbox"/>	Name ↑	Start Time ↑	Duration ↑	State ↑	Options	Labels
<input type="checkbox"/>	> demo_recording_1	11/27/2023 5:43:44 PM GMT	Continuous	RUNNING	toDisk: true	template.name: Profiling template.type: TARGET
<input type="checkbox"/>	> demo_recording_2	11/27/2023 5:44:34 PM GMT	Continuous	RUNNING	toDisk: true	template.name: Continuous template.type: TARGET

6. On the **Custom Flight Recording** tab:
 - a. In the **Name** field, enter the name of the recording you want to create. If you enter a name in an invalid format, the web console displays an error message.
 - b. If you want Cryostat to automatically restart an existing recording, select the **Restart if recording already exists** check box.

**NOTE**

If you enter a name that already exists but you do not select **Restart if recording already exists**, Cryostat refuses to create a custom recording when you click the **Create** button.

- c. In the **Duration** field, select whether you want this recording to stop after a specified duration or to run continuously without stopping. If you want Crystat to automatically archive your new JFR recording after the recording stops, click **Archive on Stop**.
- d. In the **Template** field, select the template that you want to use for the recording.

The following example shows continuous JVM monitoring, which you can enable by selecting **Continuous** from above the **Duration** field. This setting means that the recording will continue until you manually stop the recording. The example also shows selection of the **Profiling** template from the **Template** field. This provides additional JVM information to a JFR recording for troubleshooting purposes.

Figure 4.5. Example of creating a custom flight recording

Target: io.cryostat.Cryostat (service:jmx:rmi:///jndi/rmi://cryostat:9091/jmxrmi) ▾

Recordings > Create Recording

Custom Flight Recording Snapshot Recording

JDK Flight Recordings are compact records of events which have occurred within the target JVM. Many event types are built in to the JVM itself, while others are user defined.

Name * demorecording ✓
 Restart if recording already exists
Enter a recording name. This will be unique within the target JVM.

Duration * Continuous Archive on Stop
 0 Seconds ▾
A continuous recording will never be automatically stopped.

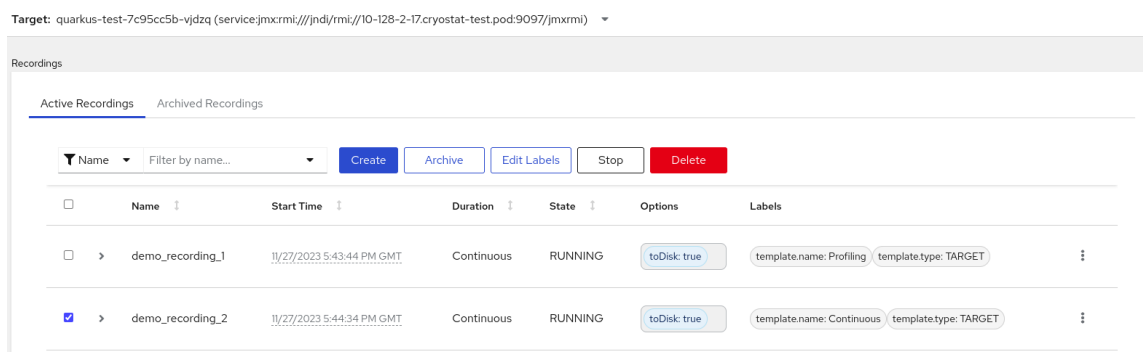
Template * Profiling ✓ ▾
The Event Template to be applied in this recording

> [Show metadata options](#)

> [Show advanced options](#)

7. To access more options, click the following expandable hyperlinks:
 - **Show advanced options**, where you can select additional options for customizing your JFR recording.
 - **Show metadata options**, where you can add custom labels and metadata to your JFR recording.
8. To create your JFR recording, click **Create**. The **Active Recordings** tab opens and lists your JFR recording.
 Your active JFR recording starts collecting data on the target JVM location inside your containerized application. If you specified a fixed duration for your JFR recordings, the target JVM stops the recording when it reaches the fixed duration setting. Otherwise, you must manually stop the recording.
9. *Optional:* On the **Active Recording** tab, you can also stop the recording.
 - a. Select the checkbox next to the JFR recording's name. On the toolbar in the **Active Recordings** tab, the Crystat web console activates the **Stop** button.
 - b. Click **Stop**. The JFR adopts the **STOPPED** status, so it stops monitoring the target JVM. The JFR still shows under the **Active Recording** tab.

Figure 4.6. Example of stopping an active recording



IMPORTANT

JFR recording data might be lost in the following situations:

- Target JVM fails
- Target JVM restarts
- Target JVM Red Hat OpenShift Deployment is scaled down

Archive your JFR recordings to ensure you do not lose your JFR recording's data.

Additional resources

- See [Uploading an SSL certificate](#) (Using Cryostat to manage a JFR recording).
- See [Archiving JDK Flight Recorder \(JFR\) recordings](#) (Using Cryostat to manage a JFR recording).

4.2. CREATING SNAPSHOTS FROM AN ACTIVE RECORDING

You can take a snapshot of an active JFR recording to capture any collected data, up to a specific point in time, for your target JVM application. A snapshot is like a checkpoint marker that has a start point and an end point for a given time segment in a running JFR recording.

A snapshot gets stored in the memory of a target JVM application. This differs from an archive in that Cryostat stores an archive on a cloud storage disk, which is a more permanent solution for storing a JFR recording's data.

You can take snapshots of recordings if you want to experiment with different configuration changes among active JFR recordings.

When you create a snapshot for your JFR recording, Cryostat creates a new target JVM named **snapshot -<snapshot_number>**, where **<snapshot_number>** is the number that Cryostat automatically assigns to your snapshot.

A target JVM recognizes a snapshot as an active recording. Cryostat sets any JFR snapshots in the **STOPPED** state, which means that the JFR snapshot does not record new data to the target JVM. Depending on the JFR configuration, active JFR recordings can continue to monitor the target JVM regardless of the number of snapshots taken.



NOTE

For a JFR recording that you set for continuous monitoring of a target JVM application, ensure that you create archived recordings to avoid losing JFR recording data.

If you choose to take regular snapshots to store your JFR recording data, the target JVM application might free some of its data storage space by replacing older recording data with newer recording data.

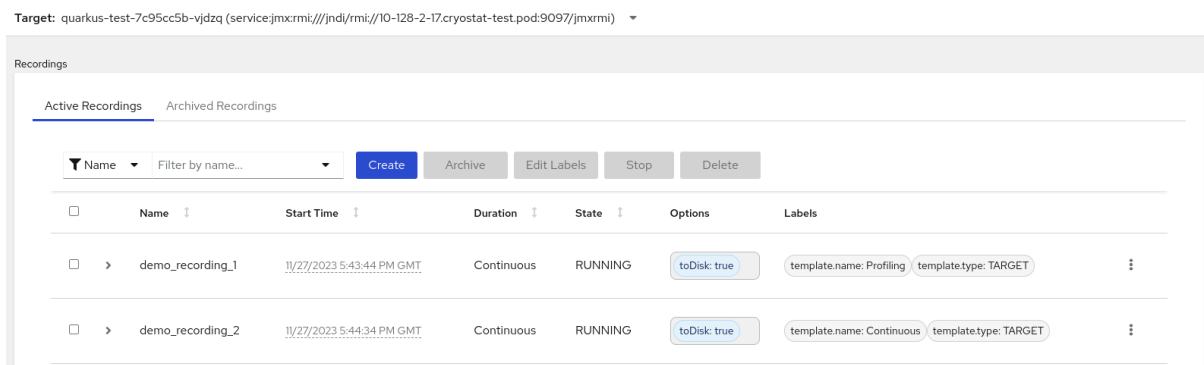
Prerequisites

- Entered your authentication details for your Crystat instance.
- Created a target JVM recording and entered your authenticated details to access the **Recordings** menu. See [Creating a JDK Flight Recorder \(JFR\) recording](#) (Creating a JFR recording with Crystat).

Procedure

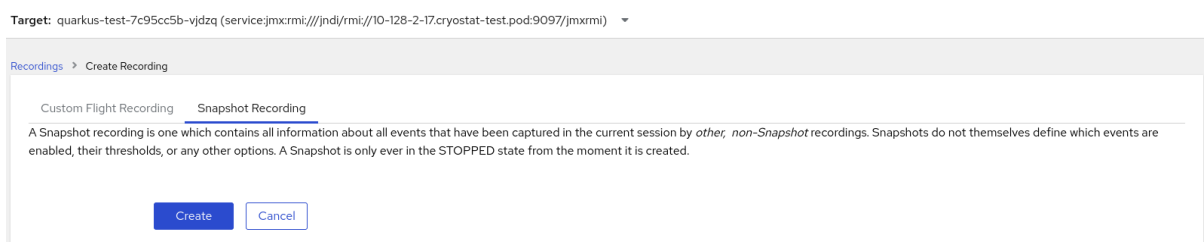
1. On the **Active Recordings** tab, click the **Create** button. A new window opens on the web console.

Figure 4.7. Example of creating an active recording



2. Click the **Snapshot Recording** tab.

Figure 4.8. Example of creating a snapshot recording



3. Click **Create**. The **Active Recordings** table opens and it lists your JFR snapshot recording. The following example shows a JFR snapshot recording named **snapshot-3**.

Figure 4.9. Example of a completed snapshot recording

Active Recordings		Archived Recordings				
Name	Filter by name...	Create	Archive	Edit Labels	Stop	Delete
<input type="checkbox"/>	Name ↑	Start Time ↑	Duration ↑	State ↑	Options	Labels
<input type="checkbox"/>	> demo_recording_1	11/27/2023 5:43:44 PM GMT	Continuous	RUNNING	toDisk: true	template.name: Profiling template.type: TARGET
<input type="checkbox"/>	> demo_recording_2	11/27/2023 5:44:34 PM GMT	Continuous	RUNNING	toDisk: true	template.name: Continuous template.type: TARGET
<input type="checkbox"/>	> snapshot-3	11/27/2023 5:43:44 PM GMT	1724s	STOPPED	toDisk: true	-

**NOTE**

You can identify snapshots by the *snapshot* prefix from the list of active recordings.

Next steps

- To archive your JFR snapshot recording, see [Archiving JDK Flight Recorder \(JFR\) recordings](#).

4.3. LABELS FOR JFR RECORDINGS

When you create a JDK Flight Recorder (JFR) recording on Cryostat 2.4, you can add metadata to the recording by specifying a series of key-value label pairs.

Additionally, you can attach custom labels to JFR recordings that are inside a target JVM, so that you can easily identify and better manage your JFR recordings.

The following list details some common recording label use cases:

- Attach metadata to your JFR recording.
- Perform batch operations on recordings that contain identical labels.
- Use labels when running queries on recordings.

You can use Cryostat to create a JFR recording that monitors the performance of your JVM in your containerized application. Additionally, you can take a snapshot of an active JFR recording to capture any collected data, up to a specific point in time, for your target JVM application.

4.3.1. Adding labels to JFR recordings

When you create a JFR recording on Cryostat 2.4, you can use labels to add metadata that contain key-value label pairs to the recording.

Cryostat applies default recording labels to a created JFR recording. These default labels capture information about the event template that Cryostat used to create the JFR recording.

You can add custom labels to your JFR recording so that you can run specific queries that meet your needs, such as identifying specific JFR recordings or performing batch operations on recordings with the same applied labels.

Prerequisites

- Logged in to your Cryostat web console.
- Created or selected a target JVM for your Cryostat instance.

Procedure

1. From your Cryostat web console, click **Recordings**.
2. Under the **Active Recordings** tab, click **Create**.
3. On the **Custom Flight Recording** tab, expand **Show metadata options**.



NOTE

On the **Custom Flight Recording** tab, you must complete any mandatory field that is marked with an asterisk.

4. Click **Add label**.

Figure 4.10. The Add Label button that is displayed under the Custom Flight Recording tab

The screenshot shows the 'Custom Flight Recording' configuration page. It includes a 'Name' field with a red asterisk, a 'Duration' field with a dropdown menu set to '30' seconds, and a 'Template' dropdown menu. Below these fields, there is a section for 'Labels' with an 'Add Label' button. The 'Create' and 'Cancel' buttons are at the bottom of the form.

5. Enter values in the provided **Key** and **Value** fields. For example, if you want to file an issue with the recordings, you could enter the reason in the **Key** field and then enter the issue type in the **Value** field.
6. Click **Create** to create your JFR recording. Your recording is then shown under the **Active Recordings** tab along with any specified recording labels and custom labels.

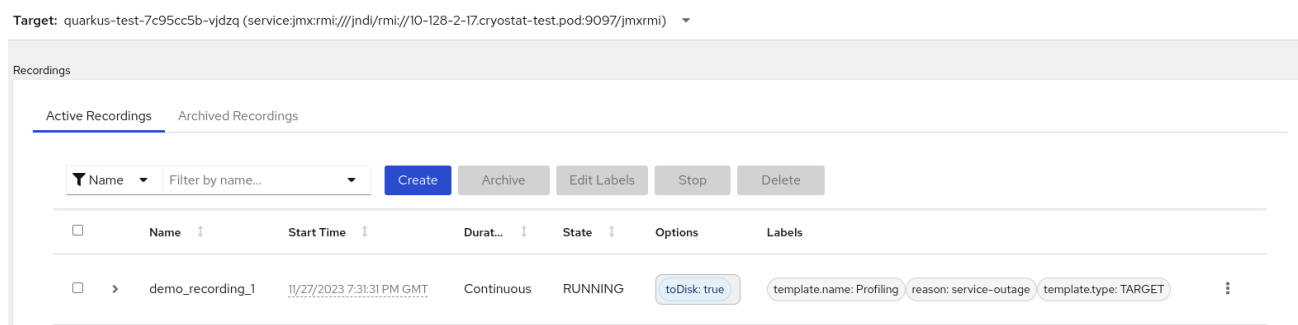
TIP

You can access archived JFR recordings from the **Archives** menu. See [Uploading a JFR recording to Cryostat archives location](#) (Using Cryostat to manage a JFR recording).

Example

The following example shows two default recording labels, **template.name: Profiling** and **template.type: TARGET**, and one custom label, **reason:service-outage**.

Figure 4.11. Example of an active recording with defined recording labels and a custom label



4.3.2. Editing a label for your JFR recording

On the Cryostat web console, you can navigate to the **Recordings** menu and then edit a label and its metadata for your JFR recording. You can also edit the label and metadata for a JFR recording that you uploaded to archives.

Prerequisites

- Logged in to your Cryostat web console.
- Created a JFR recording and attach labels to this recording.

Procedure

1. On your Cryostat web console, click the **Recording** menu.
2. From the **Active Recordings** tab, locate your JFR recording, and then select the checkbox next to it.
3. Click **Edit Labels**. An **Edit Recording Label** pane opens in your Cryostat web console, which you can use to add, edit, or delete labels for your JFR recording.

TIP

You can select multiple JFR recordings by selecting the checkbox that is next to each recording. Click the **Edit Labels** button if you want to bulk edit recordings that contain the same labels or add new identical labels to multiple recordings.

4. *Optional:* You can perform any of the following actions from the **Edit Recording Labels** pane:
 - a. Click **Add** to create a label.
 - b. Delete a label by clicking the **X** next to the label.
 - c. Edit a label by modifying any content in a field. After you edit content, a green tick is shown in the field to indicate an edit.
5. Click **Save**.
6. *Optional:* You can archive your JFR recordings along with their labels by completing the following steps:
 - a. Select the checkbox next to the recording's name.

- b. Click the **Archive** button. You can locate your recording under the **Archived Recordings** tab.

By archiving your recording with its labels, you can enhance your search capabilities when you want to locate the recording at a later stage. You can also add additional labels to any recording that you uploaded to the Cryostat archives.



NOTE

Cryostat preserves any labels with the recording for the lifetime of the archived recording.

Verification

- From the **Active Recordings** tab, check that your changes display under the **Labels** section for your recording.

Additional resources

- [Archiving JDK Flight Recorder \(JFR\) recordings \(Using Cryostat to manage a JFR recording\)](#)

Revised on 2024-03-06 19:01:14 UTC