OpenShift Container Platform 4.14

Cluster Observability Operator

Configuring and using the Cluster Observability Operator in OpenShift Container Platform

OpenShift Container Platform 4.14 Cluster Observability Operator

Configuring and using the Cluster Observability Operator in OpenShift Container Platform

Legal Notice

Abstract

Use the Cluster Observability Operator to deploy and configure observability components in OpenShift Container Platform.

Table of Contents

CHAPTER 1. CLUSTER OBSERVABILITY OPERATOR RELEASE NOTES

IMPORTANT

The Cluster Observability Operator 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 .

The Cluster Observability Operator (COO) is an optional OpenShift Container Platform Operator that enables administrators to create standalone monitoring stacks that are independently configurable for use by different services and users.

The COO complements the built-in monitoring capabilities of OpenShift Container Platform. You can deploy it in parallel with the default platform and user workload monitoring stacks managed by the Cluster Monitoring Operator (CMO).

These release notes track the development of the Cluster Observability Operator in OpenShift Container Platform.

1.1. CLUSTER OBSERVABILITY OPERATOR 0.2.0

The following advisory is available for Cluster Observability Operator 0.2.0:

- RHEA-2024:2662 Cluster Observability Operator 0.2.0

1.1.1. New features and enhancements

- With this release, the Cluster Observability Operator supports installing and managing observability-related plugins for the OpenShift Container Platform web console user interface (UI). (COO-58)

1.2. CLUSTER OBSERVABILITY OPERATOR 0.1.3

The following advisory is available for Cluster Observability Operator 0.1.3:

- RHEA-2024:1744 Cluster Observability Operator 0.1.3

1.2.1. Bug fixes

- Previously, if you tried to access the Prometheus web user interface (UI) at **http://<prometheus_url>:9090/graph**, the following error message would display: **Error opening React index.html: open web/ui/static/react/index.html: no such file or directory**. This release resolves the issue, and the Prometheus web UI now displays correctly. (COO-34)

1.3. CLUSTER OBSERVABILITY OPERATOR 0.1.2

The following advisory is available for Cluster Observability Operator 0.1.2:

- RHEA-2024:1534 Cluster Observability Operator 0.1.2

1.3.1. CVEs

- CVE-2023-45142

1.3.2. Bug fixes

- Previously, certain cluster service version (CSV) annotations were not included in the metadata for COO. Because of these missing annotations, certain COO features and capabilities did not appear in the package manifest or in the OperatorHub user interface. This release adds the missing annotations, thereby resolving this issue. (COO-11)

- Previously, automatic updates of the COO did not work, and a newer version of the Operator did not automatically replace the older version, even though the newer version was available in OperatorHub. This release resolves the issue. (COO-12)

- Previously, Thanos Querier only listened for network traffic on port 9090 of 127.0.0.1 (**localhost**), which resulted in a **502 Bad Gateway** error if you tried to reach the Thanos Querier service. With this release, the Thanos Querier configuration has been updated so that the component now listens on the default port (10902), thereby resolving the issue. As a result of this change, you can also now modify the port via server side apply (SSA) and add a proxy chain, if required. (COO-14)

1.4. CLUSTER OBSERVABILITY OPERATOR 0.1.1

The following advisory is available for Cluster Observability Operator 0.1.1:

- 2024:0550 Cluster Observability Operator 0.1.1

1.4.1. New features and enhancements

This release updates the Cluster Observability Operator to support installing the Operator in restricted networks or disconnected environments.

1.5. CLUSTER OBSERVABILITY OPERATOR 0.1

This release makes a Technology Preview version of the Cluster Observability Operator available on OperatorHub.

CHAPTER 2. CLUSTER OBSERVABILITY OPERATOR OVERVIEW

> **IMPORTANT**
>
> The Cluster Observability Operator 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 .

The Cluster Observability Operator (COO) is an optional component of the OpenShift Container Platform. You can deploy it to create standalone monitoring stacks that are independently configurable for use by different services and users.

The COO deploys the following monitoring components:

- Prometheus

- Thanos Querier (optional)

- Alertmanager (optional)

The COO components function independently of the default in-cluster monitoring stack, which is deployed and managed by the Cluster Monitoring Operator (CMO). Monitoring stacks deployed by the two Operators do not conflict. You can use a COO monitoring stack in addition to the default platform monitoring components deployed by the CMO.

2.1. UNDERSTANDING THE CLUSTER OBSERVABILITY OPERATOR

A default monitoring stack created by the Cluster Observability Operator (COO) includes a highly available Prometheus instance capable of sending metrics to an external endpoint by using remote write.

Each COO stack also includes an optional Thanos Querier component, which you can use to query a highly available Prometheus instance from a central location, and an optional Alertmanager component, which you can use to set up alert configurations for different services.

2.1.1. Advantages of using the Cluster Observability Operator

The **MonitoringStack** CRD used by the COO offers an opinionated default monitoring configuration for COO-deployed monitoring components, but you can customize it to suit more complex requirements.

Deploying a COO-managed monitoring stack can help meet monitoring needs that are difficult or impossible to address by using the core platform monitoring stack deployed by the Cluster Monitoring Operator (CMO). A monitoring stack deployed using COO has the following advantages over core platform and user workload monitoring:

Extendability

Users can add more metrics to a COO-deployed monitoring stack, which is not possible with core platform monitoring without losing support. In addition, COO-managed stacks can receive certain cluster-specific metrics from core platform monitoring by using federation.

Multi-tenancy support

The COO can create a monitoring stack per user namespace. You can also deploy multiple stacks per namespace or a single stack for multiple namespaces. For example, cluster administrators, SRE teams, and development teams can all deploy their own monitoring stacks on a single cluster, rather than having to use a single shared stack of monitoring components. Users on different teams can then independently configure features such as separate alerts, alert routing, and alert receivers for their applications and services.

Scalability

You can create COO-managed monitoring stacks as needed. Multiple monitoring stacks can run on a single cluster, which can facilitate the monitoring of very large clusters by using manual sharding. This ability addresses cases where the number of metrics exceeds the monitoring capabilities of a single Prometheus instance.

Flexibility

Deploying the COO with Operator Lifecycle Manager (OLM) decouples COO releases from OpenShift Container Platform release cycles. This method of deployment enables faster release iterations and the ability to respond rapidly to changing requirements and issues. Additionally, by deploying a COO-managed monitoring stack, users can manage alerting rules independently of OpenShift Container Platform release cycles.

Highly customizable

The COO can delegate ownership of single configurable fields in custom resources to users by using Server-Side Apply (SSA), which enhances customization.

Additional resources

- Kubernetes documentation for Server-Side Apply (SSA)

CHAPTER 3. INSTALLING THE CLUSTER OBSERVABILITY OPERATOR

IMPORTANT

The Cluster Observability Operator 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 .

As a cluster administrator, you can install or remove the Cluster Observability Operator (COO) from OperatorHub by using the OpenShift Container Platform web console. OperatorHub is a user interface that works in conjunction with Operator Lifecycle Manager (OLM), which installs and manages Operators on a cluster.

3.1. INSTALLING THE CLUSTER OBSERVABILITY OPERATOR IN THE WEB CONSOLE

Install the Cluster Observability Operator (COO) from OperatorHub by using the OpenShift Container Platform web console.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** cluster role.

- You have logged in to the OpenShift Container Platform web console.

Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.

2. Type **cluster observability operator** in the **Filter by keyword** box.

3. Click **Cluster Observability Operator** in the list of results.

4. Read the information about the Operator, and review the following default installation settings:

   - **Update channel → development**

   - **Version → <most_recent_version>**

   - **Installation mode → All namespaces on the cluster (default)**

   - **Installed Namespace → openshift-operators**

   - **Update approval → Automatic**

5. Optional: Change default installation settings to suit your requirements. For example, you can select to subscribe to a different update channel, to install an older released version of the Operator, or to require manual approval for updates to new versions of the Operator.

6. Click **Install**.

Verification

- Go to **Operators → Installed Operators**, and verify that the **Cluster Observability Operator** entry appears in the list.

Additional resources

[Adding Operators to a cluster](#)

3.2. UNINSTALLING THE CLUSTER OBSERVABILITY OPERATOR USING THE WEB CONSOLE

If you have installed the Cluster Observability Operator (COO) by using OperatorHub, you can uninstall it in the OpenShift Container Platform web console.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** cluster role.

- You have logged in to the OpenShift Container Platform web console.

Procedure

1. Go to **Operators → Installed Operators**.

2. Locate the **Cluster Observability Operator** entry in the list.

3. Click ⋮ for this entry and select **Uninstall Operator**.

Verification

- Go to **Operators → Installed Operators**, and verify that the **Cluster Observability Operator** entry no longer appears in the list.

CHAPTER 4. CONFIGURING THE CLUSTER OBSERVABILITY OPERATOR TO MONITOR A SERVICE

IMPORTANT

The Cluster Observability Operator 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 .

You can monitor metrics for a service by configuring monitoring stacks managed by the Cluster Observability Operator (COO).

To test monitoring a service, follow these steps:

- Deploy a sample service that defines a service endpoint.

- Create a **ServiceMonitor** object that specifies how the service is to be monitored by the COO.

- Create a **MonitoringStack** object to discover the **ServiceMonitor** object.

4.1. DEPLOYING A SAMPLE SERVICE FOR CLUSTER OBSERVABILITY OPERATOR

This configuration deploys a sample service named **prometheus-coo-example-app** in the user–defined **ns1-coo** project. The service exposes the custom **version** metric.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** cluster role or as a user with administrative permissions for the namespace.

Procedure

1. Create a YAML file named **prometheus-coo-example-app.yaml** that contains the following configuration details for a namespace, deployment, and service:

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1-coo
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-coo-example-app
  name: prometheus-coo-example-app
```

```
    namespace: ns1-coo
  spec:
   replicas: 1
   selector:
    matchLabels:
      app: prometheus-coo-example-app
   template:
    metadata:
      labels:
       app: prometheus-coo-example-app
    spec:
      containers:
      - image: ghcr.io/rhobs/prometheus-example-app:0.4.2
       imagePullPolicy: IfNotPresent
       name: prometheus-coo-example-app
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: prometheus-coo-example-app
  name: prometheus-coo-example-app
  namespace: ns1-coo
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-coo-example-app
  type: ClusterIP
```

2. Save the file.

3. Apply the configuration to the cluster by running the following command:

   ```
   $ oc apply -f prometheus-coo-example-app.yaml
   ```

4. Verify that the pod is running by running the following command and observing the output:

   ```
   $ oc -n ns1-coo get pod
   ```

   Example output

   ```
   NAME                               READY   STATUS    RESTARTS   AGE
   prometheus-coo-example-app-0927545cb7-anskj  1/1     Running   0          81m
   ```

4.2. SPECIFYING HOW A SERVICE IS MONITORED BY CLUSTER OBSERVABILITY OPERATOR

To use the metrics exposed by the sample service you created in the "Deploying a sample service for Cluster Observability Operator" section, you must configure monitoring components to scrape metrics from the /**metrics** endpoint.

You can create this configuration by using a **ServiceMonitor** object that specifies how the service is to be monitored, or a **PodMonitor** object that specifies how a pod is to be monitored. The **ServiceMonitor** object requires a **Service** object. The **PodMonitor** object does not, which enables the **MonitoringStack** object to scrape metrics directly from the metrics endpoint exposed by a pod.

This procedure shows how to create a **ServiceMonitor** object for a sample service named **prometheus-coo-example-app** in the **ns1-coo** namespace.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** cluster role or as a user with administrative permissions for the namespace.

- You have installed the Cluster Observability Operator.

- You have deployed the **prometheus-coo-example-app** sample service in the **ns1-coo** namespace.

> **NOTE**
>
> The **prometheus-coo-example-app** sample service does not support TLS authentication.

Procedure

1. Create a YAML file named **example-coo-app-service-monitor.yaml** that contains the following **ServiceMonitor** object configuration details:

   ```
   apiVersion: monitoring.rhobs/v1
   kind: ServiceMonitor
   metadata:
     labels:
       k8s-app: prometheus-coo-example-monitor
     name: prometheus-coo-example-monitor
     namespace: ns1-coo
   spec:
     endpoints:
     - interval: 30s
       port: web
       scheme: http
     selector:
       matchLabels:
         app: prometheus-coo-example-app
   ```

   This configuration defines a **ServiceMonitor** object that the **MonitoringStack** object will reference to scrape the metrics data exposed by the **prometheus-coo-example-app** sample service.

2. Apply the configuration to the cluster by running the following command:

   ```
   $ oc apply -f example-app-service-monitor.yaml
   ```

3. Verify that the **ServiceMonitor** resource is created by running the following command and observing the output:

```
$ oc -n ns1-coo get servicemonitors.monitoring.rhobs
```

Example output

```
NAME                            AGE
prometheus-coo-example-monitor   81m
```

4.3. CREATING A MONITORINGSTACK OBJECT FOR THE CLUSTER OBSERVABILITY OPERATOR

To scrape the metrics data exposed by the target **prometheus-coo-example-app** service, create a **MonitoringStack** object that references the **ServiceMonitor** object you created in the "Specifying how a service is monitored for Cluster Observability Operator" section. This **MonitoringStack** object can then discover the service and scrape the exposed metrics data from it.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** cluster role or as a user with administrative permissions for the namespace.

- You have installed the Cluster Observability Operator.

- You have deployed the **prometheus-coo-example-app** sample service in the **ns1-coo** namespace.

- You have created a **ServiceMonitor** object named **prometheus-coo-example-monitor** in the **ns1-coo** namespace.

Procedure

1. Create a YAML file for the **MonitoringStack** object configuration. For this example, name the file **example-coo-monitoring-stack.yaml**.

2. Add the following **MonitoringStack** object configuration details:

   Example MonitoringStack object

   ```
   apiVersion: monitoring.rhobs/v1alpha1
   kind: MonitoringStack
   metadata:
     name: example-coo-monitoring-stack
     namespace: ns1-coo
   spec:
     logLevel: debug
     retention: 1d
     resourceSelector:
       matchLabels:
         k8s-app: prometheus-coo-example-monitor
   ```

3. Apply the **MonitoringStack** object by running the following command:

```
$ oc apply -f example-coo-monitoring-stack.yaml
```

4. Verify that the **MonitoringStack** object is available by running the following command and inspecting the output:

```
$ oc -n ns1-coo get monitoringstack
```

Example output

```
NAME                       AGE
example-coo-monitoring-stack   81m
```