OpenShift Container Platform 4.16

Cluster Observability Operator

在 OpenShift Container Platform 中配置和使用 Cluster Observability Operator

Last Updated: 2024-07-02

OpenShift Container Platform 4.16 Cluster Observability Operator

在 OpenShift Container Platform 中配置和使用 Cluster Observability Operator

法律通告

摘要

使用 Cluster Observability Operator 在 OpenShift Container Platform 中部署和配置可观察性组件。

目录

第 1 章 CLUSTER OBSERVABILITY OPERATOR 发行注记

重要

Cluster Observability Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

Cluster Observability Operator (COO) 是一个可选的 OpenShift Container Platform Operator，它可让管理员创建独立监控堆栈，供不同的服务和用户使用。

COO 补充 OpenShift Container Platform 的内置监控功能。您可以使用由 Cluster Monitoring Operator (CMO) 管理的默认平台和用户工作负载监控堆栈并行部署它。

本发行注记介绍了 OpenShift Container Platform 中 Cluster Observability Operator 的开发。

1.1. CLUSTER OBSERVABILITY OPERATOR 0.2.0

以下公告可用于 Cluster Observability Operator 0.2.0 ：

- RHEA-2024:2662 Cluster Observability Operator 0.2.0

1.1.1. 新功能及功能增强

- 在这个版本中，Cluster Observability Operator 支持为 OpenShift Container Platform Web 控制台界面(UI)安装和管理与可观察性相关的插件。(COO-58)

1.2. CLUSTER OBSERVABILITY OPERATOR 0.1.3

以下公告可用于 Cluster Observability Operator 0.1.3：

- RHEA-2024:1744 Cluster Observability Operator 0.1.3

1.2.1. 程序错误修复

- 在以前的版本中，如果您试图访问 **http://<prometheus_url>:9090/graph** 的 Prometheus web 用户界面(UI)，会显示以下出错信息：**Error open React index.html: open web/ui/static/react/index.html: no such file or directory**。此发行版本解决了这个问题，Prometheus Web UI 现在可以正确显示。(COO-34)

1.3. CLUSTER OBSERVABILITY OPERATOR 0.1.2

以下公告可用于 Cluster Observability Operator 0.1.2：

- RHEA-2024:1534 Cluster Observability Operator 0.1.2

1.3.1. CVE

- CVE-2023-45142

1.3.2. 程序错误修复

- 在以前的版本中，某些集群服务版本 (CSV) 注解没有包括在 COO 的元数据中。由于这些缺少的注解，某些 COO 功能没有出现在软件包清单或 OperatorHub 用户界面中。此发行版本添加了缺少的注解，从而解决了这个问题。(COO-11)

- 在以前的版本中，COO 的自动更新无法正常工作，Operator 的较新版本不会自动替换旧版本，即使 OperatorHub 中提供了更新的版本。此发行版本解决了这个问题。(COO-12)

- 在以前的版本中，Thanos Querier 只侦听 127.0.0.1 的端口 9090 (**localhos**)上的网络流量，如果您试图访问 Thanos Querier 服务，这会导致 **502 Bad Gateway** 错误。在这个版本中，Thanos Querier 配置已被更新，因此组件现在侦听默认端口(10902)，从而解决了这个问题。现在，您还可以通过服务器端应用(SSA)修改端口，并在需要时添加代理链。(COO-14)

1.4. CLUSTER OBSERVABILITY OPERATOR 0.1.1

以下公告可用于 Cluster Observability Operator 0.1.1：

- 2024:0550 cluster Observability Operator 0.1.1

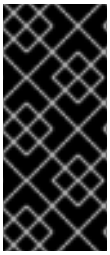
1.4.1. 新功能及功能增强

此发行版本更新了 Cluster Observability Operator，以支持在受限网络中或断开连接的环境中安装 Operator。

1.5. CLUSTER OBSERVABILITY OPERATOR 0.1

此发行版本在 OperatorHub 上提供了 Cluster Observability Operator 的技术预览版本。

第 2 章 CLUSTER OBSERVABILITY OPERATOR 概述



重要

Cluster Observability Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

Cluster Observability Operator (COO) 是 OpenShift Container Platform 的一个可选组件。您可以进行部署，以创建独立可配置供不同服务和用户使用的独立监控堆栈。

COO 部署以下监控组件：

- Prometheus
- Thanos Querier （可选）
- Alertmanager （可选）

COO 组件独立于默认的集群内监控堆栈（由 Cluster Monitoring Operator (CMO) 部署和管理）。两个 Operator 部署的监控堆栈不会冲突。除了 CMO 部署的默认平台监控组件外，您还可以使用 COO 监控堆栈。

2.1. 了解 CLUSTER OBSERVABILITY OPERATOR

Cluster Observability Operator (COO) 创建的默认监控堆栈包括可使用远程写入将指标发送到外部端点的高可用性 Prometheus 实例。

每个 COO 堆栈还包括一个可选的 Thanos Querier 组件，可用于从中央位置查询高度可用的 Prometheus 实例，以及可选的 Alertmanager 组件，您可以使用它来为不同的服务设置警报配置。

2.1.1. 使用 Cluster Observability Operator 的优点

COO 使用的 **MonitoringStack** CRD 为 COO 部署的监控组件提供了建议的默认监控配置，但您可以自定义它来满足更复杂的要求。

部署 COO 管理的监控堆栈可帮助满足监控需求，使用 Cluster Monitoring Operator (CMO) 部署的核心平台监控堆栈来解决。使用 COO 部署的监控堆栈与核心平台和用户工作负载监控相比有以下优点：

扩展性

用户可以在 COO 部署的监控堆栈中添加更多指标，该堆栈无法在没有丢失支持的情况下进行核心平台监控。另外，COO 管理的堆栈可以使用联邦从核心平台监控接收特定集群特定指标。

多租户支持

COO 可以为每个用户命名空间创建一个监控堆栈。您还可以为每个命名空间部署多个堆栈，或为多个命名空间部署单个堆栈。例如，集群管理员、SRE 团队和开发团队都可以在单个集群中部署自己的监控堆栈，而不必使用单一的监控组件的共享堆栈。然后，不同团队上的用户可以独立配置其应用程序和服务的警报、警报路由和警报接收器等功能。

可扩展性

您可以根据需要创建 COO 管理的监控堆栈。多个监控堆栈可以在单个集群中运行，这有助于使用手动分片监控非常大的集群。此功能解决了指标数量超过单个 Prometheus 实例的监控功能的情况。

灵活性

使用 Operator Lifecycle Manager (OLM) 部署 COO 与 OpenShift Container Platform 发行周期分离 COO 版本。这种部署方法可以更快地进行发布迭代，并能够快速响应变化的要求和问题。另外，通过部署 COO 管理的监控堆栈，用户可以独立于 OpenShift Container Platform 发行周期管理警报规则。

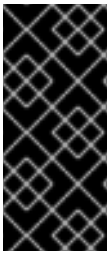
高度自定义

COO 可以使用 Server-Side Apply (SSA) 将自定义资源中单个可配置字段的所有权委派给用户，从而增强自定义。

其他资源

- 用于 Server-Side Apply (SSA)的 Kubernetes 文档

第 3 章 安装 CLUSTER OBSERVABILITY OPERATOR



重要

Cluster Observability Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

作为集群管理员，您可以使用 OpenShift Container Platform Web 控制台从 OperatorHub 安装或删除 Cluster Observability Operator (COO)。OperatorHub 是一个用户界面，可与 Operator Lifecycle Manager (OLM) 结合使用，它在集群中安装和管理 Operator。

3.1. 在 WEB 控制台中安装 CLUSTER OBSERVABILITY OPERATOR

使用 OpenShift Container Platform Web 控制台从 OperatorHub 安装 Cluster Observability Operator (COO)。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 已登陆到 OpenShift Container Platform Web 控制台。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 框中键入 **cluster observability operator**。
3. 在结果列表中点 **Cluster Observability Operator**。
4. 阅读 Operator 的信息，并查看以下默认安装设置：
   - **Update channel → development**
   - **Version →** <most_recent_version>
   - **Installation mode → All namespaces on the cluster (default)**
   - **Installed Namespace → openshift-operators**
   - **Update approval → Automatic**
5. 可选：更改默认安装设置以满足您的要求。例如，您可以选择订阅不同的更新频道，安装一个旧的 Operator 发行版本，或者需要手动批准对 Operator 的新版本进行更新。
6. 点 **Install**。

验证

- 进入 **Operators → Installed Operators**，并验证 **Cluster Observability Operator** 条目是否出现在列表中。

其他资源

在集群中添加 Operator

3.2. 使用 WEB 控制台卸载 CLUSTER OBSERVABILITY OPERATOR

如果使用 OperatorHub 安装 Cluster Observability Operator (COO)，您可以在 OpenShift Container Platform Web 控制台中卸载它。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 已登陆到 OpenShift Container Platform Web 控制台。

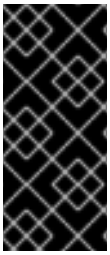
流程

1. 进入 **Operators → Installed Operators**。
2. 在列表中找到 **Cluster Observability Operator** 条目。
3. 在这个条目中点  并选择 **Uninstall Operator**。

验证

- 进入 **Operators → Installed Operators**，验证 **Cluster Observability Operator** 条目是否不再出现在列表中。

第 4 章 配置 CLUSTER OBSERVABILITY OPERATOR 以监控服务



重要

Cluster Observability Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

您可以通过配置由 Cluster Observability Operator (COO) 管理的监控堆栈来监控服务的指标。

要测试监控服务，请按照以下步骤执行：

- 部署定义服务端点的示例服务。
- 创建一个 **ServiceMonitor** 对象，用于指定服务如何被 COO 监控。
- 创建 **MonitoringStack** 对象来发现 **ServiceMonitor** 对象。

4.1. 为 CLUSTER OBSERVABILITY OPERATOR 部署示例服务

此配置会在用户定义的 **ns1-coo** 项目中部署一个名为 **prometheus-coo-example-app** 的示例服务。该服务会公开自定义 **version** 指标。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有命名空间管理权限的用户身份访问集群。

流程

1. 创建名为 **prometheus-coo-example-app.yaml** 的 YAML 文件，其中包含命名空间、部署和服务的以下配置详情：

```
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. 保存该文件。

3. 运行以下命令，将配置应用到集群：

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

4. 运行以下命令验证 pod 是否正在运行，并观察输出：

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

输出示例

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

4.2. 指定 CLUSTER OBSERVABILITY OPERATOR 如何监控服务

要使用您在 "Deploying a sample service for Cluster Observability Operator" 部分创建的样本服务公开的指标，您必须将监控组件配置为从 **/metrics** 端点中提取指标。

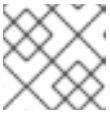
您可以使用一个 **ServiceMonitor** 对象来创建此配置，指定如何监控该服务，或指定如何监控 pod 的 **PodMonitor** 对象。**ServiceMonitor** 对象需要一个 **Service** 对象。**PodMonitor** 对象不需要，它允许 **MonitoringStack** 对象直接从 Pod 公开的指标端点中提取指标。

此流程演示了如何在 **ns1-coo** 命名空间中为名为 **prometheus-coo-example-app** 的示例服务创建 **ServiceMonitor** 对象。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有命名空间管理权限的用户身份访问集群。

- 已安装 Cluster Observability Operator。
- 您已在 **ns1-coo** 命名空间中部署了 **prometheus-coo-example-app** 示例服务。



注意

prometheus-coo-example-app 示例服务不支持 TLS 身份验证。

流程

1. 创建名为 **example-coo-app-service-monitor.yaml** 的 YAML 文件，其中包含以下 **ServiceMonitor** 对象配置详情：

```
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
```

此配置定义 **MonitoringStack** 对象将引用的 **ServiceMonitor** 对象，以提取由 **prometheus-coo-example-app** 示例服务公开的指标数据。

2. 运行以下命令，将配置应用到集群：

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

3. 运行以下命令并验证 **ServiceMonitor** 资源是否已创建并观察输出：

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

输出示例

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

4.3. 为 CLUSTER OBSERVABILITY OPERATOR 创建 MONITORINGSTACK 对象

要提取目标 **prometheus-coo-example-app** 服务公开的指标数据，请创建一个 **MonitoringStack** 对象，该对象引用您在"指定如何监控 Cluster Observability Operator"部分中创建的 **ServiceMonitor** 对象。然后，此 **MonitoringStack** 对象可以发现服务并从中提取公开的指标数据。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有命名空间管理权限的用户身份访问集群。
- 已安装 Cluster Observability Operator。
- 您已在 **ns1-coo** 命名空间中部署了 **prometheus-coo-example-app** 示例服务。
- 您已在 **ns1-coo** 命名空间中创建一个名为 **prometheus-coo-example-monitor** 的 **ServiceMonitor** 对象。

流程

1. 为 **MonitoringStack** 对象配置创建一个 YAML 文件。在本例中，将文件命名为 **example-coo-monitoring-stack.yaml**。
2. 添加以下 **MonitoringStack** 对象配置详情：

   MonitoringStack 对象示例

   ```
   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. 运行以下命令来应用 **MonitoringStack** 对象：

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

4. 运行以下命令并检查输出，验证 **MonitoringStack** 对象是否可用：

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

   输出示例

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