OpenShift Container Platform 4.9

监控

在 OpenShift Container Platform 中配置和使用监控堆栈

Last Updated: 2023-04-07

OpenShift Container Platform 4.9 监控

在 OpenShift Container Platform 中配置和使用监控堆栈

法律通告

摘要

本文提供有关在 OpenShift Container Platform 中配置和使用 Prometheus 监控堆栈的说明。

目录

第 1 章 监控概述

1.1. 关于 OPENSHIFT CONTAINER PLATFORM 监控

OpenShift Container Platform 包括一个预配置、预安装和自我更新的监控堆栈，可为核心平台组件提供监控。您还可以选择为用户定义的项目启用监控。

集群管理员可以使用支持的配置对监控堆栈进行配置。OpenShift Container Platform 提供了与监控相关的现成的最佳实践。

其中默认包括一组警报，可立即就集群问题通知集群管理员。OpenShift Container Platform Web 控制台中的默认仪表板包括集群指标的直观表示，以帮助您快速了解集群状态。

使用 OpenShift Container Platform Web 控制台，您可以 查看和管理指标、警报，并查看监控仪表板。OpenShift Container Platform 还提供对第三方接口 （如 Prometheus、Alertmanager 和 Grafana）的访问。

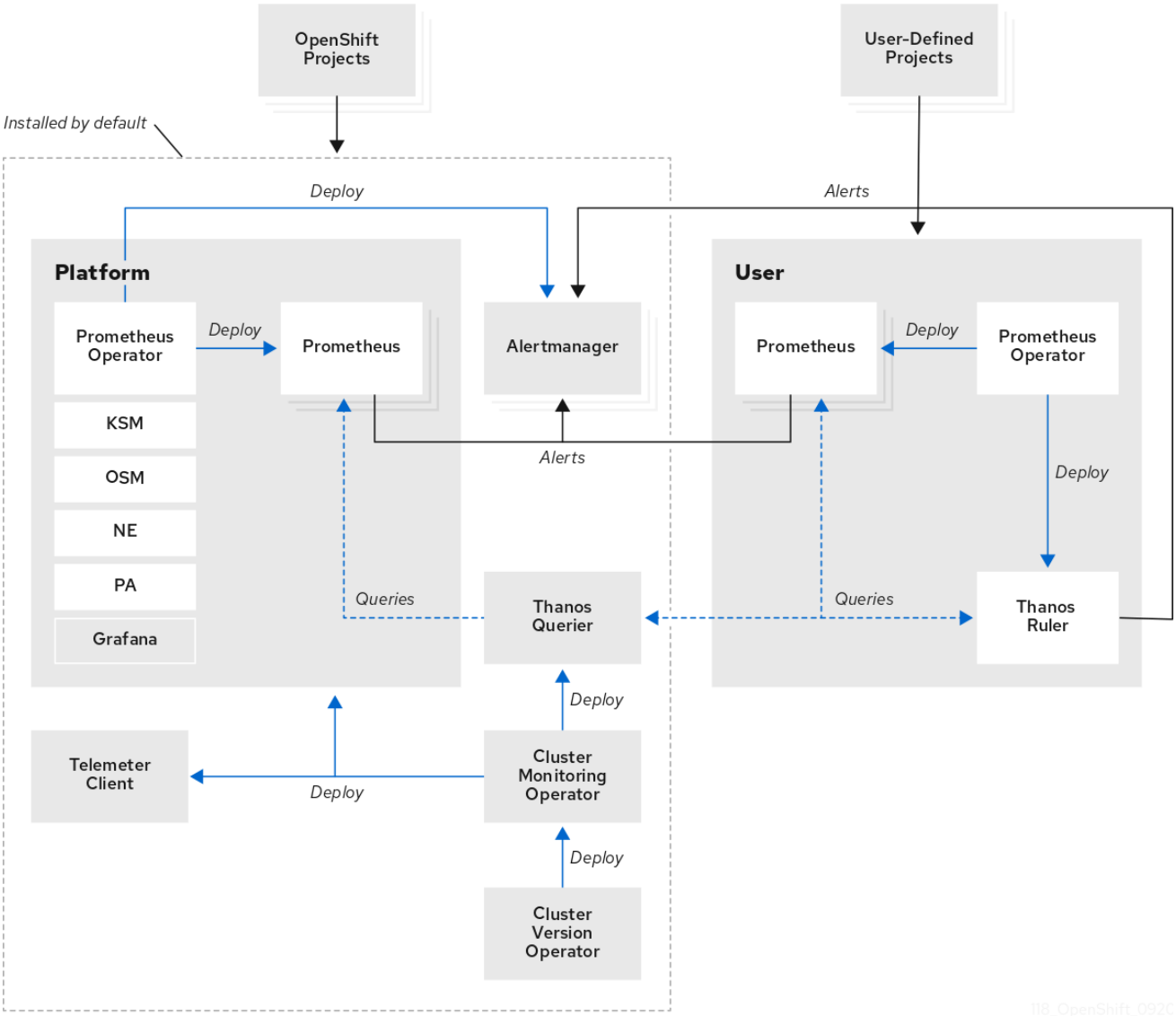
安装 OpenShift Container Platform 4.9 后，集群管理员可以选择性地为用户定义的项目启用监控。通过使用此功能，集群管理员、开发人员和其他用户可以指定在其自己的项目中如何监控服务和 Pod。作为集群管理员，您可以查找常见问题的答案，如不可用的用户指标，Prometheus 在 对监控问题进行故障排除时 消耗了大量磁盘空间。

1.2. 了解监控堆栈

OpenShift Container Platform 监控堆栈基于 Prometheus 开源项目及其更广的生态系统。监控堆栈包括以下组件：

- **默认平台监控组件**。在 OpenShift Container Platform 安装过程中，默认会在 **openshift-monitoring** 项目中安装一组平台监控组件。这为包括 Kubernetes 服务在内的 OpenShift Container Platform 核心组件提供了监控。默认监控堆栈还为集群启用远程健康状态监控。下图中的**默认安装**部分说明了这些组件。

- **用于监控用户定义项目的组件**。在选择性地为用户定义的项目启用监控后，会在 **openshift-user-workload-monitoring** 项目中安装其他监控组件。这为用户定义的项目提供了监控。下图中的**用户**部分说明了这些组件。



1.2.1. 默认监控组件

默认情况下，OpenShift Container Platform 4.9 监控堆栈包括以下组件：

表 1.1. 默认监控堆栈组件

组件	描述
Cluster Monitoring Operator	Cluster Monitoring Operator (CMO) 是监控堆栈的核心组件。它部署、管理和自动更新 Prometheus 和 Alertmanager 实例、Thanos Querier、Teleme Client 和 metrics 目标。CMO 由 Cluster Version Operator (CVO) 部署。
Prometheus Operator	**openshift-monitoring** 项目中的 Prometheus Operator (PO) 负责创建、配置和管理平台 Prometheus 实例和 Alertmanager 实例。它还会根据 Kubernetes 标签查询来自动生成监控目标配置。

组件	描述
Prometheus	Prometheus 是 OpenShift Container Platform 监控堆栈所依据的监控系统。Prometheus 是一个时间序列数据库和用于指标的规则评估引擎。Prometheus 将警报发送到 Alertmanager 进行处理。
Prometheus Adapter	Prometheus Adapter（上图中的 PA）负责转换 Kubernetes 节点和 Pod 查询以便在 Prometheus 中使用。转换的资源指标包括 CPU 和内存使用率指标。Prometheus Adapter 会公开用于 Pod 横向自动扩展的集群资源指标 API。Prometheus Adapter 也用于 **oc adm top nodes** 和 **oc adm top pods** 命令。
Alertmanager	Alertmanager 服务处理从 Prometheus 接收的警报。Alertmanager 还负责将警报发送到外部通知系统。
kube-state-metrics 代理	**kube-state-metrics** 导出器代理（上图中的 KSM）将 Kubernetes 对象转换为 Prometheus 可使用的指标。
openshift-state-metrics 代理	**openshift-state-metrics** 导出器（上图中的 OSM）通过添加了对特定 OpenShift Container Platform 资源的指标数据扩展了 **kube-state-metrics**。
node-exporter 代理	**node-exporter** 代理（上图中的 NE）负责收集有关集群中每个节点的指标。**node-exporter** 代理部署在每个节点上。
Thanos querier	Thanos Querier 将 OpenShift Container Platform 核心指标和用于用户定义项目的指标聚合在单个多租户接口下，并选择性地进行重复数据删除。
Grafana	Grafana 分析平台提供用于分析和直观呈现指标的仪表板。由监控堆栈提供的 Grafana 实例及其仪表板是只读的。
Telemeter Client	Telemeter Client 将数据的子部分从平台 Prometheus 实例发送到红帽，以便为集群提供远程健康状态监控。

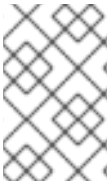
监控堆栈中的所有组件都由堆栈监控，并在 OpenShift Container Platform 更新时自动更新。

1.2.2. 默认监控目标

除了堆栈本身的组件外，默认监控堆栈还监控：

- CoreDNS

- Elasticsearch（如果安装了 Logging）
- etcd
- Fluentd（如果安装了 Logging）
- HAProxy
- 镜像 registry
- Kubelets
- Kubernetes API 服务器
- Kubernetes 控制器管理器
- Kubernetes 调度程序
- Metering（如果安装了 Metering）
- OpenShift API 服务器
- OpenShift Controller Manager
- Operator Lifecycle Manager (OLM)



注意

每个 OpenShift Container Platform 组件负责自己的监控配置。对于 OpenShift Container Platform 组件监控的问题，请针对具体组件（而非常规的监控组件）创建一个Jira 程序错误报告。

其他 OpenShift Container Platform 框架组件也可能会公开指标。如需详细信息，请参阅相应的文档。

1.2.3. 用于监控用户定义的项目的组件

OpenShift Container Platform 4.9 包括对监控堆栈的可选增强，供您用于监控用户定义的项目中的服务和 Pod。此功能包括以下组件：

表 1.2. 用于监控用户定义的项目的组件

组件	描述
Prometheus Operator	**openshift-user-workload-monitoring** 项目中的 Prometheus Operator (PO) 在同一项目中创建、配置和管理 Prometheus 和 Thanos Ruler 实例。
Prometheus	Prometheus 是为用户定义的项目提供监控的监控系统。Prometheus 将警报发送到 Alertmanager 进行处理。

组件	描述
Thanos Ruler	Thanos Ruler 是 Prometheus 的一个规则评估引擎，作为一个独立的进程来部署。在 OpenShift Container Platform 4.9 中，Thanos Ruler 为监控用户定义的项目提供规则和警报评估。

注意

在为用户定义的项目启用监控后，会部署上表中的组件。

监控堆栈中的所有组件都由堆栈监控，并在 OpenShift Container Platform 更新时自动更新。

1.2.4. 用户定义的项目的监控目标

为用户定义的项目启用监控后，您可以监控：

- 通过用户定义的项目中的服务端点提供的指标。
- 在用户定义的项目中运行的 Pod。

1.3. OPENSHIFT CONTAINER PLATFORM 监控的常见术语表

此术语表定义了 OpenShift Container Platform 架构中使用的常见术语。

Alertmanager

Alertmanager 处理从 Prometheus 接收的警报。Alertmanager 还负责将警报发送到外部通知系统。

警报规则

警报规则包含一组概述集群中特定状态的条件。当这些条件满足时会触发警报。可为警报规则分配一个严重性来定义警报的路由方式。

Cluster Monitoring Operator

Cluster Monitoring Operator (CMO) 是监控堆栈的核心组件。它部署和管理 Prometheus 实例，如 Thanos Querier、Telemeter Client 和 metrics 目标，以确保它们保持最新状态。CMO 由 Cluster Version Operator (CVO) 部署。

Cluster Version Operator

Cluster Version Operator (CVO) 管理集群 Operator 的生命周期，其中许多默认安装在 OpenShift Container Platform 中。

配置映射

配置映射提供将配置数据注入 pod 的方法。您可以在类型为 **ConfigMap** 的卷中引用存储在配置映射中的数据。在 pod 中运行的应用程序可以使用这个数据。

Container

容器是一个轻量级的可执行镜像，包括软件及其所有依赖项。容器将虚拟化操作系统。因此，您可以在数据中心、公共或私有云以及开发人员的笔记本电脑中运行容器。

自定义资源 (CR)

CR 是 Kubernetes API 的扩展。您可以创建自定义资源。

etcd

etcd 是 OpenShift Container Platform 的键值存储，它存储所有资源对象的状态。

Fluentd

Fluentd 从节点收集日志并将其传送到 Elasticsearch。

Kubelets

在节点上运行并读取容器清单。确保定义的容器已启动且正在运行。

Kubernetes API 服务器

Kubernetes API 服务器验证并配置 API 对象的数据。

Kubernetes 控制器管理器

Kubernetes 控制器管理器管理集群的状态。

Kubernetes 调度程序

Kubernetes 调度程序将 pod 分配给节点。

labels

标签是可用于组织和选择对象子集（如 pod）的键值对。

Metering

Metering 是一个通用数据分析工具，您可使用该工具编写报告，以处理来自不同数据源的数据。

node

OpenShift Container Platform 集群中的 worker 机器。节点是虚拟机 (VM) 或物理计算机。

Operator

在 OpenShift Container Platform 集群中打包、部署和管理 Kubernetes 应用程序的首选方法。Operator 将人类操作知识编码到一个软件程序中，易于打包并与客户共享。

Operator Lifecycle Manager (OLM)

OLM 可帮助您安装、更新和管理 Kubernetes 原生应用程序的生命周期。OLM 是一个开源工具包，用于以有效、自动化且可扩展的方式管理 Operator。

持久性存储

即便在设备关闭后也存储数据。Kubernetes 使用持久性卷来存储应用程序数据。

持久性卷声明 (PVC)

您可以使用 PVC 将 PersistentVolume 挂载到 Pod 中。您可以在不了解云环境的详情的情况下访问存储。

pod

pod 是 Kubernetes 中的最小逻辑单元。pod 由一个或多个容器组成，可在 worker 节点上运行。

Prometheus

Prometheus 是 OpenShift Container Platform 监控堆栈所依据的监控系统。Prometheus 是一个时间序列数据库和用于指标的规则评估引擎。Prometheus 将警报发送到 Alertmanager 进行处理。

Prometheus adapter

Prometheus Adapter 会转换 Kubernetes 节点和 pod 查询以便在 Prometheus 中使用。转换的资源指标包括 CPU 和内存使用率。Prometheus Adapter 会公开用于 Pod 横向自动扩展的集群资源指标 API。

Prometheus Operator

openshift-monitoring 项目中的 Prometheus Operator (PO) 负责创建、配置和管理平台 Prometheus 和 Alertmanager 实例。它还会根据 Kubernetes 标签查询来自动生成监控目标配置。

静默

可对警报应用静默，以防止在警报条件满足时发送通知。在您着手处理根本问题的同时，您可在初始通知后将警报静音。

storage

OpenShift Container Platform 支持许多类型的存储，包括内部存储和云供应商。您可以在 OpenShift Container Platform 集群中管理持久性和非持久性数据的容器存储。

Thanos Ruler

Thanos Ruler 是 Prometheus 的一个规则评估引擎，作为一个独立的进程来部署。在 OpenShift Container Platform 中，Thanos Ruler 为监控用户定义的项目提供规则和警报评估。

Web 控制台

用于管理 OpenShift Container Platform 的用户界面(UI)。

1.4. 其他资源

- 关于远程健康监控
- 授予用户权限来监控用户定义的项目

1.5. 后续步骤

- 配置监控堆栈

第 2 章 配置监控堆栈

安装之前，OpenShift Container Platform 4 安装程序只提供少量的配置选项。大多数 OpenShift Container Platform 框架组件（包括集群监控堆栈）都在安装后进行配置。

本节介绍支持的配置，演示如何配置监控堆栈，并且展示几个常见的配置情景。

2.1. 先决条件

- 监控堆栈会带来额外的资源需求。请参考缩放 Cluster Monitoring Operator 中的计算资源建议，并验证您是否有充足的资源。

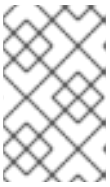
2.2. 对监控的维护和支持

若要配置 OpenShift Container Platform Monitoring，支持的方式是使用本文中介绍的选项。**请勿使用其他配置，因为不受支持。**各个 Prometheus 发行版本的配置范例可能会有所变化，只有掌握了所有可能的配置，才能稳妥应对这样的配置变化。如果使用并非本节所描述的配置，您的更改可能会丢失，因为 **cluster-monitoring-operator** 会调节差异。根据设计，Operator 默认将一切重置到定义的状态。

2.2.1. 对监控的支持注意事项

明确不支持以下修改：

- **在 openshift-* 和 kube-* 项目中创建额外的 ServiceMonitor、PodMonitor 和 PrometheusRule 对象。**
- **修改 openshift-monitoring 或 openshift-user-workload-monitoring 项目中部署的任何资源或对象。**OpenShift Container Platform 监控堆栈所创建的资源并不是为了供任何其他资源使用，因为不能保证向后兼容性。



注意

Alertmanager 配置作为 secret 资源部署在 **openshift-monitoring** 项目中。要为 Alertmanager 配置额外的路由，您需要对该 secret 进行解码、修改，然后再进行编码。该程序是对前述声明的一个受支持例外。

- **修改堆栈的资源。**OpenShift Container Platform 监控堆栈确保其资源始终处于期望的状态。如果修改了资源，堆栈会重置它们。
- **将用户定义的工作负载部署到 openshift-* 和 kube-* 项目。**这些项目是为红帽提供的组件保留的，不应该用于用户定义的工作负载。
- **修改监控堆栈 Grafana 实例。**
- **在 OpenShift Container Platform 上安装自定义 Prometheus 实例。**自定义资源 (CR) 是由 Prometheus Operator 管理的 Prometheus 自定义资源 (CR)。
- **使用 Prometheus Operator 中的 Probe 自定义资源定义（CRD）启用基于症状的监控。**
- **使用 Prometheus Operator 中的 AlertmanagerConfig CRD 修改 Alertmanager 配置。**

注意

指标、记录规则或警报规则的向后兼容性无法被保证。

2.2.2. 监控 Operator 的支持策略

监控 Operator 确保 OpenShift Container Platform 监控资源按设计和测试的方式正常工作。如果某个 Operator 的 Cluster Version Operator (CVO) 控制被覆盖，该 Operator 不会响应配置更改，协调集群对象的预期状态或接收更新。

虽然在调试过程中覆盖 Operator 的 CVO 可能有所帮助，但该操作不受支持，集群管理员需要完全掌控各个组件的配置和升级。

覆盖 Cluster Version Operator

可将 **spec.overrides** 参数添加到 CVO 的配置中，以便管理员提供对组件的 CVO 行为覆盖的列表。将一个组件的 **spec.overrides[].unmanaged** 参数设置为 **true** 会阻止集群升级并在设置 CVO 覆盖后提醒管理员：

> Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

警告

设置 CVO 覆盖会使整个集群处于不受支持的状态，并导致监控堆栈无法被协调到其预期状态。这会影响 Operator 内置的可靠性功能，并妨碍接收更新。在删除所有覆盖后，必须可以重现报告的问题方可获得支持。

2.3. 准备配置监控堆栈

您可以通过创建和更新监控配置映射来配置监控堆栈。

2.3.1. 创建集群监控配置映射

要配置 OpenShift Container Platform 核心监控组件，您必须在 **openshift-monitoring** 项目中创建 **cluster-monitoring-config ConfigMap** 对象。

注意

当您将更改保存到 **cluster-monitoring-config ConfigMap** 对象时，可能会重新部署 **openshift-monitoring** 项目中的部分或全部 Pod。有时重新部署这些组件需要花费一段时间。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 检查 **cluster-monitoring-config ConfigMap** 对象是否存在：

```
$ oc -n openshift-monitoring get configmap cluster-monitoring-config
```

2. 如果 **ConfigMap** 对象不存在：

   a. 创建以下 YAML 清单。在本例中，该文件名为 **cluster-monitoring-config.yaml**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
```

   b. 应用配置以创建 **ConfigMap** 对象：

```
$ oc apply -f cluster-monitoring-config.yaml
```

2.3.2. 创建用户定义的工作负载监控配置映射

要配置用于监控用户定义项目的组件，您必须在 **openshift-user-workload-monitoring** 项目中创建 **user-workload-monitoring-config ConfigMap**。

注意

当您将更改保存到 **user-workload-monitoring-config ConfigMap** 对象时，可能会重新部署 **openshift-user-workload-monitoring** 项目中的部分或全部 Pod。有时重新部署这些组件需要花费一段时间。在首次为用户定义的项目启用监控前，您可以创建和配置配置映射，以防止经常重新部署 pod。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 检查 **user-workload-monitoring-config ConfigMap** 对象是否存在：

```
$ oc -n openshift-user-workload-monitoring get configmap user-workload-monitoring-config
```

2. 如果 **user-workload-monitoring-config ConfigMap** 对象不存在：

   a. 创建以下 YAML 清单。在本例中，该文件名为 **user-workload-monitoring-config.yaml**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
```

b. 应用配置以创建 **ConfigMap** 对象：

```
$ oc apply -f user-workload-monitoring-config.yaml
```

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

其他资源

- 为用户定义的项目启用监控

2.4. 配置监控堆栈

在 OpenShift Container Platform 4.9 中，您可以使用 **cluster-monitoring-config** 或 **user-workload-monitoring-config ConfigMap** 配置监控堆栈。配置配置映射配置 Cluster Monitoring Operator（CMO），CMO 会配置堆栈的组件。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象。
   - **要配置 OpenShift Container Platform 核心监控组件：**
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. 将您的配置以键值对 **<component_name>: <component_configuration>** 的形式添加到 **data/config.yaml** 下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
```

```
data:
  config.yaml: |
    <component>:
      <configuration_for_the_component>
```

相应地替换 **<component>** 和 **<configuration_for_the_component>**。

以下示例 **ConfigMap** 对象为 Prometheus 配置持久性卷声明（PVC）。这与只监控 OpenShift Container Platform 核心组件的 Prometheus 实例相关：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s: 1
      volumeClaimTemplate:
       spec:
         storageClassName: fast
         volumeMode: Filesystem
         resources:
           requests:
             storage: 40Gi
```

1 定义 Prometheus 组件，后面几行则定义其配置。

- **要配置用于监控用户定义的项目的组件**：

  a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. 将您的配置以键值对 **<component_name>: <component_configuration>** 的形式添加到 **data/config.yaml** 下：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      <component>:
        <configuration_for_the_component>
  ```

  相应地替换 **<component>** 和 **<configuration_for_the_component>**。

  以下示例 **ConfigMap** 对象为 Prometheus 配置数据保留周期和最低容器资源请求。这与仅监控用户定义的项目的 Prometheus 实例相关：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus: 1
      retention: 24h 2
      resources:
        requests:
          cpu: 200m 3
          memory: 2Gi 4
```

1. 定义 Prometheus 组件，后面几行则定义其配置。
2. 为监控用户定义的项目的 Prometheus 实例配置 24 小时的数据保留周期。
3. 为 Prometheus 容器定义最低 200 毫秒的资源请求。
4. 为 Prometheus 容器定义最低 2 GiB 内存的 Pod 资源请求。

注意

Prometheus 配置映射组件在 **cluster-monitoring-configConfigMap** 对象中被称为 **prometheusK8s**，在 **user-workload-monitoring-config ConfigMap** 对象中称为 **prometheus**。

2. 保存文件以将更改应用到 **ConfigMap** 对象。受新配置影响的 Pod 会自动重启。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控

2.5. 可配置的监控组件

下表显示了您可以配置的监控组件，以及 **cluster-monitoring-config** 和 **user-workload-monitoring-config ConfigMap** 中用来指定这些组件的键：

表 2.1. 可配置的监控组件

组件	cluster-monitoring-config 配置映射键	user-workload-monitoring-config 配置映射键
Prometheus Operator	**prometheusOperator**	**prometheusOperator**
Prometheus	**prometheusK8s**	**prometheus**
Alertmanager	**alertmanagerMain**	
kube-state-metrics	**kubeStateMetrics**	
openshift-state-metrics	**openshiftStateMetrics**	
Grafana	**grafana**	
Telemeter Client	**telemeterClient**	
Prometheus Adapter	**k8sPrometheusAdapter**	
Thanos querier	**thanosQuerier**	
Thanos Ruler		**thanosRuler**

注意

Prometheus 键在 **cluster-monitoring-config ConfigMap** 对象中称为 **prometheusK8s**，在 **user-workload-monitoring-config ConfigMap** 对象中称为 **prometheus**。

2.6. 将监控组件移到其他节点

您可以将任何监控堆栈组件移到特定的节点。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。

- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。

- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：

   - **要移动用于监控 OpenShift Container Platform 核心项目的组件**：

     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

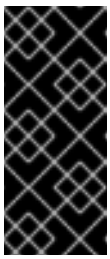
     ```
     $ oc -n openshift-monitoring edit configmap cluster-monitoring-config
     ```

     b. 在 **data/config.yaml** 下为组件指定 **nodeSelector** 约束：

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
       name: cluster-monitoring-config
       namespace: openshift-monitoring
     data:
       config.yaml: |
         <component>:
           nodeSelector:
             <node_key>: <node_value>
             <node_key>: <node_value>
             <...>
     ```

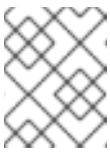
     相应地替换 **<component>**，并将 **<node_key>: <node_value>** 替换为用于指定目标节点组的键值对映射。通常只使用一个键值对。

     组件只能在以各个指定键值对作为标签的节点上运行。节点也可以有附加标签。

     重要

     许多监控组件都通过在集群中的不同节点间使用多个 Pod 来部署，以维持高可用性。将监控组件移到带标签的节点时，确保有足够的匹配节点来保持组件的弹性。如果只指定了一个标签，请确保有足够的节点包含该标签，以便将该组件的所有 Pod 分布到不同的节点。另外，您还可以指定多个标签，每个都与单个节点相关。

     注意

     如果在配置 **nodeSelector** 约束后监控组件仍然处于 **Pending** 状态，请检查 Pod 日志中与污点和容限相关的错误。

     例如，要将 OpenShift Container Platform 核心项目的监控组件移到带有 **nodename: controlplane1**、**nodename: worker1**、**nodename: worker2** 和 **nodename: worker2** 标签的特定节点中，请使用：

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
     ```

```
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: controlplane1
    prometheusK8s:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    alertmanagerMain:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    kubeStateMetrics:
      nodeSelector:
        nodename: worker1
    grafana:
      nodeSelector:
        nodename: worker1
    telemeterClient:
      nodeSelector:
        nodename: worker1
    k8sPrometheusAdapter:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    openshiftStateMetrics:
      nodeSelector:
        nodename: worker1
    thanosQuerier:
      nodeSelector:
        nodename: worker1
        nodename: worker2
```

- **要移动用于监控用户定义的项目的组件：**

  a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

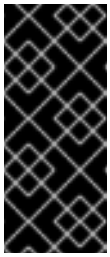
  b. 在 **data/config.yaml** 下为组件指定 **nodeSelector** 约束：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      <component>:
        nodeSelector:
  ```

```
      <node_key>: <node_value>
      <node_key>: <node_value>
      <...>
```

相应地替换 **<component>**，并将 **<node_key>: <node_value>** 替换为用于指定目标节点的键值对映射。通常只使用一个键值对。

组件只能在以各个指定键值对作为标签的节点上运行。节点也可以有附加标签。



重要

许多监控组件都通过在集群中的不同节点间使用多个 Pod 来部署，以维持高可用性。将监控组件移到带标签的节点时，确保有足够的匹配节点来保持组件的弹性。如果只指定了一个标签，请确保有足够的节点包含该标签，以便将该组件的所有 Pod 分布到不同的节点。另外，您还可以指定多个标签，每个都与单个节点相关。

注意

如果在配置 **nodeSelector** 约束后监控组件仍然处于 **Pending** 状态，请检查 Pod 日志中与污点和容限相关的错误。

例如，要将监控用户定义的项目的组件移到带有 **nodename: worker1**、**nodename: worker2** 和 **nodename: worker2** 标签的特定 worker 节点，请使用：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: worker1
    prometheus:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    thanosRuler:
      nodeSelector:
        nodename: worker1
        nodename: worker2
```

2. 保存文件以使改变生效。受新配置影响的组件会自动移到新节点上。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控
- 了解如何更新节点上的标签
- 使用节点选择器将 pod 放置到特定节点
- 参阅 Kubernetes 文档来详细了解 **nodeSelector** 约束

2.7. 为监控组件分配容忍（TOLERATIONS）

您可以为任何监控堆栈组件分配容忍，以便将其移到污点。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：
   - **要将容限分配给监控 OpenShift Container Platform 核心项目的组件：**
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

     ```
     $ oc -n openshift-monitoring edit configmap cluster-monitoring-config
     ```

     b. 为组件指定 **tolerations**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>:
      tolerations:
        <toleration_specification>
```

相应地替换 **<component>** 和 **<toleration_specification>**。

例如，**oc adm taint nodes node1 key1=value1:NoSchedule** 会将一个键为 **key1** 且值为 **value1** 的污点添加到 **node1**。这会防止监控组件在 **node1** 上部署 Pod，除非为该污点配置了容限。以下示例将 **alertmanagerMain** 组件配置为容许示例污点：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      tolerations:
      - key: "key1"
        operator: "Equal"
        value: "value1"
        effect: "NoSchedule"
```

- **要将容限分配给监控用户定义的项目的组件：**

  a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. 为组件指定 **tolerations**：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      <component>:
        tolerations:
          <toleration_specification>
  ```

  相应地替换 **<component>** 和 **<toleration_specification>**。

例如，**oc adm taint nodes node1 key1=value1:NoSchedule** 会将一个键为 **key1** 且值为 **value1** 的污点添加到 **node1**。这会防止监控组件在 **node1** 上部署 Pod，除非为该污点配置了容限。以下示例将 **thanosRuler** 组件配置为容许示例污点：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      tolerations:
      - key: "key1"
        operator: "Equal"
        value: "value1"
        effect: "NoSchedule"
```

2. 保存文件以使改变生效。这样就会自动应用新组件放置配置。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控
- 参阅 OpenShift Container Platform 文档中有关污点和容忍的内容。
- 参阅 Kubernetes 文档中有关污点和容限的内容

2.8. 配置持久性存储

如果使用持久性存储运行集群监控，您的指标将保存在持久性卷（PV）中，并可在 Pod 重新启动或重新创建后保留。如果您需要预防指标或警报数据丢失，这是理想方案。在生产环境中，强烈建议配置持久性存储。由于 IO 需求很高，使用本地存储颇有优势。

注意

请参阅建议的可配置存储技术。

2.8.1. 持久性存储的先决条件

- 分配充足的专用本地持久性存储，以确保磁盘不会被填满。您需要的存储量取决于 Pod 的数目。如需有关持久性存储系统要求的信息，请参阅 Prometheus 数据库存储要求。
- 确保持久性卷 (PV) 已准备好以供持久性卷声明 (PVC) 使用，每个副本一个 PV。由于 Prometheus 有两个副本并且 Alertmanager 有三个副本，因此您需要五个 PV 来支持整个监控堆栈。PV 应该从 Local Storage Operator 中提供。如果启用了动态置备的存储，则这项要求不适用。
- 在配置持久性卷时，使用 **Filesystem** 作为 **volumeMode** 参数的存储类型值。
- 配置本地持久性存储。

注意

如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Prometheus 无法使用原始块卷。

2.8.2. 配置本地持久性卷声明

要让监控组件使用持久性卷 (PV)，您必须配置持久性卷声明 (PVC)。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：
   - 为监控 OpenShift Container Platform 核心项目的组件配置 PVC：
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. 将组件的 PVC 配置添加到 **data/config.yaml** 下：

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>:
      volumeClaimTemplate:
        spec:
          storageClassName: <storage_class>
          resources:
            requests:
              storage: <amount_of_storage>
```

如需有关如何指定 **volumeClaimTemplate** 的信息，请参阅 Kubernetes 文档中与 PersistentVolumeClaim 相关的内容。

以下示例配置了一个 PVC 来声明用于监控 OpenShift Container Platform 核心组件的 Prometheus 实例的本地持久性存储：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

在上例中，由 Local Storage Operator 创建的存储类称为 **local-storage**。

以下示例配置了一个 PVC 来声明用于 Alertmanager 的本地持久性存储：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 10Gi
```

- **要为监控用户定义的项目的组件配置 PVC：**

a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

b. 将组件的 PVC 配置添加到 **data/config.yaml** 下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>:
      volumeClaimTemplate:
        spec:
          storageClassName: <storage_class>
          resources:
            requests:
              storage: <amount_of_storage>
```

如需有关如何指定 **volumeClaimTemplate** 的信息，请参阅 Kubernetes 文档中与 PersistentVolumeClaim 相关的内容。

以下示例配置了一个 PVC 来为监控用户定义的项目的 Prometheus 实例声明本地持久性存储：

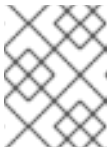
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

在上例中，由 Local Storage Operator 创建的存储类称为 **local-storage**。

以下示例配置了一个 PVC 来声明用于 Thanos Ruler 的本地持久性存储：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
```

```
thanosRuler:
  volumeClaimTemplate:
    spec:
      storageClassName: local-storage
      resources:
        requests:
          storage: 10Gi
```



注意

thanosRuler 组件的存储要求取决于要评估的规则数量以及每个规则生成的样本数量。

2. 保存文件以使改变生效。受新配置影响的 Pod 会自动重启，并且应用新的存储配置。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

2.8.3. 重新定义持久性存储卷的大小

OpenShift Container Platform 不支持重新定义 **StatefulSet** 资源使用的现有持久性存储卷的大小，即使底层 **StorageClass** 资源支持持久性卷大小。因此，即使为具有更大大小的现有持久性卷声明 (PVC) 更新 **storage** 字段，这个设置也不会传播到关联的持久性卷 (PV)。

但是，仍可使用手动过程重新定义 PV 的大小。如果要为监控组件（如 Prometheus、Thanos Ruler 或 Alertmanager）重新定义 PV 的大小，您可以更新配置该组件的相应配置映射。然后，修补 PVC 并删除 pod。Orphaning pod 立即重新创建 **StatefulSet** 资源，并使用新的 PVC 设置自动更新 pod 中挂载的卷大小。此过程中不会发生服务中断。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- **如果要配置 OpenShift Container Platform 核心监控组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
  - 至少有一个 PVC 用于 OpenShift Container Platform 核心监控组件。
- **如果您要配置用于监控用户定义的项目的组件**：

- 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 至少有一个 PVC 用于监控用户定义的项目的组件。

流程

1. 编辑 **ConfigMap** 对象：
   - **为监控 OpenShift Container Platform 核心项目的组件重新定义 PVC 的大小**：
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. 在 **data/config.yaml** 下为组件添加新存储大小：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>: 1
      volumeClaimTemplate:
        spec:
          storageClassName: <storage_class> 2
          resources:
            requests:
              storage: <amount_of_storage> 3
```

1 指定核心监控组件。

2 指定存储类。

3 指定存储卷的新大小。

以下示例配置了一个 PVC，它将监控 OpenShift Container Platform 核心组件的 Prometheus 实例的本地持久性存储设置为 100GB：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      volumeClaimTemplate:
        spec:
```

```
      storageClassName: local-storage
      resources:
        requests:
          storage: 100Gi
```

以下示例配置了一个 PVC，将 Alertmanager 的本地持久性存储设置为 40GB：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

- **为监控用户定义的项目的组件调整 PVC 大小：**

注意

您可以调整监控用户定义的项目的 Thanos Ruler 和 Prometheus 实例的大小。

a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

b. 在 **data/config.yaml** 下更新监控组件的 PVC 配置：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: 1
      volumeClaimTemplate:
        spec:
          storageClassName: <storage_class> 2
          resources:
            requests:
              storage: <amount_of_storage> 3
```

1 指定核心监控组件。

2 指定存储类。

3 指定存储卷的新大小。

以下示例将监控用户定义的项目的 Prometheus 实例的 PVC 大小配置为 100GB：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 100Gi
```

以下示例将 Thanos Ruler 的 PVC 大小设置为 20GB：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 20Gi
```

注意

thanosRuler 组件的存储要求取决于要评估的规则数量以及每个规则生成的样本数量。

2. 保存文件以使改变生效。受新配置重启影响的 Pod 会自动重启。

警告

当您将更改保存到监控配置映射时，可能会重新部署相关项目中的 Pod 和其他资源。该项目中运行的监控进程也可能被重启。

3. 使用更新的存储请求手动对每个 PVC 进行补丁。以下示例将 **openshift-monitoring** 命名空间中的 Prometheus 组件的存储大小调整为 100Gi：

```
$ for p in $(oc -n openshift-monitoring get pvc -l app.kubernetes.io/name=prometheus -o
jsonpath='{range .items[*]}{.metadata.name} {end}'); do \
  oc -n openshift-monitoring patch pvc/${p} --patch '{"spec": {"resources": {"requests":
{"storage":"100Gi"}}}}'; \
  done
```

4. 使用 **--cascade=orphan** 参数删除底层 StatefulSet:

```
$ oc delete statefulset -l app.kubernetes.io/name=prometheus --cascade=orphan
```

2.8.4. 修改 Prometheus 指标数据的保留时间

默认情况下，OpenShift Container Platform 监控堆栈将 Prometheus 数据的保留时间配置为 15 天。您可以修改保留时间来更改将在多久后删除数据。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：
   - **要修改用于监控 OpenShift Container Platform 核心项目的 Prometheus 实例的保留时间：**
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. 将保留时间配置放添加到 **data/config.yaml** 下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
```

```
  config.yaml: |
    prometheusK8s:
      retention: <time_specification>
```

将 **<time_specification>** 替换为一个数字，后面紧跟 **ms**（毫秒）、**s**（秒）、**m**（分钟）、**h**（小时）、**d**（天）、**w**（周）或 **y**（年）。

以下示例将监控 OpenShift Container Platform 核心组件的 Prometheus 实例的保留时间设置为 24 小时：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: 24h
```

- **要为监控用户定义的项目的 Prometheus 实例修改保留时间：**

  a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. 将保留时间配置放添加到 **data/config.yaml** 下：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      prometheus:
        retention: <time_specification>
  ```

  将 **<time_specification>** 替换为一个数字，后面紧跟 **ms**（毫秒）、**s**（秒）、**m**（分钟）、**h**（小时）、**d**（天）、**w**（周）或 **y**（年）。

  以下示例针对监控用户定义的项目的 Prometheus 实例，将保留时间设置为 24 小时：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      prometheus:
        retention: 24h
  ```

2. 保存文件以使改变生效。受新配置影响的 Pod 会自动重启。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控
- 了解持久性存储
- 优化存储

2.9. 配置远程写入存储

您可以配置远程写入存储，使 Prometheus 能够将最接近的指标发送到远程系统，以进行长期存储。这样做不会影响 Prometheus 存储指标的方式和时长。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果您要配置用于监控用户定义的项目的组件:**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI(**oc**)。
- 您已设置了一个远程写入兼容端点（如 Thanos），并且知道端点 URL。有关与远程写入功能兼容的端点的信息，请参阅 Prometheus 远程端点和存储文档。
- 您已为远程写入端点设置了身份验证凭证。

小心

要降低安全风险，请避免在不使用加密 HTTP 的情况下通过未加密的 HTTP 向端点发送指标。

流程

1. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

2. 在 **data/config.yaml/prometheusK8s** 下添加一个 **remoteWrite:** 部分。

3. 在本节中添加端点 URL 和身份验证凭证：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
      - url: "https://remote-write.endpoint"
        <endpoint_authentication_credentials>
```

对于 **endpoint_authentication_credentials**，请替换端点的凭据。目前支持的身份验证方法是基本身份验证 (**basicAuth**) 和客户端 TLS (**tlsConfig**) 身份验证。

- 以下示例配置基本身份验证：

```
basicAuth:
  username:
    <usernameSecret>
  password:
    <passwordSecret>
```

相应地替换 **<usernameSecret>** 和 **<passwordSecret>**。

以下示例显示了基本的身份验证配置，**remoteWriteAuth** 是 **name** 值，**user** 和 **password** 是 **key** 值。这些值包含端点身份验证凭证：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
      - url: "https://remote-write.endpoint"
        basicAuth:
          username:
            name: remoteWriteAuth
```

```
            key: user
          password:
            name: remoteWriteAuth
            key: password
```

- 以下示例配置客户端 TLS 身份验证：

```
tlsConfig:
  ca:
    <caSecret>
  cert:
    <certSecret>
  keySecret:
    <keySecret>
```

相应地替换 **<caSecret>**、**<certSecret>** 和 **<keySecret>**。

以下示例显示了一个 TLS 验证配合，使用 **selfsigned-mtls-bundle** 作为 **name** 值，**ca.crt** 为 **ca key** 值，**client.crt** 为 **cert key** 值，**client.key** 为 **keySecret key** 值：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
      - url: "https://remote-write.endpoint"
        tlsConfig:
          ca:
            secret:
              name: selfsigned-mtls-bundle
              key: ca.crt
          cert:
            secret:
              name: selfsigned-mtls-bundle
              key: client.crt
          keySecret:
            name: selfsigned-mtls-bundle
            key: client.key
```

4. 在身份验证凭证后添加 write relabel 配置值：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
```

```
    - url: "https://remote-write.endpoint"
      <endpoint_authentication_credentials>
      <write_relabel_configs>
```

对于 **<write_relabel_configs>**，请替换您要发送到远程端点的指标写入重新标记配置列表。

以下示例演示了如何转发名为 **my_metric** 的单个指标：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
      - url: "https://remote-write.endpoint"
        writeRelabelConfigs:
        - sourceLabels: [__name__]
          regex: 'my_metric'
          action: keep
```

有关写入重新标记配置选项的详情，请查看 Prometheus relabel_config 文档。

5. 如果需要，通过更改 **name** and **namespace metadata** 值，为监控用户定义的项目的 Prometheus 实例配置远程写入，如下所示：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      remoteWrite:
      - url: "https://remote-write.endpoint"
        <endpoint_authentication_credentials>
        <write_relabel_configs>
```

注意

Prometheus 配置映射组件在 **cluster-monitoring-configConfigMap** 对象中被称为 **prometheusK8s**，在 **user-workload-monitoring-config ConfigMap** 对象中称为 **prometheus**。

6. 保存文件以将更改应用到 **ConfigMap** 对象。受新配置重启影响的 Pod 会自动重启。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

保存对监控 **ConfigMap** 对象的更改可能会重新部署相关项目中的 pod 和其他资源。保存更改还可能在该项目中重新启动正在运行的监控进程。

其他资源

- 如需创建远程写入兼容端点（如 Thanos）的步骤，请参阅设置远程写入兼容端点。
- 如需有关如何针对不同用例优化远程写入设置的信息，请参阅调整远程写入设置。
- 有关其他可选字段的详情，请参考 API 文档。

2.10. 控制用户定义的项目中未绑定指标属性的影响

开发人员可以使用键值对的形式为指标定义属性。潜在的键值对数量与属性的可能值数量对应。具有无限数量可能值的属性被称为未绑定属性。例如，**customer_id** 属性不绑定，因为它有无限多个可能的值。

每个分配的键值对都有唯一的时间序列。在标签中使用许多未绑定属性可导致所创建的时间序列数量出现指数增加。这可能会影响 Prometheus 性能，并消耗大量磁盘空间。

集群管理员可以使用以下方法控制用户定义的项目中未绑定指标属性的影响：

- **限制用户定义的项目中每个目标提取可接受的示例数量**
- **创建在达到提取示例阈值或无法提取目标时触发的警报**

注意

限制提取示例可帮助防止在标签中添加多个未绑定属性导致的问题。开发人员还可以通过限制其为指标定义的未绑定属性数量来防止底层原因。使用绑定到一组有限可能值的属性可减少潜在的键-值对组合数量。

2.10.1. 为用户定义的项目设置提取示例限制

您可以限制用户定义的项目中每个目标提取可接受的示例数量。

警告

如果您设置了示例限制，则在达到限制后，不会为该目标摄取更多样本数据。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。

- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

2. 在 **data/config.yaml** 中添加 **enforcedSampleLimit** 配置，以限制用户定义的项目中每个目标提取可接受的示例数量：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      enforcedSampleLimit: 50000 1
```

1 如果指定此参数，则需要一个值。这个 **enforceSampleLimit** 示例将用户定义的项目中每个目标提取的示例数量限制为 50,000。

3. 保存文件以使改变生效。该限制会自动应用。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

将更改保存到 **user-workload-monitoring-config ConfigMap** 对象时，可能会重新部署 **openshift-user-workload-monitoring** 项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

2.10.2. 创建提取示例警报

您可以创建在以下情况下通知您的警报：

- 在指定的 **for** 持续时间内无法提取对象或对象不可用
- 在指定的 **for** 持续时间内达到或超过提取示例阈值

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 您已为用户定义的项目启用了监控。
- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 您已经使用 **enforcedSampleLimit** 限制了用户定义的项目中每个目标提取可接受的示例数量。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 创建一个包含警报的 YAML 文件，用于在目标停机以及即将达到强制的示例限制时通知您。本例中的文件名为 **monitoring-stack-alerts.yaml**：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    prometheus: k8s
    role: alert-rules
  name: monitoring-stack-alerts (1)
  namespace: ns1 (2)
spec:
  groups:
  - name: general.rules
    rules:
    - alert: TargetDown (3)
      annotations:
        message: '{{ printf "%.4g" $value }}% of the {{ $labels.job }}/{{ $labels.service
          }} targets in {{ $labels.namespace }} namespace are down.' (4)
      expr: 100 * (count(up == 0) BY (job, namespace, service) / count(up) BY (job,
        namespace, service)) > 10
      for: 10m (5)
      labels:
        severity: warning (6)
    - alert: ApproachingEnforcedSamplesLimit (7)
      annotations:
        message: '{{ $labels.container }} container of the {{ $labels.pod }} pod in the {{
$labels.namespace }} namespace consumes {{ $value | humanizePercentage }} of the
samples limit budget.' (8)
      expr: scrape_samples_scraped/50000 > 0.8 (9)
      for: 10m (10)
      labels:
        severity: warning (11)
```

(1) 定义警报规则的名称。

(2) 指定要部署警报规则的用户定义的项目。

(3) 如果在 **for** 持续时间内无法提取目标或者目标不可用，则 **TargetDown** 警报将触发。

(4) **TargetDown** 警报触发时输出的消息。

5 在这个持续时间内必须满足 **TargetDown** 警报的条件才会触发该警报。

6 定义 **TargetDown** 警报的严重性。

7 当在指定的 **for** 持续时间内达到或超过定义的提取示例阈值时，**ApproachingEnforcedSamplesLimit** 警报将触发。

8 当 **ApproachingEnforcedSamplesLimit** 警报触发时输出的消息。

9 **ApproachingEnforcedSamplesLimit** 警报的阈值。在本例中，当每个目标提取的示例数量超过强制示例限制 **50000** 的 80% 时，警报将触发。在警报触发前，还必须已经过了 **for** 持续时间。表达式 **scrape_samples_scraped/<number> > <threshold>** 中的 **<number>** 必须与 **user-workload-monitoring-config ConfigMap** 对象中定义的 **enforcedSampleLimit** 值匹配。

10 在这个持续时间内必须满足 **ApproachingEnforcedSamplesLimit** 警报的条件才会触发该警报。

11 定义 **ApproachingEnforcedSamplesLimit** 警报的严重性。

2. 将配置应用到用户定义的项目中：

```
$ oc apply -f monitoring-stack-alerts.yaml
```

其他资源

- 创建用户定义的工作负载监控配置映射
- 为用户定义的项目启用监控
- 请参阅确定为什么 Prometheus 消耗大量磁盘空间以了解通过什么步骤来查询哪些指标的提取示例数最高

第 3 章 配置外部 ALERTMANAGER 实例

OpenShift Container Platform 监控堆栈包含一个本地 Alertmanager 实例，用于从 Prometheus 路由警报。您可以通过在 **openshift-monitoring** 项目或 **user-workload-monitoring-config** 项目中配置 **cluster-monitoring-config** 配置映射来添加外部 Alertmanager 实例。

如果您为多个集群添加相同的外部 Alertmanager 配置，并且为每个集群禁用本地实例，则可以使用单个外部 Alertmanager 实例管理多个集群的警报路由。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- **如果要在 openshift-monitoring 项目中配置 OpenShift Container Platform 核心监控组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建了 **cluster-monitoring-config** 配置映射。
- **如果您要配置用于监控用户定义的项目的组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config** 配置映射。

流程

1. 编辑 **ConfigMap** 对象。
   - **配置额外的 Alertmanager 以路由来自 OpenShift Container Platform 核心项目的警报**：
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config** 配置映射：

     ```
     $ oc -n openshift-monitoring edit configmap cluster-monitoring-config
     ```

     b. 在 **data/config.yaml/prometheusK8s** 下添加一个 **additionalAlertmanagerConfigs:** 小节。

     c. 在本节中添加其他 Alertmanager 的配置详情：

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
       name: cluster-monitoring-config
       namespace: openshift-monitoring
     data:
       config.yaml: |
         prometheusK8s:
           additionalAlertmanagerConfigs:
           - <alertmanager_specification>
     ```

     对于 **<alertmanager_specification>**，请替换额外的 Alertmanager 实例的身份验证和其他配置详情。目前支持的身份验证方法有 bearer 令牌 (**bearerToken**) 和客户端 TLS (**tlsConfig**)。以下示例配置映射使用 bearer 令牌和客户端 TLS 身份验证配置额外的

Alertmanager：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      additionalAlertmanagerConfigs:
      - scheme: https
        pathPrefix: /
        timeout: "30s"
        apiVersion: v1
        bearerToken:
          name: alertmanager-bearer-token
          key: token
        tlsConfig:
          key:
            name: alertmanager-tls
            key: tls.key
          cert:
            name: alertmanager-tls
            key: tls.crt
          ca:
            name: alertmanager-tls
            key: tls.ca
        staticConfigs:
        - external-alertmanager1-remote.com
        - external-alertmanager1-remote2.com
```

- **配置额外的 Alertmanager 实例以路由来自用户定义的项目的警报 ：**

  a. 编辑 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config** 配置映射：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. 在 **data/config.yaml/** 下添加一个 **<component>/additionalAlertmanagerConfigs:** 部分。

  c. 在本节中添加其他 Alertmanager 的配置详情：

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      <component>:
        additionalAlertmanagerConfigs:
        - <alertmanager_specification>
  ```

对于 **<component>**，替换两个支持的外部 Alertmanager 组件之一：**prometheus** 或 **thanosRuler**。

对于 **<alertmanager_specification>**，请替换额外的 Alertmanager 实例的身份验证和其他配置详情。目前支持的身份验证方法有 bearer 令牌 (**bearerToken**) 和客户端 TLS (**tlsConfig**)。以下示例配置映射使用带有 bearer 令牌和客户端 TLS 身份验证的 Thanos Ruler 配置额外的 Alertmanager：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
   thanosRuler:
     additionalAlertmanagerConfigs:
    - scheme: https
      pathPrefix: /
      timeout: "30s"
      apiVersion: v1
      bearerToken:
        name: alertmanager-bearer-token
        key: token
      tlsConfig:
        key:
          name: alertmanager-tls
          key: tls.key
        cert:
          name: alertmanager-tls
          key: tls.crt
        ca:
          name: alertmanager-tls
          key: tls.ca
      staticConfigs:
      - external-alertmanager1-remote.com
      - external-alertmanager1-remote2.com
```

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

2. 保存文件以将更改应用到 **ConfigMap** 对象。这样就会自动应用新组件放置配置。

3.1. 在时间序列和警报中附加额外标签

使用 Prometheus 的外部标签功能，可以将自定义标签附加到离开 Prometheus 的所有时间序列和警报。

先决条件

- **如果要配置 OpenShift Container Platform 核心监控组件**：
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 您已创建 **cluster-monitoring-config ConfigMap** 对象。

- **如果您要配置用于监控用户定义的项目的组件：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：
   - **对于监控 OpenShift Container Platform 核心项目的 Prometheus 实例，要将自定义标签附加到离开的所有时间序列和警报：**
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. 在 **data/config.yaml** 下定义每个指标要添加的标签映射:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        <key>: <value> 1
```

1 使用键值对替换 **<key>: <value>**，其中 **<key>** 是新标签的唯一名称，**<value>** 是它的值。

警告

不要使用 **prometheus** 或 **prometheus_replica** 作为键的名称，因为它们是保留的并会被覆盖。

例如，要将关于区域和环境的元数据添加到所有时间序列和警报中，请使用：

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        region: eu
        environment: prod
```

- **对于监控用户定义的项目的 Prometheus 实例，要将自定义标签附加到离开的所有时间序列和警报：**

  a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. 在 **data/config.yaml** 下定义每个指标要添加的标签映射:

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      prometheus:
        externalLabels:
          <key>: <value> 1
  ```

  1 使用键值对替换 **<key>: <value>**，其中 **<key>** 是新标签的唯一名称，**<value>** 是它的值。

警告

不要使用 **prometheus** 或 **prometheus_replica** 作为键的名称，因为它们是保留的并会被覆盖。

注意

在 **openshift-user-workload-monitoring** 项目中，Prometheus 负责处理指标，而 Thanos Ruler 负责处理警报和记录规则。在 **user-workload-monitoring-config ConfigMap** 中为 **prometheus** 设置 **externalLabels** 只会为指标配置外部标签，而不会为任何规则配置外部标签。

例如，要将关于区域和环境的元数据添加到与用户定义的项目相关所有时间序列和警报中，请使用：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      externalLabels:
        region: eu
        environment: prod
```

2. 保存文件以使改变生效。新的配置会被自动应用。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控
- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈

3.2. 为监控组件设置日志级别

您可以为 Alertmanager、Prometheus Operator、Prometheus、Thanos Querier 和 Thanos Ruler 配置日志级别。

以下日志级别可应用到 **cluster-monitoring-config** 和 **user-workload-monitoring-config ConfigMap** 中的相关组件：

- **debug**。记录调试、信息、警告和错误消息。
- **info**。记录信息、警告和错误消息。
- **warn**。仅记录警告和错误消息。
- **error**。仅记录错误消息。

默认日志级别为 **info**。

先决条件

先决条件

- **如果要为 openshift-monitoring 项目中的 Alertmanager、Prometheus Operator、Prometheus 或 Thanos Querier 设置日志级别：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群。
  - 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- **如果要为 openshift-user-workload-monitoring 项目中的 Prometheus Operator、Prometheus 或 Thanos Ruler 设置日志级别：**
  - 您可以使用具有 **cluster-admin** 角色的用户访问集群，也可以使用在 **openshift-user-workload-monitoring** 项目中具有 **user-workload-monitoring-config-edit** 角色的用户访问集群。
  - 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 编辑 **ConfigMap** 对象：
   - **要为 openshift-monitoring 项目中的组件设置日志级别：**
     a. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

     ```
     $ oc -n openshift-monitoring edit configmap cluster-monitoring-config
     ```

     b. 在 **data/config.yaml** 下为组件添加 **logLevel: <log_level>**：

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
       name: cluster-monitoring-config
       namespace: openshift-monitoring
     data:
       config.yaml: |
         <component>: 1
           logLevel: <log_level> 2
     ```

     1 您要为其设置日志级别的监控堆栈组件。对于默认平台监控，可用组件值包括 **prometheusK8s**、**alertmanagerMain**、**prometheusOperator** 和 **thanosQuerier**。

     2 为组件设置的日志级别。可用值为 **error**、**warn**、**info** 和 **debug**。默认值为 **info**。

   - **要为 openshift-user-workload-monitoring 项目中的组件设置日志级别：**
     a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

     ```
     $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
     ```

b. 在 **data/config.yaml** 下为组件添加 **logLevel: <log_level>**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: 1
      logLevel: <log_level> 2
```

1 您要为其设置日志级别的监控堆栈组件。对于用户工作负载监控，可用组件值包括 **prometheus**、**prometheusOperator** 和 **thanosRuler**。

2 为组件设置的日志级别。可用值为 **error**、**warn**、**info** 和 **debug**。默认值为 **info**。

2. 保存文件以使改变生效。应用日志级别更改时，组件的 Pod 会自动重启。

注意

除非集群管理员为用户定义的项目启用了监控，否则应用到 **user-workload-monitoring-config ConfigMap** 的配置不会被激活。

警告

一旦将更改保存到监控配置映射，可能会重新部署相关项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

3. 通过查看相关项目中的部署或 Pod 配置来确认已应用了日志级别。以下示例检查 **openshift-user-workload-monitoring** 项目中的 **prometheus-operator** 部署中的日志级别：

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |  grep "log-level"
```

输出示例

```
        - --log-level=debug
```

4. 检查组件的 Pod 是否正在运行。以下示例列出了 **openshift-user-workload-monitoring** 项目中 Pod 的状态：

```
$ oc -n openshift-user-workload-monitoring get pods
```

注意

如果 **ConfigMap** 中包含了一个未识别的 **loglevel** 值，则组件的 Pod 可能无法成功重启。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈
- 为用户定义的项目启用监控

3.3. 禁用默认的 GRAFANA 部署

默认情况下，使用显示集群指标的仪表板集合部署了只读 Grafana 实例。Grafana 实例不可用户配置。

您可以禁用 Grafana 部署，从而导致从集群中删除相关的资源。如果您不需要这些仪表板且希望在集群中节省资源，则可能会这样做。您仍然能够查看 web 控制台中包含的指标和仪表板。Grafana 可以随时安全地启用。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

2. 在 **data/config.yaml** 下为 **grafana** 组件添加 **enabled: false** ：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    grafana:
      enabled: false
```

3. 保存文件以使改变生效。应用更改时，资源将开始自动删除。

警告

此更改会导致一些组件，包括 Prometheus 和 Thanos Querier，正在重启。如果您还没有遵循"配置持久性存储"部分中的步骤，这可能会导致之前收集的指标丢失。

4. 检查 Grafana pod 是否不再运行。以下示例列出了 **openshift-monitoring** 项目中的 pod 状态：

```
$ oc -n openshift-monitoring get pods
```

注意

应用更改后可能需要几分钟时间来终止这些 pod。

其他资源

- 有关创建监控配置映射的步骤，请参阅准备配置监控堆栈

3.4. 禁用本地 ALERTMANAGER

在 OpenShift Container Platform 监控堆栈的 **openshift-monitoring** 项目中默认启用从 Prometheus 实例路由警报的本地 Alertmanager。

如果您不需要本地 Alertmanager，可以通过在 **openshift-monitoring** 项目中配置 **cluster-monitoring-config** 配置映射来禁用它。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已创建了 **cluster-monitoring-config** 配置映射。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config** 配置映射：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

2. 在 **data/config.yaml** 下为 **alertmanagerMain** 组件添加 **enabled: false** ：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
```

```
config.yaml: |
  alertmanagerMain:
    enabled: false
```

3. 保存文件以使改变生效。应用更改时，Alertmanager 实例会自动禁用。

其他资源

- Prometheus Alertmanager 文档
- 管理警报

3.5. 后续步骤

- 为用户定义的项目启用监控
- 了解远程健康报告，如果需要，可以选择停用它

第 4 章 为用户定义的项目启用监控

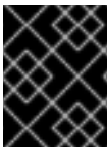
在 OpenShift Container Platform 4.9 中，除了默认的平台监控外，您还可以为用户定义的项目启用监控。现在，您可以监控 OpenShift Container Platform 中的自己的项目，而无需额外的监控解决方案。使用这个新功能可以集中监控核心平台组件和用户定义的项目。

注意

使用 Operator Lifecycle Manager (OLM) 安装的 Prometheus Operator 版本与用户定义的监控不兼容。因此，OpenShift Container Platform 不支持作为由 OLM Prometheus Operator 管理的 Prometheus 自定义资源 (CR) 安装的自定义 Prometheus 实例。

4.1. 为用户定义的项目启用监控

集群管理员可以通过在集群监控 **ConfigMap** 中设置 **enableUserWorkload: true** 字段来为用户定义的项目启用监控。



重要

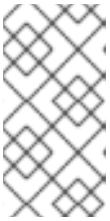
在 OpenShift Container Platform 4.9 中，要为用户定义的项目启用监控，您必须先删除任何自定义 Prometheus 实例。

注意

您必须可以使用具有 **cluster-admin** 角色的用户访问集群，才能在 OpenShift Container Platform 中为用户定义的项目启用监控。然后，集群管理员可以选择性地授予用户权限来配置负责监控用户定义的项目的组件。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您已创建 **cluster-monitoring-config ConfigMap** 对象。
- 您已选择性地创建并配置 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config ConfigMap**。您可以在该 **ConfigMap** 中为监控用户定义的项目的组件添加配置选项。



注意

每次您将配置更改保存到 **user-workload-monitoring-config ConfigMap** 时，都会重新部署 **openshift-user-workload-monitoring** 项目中的 Pod。有时重新部署这些组件需要花费一段时间。在首次为用户定义的项目启用监控前，您可以创建和配置 **ConfigMap**，以防止经常重新部署 Pod。

流程

1. 编辑 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

2. 将 **enableUserWorkload: true** 添加到 **data/config.yaml** 下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true 1
```

1 当设置为 **true** 时，**enableUserWorkload** 参数为集群中用户定义的项目启用监控。

3. 保存文件以使改变生效。然后会自动启用对用户定义的项目的监控。

警告

将更改保存到 **cluster-monitoring-config ConfigMap** 对象时，可能会重新部署 **openshift-monitoring** 项目中的 Pod 和其他资源。该项目中正在运行的监控进程也可能被重启。

4. 检查 **prometheus-operator**、**prometheus-user-workload** 和 **thanos-ruler-user-workload** Pod 是否在 **openshift-user-workload-monitoring** 项目中运行。Pod 启动可能需要片刻时间：

```
$ oc -n openshift-user-workload-monitoring get pod
```

输出示例

```
NAME                                  READY   STATUS    RESTARTS   AGE
prometheus-operator-6f7b748d5b-t7nbg  2/2     Running   0          3h
prometheus-user-workload-0            4/4     Running   1          3h
prometheus-user-workload-1            4/4     Running   1          3h
thanos-ruler-user-workload-0          3/3     Running   0          3h
thanos-ruler-user-workload-1          3/3     Running   0          3h
```

其他资源

- 创建集群监控配置映射
- 配置监控堆栈
- 授予用户权限来为用户定义的项目配置监控

4.2. 授予用户权限来监控用户定义的项目

集群管理员可以监控所有 OpenShift Container Platform 核心项目和用户定义的项目。

集群管理员可以授予开发人员和其他用户权限来监控他们自己的项目。通过分配以下监控角色中的一个即可授予权限：

- **monitoring-rules-view** 角色提供对项目的 **PrometheusRule** 自定义资源的读取访问权限。
- **monitoring-rules-edit** 角色授予用户权限来为项目创建、修改和删除 **PrometheusRule** 自定义资源。
- **monitoring-edit** 角色授予与 **monitoring-rules-edit** 角色相同的权限。另外，它还允许用户为服务或 Pod 创建新的提取目标。使用此角色，您还可以创建、修改和删除 **ServiceMonitor** 和 **PodMonitor** 资源。

您还可以授予用户权限来配置负责监控用户定义的项目的组件：

- **openshift-user-workload-monitoring** 项目中的 user-workload-monitoring-config-edit 角色允许您编辑 **user-workload-monitoring-config ConfigMap**。使用此角色，您可以编辑 **ConfigMap**，为用户定义的工作负载监控配置 Prometheus、Prometheus Operator 和 Thanos Ruler。

本节详细介绍了如何使用 OpenShift Container Platform Web 控制台或 CLI 分配这些角色。

4.2.1. 使用 Web 控制台授予用户权限

您可以使用 OpenShift Container Platform Web 控制台授予用户权限来监控其自己的项目。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 要将角色分配到的用户帐户已存在。

流程

1. 在 OpenShift Container Platform Web 控制台内的 **Administrator** 视角中，导航到 **User Management → Role Bindings → Create Binding**。
2. 在 **Binding Type** 部分中，选择“Namespace Role Binding”类型。
3. 在 **Name** 字段中输入角色绑定的名称。
4. 在 **Namespace** 字段中，选择要授予访问权限的用户定义的项目。

重要

监控角色将绑定到您在 **Namespace** 字段中应用的项目。您使用此流程向用户授予的权限将只应用于所选项目。

5. 在 **Role Name** 列表中选择 **monitoring-rules-view**、**monitoring-rules-edit** 或 **monitoring-edit**。
6. 在 **Subject** 部分，选择 **User**。
7. 在 **Subject Name** 字段中，输入用户名称。
8. 选择 **Create** 以应用角色绑定。

4.2.2. 使用 CLI 授予用户权限

您可以使用 OpenShift CLI (**oc**) 授予用户权限来监控其自己的项目。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 要将角色分配到的用户帐户已存在。
- 已安装 OpenShift CLI（**oc**）。

流程

- 为项目的用户分配一个监控角色：

```
$ oc policy add-role-to-user <role> <user> -n <namespace> 1
```

1 将 **<role>** 替换为 **monitoring-rules-view**、**monitoring-rules-edit** 或 **monitoring-edit**。

重要

无论您选择什么角色，必须以集群管理员的身份将其与特定项目进行绑定。

例如，将 **<role>** 替换为 **monitoring-edit**，**<user>** 替换为 **johnsmith**，**<namespace>** 替换为 **ns1**。这会为用户 **johnsmith** 分配在 **ns1** 命名空间内设置指标数据收集以及创建警报规则的权限。

4.3. 授予用户权限来为用户定义的项目配置监控

您可以授予用户权限来为用户定义的项目配置监控。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 要将角色分配到的用户帐户已存在。
- 已安装 OpenShift CLI（**oc**）。

流程

- 将 **user-workload-monitoring-config-edit** 角色分配给 **openshift-user-workload-monitoring** 项目中的用户：

```
$ oc -n openshift-user-workload-monitoring adm policy add-role-to-user \
  user-workload-monitoring-config-edit <user> \
  --role-namespace openshift-user-workload-monitoring
```

4.4. 从集群外部访问自定义应用程序的指标

了解如何在监控您自己的服务时从命令行查询 Prometheus 统计信息。您可以使用 **thanos-querier** 路由从集群外部访问监控数据。

先决条件

- 您已按照*用户定义的项目启用监控*部署了自己的服务。

流程

1. 提取令牌以连接到 Prometheus：

```
$ SECRET=`oc get secret -n openshift-user-workload-monitoring | grep  prometheus-user-workload-token | head -n 1 | awk '{print $1 }'`
```

```
$ TOKEN=`echo $(oc get secret $SECRET -n openshift-user-workload-monitoring -o json | jq -r '.data.token') | base64 -d`
```

2. 提取路由主机：

```
$ THANOS_QUERIER_HOST=`oc get route thanos-querier -n openshift-monitoring -o json | jq -r '.spec.host'`
```

3. 在命令行中查询您自己的服务的指标。例如：

```
$ NAMESPACE=ns1
```

```
$ curl -X GET -kG "https://$THANOS_QUERIER_HOST/api/v1/query?" --data-urlencode "query=up{namespace='$NAMESPACE'}" -H "Authorization: Bearer $TOKEN"
```

输出将显示应用容器集已启动的时间。

输出示例

```
{"status":"success","data":{"resultType":"vector","result":[{"metric":{"__name__":"up","endpoint":"web","instance":"10.129.0.46:8080","job":"prometheus-example-app","namespace":"ns1","pod":"prometheus-example-app-68d47c4fb6-jztp2","service":"prometheus-example-app"},"value":[1591881154.748,"1"]}]}}
```

4.5. 将用户定义的项目从监控中排除

用户工作负载监控中可以排除个别用户定义的项目。为此，只需将 **openshift.io/user-monitoring** 标签添加到项目的命名空间，值设为 **false**。

流程

1. 将标签添加到项目命名空间：

```
$ oc label namespace my-project 'openshift.io/user-monitoring=false'
```

2. 要重新启用监控，请从命名空间中删除该标签：

```
$ oc label namespace my-project 'openshift.io/user-monitoring-'
```

-

注意

如果项目有任何活跃的监控目标，Prometheus 可能需要几分钟时间在添加标签后停止提取它们。

4.6. 为用户定义的项目禁用监控

为用户定义的项目启用监控后，您可以通过在集群监控 **ConfigMap** 对象中设置 **enableUserWorkload: false** 来再次禁用它。

注意

另外，您也可以通过删除 **enableUserWorkload: true** 来为用户定义的项目禁用监控。

流程

1. 编辑 **cluster-monitoring-config ConfigMap** 对象：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

   a. 在 **data/config.yaml** 下将 **enableUserWorkload:** 设置为 **false**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: false
```

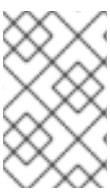
2. 保存文件以使改变生效。然后会自动禁用对用户定义的项目的监控。

3. 检查 **prometheus-operator**、**prometheus-user-workload** 和 **thanos-ruler-user-workload** Pod 是否已在 **openshift-user-workload-monitoring** 项目中终止。这可能需要片刻时间：

```
$ oc -n openshift-user-workload-monitoring get pod
```

输出示例

```
No resources found in openshift-user-workload-monitoring project.
```



注意

在禁用了对用户定义的项目的监控时，**openshift-user-workload-monitoring** 项目中的 **user-workload-config ConfigMap** 对象不会被自动删除。这是为了保留您可能已在 **ConfigMap** 中创建的任何自定义配置。

4.7. 后续步骤

- 管理指标

第 5 章 管理指标

您可以收集指标，以监控集群组件和您自己的工作负载的表现情况。

5.1. 了解指标

在 OpenShift Container Platform 4.9 中，集群组件的监控方式是提取通过服务端点公开的指标。您还可以为用户定义的项目配置指标集合。

您可以通过在应用程序级别使用 Prometheus 客户端库来定义您要为您自己的工作负载提供的指标。

在 OpenShift Container Platform 中，指标通过 **/metrics** 规范名称下的 HTTP 服务端点公开。您可以通过针对 **http://<endpoint>/metrics** 运行 **curl** 查询来列出服务的所有可用指标。例如，您可以向 **prometheus-example-app** 示例服务公开路由，然后运行以下命令来查看其所有可用指标：

```
$ curl http://<example_app_endpoint>/metrics
```

输出示例

```
# HELP http_requests_total Count of all HTTP requests
# TYPE http_requests_total counter
http_requests_total{code="200",method="get"} 4
http_requests_total{code="404",method="get"} 2
# HELP version Version information about this binary
# TYPE version gauge
version{version="v0.1.0"} 1
```

其他资源

- 有关 Prometheus 客户端库的详情，请参阅 Prometheus 文档。

5.2. 为用户定义的项目设置指标集合

您可以创建一个 **ServiceMonitor** 资源，从用户定义的项目中的服务端点提取指标。这假设您的应用程序使用 Prometheus 客户端库向 **/metrics** 规范名称公开指标。

本节介绍了如何在用户定义的项目中部署示例服务，然后创建一个 **ServiceMonitor** 资源来定义应该如何监控该服务。

5.2.1. 部署示例服务

要为用户定义的项目中服务测试监控，您可以部署示例服务。

流程

1. 为服务配置创建 YAML 文件。在本例中，该文件名为 **prometheus-example-app.yaml**。
2. 在该文件中添加以下部署和服务配置详情：

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-example-app
  name: prometheus-example-app
  namespace: ns1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-example-app
  template:
    metadata:
      labels:
        app: prometheus-example-app
    spec:
      containers:
      - image: ghcr.io/rhobs/prometheus-example-app:0.4.1
        imagePullPolicy: IfNotPresent
        name: prometheus-example-app
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: prometheus-example-app
  name: prometheus-example-app
  namespace: ns1
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-example-app
  type: ClusterIP
```

此配置会在用户定义的 **ns1** 项目中部署名为 **prometheus-example-app** 的服务。此服务会公开自定义 **version** 指标。

3. 将配置应用到集群：

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

部署该服务需要一些时间。

4. 您可以检查该 Pod 是否正在运行：

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

输出示例

```
NAME                                  READY   STATUS    RESTARTS   AGE
prometheus-example-app-7857545cb7-sbgwq   1/1     Running   0          81m
```

5.2.2. 指定如何监控服务

要使用服务公开的指标，需要将 OpenShift Container Platform 监控配置为从 **/metrics** 端点中提取指标。您可以使用一个 **ServiceMonitor** 自定义资源定义（CRD）应该如何监控服务，或使用一个 **PodMonitor** CRD 指定应该如何监控 pod。前者需要 **Service** 对象，而后者则不需要，允许 Prometheus 直接从 Pod 公开的指标端点中提取指标。

此流程演示了如何为用户定义的项目中的服务创建 **ServiceMonitor** 资源。

先决条件

- 您可以使用具有 **cluster-admin** 角色或 **monitoring-edit** 角色的用户访问集群。
- 您已为用户定义的项目启用了监控。
- 在本例中，您已在 **ns1** 项目中部署了 **prometheus-example-app** 示例服务。

注意

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

流程

1. 为 **ServiceMonitor** 资源配置创建一个 YAML 文件。在本例中，该文件名为 **example-app-service-monitor.yaml**。
2. 添加以下 **ServiceMonitor** 资源配置详情：

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

这会定义一个 **ServiceMonitor** 资源，用于提取由 **prometheus-example-app** 示例服务公开的指标，其中包含 **version** 指标。

注意

用户定义的命名空间中的 **ServiceMonitor** 资源只能发现同一命名空间中的服务。也就是说，**ServiceMonitor** 资源的 **namespaceSelector** 字段总是被忽略。

3. 将配置应用到集群：

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

部署 **ServiceMonitor** 资源需要一些时间。

4. 您可以检查 **ServiceMonitor** 资源是否正在运行：

```
$ oc -n ns1 get servicemonitor
```

输出示例

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

其他资源

- 为用户定义的项目启用监控
- 如何使用用户定义的项目中的 TLS 提取指标
- PodMonitor API
- ServiceMonitor API

5.3. 查询指标

OpenShift Container Platform 监控仪表板可供您运行 Prometheus Query Language (PromQL) 查询来查看图表中呈现的指标。此功能提供有关集群以及要监控的任何用户定义工作负载的状态信息。

作为**集群管理员**，您可以查询所有 OpenShift Container Platform 核心项目和用户定义的项目的指标。

作为**开发者**，您必须在查询指标时指定项目名称。您必须具有所需权限才能查看所选项目的指标。

5.3.1. 以集群管理员身份查询所有项目的指标

作为集群管理员，或具有所有项目的查看权限的用户，您可以在 Metrics UI 中访问所有 OpenShift Container Platform 默认项目和用户定义的项目的指标。

注意

只有集群管理员可以访问 OpenShift Container Platform Monitoring 提供的第三方 UI。

先决条件

- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 在 OpenShift Container Platform Web 控制台内的 **Administrator** 视角中，选择 **Observe → Metrics**。

2. 选择 **Insert Metric at Cursor** 来查看预定义的查询列表。
3. 要创建自定义查询，请将 Prometheus Query Language (PromQL) 查询添加到 **Expression** 字段。
4. 要添加多个查询，选择 **Add Query**。
5. 要删除查询，选择查询旁边的 ，然后选择 **Delete query**。
6. 要禁止运行查询，请选择查询旁边的 并选择 **Disable query**。
7. 选择 **Run Queries** 来运行您已创建的查询。图表中会直观呈现查询的指标。如果查询无效，则 UI 会显示错误消息。

注意

如果查询对大量数据进行运算，这可能会在绘制时序图时造成浏览器超时或过载。要避免这种情况，请选择 **Hide graph** 并且仅使用指标表来校准查询。然后，在找到可行的查询后，启用图表来绘制图形。

8. 可选：页面 URL 现在包含您运行的查询。要在以后再次使用这一组查询，请保存这个 URL。

其他资源

- 有关创建 PromQL 查询的更多详情，请参阅 Prometheus query 文档。

5.3.2. 以开发者身份查询用户定义的项目的指标

您可以以开发者或具有项目查看权限的用户身份访问用户定义项目的指标。

在 **Developer** 视角中， Metrics UI 包括所选项目的一些预定义 CPU、内存、带宽和网络数据包查询。您还可以对项目的 CPU、内存、带宽、网络数据包和应用程序指标运行自定义 Prometheus Query Language (PromQL) 查询。

注意

开发者只能使用 **Developer** 视角，而不能使用 **Administrator** 视角。作为开发者，您一次只能查询一个项目的指标。开发人员无法访问 OpenShift Container Platform 监控提供的用于核心平台组件的第三方 UI。取而代之，为您的用户定义的项目使用 Metrics UI。

先决条件

- 对于您要查看指标的项目，您可以作为开发者或具有查看权限的用户访问集群。
- 您已为用户定义的项目启用了监控。
- 您已在用户定义的项目中部署了服务。
- 您已为该服务创建了 **ServiceMonitor** 自定义资源定义（CRD），以定义如何监控该服务。

流程

1. 在 OpenShift Container Platform web 控制台中的 **Developer** 视角中，选择 **Observe → Metrics**。
2. 在 **Project:** 列表中选择您要查看指标的项目。
3. 从 **Select Query** 列表中选择查询，或者通过选择 **Show PromQL** 运行自定义 PromQL 查询。

注意

在 **Developer** 视角中，您一次只能运行一个查询。

其他资源

- 有关创建 PromQL 查询的更多详情，请参阅 Prometheus query 文档。

其他资源

- 有关以开发者或特权用户身份访问非集群指标的详情，请参阅以开发者身份查询用户定义的项目的指标

5.3.3. 探索视觉化指标

运行查询后，指标会显示在交互式图表中。图表中的 X 轴代表时间，Y 轴代表指标值。图形上的每个指标都以带颜色的线条显示。您可以交互式地操作图表并探索指标。

流程

在 **Administrator** 视角中：

1. 最初，图表中显示所有启用的查询中的所有指标。您可以要选择显示哪些指标。

注意

默认情况下，查询表会显示一个展开的视图，列出每个指标及其当前值。您可以选择 ˅ 来最小化查询的展开视图。

- 要隐藏查询的所有指标，请点击  查询并点击 **Hide all series**。
- 要隐藏特定的指标，请转至查询表，然后点击指标名称旁边带颜色的方块。

2. 要放大图表并更改时间范围，请执行以下操作之一：
   - 点击图表并在水平方向上拖动，以可视化方式选择时间范围。
   - 使用左上角的菜单来选择时间范围。
3. 要重置时间范围，请选择 **Reset Zoom**。
4. 要显示所有查询在特定时间点的输出，请将鼠标光标停留在图表中的对应点上。弹出框中会显示查询输出。
5. 要隐藏图表，请选择 **Hide Graph**。

在 **Developer** 视角中：

1. 要放大图表并更改时间范围，请执行以下操作之一：
   - 点击图表并在水平方向上拖动，以可视化方式选择时间范围。
   - 使用左上角的菜单来选择时间范围。
2. 要重置时间范围，请选择 **Reset Zoom**。
3. 要显示所有查询在特定时间点的输出，请将鼠标光标停留在图表中的对应点上。弹出框中会显示查询输出。

其他资源

- 请参阅有关使用 PromQL 接口的查询指标部分

5.4. 后续步骤

- 管理警报

第 6 章 管理警报

在 OpenShift Container Platform 4.9 中，您可以通过 Alerting UI 管理警报、静默和警报规则。

- **警报规则**。警报规则包含一组概述集群中特定状态的条件。当这些条件满足时会触发警报。可为警报规则分配一个严重性来定义警报的路由方式。
- **警报**。当警报规则中定义的条件为满足时会触发警报。警报提供一条通知，说明一组情况在 OpenShift Container Platform 集群中显而易见。
- **静默**。可对警报应用静默，以防止在警报条件满足时发送通知。在您着手处理根本问题的同时，您可在初始通知后将警报静音。

注意

Alerting UI 中可用的警报、静默和警报规则与您可访问的项目相关。例如，如果使用 **cluster-admin** 权限登录，您可以访问所有警报、静默和警报规则。

如果您是非管理员用户，如果您被分配了以下用户角色，您可以创建和静默警报：

- **cluster-monitoring-view** 角色，允许您访问 Alertmanager
- **monitoring-alertmanager-edit** 角色，允许您在 web 控制台的 **Administrator** 视角中创建和静默警报
- **monitoring-rules-edit** 角色，允许您在 web 控制台的 **Developer** 视角中创建和静默警报

6.1. 在 ADMINISTRATOR 和 DEVELOPER 视角中访问 ALERTING UI

可通过 OpenShift Container Platform Web 控制台中的 Administrator 视角和 Developer 视角访问 Alerting UI。

- 在 **Administrator** 视角中，选择 **Observe → Alerting**。在此视角中，Alerting UI 有三个主要页面，即 **Alerts**、**Silences** 和 **Alerting Rules** 页面。
- 在 **Developer** 视角中，选择 **Observe → <project_name> → Alerts**。在这个视角中，警报、静默和警报规则都通过 **Alerts** 页面管理。**Alerts** 页面中显示的结果特定于所选项目。

注意

在 Developer 视角中，您可以从有权在 **Project:** 列表中访问的 OpenShift Container Platform 核心项目和用户自定义的项目中选择，。但是，如果您没有 **cluster-admin** 特权，则不会显示与 OpenShift Container Platform 核心项目相关的警报、静默和警报规则。

6.2. 搜索和过滤警报、静默和警报规则

您可以过滤 Alerting UI 中显示的警报、静默和警报规则。本节介绍每个可用的过滤选项。

了解警报过滤器

在 **Administrator** 视角中，Alerting UI 中的 **Alerts** 页面针对与 OpenShift Container Platform 默认项目和用户定义的项目相关的警报提供了详细信息。该页面包括每个警报的严重性、状态和来源摘要。另外还会显示警报进入其当前状态的时间。

您可以按警报状态、严重性和来源进行过滤。默认情况下，只会显示处于 Firing 状态的 Platform 警报。下面描述了每个警报过滤选项：

- Alert State 过滤器：
  - Firing。警报正在触发，因为满足警报条件，且可选的 for 持续时间已过。只要条件一直满足，警报将继续触发。
  - Pending。该警报处于活跃状态，但正在等待警报规则中指定的持续时间，然后再触发警报。
  - Silenced。现在，警报在定义的时间段内处于静默状态。静默会根据您定义的一组标签选择器临时将警报静音。对于符合所有列出的值或正则表达式的警报，不会发送通知。
- Severity 过滤器：
  - Critical。触发了警报的条件可能会产生重大影响。该警报在触发时需要立即关注，并且通常会传给个人或关键响应团队。
  - Warning。该警报针对可能需要注意的事件提供警告通知，以防止问题的发生。警告通常会路由到一个问题单系统进行非即时的审阅。
  - Info。该警报仅用于提供信息。
  - None。该警报没有定义的严重性。
  - 您还可以针对与用户定义的项目相关的警报创建自定义严重性定义。
- Source 过滤器：
  - Platform。平台级别的警报仅与 OpenShift Container Platform 默认项目相关。这些项目提供 OpenShift Container Platform 核心功能。
  - User。用户警报与用户定义的项目相关。这些警报是用户创建的，并可自定义。用户定义的工作负载监控可在安装后启用，以便您观察自己的工作负载。

了解静默过滤器

在 Administrator 视角中，Alerting UI 中的 Silences 页面针对应用到 OpenShift Container Platform 默认项目和用户定义项目中的警报的静默提供了详细信息。该页面包括每个静默的状态以及静默结束时间的摘要。

您可以按静默状态进行过滤。默认情况下，仅显示 Active 和 Pending 静默。下面描述了每个静默状态过滤器选项：

- Silence State 过滤器：
  - Active。静默处于活跃状态，在静默到期前，警报将静音。
  - Pending。静默已被调度，但还没有激活。
  - Expired。静默已过期，如果满足警报条件，将发送通知。

了解警报规则过滤器

在 Administrator 视角中，Alerting UI 中的 Alerts Rules 页面针对与 OpenShift Container Platform 默认项目和用户定义的项目相关的警报规则提供了详细信息。该页面包括每个警报规则的状态、严重性和来源摘要。

您可以按警报状态、严重性和来源过滤警报规则。默认情况下，只会显示 Platform 警报规则。下面描述了每个警报规则过滤选项：

- Alert State 过滤器：
  - Firing。警报正在触发，因为满足警报条件，且可选的 for 持续时间已过。只要条件一直满足，警报将继续触发。
  - Pending。该警报处于活跃状态，但正在等待警报规则中指定的持续时间，然后再触发警报。
  - Silenced。现在，警报在定义的时间段内处于静默状态。静默会根据您定义的一组标签选择器临时将警报静音。对于符合所有列出的值或正则表达式的警报，不会发送通知。
  - Not Firing。警报未触发。
- Severity 过滤器：
  - Critical。警报规则中定义的条件可能会产生重大影响。如果满足这些条件，需要立即关注。与该规则相关的警报通常会传给个人或关键响应团队。
  - Warning。警报规则中定义的条件可能需要注意，以防止问题的发生。与该规则相关的警报通常会路由到一个问题单系统进行非即时的审阅。
  - Info。警报规则仅提供信息警报。
  - None。该警报规则没有定义的严重性。
  - 您还可以针对与用户定义的项目相关的警报规则创建自定义严重性定义。
- Source 过滤器：
  - Platform。平台级别的警报规则仅与 OpenShift Container Platform 默认项目相关。这些项目提供 OpenShift Container Platform 核心功能。
  - User。用户定义的工作负载警报规则与用户定义的项目相关。这些警报规则是用户创建的，并可自定义。用户定义的工作负载监控可在安装后启用，以便您观察自己的工作负载。

在 Developer 视角中搜索和过滤警报、静默和警报规则

在 Developer 视角中，Alerting UI 中的 Alerts 页面提供了与所选项目相关的警报和静默的组合视图。对于每个显示的警报，都提供了相关警报规则的链接。

在该视图中，您可以按警报状态和严重性进行过滤。默认情况下，如果您有访问所选项目的权限，则会显示项目中的所有警报。这些过滤器与针对 Administrator 视角描述的过滤器相同。

6.3. 获取关于警报、静默和警报规则的信息

Alerting UI 提供有关警报及其相关警报规则和静默的详细信息。

先决条件

- 对于您要查看指标的项目，您可以作为开发者或具有查看权限的用户访问集群。

流程

要在 Administrator 视角中获取有关警报的信息：

1. 打开 OpenShift Container Platform Web 控制台，并导航至 **Observe → Alerting → Alerts** 页面。
2. 可选：使用搜索列表中的 **Name** 字段按名称搜索警报。
3. 可选：通过选择 **Filter** 列表中的过滤器来按状态、严重性和来源过滤警报。
4. 可选：点击 **Name**、**Severity**、**State** 和 **Source** 列标题中的一个或多个标题对警报进行排序。
5. 选择警报的名称以导航到其 **Alert Details** 页面。该页面包含一个说明警报时间序列数据的图形。它还提供与此警报相关的信息，包括：
   - 警报的描述
   - 与警报关联的消息
   - 附加到警报的标签
   - 其相关警报规则的链接
   - 警报的静默（如果存在）

要在 Administrator 视角中获取有关静默的信息：

1. 进入到 **Observe → Alerting → Silences** 页面。
2. 可选：使用 **Search by name** 字段按名称过滤静默。
3. 可选：通过选择 **Filter** 列表中的过滤器来按状态过滤静默。默认情况下会应用 **Active** 和 **Pending** 过滤器。
4. 可选：点击 **Name**、**Firing alerts** 和 **State** 列标题中的一个或多个标题对静默进行排序。
5. 选择静默的名称以导航到其 **Silence Details** 页面。该页面包括以下详情：
   - 警报指定条件
   - 开始时间
   - 结束时间
   - 静默状态
   - 触发警报的数目和列表

要在 Administrator 视角中获取有关警报规则的信息：

1. 进入到 **Observe → Alerting → Alerting Rules** 页面。
2. 可选：通过选择 **Filter** 列表中的过滤器来按状态、严重性和来源过滤警报规则。
3. 可选：点击 **Name**、**Severity**、**Alert State** 和 **Source** 列标题中的一个或多个标题对警报规则进行排序。
4. 选择警报规则的名称以导航到其 **Alerting Rule Details** 页面。该页面提供有关警报规则的以下详情：
   - 警报规则名称、严重性和描述

- 定义触发此警报的条件的表达式
- 触发警报的条件得到满足的时间
- 受警报规则约束的各个警报的图形，其中显示了触发该警报的值
- 受警报规则约束的所有警报的列表

要在 Developer 视角中获取有关警报、静默和警报规则的信息：

1. 进入到 **Observe → <project_name> → Alerts** 页面。
2. 查看警报、静默或警报规则的详情：
   - 要查看 **Alert Details**，可选择警报名称左侧的 **>**，然后在列表中选择警报。
   - 要查看 **Silence Details**，可在 **Alert Details** 页面的 **Silenced By** 部分中选择静默。**Silence Details** 页面包括以下信息：
     - 警报指定条件
     - 开始时间
     - 结束时间
     - 静默状态
     - 触发警报的数目和列表
   - 要查看 **Alerting Rule Details**，可在 **Alerts** 页面中警告右侧的  菜单中选择 **View Alerting Rule**。

注意

Developer 视角中仅显示与所选项目相关的警报、静默和警报规则。

6.4. 管理警报规则

OpenShift Container Platform 监控附带一组默认的警报规则。作为集群管理员，您可以查看默认警报规则。

在 OpenShift Container Platform 4.9 中，您可以在用户定义的项目中创建、查看、编辑和删除警报规则。

警报规则注意事项

- 默认的警报规则专门用于 OpenShift Container Platform 集群。
- 有些警报规则特意使用相同的名称。它们发送关于同一事件但具有不同阈值和/或不同严重性的警报。
- 如果较低严重性警报在较高严重性警报触发的同时触发，禁止规则可防止在这种情况下发送通知。

6.4.1. 为用户定义的项目优化警报

要优化您自己的项目的警报，您可以在创建警报规则时考虑以下建议：

- **尽可能减少您为项目创建的警报规则数量**。创建警报规则来针对会影响您的条件通知您。如果您为不会影响您的条件生成多个警报，则更难以注意到相关警报。
- **为症状而不是原因创建警报规则**。创建警报规则来针对条件通知您，而无论根本原因是什么。然后可以调查原因。如果每个警报规则都只与特定原因相关，则需要更多警报规则。然后，可能会错过一些原因。
- **在编写警报规则前进行规划**。确定对您很重要的症状以及一旦发生您想要采取什么操作。然后为每个症状构建警报规则。
- **提供明确的警报信息**。在警报消息中说明症状和推荐操作。
- **在警报规则中包含严重性级别**。警报的严重性取决于当报告的症状发生时您需要如何做出反应。例如，如果症状需要个人或关键响应团队立即关注，就应该触发关键警报。
- **优化警报路由**。如果警报规则不查询默认的 OpenShift Container Platform 指标，则直接在 **openshift-user-workload-monitoring** 项目的 Prometheus 实例上部署该规则。这可减少警报规则的延迟，并尽可能降低监控组件的负载。

警告

用户定义的项目的默认 OpenShift Container Platform 指标提供有关 CPU 和内存用量、带宽统计和数据包速率的信息。如果您将规则直接路由到 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例，则无法将这些指标包含在警报规则中。只有在您阅读了文档并对监控架构有了全面的了解后，才应使用警报规则优化。

其他资源

- 如需更多有关优化警报的指南，请参阅 Prometheus 警报文档
- 如需了解有关 OpenShift Container Platform 4.9 监控架构的详细信息，请参阅监控概述。

6.4.2. 为用户定义的项目创建警报规则

您可以为用户定义的项目创建警报规则。这些警报规则将根据所选指标的值触发警报。

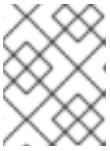
先决条件

- 您已为用户定义的项目启用了监控。
- 对于您要创建警报规则的项目，您已作为具有 **monitoring-rules-edit** 角色的用户登录。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 为警报规则创建 YAML 文件。在本例中，该文件名为 **example-app-alerting-rule.yaml**。

2. 向 YAML 文件添加警报规则配置。例如：



注意

当创建警报规则时，如果在其他项目中存在具有相同名称的规则，则对其强制使用项目标签。

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert
      expr: version{job="prometheus-example-app"} == 0
```

此配置会创建一个名为 **example-alert** 的警报规则。当示例服务公开的 **version** 指标变为 **0** 时，警报规则会触发警报。

重要

用户定义的警报规则可以包含其自身项目的指标和集群指标。您不能包含其他用户定义的项目的指标。

例如，用户定义的项目 **ns1** 的警报规则可以包含来自 **ns1** 的指标和集群指标，如 CPU 和内存指标。但是，该规则无法包含来自 **ns2** 的指标。

另外，您无法为 **openshift-*** 核心 OpenShift Container Platform 项目创建警报规则。OpenShift Container Platform 监控默认为这些项目提供一组警报规则。

3. 将配置文件应用到集群：

```
$ oc apply -f example-app-alerting-rule.yaml
```

创建警报规则需要一些时间。

6.4.3. 减少不查询平台指标的警报规则的延迟

如果用户定义的项目的警报规则不查询默认集群指标，您可以在 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例上直接部署该规则。这可绕过不需要的 Thanos Ruler，从而减少警报规则的延迟。这也有助于尽可能降低监控组件的总负载。

警告

用户定义的项目的默认 OpenShift Container Platform 指标提供有关 CPU 和内存用量、带宽统计和数据包速率的信息。如果您将规则直接部署到 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例，则无法将这些指标包含在警报规则中。只有在您阅读了文档并对监控架构有了全面的了解后，才应使用本节中所述的流程。

先决条件

- 您已为用户定义的项目启用了监控。
- 对于您要创建警报规则的项目，您已作为具有 **monitoring-rules-edit** 角色的用户登录。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 为警报规则创建 YAML 文件。在本例中，该文件名为 **example-app-alerting-rule.yaml**。
2. 向 YAML 文件添加警报规则配置，该文件中包含键为 **openshift.io/prometheus-rule-evaluation-scope** 且值为 **leaf- prometheus** 的标签。例如：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
  labels:
    openshift.io/prometheus-rule-evaluation-scope: leaf-prometheus
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert
      expr: version{job="prometheus-example-app"} == 0
```

如果存在该标签，则会在 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例上部署警报规则。如果不存在该标签，则会将警报规则部署到 Thanos Ruler。

1. 将配置文件应用到集群：

```
$ oc apply -f example-app-alerting-rule.yaml
```

创建警报规则需要一些时间。

- 如需了解有关 OpenShift Container Platform 4.9 监控架构的详细信息，请参阅监控概述。

6.4.4. 访问用户定义的项目的警报规则

要列出用户定义的项目的警报规则，您必须已被分配该项目的 **monitoring-rules-view** 角色。

先决条件

- 您已为用户定义的项目启用了监控。
- 您已作为已被分配了项目的 **monitoring-rules-view** 角色的用户登录。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 您可以列出 **<project>** 中的警报规则：

```
$ oc -n <project> get prometheusrule
```

2. 要列出警报规则的配置，请运行以下命令：

```
$ oc -n <project> get prometheusrule <rule> -o yaml
```

6.4.5. 在单个视图中列出所有项目的警报规则

作为集群管理员，您可以在单个视图中一起列出 OpenShift Container Platform 核心项目和用户定义的项目的警报规则。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 在 **Administrator** 视角中，进入到 **Observe → Alerting → Alerting Rules**。
2. 在 **Filter** 下拉菜单中选择 **Platform** 和 **User** 来源。

注意

默认会选择 **Platform** 来源。

6.4.6. 为用户定义的项目删除警报规则

您可以为用户定义的项目删除警报规则。

先决条件

- 您已为用户定义的项目启用了监控。
- 对于您要创建警报规则的项目，您已作为具有 **monitoring-rules-edit** 角色的用户登录。
- 已安装 OpenShift CLI（**oc**）。

流程

- 要删除 **<namespace>** 中的规则 **<foo>**，请运行以下命令：

```
$ oc -n <namespace> delete prometheusrule <foo>
```

其他资源

- 请参阅 Alertmanager 文档

6.5. 管理静默

您可以创建一个静默，在警报触发时停止接收有关警报的通知。在您解决根本问题的同时，在收到第一次通知后将警报置于静默状态可能很有用。

在创建静默时，您必须指定它是立即激活，还是稍后激活。您还必须设置静默在多长一段时间后到期。

您可以查看、编辑现有的静默并使其到期。

6.5.1. 静默警报

您可以静默特定的警报，或者静默符合您定义的指定条件的警报。

先决条件

- 您是一个集群管理员，可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您是一个非管理员用户，可以使用具有以下用户角色的用户访问集群：
  - **cluster-monitoring-view** 集群角色，允许您访问 Alertmanager。
  - **monitoring-alertmanager-edit** 角色，允许您在 web 控制台的 **Administrator** 视角中创建和静默警报
  - **monitoring-rules-edit** 角色，允许您在 web 控制台的 **Developer** 视角中创建和静默警报。

流程

静默特定的警报：

- 在 **Administrator** 视角中：
  1. 进入到 OpenShift Container Platform Web 控制台的 **Observe → Alerting → Alerts** 页面。
  2. 对于您要置于静默状态的警报，请选择右列中的  并选择 **Silence Alert**。这时会显示 **Silence Alert** 表单，其中预先填充了所选警报的规格。
  3. 可选：修改静默。
  4. 在创建静默前您必须添加注释。
  5. 若要创建静默，请选择 **Silence**。
- 在 **Developer** 视角中：
  1. 进入到 OpenShift Container Platform web 控制台中的 **Observe → <project_name> → Alerts** 页面。

2. 选择警报名称左侧的 > 来展开警报的详情。选择展开视图中的警报名称以打开警报的 **Alert Details** 页面。
3. 选择 **Silence Alert**。这时会显示 **Silence Alert** 表单，其中预先填充了所选警报的规格。
4. 可选：修改静默。
5. 在创建静默前您必须添加注释。
6. 若要创建静默，请选择 **Silence**。

要在 **Administrator** 视角中通过创建警报规格来将一组警报置于静默状态：

1. 进入到 OpenShift Container Platform Web 控制台中的 **Observe → Alerting → Silences** 页面。
2. 选择 **Create Silence**。
3. 在 **Create Silence** 表单中设置警报的时间表、持续时间和标签详情。您还必须为静默添加注释。
4. 要为与您在上一步中输入的标签选择器匹配的警报创建静默，请选择 **Silence**。

6.5.2. 编辑静默

您可以编辑静默，这样会导致现有静默到期，并以更改后的配置创建新静默。

流程

要在 **Administrator** 视角中编辑静默：

1. 进入到 **Observe → Alerting → Silences** 页面。
2. 针对您想要修改的静默，选择最后一列中的 ⋮，然后选择 **Edit silence**。
   另外，您还可以在静默的 **Silence Details** 页面中选择 **Actions → Edit Silence**。
3. 在 **Edit Silence** 页面中，输入您的更改并选择 **Silence**。这会使现有的静默到期，并以所选配置创建新静默。

要在 **Developer** 视角中编辑静默：

1. 进入到 **Observe → <project_name> → Alerts** 页面。
2. 选择警报名称左侧的 > 来展开警报的详情。选择展开视图中的警报名称以打开警报的 **Alert Details** 页面。
3. 在该页面的 **Silenced By** 部分中选择静默名称，以导航到该静默的 **Silence Details** 页面。
4. 选择静默的名称以导航到其 **Silence Details** 页面。
5. 在静默的 **Silence Details** 页面中选择 **Actions → Edit Silence**。
6. 在 **Edit Silence** 页面中，输入您的更改并选择 **Silence**。这会使现有的静默到期，并以所选配置创建新静默。

6.5.3. 使静默到期

您可以让静默到期。让静默到期会永久停用这一静默。

流程

在 **Administrator** 视角中使静默到期：

1. 进入到 **Observe → Alerting → Silences** 页面。
2. 针对您想要修改的静默，选择最后一列中的  ，然后选择 **Expire silence**。
   另外，您还可以在静默的 **Silence Details** 页面中选择 **Actions → Expire Silence**。

要在 **Developer** 视角中使静默到期：

1. 进入到 **Observe → <project_name> → Alerts** 页面。
2. 选择警报名称左侧的 **>** 来展开警报的详情。选择展开视图中的警报名称以打开警报的 **Alert Details** 页面。
3. 在该页面的 **Silenced By** 部分中选择静默名称，以导航到该静默的 **Silence Details** 页面。
4. 选择静默的名称以导航到其 **Silence Details** 页面。
5. 在静默的 **Silence Details** 页面中选择 **Actions → Expire Silence**。

6.6. 将通知发送到外部系统

在 OpenShift Container Platform 4.9 中，可在 Alerting UI 中查看触发警报。默认不会将警报配置为发送到任何通知系统。您可以将 OpenShift Container Platform 配置为将警报发送到以下类型的接收器：

- PagerDuty
- Webhook
- 电子邮件
- Slack

通过将警报路由到接收器，您可在出现故障时及时向适当的团队发送通知。例如，关键警报需要立即关注，通常会传给个人或关键响应团队。相反，提供非关键警告通知的警报可能会被路由到一个问题单系统进行非即时的审阅。

使用 watchdog 警报检查警报是否工作正常

OpenShift Container Platform 监控功能包含持续触发的 watchdog 警报。Alertmanager 重复向已配置的通知提供程序发送 watchdog 警报通知。此提供程序通常会配置为在其停止收到 watchdog 警报时通知管理员。这种机制可帮助您快速识别 Alertmanager 和通知提供程序之间的任何通信问题。

6.6.1. 配置警报接收器

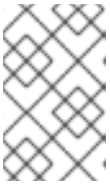
您可以配置警报接收器，以确保了解集群出现的重要问题。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 在 **Administrator** 视角中，进入到 **Administration → Cluster Settings → Configuration → Alertmanager**。



注意

或者，您还可以通过 notification drawer 访问同一页面。选择 OpenShift Container Platform Web 控制台右上角的铃铛图标，并在 **AlertmanagerReceiverNotConfigured** 警报中选择 **Configure**。

2. 在该页面的 **Receivers** 部分中选择 **Create Receiver**。
3. 在 **Create Receiver** 表单中，添加 **Receiver Name**，然后从列表中选择 **Receiver Type**。
4. 编辑接收器配置：
   - 对于 PagerDuty 接收器：
     a. 选择集成类型并添加 PagerDuty 集成密钥。
     b. 添加 PagerDuty 安装的 URL。
     c. 如果要编辑客户端和事件详情或严重性规格，请选择 **Show advanced configuration**。
   - 对于 Webhook 接收器：
     a. 添加将 HTTP POST 请求发送到的端点。
     b. 如果要编辑将已解析的警报发送给接收器的默认选项，请选择 **Show advanced configuration**。
   - 对于电子邮件接收器：
     a. 添加要将通知发送到的电子邮件地址。
     b. 添加 SMTP 配置详情，包括发送通知的地址、用来发送电子邮件的智能主机和端口号、SMTP 服务器的主机名以及验证详情。
     c. 选择是否需要 TLS。
     d. 如果要将默认选项编辑为不向接收器发送已解析的警报，或编辑电子邮件通知正文配置，请选择 **Show advanced configuration**。
   - 对于 Slack 接收器：
     a. 添加 Slack Webhook 的 URL。
     b. 添加要将通知发送到的 Slack 频道或用户名。
     c. 如果要将默认选项编辑为不向接收器发送已解析的警报，或编辑图标和用户名配置，请选择 **Show advanced configuration**。您还可以选择是否查找并链接频道名称和用户名。
5. 默认情况下，如果触发的警报带有与所有选择器匹配的标签，将被发送到接收器。如果您希望触发的警报在发送到接收器之前具有完全匹配的标签值：
   a. 在表单的 **Routing Labels** 部分中添加路由标签名称和值。
   b. 如果想要使用正则表达式，请选择 **Regular Expression**。

c. 选择 **Add Label** 来添加更多路由标签。

6. 选择 **Create** 来创建接收器。

6.7. 应用自定义 ALERTMANAGER 配置

您可以通过编辑 **openshift-monitoring** 项目中的 **alertmanager-main** secret，覆盖默认的 Alertmanager 配置。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

要从 CLI 更改 Alertmanager 配置：

1. 将当前活跃的 Alertmanager 配置输出到 **alertmanager.yaml** 文件：

```
$ oc -n openshift-monitoring get secret alertmanager-main --template='{{ index .data "alertmanager.yaml" }}' | base64 --decode > alertmanager.yaml
```

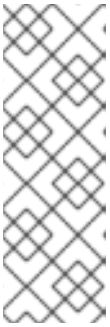
2. 编辑 **alertmanager.yaml** 中的配置：

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - matchers:
    - "alertname=Watchdog"
    repeat_interval: 5m
    receiver: watchdog
  - matchers:
    - "service=<your_service>" (1)
    routes:
    - matchers:
      - <your_matching_rules> (2)
      receiver: <receiver> (3)
receivers:
- name: default
- name: watchdog
- name: <receiver>
#  <receiver_configuration>
```

(1) **service** 指定触发警报的服务。

(2) **<your_matching_rules>** 指定目标警报。

(3) **receiver** 指定用于该警报的接收器。



注意

使用 **matchers** 键名称来指示警报要与节点匹配的匹配者。不要使用 **match** 或 **match_re** 键名称，它们都已弃用，并计划在以后的发行版本中删除。

另外，如果您定义了禁止规则，请使用 **target_matchers** 键名称来指示目标匹配者和 **source_matchers** 键名称来指示源匹配器。不要使用 **target_match**,**target_match_re**,**source_match**, 或 **source_match_re** 键名称，这些名称已弃用，并计划在以后的发行版本中删除。

以下 Alertmanager 配置示例将 PagerDuty 配置为警报接收器：

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - matchers:
    - "alertname=Watchdog"
    repeat_interval: 5m
    receiver: watchdog
  - matchers: - "service=example-app" routes: - matchers: - "severity=critical"
receiver: team-frontend-page
receivers:
- name: default
- name: watchdog
- name: team-frontend-page pagerduty_configs: - service_key: "your-key"
```

采用此配置时，由 **example-app** 服务触发的、严重性为 **critical** 的警报将使用 **team-frontend-page** 接收器发送。通常情况下，这些类型的警报会传给个人或关键响应团队。

3. 应用文件中的新配置：

```
$ oc -n openshift-monitoring create secret generic alertmanager-main --from-file=alertmanager.yaml --dry-run=client -o=yaml |  oc -n openshift-monitoring replace secret --filename=-
```

要从 OpenShift Container Platform Web 控制台更改 Alertmanager 配置：

1. 进入到 web 控制台的 **Administration → Cluster Settings → Configuration → Alertmanager → YAML** 页面。
2. 修改 YAML 配置文件。
3. 选择 **Save**。

其他资源

- 参阅 PagerDuty 官方网站来进一步了解 PagerDuty
- 参阅 PagerDuty Prometheus 集成指南来学习如何检索 **service_key**

- 参阅 Alertmanager 配置来配置通过不同警报接收器发送警报

6.8. 后续步骤

- 查看监控仪表板

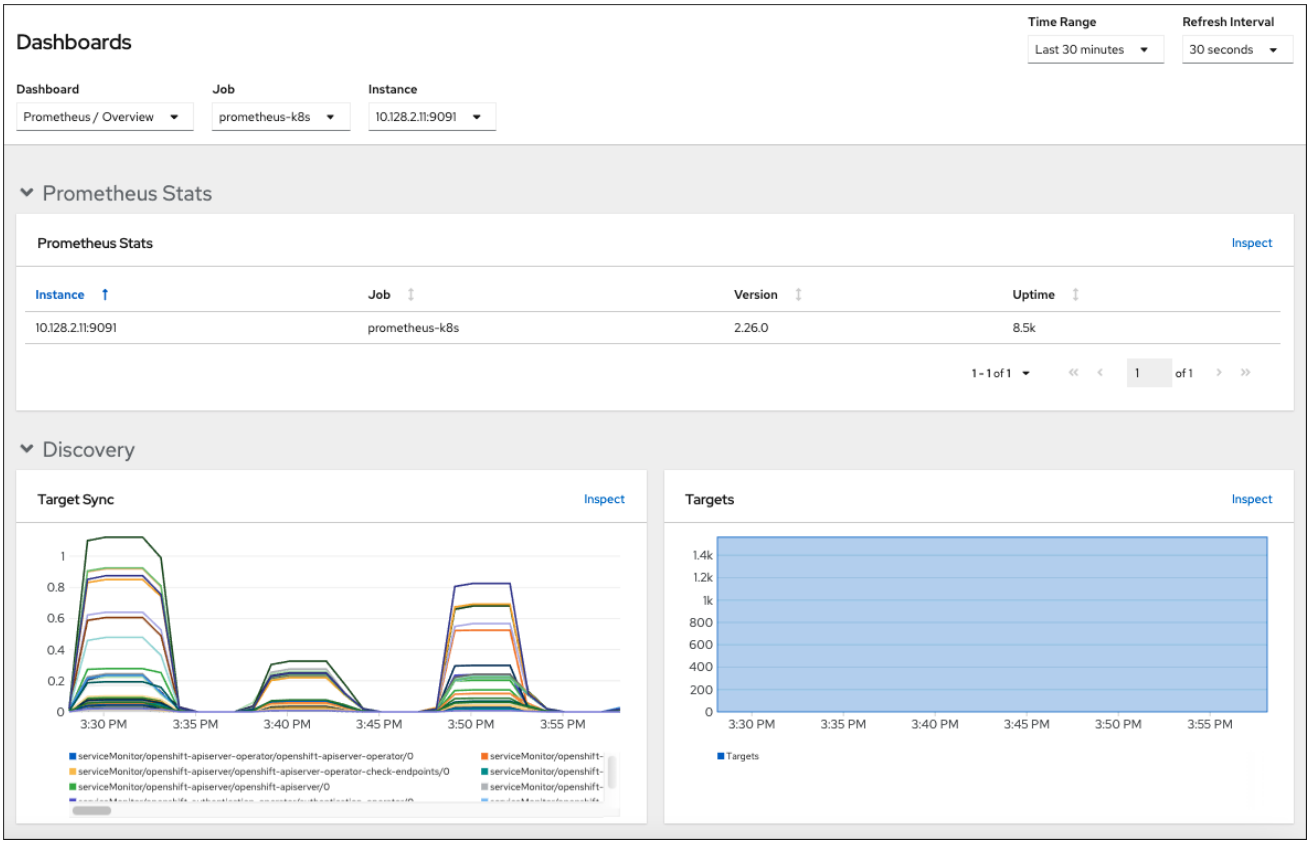
第 7 章 查看监控仪表板

OpenShift Container Platform 4.9 提供了一组全面的监控仪表板，可帮助您了解集群组件和用户定义的工作负载的状态。

使用 **Administrator** 视角访问 OpenShift Container Platform 核心组件的仪表板，包括以下项目：

- API 性能
- etcd
- Kubernetes 计算资源
- Kubernetes 网络资源
- Prometheus
- 与集群和节点性能相关的 USE 方法仪表板

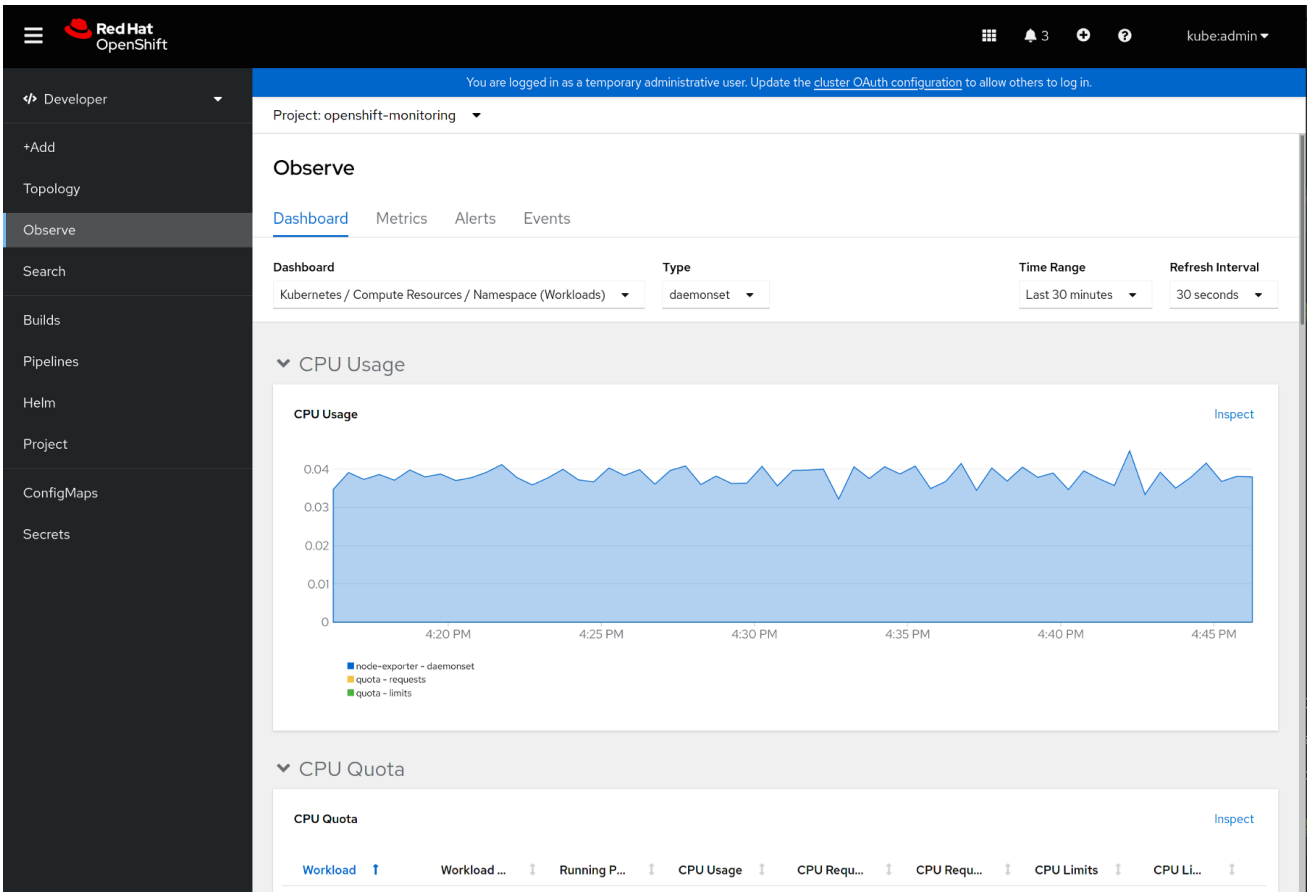
图 7.1. Administrator 视角中的仪表板示例



使用 **Developer** 视角访问为所选项目提供以下应用程序指标的 Kubernetes 计算资源仪表板：

- CPU 用量
- 内存用量
- 带宽信息
- 数据包速率信息

图 7.2. Developer 视角中的仪表板示例



注意

在 **Developer** 视角中，您一次只能查看一个项目的仪表板。

7.1. 以集群管理员身份查看监控仪表板

在 **Administrator** 视角中，您可以查看与 OpenShift Container Platform 集群核心组件相关的仪表板。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 在 OpenShift Container Platform web 控制台的 **Administrator** 视角中，进入到 **Observe → Dashboards**。
2. 在 **Dashboard** 列表中选择一个仪表板。有些仪表板（如 **etcd** 和 **Prometheus** 仪表板）在被选中时会生成额外的子菜单。
3. 可选：在 **Time Range** 列表中为图形选择一个时间范围。
   - 选择预定义的时间段。
   - 通过选择 **Time Range** 列表中的 **Custom 时间范围** 来设置自定义时间范围。
     a. 输入或选择 **From** 和 **To** date and time。

b. 单击 **Save** 以保存自定义时间范围。

4. 可选：选择一个 **Refresh Interval**。
5. 将鼠标悬停在仪表板中的每个图形上，以显示具体项目的详细信息。

7.2. 以开发者身份查看监控仪表板

使用 Developer 视角查看所选项目的 Kubernetes 计算资源仪表板。

先决条件

- 您可以使用开发人员或用户访问集群。
- 有您通过仪表板查看的项目的查看权限。

流程

1. 在 OpenShift Container Platform web 控制台的 Developer 视角中，导航到 **Observe → Dashboard**。
2. 从 **Project:** 下拉列表中选择一个项目。
3. 从 **Dashboard** 下拉列表中选择一个仪表板，以查看过滤的指标。

注意

选择时，所有仪表板会生成额外的子菜单，但 **Kubernetes / Compute Resources / Namespace(Pods)** 除外。

4. 可选：在 **Time Range** 列表中为图形选择一个时间范围。
   - 选择预定义的时间段。
   - 通过选择 **Time Range** 列表中的 **Custom 时间范围** 来设置自定义时间范围。
     a. 输入或选择 **From** 和 **To** date and time。
     b. 单击 **Save** 以保存自定义时间范围。
5. 可选：选择一个 **Refresh Interval**。
6. 将鼠标悬停在仪表板中的每个图形上，以显示特定项目的详细信息。

其他资源

- 使用 Developer 视角监控项目和应用程序的指标

7.3. 后续步骤

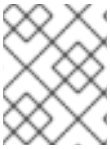
- 访问第三方 UI

第 8 章 访问第三方 UI

OpenShift Container Platform Web 控制台中提供集成的 Metrics、Alerting 和 Dashboard UI。如需了解关于使用这些集成 UI 的详细信息，请参阅以下内容：

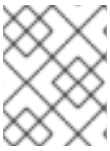
- 管理指标
- 管理警报
- 查看监控仪表板

OpenShift Container Platform 还提供对 Prometheus、Alertmanager 和 Grafana 第三方接口的访问。一些额外平台组件的仪表板包括在 OpenShift Container Platform Web 控制台的 **Monitoring → Dashboards** 中。



注意

以后的 OpenShift Container Platform 版本可能会移除对第三方监控接口的默认访问。之后，您需要使用端口转发来访问它们。



注意

随 OpenShift Container Platform 监控堆栈提供的 Grafana 实例及其仪表板是只读的。Grafana 仪表板仅包含 Kubernetes 和 **cluster-monitoring** 指标。

8.1. 使用 WEB 控制台访问第三方监控 UI

您可以通过 OpenShift Container Platform Web 控制台访问 Alertmanager、Grafana、Prometheus 和 Thanos Querier 的 Web UI。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 在 **Administrator** 视角中，导航至 **Networking → Routes**。

注意

从 Developer 视角无法访问第三方 Alertmanager、Grafana、Prometheus 和 Thanos Querier UI。取而代之，使用 Developer 视角中的 Metrics UI 链接，其中包含所选项目的一些预定义 CPU、内存、带宽和网络数据包查询。

2. 选择 **Project** 列表中的 **openshift-monitoring** 项目。
3. 访问第三方监控 UI：
   - 选择 **alertmanager-main** 行中的 URL 来打开 Alertmanager UI 的登录页面。
   - 选择 **grafana** 行中的 URL 来打开 Grafana UI 的登录页面。
   - 选择 **prometheus-k8s** 行中的 URL 来打开 Prometheus UI 的登录页面。

- 选择 **thanos-querier** 行中的 URL 来打开 Thanos Querier UI 的登录页面。

4. 选择 **Log in with OpenShift** 来使用您的 OpenShift Container Platform 凭证进行登录。

8.2. 使用 CLI 访问第三方监控 UI

您可以使用 OpenShift CLI (**oc**) 工具来获取 Prometheus、Alertmanager 和 Grafana Web UI 的 URL。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 运行以下命令来列出 **openshift-monitoring** 项目的路由：

```
$ oc -n openshift-monitoring get routes
```

输出示例

```
NAME                HOST/PORT                                                   ...
alertmanager-main   alertmanager-main-openshift-monitoring.apps._url_.openshift.com ...
grafana             grafana-openshift-monitoring.apps._url_.openshift.com       ...
prometheus-k8s      prometheus-k8s-openshift-monitoring.apps._url_.openshift.com    ...
thanos-querier      thanos-querier-openshift-monitoring.apps._url_.openshift.com    ...
```

2. 使用 Web 浏览器导航到 **HOST/PORT** 路由。
3. 选择 **Log in with OpenShift** 来使用您的 OpenShift Container Platform 凭证进行登录。

重要

监控路由由 Cluster Monitoring Operator 管理，用户不可修改。

第 9 章 监控问题的故障排除

9.1. 检查为什么用户定义的指标不可用

通过 **ServiceMonitor** 资源，您可以确定如何使用用户定义的项目中的服务公开的指标。如果您创建了 **ServiceMonitor** 资源，但无法在 Metrics UI 中看到任何对应的指标，请按该流程中所述的步骤操作。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您已为用户定义的工作负载启用并配置了监控。
- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 您已创建了 **ServiceMonitor** 资源。

流程

1. 在服务和 **ServiceMonitor** 资源配置中检查对应的标签是否匹配。
   a. 获取服务中定义的标签。以下示例在 **ns1** 项目中查询 **prometheus-example-app** 服务：

   ```
   $ oc -n ns1 get service prometheus-example-app -o yaml
   ```

   输出示例

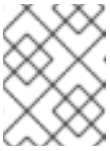
   ```
   labels:
     app: prometheus-example-app
   ```

   b. 检查 **ServiceMonitor** 资源配置中的 **matchLabels app** 标签是否与上一步中的标签输出匹配：

   ```
   $ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
   ```

   输出示例

   ```
   spec:
     endpoints:
     - interval: 30s
       port: web
       scheme: http
     selector:
       matchLabels:
         app: prometheus-example-app
   ```

   注意

   您可以作为具有项目查看权限的开发者检查服务和 **ServiceMonitor** 资源标签。

2. 在 **openshift-user-workload-monitoring** 项目中检查 **Prometheus Operator** 的日志。

   a. 列出 **openshift-user-workload-monitoring** 项目中的 Pod：

   ```
   $ oc -n openshift-user-workload-monitoring get pods
   ```

   输出示例

   ```
   NAME                                   READY   STATUS    RESTARTS   AGE
   prometheus-operator-776fcbbd56-2nbfm   2/2     Running   0          132m
   prometheus-user-workload-0             5/5     Running   1          132m
   prometheus-user-workload-1             5/5     Running   1          132m
   thanos-ruler-user-workload-0           3/3     Running   0          132m
   thanos-ruler-user-workload-1           3/3     Running   0          132m
   ```

   b. 从 **prometheus-operator** Pod 中的 **prometheus-operator** 容器获取日志。在以下示例中，Pod 名为 **prometheus-operator-776fcbbd56-2nbfm**：

   ```
   $ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-
   2nbfm -c prometheus-operator
   ```

   如果服务监控器出现问题，日志可能包含类似本例的错误：

   ```
   level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829
   component=prometheusoperator msg="skipping servicemonitor" error="it accesses file
   system via bearer token file which Prometheus specification prohibits"
   servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring
   prometheus=user-workload
   ```

3. 直接在 Prometheus UI 中查看项目的目标状态。

   a. 建立端口转发到 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例：

   ```
   $ oc port-forward -n openshift-user-workload-monitoring pod/prometheus-user-workload-
   0 9090
   ```

   b. 在 Web 浏览器中打开 http://localhost:9090/targets，并在 Prometheus UI 中直接查看项目的目标状态。检查与目标相关的错误消息。

4. 在 **openshift-user-workload-monitoring** 项目中为 **Prometheus Operator** 配置 **debug** 级别的日志记录。

   a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

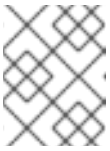
   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-
   config
   ```

   b. 在 **data/config.yaml** 下为 **prometheusOperator** 添加 **logLevel: debug**，将日志级别设置为 **debug**：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
   ```

```
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      prometheusOperator:
        logLevel: debug
```

c. 保存文件以使改变生效。



注意

在应用日志级别更改时，**openshift-user-workload-monitoring** 项目中的 **prometheus-operator** 会自动重启。

d. 确认 **debug** 日志级别已应用到 **openshift-user-workload-monitoring** 项目中的 **prometheus-operator** 部署：

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

输出示例

```
        - --log-level=debug
```

Debug 级别日志记录将显示 Prometheus Operator 发出的所有调用。

e. 检查 **prometheus-operator** Pod 是否正在运行：

```
$ oc -n openshift-user-workload-monitoring get pods
```

注意

如果配置映射中包含了一个未识别的 Prometheus Operator **loglevel** 值，则 **prometheus-operator** Pod 可能无法成功重启。

f. 查看 debug 日志，以了解 Prometheus Operator 是否在使用 **ServiceMonitor** 资源。查看日志中的其他相关错误。

其他资源

- 创建用户定义的工作负载监控配置映射
- 如需有关如何创建 **ServiceMonitor** 或 **PodMonitor** 的详细信息，请参阅指定如何监控服务

9.2. 确定为什么 PROMETHEUS 消耗大量磁盘空间

开发人员可以使用键值对的形式为指标定义属性。潜在的键值对数量与属性的可能值数量对应。具有无限数量可能值的属性被称为未绑定属性。例如，**customer_id** 属性不绑定，因为它有无限多个可能的值。

每个分配的键值对都有唯一的时间序列。在标签中使用许多未绑定属性可导致所创建的时间序列数量出现指数增加。这可能会影响 Prometheus 性能，并消耗大量磁盘空间。

当 Prometheus 消耗大量磁盘时，您可以使用以下方法：

- **检查正在收集的提取示例数量。**
- **检查 Prometheus UI 中的时间序列数据库（TSDB）状态以了解有关哪些标签创建最多时间序列的更多信息。这需要集群管理员特权。**
- **要减少创建的唯一时间序列数量，**您可以减少分配给用户定义的指标的未绑定属性数量。

注意

使用绑定到一组有限可能值的属性可减少潜在的键-值对组合数量。

- **对可在用户定义的项目中提取的示例数量实施限制。**这需要集群管理员特权。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 在 **Administrator** 视角中，进入到 **Observe → Metrics**。
2. 在 **Expression** 字段中运行以下 Prometheus Query Language (PromQL) 查询。这会返回具有最高提取示例数的十个指标：

```
topk(10,count by (job)({__name__=~".+"}))
```

3. 如果指标的提取示例数大于预期，请检查分配给指标的未绑定标签值数量。
   - **如果指标与用户定义的项目相关，**请查看分配给您的工作负载的指标键-值对。它们通过应用程序级别的 Prometheus 客户端库实施。尝试限制标签中引用的未绑定属性数量。
   - **如果指标与 OpenShift Container Platform 核心项目相关，**请在红帽客户门户网站上创建一个红帽支持问题单。
4. 检查 Prometheus UI 中的 TSDB 状态。
   a. 在 **Administrator** 视角中，导航至 **Networking → Routes**。
   b. 选择 **Project** 列表中的 **openshift-monitoring** 项目。
   c. 选择 **prometheus-k8s** 行中的 URL 来打开 Prometheus UI 的登录页面。
   d. 选择 **Log in with OpenShift** 来使用您的 OpenShift Container Platform 凭证进行登录。
   e. 在 Prometheus UI 中，进入到 **Status → TSDB Status**。

其他资源

- 如需有关如何设置提取示例限制和创建相关警报规则的详细信息，请参阅为用户定义的项目设置提取示例限制
- 提交支持问题单