



OpenShift Container Platform 4.14

监控

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

OpenShift Container Platform 4.14 监控

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

Legal Notice

Copyright © 2025 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

使用监控堆栈提供的指标和自定义警报来跟踪 OpenShift Container Platform 集群上运行的应用程序的健康状态和性能。

Table of Contents

第 1 章 关于 OPENSIFT CONTAINER PLATFORM 监控	4
1.1. 关于 OPENSIFT CONTAINER PLATFORM 监控	4
1.2. 监控堆栈架构	4
1.3. 了解监控堆栈 - 主要概念	12
第 2 章 开始使用	26
2.1. 对监控的维护和支持	26
2.2. 核心平台监控第一步	28
2.3. 用户工作负载监控第一步	29
2.4. 开发人员和非管理员用户步骤	29
第 3 章 配置核心平台监控	30
3.1. 准备配置核心平台监控堆栈	30
3.2. 为核心平台监控配置性能和可扩展性	34
3.3. 为核心平台监控存储和记录数据	43
3.4. 为核心平台监控配置指标	55
3.5. 为核心平台监控配置警报和通知	66
第 4 章 配置用户工作负载监控	79
4.1. 准备配置用户工作负载监控堆栈	79
4.2. 为用户工作负载监控配置性能和可扩展性	89
4.3. 为用户工作负载监控存储和记录数据	99
4.4. 为用户工作负载监控配置指标	108
4.5. 为用户工作负载监控配置警报和通知	125
第 5 章 访问指标	134
5.1. 以管理员身份访问指标	134
5.2. 以开发者身份访问指标	138
5.3. 使用 CLI 访问监控 API	141
第 6 章 管理警报	147
6.1. 以管理员身份管理警报	147
6.2. 将警报作为开发者管理	156
第 7 章 监控问题的故障排除	163
7.1. 调查用户定义的项目指标不可用的原因	163
7.2. 确定为什么 PROMETHEUS 消耗大量磁盘空间	166
7.3. 解决 PROMETHEUS 的 KUBEPERSISTENTVOLUMEFILLINGUP 警报触发的问题	167
7.4. 解决 ALERTMANAGERRECEIVERSNOTCONFIGURED 警报	169
第 8 章 CLUSTER MONITORING OPERATOR 的配置映射引用	170
8.1. CLUSTER MONITORING OPERATOR 配置参考	170
8.2. ADDITIONALALERTMANAGERCONFIG	170
8.3. ALERTMANAGERMAINCONFIG	171
8.4. ALERTMANAGERUSERWORKLOADCONFIG	172
8.5. CLUSTERMONITORINGCONFIGURATION	173
8.6. DEDICATEDSERVICEMONITORS	174
8.7. K8SPROMETHEUSADAPTER	175
8.8. KUBESTATEMETRICSCONFIG	175
8.9. PROMETHEUSOPERATORADMISSIONWEBHOOKCONFIG	176
8.10. MONITORINGPLUGINCONFIG	176
8.11. NODEEXPORTERCOLLECTORBUDDYINFOCONFIG	177
8.12. NODEEXPORTERCOLLECTORCONFIG	177

8.13. NODEEXPORTERCOLLECTORCPUFREQCONFIG	178
8.14. NODEEXPORTERCOLLECTORKSMDCONFIG	178
8.15. NODEEXPORTERCOLLECTORMOUNTSTATSCONFIG	179
8.16. NODEEXPORTERCOLLECTORNETCLASSCONFIG	179
8.17. NODEEXPORTERCOLLECTORNETDEVCONFIG	180
8.18. NODEEXPORTERCOLLECTORPROCESSESCONFIG	180
8.19. NODEEXPORTERCOLLECTORSYSTEMDCONFIG	181
8.20. NODEEXPORTERCOLLECTORTCPSTATCONFIG	181
8.21. NODEEXPORTERCONFIG	182
8.22. OPENSIFTSTATEMETRICSCONFIG	182
8.23. PROMETHEUSK8SCONFIG	183
8.24. PROMETHEUSOPERATORCONFIG	185
8.25. PROMETHEUSRESTRICTEDCONFIG	185
8.26. REMOTEWITESPEC	188
8.27. TLSCONFIG	189
8.28. TELEMETERCLIENTCONFIG	190
8.29. THANOSQUERIERCONFIG	190
8.30. THANOSRULERCONFIG	191
8.31. USERWORKLOADCONFIGURATION	192

第 1 章 关于 OPENSHIFT CONTAINER PLATFORM 监控

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。通过 OpenShift Container Platform Web 控制台，您可以[访问指标](#)和[管理警报](#)。

安装 OpenShift Container Platform 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回答，如用户指标不可用，以及 Prometheus [对监控问题进行故障排除](#)的磁盘空间高可用性。

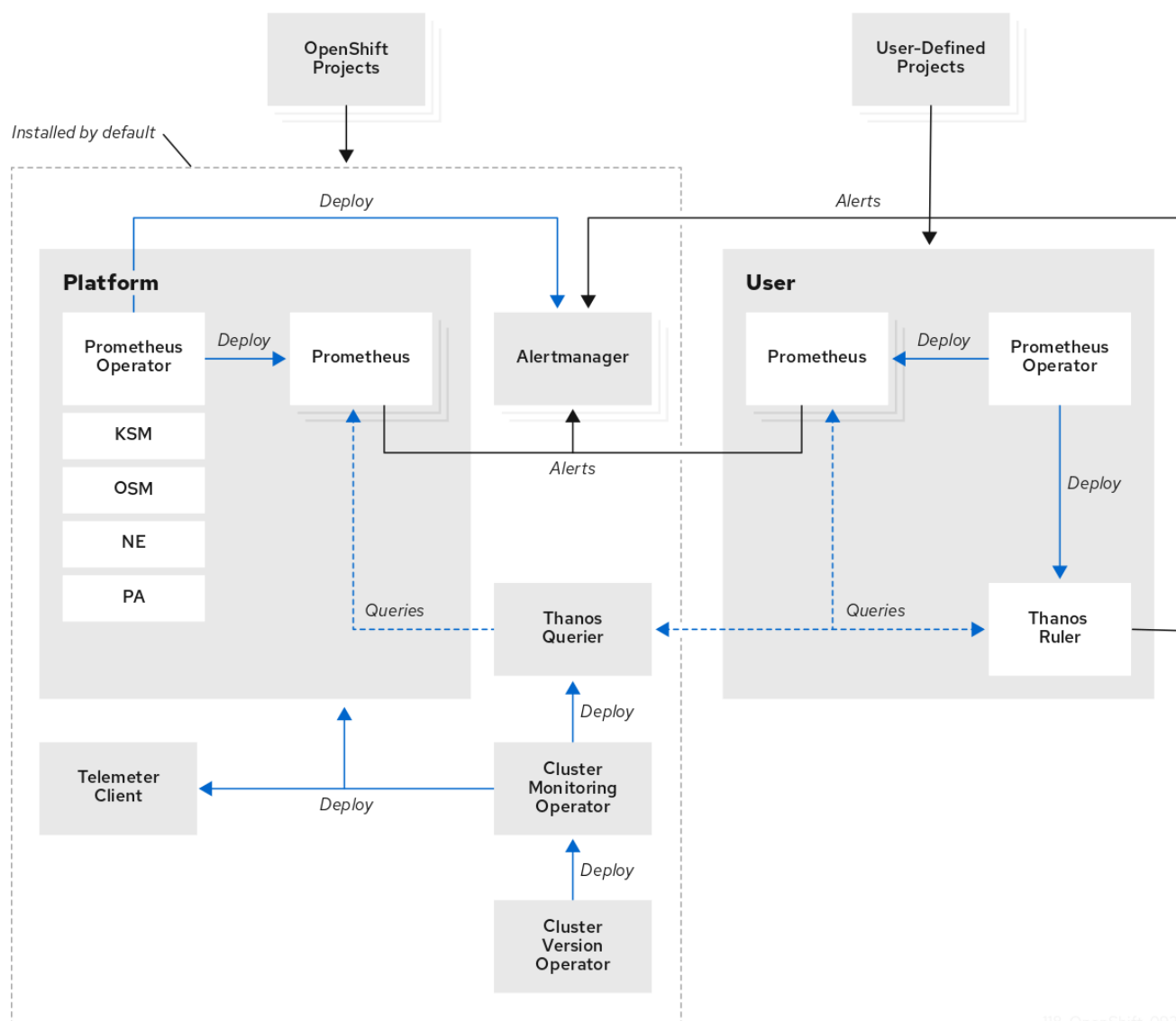
1.2. 监控堆栈架构

OpenShift Container Platform 监控堆栈基于 [Prometheus](#) 开源项目及其更广的生态系统。监控堆栈包括用于监控用户定义的项目的默认监控组件和组件。

1.2.1. 了解监控堆栈

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 服务在内的核心集群组件提供了监控。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。



118_OpenShift_092C

1.2.2. 默认监控组件

默认情况下，OpenShift Container Platform 4.14 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、Telemeter 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。
monitoring-plugin	monitoring-plugin 动态插件组件在 OpenShift Container Platform Web 控制台的 Observe 部分中部署监控页面。您可以使用 Cluster Monitoring Operator (CMO) 配置映射设置来管理 web 控制台页面的 monitoring-plugin 资源。
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）会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，并选择性地重复数据删除。
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。

监控堆栈监控堆栈内的所有组件。更新 OpenShift Container Platform 时，组件会自动更新。

1.2.2.1. 默认监控目标

除了堆栈本身的组件外，默认的监控堆栈还会监控额外的平台组件。

以下是监控目标的示例：

- CoreDNS
- etcd
- HAProxy
- 镜像 registry
- Kubelets
- Kubernetes API 服务器
- Kubernetes 控制器管理器
- Kubernetes 调度程序
- OpenShift API 服务器
- OpenShift Controller Manager
- Operator Lifecycle Manager (OLM)



注意

- 具体目标列表可能会因集群功能和安装的组件而异。
- 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。

其他资源

- [获取有关指标目标的详细信息](#)

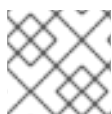
1.2.3. 用于监控用户定义的项目的组件

OpenShift Container Platform 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 中，Thanos Ruler 为监控用户定义的项目提供规则和警报评估。
Alertmanager	Alertmanager 服务处理从 Prometheus 和 Thanos Ruler 接收的警报。Alertmanager 还负责将用户定义的警报发送到外部通知系统。部署该服务是可选的。



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。

所有这些组件都由堆栈监控，并在 OpenShift Container Platform 更新时自动更新。

1.2.3.1. 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。

1.2.4. 高可用性集群中的监控堆栈

默认情况下，在多节点集群中，以下组件以高可用性(HA)模式运行，以防止数据丢失和服务中断：

- Prometheus
- Alertmanager
- Thanos Ruler
- Thanos querier
- Prometheus Adapter
- 监控插件

组件在两个 pod 之间复制，每个 pod 在单独的节点上运行。这意味着监控堆栈可以容忍一个 pod 的丢失。

Prometheus 处于 HA 模式

- 这两个副本都独立提取同一目标并评估相同的规则。
- 副本不会相互通信。因此，数据在 pod 之间可能会有所不同。

处于 HA 模式的 Alertmanager

- 这两个副本将相互同步通知和静默状态。这样可确保每个通知至少发送一次。
- 如果副本间无法通信，或者在接收端有问题，则通知仍然会被发送，但可能会重复。



重要

Prometheus、Alertmanager 和 Thanos Ruler 是有状态组件。为确保高可用性，您必须为它们配置持久性存储。

其他资源

- [高可用性或单节点集群检测和支持](#)
- [配置持久性存储](#)
- [配置性能和可扩展性](#)

1.2.5. 监控堆栈中的 TLS 安全性和轮转

了解 TLS 配置集和证书轮转如何在 OpenShift Container Platform 监控堆栈中工作，以保持通信安全。

监控组件的 TLS 安全配置集

监控堆栈的所有组件都使用集群管理员集中配置的 TLS 安全配置集设置。监控堆栈组件使用全局 OpenShift Container Platform **apiservers.config.openshift.io/cluster** 资源中的 **tlsSecurityProfile** 字段中已存在的 TLS 安全配置集设置。

TLS 证书轮转和自动重启

Cluster Monitoring Operator 管理监控组件的内部 TLS 证书生命周期。这些证书保护监控组件之间的内部通信。

在证书轮转过程中，CMO 会更新 secret 和配置映射，这会触发受影响 pod 的自动重启。这是一个预期的行为，pod 会自动恢复。

以下示例显示了证书轮转过程中发生的事件：

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

LAST SEEN	TYPE	REASON	OBJECT	MESSAGE
2h39m	Normal	SecretUpdated	deployment/cluster-monitoring-operator	Updated Secret/grpc-tls -n openshift-monitoring because it changed
2h39m	Normal	SecretCreated	deployment/cluster-monitoring-operator	Created Secret/prometheus-user-workload-grpc-tls -n openshift-user-workload-monitoring because it was missing
2h39m	Normal	SecretCreated	deployment/cluster-monitoring-operator	Created Secret/thanos-querier-grpc-tls -n openshift-monitoring because it was missing
2h39m	Normal	SecretCreated	deployment/cluster-monitoring-operator	Created Secret/thanos-ruler-grpc-tls -n openshift-user-workload-monitoring because it was missing
2h39m	Normal	SecretCreated	deployment/cluster-monitoring-operator	Created Secret/prometheus-k8s-grpc-tls -n openshift-monitoring because it was missing
2h38m	Warning	FailedMount	pod/prometheus-k8s-0	MountVolume.SetUp failed for volume "secret-grpc-tls" : secret "prometheus-k8s-grpc-tls" not found
2h39m	Normal	Created	pod/prometheus-k8s-0	Created container kube-rbac-proxy-thanos
2h39m	Normal	Started	pod/prometheus-k8s-0	Started container kube-rbac-proxy-thanos

2h39m	Normal	SuccessfulDelete	statefulset/prometheus-k8s	delete Pod
prometheus-k8s-0 in StatefulSet prometheus-k8s successful				
2h39m	Normal	SuccessfulCreate	statefulset/prometheus-k8s	create Pod
prometheus-k8s-0 in StatefulSet prometheus-k8s successful				

其他资源

- [配置 TLS 安全配置集](#)

1.2.6. 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 是一个日志收集器，它驻留在每个 OpenShift Container Platform 节点上。它收集应用程序、基础架构和审计日志并将其转发到不同的输出。



注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

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。

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。

Vector

Vector 是一个日志收集器，它部署到每个 OpenShift Container Platform 节点。它从每个节点收集日志数据，转换数据并将其转发到配置的输出。

Web 控制台

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

1.2.7. 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](#)

1.3. 了解监控堆栈 - 主要概念

熟悉 OpenShift Container Platform 监控概念和术语。了解如何提高集群的性能和规模，存储和记录数据，管理指标和警报等。

1.3.1. 关于性能和可扩展性

您可以优化集群的性能和扩展。您可以通过执行以下操作来配置默认的监控堆栈：

- 控制监控组件的放置和分发：
 - 使用节点选择器将组件移到特定的节点。
 - 分配容限以启用将组件移到污点节点。
- 使用 pod 拓扑分布限制。
- 为指标提取设置正文大小限制。
- 管理 CPU 和内存资源。
- 使用指标集合配置集。

其他资源

- [为核心平台监控配置性能和可扩展性](#)
- [为用户工作负载监控配置性能和可扩展性](#)

1.3.1.1. 使用节点选择器移动监控组件

通过将 **nodeSelector** 约束与标记的节点搭配使用，您可以将任何监控堆栈组件移到特定的节点上。通过这样做，您可以控制集群中监控组件的放置和分发。

通过控制监控组件的放置和分发，您可以根据特定要求或策略优化系统资源使用、提高性能和隔离工作负载。

节点选择器与其他约束一起使用

如果使用节点选择器约束移动监控组件，请注意集群可能存在其他限制来控制 pod 调度：

- 拓扑分布约束可能处于放置状态来控制 pod 放置。
- Prometheus、Alertmanager 和其他监控组件会放置硬反关联性规则，以确保这些组件的多个 pod 始终分散到不同的节点上，因此始终具有高可用性。

将 pod 调度到节点时，pod 调度程序会在决定 pod 放置时尝试满足所有现有的限制。也就是说，当 pod 调度程序决定将哪些 pod 放置到哪些节点上时，所有约束都会编译。

因此，如果您配置节点选择器约束，但无法满足现有的约束，pod 调度程序无法与所有约束匹配，也不会调度 pod 放置到节点上。

为保持监控组件的弹性和高可用性，请确保有足够的节点可用，并在配置节点选择器约束以移动组件时匹配所有约束。

1.3.1.2. 关于用于监控的 pod 拓扑分布限制

当 OpenShift Container Platform pod 部署到多个可用区时，您可以使用 pod 拓扑分布约束来控制监控 pod 如何分散到网络拓扑中。

Pod 拓扑分布约束适合在分层拓扑内控制 pod 调度，节点分散到不同的基础架构级别，如这些区域内的地区和区域。另外，通过能够在不同区中调度 pod，您可以在某些情况下提高网络延迟。

您可以为 Cluster Monitoring Operator 部署的所有 pod 配置 pod 拓扑分布限制，以控制如何在区调度到节点的 pod 副本。这样可确保 pod 具有高可用性并更有效地运行，因为工作负载分散在不同的数据中心或分层基础架构区域中。

1.3.1.3. 关于为监控组件指定限制和请求

您可以为以下核心平台监控组件配置资源限值和请求：

- Alertmanager
- kube-state-metrics
- monitoring-plugin
- node-exporter
- openshift-state-metrics
- Prometheus
- Prometheus Adapter
- Prometheus Operator 及其准入 Webhook 服务
- Telemeter Client
- Thanos querier

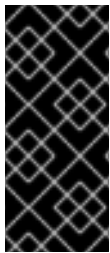
您可以为监控用户定义的项目的以下组件配置资源限值和请求：

- Alertmanager
- Prometheus
- Thanos Ruler

通过定义资源限值，您可以限制容器的资源使用情况，这会阻止容器超过 CPU 和内存资源指定的最大值。

通过定义资源请求，您可以指定容器只能调度到具有足够 CPU 和内存资源的节点，以匹配请求的资源。

1.3.1.4. 关于指标集合配置集



重要

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

有关红帽技术预览功能支持范围的更多信息，请参阅[技术预览功能支持范围](#)。

默认情况下，Prometheus 会收集由 OpenShift Container Platform 组件中的所有默认指标目标公开的指标。但是，在某些情况下，您可能希望 Prometheus 从集群收集较少的指标：

- 集群管理员只需要警报、遥测和控制台指标，且不需要其他指标数据。
- 集群大小增加，且收集的默认指标数据的大小现在需要显著增加 CPU 和内存资源。

您可以使用指标集合配置集来收集默认指标数据数量或最小指标数据。当您收集最小指标数据时，警报等基本监控功能将继续工作。同时，Prometheus 所需的 CPU 和内存资源会减少。

您可以启用两个指标集合配置集之一：

- **full**：Prometheus 会收集由所有平台组件公开的指标数据。此设置是默认设置。
- **minimal**：Prometheus 仅收集平台警报、记录规则、遥测和控制台仪表板所需的指标数据。

1.3.2. 关于存储和记录数据

您可以存储和记录数据，以帮助您保护数据并使用它们进行故障排除。您可以通过执行以下操作来配置默认的监控堆栈：

- 配置持久性存储：
 - 通过将指标和警报数据存储在持久性卷(PV)中来保护您的指标和警报数据。因此，它们可以在 pod 重启或重新创建后保留。
 - 避免获取重复的通知，并在 Alertmanager pod 重启时丢失警报静默。
- 修改 Prometheus 和 Thanos Ruler 指标数据的保留时间和大小。
- 配置日志记录以帮助您排除集群的问题：
 - 为 Metrics 服务器配置审计日志。
 - 为监控设置日志级别。
 - 为 Prometheus 和 Thanos Querier 启用查询日志记录。

其他资源

- [为核心平台监控存储和记录数据](#)
- [为用户工作负载监控存储和记录数据](#)

1.3.2.1. Prometheus 指标的保留时间和大小

默认情况下，Prometheus 会在以下持续时间内保留指标数据：

- **核心平台监控**：15 天
- **监控用户定义的项目**：24 小时

您可以修改 Prometheus 实例的保留时间，以更改在多久后删除数据。您还可以设置保留指标数据使用的最大磁盘空间量。如果数据达到这个大小限制，Prometheus 会首先删除最旧的数据，直到使用的磁盘空间重新低于限制。

请注意这些数据保留设置的行为：

- 基于大小的保留策略适用于 **/prometheus** 目录中的所有数据块目录，包括持久性块、写入级日志 (WAL) 数据和 mmaped 块。
- **/wal** 和 **/head_chunks** 目录中的数据计入保留大小限制，但 Prometheus 永远不会根据基于大小或基于时间的保留策略从这些目录中清除数据。因此，如果您设置了保留大小限制，它小于为 **/wal** 和 **/head_chunks** 目录设置的最大容量，则表示您将系统配置为不保留 **/prometheus** 数据目录中的任何数据块。
- 只有在 Prometheus 切断新的数据块时，才会应用基于大小的保留策略，即在 WAL 最多包含三小时数据后每两小时进行。
- 如果没有为 **retention** 或 **retentionSize** 明确定义值，则保留时间默认为 15 天，用于核心平台监控，为用户定义的项目监控 24 小时。不设置保留大小。
- 如果 **retention** 和 **retentionSize** 都定义了值，则会应用这两个值。如果任何数据块超过定义的保留时间或定义的大小限制，Prometheus 会清除这些数据块。
- 如果您为 **retentionSize** 定义了值，且没有定义 **retention**，则只应用 **retentionSize** 值。
- 如果您没有为 **retentionSize** 定义值，且只为 **retention** 定义了值，则只应用 **retention** 值。
- 如果将 **retentionSize** 或 **retention** 值设置为 **0**，则应用默认的设置。默认设置将核心平台监控的保留时间设置为 15 天，用户定义的项目监控为 24 小时。默认情况下，不会设置保留大小。



注意

数据压缩每两小时进行一次。因此，持久性卷 (PV) 可能会在压缩前已被填满，可能会超过 **retentionSize** 限制。在这种情况下，**KubePersistentVolumeFillingUp** 警报会触发，直到 PV 上的空间低于 **retentionSize** 限制。

1.3.3. 了解指标

在 OpenShift Container Platform 4.14 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。借助指标，您可以监控集群组件和您自己的工作负载的表现情况。

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** 示例应用程序公开路由，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

- [为核心平台监控配置指标](#)
- [为用户工作负载监控配置指标](#)
- [以管理员身份访问指标](#)
- [以开发者身份访问指标](#)

1.3.3.1. 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
- 限制提取标签数量、标签名称长度以及标签值长度
- 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。

1.3.3.2. 在指标中添加集群 ID 标签

如果您管理多个 OpenShift Container Platform 集群，并使用远程写入功能将指标数据从这些集群发送到外部存储位置，您可以添加集群 ID 标签来识别来自不同集群的指标数据。然后，您可以查询这些标签来标识指标的源集群，并区分与其他集群发送的类似指标数据的数据。

这样，如果您为多个客户管理多个集群，并将指标数据发送到单个集中存储系统，您可以使用集群 ID 标签查询特定集群或客户的指标。

创建并使用集群 ID 标签涉及三个常规步骤：

- 配置远程写入存储的写重新标记设置。

- 将集群 ID 标签添加到指标。
- 查询这些标签以标识源集群或指标客户。

1.3.4. 关于监控仪表板

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

其他资源

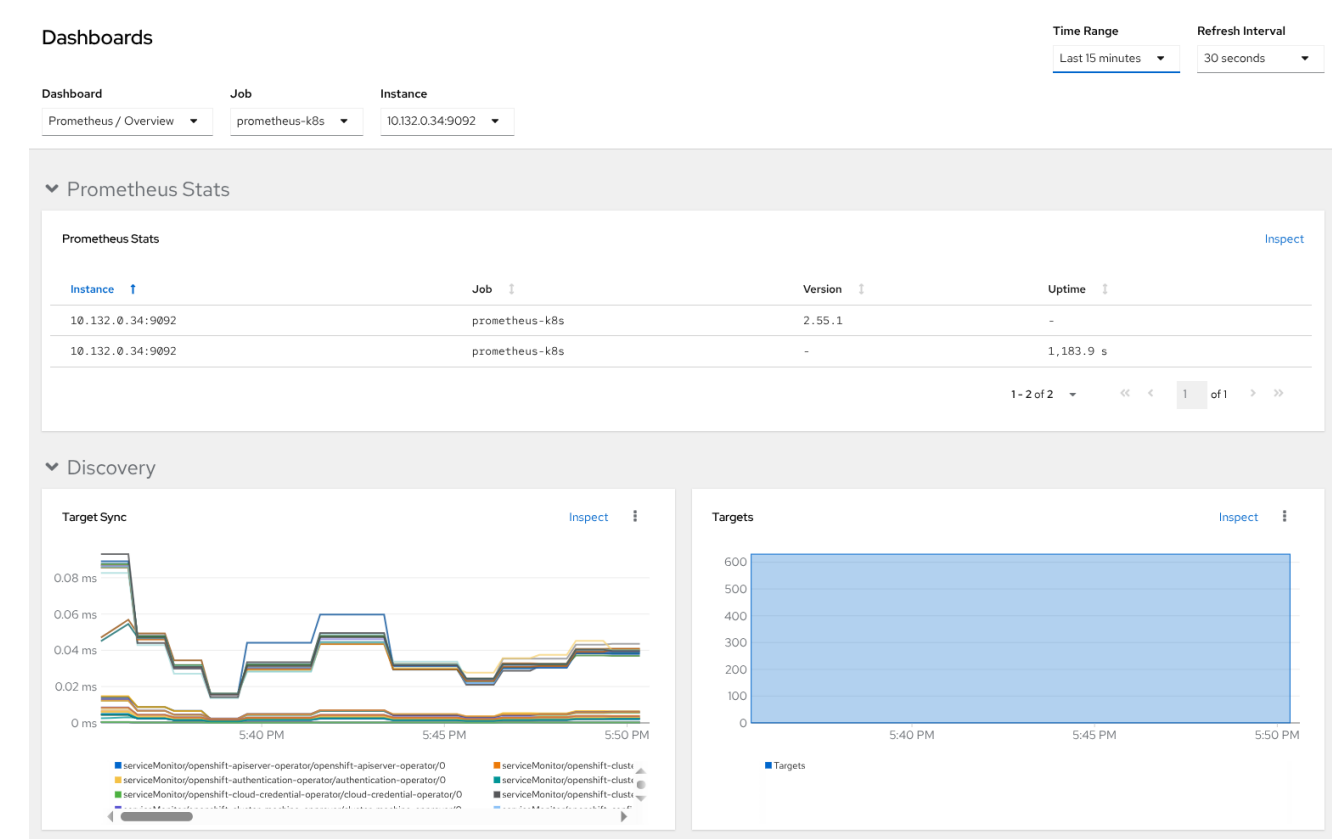
- [以集群管理员身份查看监控仪表板](#)
- [以开发者身份查看监控仪表板](#)

1.3.4.1. Administrator 视角中的监控仪表板

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
- 节点性能指标

图 1.1. Administrator 视角中的仪表板示例

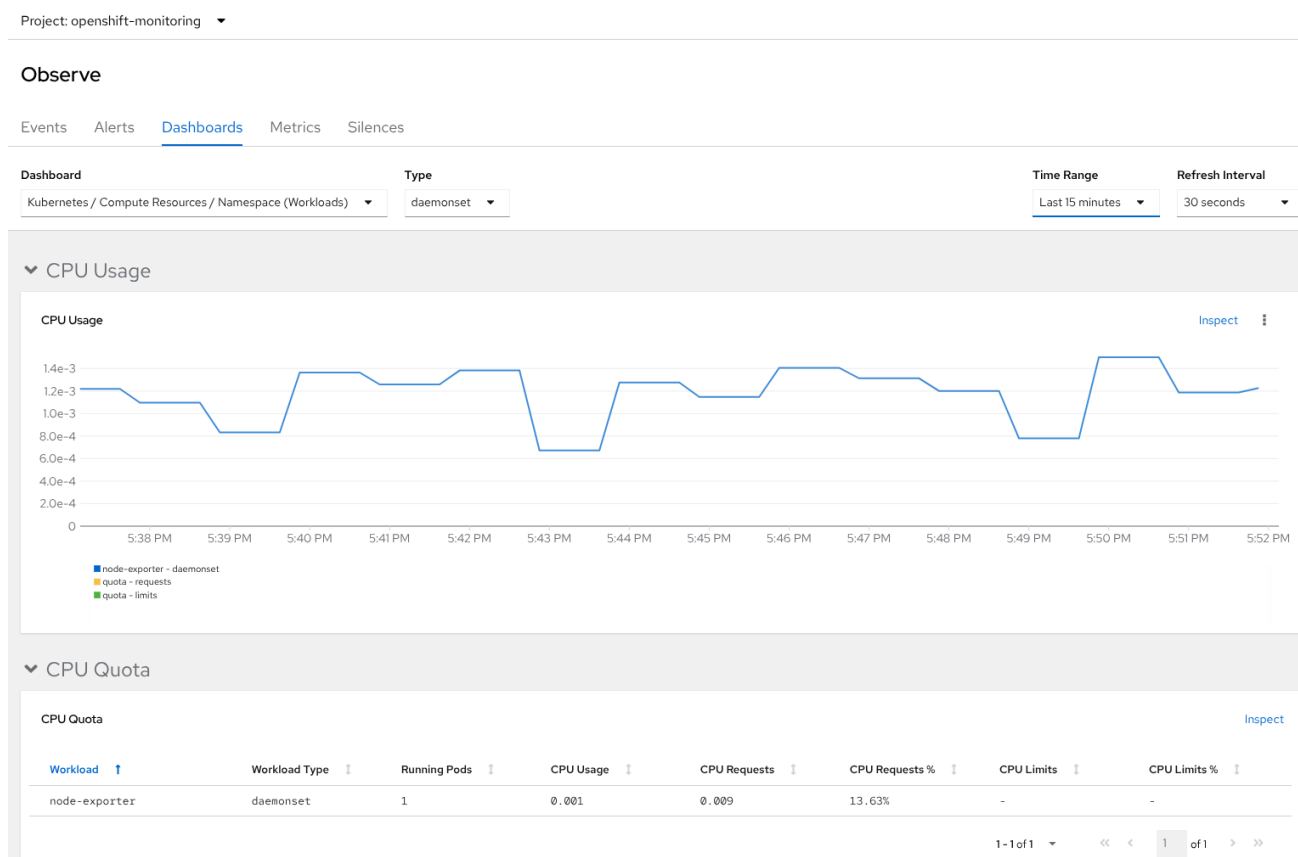


1.3.4.2. Developer 视角中的监控仪表板

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 1.2. Developer 视角中的仪表板示例



1.3.5. 管理警报

在 OpenShift Container Platform 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，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，如果您以具有 **cluster-admin** 角色的用户身份登录，您可以访问所有警报、静默和警报规则。

其他资源

- [为核心平台监控配置警报和通知](#)
- [为用户工作负载监控配置警报和通知](#)
- [以管理员身份管理警报](#)
- [将警报作为开发者管理](#)

1.3.5.1. 管理静默

您可以在 **Administrator** 和 **Developer** 视角中的 OpenShift Container Platform Web 控制台中为警报创建静默。创建静默后，不会在警报触发时收到有关警报的通知。

当您收到了初始警报通知，并且不希望在解决与这个初始警报相关的底层问题时接收到因为解决相关问题所触发的警报通知，可以创建静默。

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时间后到期。

在创建静默后，您可以查看、编辑这个静默，并可以使其过期。



注意

在创建静默时，它们会在 Alertmanager pod 之间复制。但是，如果您没有为 Alertmanager 配置持久性存储，静默可能会丢失。例如，如果所有 Alertmanager pod 同时重启，会出现这种情况。

1.3.5.2. 管理用于核心平台监控的警报规则

OpenShift Container Platform 监控包括一组用于平台指标的默认警报规则。作为集群管理员，您可以以两种方式自定义这组规则：

- 通过调整阈值或添加和修改标签来修改现有平台警报规则的设置。例如，您可以将一个警报的 **severity** 标签从 **warning** 改为 **critical** 以帮助您来处理相关的问题。
- 通过基于 **openshift-monitoring** 命名空间中的核心平台指标构建查询表达式，定义和添加新的自定义警报规则。

核心平台警报规则考虑

- 新的警报规则必须基于默认的 OpenShift Container Platform 监控指标。
- 您必须在 **openshift-monitoring** 命名空间中创建 **AlertingRule** 和 **AlertRelabelConfig** 对象。
- 您只能添加和修改警报规则。您无法创建新的记录规则或修改现有的记录规则。
- 如果您使用 **AlertRelabelConfig** 对象修改现有平台警报规则，您的修改不会反映在 Prometheus 警报 API 中。因此，任何丢弃的警报仍出现在 OpenShift Container Platform web 控制台中，即使它们不再被转发到 Alertmanager。另外，任何对警报的修改（如更改了 **severity** 标签）都不会出现在 web 控制台中。

1.3.5.3. 为核心平台监控优化警报规则的建议

如果您需要自定义核心平台警报规则来满足机构的特定需求，请遵循以下准则，确保自定义规则可以实现预期的目且高效。

- **使用尽可能少的新规则。** 仅创建符合您特定要求的规则。通过使用最少的规则数量，您可以在监控环境中创建更易管理且集中的警报系统。
- **专注于症状而不是原因。** 创建规则来通知用户症状而不是造成症状的根本原因。这可以确保，在出现相关症状时用户可以获得警报，用户可以调查触发警报的根本原因。使用这个策略，可以显著降低了您需要创建的规则的总数量。
- **在实施更改之前，规划并评估您的需求。** 首先，明确哪些症状是重要的，以及在出现这些症状时您希望用户执行哪些操作。然后，评估现有规则，决定是否可以通过修改这些规则来实现您的目

的，而不用为每个症状都创建一个新的规则。通过修改现有规则并谨慎创建新规则，可帮助您简化警报系统。

- **提供明确的警报信息。**当您创建警报消息时，包括症状的描述、可能的原因和推荐的操作。包括的信息应该明确、简明，并包括故障排除步骤或更多相关信息的链接。这样做有助于用户快速评估情况并做出适当响应。
- **包括严重性级别。**为您的规则分配严重性级别，以提示用户在出现症状并触发警报时如何响应。例如，将警报严重性级别定为 **关键（Critical）** 信号，代表相关人员需要马上做出响应。通过定义严重性级别，可以帮助用户决定在收到警报时应如何响应，并确保对紧急的问题马上做出响应。

1.3.5.4. 关于为用户定义的项目创建警报规则

如果您为用户定义的项目创建警报规则，请在定义新规则时请考虑以下关键行为和重要限制：

- 除了核心平台监控的默认指标外，用户定义的警报规则也可以包括由其自身项目公开的指标。您不能包含其他用户定义的项目的指标。
例如，**ns1** 用户定义的项目的警报规则除核心平台指标（如 CPU 和内存指标）外还可以使用 **ns1** 项目公开的指标。但是，该规则无法包含来自不同 **ns2** 用户定义的项目的指标。
- 要缩短延迟并最小化核心平台监控组件的负载，您可以将 **openshift.io/prometheus-rule-evaluation-scope: leaf-prometheus** 标签添加到规则中。此标签只强制 **openshift-user-workload-monitoring** 项目中部署的 Prometheus 实例评估警报规则，并防止 Thanos Ruler 实例这样做。



重要

如果警报规则具有此标签，则您的警报规则只能使用用户定义的项目公开的这些指标。您基于默认平台指标创建的警报规则可能无法触发警报。

1.3.5.5. 为用户定义的项目管理警报规则

在 OpenShift Container Platform 中，您可以查看、编辑和删除用户定义的项目中的警报规则。

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。

1.3.5.6. 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。

1.3.5.7. 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。

1.3.5.7.1. 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：

- **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。当条件保持 **true** 时，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。对于与所有列出的值或正则表达式匹配的警报，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。

1.3.5.7.2. 了解静默过滤器

在 **Administrator** 视角中，Alerting UI 中的 **Silenced** 页面针对应用列出的 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。

在 **Administrator** 视图中，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：

- **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。

1.3.5.7.3. 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：

- **警报状态 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。当条件保持 **true** 时，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。对于与所有列出的值或正则表达式匹配的警报，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。

1.3.5.7.4. 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。

1.3.6. 了解用户定义的项目的警报路由

作为集群管理员，您可以为用户定义的项目启用警报路由。使用此功能，您可以允许具有 **alert-routing-edit** 集群角色的用户为用户定义的项目配置警报通知路由和接收器。这些通知由默认的 Alertmanager 实例路由，如果启用，则为专用于用户定义的监控的可选 Alertmanager 实例。

然后，用户可以通过为用户定义的项目创建或编辑 **AlertmanagerConfig** 对象来创建和配置用户定义的警报路由，而无需管理员的帮助。

用户为用户定义的项目定义了警报路由后，用户定义的警报通知会路由如下：

- 如果使用默认平台 Alertmanager 实例，到 **openshift-monitoring** 命名空间中的 **alertmanager-main** pod。
- 如果您为用户定义的项目启用了单独的一个 Alertmanager 实例，到 **openshift-user-workload-monitoring** 命名空间中的 **alertmanager-user-workload** Pod。



注意

查看用户定义的项目的警报路由的以下限制：

- 对于用户定义的警报规则，用户定义的路由范围到定义资源的命名空间。例如，命名空间 **ns1** 中的路由配置仅适用于同一命名空间中的 **PrometheusRules** 资源。
- 当命名空间不包括在用户定义的监控中时，命名空间中的 **AlertmanagerConfig** 资源将成为 Alertmanager 配置的一部分。

其他资源

- [为用户定义的项目启用警报路由](#)

1.3.7. 将通知发送到外部系统

在 OpenShift Container Platform 4.14 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。

其他资源

- [为核心平台监控配置警报通知](#)
- [为用户工作负载监控配置警报通知](#)

第 2 章 开始使用

2.1. 对监控的维护和支持

并非所有监控堆栈配置选项都公开。配置 OpenShift Container Platform 监控的唯一支持的方法是，使用 Cluster Monitoring Operator (CMO) 的 [Cluster Monitoring Operator 的 Config map 参考](#) 中所述的选项来配置。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。如果您使用 [Cluster Monitoring Operator 的 Config map 引用](#) 中描述的配置，您的更改可能会丢失，因为 Cluster Monitoring Operator 会自动协调任何区别，并将任何不支持的更改重置为最初定义的状态。

2.1.1. 对监控的支持注意事项



注意

指标、记录规则或警报规则的向后兼容性无法被保证。

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 **openshift-monitoring** 命名空间中的 **alertmanager-main** secret 资源部署。如果您为用户定义的警报路由启用了单独的 Alertmanager 实例，则 Alertmanager 配置也会部署为 **openshift-user-workload-monitoring** 命名空间中的 **alertmanager-user-workload** secret 资源。要为 Alertmanager 实例配置额外的路由，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。
- **使用 Prometheus Operator 中的 **Probe** 自定义资源定义（CRD）启用基于症状的监控。**
- **手动将监控资源部署到具有 **openshift.io/cluster-monitoring: "true"** 标签的命名空间中。**
- **将 **openshift.io/cluster-monitoring: "true"** 标签添加到命名空间。** 该标签只适用于带有 OpenShift Container Platform 核心组件和红帽认证的组件的命名空间。
- **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)。

2.1.2. 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。

2.1.3. 监控组件的支持版本列表

以下列表包含有关 OpenShift Container Platform 4.12 及更新的版本的监控组件版本的信息：

表 2.1. OpenShift Container Platform 和组件版本

OpenShift Container Platform	Prometheus Operator	Prometheus	Prometheus Adapter	Alertmanager	kube-state-metrics 代理	monitoring-plugin	node-exporter 代理	Thanos
4.14	0.67.1	2.46.0	0.10.0	0.25.0	2.9.2	1.0.0	1.6.1	0.30.2
4.13	0.63.0	2.42.0	0.10.0	0.25.0	2.8.1	N/A	1.5.0	0.30.2
4.12	0.60.1	2.39.1	0.10.0	0.24.0	2.6.0	N/A	1.4.0	0.28.1



注意

openshift-state-metrics 代理和 Telemeter Client 是特定于 OpenShift 的组件。因此，它们的版本与 OpenShift Container Platform 的版本对应。

2.2. 核心平台监控第一步

安装 OpenShift Container Platform 后，核心平台监控组件会立即开始收集指标，您可以查询和查看。默认 in-cluster 监控堆栈包括核心平台 Prometheus 实例，从集群收集指标以及路由警报的核心 Alertmanager 实例，以及其他组件。根据谁将监控堆栈以及作为集群管理员使用监控堆栈，您可以进一步配置这些监控组件以适应不同场景中的不同用户的需求。

2.2.1. 配置核心平台监控：安装后步骤

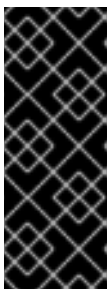
安装 OpenShift Container Platform 后，集群管理员通常会配置核心平台监控以满足您的需要。这些活动包括为 Prometheus、Alertmanager 和其他监控组件设置存储和配置选项。



注意

默认情况下，在新安装的 OpenShift Container Platform 系统中，用户可以查询和查看收集的指标。如果您希望用户接收警报通知，您只需要配置警报接收器。此处列出的任何其他配置选项都是可选的。

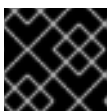
- 如果不存在，[创建 cluster-monitoring-config ConfigMap 对象](#)。
- [配置默认平台警报的通知](#)，以便 Alertmanager 可以将警报发送到外部通知系统，如电子邮件、Slack 或 PagerDuty。
- 对于短期的数据保留期，请为 Prometheus 和 Alertmanager [配置持久性存储](#)以存储指标和警报数据。为 Prometheus 和 Thanos Ruler 指定指标数据保留参数。



重要

- 在多节点集群中，您必须为 Prometheus、Alertmanager 和 Thanos Ruler 配置持久性存储，以确保高可用性。
- 默认情况下，在新安装的 OpenShift Container Platform 系统中，监控 **ClusterOperator** 资源会报告 **PrometheusDataPersistenceNotConfigured** 状态信息，提醒您没有配置存储。

- 对于长期的数据保留，[配置远程写入功能](#)，使 Prometheus 能够将最接近的指标发送到远程系统以进行存储。



重要

务必[将集群 ID 标签添加到指标数据中](#)用于远程写入存储配置的指标。

- 为需要访问特定监控功能的非管理员用户[分配监控集群角色](#)。
- 为监控堆栈组件[分配容限](#)，以便管理员可以将它们移到污点节点上。
- 为指标集合 [设置正文大小限制](#)，以帮助避免在提取目标返回包含大量数据时 Prometheus 消耗大量内存的情况。
- 为集群[修改或创建警报规则](#)。这些规则指定触发警报的条件，如高 CPU 或内存用量、网络延迟等。
- [指定监控组件的资源限值和请求](#)，以确保运行监控组件的容器有足够的 CPU 和内存资源。

当将监控堆栈配置为满足您的需求时，Prometheus 会从指定的服务收集指标，并根据设置存储这些指标。您可以进入 OpenShift Container Platform Web 控制台中的 **Observe** 页面来查看和查询收集的指标、管理警报、识别性能瓶颈以及根据需要扩展资源：

- [查看仪表板](#) 以视觉化收集的指标、对警报进行故障排除并监控集群的附加信息。
- 通过创建 PromQL 查询或使用预定义的查询来查询 [收集的指标](#)。

2.3. 用户工作负载监控第一步

作为集群管理员，除了对核心平台进行监控外，您还可以有选择性地为用户定义的项目启用监控。这样，非管理员用户（如开发人员）可以监控核心平台之外的自己的项目。

集群管理员通常需要完成以下操作来配置用户定义的项目，以使用户可以查看收集的指标、查询这些指标并接收其自己的项目的警报：

- [启用用户工作负载监控](#)。
- 通过分配 **monitoring-rules-view**, **monitoring-rules-edit**, 或 **monitoring-edit** 集群角色，[为非管理员用户授予权限来监控用户定义的项目](#)。
- [分配 user-workload-monitoring-config-edit 角色](#)，为非管理员用户授予配置用户定义的项目的权限。
- [为用户定义的项目启用警报路由](#)，以便开发人员和其他用户能够为其项目配置自定义警报和警报路由。
- 如果需要，为用户定义的项目配置警报路由，[以使用专用于用户定义的项目的可选 Alertmanager 实例](#)。
- [为用户定义的警报配置通知](#)。
- 如果您使用平台 Alertmanager 实例进行用户定义的警报路由，请为默认平台警报和用户定义的警报[配置不同的警报接收器](#)。

2.4. 开发人员和非管理员用户步骤

在启用对用户定义的项目的监控后，开发人员和其他非管理员用户可以执行以下操作来设置并使用自己项目的监控功能：

- [部署和监视服务](#)。
- [创建和管理警报规则](#)。
- [接收和管理项目的警报](#)。
- 如果授予 **alert-routing-edit** 集群角色，请[配置警报路由](#)。
- 使用 OpenShift Container Platform Web 控制台[查看仪表板](#)。
- 通过创建 PromQL 查询或使用预定义的查询来[查询收集的指标](#)。

第 3 章 配置核心平台监控

3.1. 准备配置核心平台监控堆栈

在安装前，OpenShift Container Platform 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。

本节介绍可以配置哪些监控组件以及如何准备配置监控堆栈。



重要

- 并非所有监控堆栈的配置参数都会被公开。只有在 [Cluster Monitoring Operator 的 Config map 引用](#) 中列出的参数和字段才支持进行配置。
- 监控堆栈会带来额外的资源需求。参阅 [扩展 Cluster Monitoring Operator](#) 中的计算资源建议，并确认您有足够的资源。

3.1.1. 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` 配置映射中用来指定这些组件的键。

表 3.1. 可配置的核心平台监控组件

组件	cluster-monitoring-config 配置映射键
Prometheus Operator	<code>prometheusOperator</code>
Prometheus	<code>prometheusK8s</code>
Alertmanager	<code>alertmanagerMain</code>
Thanos querier	<code>thanosQuerier</code>
kube-state-metrics	<code>kubeStateMetrics</code>
monitoring-plugin	<code>monitoringPlugin</code>
openshift-state-metrics	<code>openshiftStateMetrics</code>
Telemeter Client	<code>telemeterClient</code>
Prometheus Adapter	<code>k8sPrometheusAdapter</code>



警告

ConfigMap 对象的不同配置更改会导致不同的结果：

- Pod 不会重新部署。因此，没有服务中断。
- 受影响的 pod 被重新部署：
 - 对于单节点集群，这会导致临时服务中断。
 - 对于多节点集群，因为高可用性，受影响的 pod 会逐渐推出，监控堆栈仍然可用。
 - 无论高可用性如何，配置和重新定义持久性卷的大小始终会导致服务中断。

每个需要更改配置映射中的步骤都包括其预期的结果。

3.1.2. 创建集群监控配置映射

您可以通过在 **openshift-monitoring** 项目中创建和更新 **cluster-monitoring-config** 配置映射来配置 OpenShift Container Platform 核心监控组件。Cluster Monitoring Operator (CMO) 然后配置监控堆栈的核心组件。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 已安装 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
```

3.1.3. 为核心平台监控授予用户权限

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。

您还可以为核心平台监控授予开发人员和其他用户不同的权限。您可以通过分配以下监控角色或集群角色之一来授予权限：

Name	描述	项目
cluster-monitoring-metrics-api	具有此角色的用户能够访问 Thanos Querier API 端点。另外，它还授予对核心平台 Prometheus API 和用户定义的 Thanos Ruler API 端点的访问权限。	openshift-monitoring
cluster-monitoring-operator-alert-customization	具有此角色的用户可以管理用于核心平台监控的 AlertingRule 和 AlertRelabelConfig 资源。这些权限是警报自定义功能所必需的。	openshift-monitoring
monitoring-alertmanager-edit	具有此角色的用户可以管理用于核心平台监控的 Alertmanager API。它们也可以在 OpenShift Container Platform Web 控制台的 Administrator 视角中管理警报静默。	openshift-monitoring
monitoring-alertmanager-view	具有此角色的用户可以监控 Alertmanager API 以进行核心平台监控。它们也可以在 OpenShift Container Platform Web 控制台的 Administrator 视角中查看警报静默。	openshift-monitoring
cluster-monitoring-view	具有此集群角色的用户具有与 cluster-monitoring-metrics-api 角色相同的访问权限，以及其他额外的权限，提供对用户定义的 Prometheus 的 /federate 端点的访问权限。	必须与 ClusterRoleBinding 绑定，才能获取用户定义的 Prometheus 的 /federate 端点的访问权限。

3.1.3.1. 使用 Web 控制台授予用户权限

您可以使用 OpenShift Container Platform Web 控制台为 **openshift-monitoring** 项目或自己的项目授予用户权限。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。

- 要将角色分配到的用户帐户已存在。

流程

1. 在 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **User Management → RoleBindings → 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项目。



重要

您使用此流程向用户授予的监控角色或集群角色权限只适用于您在 **Namespace** 字段中选择的项目。

5. 从 **Role Name** 列表中选择监控角色或集群角色。
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。

3.1.3.2. 使用 CLI 授予用户权限

您可以使用 OpenShift CLI (**oc**) 为 **openshift-monitoring** 项目或自己的项目授予用户权限。



重要

无论您选择的角色或集群角色，都必须作为集群管理员将其与特定项目进行绑定。

先决条件

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

流程

- 要为项目的用户分配监控角色，请输入以下命令：

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

- 1 将 **<role>** 替换为您要监控角色，**<user>** 替换为您要为其分配角色的用户，**<namespace>** 替换为您要授予访问权限的项目。

- 要为项目的用户分配监控集群角色，请输入以下命令：

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

- 1 将 **<cluster-role>** 替换为您要监控的集群角色，**<user>** 替换为您要为其分配集群角色的用户，**<namespace>** 替换为您要授予访问权限的项目。

3.2. 为核心平台监控配置性能和可扩展性

您可以配置监控堆栈来优化集群的性能和扩展。以下文档提供了有关如何分发监控组件的信息，并控制监控堆栈对 CPU 和内存资源的影响。

其他资源

- [关于性能和可扩展性](#)

3.2.1. 控制监控组件的放置和分发

您可以将监控堆栈组件移到特定的节点上：

- 使用带有标记节点的 **nodeSelector** 约束，将任何监控堆栈组件移到特定的节点上。
- 分配容限以启用将组件移到污点节点。

通过这样做，您可以控制集群中监控组件的放置和分发。

通过控制监控组件的放置和分发，您可以根据特定要求或策略优化系统资源使用、提高性能和隔离工作负载。

其他资源

- [使用节点选择器移动监控组件](#)

3.2.1.1. 将监控组件移到其他节点

要指定运行监控堆栈组件的集群中的节点，请在 **cluster-monitoring-config** 配置映射中为组件配置 **nodeSelector** 约束，以匹配分配给节点的标签。



注意

您不能将节点选择器约束直接添加到现有调度的 pod 中。

先决条件

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

流程

1. 如果您还没有这样做，请在要运行监控组件的节点中添加标签：

```
$ oc label nodes <node_name> <node_label> ❶
```

- ❶ 将 **<node_name>** 替换为您要添加标签的节点的名称。将 **<node_label>** 替换为所需标签的名称。

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

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

3. 在 **data/config.yaml** 下为组件指定 **nodeSelector** 约束的节点标签：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    # ...
    <component>: ❶
    nodeSelector:
      <node_label_1> ❷
      <node_label_2> ❸
    # ...
```

- ❶ 将 **<component>** 替换为适当的监控堆栈组件名称。
- ❷ 将 **<node_label_1>** 替换为添加到节点的标签。
- ❸ 可选：指定附加标签。如果您指定了额外的标签，则组件的 pod 仅调度到包含所有指定标签的节点上。



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 事件中与污点和容限相关的错误。

4. 保存文件以使改变生效。新配置中指定的组件会自动移到新节点上，受新配置影响的 pod 会被重新部署。

其他资源

- [准备配置核心平台监控堆栈](#)
- [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](#)
- [nodeSelector \(Kubernetes 文档\)](#)

3.2.1.2. 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

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 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忍。以下示例将 **alertmanagerMain** 组件配置为容忍示例污点：

```
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"
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [准备配置核心平台监控堆栈](#)

- [使用节点污点控制 pod 放置](#)
- [污点和容限\(Kubernetes 文档\)](#)

3.2.2. 为指标提取设置正文大小限制

默认情况下，针对从提取的指标目标返回的数据的未压缩正文大小没有限制。您可以设置正文大小限制，以帮助避免在提取目标返回包含大量数据时 Prometheus 消耗大量内存的情况。另外，通过设置正文大小限制，您可以降低恶意目标在 Prometheus 和整个集群中可能对这个影响。

为 **enforcedBodySizeLimit** 设置了一个值后，当至少有一个 Prometheus scrape 目标回复大于配置的值时，**PrometheusScrapeBodySizeLimitHit** 会触发警报。



注意

如果从目标中提取的指标数据有一个不压缩的正文大小超过配置的大小限制，则提取会失败。然后，Prometheus 会认为这个目标为停机状态，并将其 **up** 指标值设置为 **0**，它将触发一个 **TargetDown** 警报。

先决条件

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

流程

1. 编辑 **openshift-monitoring** 命名空间中的 **cluster-monitoring-config ConfigMap** 对象：

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

2. 将 **enforcedBodySizeLimit** 的值添加到 **data/config.yaml/prometheusK8s** 中，以限制每个目标提取可接受的正文大小：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |-
    prometheusK8s:
      enforcedBodySizeLimit: 40MB 1
```

- 1 指定提取指标目标的最大正文大小。这个 **enforceBodySizeLimit** 示例将每个目标提取的未压缩大小限制为 40MB。有效数字值使用 Prometheus 数据大小格式：B (bytes), KB (kilobytes), MB (megabytes), GB (gigabytes), TB (terabytes), PB (petabytes), and EB (exabytes)。默认值为 **0**，代表没有指定限制。您还可以将值设为 **automatic**，以根据集群容量自动计算限制。

3. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [scrape_config](#) (Prometheus 文档)

3.2.3. 管理监控组件的 CPU 和内存资源

您可以通过为这些组件的资源限值和请求指定值来确保运行监控组件的容器具有足够的 CPU 和内存资源。

您可以为 **openshift-monitoring** 命名空间中的核心平台监控组件配置这些限制和请求。

3.2.3.1. 指定限制和请求

要配置 CPU 和内存资源，请在 **openshift-monitoring** 命名空间中的 **cluster-monitoring-config** **ConfigMap** 对象中指定资源限值和请求值。

先决条件

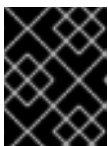
- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您已创建了名为 **cluster-monitoring-config** 的 **ConfigMap** 对象。
- 已安装 OpenShift CLI(**oc**)。

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. 添加值来为您要配置的每个组件定义资源限值和请求。



重要

确保为限制设置的值始终高于为请求设置的值。否则，会出现错误，容器将不会运行。

设置资源限值和请求示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      resources:
        limits:
          cpu: 500m
          memory: 1Gi
        requests:
          cpu: 200m
```

```
    memory: 500Mi
prometheusK8s:
  resources:
    limits:
      cpu: 500m
      memory: 3Gi
    requests:
      cpu: 200m
      memory: 500Mi
thanosQuerier:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
prometheusOperator:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
k8sPrometheusAdapter:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
kubeStateMetrics:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
telemetryClient:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
openshiftStateMetrics:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
```

```

    memory: 500Mi
nodeExporter:
  resources:
    limits:
      cpu: 50m
      memory: 150Mi
    requests:
      cpu: 20m
      memory: 50Mi
monitoringPlugin:
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 500Mi
prometheusOperatorAdmissionWebhook:
  resources:
    limits:
      cpu: 50m
      memory: 100Mi
    requests:
      cpu: 20m
      memory: 50Mi

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [关于指定限制和请求](#)
- [请求和限值\(Kubernetes 文档\)](#)

3.2.4. 选择指标集合配置集



重要

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

有关红帽技术预览功能支持范围的更多信息，请参阅[技术预览功能支持范围](#)。

要为 OpenShift Container Platform 核心监控组件选择指标集合配置集，请编辑 **cluster-monitoring-config ConfigMap** 对象。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 已使用 **FeatureGate** 自定义资源 (CR) 启用了技术预览功能。
- 您已创建 **cluster-monitoring-config ConfigMap** 对象。

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。

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 metrics collection 配置集设置：

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

- 1** 指标集合配置集的名称。可用值为 **full** 或 **minimal**。如果没有指定值，或者配置映射中不存在 **collectionProfile** 键名称，则会使用 **full** 的默认设置。

以下示例将 Prometheus 核心平台实例的指标集合配置集设置为 **minimal**：

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

3. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [关于指标集合配置集](#)
- [查看可用指标列表](#)
- [使用功能门启用功能](#)

3.2.5. 配置 pod 拓扑分布限制

您可以为 Cluster Monitoring Operator 部署的所有 pod 配置 pod 拓扑分布限制，以控制如何在区调度到节点的 pod 副本。这样可确保 pod 具有高可用性并更有效地运行，因为工作负载分散在不同的数据中心或分层基础架构区域中。

您可以使用 **cluster-monitoring-config** 配置映射为监控 pod 配置 pod 拓扑分布限制。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 在 **data/config.yaml** 字段中添加以下设置来配置 pod 拓扑分布限制：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>: ❶
    topologySpreadConstraints:
      - maxSkew: <n> ❷
        topologyKey: <key> ❸
        whenUnsatisfiable: <value> ❹
        labelSelector: ❺
          <match_option>
```

- ❶ 指定您要为其设置 pod 拓扑分布限制的组件名称。
- ❷ 为 **maxSkew** 指定数字值，它定义了允许不均匀分布 pod 的程度。
- ❸ 为 **topologyKey** 指定节点标签键。带有具有此键和相同值标签的节点被视为在同一拓扑域中。调度程序会尝试将大量 pod 放置到每个域中。
- ❹ 为 **whenUnsatisfiable** 指定一个值。可用选项包括 **DoNotSchedule** 和 **ScheduleAnyway**。如果您希望 **maxSkew** 值定义目标拓扑和全局最小值中匹配 pod 数量之间允许的最大值，则指定 **DoNotSchedule**。如果您希望调度程序仍然调度 pod，但为可能降低 skew 的节点赋予更高的优先级，请指定 **ScheduleAnyway**。
- ❺ 指定 **labelSelector** 来查找匹配的 pod。与此标签选择器匹配的 Pod 被计算，以确定其对应拓扑域中的 pod 数量。

Prometheus 配置示例

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

```
data:
  config.yaml: |
    prometheusK8s:
      topologySpreadConstraints:
        - maxSkew: 1
          topologyKey: monitoring
          whenUnsatisfiable: DoNotSchedule
      labelSelector:
        matchLabels:
          app.kubernetes.io/name: prometheus
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [关于用于监控的 pod 拓扑分布限制](#)
- [使用 pod 拓扑分布限制控制 pod 放置](#)
- [Pod 拓扑分布限制\(Kubernetes 文档\)](#)

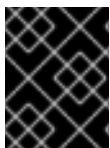
3.3. 为核心平台监控存储和记录数据

存储和记录您的指标和警报数据，配置日志以指定记录哪些活动，控制 Prometheus 保留数据的时长，并为数据设置最大磁盘空间量。这些操作可帮助您保护数据并将其用于故障排除。

3.3.1. 配置持久性存储

使用持久性存储运行集群监控以获取以下优点：

- 通过将指标和警报数据存储持久性卷(PV)中来保护您的指标和警报数据。因此，它们可以在 pod 重启或重新创建后保留。
- 避免获取重复的通知，并在 Alertmanager pod 重启时丢失警报静默。



重要

在多节点集群中，您必须为 Prometheus、Alertmanager 和 Thanos Ruler 配置持久性存储，以确保高可用性。



注意

在生产环境中，强烈建议配置持久性存储。

3.3.1.1. 持久性存储的先决条件

- 分配充足的专用持久性存储，以确保磁盘不会被填满。
- 在配置持久性卷时，使用 **Filesystem** 作为 **volumeMode** 参数的存储类型值。



重要

- 不要使用原始块卷，它由 **PersistentVolume** 资源中的 **volumeMode: Block** 描述。Prometheus 无法使用原始块卷。
- Prometheus 不支持兼容 POSIX 的文件系统。例如，一些 NFS 文件系统实现不兼容 POSIX。如果要使用 NFS 文件系统进行存储，请验证与其 NFS 实现完全兼容 POSIX 的供应商。

3.3.1.2. 配置持久性卷声明

要将持久性卷 (PV) 用于监控组件，您必须配置持久性卷声明 (PVC)。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 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>: ❶
    volumeClaimTemplate:
      spec:
        storageClassName: <storage_class> ❷
        resources:
          requests:
            storage: <amount_of_storage> ❸
```

- ❶ 指定要配置 PVC 的监控组件。
- ❷ 指定现有存储类。如果没有指定存储类，则使用默认存储类。
- ❸ 指定所需的存储量。

以下示例配置了一个 PVC 来声明 Prometheus 的持久性存储：

PVC 配置示例

■


```

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

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署并应用新的存储配置。



警告

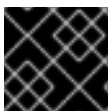
当您使用 PVC 配置更新配置映射时，受影响的 **StatefulSet** 对象会被重新创建，从而导致临时服务中断。

其他资源

- [了解持久性存储](#)
- [PersistentVolumeClaim \(Kubernetes 文档\)](#)

3.3.1.3. 重新定义持久性卷大小

您可以为监控组件（如 Prometheus 或 Alertmanager）重新定义持久性卷(PV)大小。您需要手动扩展持久性卷声明 (PVC)，然后更新配置组件的配置映射。



重要

您只能扩展 PVC 的大小。无法缩小存储大小。

先决条件

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

流程

1. 使用更新的存储请求手动扩展 PVC。如需更多信息，请参阅 *扩展持久性卷* 中的 "使用文件系统扩展持久性卷声明 (PVC)"。
2. 编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config** 配置映射：

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

3. 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>: ❶
    volumeClaimTemplate:
      spec:
        resources:
          requests:
            storage: <amount_of_storage> ❷
```

- ❶ 要更改存储大小的组件。
- ❷ 指定存储卷的新大小。它必须大于上一个值。

以下示例将 Prometheus 实例的新 PVC 请求设置为 100GB：

prometheusK8s 的存储配置示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      volumeClaimTemplate:
        spec:
          resources:
            requests:
              storage: 100Gi
```

4. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。



警告

当您使用新存储大小更新配置映射时，受影响的 **StatefulSet** 对象会被重新创建，从而导致临时服务中断。

其他资源

- [Prometheus 数据库存储要求](#)
- [使用文件系统扩展持久性卷声明 \(PVC\)](#)

3.3.2. 修改 Prometheus 指标数据的保留时间和大小

默认情况下，Prometheus 为核心平台监控保留 15 天指标数据。您可以修改 Prometheus 实例的保留时间，以便在删除数据时更改。您还可以设置保留指标数据使用的最大磁盘空间量。



注意

数据压缩每两小时进行一次。因此，持久性卷 (PV) 可能会在压缩前已被填满，可能会超过 **retentionSize** 限制。在这种情况下，**KubePersistentVolumeFillingUp** 警报会触发，直到 PV 上的空间低于 **retentionSize** 限制。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 在 **data/config.yaml** 下添加保留时间和大小配置：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: <time_specification> 1
      retentionSize: <size_specification> 2
```

- ❶ 保留时间：数字直接加上 **ms**（毫秒）、**s**（秒）、**m**（分钟）、**h**（小时）、**d**（天）、**w**（周）或 **y**（年）。您还可以组合指定时间值，如 **1h30m15s**。
- ❷ 保留大小：数字直接加上 **B** (bytes), **KB** (kilobytes), **MB** (megabytes), **GB** (gigabytes), **TB** (terabytes), **PB** (petabytes), 和 **EB** (exabytes)。

以下示例将 Prometheus 实例的保留时间设置为 24 小时，保留大小设为 10GB：

为 Prometheus 设置保留时间示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: 24h
      retentionSize: 10GB
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [Prometheus 指标的保留时间和大小](#)
- [准备配置核心平台监控堆栈](#)
- [Prometheus 数据库存储要求](#)
- [推荐的可配置存储技术](#)
- [了解持久性存储](#)
- [优化存储](#)

3.3.3. 配置专用服务监控器

您可以配置 OpenShift Container Platform 核心平台监控，以使用专用服务监控器来收集资源指标管道的指标。

启用后，专用服务监控器会从 kubelet 端点公开两个额外的指标，并将 **honorTimestamps** 字段的值设置为 true。

通过启用专用服务监控器，您可以提高由 **oc adm top pod** 命令或 Horizontal Pod Autoscaler 使用的基于 Prometheus Adapter 的 CPU 用量测量的一致性。

3.3.3.1. 启用专用服务监控器

您可以通过在 **openshift-monitoring** 命名空间中的 **cluster-monitoring-config ConfigMap** 对象中配置 **dedicatedServiceMonitors** 键，将核心平台监控配置为使用专用服务监控器。

前提条件

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

流程

1. 编辑 **openshift-monitoring** 命名空间中的 **cluster-monitoring-config ConfigMap** 对象：

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

2. 如以下示例所示，添加一个 **enabled: true** 键-值对：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    k8sPrometheusAdapter:
      dedicatedServiceMonitors:
        enabled: true ❶
```

- ❶ 将 **enabled** 字段的值设置为 **true** 以部署一个专用服务监控器，该监控器公开 kubelet **/metrics/resource** 端点。

3. 保存文件以自动应用更改。



警告

当您保存对 **cluster-monitoring-config** 配置映射的更改时，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可能会重启。

3.3.4. 为 Prometheus Adapter 设置审计日志级别

在默认平台监控中，您可以为 Prometheus Adapter 配置审计日志级别。

先决条件

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

流程

您可以在默认 **openshift-monitoring** 项目中为 Prometheus Adapter 设置审计日志级别：

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** 下的 **k8sPrometheusAdapter/audit** 部分中添加 **profile**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    k8sPrometheusAdapter:
      audit:
        profile: <audit_log_level> 1
```

- 1 应用到 Prometheus Adapter 的审计日志级别。

3. 使用 **profile** 参数的以下值之一设置审计日志级别：

- **None**：不记录事件。
- **Metadata**：仅记录请求的元数据，如用户、时间戳等。不要记录请求文本和响应文本。**metadata** 是默认的审计日志级别。
- **Request**：仅记录元数据和请求文本，而不记录响应文本。这个选项不适用于非资源请求。
- **RequestResponse**：日志事件元数据、请求文本和响应文本。这个选项不适用于非资源请求。

4. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

验证

1. 在配置映射的 **k8sPrometheusAdapter/audit/profile** 下，将日志级别设置为 **Request** 并保存文件。
2. 确认 Prometheus Adapter 的 pod 正在运行。以下示例列出了 **openshift-monitoring** 项目中的 pod 状态：

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

3. 确认正确配置了审计日志级别和审计日志文件路径：

```
$ oc -n openshift-monitoring get deploy prometheus-adapter -o yaml
```

输出示例

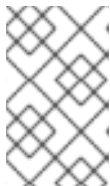
```
...
- --audit-policy-file=/etc/audit/request-profile.yaml
- --audit-log-path=/var/log/adapter/audit.log
```

4. 确认 **openshift-monitoring** 项目中的 **prometheus-adapter** 部署中应用了正确的日志级别：

```
$ oc -n openshift-monitoring exec deploy/prometheus-adapter -c prometheus-adapter -- cat
/etc/audit/request-profile.yaml
```

输出示例

```
"apiVersion": "audit.k8s.io/v1"
"kind": "Policy"
"metadata":
  "name": "Request"
"omitStages":
- "RequestReceived"
"rules":
- "level": "Request"
```



注意

如果您为 **ConfigMap** 对象中的 Prometheus Adapter 输入了一个未识别的 **profile** 值，则不会对 Prometheus Adapter 进行任何更改，Cluster Monitoring Operator 会记录错误。

5. 查看 Prometheus Adapter 的审计日志：

```
$ oc -n openshift-monitoring exec -c <prometheus_adapter_pod_name> -- cat
/var/log/adapter/audit.log
```

3.3.5. 为监控组件设置日志级别

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

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

- **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

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 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>: ❶
    logLevel: <log_level> ❷
```

- ❶ 您要为其设置日志级别的监控堆栈组件。可用的组件值包括 **prometheusK8s**、**alertmanagerMain**、**prometheusOperator** 和 **thanosQuerier**。
- ❷ 为组件设置的日志级别。可用值为 **error**、**warn**、**info** 和 **debug**。默认值为 **info**。

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。
4. 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 **prometheus-operator** 部署的日志级别：

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

输出示例

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

5. 检查组件的 Pod 是否正在运行。以下示例列出了 pod 的状态：

```
$ oc -n openshift-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。

3.3.6. 为 Prometheus 启用查询日志文件

您可以将 Prometheus 配置为将引擎运行的所有查询写入到日志文件。



重要

由于不支持日志轮转，因此仅在需要对问题进行故障排除时才临时启用此功能。完成故障排除后，通过恢复您对 **ConfigMap** 对象所做的更改来禁用查询日志记录，以启用该功能。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 将 Prometheus 的 **queryLogFile** 参数添加到 **data/config.yaml** 下：

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

- ❶ 添加要记录查询的文件的完整路径。

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。
4. 验证组件的 pod 是否正在运行。以下示例命令列出了 pod 的状态：

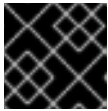
```
$ oc -n openshift-monitoring get pods
```

输出示例

```
...
prometheus-operator-567c9bc75c-96wkj 2/2 Running 0 62m
prometheus-k8s-0 6/6 Running 1 57m
prometheus-k8s-1 6/6 Running 1 57m
thanos-querier-56c76d7df4-2xkpc 6/6 Running 0 57m
thanos-querier-56c76d7df4-j5p29 6/6 Running 0 57m
...
```

5. 读取查询日志：

```
$ oc -n openshift-monitoring exec prometheus-k8s-0 -- cat <path>
```

**重要**

在检查了记录的查询信息后，恢复配置映射的设置。

其他资源

- [准备配置核心平台监控堆栈](#)

3.3.7. 为 Thanos Querier 启用查询日志记录

对于 **openshift-monitoring** 项目中的默认平台监控，您可以启用 Cluster Monitoring Operator 来记录 Thanos Querier 运行的所有查询。

**重要**

由于不支持日志轮转，因此仅在需要对问题进行故障排除时才临时启用此功能。完成故障排除后，通过恢复您对 **ConfigMap** 对象所做的更改来禁用查询日志记录，以启用该功能。

先决条件

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

流程

您可以在 **openshift-monitoring** 项目中为 Thanos Querier 启用查询日志记录：

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** 中添加 **thanosQuerier** 部分并添加值，如下例所示：

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

- 1 将值设为 **true** 以启用日志记录，并将 **false** 设置为 disable logging。默认值为 **false**。
- 2 将值设为 **debug**、**info**、**warn** 或 **error**。如果 **logLevel** 没有值，则日志级别默认为 **error**。

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

验证

1. 验证 Thanos Querier pod 是否正在运行。以下示例命令列出了 **openshift-monitoring** 项目中的 pod 状态：

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

2. 使用以下示例命令作为模型运行测试查询：

```
$ token=`oc create token prometheus-k8s -n openshift-monitoring`
```

```
$ oc -n openshift-monitoring exec -c prometheus prometheus-k8s-0 -- curl -k -H
"Authorization: Bearer $token" 'https://thanos-querier.openshift-
monitoring.svc:9091/api/v1/query?query=cluster_version'
```

3. 运行以下命令来读取查询日志：

```
$ oc -n openshift-monitoring logs <thanos_querier_pod_name> -c thanos-query
```



注意

因为 **thanos-querier** pod 是高度可用的(HA)pod，所以您可能只能看到一个 pod 的日志。

4. 检查日志记录的查询信息后，通过将配置映射中的 **enableRequestLogging** 值更改为 **false** 来禁用查询日志记录。

3.4. 为核心平台监控配置指标

配置指标集合，以监控集群组件和您自己的工作负载的执行方式。

您可以将最接近的指标发送到远程系统进行长期存储，并将集群 ID 标签添加到指标，以识别来自不同集群的数据。

其他资源

- [了解指标](#)

3.4.1. 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

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您已创建 **cluster-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](#)。



重要

红帽只提供配置远程写入发送者的信息，而不提供有关配置接收器端点的指导。客户负责设置自己的端点，这些端点与远程写入兼容。端点接收器配置的问题不包括在红帽产品支持中。

- 您已为远程写入端点在 **Secret** 对象中设置身份验证凭证。您必须在 **openshift-monitoring** 命名空间中创建 secret。



警告

要减少安全风险，请使用 HTTPS 和身份验证向端点发送指标。

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/prometheusK8s** 下添加一个 **remoteWrite:** 部分，如下例所示：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://remote-write-endpoint.example.com" 1
          <endpoint_authentication_credentials> 2
```

1

远程写入端点的 URL。

2

端点的身份验证方法和凭据。目前支持的身份验证方法有 AWS 签名版本 4，使用 HTTP **Authorization** 请求标头、基本身份验证、OAuth 2.0 和 TLS 客户端进行身份验证。有关支持的身份验证方法示例配置，请参阅 [支持的远程写入身份验证设置](#)。

3. 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.example.com"
          <endpoint_authentication_credentials>
      writeRelabelConfigs:
        - <your_write_relabel_configs> ❶

```

❶ 为您要发送到远程端点的指标添加配置。

转发一个名为 **my_metric** 的单个指标的示例

```

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

```

在 **my_namespace** 命名空间中转发名为 **my_metric_1** 和 **my_metric_2** 的指标示例

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://remote-write-endpoint.example.com"
      writeRelabelConfigs:
        - sourceLabels: [__name__, namespace]
          regex: '(my_metric_1|my_metric_2);my_namespace'
          action: keep

```

4. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [writeRelabelConfigs](#)
- [relabel_config](#) (Prometheus 文档)

3.4.1.1. 支持的远程写入身份验证设置

您可以使用不同的方法通过远程写入端点进行身份验证。目前支持的身份验证方法有 AWS 签名版本 4、基本身份验证、授权、OAuth 2.0 和 TLS 客户端。下表提供有关用于远程写入的受支持身份验证方法的详情。

身份验证方法	配置映射字段	描述
AWS 签名版本 4	sigv4	此方法使用 AWS 签名版本 4 身份验证为请求签名。您不能搭配授权、OAuth 2.0 或基本身份验证同时使用此方法。
基本身份验证 (Basic authentication)	basicAuth	基本身份验证使用配置的用户名和密码在每个远程写入请求上设置授权标头。
授权	授权	授权使用配置的令牌在每个远程写入请求上设置 Authorization 标头。
OAuth 2.0	oauth2	OAuth 2.0 配置使用客户端凭据授予类型。Prometheus 使用指定的客户端 ID 和客户端 secret 从 tokenUrl 获取访问令牌来访问远程写入端点。您不能与授权、AWS 签名版本 4 或基本身份验证同时使用此方法。
TLS 客户端	tlsConfig	TLS 客户端配置指定 CA 证书、客户端证书和客户端密钥文件信息，用于使用 TLS 与远程写入端点服务器进行身份验证。示例配置假定您已创建了 CA 证书文件、客户端证书文件和客户端密钥文件。

3.4.1.2. 远程写入身份验证设置示例

以下示例展示了可用于连接到远程写入端点的不同身份验证设置。每个示例还演示了如何配置包含身份验证凭据和其他相关设置的对应 **Secret** 对象。每个示例配置身份验证，以用于 **openshift-monitoring** 命名空间中的默认平台监控。

3.4.1.2.1. AWS 签名版本 4 验证的 YAML 示例

以下显示了 **openshift-monitoring** 命名空间中名为 **sigv4-credentials** 的 **sigv4** secret 的设置。

```
apiVersion: v1
kind: Secret
metadata:
  name: sigv4-credentials
  namespace: openshift-monitoring
stringData:
```

```
accessKey: <AWS_access_key> ❶
secretKey: <AWS_secret_key> ❷
type: Opaque
```

- ❶ AWS API 访问密钥。
- ❷ AWS API secret 密钥。

下面显示了一个 AWS Signature Version 4 远程写入身份验证设置示例，它使用一个在 **openshift-monitoring** 命名空间中的名为 **sigv4-credentials** 的 **Secret** 对象：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://authorization.example.com/api/write"
          sigv4:
            region: <AWS_region> ❶
            accessKey:
              name: sigv4-credentials ❷
              key: accessKey ❸
            secretKey:
              name: sigv4-credentials ❹
              key: secretKey ❺
            profile: <AWS_profile_name> ❻
            roleArn: <AWS_role_arn> ❼
```

- ❶ AWS 区域。
- ❷❹ 包含 AWS API 访问凭证的 **Secret** 对象的名称。
- ❸ 在指定 **Secret** 对象中包含 AWS API 访问密钥的密钥。
- ❺ 在指定的 **Secret** 对象中包含 AWS API secret 键的密钥。
- ❻ 用于验证的 AWS 配置集的名称。
- ❼ 分配给角色的 Amazon 资源名称(ARN)的唯一标识符。

3.4.1.2.2. 用于基本身份验证的 YAML 示例

以下显示了 **openshift-monitoring** 命名空间中名为 **rw-basic-auth** 的 **Secret** 对象基本身份验证设置示例：

```
apiVersion: v1
kind: Secret
metadata:
```

```

name: rw-basic-auth
namespace: openshift-monitoring
stringData:
  user: <basic_username> ❶
  password: <basic_password> ❷
type: Opaque

```

❶ 用户名。

❷ 密码。

以下示例显示了使用 **openshift-monitoring** 命名空间中名为 **rw-basic-auth** 的 **Secret** 对象的 **basicAuth** 远程写入配置。它假设您已为端点设置了身份验证凭据。

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://basicauth.example.com/api/write"
        basicAuth:
          username:
            name: rw-basic-auth ❶
            key: user ❷
          password:
            name: rw-basic-auth ❸
            key: password ❹

```

❶ ❸ 包含身份验证凭据的 **Secret** 对象的名称。

❷ 在指定的 **Secret** 对象中包含用户名的密钥。

❹ 在指定 **Secret** 对象中包含密码的密钥。

3.4.1.2.3. 使用 **Secret** 对象通过 **bearer** 令牌进行身份验证的 YAML 示例

以下显示了 **openshift-monitoring** 命名空间中名为 **rw-bearer-auth** 的 **Secret** 对象的 **bearer** 令牌设置：

```

apiVersion: v1
kind: Secret
metadata:
  name: rw-bearer-auth
  namespace: openshift-monitoring
stringData:
  token: <authentication_token> ❶
type: Opaque

```

❶ 身份验证令牌。

以下显示了在 **openshift-monitoring** 命名空间中使用名为 **rw-bearer-auth** 的 **Secret** 对象的 bearer 令牌配置映射设置示例：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://authorization.example.com/api/write"
          authorization:
            type: Bearer ❶
            credentials:
              name: rw-bearer-auth ❷
              key: token ❸
```

- ❶ 请求的验证类型。默认值为 **Bearer**。
- ❷ 包含身份验证凭据的 **Secret** 对象的名称。
- ❸ 在指定的 **Secret** 对象中包含身份验证令牌的密钥。

3.4.1.2.4. 用于 OAuth 2.0 验证的 YAML 示例

以下显示了 **openshift-monitoring** 命名空间中名为 **oauth2-credentials** 的 **Secret** 对象的 OAuth 2.0 设置示例：

```
apiVersion: v1
kind: Secret
metadata:
  name: oauth2-credentials
  namespace: openshift-monitoring
stringData:
  id: <oauth2_id> ❶
  secret: <oauth2_secret> ❷
type: Opaque
```

- ❶ OAuth 2.0 ID。
- ❷ OAuth 2.0 secret。

下面显示了一个 **oauth2** 远程写入身份验证示例配置，它使用 **openshift-monitoring** 命名空间中名为 **oauth2-credentials** 的 **Secret** 对象：

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

```

data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://test.example.com/api/write"
      oauth2:
        clientId:
          secret:
            name: oauth2-credentials ❶
            key: id ❷
        clientSecret:
          name: oauth2-credentials ❸
          key: secret ❹
        tokenUrl: https://example.com/oauth2/token ❺
        scopes: ❻
        - <scope_1>
        - <scope_2>
        endpointParams: ❼
          param1: <parameter_1>
          param2: <parameter_2>

```

❶ ❸ 对应的 **Secret** 对象的名称。请注意，**ClientId** 可以引用 **ConfigMap** 对象，但 **clientSecret** 必须引用 **Secret** 对象。

❷ ❹ 在指定 **Secret** 对象中包含 OAuth 2.0 凭证的密钥。

❺ 用于通过指定的 **clientId** 和 **clientSecret** 获取令牌的 URL。

❻ 授权请求的 OAuth 2.0 范围。这些范围限制了令牌可以访问的数据。

❼ 授权服务器所需的 OAuth 2.0 授权请求参数。

3.4.1.2.5. TLS 客户端身份验证的 YAML 示例

以下显示了 **openshift-monitoring** 命名空间中名为 **mtls-bundle** 的 **tls Secret** 对象的 TLS 客户端设置示例。

```

apiVersion: v1
kind: Secret
metadata:
  name: mtls-bundle
  namespace: openshift-monitoring
data:
  ca.crt: <ca_cert> ❶
  client.crt: <client_cert> ❷
  client.key: <client_key> ❸
type: tls

```

❶ Prometheus 容器中用于验证服务器证书的 CA 证书。

❷ 用于与服务器进行身份验证的客户端证书。

❸ 客户端密钥。

以下示例显示了使用名为 **mtls-bundle** 的 TLS **Secret** 对象的 **tlsConfig** 远程写入身份验证配置。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://remote-write-endpoint.example.com"
          tlsConfig:
            ca:
              secret:
                name: mtl-bundle ❶
                key: ca.crt ❷
            cert:
              secret:
                name: mtl-bundle ❸
                key: client.crt ❹
            keySecret:
              name: mtl-bundle ❺
              key: client.key ❻
```

❶ ❸ ❺ 包含 TLS 身份验证凭证的对应 **Secret** 对象的名称。请注意，**ca** 和 **cert** 可以引用 **ConfigMap** 对象，但 **keySecret** 必须引用 **Secret** 对象。

❷ 指定 **Secret** 对象中的键，其中包含端点的 CA 证书。

❹ 指定 **Secret** 对象中的键，其中包含端点的客户端证书。

❻ 包含客户端密钥 secret 的指定 **Secret** 对象中的密钥。

3.4.1.3. 远程写入队列配置示例

您可以使用 **queueConfig** 对象来远程写入，以调优远程写入队列参数。以下示例显示了 **queue** 参数，以及它们在 **openshift-monitoring** 命名空间中默认平台监控的默认值。

使用默认值进行远程写入参数配置示例

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

```

capacity: 10000 ❶
minShards: 1 ❷
maxShards: 50 ❸
maxSamplesPerSend: 2000 ❹
batchSendDeadline: 5s ❺
minBackoff: 30ms ❻
maxBackoff: 5s ❼
retryOnRateLimit: false ❽

```

- ❶ 在从队列中丢弃前，每个分片要缓冲的样本数量。
- ❷ 最小分片数量。
- ❸ 最大分片数量。
- ❹ 每个发送的最大样本数。
- ❺ 样本在缓冲区中等待的最长时间。
- ❻ 在重试一个失败的请求前等待的初始时间。每次重试的时间都会加倍，最长为 **maxbackoff** 时间。
- ❼ 在重试一个失败的请求前等待的最长时间。
- ❽ 将此参数设为 **true**，以在从远程写入存储接收 429 状态代码后重试请求。

其他资源

- [用于远程写入的 Prometheus REST API 参考](#)
- [远程写入兼容端点\(Prometheus 文档\)](#)
- [远程写入调整\(Prometheus 文档\)](#)
- [了解 secret](#)

3.4.1.4. 远程写入指标表

下表包含远程写入和远程 write-adjacent 指标，其中包含进一步的描述，以帮助在远程写入配置过程中解决问题。

指标	描述
prometheus_remote_storage_highest_timestamp_in_seconds	显示 Prometheus 存储在任一样本的 write-ahead 日志 (WAL) 的最新时间戳。
prometheus_remote_storage_queue_highest_sent_timestamp_seconds	显示远程写入队列成功发送的最新时间戳。
prometheus_remote_storage_samples_retried_total	远程写入无法发送的示例数量，必须重新发送到远程存储。如果此指标一直稳定在高速率，则表示网络或远程存储端点的问题。

指标	描述
<code>prometheus_remote_storage_shards</code>	显示当前为每个远程端点运行多少个分片。
<code>prometheus_remote_storage_shards_desired</code>	根据当前的写入吞吐量和传入与发送样本的速率显示计算的分片数量。
<code>prometheus_remote_storage_shards_max</code>	显示基于当前配置的分片的最大数量。
<code>prometheus_remote_storage_shards_min</code>	显示基于当前配置的分片的最小数量。
<code>prometheus_tsdb_wal_segment_current</code>	Prometheus 当前正在写入新数据的 WAL 段文件。
<code>prometheus_wal_watcher_current_segment</code>	每个远程写入实例当前读取的 WAL 段文件。

3.4.2. 为指标创建集群 ID 标签

您可以通过在 `openshift-monitoring` 命名空间中为 `cluster-monitoring-config` 配置映射中添加远程写入存储的 `write_relabel` 设置来为指标创建集群 ID 标签。

先决条件

- 您可以使用具有 `cluster-admin` 集群角色的用户身份访问集群。
- 您已创建 `cluster-monitoring-config ConfigMap` 对象。
- 已安装 OpenShift CLI(`oc`)。
- 您已配置了远程写入存储。

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/prometheusK8s/remoteWrite` 中的 `writeRelabelConfigs:` 部分添加集群 ID 重新标记配置值：

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

```
<endpoint_authentication_credentials>
writeRelabelConfigs: ❶
- <relabel_config> ❷
```

- ❶ 为您要发送到远程端点的指标添加写入重新标记配置列表。
- ❷ 替换发送到远程写入端点的指标的标签配置。

以下示例演示了如何使用集群 ID 标签 **cluster_id** 转发指标：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "https://remote-write-endpoint.example.com"
          writeRelabelConfigs:
            - sourceLabels:
                - __tmp_openshift_cluster_id__ ❶
              targetLabel: cluster_id ❷
              action: replace ❸
```

- ❶ 系统最初应用名为 **__tmp_openshift_cluster_id__** 的临时集群 ID 源标签。此临时标签由您指定的集群 ID 标签名称替换。
- ❷ 指定发送到远程写入存储的指标的集群 ID 标签名称。如果您使用指标已存在的标签名称，则该值会使用这个集群 ID 标签的名称覆盖。对于标签名称，不要使用 **__tmp_openshift_cluster_id__**。最后重新标记步骤会删除使用此名称的标签。
- ❸ **replace** write relabel 操作，将临时标签替换为传出指标的目标标签。这个操作是默认行为，如果没有指定任何操作，则会被应用。

3. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [在指标中添加集群 ID 标签](#)
- [获取集群 ID](#)

3.5. 为核心平台监控配置警报和通知

您可以配置本地或远程 Alertmanager 实例，将警报从 Prometheus 路由到端点接收器。您还可以将自定义标签附加到所有时间序列和警报，以添加有用的元数据信息。

3.5.1. 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。

您可以添加外部 Alertmanager 实例来路由 OpenShift Container Platform 核心项目的警报。

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

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 使用 **data/config.yaml/prometheusK8s** 下的配置详情添加一个 **additionalAlertmanagerConfigs** 部分：

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

- 1** 将 **<alertmanager_specification>** 替换为额外的 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客户端 TLS 身份验证的 bearer 令牌为 Prometheus 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

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

3.5.1.1. 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** 集群角色的用户身份访问集群。
- 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

- [Alertmanager \(Prometheus 文档\)](#)
- [以管理员身份管理警报](#)

3.5.2. 为 Alertmanager 配置 secret

OpenShift Container Platform 监控堆栈包括 Alertmanager，它将警报从 Prometheus 路由到端点接收器。如果您需要通过接收器进行身份验证以便 Alertmanager 能够向它发送警报，您可以将 Alertmanager 配置为使用包含接收器身份验证凭据的 secret。

例如，您可以将 Alertmanager 配置为使用 secret 与需要由私有证书颁发机构 (CA) 发布的证书的端点接收器进行身份验证。您还可以将 Alertmanager 配置为使用 secret 与需要用于基本 HTTP 身份验证密码文件的接收器进行身份验证。在这两种情况下，身份验证详情都包含在 **Secret** 对象中，而不是包含在 **ConfigMap** 对象中。

3.5.2.1. 在 Alertmanager 配置中添加 secret

您可以通过编辑 **openshift-monitoring** 项目中的 **cluster-monitoring-config** 配置映射，将 secret 添加到 Alertmanager 配置中。

将 secret 添加到配置映射后，secret 作为一个卷挂载到 Alertmanager Pod 的 **alertmanager** 容器中的 **/etc/alertmanager/secrets/<secret_name>** 的卷中。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您已创建了 **cluster-monitoring-config** 配置映射。
- 您已创建了要在 **openshift-monitoring** 项目中的 Alertmanager 中配置的 secret。
- 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/alertmanagerMain** 下添加一个 **secrets:** 部分，并具有以下配置：

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

```
secrets: ❶
- <secret_name_1> ❷
- <secret_name_2>
```

- ❶ 本节包含要挂载到 Alertmanager 中的 secret。secret 必须位于与 Alertmanager 对象相同的命名空间中。
- ❷ 包含接收器身份验证凭证的 **Secret** 对象的名称。如果您添加多个 secret，请将每个 secret 放在新行中。

以下示例配置映射设置将 Alertmanager 配置为使用名为 **test-secret-basic-auth** 和 **test-secret-api-token** 的两个 **Secret** 对象：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      secrets:
        - test-secret-basic-auth
        - test-secret-api-token
```

3. 保存文件以使改变生效。新的配置会被自动应用。

3.5.3. 在时间序列和警报中附加额外标签

您可以使用 Prometheus 的外部标签功能，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

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 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** 配置映射：

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

2. 在 **data/config.yaml** 下定义您要为每个指标添加的标签：

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

```
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        <key>: <value> ❶
```

- ❶ 将 **<key>: <value>** 替换为键值对，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。
- 不要使用 **cluster** 作为密钥名称。使用它可能会导致开发人员仪表板无法看到数据的问题。

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，请使用以下示例：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        region: eu
        environment: prod
```

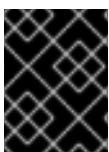
- 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [准备配置核心平台监控堆栈](#)

3.5.4. 配置警报通知

在 OpenShift Container Platform 4.14 中，您可以在 Alerting UI 中查看触发警报。您可以通过配置警报接收器，将 Alertmanager 配置为发送有关默认平台警报的通知。



重要

Alertmanager 默认不会发送通知。强烈建议您通过 web 控制台或通过 **alertmanager-main** secret 配置警报接收器来接收通知。

其他资源

- [将通知发送到外部系统](#)
- [PagerDuty 网站](#)
- [Prometheus Integration Guide \(PagerDuty 文档\)](#)
- [监控组件的支持版本列表](#)
- [为用户定义的项目启用警报路由](#)

3.5.4.1. 为默认平台警报配置警报路由

您可以将 Alertmanager 配置为发送通知，以接收来自集群的重要警报。通过编辑 **openshift-monitoring** 命名空间中的 **alertmanager-main** secret 中的默认配置来自定义 Alertmanager 如何发送通知有关默认平台警报的通知。



注意

OpenShift Container Platform Alertmanager 配置中也支持支持上游 Alertmanager 的所有功能。要检查受支持的上游 Alertmanager 版本的所有配置选项，请参阅 [Alertmanager 配置](#) (Prometheus 文档)。

先决条件

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

流程

1. 从 **alertmanager-main** secret 中提取当前活跃的 Alertmanager 配置，并将其保存为本地 **alertmanager.yaml** 文件：

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

2. 打开 **alertmanager.yaml** 文件。
3. 编辑 Alertmanager 配置：
 - a. 可选：更改默认的 Alertmanager 配置：

默认 Alertmanager secret YAML 示例

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s ①
  group_interval: 5m ②
  repeat_interval: 12h ③
  receiver: default
routes:
- matchers:
  - "alertname=Watchdog"
```

```

    repeat_interval: 2m
    receiver: watchdog
receivers:
- name: default
- name: watchdog

```

- 1 指定在为了一组警报收集初始警报时 Alertmanager 在发送通知时等待的时间。
- 2 指定 Alertmanager 向已发送初始通知的一组警报发送有关新警报的通知前必须经过的时间。
- 3 指定在通知重复前必须经过的最短时间。如果您希望通知在每个组间隔重复，请将 **repeat_interval** 值设置为小于 **group_interval** 的值。重复的通知仍然可以延迟，例如当某些 Alertmanager pod 重启或重新调度时。

b. 添加警报接收器配置：

```

# ...
receivers:
- name: default
- name: watchdog
- name: <receiver> 1
  <receiver_configuration> 2
# ...

```

- 1 接收器的名称。
- 2 接收器配置。支持的接收器有 PagerDuty、Webhook、电子邮件、Slack 和 Microsoft 团队。

将 PagerDuty 配置为警报接收器示例

```

# ...
receivers:
- name: default
- name: watchdog
- name: team-frontend-page
  pagerduty_configs:
    - routing_key: xxxxxxxxxx 1
# ...

```

- 1 定义 PagerDuty 集成密钥。

将电子邮件配置为警报接收器的示例

```

# ...
receivers:
- name: default
- name: watchdog
- name: team-frontend-page
  email_configs:
    - to: myemail@example.com 1

```

```

from: alertmanager@example.com ❷
smarthost: 'smtp.example.com:587' ❸
auth_username: alertmanager@example.com ❹
auth_password: password
hello: alertmanager ❺
# ...

```

- ❶ 指定要发送通知的电子邮件地址。
- ❷ 指定要发送通知的电子邮件地址。
- ❸ 指定用于发送电子邮件的 SMTP 服务器地址，包括端口号。
- ❹ 指定 Alertmanager 用于连接到 SMTP 服务器的身份验证凭据。这个示例使用用户名和密码。
- ❺ 指定要识别到 SMTP 服务器的主机名。如果没有包含此参数，则主机名默认为 **localhost**。



重要

Alertmanager 需要外部 SMTP 服务器来发送电子邮件警报。要配置电子邮件警报接收器，请确保您具有外部 SMTP 服务器所需的连接详情。

c. 添加路由配置：

```

# ...
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - matchers:
    - "alertname=Watchdog"
    repeat_interval: 2m
    receiver: watchdog
  - matchers: ❶
    - "<your_matching_rules>" ❷
    receiver: <receiver> ❸
# ...

```

- ❶ 使用 **matchers** 键名称指定警报要与节点匹配的匹配规则。如果您定义了禁止规则，请为目标匹配器使用 **target_matchers** 键名称，对于源匹配者使用 **source_matchers** 键名称。
- ❷ 指定与警报匹配的标签。
- ❸ 指定要用于警报的接收器名称。



警告

不要使用

match, **match_re**, **target_match**, **target_match_re**, **source_match**, 和 **source_match_re** 键名称，它们已弃用，并计划在以后的发行版本中删除。

警报路由示例

```
# ...
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
    - matchers:
      - "alertname=Watchdog"
      repeat_interval: 2m
      receiver: watchdog
    - matchers: ❶
      - "service=example-app"
      routes: ❷
      - matchers:
        - "severity=critical"
        receiver: team-frontend-page
# ...
```

- ❶ 本例匹配来自 **example-app** 服务的警报。
- ❷ 您可以在其他路由中创建路由，以进行更复杂的警报路由。

前面的示例将由 **example-app** 服务触发的 **critical** 严重性的警报路由到 **team-frontend-page** 接收器。通常，这些类型的警报会传给个人或关键响应团队。

4. 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=
```

5. 通过视觉化路由树来验证路由配置：

```
$ oc exec alertmanager-main-0 -n openshift-monitoring -- amtool config routes show --
alertmanager.url http://localhost:9093
```

输出示例

Routing tree:

```

.
├── default-route receiver: default
│   ├── {alertname="Watchdog"} receiver: Watchdog
│   └── {service="example-app"} receiver: default
│       └── {severity="critical"} receiver: team-frontend-page

```

其他资源

- 在 [OpenShift 4 中向 Alertmanager 发送测试警报（红帽客户门户）](#)

3.5.4.2. 使用 OpenShift Container Platform Web 控制台配置警报路由

您可以通过 OpenShift Container Platform Web 控制台配置警报路由，以确保了解集群出现的重要问题。



注意

OpenShift Container Platform Web 控制台提供比 **alertmanager-main** secret 配置警报路由的设置更少。要通过访问更多配置设置配置警报路由，请参阅“配置默认平台警报的警报路由”。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。

流程

1. 在 **Administrator** 视角中，进入 **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. 在页面的 **Receivers** 部分中，点 **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，请点 **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，请点 **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。



重要

Alertmanager 需要外部 SMTP 服务器来发送电子邮件警报。要配置电子邮件警报接收器，请确保您具有外部 SMTP 服务器所需的连接详情。

- 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，请点 **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前完全匹配的标签值，请执行以下步骤：
 - a. 在表单的 **Routing Labels** 部分中添加路由标签名称和值。
 - b. 点 **Add label** 来添加更多路由标签。
- 6. 点 **Create** 创建接收器。

其他资源

- [在 OpenShift 4 中向 Alertmanager 发送测试警报（红帽客户门户）](#)

3.5.4.3. 为默认平台警报和用户定义的警报配置不同的警报接收器

您可以为默认平台警报和用户定义的警报配置不同的警报接收器，以确保以下结果：

- 所有默认平台警报都发送到团队拥有的接收器，以收取这些警报。
- 所有用户定义的警报都发送到另一个接收器，以便团队只能专注于平台警报。

您可以使用 Cluster Monitoring Operator 添加到所有平台警报的 `openshift_io_alert_source="platform"` 标签来实现此目的：

- 使用 `openshift_io_alert_source="platform"` matcher 来匹配默认平台警报。

- 使用 `openshift_io_alert_source!="platform"` 或 `'openshift_io_alert_source=""` 匹配程序来匹配用户定义的警报。



注意

如果您启用了专用于用户定义的警报的 Alertmanager 实例，则此配置不适用。

第 4 章 配置用户工作负载监控

4.1. 准备配置用户工作负载监控堆栈

本节介绍可以配置哪些用户定义的监控组件、如何启用用户工作负载监控以及如何准备配置用户工作负载监控堆栈。



重要

- 并非所有监控堆栈的配置参数都会被公开。只有在 [Cluster Monitoring Operator 的 Config map 引用](#) 中列出的参数和字段才支持进行配置。
- 监控堆栈会带来额外的资源需求。参阅 [扩展 Cluster Monitoring Operator](#) 中的计算资源建议，并确认您有足够的资源。

4.1.1. 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 `user-workload-monitoring-config` 配置映射中用来指定这些组件的键。

表 4.1. 用于用户定义的项目的可配置监控组件

组件	user-workload-monitoring-config 配置映射键
Prometheus Operator	<code>prometheusOperator</code>
Prometheus	<code>prometheus</code>
Alertmanager	<code>alertmanager</code>
Thanos Ruler	<code>thanosRuler</code>



警告

ConfigMap 对象的不同配置更改会导致不同的结果：

- Pod 不会重新部署。因此，没有服务中断。
- 受影响的 pod 被重新部署：
 - 对于单节点集群，这会导致临时服务中断。
 - 对于多节点集群，因为高可用性，受影响的 pod 会逐渐推出，监控堆栈仍然可用。
 - 无论高可用性如何，配置和重新定义持久性卷的大小始终会导致服务中断。

每个需要更改配置映射中的步骤都包括其预期的结果。

4.1.2. 为用户定义的项目启用监控

在 OpenShift Container Platform 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。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。

4.1.2.1. 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

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** 集群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** 集群角色的用户身份访问集群。
- 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。可能需要稍等片刻，这些组件才会重新部署。

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。



注意

如果您为用户定义的项目启用监控，则默认创建 **user-workload-monitoring-config ConfigMap** 对象。

4. 验证 **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

- [用户工作负载监控第一步](#)

4.1.2.2. 授予用户权限来为用户定义的项目配置监控

作为集群管理员，您可以将 **user-workload-monitoring-config-edit** 角色分配给用户。这授予为用户定义的项目配置和管理监控的权限，而不授予他们配置和管理 OpenShift Container Platform 核心监控组件的权限。

先决条件

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

流程

1. 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
```

2. 通过显示相关角色绑定来验证用户是否已正确分配给 **user-workload-monitoring-config-edit** 角色：

```
$ oc describe rolebinding <role_binding_name> -n openshift-user-workload-monitoring
```

示例命令

```
$ oc describe rolebinding user-workload-monitoring-config-edit -n openshift-user-workload-monitoring
```

输出示例

```
Name:      user-workload-monitoring-config-edit
Labels:    <none>
Annotations: <none>
Role:
  Kind: Role
  Name: user-workload-monitoring-config-edit
Subjects:
  Kind  Name  Namespace
  ----  ---  -
  User  user1  1
```

- 1 在本例中，**user1** 分配给 **user-workload-monitoring-config-edit** 角色。

4.1.3. 为用户定义的项目启用警报路由

在 OpenShift Container Platform 中，管理员可以为用户定义的项目启用警报路由。这个过程由以下步骤组成：

- 为用户定义的项目启用警报路由：
 - 使用默认平台 Alertmanager 实例。
 - 仅对用户定义的项目使用单独的 Alertmanager 实例。
- 授予用户权限来为用户定义的项目配置警报路由。

完成这些步骤后，开发人员和其他用户可以为用户定义的项目配置自定义警报和警报路由。

其他资源

- [了解用户定义的项目的警报路由](#)

4.1.3.1. 为用户定义的警报路由启用平台 Alertmanager 实例

您可以允许用户创建使用 Alertmanager 主平台实例的用户定义警报路由配置。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

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

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

2. 在 **data/config.yaml** 下添加 **alertmanagerMain** 部分中的 **enableUserAlertmanagerConfig: true**：

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

- 1 将 **enableUserAlertmanagerConfig** 值设置为 **true**，以使用户创建使用 Alertmanager 主平台实例的用户定义警报路由配置。

3. 保存文件以使改变生效。新的配置会被自动应用。

4.1.3.2. 为用户定义的警报路由启用一个单独的 Alertmanager 实例

在一些集群中，您可能想要为用户定义的项目部署专用 Alertmanager 实例，这有助于降低默认平台 Alertmanager 实例的负载，并更好地将用户定义的警报与默认平台警报分开。在这些情况下，您可以选择性地启用一个单独的 Alertmanager 实例，以仅为用户定义的项目发送警报。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您已为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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** 下，添加 **alertmanager** 部分的 **enabled: true** 和 **enableAlertmanagerConfig: true**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    alertmanager:
      enabled: true ①
      enableAlertmanagerConfig: true ②
```

- ① 将 **enabled** 值设为 **true**，为集群中的用户定义的项目启用 Alertmanager 专用实例。将值设为 **false** 或省略键可完全为用户定义的项目禁用 Alertmanager。如果将此值设置为 **false**，或者如果省略了密钥，则用户定义的警报会路由到默认平台 Alertmanager 实例。

- ② 将 **enableAlertmanagerConfig** 值设置为 **true**，以便用户使用 **AlertmanagerConfig** 对象定义自己的警报路由配置。

3. 保存文件以使改变生效。用于用户定义的项目的 Alertmanager 专用实例会自动启动。

验证

- 验证 **user-workload** Alertmanager 实例是否已启动：

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

输出示例

NAME	VERSION	REPLICAS	AGE
user-workload	0.24.0	2	100s

4.1.3.3. 授予用户权限来为用户定义的项目配置警报路由

您可以授予用户权限来为用户定义的项目配置警报路由。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。
- 您已为用户定义的项目启用了监控。
- 要将角色分配到的用户帐户已存在。
- 已安装 OpenShift CLI (**oc**) 。

流程

- 将 **alert-routing-edit** 集群角色分配给用户定义的项目中的用户：

```
$ oc -n <namespace> adm policy add-role-to-user alert-routing-edit <user> 1
```

1 对于 **<namespace>**，替换用户定义的项目的命名空间，如 **ns1**。对于 **<user>**，替换您要为其分配该角色的帐户的用户名。

其他资源

- [配置警报通知](#)

4.1.4. 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。

您还可以向开发人员和其他用户授予不同的权限：

- 监控用户定义的项目
- 要配置用于监控用户定义的项目的组件。
- 为用户定义的项目配置警报路由
- 为用户定义的项目管理警报和静默

您可以通过分配以下监控角色或集群角色之一来授予权限：

表 4.2. 监控角色

角色名称	描述	项目
------	----	----

角色名称	描述	项目
user-workload-monitoring-config-edit	具有此角色的用户可以编辑 user-workload-monitoring-config ConfigMap 对象，为用户定义的工作负载监控配置 Prometheus、Prometheus Operator、Alertmanager 和 Thanos Ruler。	openshift-user-workload-monitoring
monitoring-alertmanager-api-reader	如果启用了用户定义的 Alertmanager，具有此角色的用户对用户定义的 Alertmanager API 具有读取访问权限。	openshift-user-workload-monitoring
monitoring-alertmanager-api-writer	如果启用了用户定义的 Alertmanager，具有此角色的用户对用户定义的 Alertmanager API 具有读写访问权限。	openshift-user-workload-monitoring

表 4.3. 监控集群角色

集群角色名称	描述	项目
monitoring-rules-view	具有此集群角色的用户具有用户定义的项目的 PrometheusRule 自定义资源 (CR) 的读取访问权限。它们也可以在 OpenShift Container Platform Web 控制台的 Developer 视角中查看警报和静默。	可以与 RoleBinding 绑定到任何用户项目。
monitoring-rules-edit	具有此集群角色的用户可为用户定义的项目创建、修改和删除 PrometheusRule CR。它们也可以在 OpenShift Container Platform Web 控制台的 Developer 视角中管理警报和静默。	可以与 RoleBinding 绑定到任何用户项目。
monitoring-edit	具有此集群角色的用户具有与具有 monitoring-rules-edit 集群角色的用户相同的权限。另外，用户可以创建、读取、修改和删除 ServiceMonitor 和 PodMonitor 资源，以从服务和 pod 中提取指标。	可以与 RoleBinding 绑定到任何用户项目。
alert-routing-edit	具有此集群角色的用户可为用户定义的项目创建、更新和删除 AlertmanagerConfig CR。	可以与 RoleBinding 绑定到任何用户项目。

其他资源

- [授予用户权限来为用户定义的项目配置监控](#)
- [授予用户权限来为用户定义的项目配置警报路由](#)

4.1.4.1. 使用 Web 控制台授予用户权限

您可以使用 OpenShift Container Platform Web 控制台为 **openshift-monitoring** 项目或自己的项目授予用户权限。

先决条件

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

流程

1. 在 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **User Management → RoleBindings → 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项目。



重要

您使用此流程向用户授予的监控角色或集群角色权限只适用于您在 **Namespace** 字段中选择的项目。

5. 从 **Role Name** 列表中选择监控角色或集群角色。
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。

4.1.4.2. 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。



重要

无论您选择的角色或集群角色，都必须作为集群管理员将其与特定项目进行绑定。

先决条件

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

- 已安装 OpenShift CLI (**oc**)。

流程

- 要为项目的用户分配监控角色，请输入以下命令：

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

- 1 将 **<role>** 替换为您要监控角色，**<user>** 替换为您要为其分配角色的用户，**<namespace>** 替换为您要授予访问权限的项目。

- 要为项目的用户分配监控集群角色，请输入以下命令：

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

- 1 将 **<cluster-role>** 替换为您要监控的集群角色，**<user>** 替换为您要为其分配集群角色的用户，**<namespace>** 替换为您要授予访问权限的项目。

4.1.5. 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。

4.1.6. 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。

4.2. 为用户工作负载监控配置性能和可扩展性

您可以配置监控堆栈来优化集群的性能和扩展。以下文档提供了有关如何分发监控组件的信息，并控制监控堆栈对 CPU 和内存资源的影响。

4.2.1. 控制监控组件的放置和分发

您可以将监控堆栈组件移到特定的节点上：

- 使用带有标记节点的 **nodeSelector** 约束，将任何监控堆栈组件移到特定的节点上。
- 分配容限以启用将组件移到污点节点。

通过这样做，您可以控制集群中监控组件的放置和分发。

通过控制监控组件的放置和分发，您可以根据特定要求或策略优化系统资源使用、提高性能和隔离工作负载。

其他资源

- [使用节点选择器移动监控组件](#)

4.2.1.1. 将监控组件移到其他节点

您可以将监控用户定义的项目的工作负载的任何组件移到特定的 worker 节点。



警告

不允许组件移到控制平面或基础架构节点。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 如果您还没有这样做，请在要运行监控组件的节点中添加标签：

```
$ oc label nodes <node_name> <node_label> 1
```

- 1 将 **<node_name>** 替换为您要添加标签的节点的名称。将 **<node_label>** 替换为所需标签的名称。

2. 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
```

3. 在 **data/config.yaml** 下为组件指定 **nodeSelector** 约束的节点标签：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    # ...
    <component>: 1
      nodeSelector:
        <node_label_1> 2
        <node_label_2> 3
    # ...
```

- 1 将 **<component>** 替换为适当的监控堆栈组件名称。

- 2 将 `<node_label_1>` 替换为添加到节点的标签。
- 3 可选：指定附加标签。如果您指定了额外的标签，则组件的 pod 仅调度到包含所有指定标签的节点上。



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 事件中与污点和容限相关的错误。

4. 保存文件以使改变生效。新配置中指定的组件会自动移到新节点上，受新配置影响的 pod 会被重新部署。

其他资源

- [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](#)
- [nodeSelector \(Kubernetes 文档\)](#)

4.2.1.2. 为监控组件分配容忍（tolerations）

您可以为监控用户定义的项目的组件分配容忍，以便将其移到污点的 worker 节点。在控制平面或基础架构节点上不允许调度。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 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"
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [为用户定义的项目启用监控](#)
- [使用节点污点控制 pod 放置](#)
- [污点和容限\(Kubernetes 文档\)](#)

4.2.2. 管理监控组件的 CPU 和内存资源

您可以通过为这些组件的资源限值和请求指定值来确保运行监控组件的容器具有足够的 CPU 和内存资源。

您可以为监控 **openshift-user-workload-monitoring** 命名空间中的用户定义的项目的监控组件配置这些限制和请求。

4.2.2.1. 指定限制和请求

要配置 CPU 和内存资源，请在 **openshift-user-workload-monitoring** 命名空间中的 **user-workload-monitoring-config ConfigMap** 对象中指定资源限值和请求值。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。

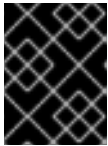
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 添加值来为您要配置的每个组件定义资源限值和请求。



重要

确保为限制设置的值始终高于为请求设置的值。否则，会出现错误，容器将不会运行。

设置资源限值和请求示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    alertmanager:
      resources:
        limits:
          cpu: 500m
          memory: 1Gi
        requests:
          cpu: 200m
          memory: 500Mi
    prometheus:
      resources:
        limits:
          cpu: 500m
          memory: 3Gi
        requests:
          cpu: 200m
          memory: 500Mi
    thanosRuler:
      resources:
        limits:
          cpu: 500m
          memory: 1Gi
        requests:
          cpu: 200m
          memory: 500Mi
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [关于为监控组件指定限制和请求](#)
- [请求和限值\(Kubernetes 文档\)](#)

4.2.3. 控制用户定义的项目中未绑定指标属性的影响

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
- 限制提取标签数量、标签名称长度以及标签值长度
- 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。

其他资源

- [控制用户定义的项目中未绑定指标属性的影响](#)
- [为用户定义的项目启用监控](#)
- [确定为什么 Prometheus 消耗大量磁盘空间](#)

4.2.3.1. 为用户定义的项目设置提取示例和标签限制

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。您还可以限制提取标签数量、标签名称长度以及标签值长度。



警告

如果您设置了 sample 或 label limits，则在达到限制后，不会为该目标提取获得进一步的示例数据。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 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 ❶
```

- ❶ 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. 将 **enforcedLabelLimit**, **enforcedLabelNameLengthLimit**, 和 **enforcedLabelValueLengthLimit** 配置添加到 **data/config.yaml**，以限制刮除的标签数量、标签名称长度以及用户定义的项目中的标签值长度：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      enforcedLabelLimit: 500 ❶
      enforcedLabelNameLengthLimit: 50 ❷
      enforcedLabelValueLengthLimit: 600 ❸
```

- ❶ 指定每次刮除的最大标签数。默认值为 **0**，代表没有指定限制。
- ❷ 指定标签名称字符的最大长度。默认值为 **0**，代表没有指定限制。
- ❸ 指定标签值字符的最大长度。默认值为 **0**，代表没有指定限制。

4. 保存文件以使改变生效。限制会自动应用。

4.2.3.2. 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** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 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 ❶
  namespace: ns1 ❷
spec:
  groups:
    - name: general.rules
      rules:
        - alert: TargetDown ❸
          annotations:
            message: '{{ printf "%.4g" $value }}% of the {{ $labels.job }}/{{ $labels.service }} targets in {{ $labels.namespace }} namespace are down.' ❹
            expr: 100 * (count(up == 0) BY (job, namespace, service) / count(up) BY (job, namespace, service)) > 10
            for: 10m ❺
            labels:
              severity: warning ❻
        - alert: ApproachingEnforcedSamplesLimit ❼
          annotations:
            message: '{{ $labels.container }} container of the {{ $labels.pod }} pod in the {{ $labels.namespace }} namespace consumes {{ $value | humanizePercentage }} of the samples limit budget.' ❽
            expr: scrape_samples_scraped/50000 > 0.8 ❾
            for: 10m ❿
            labels:
              severity: warning ⓫
```

- ❶ 定义警报规则的名称。
- ❷ 指定要部署警报规则的用户定义的项目。
- ❸ 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
```

4.2.4. 配置 pod 拓扑分布限制

您可以为用户定义的监控配置 pod 拓扑分布限制，以控制 pod 副本如何在区调度到节点。这样可确保 pod 具有高可用性并更有效地运行，因为工作负载分散在不同的数据中心或分层基础架构区域中。

您可以使用 **user-workload-monitoring-config** 配置映射为监控 pod 配置 pod 拓扑分布限制。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 在 **data/config.yaml** 字段中添加以下设置来配置 pod 拓扑分布限制：

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

```

metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: ❶
    topologySpreadConstraints:
      - maxSkew: <n> ❷
        topologyKey: <key> ❸
        whenUnsatisfiable: <value> ❹
        labelSelector: ❺
          <match_option>

```

- ❶ 指定您要为其设置 pod 拓扑分布限制的组件名称。
- ❷ 为 **maxSkew** 指定数字值，它定义了允许不均匀分布 pod 的程度。
- ❸ 为 **topologyKey** 指定节点标签键。带有具有此键和相同值标签的节点被视为在同一拓扑域中。调度程序会尝试将大量 pod 放置到每个域中。
- ❹ 为 **whenUnsatisfiable** 指定一个值。可用选项包括 **DoNotSchedule** 和 **ScheduleAnyway**。如果您希望 **maxSkew** 值定义目标拓扑和全局最小值中匹配 pod 数量之间允许的最大值，则指定 **DoNotSchedule**。如果您希望调度程序仍然调度 pod，但为可能降低 skew 的节点赋予更高的优先级，请指定 **ScheduleAnyway**。
- ❺ 指定 **labelSelector** 来查找匹配的 pod。与此标签选择器匹配的 Pod 被计算，以确定其对应拓扑域中的 pod 数量。

Thanos Ruler 的配置示例

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      topologySpreadConstraints:
        - maxSkew: 1
          topologyKey: monitoring
          whenUnsatisfiable: ScheduleAnyway
      labelSelector:
        matchLabels:
          app.kubernetes.io/name: thanos-ruler

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [关于用于监控的 pod 拓扑分布限制](#)
- [使用 pod 拓扑分布限制控制 pod 放置](#)

- [Pod 拓扑分布限制\(Kubernetes 文档\)](#)

4.3. 为用户工作负载监控存储和记录数据

存储和记录您的指标和警报数据，配置日志以指定记录哪些活动，控制 Prometheus 保留数据的时长，并为数据设置最大磁盘空间量。这些操作可帮助您保护数据并将其用于故障排除。

4.3.1. 配置持久性存储

使用持久性存储运行集群监控以获取以下优点：

- 通过将指标和警报数据存储持久性卷(PV)中来保护您的指标和警报数据。因此，它们可以在 pod 重启或重新创建后保留。
- 避免获取重复的通知，并在 Alertmanager pod 重启时丢失警报静默。



重要

在多节点集群中，您必须为 Prometheus、Alertmanager 和 Thanos Ruler 配置持久性存储，以确保高可用性。



注意

在生产环境中，强烈建议配置持久性存储。

4.3.1.1. 持久性存储的先决条件

- 分配充足的专用持久性存储，以确保磁盘不会被填满。
- 在配置持久性卷时，使用 **Filesystem** 作为 **volumeMode** 参数的存储类型值。



重要

- 不要使用原始块卷，它由 **PersistentVolume** 资源中的 **volumeMode: Block** 描述。Prometheus 无法使用原始块卷。
- Prometheus 不支持兼容 POSIX 的文件系统。例如，一些 NFS 文件系统实现不兼容 POSIX。如果要使用 NFS 文件系统进行存储，请验证与其 NFS 实现完全兼容 POSIX 的供应商。

4.3.1.2. 配置持久性卷声明

要将持久性卷 (PV) 用于监控组件，您必须配置持久性卷声明 (PVC)。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 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>: ❶
    volumeClaimTemplate:
      spec:
        storageClassName: <storage_class> ❷
        resources:
          requests:
            storage: <amount_of_storage> ❸
```

- ❶ 指定要配置 PVC 的监控组件。
- ❷ 指定现有存储类。如果没有指定存储类，则使用默认存储类。
- ❸ 指定所需的存储量。

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

PVC 配置示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      volumeClaimTemplate:
        spec:
          storageClassName: my-storage-class
          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。

- 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署并应用新的存储配置。



警告

当您使用 PVC 配置更新配置映射时，受影响的 **StatefulSet** 对象会被重新创建，从而导致临时服务中断。

其他资源

- [了解持久性存储](#)
- [PersistentVolumeClaim \(Kubernetes 文档\)](#)

4.3.1.3. 重新定义持久性卷大小

您可以为 Prometheus、Thanos Ruler 和 Alertmanager 实例重新定义持久性卷(PV)大小。您需要手动扩展持久性卷声明 (PVC)，然后更新配置组件的配置映射。



重要

您只能扩展 PVC 的大小。无法缩小存储大小。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 至少有一个 PVC 用于监控用户定义的项目的组件。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 使用更新的存储请求手动扩展 PVC。如需更多信息，请参阅 *扩展持久性卷* 中的“使用文件系统扩展持久性卷声明 (PVC)”。
2. 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
```

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

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

```

name: user-workload-monitoring-config
namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: ❶
    volumeClaimTemplate:
      spec:
        resources:
          requests:
            storage: <amount_of_storage> ❷

```

- ❶ 要更改存储大小的组件。
- ❷ 指定存储卷的新大小。它必须大于上一个值。

以下示例将 Thanos Ruler 的新 PVC 请求设置为 20GB：

thanosRuler 的存储配置示例

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      volumeClaimTemplate:
        spec:
          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。

4. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。



警告

当您使用新存储大小更新配置映射时，受影响的 **StatefulSet** 对象会被重新创建，从而导致临时服务中断。

其他资源

- [Prometheus 数据库存储要求](#)

- 使用文件系统扩展持久性卷声明 (PVC)

4.3.2. 修改 Prometheus 指标数据的保留时间和大小

默认情况下，Prometheus 会保留 24 小时的指标数据，供用户定义的项目监控。您可以修改 Prometheus 实例的保留时间，以便在删除数据时更改。您还可以设置保留指标数据使用的最大磁盘空间量。



注意

数据压缩每两小时进行一次。因此，持久性卷 (PV) 可能会在压缩前已被填满，可能会超过 **retentionSize** 限制。在这种情况下，**KubePersistentVolumeFillingUp** 警报会触发，直到 PV 上的空间低于 **retentionSize** 限制。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 在 **data/config.yaml** 下添加保留时间和大小配置：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: <time_specification> 1
      retentionSize: <size_specification> 2
```

1 保留时间：数字直接加上 **ms**（毫秒）、**s**（秒）、**m**（分钟）、**h**（小时）、**d**（天）、**w**（周）或 **y**（年）。您还可以组合指定时间值，如 **1h30m15s**。

2 保留大小：数字直接加上 **B** (bytes), **KB** (kilobytes), **MB** (megabytes), **GB** (gigabytes), **TB** (terabytes), **PB** (petabytes), 和 **EB** (exabytes)。

以下示例将 Prometheus 实例的保留时间设置为 24 小时，保留大小设为 10GB：

为 Prometheus 设置保留时间示例

■

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: 24h
      retentionSize: 10GB

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

4.3.2.1. 修改 Thanos Ruler 指标数据的保留时间

默认情况下，对于用户定义的项目，Thanos Ruler 会在 24 小时内自动保留指标数据。您可以通过在 **openshift-user-workload-monitoring** 命名空间中指定 **user-workload-monitoring-config** 配置映射中的 **time** 值来修改这些数据的保留时间。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 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. 将保留时间配置添加到 **data/config.yaml** 下：

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      retention: <time_specification> 1

```

- 1 以以下格式指定保留时间：数字直接后跟 **ms**（毫秒）、**s**（秒）、**m**（分钟）、**h**（小时）、**d**（天）、**w**（周）或 **y**（年）。您还可以组合指定时间值，如 **1h30m15s**。默认值为 **24h**。

以下示例将 Thanos Ruler 数据的保留时间设置为 10 天：

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      retention: 10d

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [Prometheus 指标的保留时间和大小](#)
- [为用户定义的项目启用监控](#)
- [Prometheus 数据库存储要求](#)
- [推荐的可配置存储技术](#)
- [了解持久性存储](#)
- [优化存储](#)

4.3.3. 为监控组件设置日志级别

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

以下日志级别可应用到 **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

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

- 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>: ❶
    logLevel: <log_level> ❷
```

- ❶ 您要为其设置日志级别的监控堆栈组件。可用的组件值包括 **prometheus**、**alertmanager**、**prometheusOperator** 和 **thanosRuler**。
- ❷ 为组件设置的日志级别。可用值为 **error**、**warn**、**info** 和 **debug**。默认值为 **info**。

- 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。
- 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 **prometheus-operator** 部署的日志级别：

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

输出示例

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

- 检查组件的 Pod 是否正在运行。以下示例列出了 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。

4.3.4. 为 Prometheus 启用查询日志文件

您可以将 Prometheus 配置为将引擎运行的所有查询写入到日志文件。



重要

由于不支持日志轮转，因此仅在需要对问题进行故障排除时才临时启用此功能。完成故障排除后，通过恢复您对 **ConfigMap** 对象所做的更改来禁用查询日志记录，以启用该功能。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 将 Prometheus 的 **queryLogFile** 参数添加到 **data/config.yaml** 下：

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

- ❶ 添加要记录查询的文件的完整路径。

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。
4. 验证组件的 pod 是否正在运行。以下示例命令列出了 pod 的状态：

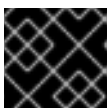
```
$ oc -n openshift-user-workload-monitoring get pods
```

输出示例

```
...
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
...
```

5. 读取查询日志：

```
$ oc -n openshift-user-workload-monitoring exec prometheus-user-workload-0 -- cat <path>
```



重要

在检查了记录的查询信息后，恢复配置映射的设置。

其他资源

- [为用户定义的项目启用监控](#)

4.4. 为用户工作负载监控配置指标

配置指标集合，以监控集群组件和您自己的工作负载的执行方式。

您可以将最接近的指标发送到远程系统进行长期存储，并将集群 ID 标签添加到指标，以识别来自不同集群的数据。

其他资源

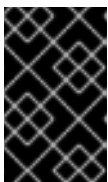
- [了解指标](#)

4.4.1. 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

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 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](#)。



重要

红帽只提供配置远程写入发送者的信息，而不提供有关配置接收器端点的指导。客户负责设置自己的端点，这些端点与远程写入兼容。端点接收器配置的问题不包括在红帽产品支持中。

- 您已为远程写入端点在 **Secret** 对象中设置身份验证凭证。您必须在 **openshift-user-workload-monitoring** 命名空间中创建 secret。



警告

要减少安全风险，请使用 HTTPS 和身份验证向端点发送指标。

流程

1. 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
```

2. 在 **data/config.yaml/prometheus** 下添加一个 **remoteWrite:** 部分，如下例所示：

```
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.example.com" ❶
          <endpoint_authentication_credentials> ❷
```

- ❶ 远程写入端点的 URL。
- ❷ 端点的身份验证方法和凭据。目前支持的身份验证方法有 AWS 签名版本 4，使用 HTTP **Authorization** 请求标头、基本身份验证、OAuth 2.0 和 TLS 客户端进行身份验证。有关支持的身份验证方法示例配置，请参阅 [支持的远程写入身份验证设置](#)。

3. 在身份验证凭证后添加 write relabel 配置值：

```
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.example.com"
          <endpoint_authentication_credentials>
          writeRelabelConfigs:
            - <your_write_relabel_configs> ❶
```

- ❶ 为您要发送到远程端点的指标添加配置。

转发一个名为 **my_metric** 的单个指标的示例

```
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.example.com"
```

```
writeRelabelConfigs:
- sourceLabels: [__name__]
  regex: 'my_metric'
  action: keep
```

在 my_namespace 命名空间中转发名为 my_metric_1 和 my_metric_2 的指标示例

```
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.example.com"
        writeRelabelConfigs:
        - sourceLabels: [__name__,namespace]
          regex: '(my_metric_1|my_metric_2);my_namespace'
          action: keep
```

4. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [writeRelabelConfigs](#)
- [relabel_config](#) (Prometheus 文档)

4.4.1.1. 支持的远程写入身份验证设置

您可以使用不同的方法通过远程写入端点进行身份验证。目前支持的身份验证方法有 AWS 签名版本 4、基本身份验证、授权、OAuth 2.0 和 TLS 客户端。下表提供有关用于远程写入的受支持身份验证方法的详情。

身份验证方法	配置映射字段	描述
AWS 签名版本 4	sigv4	此方法使用 AWS 签名版本 4 身份验证为请求签名。您不能搭配授权、OAuth 2.0 或基本身份验证同时使用此方法。
基本身份验证 (Basic authentication)	basicAuth	基本身份验证使用配置的用户名和密码在每个远程写入请求上设置授权标头。
授权	授权	授权使用配置的令牌在每个远程写入请求上设置 Authorization 标头。

身份验证方法	配置映射字段	描述
OAuth 2.0	oauth2	OAuth 2.0 配置使用客户端凭据授予类型。Prometheus 使用指定的客户端 ID 和客户端 secret 从 tokenUrl 获取访问令牌来访问远程写入端点。您不能与授权、AWS 签名版本 4 或基本身份验证同时使用此方法。
TLS 客户端	tlsConfig	TLS 客户端配置指定 CA 证书、客户端证书和客户端密钥文件信息，用于使用 TLS 与远程写入端点服务器进行身份验证。示例配置假定您已创建了 CA 证书文件、客户端证书文件和客户端密钥文件。

4.4.1.2. 远程写入身份验证设置示例

以下示例展示了可用于连接到远程写入端点的不同身份验证设置。每个示例还演示了如何配置包含身份验证凭据和其他相关设置的对应 **Secret** 对象。每个示例配置用于监控 **openshift-user-workload-monitoring** 命名空间中的用户定义的项目的身份验证。

4.4.1.2.1. AWS 签名版本 4 验证的 YAML 示例

以下显示了 **openshift-user-workload-monitoring** 命名空间中名为 **sigv4-credentials** 的 **sigv4** secret 的设置。

```
apiVersion: v1
kind: Secret
metadata:
  name: sigv4-credentials
  namespace: openshift-user-workload-monitoring
stringData:
  accessKey: <AWS_access_key> ❶
  secretKey: <AWS_secret_key> ❷
type: Opaque
```

❶ AWS API 访问密钥。

❷ AWS API secret 密钥。

下面显示了一个 AWS Signature Version 4 远程写入身份验证设置示例，它使用 **openshift-user-workload-monitoring** 命名空间中名为 **sigv4-credentials** 的 **Secret** 对象：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
```

```

namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      remoteWrite:
        - url: "https://authorization.example.com/api/write"
      sigv4:
        region: <AWS_region> ❶
        accessKey:
          name: sigv4-credentials ❷
          key: accessKey ❸
        secretKey:
          name: sigv4-credentials ❹
          key: secretKey ❺
        profile: <AWS_profile_name> ❻
        roleArn: <AWS_role_arn> ❼

```

- ❶ AWS 区域。
- ❷ ❹ 包含 AWS API 访问凭证的 **Secret** 对象的名称。
- ❸ 在指定 **Secret** 对象中包含 AWS API 访问密钥的密钥。
- ❺ 在指定的 **Secret** 对象中包含 AWS API secret 键的密钥。
- ❻ 用于验证的 AWS 配置集的名称。
- ❼ 分配给角色的 Amazon 资源名称(ARN)的唯一标识符。

4.4.1.2.2. 用于基本身份验证的 YAML 示例

以下显示了 **openshift-user-workload-monitoring** 命名空间中名为 **rw-basic-auth** 的 **Secret** 对象的基本身份验证设置示例：

```

apiVersion: v1
kind: Secret
metadata:
  name: rw-basic-auth
  namespace: openshift-user-workload-monitoring
stringData:
  user: <basic_username> ❶
  password: <basic_password> ❷
type: Opaque

```

- ❶ 用户名。
- ❷ 密码。

以下示例显示了使用 **openshift-user-workload-monitoring** 命名空间中名为 **rw-basic-auth** 的 **Secret** 对象的 **basicAuth** 远程写入配置。它假设您已为端点设置了身份验证凭据。

```

apiVersion: v1

```

```

kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      remoteWrite:
        - url: "https://basicauth.example.com/api/write"
      basicAuth:
        username:
          name: rw-basic-auth ❶
          key: user ❷
        password:
          name: rw-basic-auth ❸
          key: password ❹

```

❶ ❸ 包含身份验证凭据的 **Secret** 对象的名称。

❷ 在指定的 **Secret** 对象中包含用户名的密钥。

❹ 在指定 **Secret** 对象中包含密码的密钥。

4.4.1.2.3. 使用 **Secret** 对象通过 bearer 令牌进行身份验证的 YAML 示例

以下显示了 **openshift-user-workload-monitoring** 命名空间中名为 **rw-bearer-auth** 的 **Secret** 对象的 bearer 令牌设置：

```

apiVersion: v1
kind: Secret
metadata:
  name: rw-bearer-auth
  namespace: openshift-user-workload-monitoring
stringData:
  token: <authentication_token> ❶
type: Opaque

```

❶ 身份验证令牌。

以下显示了在 **openshift-user-workload-monitoring** 命名空间中使用名为 **rw-bearer-auth** 的 **Secret** 对象的 bearer 令牌配置映射设置示例：

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      remoteWrite:
        - url: "https://authorization.example.com/api/write"
      authorization:

```

```

type: Bearer ❶
credentials:
  name: rw-bearer-auth ❷
  key: token ❸

```

- ❶ 请求的验证类型。默认值为 **Bearer**。
- ❷ 包含身份验证凭据的 **Secret** 对象的名称。
- ❸ 在指定的 **Secret** 对象中包含身份验证令牌的密钥。

4.4.1.2.4. 用于 OAuth 2.0 验证的 YAML 示例

以下显示了 **openshift-user-workload-monitoring** 命名空间中名为 **oauth2-credentials** 的 **Secret** 对象的 OAuth 2.0 设置示例：

```

apiVersion: v1
kind: Secret
metadata:
  name: oauth2-credentials
  namespace: openshift-user-workload-monitoring
stringData:
  id: <oauth2_id> ❶
  secret: <oauth2_secret> ❷
type: Opaque

```

- ❶ OAuth 2.0 ID。
- ❷ OAuth 2.0 secret。

下面显示了一个 **oauth2** 远程写入身份验证示例配置，它使用 **openshift-user-workload-monitoring** 命名空间中名为 **oauth2-credentials** 的 **Secret** 对象：

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      remoteWrite:
        - url: "https://test.example.com/api/write"
      oauth2:
        clientId:
          secret:
            name: oauth2-credentials ❶
            key: id ❷
        clientSecret:
          name: oauth2-credentials ❸
          key: secret ❹
        tokenUrl: https://example.com/oauth2/token ❺

```

```
scopes: 6
- <scope_1>
- <scope_2>
endpointParams: 7
  param1: <parameter_1>
  param2: <parameter_2>
```

1 3 对应的 **Secret** 对象的名称。请注意，**ClientId** 可以引用 **ConfigMap** 对象，但 **clientSecret** 必须引用 **Secret** 对象。

2 4 在指定 **Secret** 对象中包含 OAuth 2.0 凭证的密钥。

5 用于通过指定的 **clientId** 和 **clientSecret** 获取令牌的 URL。

6 授权请求的 OAuth 2.0 范围。这些范围限制了令牌可以访问的数据。

7 授权服务器所需的 OAuth 2.0 授权请求参数。

4.4.1.2.5. TLS 客户端身份验证的 YAML 示例

以下显示了 **openshift-user-workload-monitoring** 命名空间中名为 **mtls-bundle** 的 **tls Secret** 对象的 TLS 客户端设置示例。

```
apiVersion: v1
kind: Secret
metadata:
  name: mtls-bundle
  namespace: openshift-user-workload-monitoring
data:
  ca.crt: <ca_cert> 1
  client.crt: <client_cert> 2
  client.key: <client_key> 3
type: tls
```

1 Prometheus 容器中用于验证服务器证书的 CA 证书。

2 用于与服务器进行身份验证的客户端证书。

3 客户端密钥。

以下示例显示了使用名为 **mtls-bundle** 的 TLS **Secret** 对象的 **tlsConfig** 远程写入身份验证配置。

```
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.example.com"
      tlsConfig:
```

```

ca:
  secret:
    name: mtls-bundle ❶
    key: ca.crt ❷
cert:
  secret:
    name: mtls-bundle ❸
    key: client.crt ❹
keySecret:
  name: mtls-bundle ❺
  key: client.key ❻

```

❶ ❸ ❺ 包含 TLS 身份验证凭证的对应 **Secret** 对象的名称。请注意，**ca** 和 **cert** 可以引用 **ConfigMap** 对象，但 **keySecret** 必须引用 **Secret** 对象。

❷ 指定 **Secret** 对象中的键，其中包含端点的 CA 证书。

❹ 指定 **Secret** 对象中的键，其中包含端点的客户端证书。

❻ 包含客户端密钥 secret 的指定 **Secret** 对象中的密钥。

4.4.1.3. 远程写入队列配置示例

您可以使用 **queueConfig** 对象来远程写入，以调优远程写入队列参数。以下示例显示了 **queue** 参数，以及它们用于监控 **openshift-user-workload-monitoring** 命名空间中的用户定义的项目的默认值。

使用默认值进行远程写入参数配置示例

```

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.example.com"
          <endpoint_authentication_credentials>
      queueConfig:
        capacity: 10000 ❶
        minShards: 1 ❷
        maxShards: 50 ❸
        maxSamplesPerSend: 2000 ❹
        batchSendDeadline: 5s ❺
        minBackoff: 30ms ❻
        maxBackoff: 5s ❼
        retryOnRateLimit: false ❽

```

❶ 在从队列中丢弃前，每个分片要缓冲的样本数量。

❷ 最小分片数量。

- 3 最大分片数量。
- 4 每个发送的最大样本数。
- 5 样本在缓冲区中等待的最长时间。
- 6 在重试一个失败的请求前等待的初始时间。每次重试的时间都会加倍，最长为 **maxbackoff** 时间。
- 7 在重试一个失败的请求前等待的最长时间。
- 8 将此参数设为 **true**，以在从远程写入存储接收 429 状态代码后重试请求。

其他资源

- [用于远程写入的 Prometheus REST API 参考](#)
- [远程写入兼容端点\(Prometheus 文档\)](#)
- [远程写入调整\(Prometheus 文档\)](#)
- [了解 secret](#)

4.4.1.4. 远程写入指标表

下表包含远程写入和远程 write-adjacent 指标，其中包含进一步的描述，以帮助在远程写入配置过程中解决问题。

指标	描述
prometheus_remote_storage_highest_timestamp_in_seconds	显示 Prometheus 存储在任意样本的 write-ahead 日志 (WAL) 的最新时间戳。
prometheus_remote_storage_queue_highest_sent_timestamp_seconds	显示远程写入队列成功发送的最新时间戳。
prometheus_remote_storage_samples_retried_total	远程写入无法发送的示例数量，必须重新发送到远程存储。如果此指标一直稳定在高速率，则表示网络或远程存储端点的问题。
prometheus_remote_storage_shards	显示当前为每个远程端点运行多少个分片。
prometheus_remote_storage_shards_desired	根据当前的写入吞吐量和传入与发送样本的速率显示计算的分片数量。
prometheus_remote_storage_shards_max	显示基于当前配置的分片的最大数量。
prometheus_remote_storage_shards_min	显示基于当前配置的分片的最小数量。
prometheus_tsdb_wal_segment_current	Prometheus 当前正在写入新数据的 WAL 段文件。

指标	描述
<code>prometheus_wal_watcher_current_segment</code>	每个远程写入实例当前读取的 WAL 段文件。

4.4.2. 为指标创建集群 ID 标签

您可以通过在 **openshift-user-workload-monitoring** 命名空间中为 **user-workload-monitoring-config** 配置映射中添加远程写入存储的 **write_relabel** 设置来为指标创建集群 ID 标签。



注意

当 Prometheus 提取公开 **namespace** 标签的用户工作负载目标时，系统会将此标签存储为 **exported_namespace**。此行为可确保最终命名空间标签值等于目标 pod 的命名空间。您不能将 **PodMonitor** 或 **ServiceMonitor** 对象的 **honorLabels** 字段的值设置为 **true** 来覆盖此默认配置。

先决条件

- 您可以使用具有 **cluster-admin** 集群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。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。
- 您已配置了远程写入存储。

流程

1. 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
```

2. 在 **data/config.yaml/prometheus/remoteWrite** 下的 **writeRelabelConfigs:** 部分中，添加集群 ID 重新标记配置值：

```
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.example.com"
          <endpoint_authentication_credentials>
          writeRelabelConfigs: 1
            - <relabel_config> 2
```

- 1 为您要发送到远程端点的指标添加写入重新标记配置列表。
- 2 替换发送到远程写入端点的指标的标签配置。

以下示例演示了如何使用集群 ID 标签 **cluster_id** 转发指标：

```
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.example.com"
          writeRelabelConfigs:
            - sourceLabels:
                - __tmp_openshift_cluster_id__ 1
              targetLabel: cluster_id 2
              action: replace 3
```

- 1 系统最初应用名为 **__tmp_openshift_cluster_id__** 的临时集群 ID 源标签。此临时标签由您指定的集群 ID 标签名称替换。
- 2 指定发送到远程写入存储的指标的集群 ID 标签名称。如果您使用指标已存在的标签名称，则该值会使用这个集群 ID 标签的名称覆盖。对于标签名称，不要使用 **__tmp_openshift_cluster_id__**。最后重新标记步骤会删除使用此名称的标签。
- 3 **replace** write relabel 操作，将临时标签替换为传出指标的目标标签。这个操作是默认行为，如果没有指定任何操作，则会被应用。

3. 保存文件以使改变生效。新的配置会被自动应用。

其他资源

- [在指标中添加集群 ID 标签](#)
- [获取集群 ID](#)

4.4.3. 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。

4.4.3.1. 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。

先决条件

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

流程

1. 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:0.4.2
          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
```

4.4.3.2. 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** 集群角色或 **monitoring-edit** 集群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。

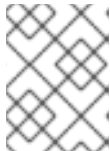
流程

1. 创建名为 **example-app-service-monitor.yaml** 的新 YAML 配置文件。
2. 将 **ServiceMonitor** 资源添加到 YAML 文件中。以下示例创建一个名为 **prometheus-example-monitor** 的服务监控器，用于提取 **ns1** 命名空间中的 **prometheus-example-app** 服务公开的指标：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1 ❶
spec:
  endpoints:
    - interval: 30s
      port: web ❷
```

```
scheme: http
selector: ❸
matchLabels:
  app: prometheus-example-app
```

- ❶ 指定服务运行的用户定义的命名空间。
- ❷ 指定要由 Prometheus 提取的端点端口。
- ❸ 配置选择器，根据其元数据标签匹配您的服务。



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. 验证 **ServiceMonitor** 资源是否正在运行：

```
$ oc -n <namespace> get servicemonitor
```

输出示例

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

4.4.3.3. 服务端点身份验证设置示例

您可以使用 **ServiceMonitor** 和 **PodMonitor** 自定义资源定义(CRD) 为用户定义的项目监控配置服务端点身份验证。

以下示例显示了 **ServiceMonitor** 资源的不同身份验证设置。每个示例演示了如何配置包含身份验证凭据和其他相关设置的对应 **Secret** 对象。

4.4.3.3.1. 使用 bearer 令牌的 YAML 身份验证示例

以下示例显示了 **ns1** 命名空间中名为 **example-bearer-auth** 的 **Secret** 对象的 bearer 令牌设置：

bearer 令牌 secret 示例

```
apiVersion: v1
kind: Secret
metadata:
  name: example-bearer-auth
  namespace: ns1
stringData:
  token: <authentication_token> ❶
```

- 1 指定身份验证令牌。

以下示例显示了 **ServiceMonitor** CRD 的 bearer 令牌身份验证设置。这个示例使用名为 **example-bearer-auth** 的 **Secret** 对象：

bearer 令牌身份验证设置示例

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
spec:
  endpoints:
    - authorization:
        credentials:
          key: token 1
          name: example-bearer-auth 2
      port: web
  selector:
    matchLabels:
      app: prometheus-example-app
```

- 1 在指定的 **Secret** 对象中包含身份验证令牌的密钥。

- 2 包含身份验证凭据的 **Secret** 对象的名称。



重要

不要使用 **bearerTokenFile** 来配置 bearer 令牌。如果使用 **bearerTokenFile** 配置，**ServiceMonitor** 资源将被拒绝。

4.4.3.3.2. 用于基本身份验证的 YAML 示例

以下示例显示了 **ns1** 命名空间中名为 **example-basic-auth** 的 **Secret** 对象基本身份验证设置：

基本身份验证 secret 示例

```
apiVersion: v1
kind: Secret
metadata:
  name: example-basic-auth
  namespace: ns1
stringData:
  user: <basic_username> 1
  password: <basic_password> 2
```

- 1 指定用于身份验证的用户名。

- 2 指定用于身份验证的密码。

以下示例显示了 **ServiceMonitor** CRD 的基本身份验证设置。这个示例使用名为 **example-basic-auth** 的 **Secret** 对象：

基本身份验证设置示例

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
spec:
  endpoints:
  - basicAuth:
      username:
        key: user ❶
        name: example-basic-auth ❷
      password:
        key: password ❸
        name: example-basic-auth ❹
    port: web
  selector:
    matchLabels:
      app: prometheus-example-app
```

❶ 在指定的 **Secret** 对象中包含用户名的密钥。

❷ ❹ 包含基本身份验证的 **Secret** 对象的名称。

❸ 在指定 **Secret** 对象中包含密码的密钥。

4.4.3.3.3. 使用 OAuth 2.0 的 YAML 身份验证示例

以下示例显示了 **ns1** 命名空间中名为 **example-oauth2** 的 **Secret** 对象的 OAuth 2.0 设置：

OAuth 2.0 secret 示例

```
apiVersion: v1
kind: Secret
metadata:
  name: example-oauth2
  namespace: ns1
stringData:
  id: <oauth2_id> ❶
  secret: <oauth2_secret> ❷
```

❶ 指定 OAuth 2.0 ID。

❷ 指定 OAuth 2.0 secret。

以下示例显示了 **ServiceMonitor** CRD 的 OAuth 2.0 身份验证设置。这个示例使用名为 **example-oauth2** 的 **Secret** 对象：

OAuth 2.0 身份验证设置示例

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
spec:
  endpoints:
  - oauth2:
      clientId:
        secret:
          key: id ❶
          name: example-oauth2 ❷
      clientSecret:
        key: secret ❸
        name: example-oauth2 ❹
      tokenUrl: https://example.com/oauth2/token ❺
  port: web
  selector:
    matchLabels:
      app: prometheus-example-app
```

- ❶ 在指定 **Secret** 对象中包含 OAuth 2.0 ID 的密钥。
- ❷ ❹ 包含 OAuth 2.0 凭证的 **Secret** 对象的名称。
- ❸ 在指定 **Secret** 对象中包含 OAuth 2.0 secret 的密钥。
- ❺ 用于通过指定的 **clientId** 和 **clientSecret** 获取令牌的 URL。

其他资源

- [为用户定义的项目启用监控](#)
- [使用 ServiceMonitor 配置中的 TLS 提取 Prometheus 指标](#)（红帽客户门户网站 的文章）
- [PodMonitor API](#)
- [ServiceMonitor API](#)

4.5. 为用户工作负载监控配置警报和通知

您可以配置本地或远程 Alertmanager 实例，将警报从 Prometheus 路由到端点接收器。您还可以将自定义标签附加到所有时间序列和警报，以添加有用的元数据信息。

4.5.1. 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。

您可以添加外部 Alertmanager 实例来路由用户定义的项目的警报。

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

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 在 **data/config.yaml/<component>** 下添加一个带有配置详情的 **additionalAlertmanagerConfigs** 部分：

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

- ❷ 将 **<alertmanager_specification>** 替换为额外的 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**)。

- ❶ 将 **<component>** 替换为两个支持的外部 Alertmanager 组件之一：**prometheus** 或 **thanosRuler**。

以下示例配置映射使用带有客户端 TLS 身份验证的 bearer 令牌为 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

```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

4.5.2. 为 Alertmanager 配置 secret

OpenShift Container Platform 监控堆栈包括 Alertmanager，它将警报从 Prometheus 路由到端点接收器。如果您需要通过接收器进行身份验证以便 Alertmanager 能够向它发送警报，您可以将 Alertmanager 配置为使用包含接收器身份验证凭据的 secret。

例如，您可以将 Alertmanager 配置为使用 secret 与需要由私有证书颁发机构 (CA) 发布的证书的端点接收器进行身份验证。您还可以将 Alertmanager 配置为使用 secret 与需要用于基本 HTTP 身份验证密码文件的接收器进行身份验证。在这两种情况下，身份验证详情都包含在 **Secret** 对象中，而不是包含在 **ConfigMap** 对象中。

4.5.2.1. 在 Alertmanager 配置中添加 secret

您可以通过编辑 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config** 配置映射，将 secret 添加到 Alertmanager 配置中。

将 secret 添加到配置映射后，secret 作为一个卷挂载到 Alertmanager Pod 的 **alertmanager** 容器中的 **/etc/alertmanager/secrets/<secret_name>** 的卷中。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。
- 您已创建了要在 **openshift-user-workload-monitoring** 项目中的 Alertmanager 中配置的 secret。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 使用以下配置在 **data/config.yaml/alertmanager** 下添加一个 **secrets:** 部分：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    alertmanager:
      secrets: ❶
      - <secret_name_1> ❷
      - <secret_name_2>
```

- ❶ 本节包含要挂载到 Alertmanager 中的 secret。secret 必须位于与 Alertmanager 对象相同的命名空间中。
- ❷ 包含接收器身份验证凭证的 **Secret** 对象的名称。如果您添加多个 secret，请将每个 secret 放在新行中。

以下示例配置映射设置将 Alertmanager 配置为使用名为 **test-secret-basic-auth** 和 **test-secret-api-token** 的两个 **Secret** 对象：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    alertmanager:
      secrets:
        - test-secret-basic-auth
        - test-secret-api-token
```

3. 保存文件以使改变生效。新的配置会被自动应用。

4.5.3. 在时间序列和警报中附加额外标签

您可以使用 Prometheus 的外部标签功能，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

- 您可以使用具有 **cluster-admin** 集群角色或具有 **openshift-user-workload-monitoring** 项目中的 **user-workload-monitoring-config-edit** 角色的用户访问集群。
- 集群管理员为用户定义的项目启用了监控。

- 已安装 OpenShift CLI(**oc**)。

流程

1. 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
```

2. 在 **data/config.yaml** 下定义您要为每个指标添加的标签：

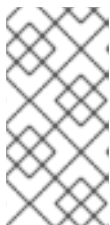
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      externalLabels:
        <key>: <value> 1
```

- 1 将 **<key>: <value>** 替换为键值对，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。
- 不要使用 **cluster** 作为密钥名称。使用它可能会导致开发人员仪表板无法看到数据的问题。



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所有时间序列和警报中，请使用以下示例：

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

```
prometheus:
  externalLabels:
    region: eu
    environment: prod
```

3. 保存文件以使改变生效。受新配置影响的 Pod 会自动重新部署。

其他资源

- [为用户定义的项目启用监控](#)

4.5.4. 配置警报通知

在 OpenShift Container Platform 中，管理员可以使用以下方法为用户定义的项目启用警报路由：

- 使用默认平台 Alertmanager 实例。
- 仅对用户定义的项目使用单独的 Alertmanager 实例。

具有 **alert-routing-edit** 集群角色的开发人员和其他用户可以通过配置警报接收器为其用户定义的项目配置自定义警报通知。



注意

查看用户定义的项目的警报路由的以下限制：

- 用户定义的警报路由的范围为定义资源的命名空间。例如，命名空间 **ns1** 中的路由配置仅适用于同一命名空间中的 **PrometheusRules** 资源。
- 当命名空间不包括在用户定义的监控中时，命名空间中的 **AlertmanagerConfig** 资源将成为 Alertmanager 配置的一部分。

其他资源

- [了解用户定义的项目的警报路由](#)
- [将通知发送到外部系统](#)
- [PagerDuty 网站](#)
- [Prometheus Integration Guide \(PagerDuty 文档\)](#)
- [监控组件的支持版本列表](#)
- [为用户定义的项目启用警报路由](#)

4.5.4.1. 为用户定义的项目配置警报路由

如果您是一个带有 **alert-routing-edit** 集群角色的非管理员用户，您可以创建或编辑用户定义的项目的警报路由。

先决条件

- 集群管理员为用户定义的项目启用了监控。

- 集群管理员为用户定义的项目启用了警报路由。
- 您以具有您要为其创建警报路由的项目的 **alert-routing-edit** 集群角色的用户身份登录。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 创建用于警报路由的 YAML 文件。此流程中的示例使用名为 **example-app-alert-routing.yaml** 的文件。
2. 在文件中添加 **AlertmanagerConfig** YAML 定义。例如：

```
apiVersion: monitoring.coreos.com/v1beta1
kind: AlertmanagerConfig
metadata:
  name: example-routing
  namespace: ns1
spec:
  route:
    receiver: default
    groupBy: [job]
  receivers:
  - name: default
    webhookConfigs:
    - url: https://example.org/post
```

3. 保存该文件。
4. 将资源应用到集群：

```
$ oc apply -f example-app-alert-routing.yaml
```

配置会自动应用到 Alertmanager pod。

其他资源

- [在 OpenShift 4 中向 Alertmanager 发送测试警报（红帽客户门户）](#)

4.5.4.2. 使用 Alertmanager secret 为用户定义的项目配置警报路由

如果您启用了专用于用户定义的警报路由的 Alertmanager 实例，您可以通过编辑 **openshift-user-workload-monitoring** 命名空间中的 **alertmanager-user-workload** secret 来自定义实例发送通知的位置和方式。



注意

OpenShift Container Platform Alertmanager 配置中也支持支持上游 Alertmanager 的所有功能。要检查受支持的上游 Alertmanager 版本的所有配置选项，请参阅 [Alertmanager 配置](#) (Prometheus 文档)。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。

- 您已为用户定义的警报路由启用了单独的 Alertmanager 实例。
- 已安装 OpenShift CLI(**oc**)。

流程

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

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

2. 编辑 **alertmanager.yaml** 中的配置：

```
route:
  receiver: Default
  group_by:
    - name: Default
  routes:
    - matchers:
        - "service = prometheus-example-monitor" ❶
      receiver: <receiver> ❷
  receivers:
    - name: Default
    - name: <receiver>
      <receiver_configuration> ❸
```

- ❶ 指定与警报匹配的标签。这个示例将带有 **service="prometheus-example-monitor"** 标签的所有警报作为目标。
- ❷ 指定用于警报组的接收器名称。
- ❸ 指定接收器配置。

3. 应用文件中的新配置：

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

其他资源

- [在 OpenShift 4 中向 Alertmanager 发送测试警报（红帽客户门户）](#)

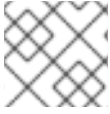
4.5.4.3. 为默认平台警报和用户定义的警报配置不同的警报接收器

您可以为默认平台警报和用户定义的警报配置不同的警报接收器，以确保以下结果：

- 所有默认平台警报都发送到团队拥有的接收器，以收取这些警报。
- 所有用户定义的警报都发送到另一个接收器，以便团队只能专注于平台警报。

您可以使用 Cluster Monitoring Operator 添加到所有平台警报的 **openshift_io_alert_source="platform"** 标签来实现此目的：

- 使用 `openshift_io_alert_source="platform"` matcher 来匹配默认平台警报。
- 使用 `openshift_io_alert_source!="platform"` 或 `'openshift_io_alert_source=""` 匹配程序来匹配用户定义的警报。



注意

如果您启用了专用于用户定义的警报的 Alertmanager 实例，则此配置不适用。

第 5 章 访问指标

5.1. 以管理员身份访问指标

您可以访问指标来监控集群组件和工作负载的性能。

其他资源

- [了解指标](#)

5.1.1. 查看可用指标列表

作为集群管理员或具有所有项目查看权限的用户，您可以查看集群中可用的指标列表，并以 JSON 格式输出列表。

先决条件

- 您是一个集群管理员，或者您可以使用具有 **cluster-monitoring-view** 集群角色的用户访问集群。
- 已安装 OpenShift Container Platform CLI (**oc**)。
- 您已获取 Thanos Querier 的 OpenShift Container Platform API 路由。
- 您可以使用 **oc whoami -t** 命令获取 bearer 令牌。



重要

您只能使用 bearer 令牌身份验证来访问 Thanos Querier API 路由。

流程

1. 如果您还没有为 Thanos Querier 获取 OpenShift Container Platform API 路由，请运行以下命令：

```
$ oc get routes -n openshift-monitoring thanos-querier -o jsonpath='{.status.ingress[0].host}'
```

2. 运行以下命令，以 JSON 格式从 Thanos Querier API 路由检索指标列表。此命令使用 **oc** 通过 bearer 令牌进行身份验证。

```
$ curl -k -H "Authorization: Bearer $(oc whoami -t)"  
https://<thanos_querier_route>/api/v1/metadata 1
```

- 1** 将 **<thanos_querier_route>** 替换为 Thanos Querier 的 OpenShift Container Platform API 路由。

5.1.2. 使用 OpenShift Container Platform Web 控制台查询所有项目的指标

您可以使用 OpenShift Container Platform 指标查询浏览器来运行 Prometheus Query Language (PromQL) 查询来检查图表中呈现的指标。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，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。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户访问集群，或者具有所有项目的查看权限。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 从 OpenShift Container Platform Web 控制台中的 **Administrator** 视角，选择 **Observe** → **Metrics**。
2. 要添加一个或多个查询，请执行以下操作之一：

选项	描述
创建自定义查询。	<p>将 Prometheus Query Language (PromQL) 查询添加到 Expression 字段中。</p> <p>当您输入 PromQL 表达式时，自动完成建议会出现在下拉列表中。这些建议包括功能、指标、标签和时间令牌。您可以使用键盘箭头选择其中一项建议的项目，然后按 Enter 将项目添加到您的表达式中。您还可以将鼠标指针移到建议的项目上，以查看该项目的简短描述。</p>
添加多个查询。	选择 Add query 。
复制现有的查询。	<p>选择查询旁边的 Options 菜单 ，然后选择 Duplicate 查询。</p>
禁用查询正在运行。	<p>选择查询旁边的 Options 菜单  并选择 Disable query。</p>

3. 要运行您创建的查询，请选择 **Run queries**。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。



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。

- 4. 可选：保存页面 URL，以便在将来再次使用此查询。
- 5. 探索视觉化指标。最初，图表中显示所有启用的查询中的所有指标。您可以通过执行以下操作来选择显示哪些指标：

选项	描述
隐藏查询中的所有指标。	点查询的 Options 菜单  并点 Hide all series。
隐藏特定指标。	前往查询表，再点指标名称旁边的带颜色的方格。
放大图表并更改时间范围。	任一： <ul style="list-style-type: none">● 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。
重置时间范围。	选择 Reset zoom。
在特定时间点显示所有查询的输出。	将鼠标光标悬停在图表上。弹出框中会显示查询输出。
隐藏图表。	选择 Hide graph。

其他资源

- [查询 Prometheus \(Prometheus 文档\)](#)

5.1.3. 获取有关指标目标的详细信息

您可以使用 OpenShift Container Platform Web 控制台来查看、搜索和过滤当前用于提取的端点，这有助于识别和排除问题。例如，您可以查看目标端点的当前状态，以查看 OpenShift Container Platform 监控无法从目标组件中提取指标。

Metrics Targets 页面显示默认 OpenShift Container Platform 项目和用户定义的项目的目标。

先决条件

- 您可以访问集群，作为您要查看指标目标的项目的管理员。

流程

1. 在 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **Observe → Targets**。此时会打开 **Metrics targets** 页面，其中包含为指标提取的所有服务端点目标的列表。本页显示了 OpenShift Container Platform 默认项目和用户定义的项目的目标的详细信息。本页列出了每个目标的以下信息：
 - 正在提取的服务端点 URL

- 被监控的 **ServiceMonitor** 资源
- 目标的 **up** 或 **down** 状态
- 命名空间
- 最后提取的时间
- 最后一次提取持续的时间

2. 可选：要查找特定的目标，请执行以下操作：

选项	描述
根据状态和源过滤目标。	<p>在 Filter 列表中选择过滤。</p> <p>可用的过滤选项如下：</p> <ul style="list-style-type: none"> • Status 过滤器： <ul style="list-style-type: none"> ◦ Up。目标当前已启动，正在主动抓取指标。 ◦ Down。目标当前已停机，没有提取指标。 • Source 过滤器： <ul style="list-style-type: none"> ◦ Platform。平台级别的目标仅与 AWS 默认项目的 Red Hat OpenShift Service 相关。这些项目提供 Red Hat OpenShift Service on AWS 核心功能。 ◦ User。用户目标与用户定义的项目相关。这些项目是用户创建的，可以进行自定义。
根据名称或标签搜索目标。	在搜索框旁边的 Text 或 Label 字段中输入搜索词。
对目标进行排序。	点一个或多个 Endpoint Status , Namespace , Last Scrape , 和 Scrape Duration 列标头。

3. 点目标的 **Endpoint** 列中的 URL，以进入其 **Target** 详情页。本页提供有关目标的信息，包括以下信息：

- 为指标提取的端点 URL
- 目标的当前 **Up** 或 **Down** 状态
- 到命名空间的链接
- 到 **ServiceMonitor** 资源详情的链接
- 附加到目标的标签
- 为指标提取目标的最长时间

5.1.4. 以集群管理员身份查看监控仪表盘

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盘。

先决条件

- 您可以使用具有 **cluster-admin** 集群角色的用户身份访问集群。

流程

1. 在 OpenShift Container Platform web 控制台的 **Administrator** 视角中，进入到 **Observe → Dashboards**。
2. 在 **Dashboard** 列表选择一个仪表盘。有些仪表盘（如 **etcd** 和 **Prometheus** 仪表盘）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。
 - 点 **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. 将鼠标悬停在仪表盘中的每个图形上，以显示特定项目的详细信息。

其他资源

- [关于监控仪表盘](#)

5.2. 以开发者身份访问指标

您可以访问指标来监控集群工作负载的性能。

其他资源

- [了解指标](#)

5.2.1. 查看可用指标列表

作为集群管理员或具有所有项目查看权限的用户，您可以查看集群中可用的指标列表，并以 JSON 格式输出列表。

先决条件

- 您是一个集群管理员，或者您可以使用具有 **cluster-monitoring-view** 集群角色的用户访问集群。
- 已安装 OpenShift Container Platform CLI (**oc**)。
- 您已获取 Thanos Querier 的 OpenShift Container Platform API 路由。

- 您可以使用 **oc whoami -t** 命令获取 bearer 令牌。



重要

您只能使用 bearer 令牌身份验证来访问 Thanos Querier API 路由。

流程

1. 如果您还没有为 Thanos Querier 获取 OpenShift Container Platform API 路由，请运行以下命令：

```
$ oc get routes -n openshift-monitoring thanos-querier -o jsonpath='{.status.ingress[0].host}'
```

2. 运行以下命令，以 JSON 格式从 Thanos Querier API 路由检索指标列表。此命令使用 **oc** 通过 bearer 令牌进行身份验证。

```
$ curl -k -H "Authorization: Bearer $(oc whoami -t)"  
https://<thanos_querier_route>/api/v1/metadata 1
```

- 1** 将 **<thanos_querier_route>** 替换为 Thanos Querier 的 OpenShift Container Platform API 路由。

5.2.2. 使用 OpenShift Container Platform Web 控制台查询用户定义的项目的指标

您可以使用 OpenShift Container Platform 指标查询浏览器来运行 Prometheus Query Language (PromQL) 查询来检查图表中呈现的指标。此功能提供有关您要监控的任何用户定义的工作负载的信息。

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。

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。

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. 从 **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** 根据所选查询创建自定义 PromQL 查询。图表中会直观呈现查询的指标。



注意

在 Developer 视角中，您一次只能运行一个查询。

- 4. 通过执行以下操作来探索视觉化的指标：

选项	描述
放大图表并更改时间范围。	任一： <ul style="list-style-type: none">● 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。
重置时间范围。	选择 Reset zoom 。
在特定时间点显示所有查询的输出。	将鼠标光标悬停在图表上。查询输出会出现在弹出窗口中。

其他资源

- [查询 Prometheus \(Prometheus 文档\)](#)

5.2.3. 以开发者身份查看监控仪表板

在 **Developer** 视角中，您可以查看与所选项目相关的仪表板。



注意

在 **Developer** 视角中，您一次只能查看一个项目的仪表板。

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:** 下拉列表选择一个项目。
- 3. 从 **Dashboard** 下拉列表选择一个仪表板，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。

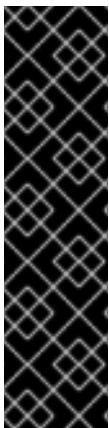
4. 可选：在 **Time Range** 列表中为图形选择一个时间范围。
 - 选择预定义的时间段。
 - 点 **Time Range** 列表中的 **Custom** 时间范围来设置自定义时间范围。
 - a. 输入或选择 **From** 和 **To** date and time。
 - b. 单击 **Save** 以保存自定义时间范围。
5. 可选：选择一个 **Refresh Interval**。
6. 将鼠标悬停在仪表板中的每个图形上，以显示特定项目的详细信息。

其他资源

- [关于监控仪表板](#)
- [使用 Developer 视角监控项目和应用程序的指标](#)

5.3. 使用 CLI 访问监控 API

在 OpenShift Container Platform 中，您可以从命令行界面(CLI)访问一些监控组件的 Web 服务 API。



重要

在某些情况下，访问 API 端点可能会降低集群的性能和可扩展性，特别是在使用端点来检索、发送或查询大量指标数据时。

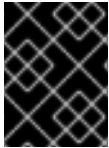
要避免这些问题，请考虑以下建议：

- 避免频繁查询端点。将查询限制为每 30 秒最多一个。
- 不要通过 Prometheus 的 **/federate** 端点检索所有指标数据。只有在您要检索有限、聚合的数据集时才会查询端点。例如，检索每个请求数量少于 1,000 个样本，有助于最大程度降低性能下降的风险。

5.3.1. 关于访问监控 Web 服务 API

您可以从命令行直接访问以下监控堆栈组件的 Web 服务 API 端点：

- Prometheus
- Alertmanager
- Thanos Ruler
- Thanos querier



重要

要访问 Thanos Ruler 和 Thanos Querier 服务 API，请求的帐户必须具有命名空间资源的 **get** 权限，这些资源可通过向帐户授予 **cluster-monitoring-view** 集群角色来完成。

当您访问监控组件的 Web 服务 API 端点时，请注意以下限制：

- 您只能使用 bearer 令牌身份验证来访问 API 端点。
- 您只能访问路由的 **/api** 路径中的端点。如果您试图在 Web 浏览器中访问 API 端点，则会出现一个 **Application is not available** 的错误。要在网页浏览器中访问监控功能，请使用 OpenShift Container Platform Web 控制台查看监控仪表板。

其他资源

- [以集群管理员身份查看监控仪表板](#)
- [以开发者身份查看监控仪表板](#)

5.3.2. 访问监控 Web 服务 API

以下示例演示了如何查询服务 API 接收器以获取核心平台监控中使用的 Alertmanager 服务。您可以使用类似的方法访问核心平台 Prometheus 的 **prometheus-k8s** 服务，以及 Thanos Ruler 的 **thanos-ruler** 服务。

先决条件

- 您已登录到与 **openshift-monitoring** 命名空间中的 **monitoring-alertmanager-edit** 角色绑定的帐户。
- 已登录到一个有权获取 Alertmanager API 路由的帐户。



注意

如果您的帐户没有获取 Alertmanager API 路由的权限，集群管理员可以提供路由的 URL。

流程

1. 运行以下命令来提取身份验证令牌：

```
$ TOKEN=$(oc whoami -t)
```

2. 运行以下命令提取 **alertmanager-main** API 路由 URL：

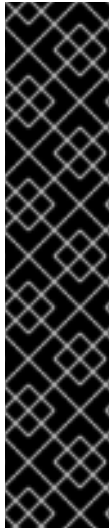
```
$ HOST=$(oc -n openshift-monitoring get route alertmanager-main -ojsonpath='{.status.ingress[].host}')
```

3. 运行以下命令，查询 Alertmanager 的服务 API 接收器：

```
$ curl -H "Authorization: Bearer $TOKEN" -k "https://$HOST/api/v2/receivers"
```

5.3.3. 使用 Prometheus 的联邦端点查询指标

您可以使用 Prometheus 的联邦端点从集群外的网络位置提取平台和用户定义的指标。为此，请通过 OpenShift Container Platform 路由访问集群的 Prometheus **/federate** 端点。



重要

使用联邦时检索指标数据的延迟。这个延迟可能会影响提取指标的准确性和时间表。

使用联邦端点也可以降低集群的性能和可扩展性，特别是在使用联邦端点来获取大量指标数据时。要避免这些问题，请遵循以下建议：

- 不要尝试通过 Prometheus 的联邦端点检索所有指标数据。只有在您要检索有限、聚合的数据集时才会查询。例如，检索每个请求数量少于 1,000 个样本，有助于最大程度降低性能下降的风险。
- 避免频繁查询 Prometheus 的联邦端点。将查询限制为每 30 秒最多一个。

如果您需要在集群外转发大量数据，请使用远程写入。如需更多信息，请参阅 [配置远程写入存储](#)部分。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-monitoring-view** 集群角色的用户，或者获取了对 **命名空间** 资源具有 **get** 权限的 bearer 令牌来访问集群。



注意

您只能使用 bearer 令牌身份验证来访问 Prometheus 联邦端点。

- 已登陆到一个有权获取 Prometheus 联邦路由的帐户。



注意

如果您的帐户没有获取 Prometheus 联邦路由的权限，集群管理员可以提供路由的 URL。

流程

1. 运行以下命令来检索 bearer 令牌：

```
$ TOKEN=$(oc whoami -t)
```

2. 运行以下命令来获取 Prometheus 联邦路由 URL：

```
$ HOST=$(oc -n openshift-monitoring get route prometheus-k8s-federate -ojsonpath='{.status.ingress[].host}')
```

3. 查询 **/federate** 路由的指标。以下示例命令 **up** 指标：

```
$ curl -G -k -H "Authorization: Bearer $TOKEN" https://$HOST/federate --data-urlencode 'match[]=up'
```

输出示例

```
# TYPE up untyped
up{apiserver="kube-
apiserver",endpoint="https",instance="10.0.143.148:6443",job="apiserver",namespace="default
",service="kubernetes",prometheus="openshift-
monitoring/k8s",prometheus_replica="prometheus-k8s-0"} 1 1657035322214
up{apiserver="kube-
apiserver",endpoint="https",instance="10.0.148.166:6443",job="apiserver",namespace="default
",service="kubernetes",prometheus="openshift-
monitoring/k8s",prometheus_replica="prometheus-k8s-0"} 1 1657035338597
up{apiserver="kube-
apiserver",endpoint="https",instance="10.0.173.16:6443",job="apiserver",namespace="default",
service="kubernetes",prometheus="openshift-
monitoring/k8s",prometheus_replica="prometheus-k8s-0"} 1 1657035343834
...
```

5.3.4. 从集群外部访问自定义应用程序的指标

在使用用户定义的项目监控您自己的服务时，您可以从集群外部查询 Prometheus 指标。使用 **thanos-querier** 路由从集群外部访问这些数据。

此访问仅支持使用 bearer 令牌进行身份验证。

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。
- 您可以使用 **cluster-monitoring-view** 集群角色登录到帐户，它提供访问 Thanos Querier API 的权限。
- 已登陆到一个有权获取 Thanos Querier API 路由的帐户。



注意

如果您的帐户没有权限获取 Thanos Querier API 路由，集群管理员可以提供路由的 URL。

流程

1. 运行以下命令，提取身份验证令牌以连接到 Prometheus：

```
$ TOKEN=$(oc whoami -t)
```

2. 运行以下命令，提取 **thanos-querier** API 路由 URL：

```
$ HOST=$(oc -n openshift-monitoring get route thanos-querier -
ojsonpath='{.status.ingress[].host}')
```

3. 使用以下命令，将命名空间设置为运行服务的命名空间：

```
$ NAMESPACE=ns1
```

4. 运行以下命令，在命令行中查询您自己的服务的指标：

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

输出显示 Prometheus 提取的每个应用程序 pod 的状态：

格式的示例输出

```
{
  "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"
        ]
      }
    ]
  }
}
```



注意

- 格式的示例输出使用过滤工具（如 **jq**）来提供格式化的缩进 JSON。有关使用 **jq** 的更多信息，请参阅 [jq Manual](#)。
- 命令会请求 Thanos Querier 服务的一个即时查询端点，它会在某个时间点评估选择器。

5.3.5. 其他资源

- [为用户定义的项目启用监控](#)
- [为核心平台监控配置远程写入存储](#)
- [配置远程写入存储来监控用户定义的项目](#)
- [以管理员身份访问指标](#)
- [以开发者身份访问指标](#)

- [以管理员身份管理警报](#)
- [将警报作为开发者管理](#)

第 6 章 管理警报

6.1. 以管理员身份管理警报

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



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，如果您以具有 **cluster-admin** 角色的用户身份登录，您可以访问所有警报、静默和警报规则。

6.1.1. 从 Administrator 视角访问 Alerting UI

Alerting UI 可通过 OpenShift Container Platform Web 控制台的 **Administrator** 视角访问。

- 从 **Administrator** 视角中，进入 **Observe → Alerting**。此视角中的 Alerting UI 中的三个主要页面是 **Alerts**、**Silences** 和 **Alerting 规则** 页面。

其他资源

- [搜索和过滤警报、静默和警报规则](#)

6.1.2. 从 Administrator 视角获取警报、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

- 您可以使用具有查看警报的项目查看权限的用户访问集群。

流程

要获取有关警报的信息：

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **Observe → Alerting → Alerts** 页。
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. 点警报的名称查看其 **Alert details** 页面。该页面包含一个说明警报时间序列数据的图形。它还提供有关警报的以下信息：
 - 警报的描述
 - 与警报关联的消息
 - 如果存在该警报的 GitHub 上 runbook 页的链接
 - 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）

要获取有关静默的信息：

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **Observe → Alerting → Silences** 页。
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：点一个或多个 **Name**、**Firing Alerts**、**State** 和 **Creator** 栏上面的标头来对静默进行排序。
5. 选择静默的名称来查看其 **Silence 详情页面**。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

要获取有关警报规则的信息：

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **Observe → Alerting → Alerting rules** 页。
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 **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来查看其 **Alerting 规则详情页面**。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。
 - 定义触发警报的条件的表达式。
 - 触发警报的条件应满足的时间。
 - 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

其他资源

- [GitHub Cluster Monitoring Operator runbooks 仓库](#)

6.1.3. 管理静默

您可以在 **Administrator** 视角中的 OpenShift Container Platform Web 控制台中为警报创建静默。在创建静默后，您可以查看、编辑这个静默，并可以使其过期。您还不会在警报触发时收到有关静默警报的通知。



注意

在创建静默时，它们会在 Alertmanager pod 之间复制。但是，如果您没有为 Alertmanager 配置持久性存储，静默可能会丢失。例如，如果所有 Alertmanager pod 同时重启，会出现这种情况。

其他资源

- [管理静默](#)
- [配置持久性存储](#)

6.1.3.1. 从 Administrator 视角静默警报

您可以静默特定的警报或静默与您定义的规格匹配的警报。


先决条件

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

流程

静默特定的警报：

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角，进入 **Observe → Alerting → Alerts**。

2. 对于您要静默的警报，点  并选择 **Silence alert** 以打开 **Silence** 警报 页，其中包含所选警报的默认配置。
3. 可选：更改静默的默认配置详情。



注意

在保存静默前，需要添加一个注释信息。

4. 要保存静默，点 **Silence**。

静默一组警报：

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角，进入 **Observe → Alerting → Silences**。
2. 点 **Create silence**。
3. 在 **Create silence** 页面中，设置警报的调度、持续时间和标签详情。



注意

在保存静默前，需要添加一个注释信息。

4. 要为与您输入的标签匹配的警报创建静默，请点 **Silence**。

6.1.3.2. 从 Administrator 视角编辑静默

您可以编辑一个静默，这会使现有静默到期，然后创建一个带有相关配置变化的静默。


先决条件

- 如果您是集群管理员，可以使用具有 **cluster-admin** 角色的用户访问集群。
- 如果您是非管理员用户，您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

流程

1. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角，进入 **Observe → Alerting → Silences**。



2. 针对您想要修改的静默，点  并选择 **Edit silence**。
或者，您可以点 **Actions**，然后在静默的 **Silence details** 页面中选择 **Edit silence**。
3. 在 **Edit silence** 页面中，进行更改并点 **Silence**。这样做会使现有静默到期，并创建带有更新的配置。

6.1.3.3. 从 Administrator 视角使静默到期

您可以使单个静默或多个静默到期。使静默过期的效果是，永久取消激活它。



注意

您无法删除已过期、静默的警报。超过 120 小时的过期的静默会被垃圾收集。

先决条件

- 如果您是集群管理员，可以使用具有 **cluster-admin** 角色的用户访问集群。
- 如果您是非管理员用户，您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

流程

1. 进入 **Observe → Alerting → Silences**。
2. 对于您要过期的静默，选择对应行的复选框。
3. 点 **Expire 1 silence** 使一个静默过期，或点 **Expire <n> silences** 使多个静默过期（其中 <n> 是你选择的静默数）。
另外，要使单个静默到期，您还可以点 **Actions**，然后在静默的 **Silence details** 页面中选择 **Expire silence**。

6.1.4. 管理用于核心平台监控的警报规则

OpenShift Container Platform 监控包括一组用于平台指标的默认警报规则。作为集群管理员，您可以以两种方式自定义这组规则：

- 通过调整阈值或添加和修改标签来修改现有平台警报规则的设置。例如，您可以将一个警报的 **severity** 标签从 **warning** 改为 **critical** 以帮助您来处理相关的问题。
- 通过基于 **openshift-monitoring** 项目中的核心平台指标构建查询表达式来定义和添加新的自定义警报规则。

其他资源

- [管理用于核心平台监控的警报规则](#)
- [为核心平台监控优化警报规则的建议](#)

6.1.4.1. 创建新警报规则

作为集群管理员，您可以根据平台指标创建新的警报规则。这些警报规则根据所选指标的值触发警报。



注意

- 如果您基于现有平台警报规则创建自定义 **AlertingRule** 资源，请静默原始警报以避免收到冲突的警报。
- 为了帮助用户了解警报的影响和原因，请确保您的警报规则包含警报消息和严重性值。

先决条件

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

流程

1. 创建一个名为 **example-alerting-rule.yaml** 的新 YAML 配置文件。
2. 向 YAML 文件添加 **AlertingRule** 资源。以下示例创建一个名为 **example** 的新警报规则，类似于默认的 **Watchdog** 警报：

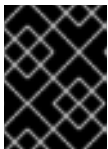
```
apiVersion: monitoring.openshift.io/v1
kind: AlertingRule
metadata:
```

```

name: example
namespace: openshift-monitoring ❶
spec:
  groups:
  - name: example-rules
    rules:
    - alert: ExampleAlert ❷
      for: 1m ❸
      expr: vector(1) ❹
      labels:
        severity: warning ❺
      annotations:
        message: This is an example alert. ❻

```

- ❶ 确保命名空间为 **openshift-monitoring**。
- ❷ 您要创建的警报规则的名称。
- ❸ 触发警报前条件应为 **true** 的持续时间。
- ❹ 定义新规则的 PromQL 查询表达式。
- ❺ 警报规则分配给警报的严重性。
- ❻ 与警报关联的消息。



重要

您必须在 **openshift-monitoring** 命名空间中创建 **AlertingRule** 对象。否则，将不接受警报规则。

3. 将配置文件应用到集群：

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

6.1.4.2. 修改核心平台警报规则

作为集群管理员，您可以在 Alertmanager 将核心平台警报路由到接收器前修改核心平台警报。例如，您可以更改警报的严重性标签，添加自定义标签，或者从发送到 Alertmanager 中排除警报。

先决条件

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

流程

1. 创建名为 **example-modified-alerting-rule.yaml** 的新 YAML 配置文件。
2. 在 YAML 文件中添加 **AlertRelabelConfig** 资源。以下示例将默认平台 **watchdog** 警报规则的 **severity** 设置改为 **critical**：

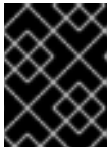
—

```

apiVersion: monitoring.openshift.io/v1
kind: AlertRelabelConfig
metadata:
  name: watchdog
  namespace: openshift-monitoring ❶
spec:
  configs:
    - sourceLabels: [alertname,severity] ❷
      regex: "Watchdog;none" ❸
      targetLabel: severity ❹
      replacement: critical ❺
      action: Replace ❻

```

- ❶ 确保命名空间为 **openshift-monitoring**。
- ❷ 要修改的值的源标签。
- ❸ 匹配的 **sourceLabels** 值的正则表达式。
- ❹ 要修改的值的目标标签。
- ❺ 要替换目标标签的新值。
- ❻ 基于正则表达式匹配替换旧值的重新标记操作。默认操作为 **Replace**。其他可能的值有 **Keep,Drop,HashMod,LabelMap,LabelDrop, LabelKeep**。



重要

您必须在 **openshift-monitoring** 命名空间中创建 **AlertRelabelConfig** 对象。否则，警报标签不会改变。

3. 将配置文件应用到集群：

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

其他资源

- [监控堆栈架构](#)
- [Alertmanager \(Prometheus 文档\)](#)
- [relabel_config \(Prometheus 文档\)](#)
- [警报\(Prometheus 文档\)](#)

6.1.5. 为用户定义的项目管理警报规则

在 OpenShift Container Platform 中，您可以为用户定义的项目创建、查看、编辑和删除警报规则。这些警报规则将根据所选指标的值触发警报。

其他资源

- [为用户定义的项目创建警报规则](#)
- [为用户定义的项目管理警报规则](#)
- [为用户定义的项目优化警报](#)

6.1.5.1. 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。



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。
- 为了帮助用户了解警报的影响和原因，请确保您的警报规则包含警报消息和严重性值。

先决条件

- 您已为用户定义的项目启用了监控。
- 您以集群管理员身份登录，或作为具有用于创建警报规则的项目的 **monitoring-rules-edit** 集群角色的用户登录。
- 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。以下示例创建一个名为 **example-alert** 的新警报规则。当示例服务公开的 **version** 指标变为 **0** 时，警报规则会触发警报：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert ❶
      for: 1m ❷
      expr: version{job="prometheus-example-app"} == 0 ❸
      labels:
        severity: warning ❹
      annotations:
        message: This is an example alert. ❺
```

- ❶ 您要创建的警报规则的名称。
- ❷ 触发警报前条件应为 **true** 的持续时间。

- 3 定义新规则的 PromQL 查询表达式。
- 4 警报规则分配给警报的严重性。
- 5 与警报关联的消息。

3. 将配置文件应用到集群：

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

其他资源

- [监控堆栈架构](#)
- [警报\(Prometheus 文档\)](#)

6.1.5.2. 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. 从 OpenShift Container Platform Web 控制台的 **Administrator** 视角，进入 **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。

6.1.5.3. 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。
- 您以集群管理员身份登录，或作为具有用于创建警报规则的项目的 **monitoring-rules-edit** 集群角色的用户登录。
- 已安装 OpenShift CLI(**oc**)。

流程

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

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

其他资源

- [Alertmanager \(Prometheus 文档\)](#)

6.2. 将警报作为开发者管理

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



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。

6.2.1. 从 Developer 视角访问 Alerting UI

Alerting UI 可通过 OpenShift Container Platform Web 控制台的 **Developer** 视角访问。

- 从 **Developer** 视角中，进入 **Observe** 并进入 **Alerts** 选项卡。
- 从 **Project:** 列表中选择您要管理警报的项目。

在这个视角中，警报、静默和警报规则都通过 **Alerts** 标签页管理。**Alerts** 选项卡中显示的结果特定于所选项目。



注意

在 **Developer** 视角中，您可以从可在 **Project: <project_name>** 列表中访问的核心 OpenShift Container Platform 和用户定义的项目中选择。但是，如果您没有以集群管理员身份登录，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。

其他资源

- [搜索和过滤警报、静默和警报规则](#)

6.2.2. 从 Developer 视角获取警报、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

- 您可以使用具有查看警报的项目查看权限的用户访问集群。

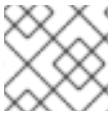
流程

要获取有关警报、静默和警报规则的信息：

1. 从 OpenShift Container Platform Web 控制台的 **Developer** 视角中，进入 **Observe** → **<project_name>** → **Alerts** 页。
2. 查看警报、静默或警报规则的详情：

- 可以通过在警报名称旁边点大于符号(>)来查看**警报详情**，然后从列表中选择警报。
- 可以通过在 **Alert details** 页面的 **Silenced by** 部分点静默详情来查看**静默详情**。**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

- 点 **Alerts** 页中的警报旁的  菜单来查看 **警报规则详情**，然后点 **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。

其他资源

- [GitHub Cluster Monitoring Operator runbooks 仓库](#)

6.2.3. 管理静默

您可以在 **Developer** 视角中为 OpenShift Container Platform Web 控制台中的警报创建静默。在创建静默后，您可以查看、编辑这个静默，并可以使其过期。您还不会在警报触发时收到有关静默警报的通知。



注意

在创建静默时，它们会在 Alertmanager pod 之间复制。但是，如果您没有为 Alertmanager 配置持久性存储，静默可能会丢失。例如，如果所有 Alertmanager pod 同时重启，会出现这种情况。

其他资源

- [管理静默](#)
- [配置持久性存储](#)

6.2.3.1. 从 Developer 视角静默警报

您可以静默特定的警报或静默与您定义的规格匹配的警报。

先决条件

- 如果您是集群管理员，可以使用具有 **cluster-admin** 角色的用户访问集群。
- 如果您是非管理员用户，您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** 集群角色，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：

1. 从 OpenShift Container Platform Web 控制台的 **Developer** 视角，进入 **Observe** 并进入 **Alerts** 选项卡。
2. 从 **Project:** 列表中选择您要静默警报的项目。
3. 如有必要，点警报名称旁边的大于符号(>)来扩展警报的详情。
4. 在扩展视图中点警报消息以打开警报的 **Alert details** 页面。
5. 点 **Silence alert**，打开带有警报默认配置的**静默警报**页。
6. 可选：更改静默的默认配置详情。



注意

在保存静默前，需要添加一个注释信息。

7. 要保存静默，点 **Silence**。

静默一组警报：

1. 从 OpenShift Container Platform Web 控制台的 **Developer** 视角，进入 **Observe** 并进入 **Silences** 选项卡。
2. 从 **Project:** 列表中选择您要静默警报的项目。
3. 点 **Create silence**。
4. 在 **Create silence** 页面中，设置警报的持续时间和标签详情。



注意

在保存静默前，需要添加一个注释信息。

5. 要为与您输入的标签匹配的警报创建静默，请点 **Silence**。

6.2.3.2. 从 Developer 视角编辑静默

您可以编辑一个静默，这会使现有静默到期，然后创建一个带有相关配置变化的静默。

先决条件

- 如果您是集群管理员，可以使用具有 **cluster-admin** 角色的用户访问集群。
- 如果您是非管理员用户，您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-rules-edit** 集群角色，允许您在 web 控制台的 **Developer** 视角中创建和静默警报。

流程

1. 从 OpenShift Container Platform Web 控制台的 **Developer** 视角，进入 **Observe** 并进入 **Silences** 选项卡。
2. 从 **Project:** 列表中选择您要编辑静默的项目。
3. 针对您想要修改的静默，点  并选择 **Edit silence**。
或者，您可以点 **Actions**，然后在静默的 **Silence details** 页面中选择 **Edit silence**。
4. 在 **Edit silence** 页面中，进行更改并点 **Silence**。这样做会使现有静默到期，并创建带有更新的配置。

6.2.3.3. 从 Developer 视角使静默到期

您可以使单个静默或多个静默到期。使静默过期的效果是，永久取消激活它。



注意

您无法删除已过期、静默的警报。超过 120 小时的过期的静默会被垃圾收集。

先决条件

- 如果您是集群管理员，可以使用具有 **cluster-admin** 角色的用户访问集群。
- 如果您是非管理员用户，您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-rules-edit** 集群角色，允许您在 web 控制台的 **Developer** 视角中创建和静默警报。

流程

1. 从 OpenShift Container Platform Web 控制台的 **Developer** 视角，进入 **Observe** 并进入 **Silences** 选项卡。
2. 从 **Project:** 列表中选择您要使静默到期的项目。
3. 对于您要过期的静默，选择对应行的复选框。
4. 点 **Expire 1 silence** 使一个静默过期，或点 **Expire <n> silences** 使多个静默过期（其中 <n> 是您选择的静默数）。
另外，要使单个静默到期，您还可以点 **Actions**，然后在静默的 **Silence details** 页面中选择 **Expire silence**。

6.2.4. 为用户定义的项目管理警报规则

在 OpenShift Container Platform 中，您可以为用户定义的项目创建、查看、编辑和删除警报规则。这些警报规则将根据所选指标的值触发警报。

其他资源

- [为用户定义的项目创建警报规则](#)
- [为用户定义的项目管理警报规则](#)
- [为用户定义的项目优化警报](#)

6.2.4.1. 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。



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。
- 为了帮助用户了解警报的影响和原因，请确保您的警报规则包含警报消息和严重性值。

先决条件

- 您已为用户定义的项目启用了监控。
- 您以集群管理员身份登录，或作为具有用于创建警报规则的项目的 **monitoring-rules-edit** 集群角色的用户登录。
- 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。以下示例创建一个名为 **example-alert** 的新警报规则。当示例服务公开的 **version** 指标变为 **0** 时，警报规则会触发警报：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert 1
      for: 1m 2
      expr: version{job="prometheus-example-app"} == 0 3
      labels:
```

```
severity: warning 4
annotations:
  message: This is an example alert. 5
```

- 1 您要创建的警报规则的名称。
- 2 触发警报前条件应为 true 的持续时间。
- 3 定义新规则的 PromQL 查询表达式。
- 4 警报规则分配给警报的严重性。
- 5 与警报关联的消息。

3. 将配置文件应用到集群：

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

其他资源

- [监控堆栈架构](#)
- [警报\(Prometheus 文档\)](#)

6.2.4.2. 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** 集群角色。

先决条件

- 您已为用户定义的项目启用了监控。
- 您以具有项目的 **monitoring-rules-view** 集群角色的用户身份登录。
- 已安装 OpenShift CLI (**oc**)。

流程

1. 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
```

6.2.4.3. 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。
- 您以集群管理员身份登录，或作为具有用于创建警报规则的项目的 **monitoring-rules-edit** 集群角色的用户登录。
- 已安装 OpenShift CLI(**oc**)。

流程

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

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

其他资源

- [Alertmanager \(Prometheus 文档\)](#)

第 7 章 监控问题的故障排除

查找核心平台和用户定义的项目监控常见问题的故障排除步骤。

7.1. 调查用户定义的项目指标不可用的原因

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**)。
- 您已为用户定义的项目启用并配置了监控。
- 您已创建了 **ServiceMonitor** 资源。

流程

1. 确保您的项目没有从用户工作负载监控中被排除。以下示例使用 **ns1** 项目。

- a. 验证项目没有附加 **openshift.io/user-monitoring=false** 标签：

```
$ oc get namespace ns1 --show-labels | grep 'openshift.io/user-monitoring=false'
```



注意

用户工作负载项目设置的默认标签为 **openshift.io/user-monitoring=true**。但是，除非手动应用了该标签，否则标签不可见。

- b. 如果附加了标签，请删除该标签：

从项目中删除该标签的示例

```
$ oc label namespace ns1 'openshift.io/user-monitoring-'
```

输出示例

```
namespace/ns1 unlabeled
```

2. 在服务和 **ServiceMonitor** 资源配置中，检查对应的标签是否匹配。以下示例使用 **prometheus-example-app** 服务、**prometheus-example-monitor** 服务监控和 **ns1** 项目。

- a. 获取服务中定义的标签。

```
$ 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** 定义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

```
apiVersion: v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
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。

3. 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-776fcbdd56-2nbfm	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-776fcbdd56-2nbfm** :

```
$ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbdd56-2nbfm -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
```

4. 在 OpenShift Container Platform Web 控制台 UI 中的 **Metrics 目标** 页中查看您的端点的目标状态。
 - a. 登录到 OpenShift Container Platform web 控制台，进入 **Administrator** 视角中的 **Observe → Targets**。
 - b. 在列表中找到指标端点，并在 **Status** 列中查看目标的状态。
 - c. 如果 **Status** 为 **Down**，点端点的 URL 查看该指标目标的 **Target Details** 页面的更多信息。
5. 在 **openshift-user-workload-monitoring** 项目中为 Prometheus Operator 配置 debug 级别日志记录。
 - 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。受影响的 **prometheus-operator** Pod 会自动重新部署。
 - 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

- [为用户定义的项目启用监控](#)
- [指定如何监控服务](#)
- [获取有关指标目标的详细信息](#)

7.2. 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：

- 使用 Prometheus HTTP API 检查时间序列数据库(TSDB)状态，以了解有关哪些标签创建最多时间序列数据的更多信息。这样做需要集群管理员特权。
- 检查正在收集的提取示例数量。
- 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** 集群角色的用户身份访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 在 **Administrator** 视角中，进入到 **Observe → Metrics**。
2. 在 **Expression** 字段中输入 Prometheus Query Language (PromQL) 查询。以下示例查询有助于识别可能导致高磁盘空间消耗的高卡性指标：
 - 通过运行以下查询，您可以识别具有最高提取示例数的十个作业：

```
topk(10, max by(namespace, job) (topk by(namespace, job) (1,
scrape_samples_post_metric_relabeling)))
```

- 通过运行以下查询，您可以通过识别在上一小时内创建了最多时间序列数据的十个作业，从而找出相关的时间序列：

```
topk(10, sum by(namespace, job) (sum_over_time(scrape_series_added[1h])))
```

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. 以集群管理员身份登录时，按照以下步骤使用 Prometheus HTTP API 查看 TSDB 状态：

- a. 运行以下命令来获取 Prometheus API 路由 URL：

```
$ HOST=$(oc -n openshift-monitoring get route prometheus-k8s -
ojsonpath='{.status.ingress[].host}')
```

- b. 运行以下命令来提取身份验证令牌：

```
$ TOKEN=$(oc whoami -t)
```

- c. 运行以下命令，查询 Prometheus 的 TSDB 状态：

```
$ curl -H "Authorization: Bearer $TOKEN" -k "https://$HOST/api/v1/status/tsdb"
```

输出示例

```
"status": "success", "data": {"headStats": {"numSeries": 507473,
"numLabelPairs": 19832, "chunkCount": 946298, "minTime": 1712253600010,
"maxTime": 1712257935346}, "seriesCountByMetricName":
[{"name": "etcd_request_duration_seconds_bucket", "value": 51840},
{"name": "apiserver_request_sli_duration_seconds_bucket", "value": 47718},
...]
```

其他资源

- [使用 CLI 访问监控 API](#)
- [为用户定义的项目设置提取示例限制](#)
- [提交支持问题单](#)

7.3. 解决 PROMETHEUS 的 KUBEPERSISTENTVOLUMEFILLINGUP 警报触发的問題

作为集群管理员，您可以解析 Prometheus 触发的 **KubePersistentVolumeFillingUp** 警报。

当 **openshift-monitoring** 项目中的 **prometheus-k8s-*** pod 声明的持久性卷 (PV) 时，关键警报会在剩余的总空间少于 3% 时触发。这可能导致 Prometheus 正常工作。



注意

有两个 **KubePersistentVolumeFillingUp** 警报：

- **Critical 警报**：当挂载的 PV 小于 3% 的总空间时，会触发具有 **severity="critical"** 标签的警报。
- **Warning 警报**：当挂载的 PV 的总空间低于 15% 时，会触发带有 **severity="warning"** 标签的警报，且预期在四天内填满。

要解决这个问题，您可以删除 Prometheus 时间序列数据库 (TSDB) 块来为 PV 创建更多空间。

先决条件

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

流程

1. 运行以下命令，列出所有 TSDB 块的大小，从最旧的到最新排序：

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \
-c prometheus --image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') \
-- sh -c 'cd /prometheus/;du -hs $(ls -dtr */ | grep -Eo "[0-9|A-Z]{26}")'
```

- 1 2 将 **<prometheus_k8s_pod_name>** 替换为 **KubePersistentVolumeFillingUp** 警报描述中提到的 pod。

输出示例

```
308M 01HVKMPKQWZYWS8WVDAYQHNMW6
52M 01HVK64DTDA81799TBR9QDECEZ
102M 01HVK64DS7TRZRW2756KHST5X
140M 01HVJS59K11FBVAPVY57K88Z11
90M 01HVVH2A5Z58SKT810EM6B9AT50
152M 01HV8ZDVQMX41MKCN84S32RRZ1
354M 01HV6Q2N26BK63G4RYTST71FBF
156M 01HV664H9J9Z1FTZD73RD1563E
216M 01HTHXB60A7F239HN7S2TENPNS
104M 01HTHMGRXGS0WXA3WATRXHR36B
```

2. 确定可以删除哪些块以及多少块，然后删除块。以下示例命令从 **prometheus-k8s-0** pod 中删除三个最旧的 Prometheus TSDB 块：

```
$ oc debug prometheus-k8s-0 -n openshift-monitoring \
-c prometheus --image=$(oc get po -n openshift-monitoring prometheus-k8s-0 \
```

```
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}')) \
-- sh -c 'ls -latr /prometheus/ | egrep -o "[0-9A-Z]{26}" | head -3 | \
while read BLOCK; do rm -r /prometheus/$BLOCK; done'
```

3. 运行以下命令，验证挂载的 PV 的使用并确保有足够的可用空间：

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \ 1
--image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \ 2
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') -- df -h /prometheus/
```

1 2 将 **<prometheus_k8s_pod_name>** 替换为 **KubePersistentVolumeFillingUp** 警报描述中提到的 pod。

以下示例显示了由 **prometheus-k8s-0** pod 声明的挂载的 PV，该 pod 剩余 63%：

输出示例

```
Starting pod/prometheus-k8s-0-debug-j82w4 ...
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1p4 40G   15G  40G   37% /prometheus

Removing debug pod ...
```

7.4. 解决 ALERTMANAGERRECEIVERSNOTCONFIGURED 警报

部署的每个集群都会默认触发 **AlertmanagerReceiversNotConfigured** 警报。要解决这个问题，您必须配置警报接收器。

- 对于默认平台监控，请按照 [配置核心平台监控](#) 中的“配置警报通知”中的步骤操作。
- 对于用户工作负载监控，请按照 [配置核心平台监控](#) 中的“配置警报通知”中的步骤操作。

其他资源

- [为默认平台监控配置警报通知](#)
- [为用户工作负载监控配置警报通知](#)

第 8 章 CLUSTER MONITORING OPERATOR 的配置映射引用

8.1. CLUSTER MONITORING OPERATOR 配置参考

OpenShift Container Platform 集群监控的一部分是可配置的。该 API 可通过设置各种配置映射中定义的参数来访问。

- 要配置监控组件，请编辑 `openshift-monitoring` 命名空间中的名为 `cluster-monitoring-config` 的 `ConfigMap` 对象。这些配置由 [ClusterMonitoringConfiguration](#) 定义。
- 要配置用于监控用户定义的项目的监控组件，请编辑 `openshift-user-workload-monitoring` 命名空间中名为 `user-workload-monitoring-config` 的 `ConfigMap` 对象。这些配置由 [UserWorkloadConfiguration](#) 定义。

配置文件始终在配置映射数据的 `config.yaml` 键下定义。



重要

- 并非所有监控堆栈的配置参数都会被公开。只有此引用中列出的参数和字段才支持进行配置。有关支持的配置的更多信息，请参阅
- [对监控的维护和支持](#)
- 配置集群监控是可选的。
- 如果配置不存在或为空，则使用默认值。
- 如果配置无效 YAML 数据，Cluster Monitoring Operator 会在 Operator 状态条件中停止协调资源并报告 **Degraded=True**。

8.2. ADDITIONALALERTMANAGERCONFIG

8.2.1. 描述

`AdditionalAlertmanagerConfig` 资源定义组件如何与其他 Alertmanager 实例通信的设置。

8.2.2. 必需

- `apiVersion`

会出现在：[PrometheusK8sConfig](#)、[PrometheusRestrictedConfig](#)、[ThanosRulerConfig](#)

属性	类型	描述
<code>apiVersion</code>	字符串	定义 Alertmanager 的 API 版本。可能的值有 v1 或 v2 。默认值为 v2 。
<code>bearerToken</code>	<code>*v1.SecretKeySelector</code>	定义包含 Alertmanager 身份验证时要使用的 bearer 令牌的 secret 密钥引用。

属性	类型	描述
pathPrefix	字符串	定义要在推送端点路径前面添加的路径前缀。
scheme	字符串	定义与 Alertmanager 实例通信时要使用的 URL 方案。可能的值有 http 或 https 。默认值为 http 。
staticConfigs	[]string	以 <hosts>:<port> 的形式静态配置的 Alertmanager 端点列表。
timeout	*string	定义发送警报时使用的超时值。
tlsConfig	TLSConfig	定义用于 Alertmanager 连接的 TLS 设置。

8.3. ALERTMANAGERMAINCONFIG

8.3.1. 描述

AlertmanagerMainConfig 资源定义 **openshift-monitoring** 命名空间中的 Alertmanager 组件的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
enabled	*bool	此布尔值标志，用于启用或禁用 openshift-monitoring 命名空间中的主 Alertmanager 实例。默认值为 true 。
enableUserAlertmanagerConfig	bool	一个布尔值标志，用于启用或禁用要用于 AlertmanagerConfig 查找的用户定义命名空间。只有在未启用 Alertmanager 的用户工作负载监控实例时，此设置才适用。默认值为 false 。
logLevel	字符串	定义 Alertmanager 的日志级别设置。可能的值有： error,warn,info,debug 。默认值为 info 。
nodeSelector	map[string]string	定义 Pod 被调度到的节点。
资源	*v1.ResourceRequirements	为 Alertmanager 容器定义资源请求和限值。

属性	类型	描述
secrets	[]string	定义要挂载到 Alertmanager 中的 secret 列表。secret 必须位于与 Alertmanager 对象相同的命名空间中。它们作为名为 secret- <secret-name> 的卷添加，并挂载到 Alertmanager pod 的 alertmanager 容器中的 /etc/alertmanager/secrets/<secret-name> 。
容忍（tolerations）	[]v1.Toleration	为 pod 定义容忍。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	为 Alertmanager 定义持久性存储。使用这个设置配置持久性卷声明，包括存储类、卷大小和名称。

8.4. ALERTMANAGERUSERWORKLOADCONFIG

8.4.1. 描述

AlertmanagerUserWorkloadConfig 资源定义用于用户定义的项目的 Alertmanager 实例的设置。

出现在：[UserWorkloadConfiguration](#)

属性	类型	描述
enabled	bool	此布尔值标志，用于启用或禁用 openshift-user-workload-monitoring 命名空间中用户定义的警报的 Alertmanager 专用实例。默认值为 false 。
enableAlertmanagerConfig	bool	为 AlertmanagerConfig 查找启用或禁用用户定义的命名空间的布尔值标志。默认值为 false 。
logLevel	字符串	定义 Alertmanager 用户工作负载监控的日志级别设置。可能的值有 error 、 warn 、 info 和 debug 。默认值为 info 。
资源	*v1.ResourceRequirements	为 Alertmanager 容器定义资源请求和限值。

属性	类型	描述
secrets	[]string	定义要挂载到 Alertmanager 中的 secret 列表。secret 必须位于与 Alertmanager 对象相同的命名空间中。它们作为名为 secret-<secret-name> 的卷添加，并挂载到 Alertmanager pod 的 alertmanager 容器中的 /etc/alertmanager/secrets/<secret-name> 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
容限（tolerations）	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	为 Alertmanager 定义持久性存储。使用这个设置配置持久性卷声明，包括存储类、卷大小和名称。

8.5. CLUSTERMONITORINGCONFIGURATION

8.5.1. 描述

ClusterMonitoringConfiguration 资源定义通过 **openshift-monitoring** 命名空间中的 **cluster-monitoring-config** 配置映射自定义默认平台监控堆栈的设置。

属性	类型	描述
alertmanagerMain	* AlertmanagerMainConfig	AlertmanagerMainConfig 定义了 openshift-monitoring 命名空间中的 Alertmanager 组件的设置。
enableUserWorkload	*bool	UserWorkloadEnabled 是一个布尔值标志，可为用户定义的项目启用监控。
k8sPrometheusAdapter	* K8sPrometheusAdapter	K8sPrometheusAdapter 为 Prometheus Adapter 组件定义设置。
kubeStateMetrics	* KubeStateMetricsConfig	KubeStateMetricsConfig 定义 kube-state-metrics 代理的设置。

属性	类型	描述
prometheusK8s	*PrometheusK8sConfig	PrometheusK8sConfig 定义 Prometheus 组件的设置。
prometheusOperator	*PrometheusOperatorConfig	PrometheusOperatorConfig 定义 Prometheus Operator 组件的设置。
prometheusOperatorAdmissionWebhook	*PrometheusOperatorAdmissionWebhookConfig	PrometheusOperatorAdmissionWebhookConfig 定义 Prometheus Operator 的准入 webhook 组件的设置。
openshiftStateMetrics	*OpenShiftStateMetricsConfig	OpenShiftMetricsConfig 定义 openshift-state-metrics 代理的设置。
telemeterClient	*TelemeterClientConfig	TelemeterClientConfig 为 Telemeter Client 组件定义设置。
thanosQuerier	*ThanosQuerierConfig	ThanosQuerierConfig 定义 Thanos Querier 组件的设置。
nodeExporter	NodeExporterConfig	NodeExporterConfig 定义 node-exporter 代理的设置。
monitoringPlugin	*MonitoringPluginConfig	MonitoringPluginConfig 定义了监控 console-plugin 组件的设置。

8.6. DEDICATEDSERVICEMONITORS

8.6.1. 描述

您可以使用 **DedicatedServiceMonitors** 资源为 Prometheus Adapter 配置专用 Service Monitors

会出现：[K8sPrometheusAdapter](#)

属性	类型	描述
----	----	----

属性	类型	描述
enabled	bool	当 enabled 设置为 true 时，Cluster Monitoring Operator (CMO) 会部署一个专门的 Service Monitor，它会公开 kubelet /metrics/resource 端点。此 Service Monitor 会设置 honorTimestamps: true ，并只保留与 Prometheus Adapter 的 pod 资源查询相关的指标。另外，Prometheus Adapter 被配置为使用这些专用指标。总体而言，此功能提高了由 oc adm top pod 命令或 Horizontal Pod Autoscaler 使用的基于 Prometheus Adapter 的 CPU 用量测量的一致性。

8.7. K8SPROMETHEUSADAPTER

8.7.1. 描述

K8sPrometheusAdapter 资源定义 Prometheus Adapter 组件的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
audit	*Audit	定义 Prometheus Adapter 实例使用的审计配置。可能的配置集值包括： Metadata , Request , RequestResponse , 和 None 默认值为 Metadata 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
资源	*v1.ResourceRequirements	为 PrometheusAdapter 容器定义资源请求和限值。
容限 (tolerations)	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。
dedicatedServiceMonitors	* DedicatedServiceMonitors	定义专用服务监控器。

8.8. KUBESTATEMETRICSCONFIG

8.8.1. 描述

KubeStateMetricsConfig 资源定义 **kube-state-metrics** 代理的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
资源	*v1.ResourceRequirements	为 KubeStateMetrics 容器定义资源请求和限值。
容限（tolerations）	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。

8.9. PROMETHEUSOPERATORADMISSIONWEBHOOKCONFIG

8.9.1. 描述

PrometheusOperatorAdmissionWebhookConfig 资源定义 Prometheus Operator 的准入 Webhook 工作负载的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
resources	*v1.ResourceRequirements	为 prometheus-operator-admission-webhook 容器定义资源请求和限值。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。

8.10. MONITORINGPLUGINCONFIG

8.10.1. 描述

MonitoringPluginConfig 资源定义 **openshift-monitoring** 命名空间中的 web 控制台插件组件的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
nodeSelector	map[string]string	定义在其上调度 pod 的节点。

属性	类型	描述
资源	*v1.ResourceRequirements	为 console-plugin 容器定义资源请求和限值。
容限（tolerations）	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布约束。

8.11. NODEEXPORTERCOLLECTORBUDDYINFOCONFIG

8.11.1. 描述

NodeExporterCollectorBuddyInfoConfig 资源充当 **node-exporter** 代理的 **buddyinfo** 收集器的 on/off 开关。默认情况下，**buddyinfo** 收集器被禁用。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 buddyinfo 收集器的布尔值标志。

8.12. NODEEXPORTERCOLLECTORCONFIG

8.12.1. 描述

NodeExporterCollectorConfig 资源定义 **node-exporter** 代理的独立收集器的设置。

出现在：[NodeExporterConfig](#)

属性	类型	描述
cpufreq	NodeExporterCollectorCpuFreqConfig	定义 cpufreq 收集器的配置，用于收集 CPU 频率统计。默认禁用此选项。
tcpstat	NodeExporterCollectorTcpStatConfig	定义 tcpstat 收集器的配置，用于收集 TCP 连接统计信息。默认禁用此选项。
netdev	NodeExporterCollectorNetDevConfig	定义 netdev 收集器的配置，用于收集网络设备统计信息。默认启用此选项。

属性	类型	描述
netclass	NodeExporterCollectorNetClassConfig	定义 netclass 收集器的配置，用于收集有关网络设备的信息。默认启用此选项。
buddyinfo	NodeExporterCollectorBuddyInfoConfig	定义 buddyinfo 收集器的配置，它从 node_buddyinfo_blocks 指标收集有关内存碎片的统计信息。此指标从 /proc/buddyinfo 收集数据。默认禁用此选项。
mountstats	NodeExporterCollectorMountStatsConfig	定义 mountstats 收集器的配置，用于收集有关 NFS 卷 I/O 活动的统计信息。默认禁用此选项。
ksmd	NodeExporterCollectorKSMDConfig	定义 ksmd 收集器的配置，该收集器从内核的 same-page merger 守护进程中收集统计信息。默认禁用此选项。
进程	NodeExporterCollectorProcessesConfig	定义 processes 收集器的配置，它从系统的进程和线程中收集统计信息。默认禁用此选项。
systemd	NodeExporterCollectorSystemdConfig	定义 systemd 收集器的配置，用于收集 systemd 守护进程及其受管服务的统计信息。默认禁用此选项。

8.13. NODEEXPORTERCOLLECTORCPUFREQCONFIG

8.13.1. 描述

使用 **NodeExporterCollectorCpufreqConfig** 资源启用或禁用 **node-exporter** 代理的 **cpufreq** 收集器。默认情况下禁用 **cpufreq** 收集器。在某些情况下，启用 **cpufreq** 收集器会增加带有多个内核的机器上的 CPU 使用量。如果您启用此收集器并具有许多内核的机器，请密切监控您的系统以了解过量 CPU 用量。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 cpufreq 收集器的布尔值标志。

8.14. NODEEXPORTERCOLLECTORKSMDCONFIG

8.14.1. 描述

使用 **NodeExporterCollectorKSMDConfig** 资源启用或禁用 **node-exporter** 代理的 **ksmd** 收集器。默认情况下禁用 **ksmd** 收集器。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 ksmd 收集器的布尔值标志。

8.15. NODEEXPORTERCOLLECTORMOUNTSTATSCONFIG

8.15.1. 描述

使用 **NodeExporterCollectorMountStatsConfig** 资源启用或禁用 **node-exporter** 代理的 **mountstats** 收集器。默认情况下禁用 **mountstats** 收集器。如果启用收集器，则以下指标可用：

node_mountstats_nfs_read_bytes_total, **node_mountstats_nfs_write_bytes_total**, 和 **node_mountstats_nfs_operations_requests_total**. 请注意，这些指标可能会有高基数如果启用此收集器，请密切监控 **prometheus-k8s** Pod 的内存用量增加。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 mountstats 收集器的布尔值标志。

8.16. NODEEXPORTERCOLLECTORNETCLASSCONFIG

8.16.1. 描述

使用 **NodeExporterCollectorNetClassConfig** 资源启用或禁用 **node-exporter** 代理的 **netclass** 收集器。默认情况下启用 **netclass** 收集器。如果禁用此收集器，则以下指标不可用：**node_network_info**, **node_network_address_assign_type**, **node_network_carrier**, **node_network_carrier_changes_total**, **node_network_carrier_up_changes_total**, **node_network_carrier_down_changes_total**, **node_network_device_id**, **node_network_dormant**, **node_network_flags**, **node_network_iface_id**, **node_network_iface_link**, **node_network_iface_link_mode**, **node_network_mtu_bytes**, **node_network_name_assign_type**, **node_network_net_dev_group**, **node_network_speed_bytes**, **node_network_transmit_queue_length**, 和 **node_network_protocol_type**。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 netclass 收集器的布尔值标志。

属性	类型	描述
useNetlink	bool	激活 netclass 收集器的 netlink 实现的布尔值标志。默认值是 true ，这会激活 netlink 模式。此实现提高了 netclass 收集器的性能。

8.17. NODEEXPORTERCOLLECTORNETDEVCONFIG

8.17.1. 描述

使用 **NodeExporterCollectorNetDevConfig** 资源启用或禁用 **node-exporter** 代理的 **netdev** 收集器。默认情况下启用 **netdev** 收集器。如果禁用，以下指标不可用：**node_network_receive_bytes_total**, **node_network_receive_compressed_total**, **node_network_receive_drop_total**, **node_network_receive_errs_total**, **node_network_receive_fifo_total**, **node_network_receive_frame_total**, **node_network_receive_multicast_total**, **node_network_receive_nohandler_total**, **node_network_receive_packets_total**, **node_network_transmit_bytes_total**, **node_network_transmit_carrier_total**, **node_network_transmit_colls_total**, **node_network_transmit_compressed_total**, **node_network_transmit_drop_total**, **node_network_transmit_errs_total**, **node_network_transmit_fifo_total**, 和 **node_network_transmit_packets_total**。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 netdev 收集器的布尔值标志。

8.18. NODEEXPORTERCOLLECTORPROCESSESCONFIG

8.18.1. 描述

使用 **NodeExporterCollectorProcessesConfig** 资源启用或禁用 **node-exporter** 代理的 **processes** 收集器。如果启用了收集器，则以下指标可用：**node_processes_max_processes**, **node_processes_pids**, **node_processes_state**, **node_processes_threads**, **node_processes_threads_state**。指标 **node_processes_state** 和 **node_processes_threads_state** 最多可以有五个系列，具体取决于进程和线程的状态。进程或线程的可能状态为：**D** (UNINTERRUPTABLE_SLEEP), **R** (RUNNING & RUNNABLE), **S** (INTERRUPTABLE_SLEEP), **T** (STOPPED), or **Z** (ZOMBIE)。默认情况下禁用 **processes** 收集器。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	用于启用或禁用 processes 收集器的布尔值标志。

8.19. NODEEXPORTERCOLLECTORSYSTEMDCONFIG

8.19.1. 描述

使用 **NodeExporterCollectorSystemdConfig** 资源启用或禁用 **node-exporter** 代理的 **systemd** 收集器。默认情况下禁用 **systemd** 收集器。如果启用，则以下指标可用：

node_systemd_system_running,node_systemd_units,node_systemd_version。如果单元使用一个套接字，它还会生成以下指

标：**node_systemd_socket_accepted_connections_total,node_systemd_socket_current_connections,node_systemd_socket_refused_connections_total**。您可以使用 **units** 参数选择 **systemd** 收集器中包含的 **systemd** 单元。所选单元用于生成 **node_systemd_unit_state** 指标，其中显示每个 **systemd** 单元的状态。但是，这个指标的基数可能很高（每个节点至少五个系列）。如果您使用所选单元列表启用此收集器，请仔细监控 **prometheus-k8s** 部署以了解过量内存用量。请注意，只有在将 **units** 参数的值配置为 **logrotate.timer** 时才会显示 **node_systemd_timer_last_trigger_seconds** 指标。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 systemd 收集器的布尔值标志。
units	[]string	与 systemd 收集器包含的 systemd 单元匹配的正则表达式 (regex) 特征列表。默认情况下，列表为空，因此收集器不会公开 systemd 单元的指标。

8.20. NODEEXPORTERCOLLECTORTCPSTATCONFIG

8.20.1. 描述

NodeExporterCollectorTcpStatConfig 资源充当 **node-exporter** 代理的 **tcpstat** 收集器的 on/off 开关。默认情况下，**tcpstat** 收集器被禁用。

出现在：[NodeExporterCollectorConfig](#)

属性	类型	描述
enabled	bool	启用或禁用 tcpstat 收集器的布尔值标志。

8.21. NODEEXPORTERCONFIG

8.21.1. 描述

NodeExporterConfig 资源定义 **node-exporter** 代理的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
收集器	NodeExporterCollectorConfig	定义启用哪些收集器及其额外的配置参数。
maxProcs	uint32	运行 node-exporter 的 CPU 的目标数量。默认值为 0 ，这意味着 node-exporter 在所有 CPU 上运行。如果发生内核死锁，或者当同时从 sysfs 读取时性能下降，您可以将这个值改为 1 ，这会将 node-exporter 限制为在一个 CPU 上运行。对于具有高 CPU 数量的节点，您可以将此限制设置为一个低的数量，这可以防止将 Go 例程调度到所有 CPU 上来运行，从而达到保存资源的目的。但是，如果 maxProcs 值设置得太低，并收集很多指标，则 I/O 性能会降低。
ignoredNetworkDevices	*[]string	您要从相关收集器配置中排除的网络设备列表，如 netdev 和 netclass 。如果没有指定列表，Cluster Monitoring Operator 将使用预定义的设备列表来最小化对内存用量的影响。如果列表为空，则不会排除任何设备。如果修改了此设置，请密切监控 prometheus-k8s 部署以了解过量内存用量。
resources	*v1.ResourceRequirements	为 NodeExporter 容器定义资源请求和限值。

8.22. OPENSIFTSTATEMETRICSCONFIG

8.22.1. 描述

OpenShiftStateMetricsConfig 资源定义 **openshift-state-metrics** 代理的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
资源	*v1.ResourceRequirements	为 OpenShiftStateMetrics 容器定义资源请求和限值。
容限（tolerations）	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

8.23. PROMETHEUSK8SCONFIG

8.23.1. 描述

PrometheusK8sConfig 资源定义 Prometheus 组件的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
additionalAlertmanagerConfigs	[] AdditionalAlertmanagerConfig	配置额外的 Alertmanager 实例，从 Prometheus 组件接收警报。默认情况下，没有配置额外的 Alertmanager 实例。
enforcedBodySizeLimit	字符串	为 Prometheus 提取的指标强制实施正文大小限制。如果提取的目标正文响应大于限制，则提取将失败。以下值是有效的：一个空值，用于指定没有限制、以 Prometheus 大小格式（如 64MB ）的数字值，或者字符串 automatic ，这表示将根据集群容量自动计算限制。默认值为空，这代表没有限制。
externalLabels	map[string]string	定义在与外部系统通信时要添加到任何时间序列或警报的标签，如联邦、远程存储和 Alertmanager。默认情况下不会添加任何标签。
logLevel	字符串	定义 Prometheus 的日志级别设置。可能的值有： error 、 warn 、 info 和 debug 。默认值为 info 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。

属性	类型	描述
queryLogFile	字符串	指定记录 PromQL 查询的文件。此设置可以是文件名，在这种情况下，查询被保存到位于 /var/log/prometheus 的 emptyDir 卷，或者挂载到挂载 emptyDir 卷的位置的完整路径，并保存查询。支持写入 /dev/stderr 、 /dev/stdout 或 /dev/null ，但不支持写入任何其他 /dev/ 路径。不支持相对路径。默认情况下，PromQL 查询不会被记录。
remoteWrite	[]RemoteWriteSpec	定义远程写入配置，包括 URL、身份验证和重新标记设置。
资源	*v1.ResourceRequirements	为 Prometheus 容器定义资源请求和限值。
保留	字符串	定义 Prometheus 保留数据的持续时间。这个定义必须使用以下正则表达式模式指定： [0-9]+(ms s m h d w y) (ms = milliseconds, s = seconds, m = minutes, h = hours, d = days, w = weeks, y = years)。默认值为 15d 。
retentionSize	字符串	定义数据块使用的最大磁盘空间量加上 write-ahead log (WAL)。支持的值包括 B, KB, KiB, MB, MiB, GB, GiB, TB, TiB, PB, PiB, EB ，和 EiB 。默认情况下不定义任何限制。
容忍 (tolerations)	[]v1.Toleration	为 pod 定义容忍。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。
collectionProfile	CollectionProfile	定义 Prometheus 用来从平台组件收集指标的指标集合配置集。支持的值是 full 或 minimal 。在 full 配置集（默认）中，Prometheus 会收集平台组件公开的所有指标。在 minimal 配置集中，Prometheus 只收集默认平台警报、记录规则、遥测和控制台仪表盘所需的指标。

属性	类型	描述
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	为 Prometheus 定义持久性存储。使用这个设置配置持久性卷声明，包括存储类、卷大小和名称。

8.24. PROMETHEUSOPERATORCONFIG

8.24.1. 描述

PrometheusOperatorConfig 资源定义 Prometheus Operator 组件的设置。

出现在：[ClusterMonitoringConfiguration](#),[UserWorkloadConfiguration](#)

属性	类型	描述
logLevel	字符串	定义 Prometheus Operator 的日志级别设置。可能的值有 error 、 warn 、 info 和 debug 。默认值为 info 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
资源	*v1.ResourceRequirements	为 PrometheusOperator 容器定义资源请求和限值。
容限（tolerations）	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

8.25. PROMETHEUSRESTRICTEDCONFIG

8.25.1. 描述

PrometheusRestrictedConfig 资源定义监控用户定义的项目的 Prometheus 组件的设置。

出现在：[UserWorkloadConfiguration](#)

属性	类型	描述
additionalAlertmanagerConfigs	<code>[]AdditionalAlertmanagerConfig</code>	配置额外的 Alertmanager 实例，从 Prometheus 组件接收警报。默认情况下，没有配置额外的 Alertmanager 实例。
enforcedLabelLimit	<code>*uint64</code>	指定示例可接受的标签数的 per-scrape 限制。如果标签数量在指标重新标记后超过这个限制，则整个提取将被视为失败。默认值为 0 ，这表示没有设置限制。
enforcedLabelNameLengthLimit	<code>*uint64</code>	为示例指定标签名称长度的 per-scrape 限制。如果标签名称的长度在指标重新标记后超过这个限制，则整个提取将被视为失败。默认值为 0 ，这表示没有设置限制。
enforcedLabelValueLengthLimit	<code>*uint64</code>	为示例指定标签值长度的 per-scrape 限值。如果标签值的长度在指标重新标记后超过这个限制，则整个提取将被视为失败。默认值为 0 ，这表示没有设置限制。
enforcedSampleLimit	<code>*uint64</code>	指定一个接受的提取示例数量的全局限制。如果值大于 enforcedTargetLimit ，则此设置覆盖任何用户定义的 ServiceMonitor 或 PodMonitor 对象中设置的 SampleLimit 值。管理员可以使用此设置来保持控制下的总样本数量。默认值为 0 ，这表示没有设置限制。
enforcedTargetLimit	<code>*uint64</code>	指定提取目标数量的全局限制。如果值大于 enforcedSampleLimit ，则此设置会覆盖任何用户定义的 ServiceMonitor 或 PodMonitor 对象中设置的 TargetLimit 值。管理员可以使用此设置保持控制下的目标总数。默认值为 0 。
externalLabels	<code>map[string]string</code>	定义在与外部系统通信时要添加到任何时间序列或警报的标签，如联邦、远程存储和 Alertmanager。默认情况下不会添加任何标签。

属性	类型	描述
logLevel	字符串	定义 Prometheus 的日志级别设置。可能的值有 error 、 warn 、 info 和 debug 。默认设置为 info 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
queryLogFile	字符串	指定记录 PromQL 查询的文件。此设置可以是文件名，在这种情况下，查询被保存到位于 /var/log/prometheus 的 emptyDir 卷，或者挂载到挂载 emptyDir 卷的位置的完整路径，并保存查询。支持写入 /dev/stderr 、 /dev/stdout 或 /dev/null ，但不支持写入任何其他 /dev/ 路径。不支持相对路径。默认情况下，PromQL 查询不会被记录。
remoteWrite	[]RemoteWriteSpec	定义远程写入配置，包括 URL、身份验证和重新标记设置。
资源	*v1.ResourceRequirements	为 Prometheus 容器定义资源请求和限值。
保留	字符串	定义 Prometheus 保留数据的持续时间。这个定义必须使用以下正则表达式模式指定： [0-9]+(ms s m h d w y) (ms = milliseconds, s = seconds, m = minutes, h = hours, d = days, w = weeks, y = years)。默认值为 15d 。
retentionSize	字符串	定义数据块使用的最大磁盘空间量加上 write-ahead log (WAL)。支持的值包括 B , KB , KiB , MB , MiB , GB , GiB , TB , TiB , PB , PiB , EB , 和 EiB 。默认值为 nil 。
容限 (tolerations)	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

属性	类型	描述
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	为 Prometheus 定义持久性存储。使用此设置配置卷的存储类和大小。

8.26. REMOTEWITESPEC

8.26.1. 描述

RemoteWriteSpec 资源定义远程写入存储的设置。

8.26.2. 必需

- **url**

会出现在：[PrometheusK8sConfig](#)、[PrometheusRestrictedConfig](#)

属性	类型	描述
授权	*monv1.SafeAuthorization	定义远程写入存储的授权设置。
basicAuth	*monv1.BasicAuth	定义远程写入端点 URL 的基本身份验证设置。
bearerTokenFile	字符串	定义包含远程写入端点的 bearer 令牌的文件。但是，因为您无法在 pod 中挂载 secret，所以在实践中，您只能引用服务帐户的令牌。
标头	map[string]string	指定要随每个远程写入请求一起发送的自定义 HTTP 标头。Prometheus 设置的标头不能被覆盖。
metadataConfig	*monv1.MetadataConfig	定义向远程写入存储发送一系列元数据的设置。
name	字符串	定义远程写入队列的名称。此名称用于指标和日志记录来区分队列。如果指定，此名称必须是唯一的。
oauth2	*monv1.OAuth2	定义远程写入端点的 OAuth2 身份验证设置。
proxyUrl	字符串	定义可选的代理 URL。如果启用，它将被集群范围的代理取代。

属性	类型	描述
queueConfig	*monv1.QueueConfig	允许针对远程写入队列参数调整配置。
remoteTimeout	字符串	定义对远程写入端点的请求的超时值。
sigv4	*monv1.Sigv4	定义 AWS 签名版本 4 身份验证设置。
tlsConfig	*monv1.SafeTLSConfig	定义远程写入端点的 TLS 身份验证设置。
url	字符串	定义要向其发送示例的远程写入端点的 URL。
writeRelabelConfigs	[]monv1.RelabelConfig	定义远程写入重新标记配置的列表。

8.27. TLSCONFIG

8.27.1. 描述

TLSConfig 资源配置 TLS 连接的设置。

8.27.2. 必需

- **insecureSkipVerify**

出现在：[AdditionalAlertmanagerConfig](#)

属性	类型	描述
ca	*v1.SecretKeySelector	定义包含用于远程主机的证书颁发机构 (CA) 的 secret 密钥引用。
cert	*v1.SecretKeySelector	定义包含用于远程主机的公共证书的 secret 密钥引用。
key	*v1.SecretKeySelector	定义包含用于远程主机的私钥的 secret 密钥引用。
serverName	字符串	用于验证返回的证书主机名。
insecureSkipVerify	bool	当设置为 true 时，将禁用远程主机的证书和名称的验证。

8.28. TELEMETRICCLIENTCONFIG

8.28.1. 描述

TelemeterClientConfig 为 Telemeter Client 组件定义设置。

8.28.2. 必需

- **nodeSelector**
- **容限 (tolerations)**

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
nodeSelector	map[string]string	定义在其上调度 pod 的节点。
资源	*v1.ResourceRequirements	为 TelemeterClient 容器定义资源请求和限值。
容限 (tolerations)	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

8.29. THANOSQUERIERCONFIG

8.29.1. 描述

ThanosQuerierConfig 资源定义 Thanos Querier 组件的设置。

出现在：[ClusterMonitoringConfiguration](#)

属性	类型	描述
enableRequestLogging	bool	启用或禁用请求日志记录的布尔值标志。默认值为 false 。
logLevel	字符串	定义 Thanos Querier 的日志级别设置。可能的值有 error 、 warn 、 info 和 debug 。默认值为 info 。
enableCORS	bool	启用设置 CORS 标头的布尔值标志。标头允许从任何来源访问。默认值为 false 。
nodeSelector	map[string]string	定义在其上调度 pod 的节点。

属性	类型	描述
资源	*v1.ResourceRequirements	为 Thanos Querier 容器定义资源请求和限值。
容限 (tolerations)	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

8.30. THANOSRULERCONFIG

8.30.1. 描述

ThanosRulerConfig 资源定义面向用户定义的项目的 Thanos Ruler 实例的配置。

出现在：[UserWorkloadConfiguration](#)

属性	类型	描述
additionalAlertmanagerConfigs	[] AdditionalAlertmanagerConfig	配置 Thanos Ruler 组件如何与其他 Alertmanager 实例通信。默认值为 nil 。
logLevel	字符串	定义 Thanos Ruler 的日志级别设置。可能的值有 error 、 warn 、 info 和 debug 。默认值为 info 。
nodeSelector	map[string]string	定义 Pod 被调度到的节点。
资源	*v1.ResourceRequirements	为 Alertmanager 容器定义资源请求和限值。
保留	字符串	定义 Prometheus 保留数据的持续时间。这个定义必须使用以下正则表达式模式指定： [0-9]+(ms s m h d w y) (ms = milliseconds, s = seconds, m = minutes, h = hours, d = days, w = weeks, y = years)。默认值为 15d 。
容限 (tolerations)	[]v1.Toleration	为 pod 定义容限。
topologySpreadConstraints	[]v1.TopologySpreadConstraint	定义 pod 的拓扑分布限制。

属性	类型	描述
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	为 Thanos Ruler 定义持久性存储。使用此设置配置卷的存储类和大小。

8.31. USERWORKLOADCONFIGURATION

8.31.1. 描述

UserWorkloadConfiguration 资源定义了 **openshift-user-workload-monitoring** 命名空间中的 **user-workload-monitoring-config** 配置映射中的用于定义的项目的设置。只有在 **openshift-monitoring** 命名空间内的 **cluster-monitoring-config** 配置映射中的 **enableUserWorkload** 设置被为 **true** 后，您才可以启用 **UserWorkloadConfiguration**。

属性	类型	描述
alertmanager	* AlertmanagerUserWorkloadConfig	在用户工作负载监控中定义 Alertmanager 组件的设置。
prometheus	* PrometheusRestrictedConfig	在用户工作负载监控中定义 Prometheus 组件的设置。
prometheusOperator	* PrometheusOperatorConfig	在用户工作负载监控中定义 Prometheus Operator 组件的设置。
thanosRuler	* ThanosRulerConfig	在用户工作负载监控中定义 Thanos Ruler 组件的设置。