Monitoring stack for Red Hat OpenShift 4.20

About monitoring

Introduction to OpenShift monitoring.

Last Updated: 2026-01-12

Monitoring stack for Red Hat OpenShift 4.20 About monitoring

Introduction to OpenShift monitoring.

Legal Notice

Abstract

This document provides an overview and the architecture of the OpenShift monitoring stack. It also includes key monitoring concepts and terms.

Table of Contents

CHAPTER 1. ABOUT OPENSHIFT CONTAINER PLATFORM MONITORING

Learn how the OpenShift Container Platform provides built-in monitoring for your cluster, including metrics, alerts, and dashboards powered by the Prometheus open source project and its ecosystem.

1.1. MONITORING STACK FOR RED HAT OPENSHIFT OVERVIEW

OpenShift Container Platform includes a preconfigured, preinstalled, and self-updating monitoring stack that monitors core platform components. You also have the option to enable *user workload monitoring* to monitor your own projects.

Core platform monitoring

A cluster administrator can configure the monitoring stack with the supported configurations. OpenShift Container Platform delivers monitoring best practices out of the box.
A set of alerts is included by default that notify administrators about issues with a cluster. *Default dashboards* in the OpenShift Container Platform web console include visual representations of *cluster metrics* to help you quickly understand the state of your cluster. With the OpenShift Container Platform web console, you can access metrics and manage alerts.

User workload monitoring

After installing OpenShift Container Platform, cluster administrators can optionally enable user workload monitoring. By using this feature, cluster administrators, developers, and other users can configure how services and pods are monitored in their own projects.

As a cluster administrator, you can find answers to common problems, such as user metrics unavailability and high consumption of disk space by Prometheus, in "Troubleshooting monitoring issues".

1.2. ADDITIONAL RESOURCES

- Core platform monitoring first steps
- User workload monitoring first steps
- Developer and non-administrator steps
- Troubleshooting monitoring issues

CHAPTER 2. MONITORING STACK ARCHITECTURE

The OpenShift Container Platform monitoring stack is based on the Prometheus open source project and its wider ecosystem. You can learn about the monitoring stack architecture, which includes default monitoring components and components for monitoring user-defined projects.

2.1. UNDERSTANDING THE MONITORING STACK

The monitoring stack includes the following components:

Default platform monitoring components

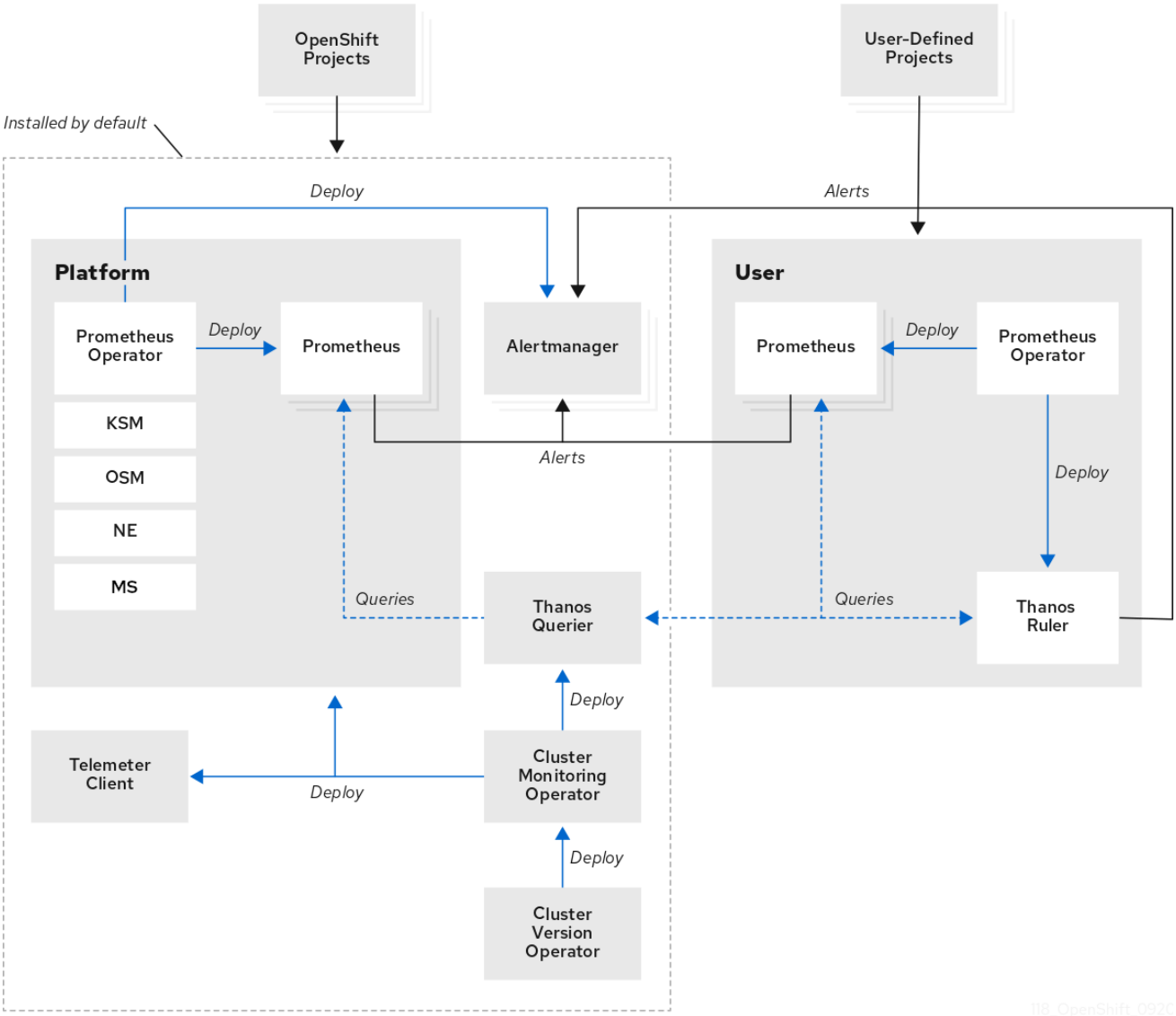
A set of platform monitoring components are installed in the **openshift-monitoring** project by default during an OpenShift Container Platform installation. This provides monitoring for core cluster components including Kubernetes services. The default monitoring stack also enables remote health monitoring for clusters.

You can see these components in the **Installed by default** section in the following diagram.

Components for monitoring user-defined projects

If you enable monitoring for user-defined projects, additional monitoring components are installed in the **openshift-user-workload-monitoring** project. This provides optional monitoring for user-defined projects.

You can see these components in the **User** section in the following diagram.



2.2. DEFAULT MONITORING COMPONENTS

By default, the OpenShift Container Platform 4.20 monitoring stack includes the following components:

Table 2.1. Default monitoring stack components

Component	Description
Cluster Monitoring Operator	The Cluster Monitoring Operator (CMO) is a central component of the monitoring stack. It deploys, manages, and automatically updates Prometheus and Alertmanager instances, Thanos Querier, Telemeter Client, and metrics targets. The CMO is deployed by the Cluster Version Operator (CVO).

Component	Description
Prometheus Operator	The Prometheus Operator in the **openshift-monitoring** project creates, configures, and manages platform Prometheus instances and Alertmanager instances. It also automatically generates monitoring target configurations based on Kubernetes label queries.
Prometheus	The OpenShift Container Platform monitoring stack is based on the Prometheus monitoring system. Prometheus is a time-series database and a rule evaluation engine for metrics. Prometheus sends alerts to Alertmanager for processing.
Metrics Server	The Metrics Server component (MS in the preceding diagram) collects resource metrics and exposes them in the **metrics.k8s.io** Metrics API service for use by other tools and APIs, which frees the core platform Prometheus stack from handling this functionality. Note that with the OpenShift Container Platform 4.16 release, Metrics Server replaces Prometheus Adapter.
Alertmanager	The Alertmanager service handles alerts received from Prometheus. Alertmanager is also responsible for sending the alerts to external notification systems.
kube-state-metrics agent	The kube-state-metrics exporter agent (KSM in the preceding diagram) converts Kubernetes objects to metrics that Prometheus can use.
monitoring-plugin	The monitoring-plugin dynamic plugin component deploys the monitoring pages in the **Observe** section of the OpenShift Container Platform web console. You can use Cluster Monitoring Operator config map settings to manage monitoring-plugin resources for the web console pages.
openshift-state-metrics agent	The openshift-state-metrics exporter (OSM in the preceding diagram) expands upon kube-state-metrics by adding metrics for OpenShift Container Platform-specific resources.
node-exporter agent	The node-exporter agent (NE in the preceding diagram) collects metrics about every node in a cluster. The node-exporter agent is deployed on every node.

Component	Description
Thanos Querier	Thanos Querier aggregates and optionally deduplicates core OpenShift Container Platform metrics and metrics for user-defined projects under a single, multi-tenant interface.
Telemeter Client	Telemeter Client sends a subsection of the data from platform Prometheus instances to Red Hat to enable remote health monitoring for clusters.

The monitoring stack monitors all components within the stack. The components are automatically updated when OpenShift Container Platform is updated.

2.2.1. Default monitoring targets

In addition to the components of the stack itself, the default monitoring stack monitors additional platform components.

The following are examples of monitoring targets:

- CoreDNS
- etcd
- HAProxy
- Image registry
- Kubelets
- Kubernetes API server
- Kubernetes controller manager
- Kubernetes scheduler
- OpenShift API server
- OpenShift Controller Manager
- Operator Lifecycle Manager (OLM)

NOTE

- The exact list of targets can vary depending on your cluster capabilities and installed components.
- Each OpenShift Container Platform component is responsible for its monitoring configuration. For problems with the monitoring of an OpenShift Container Platform component, open a Jira issue against that component, not against the general monitoring component.

Other OpenShift Container Platform framework components might be exposing metrics as well. For details, see their respective documentation.

Additional resources

- Getting detailed information about a metrics target

2.3. COMPONENTS FOR MONITORING USER-DEFINED PROJECTS

OpenShift Container Platform includes an optional enhancement to the monitoring stack that helps you monitor services and pods in user-defined projects. This feature includes the following components:

Table 2.2. Components for monitoring user-defined projects

Component	Description
Prometheus Operator	The Prometheus Operator in the **openshift-user-workload-monitoring** project creates, configures, and manages Prometheus and Thanos Ruler instances in the same project.
Prometheus	Prometheus is the monitoring system that provides monitoring for user-defined projects. Prometheus sends alerts to Alertmanager for processing.
Thanos Ruler	The Thanos Ruler is a rule evaluation engine for Prometheus that is deployed as a separate process. In OpenShift Container Platform , Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.
Alertmanager	The Alertmanager service handles alerts received from Prometheus and Thanos Ruler. Alertmanager is also responsible for sending user-defined alerts to external notification systems. Deploying this service is optional.

NOTE

The components in the preceding table are deployed after you enable monitoring for user-defined projects.

The monitoring stack monitors all components for user-defined projects. The components are automatically updated when OpenShift Container Platform is updated.

2.3.1. Monitoring targets for user-defined projects

When monitoring is enabled for user-defined projects, you can monitor:

- Metrics provided through service endpoints in user-defined projects.
- Pods running in user-defined projects.

2.4. THE MONITORING STACK IN HIGH-AVAILABILITY CLUSTERS

By default, in multi-node clusters, the following components run in high-availability (HA) mode to prevent data loss and service interruption:

- Prometheus
- Alertmanager
- Thanos Ruler
- Thanos Querier
- Metrics Server
- Monitoring plugin

The component is replicated across two pods, each running on a separate node. This means that the monitoring stack can tolerate the loss of one pod.

Prometheus in HA mode

- Both replicas independently scrape the same targets and evaluate the same rules.
- The replicas do not communicate with each other. Therefore, data might differ between the pods.

Alertmanager in HA mode

- The two replicas synchronize notification and silence states with each other. This ensures that each notification is sent at least once.
- If the replicas fail to communicate or if there is an issue on the receiving side, notifications are still sent, but they might be duplicated.

IMPORTANT

Prometheus, Alertmanager, and Thanos Ruler are stateful components. To ensure high availability, you must configure them with persistent storage.

Additional resources

- Configuring persistent storage
- Configuring performance and scalability

2.5. TLS SECURITY AND ROTATION IN THE MONITORING STACK

Learn how TLS profiles and certificate rotation work in the OpenShift Container Platform monitoring stack to keep communication secure.

TLS security profiles for monitoring components

All components of the monitoring stack use the TLS security profile settings that are centrally configured by a cluster administrator. The monitoring stack component uses the TLS security profile settings that already exist in the **tlsSecurityProfile** field in the global OpenShift Container Platform

apiservers.config.openshift.io/cluster resource.

TLS certificate rotation and automatic restarts

The Cluster Monitoring Operator (CMO) manages the internal TLS certificate lifecycle for the monitoring components. These certificates secure the internal communication between the monitoring components.
During certificate rotation, the CMO updates secrets and config maps, which triggers automatic restarts of affected pods. This is an expected behavior, and the pods recover automatically.

The following example shows events that occur during certificate rotation:

```
$ 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
```

Additional resources

- Configuring TLS security profiles

2.6. GLOSSARY OF COMMON TERMS FOR OPENSHIFT CONTAINER PLATFORM MONITORING

This glossary defines common terms that are used in OpenShift Container Platform architecture.

Alertmanager

Alertmanager handles alerts received from Prometheus. Alertmanager is also responsible for sending the alerts to external notification systems.

Alerting rules

Alerting rules contain a set of conditions that outline a particular state within a cluster. Alerts are triggered when those conditions are true. An alerting rule can be assigned a severity that defines how the alerts are routed.

Cluster Monitoring Operator

The Cluster Monitoring Operator (CMO) is a central component of the monitoring stack. It deploys and manages Prometheus instances such as, the Thanos Querier, the Telemeter Client, and metrics targets to ensure that they are up to date. The CMO is deployed by the Cluster Version Operator (CVO).

Cluster Version Operator

The Cluster Version Operator (CVO) manages the lifecycle of cluster Operators, many of which are installed in OpenShift Container Platform by default.

config map

A config map provides a way to inject configuration data into pods. You can reference the data stored in a config map in a volume of type **ConfigMap**. Applications running in a pod can use this data.

Container

A container is a lightweight and executable image that includes software and all its dependencies. Containers virtualize the operating system. As a result, you can run containers anywhere from a data center to a public or private cloud as well as a developer's laptop.

custom resource (CR)

A CR is an extension of the Kubernetes API. You can create custom resources.

etcd

etcd is the key-value store for OpenShift Container Platform, which stores the state of all resource objects.

Kubelets

Runs on nodes and reads the container manifests. Ensures that the defined containers have started and are running.

Kubernetes API server

Kubernetes API server validates and configures data for the API objects.

Kubernetes controller manager

Kubernetes controller manager governs the state of the cluster.

Kubernetes scheduler

Kubernetes scheduler allocates pods to nodes.

labels

Labels are key-value pairs that you can use to organize and select subsets of objects such as a pod.

Metrics Server

The Metrics Server monitoring component collects resource metrics and exposes them in the **metrics.k8s.io** Metrics API service for use by other tools and APIs, which frees the core platform Prometheus stack from handling this functionality.

node

A compute machine in the OpenShift Container Platform cluster. A node is either a virtual machine (VM) or a physical machine.

Operator

The preferred method of packaging, deploying, and managing a Kubernetes application in an OpenShift Container Platform cluster. An Operator takes human operational knowledge and encodes it into software that is packaged and shared with customers.

Operator Lifecycle Manager (OLM)

OLM helps you install, update, and manage the lifecycle of Kubernetes native applications. OLM is an open source toolkit designed to manage Operators in an effective, automated, and scalable way.

Persistent storage

Stores the data even after the device is shut down. Kubernetes uses persistent volumes to store the application data.

Persistent volume claim (PVC)

You can use a PVC to mount a PersistentVolume into a Pod. You can access the storage without knowing the details of the cloud environment.

pod

The pod is the smallest logical unit in Kubernetes. A pod is comprised of one or more containers to run in a worker node.

Prometheus

Prometheus is the monitoring system on which the OpenShift Container Platform monitoring stack is based. Prometheus is a time-series database and a rule evaluation engine for metrics. Prometheus sends alerts to Alertmanager for processing.

Prometheus Operator

The Prometheus Operator in the **openshift-monitoring** project creates, configures, and manages platform Prometheus and Alertmanager instances. It also automatically generates monitoring target configurations based on Kubernetes label queries.

Silences

A silence can be applied to an alert to prevent notifications from being sent when the conditions for an alert are true. You can mute an alert after the initial notification, while you work on resolving the underlying issue.

storage

OpenShift Container Platform supports many types of storage, both for on-premise and cloud providers. You can manage container storage for persistent and non-persistent data in an OpenShift Container Platform cluster.

Thanos Ruler

The Thanos Ruler is a rule evaluation engine for Prometheus that is deployed as a separate process. In OpenShift Container Platform, Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.

Vector

Vector is a log collector that deploys to each OpenShift Container Platform node. It collects log data from each node, transforms the data, and forwards it to configured outputs.

web console

A user interface (UI) to manage OpenShift Container Platform.

2.7. ADDITIONAL RESOURCES

- About remote health monitoring
- Granting users permissions for monitoring for user-defined projects