OpenShift Container Platform 4.11

모니터링

OpenShift Container Platform에서 모니터링 스택 구성 및 사용

Last Updated: 2024-02-10

OpenShift Container Platform 4.11 모니터링

OpenShift Container Platform에서 모니터링 스택 구성 및 사용

법적 공지

초록

이 문서에서는 OpenShift Container Platform에서 Prometheus를 구성 및 사용하는 방법에 대한 지침을 설명합니다.

차례

1장. 모니터링 개요

1.1. OPENSHIFT CONTAINER PLATFORM 모니터링 정보

OpenShift Container Platform에는 코어 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치된 자체 모니터링 스택이 포함되어 있습니다. 사용자 정의 프로젝트에 대한 모니터링을 활성화하는 옵션도 있습니다.

클러스터 관리자는 지원되는 구성으로 모니터링 스택을 구성 할 수 있습니다. OpenShift Container Platform은 즉시 사용 가능한 모니터링 모범 사례를 제공합니다.

클러스터 문제에 대해 즉시 알리는 일련의 경고가 기본적으로 포함되어 있습니다. OpenShift Container Platform 웹 콘솔의 기본 대시보드에는 클러스터 상태를 빠르게 파악할 수 있도록 클러스터 메트릭에 대한 그래픽 표현이 포함되어 있습니다. OpenShift Container Platform 웹 콘솔을 사용하면 메트릭 ,알림, 모니터링 대시보드 를 보고 관리할 수 있습니다.

OpenShift Container Platform 웹 콘솔의 **Observe** 섹션에서 메트릭 , 경고, 모니터링대시보드 및 메트릭 대상 과 같은 모니터링 기능에 액세스하고 관리할 수 있습니다.

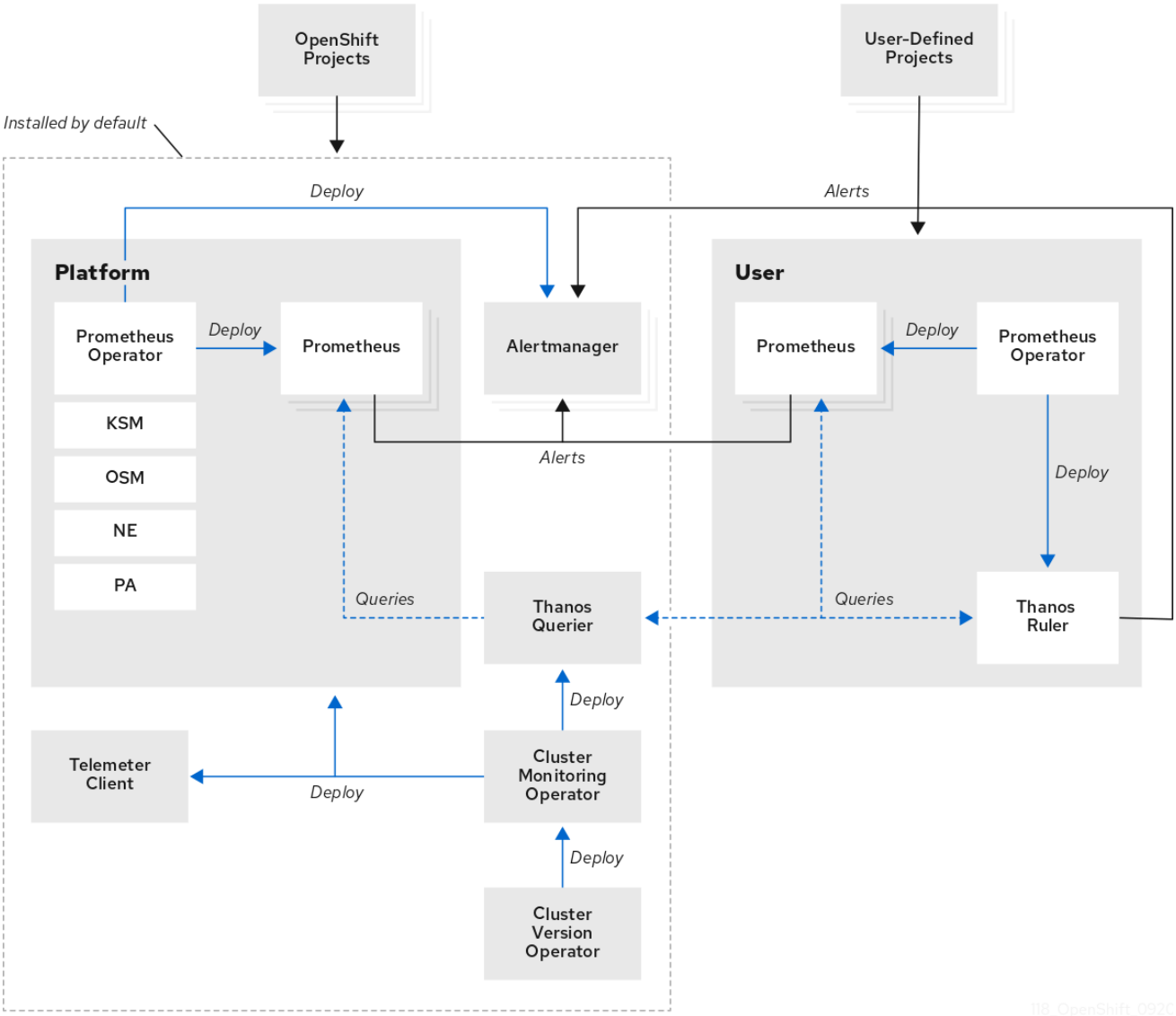
OpenShift Container Platform을 설치한 후 클러스터 관리자는 선택 옵션으로 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다. 이 기능을 사용하면 클러스터 관리자, 개발자, 기타 사용자가 자신의 프로젝트에서 서비스와 Pod를 모니터링하는 방법을 지정할 수 있습니다. 클러스터 관리자는 모니터링 문제 해결에서 Prometheus가 사용할 수 없는 사용자 지표 및 고가용성 디스크 사용량과 같은 일반적인 문제에 대한 답변을 찾을 수 있습니다.

1.2. 모니터링 스택 이해

OpenShift Container Platform 모니터링 스택은 Prometheus 오픈 소스 프로젝트 및 광범위한 에코시스템을 기반으로 합니다. 모니터링 스택에는 다음이 포함됩니다.

- **기본 플랫폼 모니터링 구성 요소**입니다. OpenShift Container Platform을 설치하는 동안 기본적으로 플랫폼 모니터링 구성 요소가 **openshift-monitoring** 프로젝트에 설치됩니다. 이를 통해 Kubernetes 서비스를 포함한 주요 OpenShift Container Platform 구성 요소 모니터링이 제공됩니다. 기본 모니터링 스택은 클러스터에 대한 원격 상태 모니터링도 가능합니다. 이러한 구성 요소는 다음 다이어그램의 **기본적으로 설치됨** 섹션에 설명되어 있습니다.

- **사용자 정의 프로젝트를 모니터링하기 위한 구성 요소**입니다. 선택적으로 사용자 정의 프로젝트에 대한 모니터링을 활성화하면 **openshift-user-workload-monitoring** 프로젝트에 추가 모니터링 구성 요소가 설치됩니다. 이는 사용자 정의 프로젝트에 대한 모니터링을 제공합니다. 이러한 구성 요소는 다음 다이어그램의 **사용자** 섹션에 설명되어 있습니다.



1.2.1. 기본 모니터링 구성 요소

기본적으로 OpenShift Container Platform 4.11 모니터링 스택에는 다음 구성 요소가 포함됩니다.

표 1.1. 기본 모니터링 스택 구성 요소

구성 요소	설명
Cluster Monitoring Operator	CMO(Cluster Monitoring Operator)는 모니터링 스택의 핵심 구성 요소입니다. Prometheus 및 Alertmanager 인스턴스, Thanos Querier, Telemeter Client 및 지표 대상을 배포, 관리 및 자동으로 업데이트합니다. CMO는 CVO(Cluster Version Operator)에 의해 배포됩니다.
Prometheus Operator	**openshift-monitoring** 프로젝트의 PO(Prometheus Operator)는 플랫폼 Prometheus 인스턴스 및 Alertmanager 인스턴스를 생성, 구성 및 관리합니다. 또한 Kubernetes 라벨 쿼리를 기반으로 모니터링 대상 구성을 자동으로 생성합니다.

구성 요소	설명
Prometheus	Prometheus는 OpenShift Container Platform 모니터링 스택을 기반으로 하는 모니터링 시스템입니다. Prometheus는 시계열 데이터베이스이며 메트릭에 대한 규칙 평가 엔진입니다. Prometheus는 처리를 위해 Alertmanager에 경고를 보냅니다.
Prometheus Adapter	Prometheus Adapter(이전 다이어그램의PA)는 Kubernetes 및 Pod 쿼리를 Prometheus에서 사용합니다. 변환된 리소스 메트릭에는 CPU 및 메모리 사용량 메트릭이 포함됩니다. Prometheus Adapter는 수평 Pod 자동 스케일링을 위해 클러스터 리소스 메트릭 API를 노출합니다. Prometheus Adapter는 **oc adm top node** 및 **oc adm top pods** 명령에서도 사용합니다.
Alertmanager	Alertmanager 서비스는 Prometheus에서 수신한 경고를 처리합니다. 또한 Alertmanager는 경고를 외부 알림 시스템으로 전송해야 합니다.
kube-state-metrics 에이전트	이전 다이어그램의 **kube-state-metrics** 내보내기 에이전트(이전 다이어그램의 KSM)는 Kubernetes 오브젝트를 Prometheus가 사용할 수 있는 메트릭으로 변환합니다.
openshift-state-metrics 에이전트	**openshift-state-metrics** 내보내기(이전 다이어그램의 OSM)는 OpenShift Container Platform 특정 리소스에 대한 메트릭을 추가하여 **kube-state-metrics**에 기반하여 확장됩니다.
node-exporter 에이전트	**node-exporter** 에이전트(이전 다이어그램의 NE)는 클러스터의 모든 노드에 대한 메트릭을 수집합니다. **node-exporter** 에이전트는 모든 노드에 배포됩니다.
Thanos Querier	Thanos Querier는 단일 다중 테넌트 인터페이스에서 핵심 OpenShift Container Platform 메트릭과 사용자 정의 프로젝트에 대한 메트릭을 집계하고 선택적으로 중복을 제거합니다.
Telemeter Client	Telemeter Client는 플랫폼 Prometheus 인스턴스에서 Red Hat으로 데이터의 하위 섹션을 보내 클러스터의 원격 상태 모니터링을 용이하게 합니다.

모니터링 스택의 모든 구성 요소는 스택에서 모니터링되며 OpenShift Container Platform 업데이트 시 자동으로 업데이트됩니다.

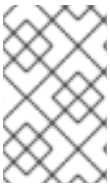
참고

모니터링 스택의 모든 구성 요소는 클러스터 관리자가 중앙에서 구성하는 TLS 보안 프로필 설정을 사용합니다. TLS 보안 설정을 사용하는 모니터링 스택 구성 요소를 구성하는 경우 구성 요소는 글로벌 OpenShift Container Platform **apiservers.config.openshift.io/cluster** 리소스의 **tlsSecurityProfile** 필드에 이미 존재하는 TLS 보안 프로필 설정을 사용합니다.

1.2.2. 기본 모니터링 대상

스택 자체의 구성 요소 외에도 기본 모니터링 스택은 다음을 모니터링합니다.

- CoreDNS
- Elasticsearch(로깅이 설치된 경우)
- etcd
- Fluentd(로깅이 설치된 경우)
- HAProxy
- 이미지 레지스트리
- Kubelets
- Kubernetes API 서버
- Kubernetes 컨트롤러 관리자
- Kubernetes 스케줄러
- OpenShift API 서버
- OpenShift Controller Manager
- OLM(Operator Lifecycle Manager)



참고

각 OpenShift Container Platform 구성 요소는 모니터링 구성을 담당합니다. OpenShift Container Platform 구성 요소 모니터링에 문제가 발생하면 일반 모니터링 구성 요소에 대한 것이 아니라 해당 구성 요소에 대해 Jira 문제를 엽니다.

다른 OpenShift Container Platform 프레임워크 구성 요소도 메트릭을 노출할 수 있습니다. 자세한 내용은 해당 문서를 참조하십시오.

1.2.3. 사용자 정의 프로젝트를 모니터링하기 위한 구성 요소

OpenShift Container Platform 4.11에는 사용자 정의 프로젝트의 서비스 및 Pod를 모니터링할 수 있는 모니터링 스택에 선택적 개선 사항이 포함되어 있습니다. 이 기능에는 다음과 같은 구성 요소가 포함됩니다.

표 1.2. 사용자 정의 프로젝트를 모니터링하기 위한 구성 요소

구성 요소	설명
Prometheus Operator	**openshift-user-workload-monitoring** 프로젝트의 PO(Prometheus Operator)는 동일한 프로젝트에서 Prometheus 및 Thanos Ruler 인스턴스를 생성, 구성 및 관리합니다.
Prometheus	Prometheus는 사용자 정의 프로젝트에 대한 모니터링이 제공되는 모니터링 시스템입니다. Prometheus는 처리를 위해 Alertmanager에 경고를 보냅니다.
Thanos Ruler	Thanos Ruler는 별도의 프로세스로 배포되는 Prometheus의 규칙 평가 엔진입니다. OpenShift Container Platform 4.11에서 Thanos Ruler는 사용자 정의 프로젝트의 모니터링에 대한 규칙 및 경고 평가를 제공합니다.
Alertmanager	Alertmanager 서비스는 Prometheus 및 Thanos Ruler에서 수신한 경고를 처리합니다. 또한 Alertmanager는 사용자 정의 경고를 외부 알림 시스템으로 전송합니다. 이 서비스 배포는 선택 사항입니다.

참고

이전 표의 구성 요소는 사용자 정의 프로젝트에 대한 모니터링이 활성화된 후 배포됩니다.

모니터링 스택의 모든 구성 요소는 스택에서 모니터링되며 OpenShift Container Platform 업데이트 시 자동으로 업데이트됩니다.

1.2.4. 사용자 정의 프로젝트의 대상 모니터링

사용자 정의 프로젝트에 대한 모니터링이 활성화된 경우 다음을 모니터링할 수 있습니다.

- 사용자 정의 프로젝트에서 서비스 끝점을 통해 제공되는 메트릭입니다.
- 사용자 정의 프로젝트에서 실행 중인 Pod.

1.3. OPENSHIFT CONTAINER PLATFORM 모니터링에 대한 일반 용어

이 용어집은 OpenShift Container Platform 아키텍처에서 사용되는 일반적인 용어를 정의합니다.

Alertmanager

Alertmanager는 Prometheus에서 수신한 경고를 처리합니다. 또한 Alertmanager는 경고를 외부 알림 시스템으로 전송해야 합니다.

경고 규칙

경고 규칙에는 클러스터 내에서 특정 상태를 설명하는 일련의 조건이 포함되어 있습니다. 이러한 조건이 true이면 경고가 트리거됩니다. 경고 규칙은 경고의 라우팅 방법을 정의하는 심각도를 할당할 수 있습니다.

Cluster Monitoring Operator

CMO(Cluster Monitoring Operator)는 모니터링 스택의 핵심 구성 요소입니다. Thanos Querier, Telemeter Client 및 메트릭 대상과 같은 Prometheus 인스턴스를 배포 및 관리하여 이러한 인스턴스를 최신 상태로 유지합니다. CMO는 CVO(Cluster Version Operator)에 의해 배포됩니다.

Cluster Version Operator

CVO(Cluster Version Operator)는 기본적으로 OpenShift Container Platform에 설치된 클러스터 Operator의 라이프사이클을 관리합니다.

구성 맵

구성 맵에서는 구성 데이터를 Pod에 삽입하는 방법을 제공합니다. 구성 맵에 저장된 데이터를 **ConfigMap** 유형의 볼륨에서 참조할 수 있습니다. Pod에서 실행되는 애플리케이션에서는 이 데이터를 사용할 수 있습니다.

컨테이너

컨테이너는 소프트웨어 및 모든 종속 항목을 포함하는 가볍고 실행 가능한 이미지입니다. 컨테이너는 운영 체제를 가상화합니다. 결과적으로 데이터 센터에서 퍼블릭 또는 프라이빗 클라우드에 이르기까지 어디에서나 컨테이너를 실행할 수 있으며 개발자 노트북도 실행할 수 있습니다.

CR(사용자 정의 리소스)

CR은 Kubernetes API의 확장입니다. 사용자 정의 리소스를 생성할 수 있습니다.

etcd

etcd는 모든 리소스 오브젝트의 상태를 저장하는 OpenShift Container Platform의 키-값 저장소입니다.

fluentd

Fluentd는 노드에서 로그를 수집하여 Elasticsearch에 제공합니다.

Kubelets

노드에서 실행되며 컨테이너 매니페스트를 읽습니다. 정의된 컨테이너가 시작되어 실행 중인지 확인합니다.

Kubernetes API 서버

Kubernetes API 서버는 API 오브젝트의 데이터를 검증하고 구성합니다.

Kubernetes 컨트롤러 관리자

Kubernetes 컨트롤러 관리자는 클러스터 상태를 관리합니다.

Kubernetes 스케줄러

Kubernetes 스케줄러는 노드에 Pod를 할당합니다.

labels

레이블은 Pod와 같은 오브젝트의 하위 집합을 구성하고 선택하는 데 사용할 수 있는 키-값 쌍입니다.

node

OpenShift Container Platform 클러스터의 작업자 시스템입니다. 노드는 VM(가상 머신) 또는 물리적 머신입니다.

Operator

OpenShift Container Platform 클러스터에서 Kubernetes 애플리케이션을 패키징, 배포 및 관리하는 기본 방법입니다. Operator는 사람의 운영 지식을 패키지하고 고객과 공유하는 소프트웨어로 인코딩합니다.

OLM(Operator Lifecycle Manager)

OLM은 Kubernetes 네이티브 애플리케이션의 라이프사이클을 설치, 업데이트 및 관리할 수 있도록 지원합니다. OLM은 효과적이고 자동화되고 확장 가능한 방식으로 Operator를 관리하도록 설계된 오픈 소스 툴킷입니다.

영구 스토리지

장치가 종료된 후에도 데이터를 저장합니다. Kubernetes는 영구 볼륨을 사용하여 애플리케이션 데이터를 저장합니다.

PVC(영구 볼륨 클레임)

PVC를 사용하여 PersistentVolume을 포드에 마운트할 수 있습니다. 클라우드 환경의 세부 사항을 모르는 상태에서 스토리지에 액세스할 수 있습니다.

Pod

Pod는 Kubernetes에서 가장 작은 논리 단위입니다. Pod는 작업자 노드에서 실행할 하나 이상의 컨테이너로 구성됩니다.

Prometheus

Prometheus는 OpenShift Container Platform 모니터링 스택을 기반으로 하는 모니터링 시스템입니다. Prometheus는 시계열 데이터베이스이며 메트릭에 대한 규칙 평가 엔진입니다. Prometheus는 처리를 위해 Alertmanager에 경고를 보냅니다.

Prometheus 어댑터

Prometheus Adapter는 Kubernetes 노드와 Prometheus에서 사용할 Pod 쿼리를 변환합니다. 변환된 리소스 메트릭에는 CPU 및 메모리 사용률이 포함됩니다. Prometheus Adapter는 수평 Pod 자동 스케일링을 위해 클러스터 리소스 메트릭 API를 노출합니다.

Prometheus Operator

openshift-monitoring 프로젝트의 PO(Prometheus Operator)는 플랫폼 Prometheus 및 Alertmanager 인스턴스를 생성, 구성 및 관리합니다. 또한 Kubernetes 라벨 쿼리를 기반으로 모니터링 대상 구성을 자동으로 생성합니다.

음소거

경고 조건이 true일 때 알림이 전송되는 것을 방지하기 위해 경고에 음소거를 적용할 수 있습니다. 기본 문제를 해결하는 동안 초기 알림 후 경고를 음소거할 수 있습니다.

storage

OpenShift Container Platform은 온프레미스 및 클라우드 공급자를 위해 다양한 유형의 스토리지를 지원합니다. OpenShift Container Platform 클러스터에서 영구 및 비영구 데이터에 대한 컨테이너 스토리지를 관리할 수 있습니다.

Thanos Ruler

Thanos Ruler는 별도의 프로세스로 배포되는 Prometheus의 규칙 평가 엔진입니다. OpenShift Container Platform에서 Thanos Ruler는 사용자 정의 프로젝트의 모니터링에 대한 규칙 및 경고 평가를 제공합니다.

웹 콘솔

OpenShift Container Platform을 관리할 UI(사용자 인터페이스)입니다.

1.4. 추가 리소스

- 원격 상태 모니터링 정보
- 사용자 정의 프로젝트를 모니터링할 수 있는 사용자 권한 부여
- TLS 보안 프로필 설정

1.5. 다음 단계

- 모니터링 스택 구성

2장. 모니터링 스택 구성

OpenShift Container Platform 4 설치 프로그램은 설치 전에 몇 가지 구성 옵션만 제공합니다. 클러스터 모니터링 스택을 포함한 대부분의 OpenShift Container Platform 프레임워크 구성 요소는 설치 후 수행됩니다.

이 섹션에서는 지원되는 구성에 대해 설명하고 모니터링 스택을 구성하는 방법을 보여주고 몇 가지 일반적인 구성 시나리오를 보여줍니다.

2.1. 사전 요구 사항

- 모니터링 스택은 추가 리소스 요구 사항을 적용합니다. Cluster Monitoring Operator 스케일링 에서 컴퓨팅 리소스 권장 사항을 참조하고 충분한 리소스가 있는지 확인합니다.

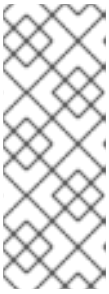
2.2. 모니터링의 유지보수 및 지원

지원되는 OpenShift Container Platform 모니터링 구성 방법은 이 설명서에 설명된 옵션을 사용하여 구성하는 것입니다. **다른 구성은 지원되지 않으므로 사용하지 마십시오.** 구성 패러다임은 Prometheus 릴리스마다 변경될 수 있으며 이러한 경우는 모든 구성 가능성이 제어되는 경우에만 정상적으로 처리될 수 있습니다. 이 섹션에 설명된 것과 다른 구성을 사용하는 경우 **cluster-monitoring-operator**가 차이를 조정하므로 변경한 내용이 사라집니다. Operator는 원래 기본적으로 모든 항목이 정의된 상태로 재설정합니다.

2.2.1. 모니터링에 대한 지원 고려 사항

다음과 같은 수정 사항은 명시적으로 지원되지 않습니다.

- **openshift-* 및 kube-* 프로젝트에서 추가 ServiceMonitor, PodMonitor 및 PrometheusRule 오브젝트 생성.**
- **openshift-monitoring 또는 openshift-user-workload-monitoring 프로젝트에 배포된 모든 리소스 또는 오브젝트 수정.** OpenShift Container Platform 모니터링 스택에서 생성된 리소스는 이전 버전과의 호환성을 보장하지 않으므로 다른 리소스에서 사용할 수 없습니다.



참고

Alertmanager 구성은 **openshift-monitoring** 네임스페이스에 시크릿 리소스로 배포됩니다. 사용자 정의 경고 라우팅에 대해 별도의 Alertmanager 인스턴스를 활성화한 경우 Alertmanager 설정도 **openshift-user-workload-monitoring** 네임스페이스에 시크릿 리소스로 배포됩니다. Alertmanager 인스턴스에 대한 추가 경로를 구성하려면 해당 시크릿을 디코딩, 수정, 인코딩해야 합니다. 이 프로세스는 이전 조건에 예외적으로 지원됩니다.

- **스택의 리소스 수정** OpenShift Container Platform 모니터링 스택을 통해 해당 리소스가 항상 예상되는 상태에 있습니다. 이 기능이 수정되면 스택이 이를 재설정합니다.
- **사용자 정의 워크로드를 openshift-* 및 kube-* 프로젝트에 배포.** 이러한 프로젝트는 Red Hat 제공 구성 요소용으로 예약되어 있으며 사용자 정의 워크로드에는 사용할 수 없습니다.
- **OpenShift Container Platform에 사용자 정의 Prometheus 인스턴스 설치.** 사용자 정의 인스턴스는 Prometheus Operator에서 관리하는 Prometheus 사용자 정의 리소스(CR)입니다.
- **Prometheus Operator에서 Probe CRD(사용자 정의 리소스 정의)를 사용하여 증상 기반 모니터링을 활성화.**

참고

메트릭, 기록 규칙 또는 경고 규칙에 대한 이전 버전과의 호환성은 보장되지 않습니다.

2.2.2. Operator 모니터링에 대한 지원 정책

Operator 모니터링은 OpenShift Container Platform 모니터링 리소스가 설계 및 테스트된 대로 작동하는지 확인합니다. Operator의 CVO(Cluster Version Operator) 제어가 재정의되면 Operator가 설정 변경에 응답하지 않거나, 클러스터 오브젝트의 상태를 조정하거나 업데이트를 수신합니다.

디버깅 중에는 Operator에 대한 CVO 제어 재정의가 유용할 수 있지만, 이는 지원되지 않으며 개별 구성 요소의 구성 및 업그레이드를 클러스터 관리자가 전적으로 통제하게 됩니다.

Cluster Version Operator 재정의

spec.overrides 매개변수를 CVO의 구성에 추가하여 관리자가 구성 요소에 대한 CVO 동작에 대한 재정의 목록을 제공할 수 있습니다. 구성 요소에 대해 **spec.overrides[].unmanaged** 매개변수를 **true**로 설정하면 클러스터 업그레이드가 차단되고 CVO 재정의가 설정된 후 관리자에게 경고합니다.

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

주의

CVO 재정의를 설정하면 전체 클러스터가 지원되지 않는 상태가 되고 모니터링 스택이 의도된 상태와 조정되지 않도록 합니다. 이는 Operator에 빌드된 신뢰성 기능에 영향을 미치며 업데이트가 수신되지 않습니다. 지원을 계속하려면 재정의를 제거한 후 보고된 문제를 재현해야 합니다.

2.3. 모니터링 스택 구성 준비

모니터링 구성 맵을 생성하고 업데이트하여 모니터링 스택을 구성할 수 있습니다.

2.3.1. 클러스터 모니터링 구성 맵 생성

주요 OpenShift Container Platform 모니터링 구성 요소를 구성하려면 **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 생성해야 합니다.

참고

cluster-monitoring-config ConfigMap 오브젝트에 대한 변경 사항을 저장하면 **openshift-monitoring** 프로젝트의 일부 또는 모든 Pod가 재배포될 수 있습니다. 이러한 구성 요소가 재배포되는 데 시간이 다소 걸릴 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. **cluster-monitoring-config ConfigMap** 오브젝트가 있는지 확인합니다.

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

2. **ConfigMap** 오브젝트가 없는 경우:

   a. 다음 YAML 매니페스트를 생성합니다. 이 예제에서는 파일은 **cluster-monitoring-config.yaml**이라고 합니다.

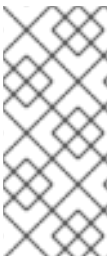
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
```

   b. **ConfigMap** 오브젝트를 생성하기 위해 구성을 적용합니다.

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

2.3.2. 사용자 정의 워크로드 모니터링 구성 맵 생성

사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하려면 **openshift -user-workload-monitoring**에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 생성해야 합니다.



참고

user-workload-monitoring-config ConfigMap 오브젝트에 대한 변경 사항을 저장하면 **openshift-user-workload-monitoring** 프로젝트의 일부 또는 모든 Pod가 재배포될 수 있습니다. 이러한 구성 요소가 재배포되는 데 시간이 다소 걸릴 수 있습니다. 먼저 사용자 정의 프로젝트에 대한 모니터링을 활성화하기 전에 구성 맵을 생성하고 구성할 수 있습니다. Pod를 재배포하지 않도록 합니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. **user-workload-monitoring-config ConfigMap** 오브젝트가 있는지 확인합니다.

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

2. **user-workload-monitoring-config ConfigMap** 오브젝트가 없는 경우:

   a. 다음 YAML 매니페스트를 생성합니다. 이 예제에서는 파일을 **user-workload-monitoring-config.yaml**이라고 합니다.

```
apiVersion: v1
```

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

b. **ConfigMap** 오브젝트를 생성하기 위해 구성을 적용합니다.

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

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

추가 리소스

- 사용자 정의 프로젝트 모니터링 활성화

2.4. 모니터링 스택 구성

OpenShift Container Platform 4.11에서는 **cluster-monitoring-config** 또는 **user-workload-monitoring-config ConfigMap** 오브젝트를 사용하여 모니터링 스택을 구성할 수 있습니다. 구성 맵은 CCMO(Cluster Monitoring Operator)를 구성한 다음 스택의 구성 요소를 구성합니다.

사전 요구 사항

- **핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. **ConfigMap** 오브젝트를 편집합니다.
   - **핵심 OpenShift Container Platform 모니터링 구성 요소를 구성하려면**다음을 수행합니다.
     a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

b. **data/config.yaml** 아래의 구성을 키-값 쌍 % **<component_name>: <component_configuration>**으로 추가합니다.

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

이에 따라 **<component>** 및 **<configuration_for_the_component>**를 바꿉니다.

다음 예제 **ConfigMap** 오브젝트는 Prometheus의 PVC(영구 볼륨 클레임)를 구성합니다. 이는 핵심 OpenShift Container Platform 구성 요소만 모니터링하는 Prometheus 인스턴스와 관련이 있습니다.

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

1 Prometheus 구성 요소를 정의하면 후속 행은 해당 구성을 정의합니다.

- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하려면**다음을 수행합니다.

a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

b. **data/config.yaml** 아래의 구성을 키-값 쌍 % **<component_name>: <component_configuration>**으로 추가합니다.

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

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

이에 따라 **<component>** 및 **<configuration_for_the_component>**를 바꿉니다.

다음 예제 **ConfigMap** 오브젝트는 Prometheus에 대한 데이터 보존 기간 및 최소 컨테이너 리소스 요청을 구성합니다. 이는 사용자 정의 프로젝트만 모니터링하는 Prometheus 인스턴스와 관련이 있습니다.

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

1. Prometheus 구성 요소를 정의하면 후속 행은 해당 구성을 정의합니다.
2. 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스에 대해 24시간 데이터 보존 기간을 구성합니다.
3. Prometheus 컨테이너에 대한 200밀리코어의 최소 리소스 요청을 정의합니다.
4. Prometheus 컨테이너에 대한 메모리 2GiB의 최소 Pod 리소스 요청을 정의합니다.

참고

Prometheus 구성 맵 구성 요소는 **cluster-monitoring-config ConfigMap** 오브젝트에서 **prometheusK8s**라고 하며 **user-workload-monitoring-config ConfigMap** 오브젝트에서 **prometheus**라고 합니다.

2. 파일을 저장하여 **ConfigMap** 오브젝트에 대한 변경 사항을 적용합니다. 새 구성의 영향을 받는 Pod가 자동으로 다시 시작됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화

2.5. 구성 가능한 모니터링 구성 요소

이 표는 구성할 수 있는 모니터링 구성 요소와 **cluster-monitoring-config** 및 **user-workload-monitoring-config ConfigMap** 오브젝트에서 구성 요소를 지정하는 데 사용하는 키를 보여줍니다.

표 2.1. 구성 가능한 모니터링 구성 요소

구성 요소	cluster-monitoring-config 구성 맵 키	user-workload-monitoring-config 구성 맵 키
Prometheus Operator	**prometheusOperator**	**prometheusOperator**
Prometheus	**prometheusK8s**	**prometheus**
Alertmanager	**alertmanagerMain**	**alertmanager**
kube-state-metrics	**kubeStateMetrics**	
openshift-state-metrics	**openshiftStateMetrics**	
Telemeter Client	**telemeterClient**	
Prometheus Adapter	**k8sPrometheusAdapter**	
Thanos Querier	**thanosQuerier**	
Thanos Ruler		**thanosRuler**

참고

Prometheus 키는 **cluster-monitoring-config ConfigMap** 오브젝트에서 **prometheusK8s**라고 하며 **user- workload-monitoring-config ConfigMap** 오브젝트에서 **prometheus**라고 합니다.

2.6. 노드 선택기를 사용하여 모니터링 구성 요소 이동

라벨이 지정된 노드와 함께 **nodeSelector** 제약 조건을 사용하면 모니터링 스택 구성 요소 중 하나를 특정 노드로 이동할 수 있습니다. 이렇게 하면 클러스터 전체에서 모니터링 구성 요소의 배치 및 배포를 제어할 수 있습니다.

모니터링 구성 요소의 배치 및 배포를 제어하여 시스템 리소스 사용을 최적화하고, 성능을 개선하며, 특정 요구 사항 또는 정책에 따라 워크로드를 분리할 수 있습니다.

2.6.1. 노드 선택기가 다른 제약 조건에서 작동하는 방법

노드 선택기 제약 조건을 사용하여 모니터링 구성 요소를 이동하는 경우 클러스터에 대한 Pod 예약을 제어하는 다른 제약 조건이 존재할 수 있습니다.

- Pod 배치를 제어하기 위해 토폴로지 분배 제약 조건이 적용될 수 있습니다.
- Prometheus, Thanos Querier, Alertmanager 및 기타 모니터링 구성 요소에 대해 하드 유사성 방지 규칙이 적용되어 이러한 구성 요소의 여러 Pod가 항상 다른 노드에 분배되므로 항상 가용성이 높아집니다.

노드에 Pod를 예약할 때 Pod 스케줄러는 Pod 배치를 결정할 때 기존 제약 조건을 모두 충족합니다. 즉, Pod 스케줄러가 어떤 노드에 배치될 Pod를 결정할 때 모든 제약 조건이 혼합됩니다.

따라서 노드 선택기 제약 조건을 구성하지만 기존 제약 조건을 모두 충족할 수 없는 경우 Pod 스케줄러는 모든 제약 조건과 일치시킬 수 없으며 노드에 배치할 Pod를 예약하지 않습니다.

모니터링 구성 요소에 대한 탄력성과 고가용성을 유지하려면 충분한 노드를 사용할 수 있는지 확인하고 구성 요소를 이동하기 위해 노드 선택기 제약 조건을 구성할 때 모든 제약 조건과 일치하는지 확인합니다.

추가 리소스

- 노드에서 라벨을 업데이트하는 방법 이해
- 노드 선택기를 사용하여 특정 노드에 Pod 배치
- 유사성 및 유사성 방지 규칙을 사용하여 다른 Pod에 상대적인 Pod 배치
- Pod 토폴로지 분배 제약 조건을 사용하여 Pod 배치 제어
- 노드 선택기에 대한 Kubernetes 설명서

2.6.2. 다른 노드로 모니터링 구성 요소 이동

스택 구성 요소를 실행할 클러스터의 노드를 지정하려면 노드에 할당된 라벨과 일치하도록 구성 요소의 **ConfigMap** 오브젝트에서 **nodeSelector** 제약 조건을 구성합니다.

참고

예약된 기존 Pod에 노드 선택기 제약 조건을 직접 추가할 수 없습니다.

사전 요구 사항

- **핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

- **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.

- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. 아직 수행하지 않은 경우 모니터링 구성 요소를 실행할 노드에 라벨을 추가합니다.

```
$ oc label nodes <node-name> <node-label>
```

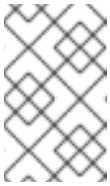
2. **ConfigMap** 오브젝트를 편집합니다.
   - **핵심 OpenShift Container Platform 프로젝트를 모니터링하는 구성 요소를 이동하려면**다음을 수행합니다.
     a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

     b. **data/config.yaml** 에서 구성 요소의 **nodeSelector** 제약 조건의 노드 레이블을 지정합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>: 1
      nodeSelector:
        <node-label-1> 2
        <node-label-2> 3
        <...>
```

1. &lt **;component&** gt;를 적절한 모니터링 스택 구성 요소 이름으로 바꿉니다.
2. & **lt;node-label-1** >을 노드에 추가한 레이블로 바꿉니다.
3. 선택 사항: 추가 라벨을 지정합니다. 추가 라벨을 지정하면 구성 요소의 Pod는 지정된 모든 라벨이 포함된 노드에만 예약됩니다.



참고

nodeSelector 제약 조건을 구성한 후 모니터링 구성 요소가 **Pending** 상태인 경우 테인트(Taints) 및 톨러레이션(Tolerations)과 관련된 오류에 대한 Pod 이벤트를 확인합니다.

- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 이동하려면**다음을 수행합니다.

  a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

  b. **data/config.yaml** 에서 구성 요소의 **nodeSelector** 제약 조건의 노드 레이블을 지정합니다.

  ```
  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. &lt **;component&** gt;를 적절한 모니터링 스택 구성 요소 이름으로 바꿉니다.
  2. & **lt;node-label-1** >을 노드에 추가한 레이블로 바꿉니다.
  3. 선택 사항: 추가 라벨을 지정합니다. 추가 라벨을 지정하면 구성 요소의 Pod는 지정된 모든 라벨이 포함된 노드에만 예약됩니다.

참고

nodeSelector 제약 조건을 구성한 후 모니터링 구성 요소가 **Pending** 상태인 경우 테인트(Taints) 및 톨러레이션(Tolerations)과 관련된 오류에 대한 Pod 이벤트를 확인합니다.

3. 파일을 저장하여 변경 사항을 적용합니다. 새 구성에 지정된 구성 요소가 새 노드로 자동 이동됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항을 저장하면 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화

2.7. 모니터링 구성 요소에 허용 오차 할당

모니터링 스택 구성 요소에 허용 오차를 할당하여 테인트 표시된 노드로 이동할 수 있습니다.

사전 요구 사항

- **핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. **ConfigMap** 오브젝트를 편집합니다.
   - **핵심 OpenShift Container Platform 프로젝트를 모니터링하는 구성 요소에 허용 오차를 할당하려면** 다음을 수행합니다.
     a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

     b. 구성 요소에 대한 **tolerations**를 지정합니다.

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

```
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    <component>:
      tolerations:
        <toleration_specification>
```

이에 따라 **<component>** 및 **<toleration_specification>**을 바꿉니다.

예를 들어 **oc adm taint nodes node1 key1=value1:NoSchedule**은 **key1**의 키와 **value1**의 값이 있는 **node1**에 테인트를 추가합니다. 이렇게 하면 해당 테인트에 허용 오차가 구성되지 않는 한 **node1**에 모니터링 구성 요소가 배포되지 않습니다. 다음 예제는 예제 테인트를 허용하도록 **alertmanagerMain** 구성 요소를 구성합니다.

```
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"
```

- **사용자 정의 프로젝트를 모니터링하는 구성 요소에 허용 오차를 할당하려면**다음을 수행합니다.

  a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

  b. 구성 요소에 대한 **tolerations**를 지정합니다.

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

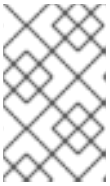
  이에 따라 **<component>** 및 **<toleration_specification>**을 바꿉니다.

  예를 들어 **oc adm taint nodes node1 key1=value1:NoSchedule**은 **key1**의 키와 **value1**의 값이 있는 **node1**에 테인트를 추가합니다. 이렇게 하면 해당 테인트에 허용 오

차가 구성되지 않는 한 **node1**에 모니터링 구성 요소가 배포되지 않습니다. 다음 예제는 예제 테인트를 허용하도록 **thanosRuler** 구성 요소를 구성합니다.

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

2. 파일을 저장하여 변경 사항을 적용합니다. 새로운 구성 요소 배치 구성이 자동으로 적용됩니다.



참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

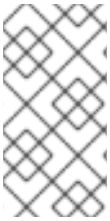
추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화
- 테인트(Taints) 및 톨러레이션(Tolerations)에 대한 OpenShift Container Platform 문서 를 참조하십시오.
- 테인트(Taints) 및 톨러레이션(Tolerations)에 대한 Kubernetes 문서를 참조하십시오.

2.8. 메트릭 스크랩에 대한 본문 크기 제한 설정

기본적으로 스크랩된 메트릭 대상에서 반환된 데이터의 압축되지 않은 본문 크기에 대한 제한이 없습니다. 스크랩된 대상이 대량의 데이터가 포함된 응답을 반환할 때 Prometheus가 과도한 메모리를 사용하는 상황을 방지하는 데 도움이 되도록 본문 크기 제한을 설정할 수 있습니다. 또한 본문 크기 제한을 설정하여 Prometheus 및 클러스터 전체에 악성 대상이 있을 수 있는 영향을 줄일 수 있습니다.

enforcedBodySizeLimit 의 값을 설정하면 구성된 값보다 하나 이상의 Prometheus 스크랩 대상 응답을 하나 이상 사용하면 **PrometheusScrapeBodySizeLimitHit** 경고 PrometheusScrapeBodySizeLimitHit이 실행됩니다.



참고

대상에서 스크랩한 메트릭 데이터에 구성된 크기보다 압축되지 않은 본문 크기가 있는 경우 스크랩이 실패합니다. 그런 다음 Prometheus는 이 대상이 down으로 간주되고 지표 값을 **0** 으로 설정하면 **TargetDown** 경고가 트리거될 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. **openshift-monitoring** 네임스페이스에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

2. 대상 스크랩별로 허용할 수 있는 본문 크기를 제한하려면 **enforcedBodySizeLimit** 값을 **data/config.yaml/prometheusK8s** 에 추가합니다.

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

(1) 스크랩된 지표 대상의 최대 본문 크기를 지정합니다. 이 **enforcedBodySizeLimit** 예제에서는 대상 스크랩당 압축되지 않은 크기를 **40MB**로 제한합니다. 유효한 숫자 값은 **Prometheus** 데이터 크기 형식을 사용합니다. **B**(바이트), **KB**(KB), **MB**(MB), **GB**(기가바이트), **TB**(terabytes), **PB**(petabytes), **WebAssembly**(exabytes)입니다. 기본값은 **0** 이며 제한이 없음을 지정합니다. 클러스터 용량에 따라 자동으로 제한을 계산하도록 값을 **automatic** 으로 설정할 수도 있습니다.

3. 파일을 저장하여 변경 사항을 자동으로 적용합니다.

주의

cluster-monitoring-config 구성 맵에 대한 변경 사항을 저장하면 **openshift-monitoring** 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

추가 리소스

- Prometheus scrape 구성 문서

2.9. 전용 서비스 모니터 구성

전용 서비스 모니터를 사용하여 리소스 지표 파이프라인에 대한 지표를 수집하도록 **OpenShift Container Platform** 코어 플랫폼 모니터링을 구성할 수 있습니다.

활성화하면 전용 서비스 모니터에서 **kubelet** 끝점에서 두 개의 추가 메트릭을 노출하고 **honorTimestamps** 필드 값을 **true**로 설정합니다.

전용 서비스 모니터를 활성화하면 **oc adm top pod** 명령 또는 **Horizontal Pod Autoscaler**에서 사용하는 **Prometheus Adapter** 기반 **CPU** 사용량 측정의 일관성을 개선할 수 있습니다.

2.9.1. 전용 서비스 모니터 활성화

openshift-monitoring 네임스페이스의 **cluster-monitoring-config ConfigMap** 오브젝트에서 **dedicatedServiceMonitors** 키를 구성하여 전용 서비스 모니터를 사용하도록 코어 플랫폼 모니터링을 구성할 수 있습니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.

프로세스

1. openshift-monitoring 네임스페이스에서 cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. 다음 샘플에 표시된 대로 enabled: true key-value 쌍을 추가합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    k8sPrometheusAdapter:
      dedicatedServiceMonitors:
        enabled: true 1
```

1 kubelet /metrics/resource 끝점을 노출하는 전용 서비스 모니터를 배포하려면 enabled 필드의 값을 true 로 설정합니다.

3. 파일을 저장하여 변경 사항을 자동으로 적용합니다.

주의

cluster-monitoring-config 구성 맵에 대한 변경 사항을 저장하면 openshift-monitoring 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

2.10. 영구 스토리지 구성

영구 스토리지로 클러스터 모니터링을 실행하면 메트릭이 PV(영구 볼륨)에 저장되며 Pod를 다시 시작하거나 재생성할 수 있습니다. 데이터 손실에서 메트릭 또는 경고 데이터가 필요한 경우 이상적입니다. 프로덕션 환경의 경우 영구 스토리지를 구성하는 것이 매우 좋습니다. 높은 IO 요구로 인해 로컬 스토리지를 사용하는 것이 이점이 됩니다.

2.10.1. 영구 스토리지 사전 요구 사항

- 디스크가 가득 차지 않도록 충분한 로컬 영구 스토리지를 전용으로 지정합니다. 필요한 스토리지의 양은 Pod 수에 따라 달라집니다.
- 각 복제본에 대해 하나의 PV(영구 볼륨 클레임)에서 PV(영구 볼륨)를 클레임할 준비가 되어 있는지 확인합니다. Prometheus 및 Alertmanager에는 두 개의 복제본이 있으므로 전체 모니터링 스택을 지원하는 데 4개의 PV가 필요합니다. Local Storage Operator에서 PV를 사용할 수 있지만 동적으로 프로비저닝된 스토리지를 활성화한 경우에는 사용할 수 없습니다.
- 영구 볼륨을 구성할 때 Filesystem 을 volumeMode 매개변수의 스토리지 유형 값으로 사용합니다.

참고

영구 스토리지에 로컬 볼륨을 사용하는 경우 LocalVolume 오브젝트에서 volumeMode: Block 에 설명된 원시 블록 볼륨을 사용하지 마십시오. Prometheus는 원시 블록 볼륨을 사용할 수 없습니다.

중요

Prometheus는 POSIX 호환이 아닌 파일 시스템을 지원하지 않습니다. 예를 들어 일부 NFS 파일 시스템 구현은 POSIX와 호환되지 않습니다. 스토리지를 위해 NFS 파일 시스템을 사용하려면 공급 업체에 NFS 구현이 완전히 POSIX와 호환되는지 확인하십시오.

2.10.2. 로컬 영구 볼륨 클레임 구성

PV(영구 볼륨)를 사용하기 위한 구성 요소를 모니터링하는 경우 PVC(영구 볼륨 클레임)를 구성해야 합니다.

사전 요구 사항

- 핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:
  - cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - user-workload-monitoring-config ConfigMap 오브젝트가 생성되어 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

프로세스

1. ConfigMap 오브젝트를 편집합니다.
   - 핵심 OpenShift Container Platform 프로젝트를 모니터링하는 구성 요소의 PVC를 구성하려면 다음을 수행합니다.
     a. openshift-monitoring 프로젝트에서 cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

     b. data/config.yaml 아래의 구성 요소에 대한 PVC 구성을 추가합니다.

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

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

volumeClaimTemplate을 지정하는 방법에 대한 내용은 PersistentVolumeClaims에 대한 Kubernetes 문서를 참조하십시오.

다음 예제는 핵심 OpenShift Container Platform 구성 요소를 모니터링하는 Prometheus 인스턴스의 로컬 영구 스토리지를 요청하는 PVC를 구성합니다.

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

위의 예에서 Local Storage Operator에 의해 생성된 스토리지 클래스를 local-storage라고 합니다.

다음 예제는 Alertmanager에 대한 로컬 영구 스토리지를 요청하는 PVC를 구성합니다.

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

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

- 사용자 정의 프로젝트를 모니터링하는 구성 요소의 PVC를 구성하려면 다음을 수행합니다.

  a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

  b. data/config.yaml 아래의 구성 요소에 대한 PVC 구성을 추가합니다.

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

volumeClaimTemplate을 지정하는 방법에 대한 내용은 PersistentVolumeClaims에 대한 Kubernetes 문서를 참조하십시오.

다음 예제는 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스의 로컬 영구 스토리지를 요청하는 PVC를 구성합니다.

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

```
    prometheus:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

위의 예에서 **Local Storage Operator**에 의해 생성된 스토리지 클래스를 **local-storage**라고 합니다.

다음 예제에서는 **Thanos Ruler**의 로컬 영구 스토리지를 요청하는 **PVC**를 구성합니다.

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

참고

thanosRuler 구성 요소의 스토리지 요구 사항은 평가되는 규칙 수와 각 규칙이 생성하는 샘플 수에 따라 달라집니다.

2. 파일을 저장하여 변경 사항을 적용합니다. 새 구성의 영향을 받는 **Pod**가 자동으로 다시 시작되고 새 스토리지 구성이 적용됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

2.10.3. 영구 스토리지 볼륨 크기 조정

OpenShift Container Platform은 기본 **StorageClass** 리소스가 영구 볼륨 크기 조정을 지원하는 경우에도 **StatefulSet** 리소스에서 사용하는 기존 영구 스토리지 볼륨 크기 조정을 지원하지 않습니다. 따라서 더 큰 크기의 기존 **PVC**(영구 볼륨 클레임)의 스토리지 필드를 업데이트해도 이 설정은 연결된 **PV**(영구 볼륨)로 전파되지 않습니다.

그러나 수동 프로세스를 사용하면 **PV**의 크기를 조정할 수 있습니다. **Prometheus, Thanos Ruler** 또는 **Alertmanager**와 같은 모니터링 구성 요소의 **PV**의 크기를 조정하려면 구성 요소가 구성된 적절한 구성 맵을 업데이트할 수 있습니다. 그런 다음 **PVC**에 패치를 적용하고 **pod**를 삭제하고 분리합니다. 포드를 고립하면 **StatefulSet** 리소스가 즉시 다시 생성되고 새 **PVC** 설정으로 **Pod**에 마운트된 볼륨의 크기를 자동으로 업데이트합니다. 이 과정에서 서비스 중단이 발생하지 않습니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 핵심 **OpenShift Container Platform** 모니터링 구성 요소인 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
  - 핵심 **OpenShift Container Platform** 모니터링 구성 요소를 위해 하나 이상의 **PVC**를 구성했습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:

- cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- user-workload-monitoring-config ConfigMap 오브젝트가 생성되어 있습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소에 대해 하나 이상의 PVC를 구성했습니다.

절차

1. ConfigMap 오브젝트를 편집합니다.

- 핵심 OpenShift Container Platform 프로젝트를 모니터링하는 구성 요소의 PVC의 크기를 조정하려면 다음을 수행합니다.

a. openshift-monitoring 프로젝트에서 cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b. data/config.yaml 아래의 구성 요소에 대한 PVC 구성에 대한 새 스토리지 크기를 추가합니다.

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

1

코어 모니터링 구성 요소를 지정합니다.

2

스토리지 클래스를 지정합니다.

3

스토리지 볼륨의 새 크기를 지정합니다.

다음 예제에서는 핵심 **OpenShift Container Platform** 구성 요소를 모니터링하는 **Prometheus** 인스턴스의 로컬 영구 스토리지를 **100GB**로 설정하는 **PVC**를 구성합니다.

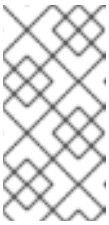
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: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

다음 예제에서는 **Alertmanager**의 로컬 영구 스토리지를 **40GB**로 설정하는 **PVC**를 구성합니다.

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

-

사용자 정의 프로젝트를 모니터링하는 구성 요소의 PVC의 크기를 조정하려면 다음을 수행합니다.



참고

사용자 정의 프로젝트를 모니터링하는 Thanos Ruler 및 Prometheus 인스턴스의 볼륨의 크기를 조정할 수 있습니다.

a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b. data/config.yaml 에서 모니터링 구성 요소에 대한 PVC 구성을 업데이트합니다.

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

1 코어 모니터링 구성 요소를 지정합니다.

2 스토리지 클래스를 지정합니다.

3 스토리지 볼륨의 새 크기를 지정합니다.

다음 예제는 사용자 정의 프로젝트를 모니터링하는 **Prometheus** 인스턴스의 **PVC** 크기를 **100GB**로 구성합니다.

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

다음 예제에서는 **Thanos Ruler**의 **PVC** 크기를 **20GB**로 설정합니다.

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

참고

thanosRuler 구성 요소의 스토리지 요구 사항은 평가되는 규칙 수와 각 규칙이 생성하는 샘플 수에 따라 달라집니다.

2. 파일을 저장하여 변경 사항을 적용합니다. 새 구성의 영향을 받는 **Pod**가 자동으로 다시 시작됩니다.

주의

모니터링 구성 맵에 변경 사항을 저장하면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

3. 업데이트된 스토리지 요청으로 모든 **PVC**를 수동으로 패치합니다. 다음 예제는 **openshift-monitoring** 네임스페이스의 **Prometheus** 구성 요소의 스토리지 크기를 **100Gi**로 조정합니다.

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

4. **--cascade=orphan** 매개변수를 사용하여 기본 **StatefulSet**을 삭제합니다.

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

2.10.4. Prometheus 메트릭 데이터의 보존 시간 및 크기 수정

기본적으로 **Prometheus**는 지표 데이터를 **15**일 동안 자동으로 유지합니다. 보존 시간에는 보존 필드에 시간 값을 지정하여 데이터가 삭제되는 시간을 변경할 수 있습니다. **retentionSize** 필드에 크기 값을 지정하여 보존 지표 데이터가 사용하는 최대 디스크 공간을 구성할 수도 있습니다. 데이터가 이 크기 제한에 도달하면 **Prometheus**는 사용된 디스크 공간이 제한보다 다시 될 때까지 가장 오래된 데이터를 먼저 삭제합니다.

이러한 데이터 보존 설정의 다음 동작에 유의하십시오.

- 크기 기반 보존 정책은 영구 블록, **WAL(Write-ahead log)** 데이터, **m-mapped** 청크를 포함하여 **/prometheus** 디렉토리의 모든 데이터 블록 디렉터리에 적용됩니다.

- **/wal** 및 **/head_chunks** 디렉터리의 데이터는 보존 크기 제한에 반영되지만 **Prometheus**는 크기 또는 시간 기반 보존 정책에 따라 해당 디렉터리에서 데이터를 제거하지 않습니다. 따라서 **/wal** 및 **/head_chunks** 디렉토리에 대해 설정된 최대 크기보다 낮은 보존 크기 제한을 설정하는 경우 **/prometheus** 데이터 디렉터리에 데이터 블록을 유지하지 않도록 시스템을 구성했습니다.

- 크기 기반 보존 정책은 Prometheus가 새 데이터 블록을 잘라내어 WAL에 3시간 이상의 데이터가 포함된 2시간마다 실행되는 경우에만 적용됩니다.

- retention 또는 retentionSize 에 대한 값을 명시적으로 정의하지 않으면 보존 시간은 기본적으로 15일로 설정되고 보존 크기는 설정되지 않습니다.

- retention 및 retentionSize 에 대한 값을 모두 정의한 경우 두 값이 모두 적용됩니다. 데이터 블록이 정의된 보존 시간 또는 정의된 크기 제한을 초과하는 경우 Prometheus는 이러한 데이터 블록을 제거합니다.

- retentionSize 값을 정의하고 보존을 정의하지 않으면 retention Size 값만 적용됩니다.

- retention Size 값을 정의하지 않고 보존 값만 정의하는 경우 보존 값만 적용됩니다.

사전 요구 사항

- 핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:
  - cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - 클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
  - cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - user-workload-monitoring-config ConfigMap 오브젝트가 생성되어 있습니다.

- **OpenShift CLI(oc)**가 설치되어 있습니다.

주의

모니터링 구성 맵에 대한 변경 사항을 저장하면 모니터링 프로세스를 다시 시작하고 관련 프로젝트에서 **Pod** 및 기타 리소스를 재배포할 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

절차

1. **ConfigMap** 오브젝트를 편집합니다.

- 핵심 **OpenShift Container Platform** 프로젝트를 모니터링하는 **Prometheus** 인스턴스의 보존 시간과 크기를 수정하려면 다음을 수행합니다.

a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

b. **data/config.yaml** 아래에 보존 시간 및 크기 구성을 추가합니다.

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

1 보존 시간: **ms** (밀리초), **s** (초), **m** (분), **h** (시간), **d** (일), **w** (주) 또는 **y** (년)가 직접 따르는 숫자입니다. 또한 **1h30m15s** 와 같은 특정 시간에 시간 값을 결합 할 수도 있습니다.

2

보존 크기: B (바이트), KB (KB), MB(MB), GB(GB), GB(GB), GB(GB), PB (비바이트), PAB(exabytes)가 바로 뒤에 오는 숫자입니다.

다음 예제에서는 핵심 OpenShift Container Platform 구성 요소를 모니터링하는 Prometheus 인스턴스의 보존 시간을 24시간, 보존 크기를 10GB로 설정합니다.

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

- 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스의 보존 시간과 크기를 수정하려면 다음을 수행합니다.

a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b. data/config.yaml 아래에 보존 시간 및 크기 구성을 추가합니다.

```
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

보존 시간: ms (밀리초), s (초), m (분), h (시간), d (일), w (주) 또는 y (년)가 직접 따르는 숫자입니다. 또한 1h30m15s 와 같은 특정 시간에 시간 값을 결합 할

수도 있습니다.

2

보존 크기: B (바이트), KB (KB), MB(MB), GB(GB), GB(기가바이트), GB (GB), PB(비바이트), PB (exabytes)가 바로 뒤에 오는 숫자입니다.

다음 예제에서는 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스의 보존 시간을 24시간, 보존 크기를 10GB로 설정합니다.

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

2. 파일을 저장하여 변경 사항을 적용합니다. 새 구성의 영향을 받는 Pod가 자동으로 다시 시작됩니다.

2.10.5. Thanos Ruler 메트릭 데이터의 보존 시간 수정

기본적으로 사용자 정의 프로젝트의 경우 Thanos Ruler는 메트릭 데이터를 24시간 동안 자동으로 유지합니다. openshift-user-workload-monitoring 네임스페이스의 user-workload-monitoring-config 구성 맵에 시간 값을 지정하여 이 데이터가 유지되는 기간을 변경하도록 보존 시간을 수정할 수 있습니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.
- 클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.

- user-workload-monitoring-config ConfigMap 오브젝트가 생성되어 있습니다.

주의

모니터링 구성 맵에 대한 변경 사항을 저장하면 모니터링 프로세스를 다시 시작하고 관련 프로젝트에서 Pod 및 기타 리소스를 재배포할 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

절차

1. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. data/config.yaml 아래에 보존 시간 구성을 추가합니다.

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

1

보존 시간을 ms (밀리초), s (초), m (분), h (시간), d (일), w (주) 또는 y (년)로 바로 따라 지정합니다. 또한 1h30m15s 와 같은 특정 시간에 시간 값을 결합 할 수도 있습니다. 기본 값은 24h 입니다.

다음 예제에서는 Thanos Ruler 데이터에 대해 보존 시간을 10일로 설정합니다.

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

```
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      retention: 10d
```

3. 파일을 저장하여 변경 사항을 적용합니다. 새 구성의 영향을 받는 **Pod**가 자동으로 다시 시작됩니다.

추가 리소스

- 클러스터 모니터링 구성 맵 생성
- **Prometheus** 데이터베이스 스토리지 요구사항
- 권장되는 구성 가능한 스토리지 기술
- 영구저장장치 이해
- 스토리지 최적화
- 로컬 영구 스토리지 구성
- 사용자 정의 프로젝트 모니터링 활성화

2.11. 원격 쓰기 스토리지 구성

Prometheus가 장기 스토리지의 원격 시스템에 인가된 지표를 보낼 수 있도록 원격 쓰기 스토리지를 구성할 수 있습니다. 이렇게 하면 **Prometheus**가 메트릭을 저장하는 방법 또는 길이에 영향을 미치지 않습니다.

사전 요구 사항

- 핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:
  - cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - user-workload-monitoring-config ConfigMap 오브젝트가 생성되어 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.
- Thanos와 같은 원격 쓰기 호환 엔드포인트를 설정하고 엔드포인트 URL을 알고 있어야 합니다. 원격 쓰기 기능과 호환되는 엔드포인트에 대한 정보는 Prometheus 원격 엔드포인트 및 스토리지 설명서를 참조하십시오.
- 원격 쓰기 엔드 포인트의 Secret 오브젝트에 인증 인증 정보를 설정했습니다. 사용자 워크로드 모니터링을 위해 원격 쓰기를 구성하는 Prometheus 오브젝트와 동일한 네임스페이스에 시크릿을 생성해야 합니다. 기본 플랫폼 모니터링을 위해 openshift-monitoring 네임스페이스 또는 사용자 워크로드 모니터링을 위한 openshift-user-workload-monitoring 네임스페이스를 생성해야 합니다.

  경고

  보안 위험을 줄이려면 HTTPS 및 인증을 사용하여 메트릭을 엔드포인트에 보냅니다.

절차

다음 단계에 따라 openshift-monitoring 네임스페이스의 cluster-monitoring-config 구성 맵에서 기본 플랫폼 모니터링에 대한 원격 쓰기를 구성합니다.

참고

사용자 정의 프로젝트를 모니터링하는 **Prometheus** 인스턴스에 대한 원격 쓰기를 구성하는 경우 **openshift-user-workload-monitoring** 네임스페이스에서 **user-workload-monitoring-config** 구성 맵과 유사하게 편집합니다. **Prometheus** 구성 맵 구성 요소는 **cluster-monitoring-config ConfigMap** 오브젝트에 있으므로 **user-workload-monitoring-config ConfigMap** 오브젝트에서 **prometheus** 라고 합니다.

1. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

2. **data/config.yaml/prometheusK8s**에 **remoteWrite:** 섹션을 추가합니다.

3. 이 섹션에 엔드포인트 **URL** 및 인증 정보를 추가합니다.

```
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. 원격 쓰기 끝점의 **URL**입니다.

2. 끝점의 인증 방법 및 자격 증명입니다. 현재 지원되는 인증 방법은 **AWS Signature Version 4**입니다. 인증 요청 헤더에서 **HTTP**를 사용한 인증, 기본 인증, **OAuth 2.0** 및 **TLS** 클라이언트입니다. 지원되는 *인증 방법의 샘플 구성에 대한 지원 원격 쓰기 인증 설정*을 참조하십시오.

4. 인증 정보 뒤에 쓰기 재레이블 구성 값을 추가합니다.

```
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>
        <write_relabel_configs> 1
```

1

쓰기 레이블 설정입니다.

<write_relabel_configs>는 원격 엔드포인트로 보낼 메트릭의 쓰기 재레이블 구성 목록을 대체합니다.

다음 샘플은 my_metric이라는 단일 메트릭을 전달하는 방법을 보여줍니다.

```
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
```

쓰기 재레이블 구성 옵션에 대한 정보는 Prometheus relabel_config 설명서를 참조하십시오.

5. 파일을 저장하여 ConfigMap 오브젝트에 대한 변경 사항을 적용합니다. 새 구성의 영향을 받는 Pod가 자동으로 다시 시작됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 **ConfigMap** 오브젝트에 대한 변경 사항을 저장하면 관련 프로젝트에서 **Pod** 및 기타 리소스를 재배포할 수 있습니다. 변경 사항을 저장하면 해당 프로젝트에서 실행 중인 모니터링 를 다시 시작할 수도 있습니다.

2.11.1. 지원되는 원격 쓰기 인증 설정

다른 방법을 사용하여 원격 쓰기 엔드포인트로 인증할 수 있습니다. 현재 지원되는 인증 방법은 **AWS Signature Version 4, Basic authentication, Authorization** 요청 헤더, **OAuth 2.0** 및 **TLS** 클라이언트에서 **HTTP**를 사용한 인증입니다. 다음 표에서는 원격 쓰기에 사용되는 지원되는 인증 방법에 대해 자세히 설명합니다.

인증 방법	구성 맵 필드	설명
AWS Signature Version 4	**sigv4**	이 방법은 AWS Signature Version 4 인증을 사용하여 요청을 서명합니다. 이 방법은 권한 부여, OAuth 2.0 또는 기본 인증과 동시에 사용할 수 없습니다.
기본 인증	**basicAuth**	기본 인증은 구성된 사용자 이름 및 암호로 모든 원격 쓰기 요청에 권한 부여 헤더를 설정합니다.
권한 부여	**권한 부여**	권한 부여는 구성된 토큰을 사용하여 모든 원격 쓰기 요청에 **Authorization** 헤더를 설정합니다.

인증 방법	구성 맵 필드	설명
OAuth 2.0	**oauth2**	OAuth 2.0 구성에서는 클라이언트 자격 증명 부여 유형을 사용합니다. Prometheus는 지정된 클라이언트 ID 및 클라이언트 시크릿을 사용하여 **tokenUrl** 에서 원격 쓰기 엔드포인트에 액세스 토큰을 가져옵니다. 이 방법은 권한 부여, AWS Signature 버전 4 또는 기본 인증과 동시에 사용할 수 없습니다.
TLS 클라이언트	**tlsConfig**	TLS 클라이언트 구성은 TLS를 사용하여 원격 쓰기 엔드포인트 서버에서 인증하는 데 사용되는 CA 인증서, 클라이언트 인증서, 클라이언트 키 정보를 지정합니다. 샘플 구성에서는 CA 인증서 파일, 클라이언트 인증서 파일 및 클라이언트 키 파일을 이미 생성했다고 가정합니다.

2.11.1.1. 인증 설정의 구성 맵 위치

다음은 기본 플랫폼 모니터링을 위한 **ConfigMap** 오브젝트에서 인증 구성의 위치를 보여줍니다.

```
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_details> 2
```

1

원격 쓰기 끝점의 **URL**입니다.

2

엔드포인트의 인증 방법에 필요한 구성 세부 정보입니다. 현재 지원되는 인증 방법은 **Amazon Web Services(AWS)** 서명 버전 **4**로, 인증 요청 헤더, 기본 인증, **OAuth 2.0** 및 **TLS** 클라이언트에서 **HTTP**를 사용하는 인증입니다.

참고

사용자 정의 프로젝트를 모니터링하는 **Prometheus** 인스턴스에 대한 원격 쓰기를 구성하는 경우 **openshift-user-workload-monitoring** 네임스페이스에서 **user-workload-monitoring-config** 구성 맵을 편집합니다. **Prometheus** 구성 맵 구성 요소는 **cluster-monitoring-config ConfigMap** 오브젝트에 있으므로 **user-workload-monitoring-config ConfigMap** 오브젝트에서 **prometheus** 라고 합니다.

2.11.1.2. 원격 쓰기 인증 설정 예

다음 샘플은 원격 쓰기 끝점에 연결하는 데 사용할 수 있는 다른 인증 설정을 보여줍니다. 각 샘플에서는 인증 인증 정보 및 기타 관련 설정이 포함된 해당 **Secret** 오브젝트를 구성하는 방법도 보여줍니다. 각 샘플은 **openshift-monitoring** 네임스페이스에서 기본 플랫폼 모니터링과 함께 사용할 수 있도록 인증을 구성합니다.

AWS Signature Version 4 인증을 위한 샘플 **YAML**

다음은 **openshift-monitoring** 네임스페이스의 **sigv4-credentials** 라는 **sigv4** 시크릿의 설정을 보여줍니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: sigv4-credentials
  namespace: openshift-monitoring
stringData:
  accessKey: <AWS_access_key> 1
  secretKey: <AWS_secret_key> 2
type: Opaque
```

1

AWS API 액세스 키입니다.

2

AWS API 시크릿 키입니다.

다음은 **openshift-monitoring** 네임스페이스에서 **sigv4-credentials** 라는 **Secret** 오브젝트를 사용하는 샘플 **AWS Signature Version 4** 원격 쓰기 인증 설정을 보여줍니다.

```
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> 1
          accessKey:
            name: sigv4-credentials 2
            key: accessKey 3
          secretKey:
            name: sigv4-credentials 4
            key: secretKey 5
          profile: <AWS_profile_name> 6
          roleArn: <AWS_role_arn> 7
```

1 AWS 리전입니다.

2 4 AWS API 액세스 인증 정보가 포함된 Secret 오브젝트의 이름입니다.

3 지정된 Secret 오브젝트에 AWS API 액세스 키가 포함된 키입니다.

5 지정된 Secret 오브젝트에 AWS API 시크릿 키가 포함된 키입니다.

6 인증에 사용되는 AWS 프로필의 이름입니다.

7 역할에 할당된 Amazon Resource Name(ARN)의 고유 식별자입니다.

기본 인증을 위한 YAML 샘플

다음은 openshift-monitoring 네임스페이스에서 rw-basic-auth 라는 Secret 오브젝트에 대한 기본 인증 설정을 보여줍니다.

```
apiVersion: v1
```

```
kind: Secret
metadata:
  name: rw-basic-auth
  namespace: openshift-monitoring
stringData:
  user: <basic_username> 1
  password: <basic_password> 2
type: Opaque
```

1 사용자 이름.

2 암호입니다.

다음 샘플은 **openshift-monitoring** 네임스페이스에서 **rw-basic-auth** 라는 **Secret** 오브젝트를 사용하는 **basicAuth** 원격 쓰기 구성을 보여줍니다. 끝점에 대한 인증 자격 증명을 이미 설정했다고 가정합니다.

```
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 1
            key: user 2
          password:
            name: rw-basic-auth 3
            key: password 4
```

1 3 인증 인증 정보가 포함된 **Secret** 오브젝트의 이름입니다.

2 지정된 **Secret** 오브젝트에 사용자 이름이 포함된 키입니다.

4

Secret 오브젝트를 사용하여 전달자 토큰을 사용한 인증을 위한 **YAML** 샘플

다음은 **openshift-monitoring** 네임스페이스의 **rw-bearer-auth** 라는 **Secret** 오브젝트에 대한 전달자 토큰 설정을 보여줍니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: rw-bearer-auth
  namespace: openshift-monitoring
stringData:
  token: <authentication_token> 1
type: Opaque
```

1

인증 토큰입니다.

다음은 **openshift-monitoring** 네임스페이스에서 **rw-bearer-auth** 라는 **Secret** 오브젝트를 사용하는 샘플 전달자 토큰 구성 맵 설정을 보여줍니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true
    prometheusK8s:
      remoteWrite:
      - url: "https://authorization.example.com/api/write"
        authorization:
          type: Bearer 1
          credentials:
            name: rw-bearer-auth 2
            key: token 3
```

1

요청의 인증 유형입니다. 기본값은 **Bearer** 입니다.

2

3

지정된 **Secret** 오브젝트에 인증 토큰이 포함된 키입니다.

OAuth 2.0 인증을 위한 샘플 **YAML**

다음은 **openshift-monitoring** 네임스페이스에서 **oauth2-credentials** 라는 **Secret** 오브젝트에 대한 샘플 **OAuth 2.0** 설정을 보여줍니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: oauth2-credentials
  namespace: openshift-monitoring
stringData:
  id: <oauth2_id> 1
  secret: <oauth2_secret> 2
  token: <oauth2_authentication_token> 3
type: Opaque
```

1

Oauth 2.0 ID입니다.

2

OAuth 2.0 시크릿입니다.

3

OAuth 2.0 토큰입니다.

다음은 **openshift-monitoring** 네임스페이스에서 **oauth2-credentials** 라는 **Secret** 오브젝트를 사용하는 **oauth2** 원격 쓰기 인증 샘플 구성을 보여줍니다.

```
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 (1)
            key: id (2)
        clientSecret:
          name: oauth2-credentials (3)
          key: secret (4)
        tokenUrl: https://example.com/oauth2/token (5)
        scopes: (6)
        - <scope_1>
        - <scope_2>
        endpointParams: (7)
          param1: <parameter_1>
          param2: <parameter_2>
```

(1) (3)

해당 **Secret** 오브젝트의 이름입니다. **clientSecret** 은 **Secret** 오브젝트를 참조해야 하지만 **ClientId** 는 **ConfigMap** 오브젝트를 참조할 수 있습니다.

(2) (4)

지정된 **Secret** 오브젝트에 **OAuth 2.0** 자격 증명이 포함된 키입니다.

(5)

지정된 **clientId** 및 **clientSecret** 을 사용하여 토큰을 가져오는 데 사용되는 **URL**입니다.

(6)

권한 부여 요청의 **OAuth 2.0** 범위입니다. 이러한 범위는 토큰이 액세스할 수 있는 데이터를 제한합니다.

(7)

권한 부여 서버에 필요한 **OAuth 2.0** 인증 요청 매개 변수입니다.

TLS 클라이언트 인증을 위한 **YAML** 샘플

다음은 **openshift-monitoring** 네임스페이스에 **m tls -bundle** 이라는 **tls Secret** 오브젝트에 대한 샘플 **TLS** 클라이언트 설정을 보여줍니다.

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

```
  name: mtls-bundle
  namespace: openshift-monitoring
data:
  ca.crt: <ca_cert> 1
  client.crt: <client_cert> 2
  client.key: <client_key> 3
type: tls
```

1 서버 인증서를 검증할 **Prometheus** 컨테이너의 **CA** 인증서입니다.

2 서버를 사용하여 인증을 위한 클라이언트 인증서입니다.

3 클라이언트 키입니다.

다음 샘플은 **mtls-bundle** 이라는 **TLS Secret** 오브젝트를 사용하는 **tlsConfig** 원격 쓰기 인증 구성을 보여줍니다.

```
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: mtls-bundle 1
              key: ca.crt 2
          cert:
            secret:
              name: mtls-bundle 3
              key: client.crt 4
          keySecret:
            name: mtls-bundle 5
            key: client.key 6
```

1 3 5

2

끝점의 CA 인증서가 포함된 지정된 Secret 오브젝트의 키입니다.

4

끝점의 클라이언트 인증서가 포함된 지정된 Secret 오브젝트의 키입니다.

6

클라이언트 키 시크릿을 포함하는 지정된 Secret 오브젝트의 키입니다.

추가 리소스

- 원격 쓰기 호환 엔드포인트 (예: Thanos)을 생성하는 단계는 원격 쓰기 호환 엔드포인트 설정을 참조하십시오.
- 다양한 사용 사례에 맞게 원격 쓰기 설정을 최적화하는 방법에 대한 자세한 내용은 원격 쓰기 설정 튜닝을 참조하십시오.
- OpenShift Container Platform에서 Secret 오브젝트를 생성하고 구성하는 단계는 시크릿 이해를 참조하십시오.
- 추가 선택적 필드에 대한 자세한 내용은 원격 쓰기에 대한 Prometheus REST API 참조 를 참조하십시오.

2.12. 메트릭에 클러스터 ID 라벨 추가

여러 OpenShift Container Platform 클러스터를 관리하고 원격 쓰기 기능을 사용하여 이러한 클러스터에서 외부 스토리지 위치로 지표 데이터를 전송하는 경우 클러스터 ID 레이블을 추가하여 다른 클러스터에서 들어오는 지표 데이터를 확인할 수 있습니다. 그런 다음 이러한 라벨을 쿼리하여 지표에 대해 소스 클러스터를 식별하고 해당 데이터를 다른 클러스터에서 보낸 유사한 메트릭 데이터와 구별할 수 있습니다.

이렇게 하면 여러 고객에게 여러 클러스터를 관리하고 지표 데이터를 단일 중앙 집중식 스토리지 시스템에 전송하는 경우 클러스터 ID 레이블을 사용하여 특정 클러스터 또는 고객의 메트릭을 쿼리할 수 있습니다.

클러스터 ID 라벨을 생성하고 사용하는 데는 세 가지 일반적인 단계가 있습니다.

- 원격 쓰기 스토리지의 쓰기 레이블 설정 구성.
- 메트릭에 클러스터 ID 레이블 추가.
- 이러한 라벨을 쿼리하여 메트릭의 소스 클러스터 또는 고객을 식별합니다.

2.12.1. 메트릭의 클러스터 ID 라벨 생성

기본 플랫폼 모니터링 및 사용자 워크로드 모니터링에 대한 메트릭에 대한 클러스터 ID 레이블을 생성할 수 있습니다.

기본 플랫폼 모니터링의 경우 **openshift-monitoring** 네임스페이스의 **cluster-monitoring-config** 구성 맵의 원격 쓰기 스토리지에 대한 **write_relabel** 설정의 지표에 클러스터 ID 라벨을 추가합니다.

사용자 워크로드 모니터링의 경우 **openshift-user-workload-monitoring** 네임스페이스의 **user-workload-monitoring-config** 구성 맵의 설정을 편집합니다.

참고

Prometheus가 네임스페이스 레이블을 노출하는 사용자 워크로드 대상을 스크랩하면 시스템은 이 라벨을 **exported_namespace** 로 저장합니다. 이 동작을 수행하면 최종 네임스페이스 레이블 값이 대상 **Pod**의 네임스페이스와 동일합니다. **PodMonitor** 또는 **ServiceMonitor** 오브젝트에 대해 **honorLabels** 필드의 값을 **true** 로 설정하여 이 기본 구성을 덮어쓸 수 없습니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 원격 쓰기 스토리지를 구성했습니다.

- 기본 플랫폼 모니터링 구성 요소를 구성하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.

프로세스

1. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

참고

사용자 정의 프로젝트를 모니터링하는 **Prometheus** 인스턴스에 대한 클러스터 **ID** 레이블을 구성하는 경우 **openshift-user-workload-monitoring** 네임스페이스에서 **user-workload-monitoring-config** 구성 맵을 편집합니다. **Prometheus** 구성 요소는 이 구성 맵에서 **prometheus** 라고 하며 **prometheusK8s** 가 아닌 **cluster-monitoring-config** 구성 맵에 사용되는 이름입니다.

2. **data/config.yaml/prometheusK8s/remoteWrite** 의 **writeRelabelConfigs:** 섹션에서 클러스터 **ID** 레이블을 다시 지정한 구성 값을 추가합니다.

```
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: 1
          - <relabel_config> 2
```

1

원격 끝점으로 보낼 지표에 대한 쓰기 레이블 구성 목록을 추가합니다.

2

원격 쓰기 엔드포인트로 전송된 지표의 레이블 구성을 사용합니다.

다음 샘플은 기본 플랫폼 모니터링에서 클러스터 ID 레이블 cluster_id 로 메트릭을 전달하는 방법을 보여줍니다.

```
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__ 1
          targetLabel: cluster_id 2
          action: replace 3
```

1

처음에는 시스템이 __tmp_openshift_cluster_id__라는 임시 클러스터 ID 소스 레이블을 적용합니다. 이 임시 레이블은 사용자가 지정하는 클러스터 ID 레이블 이름으로 대체됩니다.

2

원격 쓰기 스토리지로 전송된 메트릭의 클러스터 ID 레이블 이름을 지정합니다. 지표에 이미 존재하는 레이블 이름을 사용하는 경우 해당 값은 이 클러스터 ID 레이블의 이름으로 덮어씁니다. 레이블 이름의 경우 __tmp_openshift_cluster_id__ 를 사용하지 마십시오. 마지막 레이블 재지정 단계는 이 이름을 사용하는 라벨을 제거합니다.

3

교체 **write** 레이블은 임시 레이블을 발신 메트릭의 **target** 레이블로 교체합니다. 이 작업은 기본값이며 작업이 지정되지 않은 경우 적용됩니다.

3. 파일을 저장하여 **ConfigMap** 오브젝트에 대한 변경 사항을 적용합니다. 업데이트된 구성의 영향을 받는 **Pod**가 자동으로 다시 시작됩니다.

주의

모니터링 **ConfigMap** 오브젝트에 대한 변경 사항을 저장하면 관련 프로젝트에서 **Pod** 및 기타 리소스를 재배포할 수 있습니다. 변경 사항을 저장하면 해당 프로젝트에서 실행 중인 모니터링 를 다시 시작할 수도 있습니다.

추가 리소스

- 쓰기 레이블 구성에 대한 자세한 내용은 원격 쓰기 스토리지 구성을 참조하십시오.

2.13. 사용자 정의 프로젝트에서 바인딩되지 않은 메트릭 속성의 영향 제어

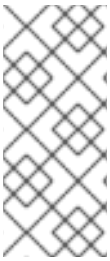
개발자는 라벨을 생성하여 키-값 쌍의 형식으로 메트릭의 속성을 정의할 수 있습니다. 잠재적인 키-값 쌍의 수는 속성에 사용 가능한 값의 수에 해당합니다. 무제한의 잠재적인 값이 있는 속성을 바인딩되지 않은 속성이라고 합니다. 예를 들어, **customer_id** 속성은 무제한 가능한 값이 있기 때문에 바인딩되지 않은 속성입니다.

할당된 모든 키-값 쌍에는 고유한 시계열이 있습니다. 레이블에서 많은 바인딩되지 않은 속성을 사용하면 **Prometheus** 성능 및 사용 가능한 디스크 공간에 영향을 줄 수 있는 기하급수적으로 시계열이 발생할 수 있습니다.

클러스터 관리자는 다음 방법을 사용하여 사용자 정의 프로젝트에서 바인딩되지 않은 메트릭 속성의 영향을 제어할 수 있습니다.

- 사용자 정의 프로젝트에서 대상 스크랩별로 허용할 수 있는 샘플 수 제한

- 스크랩된 라벨 수, 레이블 이름 길이, 레이블 값 길이를 제한합니다.
- 스크랩 샘플 임계값에 도달하거나 대상을 스크랩할 수 없는 경고 생성



참고

많은 바인딩되지 않은 속성을 추가하여 문제를 방지하려면 메트릭에 정의된 스크랩 샘플, 라벨 이름 및 바인딩되지 않은 속성 수를 제한합니다. 또한 제한된 가능한 값 집합에 바인딩된 특성을 사용하여 잠재적인 키-값 쌍 조합의 수를 줄입니다.

2.13.1. 사용자 정의 프로젝트에 대한 스크랩 샘플 및 라벨 제한 설정

사용자 정의 프로젝트에서 대상 스크랩별로 허용할 수 있는 샘플 수를 제한할 수 있습니다. 스크랩된 라벨 수, 레이블 이름 길이, 레이블 값 길이를 제한할 수도 있습니다.

주의

샘플 또는 라벨 제한을 설정하면 제한에 도달한 후 해당 대상 스크랩에 대한 추가 샘플 데이터가 수집되지 않습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

절차

1. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config**

ConfigMap 오브젝트를 편집합니다.

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

2. 사용자 정의 프로젝트에서 대상 스크랩별로 허용되는 샘플 수를 제한하려면 enforcedSampleLimit 구성을 data/config.yaml에 추가합니다.

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

1 이 매개변수가 지정된 경우 값이 필요합니다. 이 enforcedSampleLimit 예제에서는 사용자 정의 프로젝트의 대상 스크랩별로 허용할 수 있는 샘플 수를 50,000개로 제한합니다.

3. enforcedLabelLimit,enforcedLabelNameLengthLimit, enforcedLabelNameLengthLimit 구성을 data/config.yaml 에 추가하여 스크랩된 라벨 수, 라벨 이름 길이, 사용자 정의 프로젝트의 라벨 값 길이를 제한합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      enforcedLabelLimit: 500 1
      enforcedLabelNameLengthLimit: 50 2
      enforcedLabelValueLengthLimit: 600 3
```

1 스크랩당 최대 라벨 수를 지정합니다. 기본값은 0 이며 제한이 없음을 지정합니다.

2 레이블 이름의 최대 길이를 지정합니다. 기본값은 0 이며 제한이 없음을 지정합니다.

3

레이블 값의 최대 길이를 지정합니다. 기본값은 **0** 이며 제한이 없음을 지정합니다.

4. 파일을 저장하여 변경 사항을 적용합니다. 제한이 자동으로 적용됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

user-workload-monitoring-config ConfigMap 오브젝트에 변경 사항이 저장되면 **openshift-user-workload-monitoring** 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

2.13.2. 스크랩 샘플 경고 생성

다음의 경우를 알리는 경고를 생성할 수 있습니다.

- 대상을 스크랩할 수 없거나 지정된 기간 동안 사용할 수 없습니다.
- 스크랩 샘플 임계값에 도달하거나 기간 동안 지정된 항목에 대해 초과하였습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- **enforcedSampleLimit**을 사용하여 사용자 정의 프로젝트에서 대상 스크랩별로 허용할 수 있는 샘플 수를 제한했습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. 대상이 중단되는 경우 및 시행된 샘플 제한에 도달하는 경우를 알려주는 경고와 함께 **YAML** 파일을 생성합니다. 이 예제의 파일은 **monitoring-stack-alerts.yaml**이라고 합니다.

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

1

경고 규칙의 이름을 정의합니다.

2

경고 규칙이 배포될 사용자 정의 프로젝트를 지정합니다.

3

대상을 스크랩할 수 없거나 기간 동안 사용할 수 없는 경우 **TargetDown** 경고가 실행됩니다.

4

TargetDown 경고가 실행되는 경우 출력될 메시지입니다.

5

경고가 실행되기 전에 **TargetDown** 경고의 조건이 이 기간에 **true**여야 합니다.

6

TargetDown 경고의 심각도를 정의합니다.

7

정의된 스크랩 샘플 임계값에 도달하거나 기간 동안 지정된 항목에 대해 초과하면 **ApproachingEnforcedSamplesLimit** 경고가 실행됩니다.

8

ApproachingEnforcedSamplesLimit 경고가 실행될 때 출력되는 메시지입니다.

9

ApproachingEnforcedSamplesLimit 경고의 임계값입니다. 이 예에서 대상 스크랩당 샘플 수가 **50000** 개 중 **80%**를 초과하면 경고가 발생합니다. 또한 경고가 실행되기 전에 기간을 통과해야 합니다. 표현식 **scrape_samples_scraped/<number> > <threshold>**에 있는 **<number>**는 **user-workload-monitoring-config ConfigMap** 오브젝트에 정의된 **enforcedSampleLimit** 값과 일치해야 합니다.

10

경고가 실행되기 전에 **ApproachingEnforcedSamplesLimit** 경고의 조건이 이 기간에 **true**여야 합니다.

11 **ApproachingEnforcedSamplesLimit 경고의 심각도를 정의합니다.**

2. 사용자 정의 프로젝트에 구성을 적용합니다.

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

추가 리소스

- 사용자 정의 워크로드 모니터링 구성 맵 생성
- 사용자 정의 프로젝트 모니터링 활성화
- 스크랩 샘플이 가장 많은 메트릭을 쿼리하는 단계는 Prometheus가 많은 디스크 공간을 소비하는 이유 확인을 참조하십시오.

3장. 외부 ALERTMANAGER 인스턴스 구성

OpenShift Container Platform 모니터링 스택에는 Prometheus의 경고를 라우팅하는 로컬 Alertmanager 인스턴스가 포함되어 있습니다. openshift-monitoring 프로젝트 또는 user-workload-monitoring-config 프로젝트에서 cluster-monitoring-config 구성 맵을 구성하여 외부 Alertmanager 인스턴스를 추가할 수 있습니다.

여러 클러스터에 대해 동일한 외부 Alertmanager 구성을 추가하고 각 클러스터에 대해 로컬 인스턴스를 비활성화하면 단일 외부 Alertmanager 인스턴스를 사용하여 여러 클러스터에 대한 경고 라우팅을 관리할 수 있습니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.
- openshift-monitoring 프로젝트에서 핵심 OpenShift Container Platform 모니터링 구성 요소를 구성하는 경우:
  - cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - cluster-monitoring-config 구성 맵을 생성했습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - cluster-admin 클러스터 역할의 사용자로 또는 openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - user-workload-monitoring-config 구성 맵을 생성했습니다.

절차

1. ConfigMap 오브젝트를 편집합니다.
   - 핵심 OpenShift Container Platform 프로젝트의 라우팅 경고에 대한 추가

Alertmanager를 구성하려면 다음을 수행합니다.

a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config** 구성 맵을 편집합니다.

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

b. **data/config.yaml/prometheusK8s** 아래에 **additionalAlertmanagerConfigs:** 섹션을 추가합니다.

c. 이 섹션에서 추가 **Alertmanagers**의 설정 세부 정보를 추가합니다.

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

<alertmanager_specification>의 경우 추가 **Alertmanager** 인스턴스의 인증 및 기타 구성 세부 정보를 대체합니다. 현재 지원되는 인증 방법은 베어러 토큰 **(bearerToken)** 및 클라이언트 **TLS (tlsConfig)**입니다. 다음 샘플 구성 맵은 클라이언트 **TLS** 인증이 있는 베어러 토큰을 사용하여 추가 **Alertmanager**를 구성합니다.

```
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
```

- 사용자 정의 프로젝트에서 라우팅 경고에 대한 추가 **Alertmanager** 인스턴스를 구성하려면 다음을 수행합니다.

a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config** 구성 맵을 편집합니다.

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

b. **data/config.yaml/**에 **<component>/additionalAlertmanagerConfigs:** 섹션을 추가합니다.

c. 이 섹션에서 추가 **Alertmanagers**의 설정 세부 정보를 추가합니다.

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

<component>의 경우 지원되는 외부 **Alertmanager** 구성 요소인 **prometheus** 또는 **thanosRuler** 중 하나를 바꿉니다.

<alertmanager_specification>의 경우 추가 **Alertmanager** 인스턴스의 인증 및 기타 구성 세부 정보를 대체합니다. 현재 지원되는 인증 방법은 베어러 토큰 **(bearerToken)** 및 클라이언트 **TLS (tlsConfig)**입니다. 다음 샘플 구성 맵은 베어러 토큰 및 클라이언트 **TLS** 인증으로 **Thanos Ruler**를 사용하여 추가 **Alertmanager**를 구성합니다.

```
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
```

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

2. 파일을 저장하여 **ConfigMap** 오브젝트에 대한 변경 사항을 적용합니다. 새로운 구성 요소 배치 구성이 자동으로 적용됩니다.

3.1. 시계열 및 경고에 추가 라벨 연결

Prometheus의 외부 라벨 기능을 사용하여 사용자 정의 라벨을 모든 시계열 및 경고에 연결할 수 있습니다.

사전 요구 사항

- 핵심 **OpenShift Container Platform** 모니터링 구성 요소인 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. **ConfigMap** 오브젝트를 편집합니다.
   - 주요 **OpenShift Container Platform** 프로젝트를 모니터링하는 **Prometheus** 인스턴스를 남아 있는 모든 시계열 및 경고에 사용자 정의 라벨을 연결하려면 다음을 수행합니다.
     a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

     b. **data/config.yaml** 아래의 모든 메트릭에 추가할 라벨 맵을 정의합니다.

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

```
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        <key>: <value> 1
```

1

<key>: <value>를 <key>가 새 라벨에 고유한 이름이며 <value>가 값인 키-값 쌍의 맵으로 바꿉니다.

주의

prometheus 또는 prometheus_replica를 키 이름으로 사용하지 마십시오. 예약되어 있으며 덮어쓸 예정이기 때문입니다.

예를 들어, 모든 시계열 및 경고에 리전 및 환경에 대한 메타데이터를 추가하려면 다음을 사용합니다.

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

- 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스를 모든 시계열 및 경고에 연결하려면 다음을 수행합니다.

  a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b.

data/config.yaml 아래의 모든 메트릭에 추가할 라벨 맵을 정의합니다.

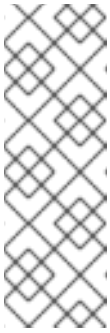
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      externalLabels:
        <key>: <value> 1
```

1

<key>: <value>를 <key>가 새 라벨에 고유한 이름이며 <value>가 값인 키-값 쌍의 맵으로 바꿉니다.

주의

prometheus 또는 prometheus_replica를 키 이름으로 사용하지 마십시오. 예약되어 있으며 덮어쓸 예정이기 때문입니다.



참고

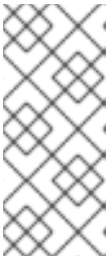
openshift-user-workload-monitoring 프로젝트에서 Prometheus는 메트릭을 처리하고 Thanos Ruler는 경고 및 레코딩 규칙을 처리합니다. user-workload-monitoring-config ConfigMap 오브젝트에서 prometheus에 대한 externalLabels를 설정하면 규칙에는 적용되지 않고 메트릭에 대한 외부 라벨만 구성됩니다.

예를 들어, 사용자 정의 프로젝트와 관련된 모든 시계열 및 경고에 리전 및 환경에 대한 메타데이터를 추가하려면 다음을 사용합니다.

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

```
config.yaml: |
  prometheus:
    externalLabels:
      region: eu
      environment: prod
```

2. 파일을 저장하여 변경 사항을 적용합니다. 새 구성이 자동으로 적용됩니다.



참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 **user-workload-monitoring-config ConfigMap** 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화

3.2. 모니터링 구성 요소에 대한 로그 수준 설정

Alertmanager, Prometheus Operator, Prometheus, Thanos Querier, Thanos Ruler의 로그 수준을 구성할 수 있습니다.

다음 로그 수준은 **cluster-monitoring-config** 및 **user-workload-monitoring-config ConfigMap** 오브젝트의 관련 구성 요소에 적용할 수 있습니다.

- **debug.** 디버그, 정보, 경고 및 오류 메시지를 기록합니다.
- **info.** 정보, 경고 및 오류 메시지를 기록합니다.
- **warn.** 경고 및 오류 메시지만 기록합니다.
- **error.** 오류 메시지만 기록합니다.

기본값 로그 수준은 **info**입니다.

사전 요구 사항

- **openshift-monitoring** 프로젝트에서 **Alertmanager, Prometheus Operator, Prometheus** 또는 **Thanos Querier**의 로그 수준을 설정하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **openshift-user-workload-monitoring** 프로젝트에서 **Prometheus Operator, Prometheus** 또는 **Thanos Ruler**의 로그 수준을 설정하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. ConfigMap 오브젝트를 편집합니다.

- openshift-monitoring 프로젝트에서 구성 요소의 로그 수준을 설정하려면 다음을 수행합니다.

  a. openshift-monitoring 프로젝트에서 cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

  b. data/config.yaml 아래의 구성 요소에 logLevel: <log_level>을 추가합니다.

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

  1 로그 수준을 설정하는 모니터링 스택 구성 요소입니다. 기본 플랫폼 모니터링의 경우 사용 가능한 구성 요소 값은 prometheusK8s,alertmanagerMain,prometheusOperator, thanosQuerier 입니다.

  2 구성 요소에 설정할 로그 수준입니다. 사용 가능한 값은 오류,경고,info, debug 입니다. 기본값은 info 입니다.

- openshift-user-workload-monitoring 프로젝트에서 구성 요소의 로그 수준을 설정하려면 다음을 수행합니다.

  a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b. data/config.yaml 아래의 구성 요소에 logLevel: <log_level>을 추가합니다.

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

1 로그 수준을 설정하는 모니터링 스택 구성 요소입니다. 사용자 워크로드 모니터링의 경우 사용 가능한 구성 요소 값은 prometheus,prometheusOperator, thanosRuler 입니다.

2 구성 요소에 설정할 로그 수준입니다. 사용 가능한 값은 오류,경고,info, debug 입니다. 기본값은 info 입니다.

2. 파일을 저장하여 변경 사항을 적용합니다. 로그 수준 변경을 적용하면 구성 요소의 Pod가 자동으로 다시 시작됩니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 user-workload-monitoring-config ConfigMap 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

3. 관련 프로젝트의 배포 또는 **Pod** 구성을 검토하여 로그 수준이 적용되었는지 확인합니다. 다음 예제는 **openshift-user-workload-monitoring** 프로젝트의 **prometheus-operator** 배포에서 로그 수준을 확인합니다.

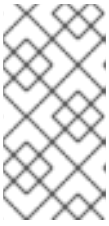
```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

출력 예

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

4. 구성 요소의 **Pod**가 실행 중인지 확인합니다. 다음 예제는 **openshift-user-workload-monitoring** 프로젝트의 **Pod** 상태를 나열합니다.

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

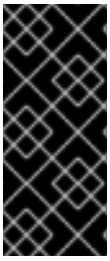


참고

인식할 수 없는 **Prometheus Operator loglevel** 값이 **ConfigMap**에 포함된 경우 **Pod** 구성 요소가 성공적으로 다시 시작되지 않을 수 있습니다.

3.3. PROMETHEUS의 쿼리 로그 파일 활성화

엔진에서 실행한 모든 쿼리를 로그 파일에 작성하도록 **Prometheus**를 구성할 수 있습니다. 기본 플랫폼 모니터링 및 사용자 정의 워크로드 모니터링에 대해 이를 수행할 수 있습니다.



중요

로그 순환은 지원되지 않으므로 문제를 해결해야 하는 경우에만 이 기능을 일시적으로 활성화합니다. 문제 해결을 마친 후 **ConfigMap** 오브젝트의 변경 사항을 되돌려서 쿼리 로깅을 비활성화하여 기능을 활성화합니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **openshift-monitoring** 프로젝트에서 **Prometheus**의 쿼리 로그 파일 기능을 활성화하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **openshift-user-workload-monitoring** 프로젝트에서 **Prometheus**의 쿼리 로그 파일 기능을 활성화하는 경우:
  - **cluster-admin** 클러스터 역할의 사용자로 또는 **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
  - **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.

절차

- **openshift-monitoring** 프로젝트에서 **Prometheus**의 쿼리 로그 파일을 설정하려면 다음을 수행합니다.
  1. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

2. data/config.yaml 아래에 prometheusK8s 의 queryLogFile: <path >를 추가합니다.

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

1 쿼리가 기록될 파일의 전체 경로입니다.

3. 파일을 저장하여 변경 사항을 적용합니다.

> 주의
>
> 모니터링 구성 맵에 대한 변경 사항을 저장하면 관련 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

4. 구성 요소의 포드가 실행 중인지 확인합니다. 다음 샘플 명령은 openshift-monitoring 프로젝트의 Pod 상태를 나열합니다.

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

5. 쿼리 로그를 읽습니다.Read the query log:

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

중요

기록된 쿼리 정보를 검사한 후 구성 맵에서 설정을 되돌립니다.

- openshift-user-workload-monitoring 프로젝트에서 Prometheus의 쿼리 로그 파일을 설정하려면 다음을 수행합니다.

1. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. data/config.yaml 아래의 prometheus 에 queryLogFile: <path >를 추가합니다.

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

1 쿼리가 기록될 파일의 전체 경로입니다.

3. 파일을 저장하여 변경 사항을 적용합니다.

참고

클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화하지 않는 한 user-workload-monitoring-config ConfigMap 오브젝트에 적용되는 구성이 활성화되어 있지 않습니다.

주의

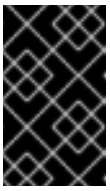
모니터링 구성 맵에 대한 변경 사항을 저장하면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

4. 구성 요소의 포드가 실행 중인지 확인합니다. 다음 예제 명령은 **openshift-user-workload-monitoring** 프로젝트의 **Pod** 상태를 나열합니다.

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

5. 쿼리 로그를 읽습니다.**Read the query log:**

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



중요

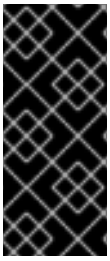
기록된 쿼리 정보를 검사한 후 구성 맵에서 설정을 되돌립니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 모니터링을 활성화하는 단계는 사용자 정의 프로젝트에 대한 모니터링 활성화를 참조하십시오.

3.4. THANOS QUERIER에 대한 쿼리 로깅 활성화

openshift-monitoring 프로젝트에서 기본 플랫폼 모니터링의 경우 **Cluster Monitoring Operator**가 **Thanos Querier**에서 실행하는 모든 쿼리를 로깅할 수 있습니다.



중요

로그 순환은 지원되지 않으므로 문제를 해결해야 하는 경우에만 이 기능을 일시적으로 활성화합니다. 문제 해결을 마친 후 **ConfigMap** 오브젝트의 변경 사항을 되돌려서 쿼리 로깅을 비활성화하여 기능을 활성화합니다.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.

절차

openshift-monitoring 프로젝트에서 **Thanos Querier**에 대한 쿼리 로깅을 활성화할 수 있습니다.

1. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

2. **data/config.yaml** 아래에 **thanosQuerier** 섹션을 추가하고 다음 예와 같이 값을 추가합니다.

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

1. 로깅을 활성화하려면 값을 **true** 로 설정하고 로깅을 활성화하려면 **false** 를 설정합니다. 기본값은 **false**입니다.

2

값을 **debug,info,warn, error** 로 설정합니다. **logLevel** 에 값이 없는 경우 로그 수준은 기본적으로 오류 입니다.

3. 파일을 저장하여 변경 사항을 적용합니다.

주의

모니터링 구성 맵에 대한 변경 사항을 저장하면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

검증

1. **Thanos Querier Pod**가 실행 중인지 확인합니다. 다음 샘플 명령은 **openshift-monitoring** 프로젝트의 **Pod** 상태를 나열합니다.

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

2. 다음 샘플 명령을 모델로 사용하여 테스트 쿼리를 실행합니다.

```
$ 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. 다음 명령을 실행하여 쿼리 로그를 읽습니다.

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

참고

thanos-querier Pod는 고가용성(**HA**) **Pod**이므로 하나의 **Pod**에서만 로그를 볼 수 있습니다.

4. 기록된 쿼리 정보를 검사한 후 구성 맵에서 **enableRequestLogging** 값을 **false** 로 변경하여 쿼리 로깅을 비활성화합니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.

4장. PROMETHEUS ADAPTER에 대한 감사 로그 수준 설정

기본 플랫폼 모니터링에서는 Prometheus 어댑터의 감사 로그 수준을 구성할 수 있습니다.

사전 요구 사항

- OpenShift CLI(oc)가 설치되어 있습니다.
- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.

절차

기본 openshift-monitoring 프로젝트에서 Prometheus 어댑터의 감사 로그 수준을 설정할 수 있습니다.

1. openshift-monitoring 프로젝트에서 cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. data/config.yaml 의 k8sPrometheusAdapter/audit 섹션에 profile: 을 추가합니다.

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

1. Prometheus 어댑터에 적용할 감사 로그 수준입니다.

3. **profile:** 매개변수에 다음 값 중 하나를 사용하여 감사 로그 수준을 설정합니다.

- **None:** 이벤트를 기록하지 마십시오.
- **metadata:** 사용자, 타임스탬프 등과 같은 요청에 대한 메타데이터만 기록합니다. 요청 텍스트와 응답 텍스트를 기록하지 마십시오. 메타데이터 는 기본 감사 로그 수준입니다.
- **request :** 메타데이터와 요청 텍스트만 기록하지만 응답 텍스트는 기록하지 않습니다. 이 옵션은 리소스가 아닌 요청에는 적용되지 않습니다.
- **RequestResponse:** 로그 이벤트 메타데이터, 텍스트 요청 및 응답 텍스트입니다. 이 옵션은 리소스가 아닌 요청에는 적용되지 않습니다.

4. 파일을 저장하여 변경 사항을 적용합니다. 변경 사항을 적용할 때 **Prometheus Adapter**의 **Pod**가 자동으로 다시 시작됩니다.

주의

모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

검증

1. 구성 맵의 **k8sPrometheusAdapter/audit/profile** 에서 로그 수준을 **Request** 로 설정하고 파일을 저장합니다.

2. **Prometheus Adapter**의 **Pod**가 실행 중인지 확인합니다. 다음 예제에서는 **openshift-monitoring** 프로젝트의 **Pod** 상태를 나열합니다.

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

3. 감사 로그 수준 및 감사 로그 파일 경로가 올바르게 구성되었는지 확인합니다.

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

출력 예

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

4. openshift-monitoring 프로젝트의 prometheus-adapter 배포에 올바른 로그 수준이 적용되었는지 확인합니다.

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

출력 예

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

참고

ConfigMap 오브젝트에서 Prometheus Adapter에 대해 인식할 수 없는 프로필 값을 입력하면 Prometheus 어댑터에 대한 변경 사항이 없으며 Cluster Monitoring Operator에서 오류가 기록됩니다.

5. Prometheus Adapter의 감사 로그를 확인합니다.

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

추가 리소스

- 모니터링 구성 맵을 생성하는 단계는 모니터링 스택 구성 준비를 참조하십시오.

4.1. 로컬 ALERTMANAGER 비활성화

OpenShift Container Platform 모니터링 스택의 openshift-monitoring 프로젝트에서 Prometheus 인스턴스에서 경고를 라우팅하는 로컬 Alertmanager가 기본적으로 활성화되어 있습니다.

로컬 Alertmanager가 필요하지 않은 경우 openshift-monitoring 프로젝트에서 cluster-monitoring-config 구성 맵을 구성하여 비활성화할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- cluster-monitoring-config 구성 맵을 생성했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

절차

1. openshift-monitoring 프로젝트에서 cluster-monitoring-config 구성 맵을 편집합니다.

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

2. data/config.yaml 아래의 alertmanagerMain 구성 요소에 대해 enabled: false를 추가합니다.

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

```
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      enabled: false
```

3. 파일을 저장하여 변경 사항을 적용합니다. 변경 사항을 적용하면 **Alertmanager** 인스턴스가 자동으로 비활성화됩니다.

추가 리소스

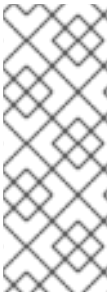
- Prometheus Alertmanager 문서
- 경고 관리

4.2. 다음 단계

- 사용자 정의 프로젝트 모니터링 활성화
- 원격 상태 보고에 대해 알아보고 필요한 경우 옵트아웃합니다.

5장. 사용자 정의 프로젝트 모니터링 활성화

OpenShift Container Platform 4.11에서는 기본 플랫폼 모니터링 외에도 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다. 추가 모니터링 솔루션 없이도 OpenShift Container Platform에서 자체 프로젝트를 모니터링할 수 있습니다. 이 기능을 사용하면 핵심 플랫폼 구성 요소 및 사용자 정의 프로젝트에 대한 모니터링을 중앙 집중화합니다.



참고

OLM(Operator Lifecycle Manager)을 사용하여 설치한 Prometheus Operator 버전은 사용자 정의 모니터링과 호환되지 않습니다. 따라서 OLM Prometheus Operator에서 관리하는 Prometheus 사용자 정의 리소스(CR)로 설치된 사용자 정의 Prometheus 인스턴스는 OpenShift Container Platform에서 지원되지 않습니다.

5.1. 사용자 정의 프로젝트 모니터링 활성화

클러스터 관리자는 클러스터 모니터링 ConfigMap 오브젝트에서 enableUserWorkload: true 필드를 설정하여 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다.

중요

OpenShift Container Platform 4.11에서는 사용자 정의 프로젝트에 대한 모니터링을 활성화하기 전에 모든 사용자 정의 Prometheus 인스턴스를 제거해야 합니다.

참고

OpenShift Container Platform에서 사용자 정의 프로젝트에 대한 모니터링을 활성화하려면 cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다. 그러면 클러스터 관리자가 선택적으로 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하기 위해 사용자에게 권한을 부여할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

- cluster-monitoring-config ConfigMap 오브젝트를 생성하셨습니다.

- openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 선택적으로 생성 및 구성했습니다. 사용자 정의 프로젝트를 모니터링하는 구성 요소에 대한 구성 옵션을 이 ConfigMap 오브젝트에 추가할 수 있습니다.

참고

user-workload-monitoring-config ConfigMap 오브젝트에 대한 구성 변경을 저장할 때마다 openshift-user-workload-monitoring 프로젝트의 Pod가 재배포됩니다. 이러한 구성 요소가 재배포되는 데 시간이 다소 걸릴 수 있습니다. 먼저 사용자 정의 프로젝트에 대한 모니터링을 활성화하기 전에 ConfigMap 오브젝트를 생성하고 구성하여 Pod를 자주 재배포하지 않도록 할 수 있습니다.

프로세스

1. cluster-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. data/config.yaml에 enableUserWorkload: true를 추가합니다.

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

1 true로 설정하는 경우 enableUserWorkload 매개변수를 사용하면 클러스터에서 사용자 정의 프로젝트를 모니터링할 수 있습니다.

3. 파일을 저장하여 변경 사항을 적용합니다. 그런 다음 사용자 정의 프로젝트에 대한 모니터링이 자동으로 활성화됩니다.

주의

cluster-monitoring-config ConfigMap 오브젝트에 변경 사항이 저장되면 **openshift-monitoring** 프로젝트의 **Pod** 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

4. **prometheus-operator, prometheus-user-workload** 및 **thanos-ruler-user-workload Pod**가 **openshift-user-workload-monitoring** 프로젝트에서 실행 중인지 확인합니다. **Pod**를 시작하는 데 시간이 걸릴 수 있습니다.

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

출력 예

```
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
```

추가 리소스

- 클러스터 모니터링 구성 맵 생성
- 모니터링 스택 구성
- 사용자 정의 프로젝트 모니터링을 구성할 수 있는 사용자 권한 부여

5.2. 사용자 정의 프로젝트를 모니터링할 수 있는 사용자 권한 부여

클러스터 관리자는 모든 핵심 OpenShift Container Platform 및 사용자 정의 프로젝트를 모니터링할 수 있습니다.

클러스터 관리자는 개발자 및 다른 사용자에게 자신의 프로젝트를 모니터링할 수 있는 권한을 부여할 수 있습니다. 권한은 다음 모니터링 역할 중 하나를 할당하는 방식으로 부여합니다.

- monitoring-rules-view 클러스터 역할은 프로젝트의 PrometheusRule 사용자 정의 리소스에 대한 읽기 액세스 권한을 제공합니다.
- monitoring-rules-edit 클러스터 역할은 사용자에게 프로젝트의 PrometheusRule 사용자 정의 리소스를 생성, 수정, 삭제할 수 있는 권한을 부여합니다.
- monitoring-edit 클러스터 역할은 monitoring-rules-edit 클러스터 역할과 동일한 권한을 부여합니다. 또한 사용자는 서비스 또는 Pod에 대한 새로운 스크랩 대상을 생성할 수 있습니다. 이 역할을 사용하면 ServiceMonitor 및 PodMonitor 리소스를 생성, 수정, 삭제할 수도 있습니다.

또한 사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성할 수 있는 권한을 사용자에게 부여할 수도 있습니다.

- openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config-edit 역할을 통해 user-workload-monitoring-config ConfigMap 오브젝트를 편집할 수 있습니다. 이 역할을 사용하면 ConfigMap 오브젝트를 편집하여 사용자 정의 워크로드 모니터링에 대해 Prometheus, Prometheus Operator 및 Thanos Ruler를 구성할 수 있습니다.

사용자 정의 프로젝트에 대한 경고 라우팅을 구성할 수 있는 권한을 사용자에게 부여할 수도 있습니다.

- alert-routing-edit 클러스터 역할은 사용자에게 프로젝트의 AlertmanagerConfig 사용자 정의 리소스를 생성, 업데이트 및 삭제할 수 있는 권한을 부여합니다.

이 섹션에서는 OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 이러한 역할을 할당하는 방법에 대해 자세히 설명합니다.

5.2.1. 웹 콘솔을 사용하여 사용자에게 권한 부여

OpenShift Container Platform 웹 콘솔을 사용하여 사용자의 프로젝트를 모니터링할 수 있는 권한을 사용자에게 부여할 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 이미 존재하는 역할에 할당 중인 사용자 계정입니다.

프로세스

1. **OpenShift Container Platform** 웹 콘솔의 관리자 관점에서 사용자 관리 → 역할 바인딩 → 바인딩 생성으로 이동합니다.
2. 바인딩 유형 섹션에서 "네임스페이스 역할 바인딩" 유형을 선택합니다.
3. 이름 필드에 역할 바인딩의 이름을 입력합니다.
4. 네임스페이스 필드에서 액세스 권한을 부여하려는 사용자 정의 프로젝트를 선택합니다.

중요

모니터링 역할은 네임스페이스 필드에 적용하는 프로젝트에 바인딩됩니다. 이 프로세스를 사용하여 사용자에게 부여한 권한은 선택한 프로젝트에만 적용됩니다.

5. 역할 이름 목록에서 **monitoring-rules-view, monitoring-rules-edit** 또는 **monitoring-edit**를 선택합니다.
6. 주체 섹션에서 사용자를 선택합니다.
7. 주체 이름 필드에 사용자 이름을 입력합니다.
8. 역할 바인딩을 적용하려면 만들기를 선택합니다.

5.2.2. CLI를 사용하여 사용자에게 권한 부여

OpenShift CLI(oc)를 사용하여 자체 프로젝트를 모니터링할 수 있는 권한을 사용자에게 부여할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 이미 존재하는 역할에 할당 중인 사용자 계정입니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

프로세스

- 프로젝트의 사용자에게 모니터링 역할을 할당합니다.

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

1 <role>을 monitoring-rules-view, monitoring-rules-edit 또는 monitoring-edit로 바꿉니다.

중요

선택한 역할이 무엇이든 클러스터 관리자로 특정 프로젝트에 대해 바인딩해야 합니다.

예를 들어 <role>을 monitoring-edit로 바꾸고, <user>를 johnsmith로 바꾸고, <namespace>를 ns1으로 바꿉니다. 이를 통해 메트릭 컬렉션을 설정하고 ns1 네임스페이스에서 경고 규칙을 생성할 수 있는 사용자 johnsmith 권한이 할당됩니다.

5.3. 사용자 정의 프로젝트 모니터링을 구성할 수 있는 사용자 권한 부여

사용자 정의 프로젝트에 대한 모니터링을 구성할 수 있는 권한을 사용자에게 부여할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 이미 존재하는 역할에 할당 중인 사용자 계정입니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

프로세스

- user-workload-monitoring-config-edit 역할을 openshift-user-workload-monitoring 프로젝트에 있는 사용자에게 할당합니다.

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

5.4. 사용자 지정 애플리케이션을 위해 클러스터 외부에서 메트릭에 액세스

자체 서비스를 모니터링할 때 명령줄에서 Prometheus 통계를 쿼리하는 방법을 알아봅니다. thanos-querier 경로를 사용하여 클러스터 외부의 모니터링 데이터에 액세스할 수 있습니다.

사전 요구 사항

- *사용자 정의 프로젝트 모니터링 활성화*에 따라 자체 서비스를 배포했습니다.

절차

1. Prometheus에 연결할 토큰을 추출합니다.

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

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

2. 경로 호스트를 추출합니다.

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

3. 명령줄에서 자체 서비스의 메트릭을 쿼리합니다. 예를 들어 다음과 같습니다.

```
$ NAMESPACE=ns1
```

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

출력에 애플리케이션 pod가 가동된 기간이 표시됩니다.

출력 예

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

5.5. 모니터링에서 사용자 정의 프로젝트 제외

개별 사용자 정의 프로젝트는 사용자 워크로드 모니터링에서 제외될 수 있습니다. 이렇게 하려면 값이 false인 프로젝트 네임스페이스에 openshift.io/user-monitoring 레이블을 추가하면 됩니다.

절차

1. 프로젝트 네임스페이스에 라벨을 추가합니다.

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

2. 모니터링을 다시 활성화하려면 네임스페이스에서 라벨을 제거합니다.

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

참고

프로젝트에 대한 활성 모니터링 대상이 있는 경우 레이블을 추가한 후 **Prometheus**가 스크랩을 중지하는 데 몇 분이 걸릴 수 있습니다.

5.6. 사용자 정의 프로젝트 모니터링 비활성화

사용자 정의 프로젝트에 대한 모니터링을 활성화한 후 클러스터 모니터링 **ConfigMap** 오브젝트에서 **enableUserWorkload: false**를 설정하여 다시 비활성화할 수 있습니다.

참고

또는 사용자 정의 프로젝트에 대한 모니터링을 비활성화하려면 **enableUserWorkload: true**를 제거할 수 있습니다.

프로세스

1. **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

   a. **data/config.yaml**에서 **enableUserWorkload:**를 **false**로 설정합니다.

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

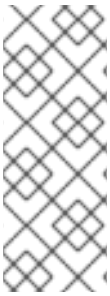
2. 파일을 저장하여 변경 사항을 적용합니다. 그런 다음 사용자 정의 프로젝트에 대한 모니터링이 자동으로 비활성화됩니다.

3. **prometheus-operator, prometheus-user-workload** 및 **thanos-ruler-user-workload Pod**가 **openshift-user-workload-monitoring** 프로젝트에서 제거되었는지 확인합니다. 이 작업을 수행하는 데 다소의 시간이 걸릴 수 있습니다.

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

출력 예

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



참고

사용자 정의 프로젝트의 모니터링이 비활성화되면 **openshift-user-workload-monitoring** 프로젝트의 **user-workload-monitoring-config ConfigMap** 오브젝트는 자동으로 삭제되지 않습니다. 이는 **ConfigMap** 오브젝트에서 생성한 모든 사용자 지정 구성을 유지하기 위한 것입니다.

5.7. 다음 단계

- 메트릭 관리

6장. 사용자 정의 프로젝트에 대한 경고 라우팅 활성화

OpenShift Container Platform 4.11에서 클러스터 관리자는 사용자 정의 프로젝트에 대한 경고 라우팅을 활성화할 수 있습니다. 이 프로세스는 두 가지 일반적인 단계로 구성됩니다.

- 사용자 정의 프로젝트에 대한 경고 라우팅을 활성화하여 기본 플랫폼 **Alertmanager** 인스턴스를 사용하거나 선택 옵션으로 사용자 정의 프로젝트에 대해서만 별도의 **Alertmanager** 인스턴스를 사용합니다.
- 사용자 정의 프로젝트에 대한 경고 라우팅을 구성할 수 있는 권한을 사용자에게 부여합니다.

이러한 단계를 완료한 후 개발자와 기타 사용자는 사용자 정의 프로젝트에 대한 사용자 정의 경고 및 경고 라우팅을 구성할 수 있습니다.

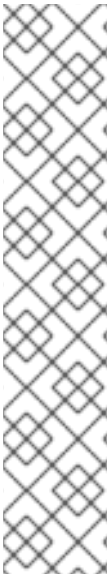
6.1. 사용자 정의 프로젝트의 경고 라우팅 이해

클러스터 관리자는 사용자 정의 프로젝트에 대한 경고 라우팅을 활성화할 수 있습니다. 이 기능을 사용하면 **alert-routing-edit** 역할을 가진 사용자가 사용자 정의 프로젝트에 대한 경고 알림 라우팅 및 수신자를 구성할 수 있습니다. 이러한 알림은 기본 **Alertmanager** 인스턴스에서 라우팅되거나 활성화된 경우 사용자 정의 모니터링을 위한 선택적 **Alertmanager** 인스턴스에서 라우팅됩니다.

그러면 관리자의 도움말 없이 사용자 정의 프로젝트에 대한 **AlertmanagerConfig** 오브젝트를 생성하거나 편집하여 사용자 정의 경고 라우팅을 만들고 구성할 수 있습니다.

사용자 정의 프로젝트에 대한 경고 라우팅을 정의한 후 사용자 정의 경고 알림이 다음과 같이 라우팅됩니다.

- 기본 플랫폼 **Alertmanager** 인스턴스를 사용하는 경우 **openshift-monitoring** 네임스페이스의 **alertmanager-main Pod**로 이동합니다.
- 사용자 정의 프로젝트에 대해 **Alertmanager**의 별도의 인스턴스를 활성화한 경우 **openshift-user-workload-monitoring** 네임스페이스의 **alertmanager-user-workload Pod**로 이동합니다.



참고

다음은 사용자 정의 프로젝트에 대한 경고 라우팅의 제한 사항입니다.

- 사용자 정의 경고 규칙의 경우 사용자 정의 라우팅은 리소스가 정의된 네임스페이스로 범위가 지정됩니다. 예를 들어 네임스페이스 **ns1** 의 라우팅 구성은 동일한 네임스페이스의 **PrometheusRules** 리소스에만 적용됩니다.

- 네임스페이스가 사용자 정의 모니터링에서 제외되면 네임스페이스의 **AlertmanagerConfig** 리소스가 **Alertmanager** 구성의 일부가 되지 않습니다.

6.2. 사용자 정의 경고 라우팅에 대한 플랫폼 ALERTMANAGER 인스턴스 활성화

사용자가 **Alertmanager**의 기본 플랫폼 인스턴스를 사용하는 사용자 정의 경고 라우팅 구성을 만들 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

2. **data/config.yaml** 아래의 **alertmanagerMain** 섹션에 **enableUserAlertmanagerConfig: true** 를 추가합니다.

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

```
config.yaml: |
  alertmanagerMain:
    enableUserAlertmanagerConfig: true (1)
```

(1)

사용자가 Alertmanager의 기본 플랫폼 인스턴스를 사용하는 사용자 정의 경고 라우팅 구성을 생성할 수 있도록 enableUserAlertmanagerConfig 값을 true 로 설정합니다.

3. 파일을 저장하여 변경 사항을 적용합니다.

6.3. 사용자 정의 경고 라우팅에 대해 별도의 ALERTMANAGER 인스턴스 활성화

일부 클러스터에서는 기본 플랫폼 Alertmanager 인스턴스의 부하를 줄이고 기본 플랫폼 경고와 사용자 정의 경고를 더 잘 분리할 수 있는 사용자 정의 프로젝트에 전용 Alertmanager 인스턴스를 배포할 수 있습니다. 이 경우 필요에 따라 Alertmanager의 별도의 인스턴스를 활성화하여 사용자 정의 프로젝트에 대한 경고만 보낼 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- openshift-monitoring 네임스페이스의 cluster-monitoring-config 구성 맵에서 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

절차

1. user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

2. data/config.yaml 아래의 alertmanager 섹션에 enabled: true 및 enableAlertmanagerConfig: true 를 추가합니다.

```
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

클러스터에서 사용자 정의 프로젝트에 대해 **Alertmanager**의 전용 인스턴스를 활성화하려면 **enabled** 값을 **true** 로 설정합니다. 값을 **false** 로 설정하거나 사용자 정의 프로젝트에 대한 **Alertmanager**를 비활성화하려면 키를 완전히 생략합니다. 이 값을 **false** 로 설정하거나 키가 생략된 경우 사용자 정의 경고가 기본 플랫폼 **Alertmanager** 인스턴스로 라우팅됩니다.

2

사용자가 **AlertmanagerConfig** 오브젝트로 자체 경고 라우팅 구성을 정의하도록 **enableAlertmanagerConfig** 값을 **true** 로 설정합니다.

3. 파일을 저장하여 변경 사항을 적용합니다. 사용자 정의 프로젝트에 대한 **Alertmanager**의 전용 인스턴스가 자동으로 시작됩니다.

검증

- **user-workload Alertmanager** 인스턴스가 시작되었는지 확인합니다.

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

출력 예

```
NAME            VERSION   REPLICAS   AGE
user-workload   0.24.0    2          100s
```

6.4. 사용자 정의 프로젝트에 대한 경고 라우팅을 구성할 수 있는 사용자 권한 부여

사용자 정의 프로젝트에 대한 경고 라우팅을 구성할 수 있는 권한을 사용자에게 부여할 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 이미 존재하는 역할에 할당 중인 사용자 계정입니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

절차

- 사용자 정의 프로젝트의 사용자에게 **alert-routing-edit** 클러스터 역할을 할당합니다.

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

1

& lt;namespace >의 경우 네임스페이스를 **ns1** 과 같은 사용자 정의 프로젝트의 네임스페이스를 대체합니다. & lt;user > 의 경우 역할을 할당할 계정의 사용자 이름을 대체합니다.

추가 리소스

- 사용자 정의 프로젝트 모니터링 활성화
- 사용자 정의 프로젝트에 대한 경고 라우팅 생성

6.5. 다음 단계

- 경고 관리

7장. 메트릭 관리

메트릭을 수집하여 클러스터 구성 요소 및 자체 워크로드가 수행하는 방법을 모니터링할 수 있습니다.

7.1. 메트릭 이해

OpenShift Container Platform 4.11에서는 서비스 끝점을 통해 노출된 스크랩 메트릭으로 클러스터 구성 요소를 모니터링합니다. 사용자 정의 프로젝트에 대한 메트릭 컬렉션을 구성할 수도 있습니다.

애플리케이션 수준에서 Prometheus 클라이언트 라이브러리를 사용하여 자체 워크로드에 대해 제공할 메트릭을 정의할 수 있습니다.

OpenShift Container Platform에서 /metrics 표준 이름 아래에 HTTP 서비스 끝점을 통해 메트릭이 노출됩니다. http://<endpoint>/metrics에 대해 curl 쿼리를 실행하여 서비스에 사용 가능한 모든 메트릭을 나열할 수 있습니다. 예를 들어 prometheus-example-app 예제 서비스에 대한 경로를 노출한 다음 다음을 실행하여 사용 가능한 모든 메트릭을 확인할 수 있습니다.

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

출력 예

```
# 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
```

추가 리소스

- Prometheus 클라이언트 라이브러리 문서

7.2. 사용자 정의 프로젝트에 대한 메트릭 컬렉션 설정

ServiceMonitor 리소스를 생성하여 사용자 정의 프로젝트의 서비스 끝점에서 메트릭을 스크랩할 수 있습니다. 애플리케이션은 **Prometheus** 클라이언트 라이브러리를 사용하여 메트릭을 **/metrics** 표준 이름에 노출한다고 가정합니다.

이 섹션에서는 사용자 정의 프로젝트에 샘플 서비스를 배포한 후 서비스 모니터링 방법을 정의하는 **ServiceMonitor** 리소스를 만드는 방법에 대해 설명합니다.

7.2.1. 샘플 서비스 배포

사용자 정의 프로젝트에서 서비스 모니터링을 테스트하기 위해 샘플 서비스를 배포할 수 있습니다.

프로세스

1. 서비스 구성에 대한 **YAML** 파일을 생성합니다. 이 예에서는 **prometheus-example-app.yaml**이라고 합니다.

2. 파일에 다음 배포 및 서비스 구성 세부 정보를 추가합니다.

```
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
```

이 구성은 사용자 정의 **ns1** 프로젝트에 **prometheus-example-app**이라는 서비스를 배포합니다. 이 서비스는 사용자 정의 **version** 메트릭을 노출합니다.

3. 클러스터에 구성을 적용합니다.

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

서비스를 배포하는 데 시간이 다소 걸립니다.

4. **Pod**가 실행 중인지 확인할 수 있습니다.

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

출력 예

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

7.2.2. 서비스 모니터링 방법 지정

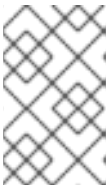
서비스에서 노출하는 메트릭을 사용하려면 **/metrics** 끝점에서 메트릭을 스크랩하도록 **OpenShift Container Platform** 모니터링을 구성해야 합니다. 서비스를 모니터링해야 하는 방법을 지정하는 **ServiceMonitor(CRD)** 또는 **Pod**를 모니터링해야 하는 방법을 지정하는 **PodMonitor CRD**를 사용하여

이 작업을 수행할 수 있습니다. 전자에는 **Service** 오브젝트가 필요하지만 후자에는 필요하지 않으며 **Prometheus**가 **Pod**에서 노출하는 메트릭 끝점에서 메트릭을 직접 스크랩할 수 있습니다.

다음 프로세스에서는 사용자 정의 프로젝트에서 서비스에 대한 **ServiceMonitor** 리소스를 생성하는 방법을 보여줍니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할 또는 **monitoring-edit** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 이 예제에서는 **prometheus-example-app** 샘플 서비스를 **ns1** 프로젝트에 배포했습니다.



참고

prometheus-example-app 샘플 서비스는 **TLS** 인증을 지원하지 않습니다.

절차

1. **ServiceMonitor** 리소스 구성에 대한 **YAML** 파일을 생성합니다. 이 예제에서 파일은 **example-app-service-monitor.yaml**이라고 합니다.
2. 다음 **ServiceMonitor** 리소스 구성 세부 정보를 추가합니다.

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

```
  selector:
    matchLabels:
      app: prometheus-example-app
```

이는 버전 메트릭이 포함된 **prometheus-example-app** 샘플 서비스에서 노출하는 메트릭을 스크랩하는 **ServiceMonitor** 리소스를 정의합니다.

참고

사용자 정의 네임스페이스의 **ServiceMonitor** 리소스는 동일한 네임스페이스에서 서비스만 검색할 수 있습니다. 즉 **ServiceMonitor** 리소스의 **namespaceSelector** 필드는 항상 무시됩니다.

3. 클러스터에 구성을 적용합니다.

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

ServiceMonitor 리소스를 배포하는 데 시간이 다소 걸립니다.

4. **ServiceMonitor** 리소스가 실행 중인지 확인할 수 있습니다.

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

출력 예

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

추가 리소스

- 사용자 정의 프로젝트 모니터링 활성화
- 사용자 정의 프로젝트의 **ServiceMonitor** 구성에서 **TLS**를 사용하여 메트릭을 스크랩하는 방법

- PodMonitor API
- ServiceMonitor API

7.3. 다음 단계

- 메트릭 쿼리

8장. 메트릭 쿼리

메트릭을 쿼리하여 클러스터 구성 요소 및 자체 워크로드가 수행하는 방법에 대한 데이터를 볼 수 있습니다.

8.1. 메트릭 쿼리 정보

OpenShift Container Platform 모니터링 대시보드를 사용하면 Pacemaker에서 표시되는 메트릭을 검사하기 위해 Prometheus Query Language(PromQL) 쿼리를 실행할 수 있습니다. 이 기능을 사용하면 클러스터 상태 및 모니터링 중인 모든 사용자 정의 워크로드에 대한 정보가 제공됩니다.

클러스터 관리자는 모든 핵심 OpenShift Container Platform 및 사용자 정의 프로젝트에 대한 메트릭을 쿼리할 수 있습니다.

개발자는 메트릭을 쿼리할 때 프로젝트 이름을 지정해야 합니다. 선택한 프로젝트의 메트릭을 확인하는 데 필요한 권한이 있어야 합니다.

8.1.1. 클러스터 관리자로서 모든 프로젝트의 메트릭 쿼리

클러스터 관리자 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 Metrics UI에서 모든 기본 OpenShift Container Platform 및 사용자 정의 프로젝트에 대한 메트릭에 액세스할 수 있습니다.

사전 요구 사항

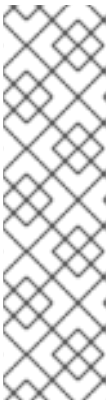
- cluster-admin 클러스터 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

절차

1. OpenShift Container Platform 웹 콘솔에서 관리자 화면을 선택합니다.
2. Observe → Metrics 를 선택합니다.

3. 커서에 표시기 삽입을 선택하여 사전 정의된 쿼리 목록을 확인합니다.

4. 사용자 정의 쿼리를 생성하려면 표현식 필드에 **PromQL(Prometheus Query Language)** 쿼리를 추가합니다.



참고

PromQL 표현식을 입력하면 드롭다운 목록에 자동 완성 제안이 표시됩니다. 이러한 제안에는 함수, 메트릭, 라벨, 시간 토큰이 있습니다. 키보드 화살표를 사용하여 이러한 제안된 항목 중 하나를 선택한 다음 **Enter**를 눌러 식에 항목을 추가할 수 있습니다.**You can use the keyboard arrows to select one of these suggested items and then press Enter to add the item to your expression.** 또한 제안된 항목에 마우스 포인터를 이동하여 해당 항목에 대한 간략한 설명을 볼 수도 있습니다.

5. 여러 쿼리를 추가하려면 쿼리 추가를 선택합니다.

6. 기존 쿼리를 복제하려면 쿼리 옆에 있는

를 선택한 다음 쿼리 중복 을 선택합니다.

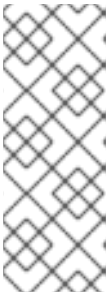
7. 쿼리를 삭제하려면 쿼리 옆에 있는

를 선택한 다음 쿼리 삭제를 선택합니다.

8. 쿼리 실행을 비활성화하려면 쿼리 옆에 있는

를 선택하고 쿼리 비활성화를 선택합니다.

9. 생성한 쿼리를 실행하려면 쿼리 실행 을 선택합니다. 쿼리의 메트릭은 플롯에 시각화됩니다. 쿼리가 유효하지 않으면 **UI**에 오류 메시지가 표시됩니다.



참고

대량의 데이터에서 작동하는 쿼리 시간이 초과되거나 시계열 그래프에 있을 때 브라우저가 과부하될 수 있습니다. 이를 방지하려면 그래프 숨기기를 선택하고 메트릭 테이블만 사용하여 쿼리를 조정합니다. 그런 다음 실행 가능한 쿼리를 검색한 후 플롯을 활성화하여 그래프를 그립니다.

10. 선택 사항: 이제 페이지 URL에 실행한 쿼리가 포함되어 있습니다. 나중에 이 쿼리 세트를 다시 사용하려면 이 URL을 저장합니다.

추가 리소스

- PromQL 쿼리 생성에 대한 자세한 내용은 Prometheus 쿼리 설명서를 참조하십시오.

8.1.2. 개발자로 사용자 정의 프로젝트의 메트릭 쿼리

사용자 정의 프로젝트의 메트릭에 대해 개발자 또는 프로젝트에 대한 보기 권한이 있는 사용자로 액세스할 수 있습니다.

개발자 관점에서 Metrics UI에는 선택한 프로젝트에 대한 사전 정의된 CPU, 메모리, 대역폭 및 네트워크 패킷 쿼리가 포함되어 있습니다. 프로젝트에 대한 CPU, 메모리, 대역폭, 네트워크 패킷 및 애플리케이션 메트릭에 대해 사용자 정의 Prometheus Query Language(PromQL) 쿼리를 실행할 수도 있습니다.

참고

개발자는 관리자 관점이 아닌 개발자 관점만 사용할 수 있습니다. 개발자는 웹 콘솔의 사용자 정의 프로젝트에 대해 한 번에 한 프로젝트의 메트릭 만 쿼리할 수 있습니다.

사전 요구 사항

- 개발자로 또는 메트릭을 확인하는 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 사용자 정의 프로젝트에 서비스를 배포했습니다.

- 서비스에서 모니터링 방법을 정의하는 데 사용할 **ServiceMonitor CRD**(사용자 정의 리소스 정의(**Custom Resource Definition**))가 생성되었습니다.

프로세스

1. **OpenShift Container Platform** 웹 콘솔에서 개발자 화면을 선택합니다.
2. **Observe** → **Metrics** 를 선택합니다.
3. **Project:** 목록에서 메트릭을 보려는 프로젝트를 선택합니다.
4. 쿼리 선택 목록에서 쿼리를 선택하거나 **PromQL** 표시를 선택하여 선택한 쿼리에 따라 사용자 정의 **PromQL** 쿼리를 만듭니다.
5. 선택 사항: 쿼리 선택 목록에서 사용자 지정 쿼리를 선택하여 새 쿼리를 입력합니다. 입력하는 경우 드롭다운 목록에 자동 완성 제안이 표시됩니다. 이러한 제안에는 함수 및 메트릭이 포함됩니다. 제안된 항목을 클릭하여 선택합니다.

참고

개발자 관점에서는 한 번에 하나의 쿼리만 실행할 수 있습니다.

추가 리소스

- **PromQL** 쿼리 생성에 대한 자세한 내용은 **Prometheus** 쿼리 설명서를 참조하십시오.

8.1.3. 시각화된 메트릭 살펴보기

쿼리를 실행하면 대화형 플롯에 메트릭이 표시됩니다. 플롯의 **X**축은 시간을 나타내며, **Y**축은 메트릭 값을 나타냅니다. 각 메트릭은 그래프에 색상이 지정된 선으로 표시됩니다. 대화형으로 플롯을 조작하고 메트릭을 살펴볼 수 있습니다.

절차

관리자 관점에서:

1. 처음에 활성화된 모든 쿼리의 모든 메트릭이 플롯에 표시됩니다. 표시된 메트릭을 선택할 수 있습니다.

참고

기본적으로 쿼리 테이블은 모든 메트릭과 해당 현재 값을 나열하는 확장된 보기를 표시합니다. 쿼리에 대해 확장된 보기를 최소화하려면 ˅를 선택할 수 있습니다.

- 쿼리에서 모든 메트릭을 숨기려면 쿼리에 대해

을 클릭하고 모든 시리즈 숨기기를 클릭합니다.

- 특정 메트릭을 숨기려면 쿼리 테이블로 이동하여 메트릭 이름 근처에 있는 색상이 지정된 사각형을 클릭합니다.

2. 플롯을 확장하고 시간 범위를 변경하려면 다음 중 하나를 수행하십시오.

- 수평으로 플롯을 클릭하고 드래그하여 시간 범위를 시각적으로 선택합니다.

- 왼쪽 상단 코너의 메뉴를 사용하여 시간 범위를 선택합니다.

3. 시간 범위를 재설정하려면 확대/축소 재설정을 선택합니다.

4. 특정 시점에서 모든 쿼리에 대한 출력을 표시하려면 해당 시점에서 플롯에 마우스 커서를 유지합니다. 쿼리 출력이 팝업 상자에 나타납니다.

5. 플롯을 숨기려면 그래프 숨기기를 선택합니다.

Developer 관점에서:

1. 플롯을 확장하고 시간 범위를 변경하려면 다음 중 하나를 수행하십시오.

   - 수평으로 플롯을 클릭하고 드래그하여 시간 범위를 시각적으로 선택합니다.
   - 왼쪽 상단 코너의 메뉴를 사용하여 시간 범위를 선택합니다.

2. 시간 범위를 재설정하려면 확대/축소 재설정을 선택합니다.

3. 특정 시점에서 모든 쿼리에 대한 출력을 표시하려면 해당 시점에서 플롯에 마우스 커서를 유지합니다. 쿼리 출력이 팝업 상자에 나타납니다.

추가 리소스

- **PromQL** 인터페이스 사용에 대한 자세한 내용은 메트릭 쿼리 를 참조하십시오.
- 관리자로 모든 프로젝트의 메트릭에 액세스하는 방법에 대한 자세한 내용은 관리자 권한으로 모든 프로젝트의 메트릭 쿼리 를 참조하십시오.
- 개발자 또는 권한 있는 사용자로 비 클러스터 메트릭에 액세스하는 방법에 대한 자세한 내용은 개발자로 사용자 정의 프로젝트의 메트릭 쿼리 를 참조하십시오.

8.2. 다음 단계

- 메트릭 대상 관리

9장. 메트릭 대상 관리

OpenShift Container Platform 모니터링은 노출된 서비스 끝점에서 데이터를 스크랩하여 대상 클러스터 구성 요소에서 지표를 수집합니다.

OpenShift Container Platform 웹 콘솔의 관리자 관점에서 Metrics 대상 페이지를 사용하여 현재 스크랩 대상인 엔드 포인트를 확인, 검색 및 필터링할 수 있습니다. 이 경우 문제를 식별하고 해결하는 데 도움이 됩니다. 예를 들어 대상 끝점의 현재 상태를 보고 OpenShift Container Platform 모니터링이 대상 구성 요소에서 메트릭을 스크랩할 수 없는 시기를 확인할 수 있습니다.

Metrics 대상 페이지에는 기본 OpenShift Container Platform 프로젝트 및 사용자 정의 프로젝트의 대상이 표시됩니다.

9.1. 관리자 관점에서 지표 대상 페이지에 액세스

OpenShift Container Platform 웹 콘솔의 관리자 화면에서 Metrics 대상 페이지를 볼 수 있습니다.

사전 요구 사항

- 메트릭 대상을 보려는 프로젝트의 관리자로 클러스터에 액세스할 수 있습니다.

절차

- 관리자 관점에서 Observe → Targets 를 선택합니다. Metrics 대상 페이지는 메트릭에 대해 스크랩되는 모든 서비스 끝점 대상 목록으로 열립니다.

9.2. 메트릭 대상 검색 및 필터링

지표 대상 목록은 길 수 있습니다. 다양한 기준에 따라 이러한 대상을 필터링하고 검색할 수 있습니다.

관리자 관점에서 Metrics 대상 페이지는 기본 OpenShift Container Platform 및 사용자 정의 프로젝트의 대상에 대한 세부 정보를 제공합니다. 이 페이지에는 각 대상에 대해 다음 정보가 나열됩니다.

- 스크랩되는 서비스 끝점 URL

- **ServiceMonitor** 구성 요소를 모니터링
- 대상의 **up** 또는 **down** 상태입니다.
- 네임스페이스
- 마지막 스크랩 시간
- 마지막 스크랩의 기간

대상 목록을 상태 및 소스로 필터링할 수 있습니다. 다음 필터링 옵션을 사용할 수 있습니다.

- 상태 필터:
  - 위로. 현재 대상이 작동 중이고 메트릭에 대해 적극적으로 스크랩되고 있습니다.
  - 아래로. 현재 대상이 다운되어 메트릭에 대해 스크랩되지 않습니다.
- 소스 필터:
  - 플랫폼. 플랫폼 수준 대상은 기본 **OpenShift Container Platform** 프로젝트에만 관련이 있습니다. 이러한 프로젝트는 핵심 **OpenShift Container Platform** 기능을 제공합니다.
  - 사용자 사용자 대상은 사용자 정의 프로젝트와 관련이 있습니다. 이러한 프로젝트는 사용자가 생성하며 사용자 지정할 수 있습니다.

검색 상자를 사용하여 대상 이름 또는 레이블로 대상을 찾을 수도 있습니다. 검색 상자 메뉴에서 텍스트 또는 레이블 을 선택하여 검색을 제한합니다.

9.3. 대상에 대한 자세한 정보 얻기

대상 세부 정보 페이지에서 메트릭 대상에 대한 자세한 정보를 볼 수 있습니다.

사전 요구 사항

- 메트릭 대상을 보려는 프로젝트의 관리자로 클러스터에 액세스할 수 있습니다.

절차

관리자 관점에서 대상에 대한 자세한 정보를 보려면 다음을 수행합니다.

1. OpenShift Container Platform 웹 콘솔을 열고 Observe → Targets 로 이동합니다.
2. 선택 사항: 필터 목록에서 필터를 선택하여 상태 및 소스로 대상을 필터링 합니다.
3. 선택 사항: 검색 상자 옆에 있는 텍스트 또는 라벨 필드를 사용하여 이름 또는 레이블로 대상을 검색합니다.
4. 선택 사항: Endpoint,Status,Namespace,Last Scrape, Scrape 열 헤더 중 하나 이상을 클릭하여 대상을 정렬합니다.
5. 대상의 끝점 열에서 URL을 클릭하여 대상 세부 정보 페이지로 이동합니다. 이 페이지에서는 다음을 포함하여 대상에 대한 정보를 제공합니다.
   - 메트릭에 대해 스크랩되는 끝점 URL
   - 대상의 현재 Up 또는 Down 상태
   - 네임스페이스에 대한 링크
   - ServiceMonitor 세부 정보 링크

- 대상에 연결된 라벨

- 메트릭에 대해 대상이 스크랩된 가장 최근 시간입니다.

9.4. 다음 단계

- 경고 관리

10장. 경고 관리

OpenShift Container Platform 4.11에서 경고 UI를 사용하면 경고, 음소거, 경고 규칙을 관리할 수 있습니다.

- 경고 규칙. 경고 규칙에는 클러스터 내에서 특정 상태를 설명하는 일련의 조건이 포함되어 있습니다. 이러한 조건이 true이면 경고가 트리거됩니다. 경고 규칙은 경고의 라우팅 방법을 정의하는 심각도를 할당할 수 있습니다.

- 경고. 경고 규칙에 정의된 조건이 true이면 경고가 실행됩니다. 경고는 일련의 상황이 OpenShift Container Platform 클러스터 내에서 발생한다는 통지를 제공합니다.

- 음소거. 경고 조건이 true일 때 알림이 전송되는 것을 방지하기 위해 경고에 음소거를 적용할 수 있습니다. 기본 문제를 해결하는 동안 초기 알림 후 경고를 음소거할 수 있습니다.

참고

경고 UI에서 사용할 수 있는 경고, 음소거, 경고 규칙은 액세스할 수 있는 프로젝트와 관련이 있습니다. 예를 들어 cluster-admin 권한으로 로그인한 경우 모든 경고, 음소거 및 경고 규칙에 액세스할 수 있습니다.

관리자가 아닌 사용자인 경우 다음 사용자 역할이 할당된 경우 경고를 생성하고 음소거할 수 있습니다.

- Alertmanager에 액세스할 수 있는 cluster-monitoring-view 클러스터 역할
- 웹 콘솔의 관리자 화면에서 경고를 생성하고 음소거할 수 있는 monitoring-alertmanager-edit 역할
- 웹 콘솔의 개발자 화면에서 경고를 생성하고 음소거할 수 있는 monitoring-rules-edit 클러스터 역할

10.1. 관리자 및 개발자 관점에서 경고 UI에 액세스

경고 UI는 OpenShift Container Platform 웹 콘솔에서 관리자 관점 및 개발자 관점을 통해 액세스할 수 있습니다.

- 관리자 관점에서 **Observe → Alerting** 을 선택합니다. 이 관점에서 경고 **UI**의 세 가지 주요 페이지는 경고, 음소거 및 경고 규칙 페이지입니다.

- 개발자 관점에서 **Observe → < project_name> →** 경고를 선택합니다. 이 관점에서 경고, 음소거 및 경고 규칙은 모두 경고 페이지에서 관리됩니다. 경고 페이지에 표시된 결과는 선택한 프로젝트에 특정적입니다.

참고

개발자 관점에서는 프로젝트 **Project:** 목록에서 액세스할 수 있는 핵심 **OpenShift Container Platform**및 사용자 정의 프로젝트에서 선택할 수 있습니다. 그러나 **cluster-admin** 권한이 없는 경우 핵심 **OpenShift Container Platform** 프로젝트와 관련된 경고, 음소거, 경고 규칙이 표시되지 않습니다.

10.2. 경고, 음소거, 경고 규칙 검색 및 필터링

경고 **UI**에 표시되는 경고, 음소거 및 경고 규칙을 필터링할 수 있습니다. 이 섹션에서는 사용 가능한 필터링 옵션 각각에 대해 설명합니다.

경고 필터 이해

관리자 관점에서 경고 **UI**의 경고 페이지는 기본 **OpenShift Container Platform** 및 사용자 정의 프로젝트와 관련된 경고에 대한 세부 정보를 제공합니다. 페이지에는 각 경고에 대한 심각도, 상태 및 소스가 요약되어 있습니다. 현재 상태로 경고가 표시되는 시간도 표시됩니다.

경고 상태, 심각도 및 소스로 필터링할 수 있습니다. 기본적으로 실행되는 플랫폼만 표시됩니다. 다음은 각 경고 필터링 옵션을 설명합니다.

- 경고 상태 필터:
  - 실행. 경고 조건이 **true** 이므로 경고가 실행되고 기간 동안 선택 사항이 전달됩니다. 상태가 **true**로 유지되는 동안 경고는 계속 실행됩니다.
  - 보류 중. 경고는 활성화되어 있지만 실행되기 전에 경고 규칙에 지정된 기간 동안 대기 중입니다.

- 음소거. 이제 정의된 기간 동안 경고가 음소거되었습니다. 사용자가 정의한 라벨 선택기 집합에 따라 일시적으로 음소거가 경고를 음소거합니다. 나열된 모든 값 또는 정규식과 일치하는 경고에 대해 알림이 전송되지 않습니다.

- 심각도 필터:
  - 심각. 경고를 트리거한 조건으로 심각한 영향을 미칠 수 있습니다. 경고는 실행 시 즉각적인 주의가 필요하며 일반적으로 개인 또는 문제 대응팀으로 호출됩니다.
  - 경고. 경고는 문제가 발생하지 않도록 주의가 필요할 수 있는 사항에 대한 경고 알림을 제공합니다. 경고는 일반적으로 즉각적이 아닌 검토에 대해 티켓 시스템으로 라우팅됩니다.
  - 정보. 경고는 정보 목적으로만 제공됩니다.
  - 없음. 경고에 정의된 심각도가 없습니다.
  - 사용자 정의 프로젝트와 관련된 경고에 대한 사용자 정의 심각도 정의를 생성할 수도 있습니다.

- 소스 필터:
  - 플랫폼. 플랫폼 수준 경고는 기본 **OpenShift Container Platform** 프로젝트에만 관련이 있습니다. 이러한 프로젝트는 핵심 **OpenShift Container Platform** 기능을 제공합니다.
  - 사용자 사용자 경고는 사용자 정의 프로젝트와 관련되어 있습니다. 이러한 경고는 사용자가 생성하며 사용자 정의할 수 있습니다. 사용자 정의 워크로드 모니터링은 설치 후 활성화하여 자체 워크로드에 관찰 기능을 제공할 수 있습니다.

음소거 필터 이해

관리자 관점에서 경고 **UI**의 음소거 페이지는 기본 **OpenShift Container Platform** 및 사용자 정의 프로젝트의 경고에 적용된 음소거에 대한 세부 정보를 제공합니다. 페이지에는 각 음소거의 상태 요약과 음소거가 종료되는 시점이 포함되어 있습니다.

음소거 상태별로 필터링할 수 있습니다. 기본적으로 활성 및 보류 중 음소거만 표시됩니다. 다음은 각

음소거 상태 필터 옵션을 설명합니다.

- 음소거 상태 필터:
  - 활성. 음소거가 활성 상태이며 음소거가 만료될 때까지 경고가 음소거됩니다.
  - 보류 중. 음소거 예정되어 있고 아직 활성화되지 않았습니다.
  - 만료. 경고 조건이 **true**이면 음소거가 만료되고 알림이 전송됩니다.

경고 규칙 필터 이해

관리자 관점에서 경고 **UI**의 경고 규칙 페이지는 기본 **OpenShift Container Platform** 및 사용자 정의 프로젝트와 관련된 경고 규칙에 대한 세부 정보를 제공합니다. 페이지에는 각 경고 규칙의 상태, 심각도 및 소스가 요약되어 있습니다.

경고 상태, 심각도 및 소스에 따라 경고 규칙을 필터링할 수 있습니다. 기본적으로 플랫폼 경고 규칙만 표시됩니다. 다음은 각 경고 규칙 필터링 옵션을 설명합니다.

- 경고 상태 필터:
  - 실행. 경고 조건이 **true** 이므로 경고가 실행되고 기간 동안 선택 사항이 전달됩니다. 상태가 **true**로 유지되는 동안 경고는 계속 실행됩니다.
  - 보류 중. 경고는 활성화되어 있지만 실행되기 전에 경고 규칙에 지정된 기간 동안 대기 중입니다.
  - 음소거. 이제 정의된 기간 동안 경고가 음소거되었습니다. 사용자가 정의한 라벨 선택기 집합에 따라 일시적으로 음소거가 경고를 음소거합니다. 나열된 모든 값 또는 정규식과 일치하는 경고에 대해 알림이 전송되지 않습니다.
  - 실행하지 않음. 경고가 실행되지 않습니다.

- 심각도 필터:
  - 심각. 경고 규칙에 정의된 조건이 심각한 영향을 미칠 수 있습니다. **true**인 경우 이러한 조건에는 즉각적인 주의가 필요합니다. 규칙과 관련된 경고는 일반적으로 개인 또는 문제 대응팀으로 호출됩니다.
  - 경고. 경고 규칙에 정의된 조건은 문제가 발생하지 않도록 주의해야 할 수 있습니다. 규칙과 관련된 경고는 일반적으로 즉각적이 아닌 검토에 대해 티켓 시스템으로 라우팅됩니다.
  - 정보. 경고 규칙은 정보성 경고만 제공합니다.
  - 없음. 경고 규칙에는 정의된 심각도가 없습니다.
  - 사용자 정의 프로젝트와 관련된 경고 규칙에 대한 사용자 정의 심각도 정의를 생성할 수도 있습니다.
- 소스 필터:
  - 플랫폼. 플랫폼 수준 경고 규칙은 기본 **OpenShift Container Platform** 프로젝트에만 관련이 있습니다. 이러한 프로젝트는 핵심 **OpenShift Container Platform** 기능을 제공합니다.
  - 사용자 사용자 정의 워크로드 경고 규칙은 사용자 정의 프로젝트와 관련이 있습니다. 이러한 경고 규칙은 사용자가 생성하며 사용자 정의할 수 있습니다. 사용자 정의 워크로드 모니터링은 설치 후 활성화하여 자체 워크로드에 관찰 기능을 제공할 수 있습니다.

개발자 관점에서 경고, 음소거, 경고 규칙 검색 및 필터링

개발자 관점에서 경고 **UI**의 경고 페이지에서는 선택한 프로젝트와 관련된 경고 및 음소거의 결합된 보기를 제공합니다. 표시된 경고마다 관리 경고 규칙에 대한 링크가 제공됩니다.

이 보기에서는 경고 상태 및 심각도로 필터링할 수 있습니다. 기본적으로 프로젝트에 액세스할 수 있는 권한이 있는 경우 선택한 프로젝트의 모든 경고가 표시됩니다. 이러한 필터는 관리자 관점에서 설명한 항목과 동일합니다.

10.3. 경고, 음소거 및 경고 규칙에 대한 정보 받기

경고 UI는 경고 및 관리 경고 규칙과 음소거에 대한 자세한 정보를 제공합니다.

사전 요구 사항

- 개발자로 또는 메트릭을 확인하는 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

프로세스

관리자 관점에서 경고에 대한 정보를 얻으려면 다음을 수행합니다.

1. OpenShift Container Platform 웹 콘솔을 열고 **Observe** → **Alerting** → 경고 페이지로 이동합니다.
2. 선택 사항: 검색 목록에서 이름 필드를 사용하여 이름별로 경고를 검색합니다.
3. 선택 사항: 필터 목록에서 필터를 선택하여 상태, 심각도 및 소스별로 경고를 필터링합니다.
4. 선택 사항: 이름, 심각도, 상태 및 소스 열 헤더 중 하나 이상을 클릭하여 경고를 정렬합니다.
5. 경고 세부 정보 페이지로 이동하도록 경고 이름을 선택합니다. 페이지에는 경고 시계열 데이터를 설명하는 그래프가 포함되어 있습니다. 또한 다음을 포함하여 경고에 대한 정보를 제공합니다.
   - 경고에 대한 설명
   - 경고와 관련된 메시지
   - 경고에 연결된 라벨
   - 관리 경고 규칙에 대한 링크

- 존재하는 경우 경고에 대한 음소거

관리자 관점에서 음소거에 대한 정보를 얻으려면 다음을 수행합니다.

1. **Observe → Alerting → Silences** 페이지로 이동합니다.
2. 선택 사항: 이름으로 검색 필드를 사용하여 이름으로 음소거를 필터링합니다.
3. 선택 사항: 필터 목록에서 필터를 선택하여 상태별로 음소거를 필터링합니다. 기본적으로 활성 및 보류 중 필터가 적용됩니다.
4. 선택 사항: 이름, 경고 실행 및 상태 열 헤더를 하나 이상 클릭하여 음소거를 정렬합니다.
5. 음소거의 이름을 선택하여 음소거 상세 정보 페이지로 이동합니다. 페이지에는 다음과 같은 세부 정보가 포함됩니다.
   - 경고 사양
   - 시작 시간
   - 종료 시간
   - 음소거 상태
   - 실행 경고 수 및 목록

관리자 관점에서 경고 규칙에 대한 정보를 얻으려면 다음을 수행합니다.

1. **Observe → Alerting → Alerting Rules** 페이지로 이동합니다.

2. 선택 사항: 필터 목록에서 필터를 선택하여 상태, 심각도 및 소스로 경고 규칙을 필터링합니다.

3. 선택 사항: 이름, 심각도, 경고 상태 및 소스 열 헤더 중 하나 이상을 클릭하여 경고 규칙을 정렬합니다.

4. 경고 규칙의 이름을 선택하여 경고 규칙 세부 정보 페이지로 이동합니다. 페이지는 경고 규칙에 대한 다음 세부 정보를 제공합니다.

- 경고 규칙 이름, 심각도 및 설명
- 경고를 실행하기 위한 조건을 정의하는 표현식
- 경고가 실행되기 위한 조건이 **true**여야 하는 시간
- 경고 규칙에 의해 관리되는 각 경고에 대한 그래프로, 경고가 실행되는 값을 표시
- 경고 규칙에 의해 관리되는 모든 경고의 테이블

개발자 관점에서 경고, 음소거 및 경고 규칙에 대한 정보를 얻으려면 다음을 수행합니다.

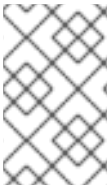
1. **Observe** → **< project_name>** → 경고 페이지로 이동합니다.

2. 경고, 음소거 또는 경고 규칙에 대한 세부 정보를 표시합니다.

- 경고 세부 정보는 경고 이름 왼쪽의 >를 선택한 다음 목록에 있는 경고를 선택하여 볼 수 있습니다.
- 음소거 상세 정보는 경고 세부 정보 페이지의 음소거 기준 섹션에서 음소거를 선택하여 볼 수 있습니다. 음소거 상세 정보 페이지에는 다음 정보가 포함됩니다.

- 경고 사양
- 시작 시간
- 종료 시간
- 음소거 상태
- 실행 경고 수 및 목록

- 경고 규칙 세부 정보는 경고 페이지에 있는 경고 오른쪽의 
메뉴에서 경고 규칙 보기를 선택하여 볼 수 있습니다.



참고

선택한 프로젝트와 관련된 경고, 음소거, 경고 규칙만 개발자 관점에 표시됩니다.

추가 리소스

- 특정 **OpenShift Container Platform** 모니터링 경고를 트리거하는 문제를 진단하고 해결하는 데 도움이 되는 **Cluster Monitoring Operator runbook** 을 참조하십시오.

10.4. 음소거 관리

경고가 실행될 때 경고에 대한 알림을 수신하지 못하도록 음소거를 생성할 수 있습니다. 기본 문제를 해결하는 동안 처음 알림을 받은 후 음소거하는 것이 유용할 수 있습니다.

음소거를 생성할 때 즉시 또는 나중에 활성 상태가 되는지 여부를 지정해야 합니다. 또한 음소거가 만료된 후 기간을 설정해야 합니다.

기존 음소거를 보고 편집하고 만료할 수 있습니다.

10.4.1. 음소거 경고

특정 경고를 음소거하거나 사용자가 정의한 사양과 일치하는 경고를 음소거할 수 있습니다.

사전 요구 사항

- 클러스터 관리자이며 **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 관리자가 아닌 사용자이며 다음 사용자 역할이 있는 사용자로 클러스터에 액세스할 수 있습니다.
  - **Alertmanager**에 액세스할 수 있는 **cluster-monitoring-view** 클러스터 역할입니다.
  - 웹 콘솔의 관리자 화면에서 경고를 생성하고 음소거할 수 있는 **monitoring-alertmanager-edit** 역할입니다.
  - 웹 콘솔의 개발자 화면에서 경고를 생성하고 음소거할 수 있는 **monitoring-rules-edit** 클러스터 역할입니다.

프로세스

특정 경고를 음소거하려면 다음을 수행합니다.

- 관리자 관점에서:
  1. **OpenShift Container Platform** 웹 콘솔의 **Observe** → **Alerting** → 경고 페이지로 이동합니다.
  2. 음소거할 경고의 경우 오른쪽 열에 있는

     를 선택하고 음소거 경고를 선택합니다. 음소거 경고 형식은 선택한 경고에 대해 미리 채워진 사양으로 표시됩니다.

3. 선택 사항: 음소거를 수정합니다.

4. 음소거를 생성하기 전에 코멘트를 추가해야 합니다.

5. 음소거를 생성하려면 음소거를 선택합니다.

- 개발자 화면에서:

1. OpenShift Container Platform 웹 콘솔의 **Observe → < project_name > →** 경고 페이지로 이동합니다.

2. 경고 이름의 왼쪽에 있는 >를 선택하여 경고에 대한 세부 사항을 확장합니다. 경고 세부 정보 페이지를 열어 확장된 보기에서 경고의 이름을 선택합니다.

3. 음소거 경고를 선택합니다. 음소거 경고 형식은 선택한 경고에 대해 미리 채워진 사양으로 표시됩니다.

4. 선택 사항: 음소거를 수정합니다.

5. 음소거를 생성하기 전에 코멘트를 추가해야 합니다.

6. 음소거를 생성하려면 음소거를 선택합니다.

관리자 관점에서 경고 사양을 생성하여 일련의 경고를 음소거하려면 다음을 수행합니다.

1. OpenShift Container Platform 웹 콘솔의 **Observe → Alerting → Silences** 페이지로 이동합니다.

2. 음소거 상태 만들기를 선택합니다.

3.

음소거 상태 만들기 형식에서 경고의 일정, 기간 및 라벨 세부 정보를 설정합니다. 음소거에 대한 코멘트를 추가해야 합니다.

4. 이전 단계에서 입력한 라벨 셀터와 일치하는 경고에 대한 음소거를 생성하려면 음소거를 선택합니다.

10.4.2. 음소거 편집

음소거를 편집하면 기존 음소거가 만료되고 변경된 구성으로 새 파일을 만들 수 있습니다.

프로세스

관리자 관점에서 음소거를 편집하려면 다음을 수행합니다.

1. **Observe → Alerting → Silences** 페이지로 이동합니다.

2. 수정하려는 음소거의 경우 마지막 열에서

를 선택하고 음소거 편집을 선택합니다.

또는 음소거에 대한 음소거 상세 정보 페이지에서 동작 → 음소거 편집을 선택할 수 있습니다.

3. 음소거 편지 페이지에서 변경 사항을 입력하고 음소거를 선택합니다. 이를 통해 기존 음소거가 만료되고 선택한 구성으로 생성됩니다.

개발자 관점에서 음소거를 편집하려면 다음을 수행합니다.

1. **Observe → < project_name> →** 경고 페이지로 이동합니다.

2. 경고 이름의 왼쪽에 있는 >를 선택하여 경고에 대한 세부 사항을 확장합니다. 경고 세부 정보 페이지를 열어 확장된 보기에서 경고의 이름을 선택합니다.

3. 해당 페이지의 음소거 기준 섹션에서 음소거의 이름을 선택하여 음소거에 대한 음소거 상세

정보 페이지로 이동합니다.

4. 음소거의 이름을 선택하여 음소거 상세 정보 페이지로 이동합니다.

5. 음소거에 대한 음소거 상세 정보 페이지에서 동작 → 음소거 편집을 선택합니다.

6. 음소거 편지 페이지에서 변경 사항을 입력하고 음소거를 선택합니다. 이를 통해 기존 음소거가 만료되고 선택한 구성으로 생성됩니다.

10.4.3. 음소거 만료

음소거를 만료할 수 있습니다. 음소거를 만료하면 영구적으로 비활성화됩니다.

참고

만료된 경고를 삭제할 수 없습니다. 120시간 이상 경과한 음소거는 가비지 수집됩니다.

프로세스

관리자 관점에서 음소거를 만료하려면 다음을 수행합니다.

1. **Observe** → **Alerting** → **Silences** 페이지로 이동합니다.

2. 수정하려는 음소거의 경우 마지막 열에서

를 선택하고 음소거 만료를 선택합니다.

또는 음소거에 대한 음소거 상세 정보 페이지에서 동작 → 음소거 만료를 선택할 수 있습니다.

개발자 관점에서의 음소거를 만료하려면 다음을 수행합니다.

1. **Observe → < project_name> → 경고 페이지로 이동합니다.**

2. **경고 이름의 왼쪽에 있는 >를 선택하여 경고에 대한 세부 사항을 확장합니다. 경고 세부 정보 페이지를 열어 확장된 보기에서 경고의 이름을 선택합니다.**

3. **해당 페이지의 음소거 기준 섹션에서 음소거의 이름을 선택하여 음소거에 대한 음소거 상세 정보 페이지로 이동합니다.**

4. **음소거의 이름을 선택하여 음소거 상세 정보 페이지로 이동합니다.**

5. **음소거에 대한 음소거 상세 정보 페이지에서 동작 → 음소거 만료를 선택합니다.**

10.5. 사용자 정의 프로젝트에 대한 경고 규칙 관리

OpenShift Container Platform 모니터링에는 기본 경고 규칙 집합이 제공됩니다. 클러스터 관리자는 기본 경고 규칙을 볼 수 있습니다.

OpenShift Container Platform 4.11에서는 사용자 정의 프로젝트에서 경고 규칙을 생성, 보기, 편집 및 제거할 수 있습니다.

경고 규칙 고려 사항

- 기본 경고 규칙은 특히 **OpenShift Container Platform** 클러스터에 사용됩니다.

- 일부 경고 규칙은 의도적으로 이름이 동일합니다. 임계값, 다른 심각도 또는 둘 다의 경우와 동일한 이벤트에 대한 경고를 보냅니다.

- 억제 규칙은 심각도가 높은 경고가 실행될 때 실행되는 심각도가 낮은 경고에 대한 알림을 방지합니다.

10.5.1. 사용자 정의 프로젝트에 대한 경고 최적화

경고 규칙을 생성할 때 다음 권장 사항을 따라 자체 프로젝트에 대한 경고를 최적화할 수 있습니다.

- 프로젝트에 생성하는 경고 규칙 수를 최소화합니다. 사용자에게 영향을 미치는 조건에 대해 알리는 경고 규칙을 생성합니다. 영향을 주지 않는 조건에 대한 여러 경고를 생성하면 관련 경고를 알리기가 더 어렵습니다.

- 원인 대신 증상에 대한 경고 규칙을 만듭니다. 기본 원인과 관계없이 조건을 알리는 경고 규칙을 만듭니다. 그러면 원인을 조사할 수 있습니다. 각 항목이 특정 원인에만 관련된 경우 더 많은 경고 규칙이 필요합니다. 그러면 일부 원인으로 인해 누락될 가능성이 큽니다.

- 경고 규칙을 작성하기 전에 계획합니다. 어떤 증상이 사용자에게 중요한지, 발생 시 어떤 조치를 수행할지를 결정합니다. 그런 다음 각 증상에 대한 경고 규칙을 구축합니다.

- 명확한 경고 메시지를 제공합니다. 경고 메시지에서 증상과 권장 작업을 설명합니다.

- 경고 규칙에 심각도 수준을 포함합니다. 경고의 심각도는 보고된 증상이 발생하는 경우 어떻게 대응해야 하는지에 따라 다릅니다. 예를 들어 증상이 개인 또는 문제 대응팀에서 즉각적인 주의가 필요한 경우 심각한 경고를 트리거해야 합니다.

추가 리소스

- 경고 최적화에 대한 추가 지침은 **Prometheus** 경고 문서를 참조하십시오.

- **OpenShift Container Platform 4.11** 모니터링 아키텍처에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.

10.5.2. 사용자 정의 프로젝트에 대한 경고 규칙 생성 정보

사용자 정의 프로젝트에 대한 경고 규칙을 생성하는 경우 새 규칙을 정의할 때 다음 주요 동작 및 중요한 제한 사항을 고려하십시오.

- 사용자 정의 경고 규칙은 코어 플랫폼 모니터링의 기본 메트릭 외에도 자체 프로젝트에서 노출하는 메트릭을 포함할 수 있습니다. 다른 사용자 정의 프로젝트의 메트릭을 포함할 수 없습니다.

  예를 들어 **ns1** 사용자 정의 프로젝트에 대한 경고 규칙은 **CPU** 및 메모리 지표와 같은 코어 플랫폼 메트릭 외에도 **ns1** 프로젝트에서 노출하는 메트릭을 사용할 수 있습니다. 그러나 규칙에는 다른 **ns2** 사용자 정의 프로젝트의 메트릭을 포함할 수 없습니다.

- 대기 시간을 줄이고 핵심 플랫폼 모니터링 구성 요소의 부하를 최소화하기 위해 **openshift.io/prometheus-rule-evaluation-scope: leaf-prometheus** 레이블을 규칙에 추가할 수 있습니다. 이 레이블은 **openshift-user-workload-monitoring** 프로젝트에 배포된 **Prometheus** 인스턴스만 경고 규칙을 평가하고 **Thanos Ruler** 인스턴스가 이를 수행하지 못하도록 합니다.

중요

경고 규칙에 이 레이블이 있는 경우 경고 규칙은 사용자 정의 프로젝트에서 노출하는 메트릭만 사용할 수 있습니다. 기본 플랫폼 메트릭을 기반으로 생성하는 경고 규칙은 경고가 트리거되지 않을 수 있습니다.

10.5.3. 사용자 정의 프로젝트에 대한 경고 규칙 생성

사용자 정의 프로젝트에 대한 경고 규칙을 생성할 수 있습니다. 이러한 경고 규칙은 선택한 메트릭의 값에 따라 경고를 트리거합니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 경고 규칙을 생성하려는 프로젝트에 대한 **monitoring-rules-edit** 클러스터 역할이 있는 사용자로 로그인했습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. 경고 규칙에 사용할 **YAML** 파일을 생성합니다. 이 예에서는 **example-app-alerting-rule.yaml**이라고 합니다.
2. **YAML** 파일에 경고 규칙 구성을 추가합니다. 예를 들면 다음과 같습니다.

참고

경고 규칙을 생성할 때 동일한 이름의 규칙이 다른 프로젝트에 존재하는 경우 프로젝트 라벨이 적용됩니다.

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

이 구성에서는 **example-alert**라는 경고 규칙이 생성됩니다. 경고 규칙은 샘플 서비스에서 노출된 **version** 메트릭이 **0**이 되면 경고를 실행합니다.

3. 구성 파일을 리클러스터에 적용합니다.

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

- **OpenShift Container Platform 4.11** 모니터링 아키텍처에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.

10.5.4. 사용자 정의 프로젝트의 경고 규칙에 액세스

사용자 정의 프로젝트에 대한 경고 규칙을 나열하려면 프로젝트의 **monitoring-rules-view** 클러스터 역할이 할당되어야 합니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 프로젝트에 대한 **monitoring-rules-view** 클러스터 역할이 있는 사용자로 로그인했습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. **<project>**에서 경고 규칙을 나열할 수 있습니다.

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

2. 경고 규칙의 구성을 나열하려면 다음을 실행합니다.

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

10.5.5. 단일 보기에서 모든 프로젝트의 경고 규칙 나열

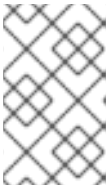
클러스터 관리자는 핵심 **OpenShift Container Platform** 및 사용자 정의 프로젝트에 대한 경고 규칙을 단일 보기에서 나열할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

절차

1. 관리자 관점에서 **Observe** → **Alerting** → **Alerting Rules** 로 이동합니다.
2. 필터 드롭다운 메뉴에서 플랫폼 및 사용자 소스를 선택합니다.



참고

플랫폼 소스가 기본적으로 선택됩니다.

10.5.6. 사용자 정의 프로젝트에 대한 경고 규칙 제거

사용자 정의 프로젝트에 대한 경고 규칙을 제거할 수 있습니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

- 경고 규칙을 생성하려는 프로젝트에 대한 **monitoring-rules-edit** 클러스터 역할이 있는 사용자로 로그인했습니다.

- **OpenShift CLI(oc)**가 설치되어 있습니다.

절차

- 규칙 **<foo>**를 **<namespace>**에서 제거하려면 다음을 실행합니다.

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

추가 리소스

- Alertmanager 문서 참조

10.6. 코어 플랫폼 모니터링을 위한 경고 규칙 관리

중요

핵심 플랫폼 모니터링에 대한 경고 규칙 생성 및 수정은 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

OpenShift Container Platform 4.11 모니터링에는 플랫폼 메트릭에 대한 많은 기본 경고 규칙이 포함되어 있습니다. 클러스터 관리자는 다음 두 가지 방법으로 이 규칙 세트를 사용자 지정할 수 있습니다.

- 임계값을 조정하거나 레이블을 추가 및 수정하여 기존 플랫폼 경고 규칙의 설정을 수정합니다. 예를 들어 경고에 대한 심각도 레이블을 경고 에서 심각으로 변경하여 경고로 플래그가 지정된 문제를 라우팅하고 분류하는 데 도움이 될 수 있습니다.

- **openshift-monitoring** 네임스페이스의 코어 플랫폼 지표를 기반으로 쿼리 표현식을 구성하여 새 사용자 정의 경고 규칙을 정의하고 추가합니다.

코어 플랫폼 경고 규칙 고려 사항

- 새 경고 규칙은 기본 **OpenShift Container Platform** 모니터링 메트릭을 기반으로 해야 합니다.
- 경고 규칙만 추가하고 수정할 수 있습니다. 새 레코딩 규칙을 생성하거나 기존 레코딩 규칙을 수정할 수 없습니다.
- **AlertRelabelConfig** 오브젝트를 사용하여 기존 플랫폼 경고 규칙을 수정하는 경우 수정 사항이 **Prometheus** 경고 **API**에 반영되지 않습니다. 따라서 더 이상 **Alertmanager**로 전달되지 않더라도 삭제된 경고가 **OpenShift Container Platform** 웹 콘솔에 계속 표시됩니다. 또한 변경된 심각도 라벨과 같은 경고 수정은 웹 콘솔에 표시되지 않습니다.

10.6.1. 코어 플랫폼 경고 규칙 수정

클러스터 관리자는 **Alertmanager**가 수신자에게 라우팅하기 전에 코어 플랫폼 경고를 수정할 수 있습니다. 예를 들어 경고의 심각도 레이블을 변경하거나, 사용자 정의 레이블을 추가하거나, 경고를 **Alertmanager**로 보내지 않도록 제외할 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 기술 프리뷰 기능을 활성화했으며 클러스터의 모든 노드가 준비되어 있습니다.

절차

1. **openshift-monitoring** 네임스페이스에 **example-modified-alerting-rule.yaml** 이라는 새 **YAML** 구성 파일을 생성합니다.
2. **AlertRelabelConfig** 리소스를 **YAML** 파일에 추가합니다. 다음 예제에서는 기본 플랫폼 워치독 경고 규칙에 대해 심각도 설정을 중요 로 수정합니다.

```
apiVersion: monitoring.openshift.io/v1alpha1
kind: AlertRelabelConfig
metadata:
  name: watchdog
  namespace: openshift-monitoring
spec:
  configs:
  - sourceLabels: [alertname,severity] (1)
    regex: "Watchdog;none" (2)
    targetLabel: severity (3)
    replacement: critical (4)
    action: Replace (5)
```

(1) 수정할 값의 소스 레이블입니다.

(2) **sourceLabels** 의 값이 일치하는 정규식입니다.

(3) 수정할 값의 **target** 레이블입니다.

(4) **target** 라벨을 대체할 새 값입니다.

(5) **regex** 일치에 따라 이전 값을 대체하는 재레이블 작업입니다. 기본 설정은 **Replace** 입니다. 기타 가능한 값은 **Keep, Drop ,Drop,HashMod,LabelMap,LabelDrop** 및 **LabelKeep.**

3. 구성 파일을 리클러스터에 적용합니다.

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

10.6.2. 새 경고 규칙 생성

클러스터 관리자는 플랫폼 메트릭을 기반으로 새 경고 규칙을 생성할 수 있습니다. 이러한 경고 규칙은 선택한 메트릭 값을 기반으로 경고를 트리거합니다.

참고

기존 플랫폼 경고 규칙을 기반으로 사용자 정의 **AlertingRule** 리소스를 생성하는 경우 원래 경고를 음소거하여 충돌하는 경고를 받지 않도록 합니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 기술 프리뷰 기능을 활성화했으며 클러스터의 모든 노드가 준비되어 있습니다.

프로세스

1. **openshift-monitoring** 네임스페이스에 **example-alerting-rule.yaml** 이라는 새 **YAML** 구성 파일을 만듭니다.
2. **AlertingRule** 리소스를 **YAML** 파일에 추가합니다. 다음 예제에서는 기본 **watchdog** 경고와 유사하게 **example** 이라는 새 경고 규칙을 생성합니다.

```
apiVersion: monitoring.openshift.io/v1alpha1
kind: AlertingRule
metadata:
  name: example
  namespace: openshift-monitoring
spec:
  groups:
  - name: example-rules
    rules:
    - alert: ExampleAlert 1
      expr: vector(1) 2
```

1. 생성할 경고 규칙의 이름입니다.
2. 새 규칙을 정의하는 **PromQL** 쿼리 식입니다.

3. 구성 파일을 리클러스터에 적용합니다.

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

추가 리소스

- OpenShift Container Platform 4.11 모니터링 아키텍처에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.
- 경고 규칙에 대한 정보는 Alertmanager 설명서를 참조하십시오.
- 레이블을 다시 지정하는 방법에 대한 정보는 Prometheus 레이블 재지정 문서 를 참조하십시오.
- 경고 최적화에 대한 추가 지침은 Prometheus 경고 문서 를 참조하십시오.

10.7. 외부 시스템에 알림 전송

OpenShift Container Platform 4.11에서는 알림 UI에서 실행 경고를 볼 수 있습니다. 알림은 기본적으로 모든 알림 시스템으로 전송되지 않습니다. 다음 수신자 유형으로 알림을 전송하도록 OpenShift Container Platform을 구성할 수 있습니다.

- PagerDuty
- Webhook
- 이메일
- Slack

알림을 수신기로 라우팅하면 오류가 발생할 때 적절한 팀에게 적절한 알림을 보낼 수 있습니다. 예를 들어, 심각한 경고는 즉각적인 주의가 필요하며 일반적으로 개인 또는 문제 대응팀으로 호출됩니다. 심각하

지 않은 경고 알림을 제공하는 경고는 즉각적이지 않은 검토를 위해 티켓팅 시스템으로 라우팅할 수 있습니다.

워치독 경고를 사용하여 해당 경고가 제대로 작동하는지 확인

OpenShift Container Platform 모니터링에는 지속적으로 트리거되는 워치독 경고가 포함되어 있습니다. Alertmanager는 구성된 알림 공급자에게 워치독 경고 알림을 반복적으로 보냅니다. 일반적으로 공급자는 워치독 경고를 수신하지 않을 때 관리자에게 알리도록 구성됩니다. 이 메커니즘을 사용하면 Alertmanager와 알림 공급자 간의 모든 통신 문제를 빠르게 식별할 수 있습니다.

10.7.1. 경고 수신자 구성

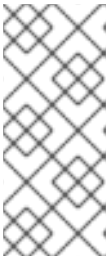
클러스터의 중요한 문제를 파악할 수 있도록 경고 수신자를 설정할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 관리자 관점에서 관리 → 클러스터 설정 → 구성 → Alertmanager 로 이동합니다.



참고

또는 알림 창을 통해 동일한 페이지로 이동할 수 있습니다. OpenShift Container Platform 웹 콘솔의 오른쪽 상단에서 호출 아이콘을 선택하고 AlertmanagerReceiverNotConfigured 경고에서 구성을 선택합니다.

2. 이 페이지의 수신자 섹션에서 수신자 만들기를 선택합니다.

3. 수신자 만들기에서 수신자 이름을 추가하고 목록에서 수신자 유형을 선택합니다.

4. 수신자 구성을 편집합니다.

- PagerDuty 수신자의 경우:

a. 통합 유형을 선택하고 **PagerDuty** 통합 키를 추가합니다.

b. **PagerDuty** 설치의 **URL**을 추가합니다.

c. 클라이언트와 인스턴스 세부 정보 또는 심각도 사양을 편집하려면 고급 설정 표시를 선택합니다.

- **Webhook** 수신자의 경우:

a. **HTTP POST** 요청이 전송되는 끝점을 추가합니다.

b. 해결된 경보를 수신자에게 보내는 기본 옵션을 편집하려면 고급 설정 표시를 선택합니다.

- 이메일 수신자의 경우:

a. 알림을 받을 이메일 주소를 추가합니다.

b. 알림을 전송할 주소, 이메일 전송에 사용되는 스마트 호스트 및 포트 번호, **SMTP** 서버의 호스트 이름, 인증 세부 정보를 포함하여 **SMTP** 구성 세부 정보를 추가합니다.

c. **TLS**가 필요한지 여부를 선택합니다.

d. 해결된 경고를 수신자에게 보내지 않도록 기본 옵션을 편집하거나 이메일 알림 구성을 편집하려면 고급 설정 표시를 선택합니다.

- **Slack** 수신자의 경우:

a. **Slack Webhook**의 **URL**을 추가합니다.

b. 알림을 보낼 **Slack** 채널 또는 사용자 이름을 추가합니다.

c. 해결된 경고를 수신자에게 보내지 않도록 기본 옵션을 편집하거나 아이콘 및 사용자 이름 구성을 편집하려면 고급 설정 표시를 선택합니다. 채널 이름과 사용자 이름을 찾고 연결할지 여부를 선택할 수도 있습니다.

5. 기본적으로 모든 선택 항목과 일치하는 라벨을 사용하여 경고를 수신자에게 보냅니다. 수신자로 전송되기 전에 실행 경고에 대한 라벨 값을 정확히 일치시키려면 다음을 수행하십시오.

a. 양식의 라우팅 라벨 섹션에 라우팅 라벨 이름과 값을 추가합니다.

b. 정규식을 사용하려면 정규식을 선택합니다.

c. 라벨 추가를 선택하여 추가 라우팅 라벨을 추가합니다.

6. 만들기를 선택하여 수신자를 생성합니다.

10.7.2. 사용자 정의 프로젝트에 대한 경고 라우팅 생성

alert-routing-edit 클러스터 역할이 지정된 관리자가 아닌 사용자인 경우 사용자 정의 프로젝트에 대한 경고 라우팅을 생성하거나 편집할 수 있습니다.

사전 요구 사항

- 클러스터 관리자가 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 클러스터 관리자가 사용자 정의 프로젝트에 대한 경고 라우팅을 활성화했습니다.
- 경고 라우팅을 생성하려는 프로젝트에 대한 **alert-routing-edit** 클러스터 역할이 있는 사용자로 로그인했습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

절차

1. 경고 라우팅을 위한 YAML 파일을 만듭니다. 이 절차의 예제에서는 example-app-alert-routing.yaml 이라는 파일을 사용합니다.

2. AlertmanagerConfig YAML 정의를 파일에 추가합니다. 예를 들어 다음과 같습니다.

```
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
```

참고

사용자 정의 경고 규칙의 경우 사용자 정의 라우팅은 리소스가 정의된 네임스페이스로 범위가 지정됩니다. 예를 들어 네임스페이스 ns1 의 AlertmanagerConfig 오브젝트에 정의된 라우팅 구성은 동일한 네임스페이스의 PrometheusRules 리소스에만 적용됩니다.

3. 파일을 저장합니다.

4. 클러스터에 리소스를 적용합니다.

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

구성은 Alertmanager Pod에 자동으로 적용됩니다.

10.8. 사용자 정의 ALERTMANAGER 설정 적용

Alertmanager의 플랫폼 인스턴스에 대한 openshift-monitoring 네임스페이스에서 alertmanager-main 시크릿을 편집하여 기본 Alertmanager 설정을 덮어쓸 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

프로세스

CLI에서 Alertmanager 설정을 변경하려면 다음을 수행합니다.

1. 현재 활성화된 Alertmanager 구성을 파일 alertmanager.yaml로 출력합니다.

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

2. alertmanager.yaml에서 설정을 편집합니다.

```
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
  - matchers:
    - "service=<your_service>" (4)
    routes:
    - matchers:
      - <your_matching_rules> (5)
      receiver: <receiver> (6)
receivers:
- name: default
- name: watchdog
- name: <receiver>
#  <receiver_configuration>
```

(1) group_wait 값은 경고 그룹에 대한 초기 알림을 보내기 전에 Alertmanager 대기 기간을 지정합니다. 이 값은 알림을 보내기 전에 동일한 그룹에 대한 초기 경고를 수집하는 동안 Alertmanager가 대기하는 시간을 제어합니다.

(2)

3

repeat_interval 값은 경고 알림을 반복하기 전에 전달해야 하는 최소 시간을 지정합니다. 각 그룹 간격마다 알림을 반복하려면 repeat_interval 값을 group_interval 값보다 적도록 설정합니다. 그러나 반복된 알림은 특정 Alertmanager Pod가 재시작되거나 다시 예약되는 경우와 같이 계속 지연될 수 있습니다.

4

서비스 값은 경고를 실행하는 서비스를 지정합니다.

5

& lt;your_matching_rules& gt; 값은 대상 경고를 지정합니다.

6

수신자 값은 경고에 사용할 수신자를 지정합니다.

참고

일치자 키 이름을 사용하여 노드와 일치하도록 경고가 충족해야 하는 일치 항목을 나타냅니다. 더 이상 사용되지 않고 향후 릴리스에서 제거될 예정인 match 또는 match_re 키 이름을 사용하지 마십시오.

또한 억제 규칙을 정의하는 경우 target_matchers 키 이름을 사용하여 대상 일치 항목과 source_matchers 키 이름을 지정하여 소스 일치 항목을 나타냅니다. target_match , target_match _re,source_match 또는 source_match_re 키 이름을 사용하지 마십시오. 이 이름은 더 이상 사용되지 않고 향후 릴리스에서 제거될 예정입니다.

다음 Alertmanager 설정 예제에서는 PagerDuty를 경고 수신자로 구성합니다.

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - matchers:
```

```
    - "alertname=Watchdog"
    repeat_interval: 2m
    receiver: watchdog
  - matchers:
    - "service=example-app"
    routes:
    - matchers:
      - "severity=critical"
      receiver: team-frontend-page*
receivers:
- name: default
- name: watchdog
- name: team-frontend-page
  pagerduty_configs:
  - service_key: "_your-key_"
```

이 설정을 사용하면 **example-app** 서비스에서 실행되는 **critical** 심각도 경고가 **team-frontend-page** 수신자를 사용하여 전송됩니다. 일반적으로 이러한 유형의 경고는 개인 또는 문제 대응팀으로 호출됩니다.

3. 파일에 새 설정을 적용합니다.

```
$ 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=-
```

OpenShift Container Platform 웹 콘솔에서 **Alertmanager** 설정을 변경하려면 다음을 수행합니다.

1. 웹 콘솔의 관리 → 클러스터 설정 → 구성 → **Alertmanager** → **YAML** 페이지로 이동합니다.
2. **YAML** 설정 파일을 수정합니다.
3. 저장을 선택합니다.

10.9. 사용자 정의 경고 라우팅에 대한 ALERTMANAGER에 사용자 정의 설정 적용

사용자 정의 경고 라우팅 전용 **Alertmanager**의 별도의 인스턴스를 활성화한 경우 **openshift-user-workload-monitoring** 네임스페이스에서 **alertmanager-user-workload** 보안을 편집하여 **Alertmanager** 인스턴스에 대한 구성을 덮어쓸 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 현재 활성화된 Alertmanager 설정을 파일 alertmanager.yaml 로 출력합니다.

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

2. alertmanager.yaml에서 설정을 편집합니다.

```
route:
  receiver: Default
  group_by:
  - name: Default
  routes:
  - matchers:
    - "service = prometheus-example-monitor" 1
    receiver: <receiver> 2
receivers:
- name: Default
- name: <receiver>
#  <receiver_configuration>
```

1 경로에 일치하는 경고를 지정합니다. 이 예에서는 service="prometheus-example-monitor" 레이블이 있는 모든 경고를 보여줍니다.

2 경고 그룹에 사용할 수신자를 지정합니다.

3. 파일에 새 설정을 적용합니다.

```
$ 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=-
```

추가 리소스

- PagerDuty에 대한 자세한 내용은 PagerDuty 공식 사이트를 참조하십시오.
- service_key 검색 방법을 알아보려면 PagerDuty Prometheus 통합 가이드 를 참조하십시오.
- 다양한 경고 수신자를 통한 경고 구성은 Alertmanager 설정을 참조하십시오.
- 사용자 정의 프로젝트의 경고 라우팅 활성화에서 사용자 정의 경고 라우팅 전용 Alertmanager 인스턴스를 활성화하는 방법을 참조하십시오.

10.10. 다음 단계

- 모니터링 대시보드 검토

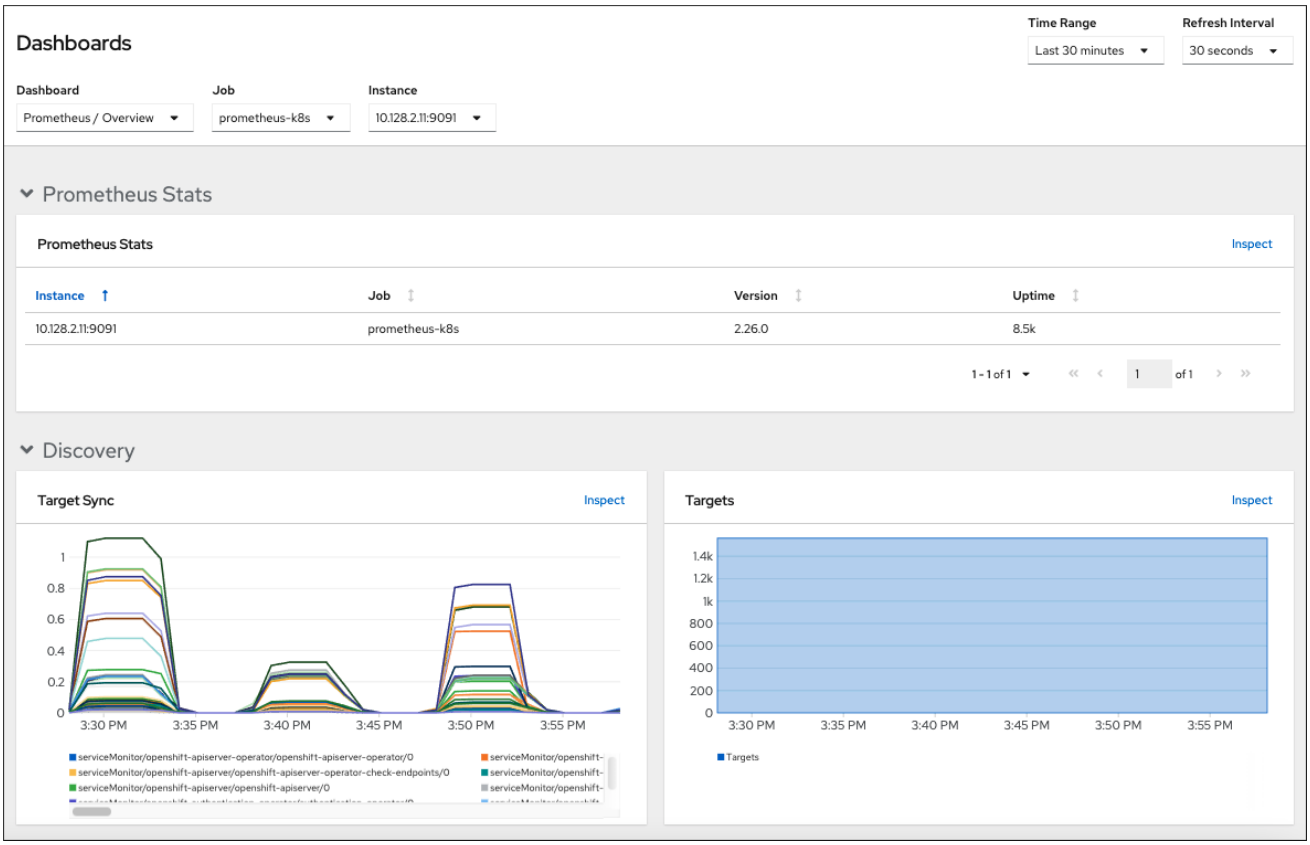
11장. 모니터링 대시보드 검토

OpenShift Container Platform 4.11은 클러스터 구성 요소 및 사용자 정의 워크로드의 상태를 이해하는 데 도움이 되는 포괄적인 모니터링 대시보드 세트를 제공합니다.

다음 항목을 포함하여 핵심 OpenShift Container Platform 구성 요소의 대시보드에 액세스하려면 관리자 관점을 사용합니다.

- API 성능
- etcd
- Kubernetes 컴퓨팅 리소스
- Kubernetes 네트워크 리소스
- Prometheus
- 클러스터 및 노드 성능과 관련된 USE 메서드 대시보드

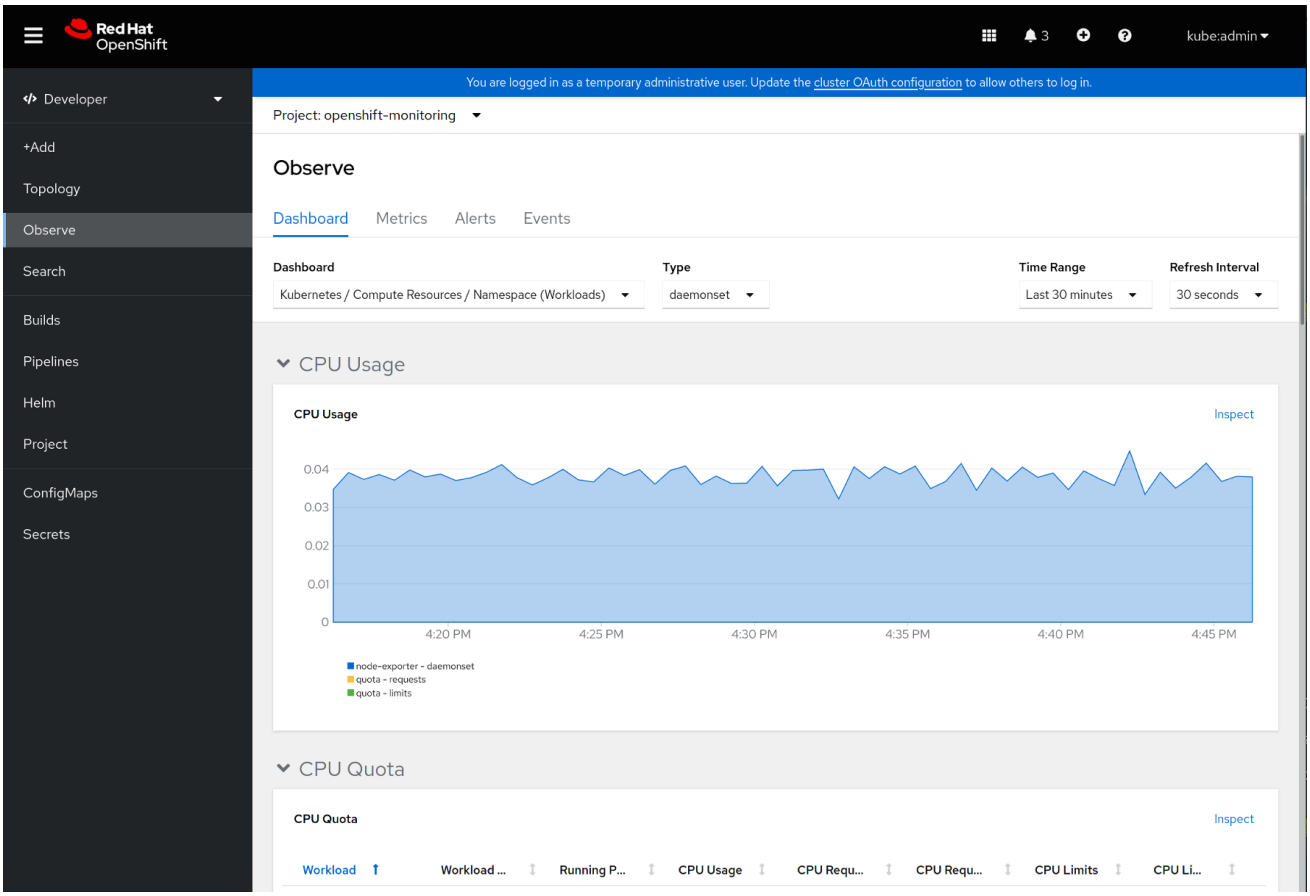
그림 11.1. 관리자 관점의 대시보드 예



개발자 화면을 사용하여 선택한 프로젝트에 대해 다음 애플리케이션 지표를 제공하는 **Kubernetes** 컴퓨팅 리소스 대시보드에 액세스합니다.

- **CPU** 사용량
- 메모리 사용량
- 대역폭 정보
- 패킷 속도 정보

그림 11.2. 개발자 관점의 대시보드 예



참고

개발자 관점에서 한 번에 하나의 프로젝트에 대해서만 대시보드를 볼 수 있습니다.

11.1. 클러스터 관리자로 모니터링 대시보드 검토

관리자 관점에서 핵심 **OpenShift Container Platform** 클러스터 구성 요소와 관련된 대시보드를 볼 수 있습니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. **OpenShift Container Platform** 웹 콘솔의 관리자 화면에서 모니터링 → 대시보드로 이동합니다.

2.

대시보드 목록에서 대시보드를 선택합니다. **etcd** 및 **Prometheus** 대시보드와 같은 일부 대시보드는 선택하면 추가 하위 메뉴가 생성됩니다.

3. 선택 사항: 시간 범위 목록에서 그래프의 시간 범위를 선택합니다.

- 미리 정의된 기간을 선택합니다.
- 시간 범위 목록에서 사용자 지정 시간 범위를 선택하여 사용자 지정 시간 범위를 설정합니다.

a. 시작 및 종료 날짜 및 시간을 입력하거나 선택합니다.

b. 저장을 클릭하여 사용자 지정 시간 범위를 저장합니다.

4. 선택 사항: 새로 고침 간격을 선택합니다.

5. 대시보드 내에서 각각의 그래프로 마우스를 이동하여 특정 항목에 대한 세부 정보를 표시합니다.

11.2. 개발자로 모니터링 대시보드 검토

개발자 화면을 사용하여 선택한 프로젝트의 **Kubernetes** 컴퓨팅 리소스 대시보드를 봅니다.

사전 요구 사항

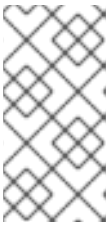
- 개발자 또는 사용자로 클러스터에 액세스할 수 있습니다.
- 대시보드를 보고 있는 프로젝트에 대한 보기 권한이 있습니다.

절차

1. **OpenShift Container Platform** 웹 콘솔의 개발자 관점에서 **Observe → Dashboard** 로 이동합니다.

2. 프로젝트: 드롭다운 목록에서 프로젝트를 선택합니다.

3. 대시보드 드롭다운 목록에서 대시보드 를 선택하여 필터링된 지표를 확인합니다.



참고

모든 대시보드는 **Kubernetes / Compute Resources / Namespace (Pods)** 를 제외하고 선택한 경우 추가 하위 메뉴를 생성합니다.

4. 선택 사항: 시간 범위 목록에서 그래프의 시간 범위를 선택합니다.

- 미리 정의된 기간을 선택합니다.
- 시간 범위 목록에서 사용자 지정 시간 범위를 선택하여 사용자 지정 시간 범위를 설정합니다.

a. 시작 및 종료 날짜 및 시간을 입력하거나 선택합니다.

b. 저장을 클릭하여 사용자 지정 시간 범위를 저장합니다.

5. 선택 사항: 새로 고침 간격을 선택합니다.

6. 대시보드 내에서 각각의 그래프로 마우스를 이동하여 특정 항목에 대한 세부 정보를 표시합니다.

추가 리소스

- 개발자 관점을 사용하여 프로젝트 및 애플리케이션 메트릭 모니터링

11.3. 다음 단계

- 타사 모니터링 API에 액세스

12장. NVIDIA GPU 관리 대시보드

12.1. 소개

OpenShift Console NVIDIA GPU 플러그인은 OCP(OpenShift Container Platform) 콘솔에서 NVIDIA GPU 사용 시각화를 위한 전용 관리 대시보드입니다. 관리 대시보드의 시각화는 GPU가 부족하거나 과도하게 사용되는 경우와 같이 클러스터에서 GPU 리소스를 최적화하는 방법에 대한 지침을 제공합니다.

OpenShift Console NVIDIA GPU 플러그인은 OCP 콘솔의 원격 번들로 작동합니다. 플러그인을 실행하려면 OCP 콘솔이 실행 중이어야 합니다.

12.2. NVIDIA GPU 관리 대시보드 설치

OCP(OpenShift Container Platform) 콘솔에서 Helm을 사용하여 GPU 기능을 추가하여 NVIDIA GPU 플러그인을 설치합니다.

OpenShift Console NVIDIA GPU 플러그인은 OCP 콘솔의 원격 번들로 작동합니다. OpenShift Console NVIDIA GPU 플러그인을 실행하려면 OCP 콘솔 인스턴스가 실행 중이어야 합니다.

사전 요구 사항

- Red Hat OpenShift 4.11+
- NVIDIA GPU Operator
- Helm

절차

다음 절차에 따라 OpenShift Console NVIDIA GPU 플러그인을 설치합니다.

1. Helm 리포지터리를 추가합니다.

```
$ helm repo add rh-ecosystem-edge https://rh-ecosystem-edge.github.io/console-plugin-nvidia-gpu
```

```
$ helm repo update
```

2. 기본 **NVIDIA GPU Operator** 네임스페이스에 **Helm** 차트를 설치합니다.

```
$ helm install -n nvidia-gpu-operator console-plugin-nvidia-gpu rh-ecosystem-
edge/console-plugin-nvidia-gpu
```

출력 예

```
NAME: console-plugin-nvidia-gpu
LAST DEPLOYED: Tue Aug 23 15:37:35 2022
NAMESPACE: nvidia-gpu-operator
STATUS: deployed
REVISION: 1
NOTES:
View the Console Plugin NVIDIA GPU deployed resources by running the following
command:

$ oc -n {{ .Release.Namespace }} get all -l app.kubernetes.io/name=console-plugin-
nvidia-gpu

Enable the plugin by running the following command:

# Check if a plugins field is specified
$ oc get consoles.operator.openshift.io cluster --output=jsonpath="{.spec.plugins}"

# if not, then run the following command to enable the plugin
$ oc patch consoles.operator.openshift.io cluster --patch '{ "spec": { "plugins":
["console-plugin-nvidia-gpu"] } }' --type=merge

# if yes, then run the following command to enable the plugin
$ oc patch consoles.operator.openshift.io cluster --patch '[{"op": "add", "path":
"/spec/plugins/-", "value": "console-plugin-nvidia-gpu" }]' --type=json

# add the required DCGM Exporter metrics ConfigMap to the existing NVIDIA operator
ClusterPolicy CR:
oc patch clusterpolicies.nvidia.com gpu-cluster-policy --patch '{ "spec": {
"dcgmExporter": { "config": { "name": "console-plugin-nvidia-gpu" } } } }' --type=merge
```

대시보드는 **NVIDIA DCGM Exporter**에서 노출하는 **Prometheus** 지표에 주로 사용하지만 기본 노출된 메트릭으로 인해 대시보드가 필요한 지표를 렌더링하는 데 충분하지 않습니다. 따라서 **DGCM** 내보내기는 다음과 같이 사용자 지정 메트릭 세트를 노출하도록 구성됩니다.

```
apiVersion: v1
```

```
data:
  dcgm-metrics.csv: |
    DCGM_FI_PROF_GR_ENGINE_ACTIVE, gauge, gpu utilization.
    DCGM_FI_DEV_MEM_COPY_UTIL, gauge, mem utilization.
    DCGM_FI_DEV_ENC_UTIL, gauge, enc utilization.
    DCGM_FI_DEV_DEC_UTIL, gauge, dec utilization.
    DCGM_FI_DEV_POWER_USAGE, gauge, power usage.
    DCGM_FI_DEV_POWER_MGMT_LIMIT_MAX, gauge, power mgmt limit.
    DCGM_FI_DEV_GPU_TEMP, gauge, gpu temp.
    DCGM_FI_DEV_SM_CLOCK, gauge, sm clock.
    DCGM_FI_DEV_MAX_SM_CLOCK, gauge, max sm clock.
    DCGM_FI_DEV_MEM_CLOCK, gauge, mem clock.
    DCGM_FI_DEV_MAX_MEM_CLOCK, gauge, max mem clock.
kind: ConfigMap
metadata:
  annotations:
    meta.helm.sh/release-name: console-plugin-nvidia-gpu
    meta.helm.sh/release-namespace: nvidia-gpu-operator
  creationTimestamp: "2022-10-26T19:46:41Z"
  labels:
    app.kubernetes.io/component: console-plugin-nvidia-gpu
    app.kubernetes.io/instance: console-plugin-nvidia-gpu
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: console-plugin-nvidia-gpu
    app.kubernetes.io/part-of: console-plugin-nvidia-gpu
    app.kubernetes.io/version: latest
    helm.sh/chart: console-plugin-nvidia-gpu-0.2.3
  name: console-plugin-nvidia-gpu
  namespace: nvidia-gpu-operator
  resourceVersion: "19096623"
  uid: 96cdf700-dd27-437b-897d-5cbb1c255068
```

ConfigMap을 설치하고 NVIDIA Operator ClusterPolicy CR을 편집하여 DCGM 내보내기 구성에 ConfigMap을 추가합니다. ConfigMap 설치는 새 버전의 Console Plugin NVIDIA GPU Helm 차트에서 수행하지만 ClusterPolicy CR 편집은 사용자가 수행합니다.

3. 배포된 리소스를 확인합니다.

```
$ oc -n nvidia-gpu-operator get all -l app.kubernetes.io/name=console-plugin-nvidia-gpu
```

출력 예

```
NAME                                           READY   STATUS    RESTARTS   AGE
pod/console-plugin-nvidia-gpu-7dc9cfb5df-ztksx   1/1     Running   0          2m6s

NAME                                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/console-plugin-nvidia-gpu   ClusterIP   172.30.240.138   <none>        9443/TCP   2m6s
```

```
NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/console-plugin-nvidia-gpu   1/1     1            1           2m6s

NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/console-plugin-nvidia-gpu-7dc9cfb5df   1         1         1       2m6s
```

12.3. NVIDIA GPU 관리 대시보드 사용

OpenShift Console NVIDIA GPU 플러그인을 배포한 후 로그인 인증 정보를 사용하여 OpenShift Container Platform 웹 콘솔에 로그인하여 관리자 화면에 액세스합니다.

변경 사항을 보려면 콘솔을 새로 고쳐 Compute 아래에 GPU 탭을 확인해야 합니다.

12.3.1. 클러스터 GPU 개요 보기

홈 섹션에서 개요를 선택하여 개요 페이지에서 클러스터 GPU의 상태를 볼 수 있습니다.

개요 페이지에서는 다음을 포함하여 클러스터 GPU에 대한 정보를 제공합니다.

- GPU 공급자에 대한 세부 정보
- GPU 상태
- GPU의 클러스터 사용률

12.3.2. GPU 대시보드 보기

OpenShift 콘솔의 Compute 섹션에서 GPU를 선택하여 NVIDIA GPU 관리 대시보드를 볼 수 있습니다.

GPU 대시보드의 차트는 다음과 같습니다.

- GPU 사용률: 그래픽 엔진이 활성화된 시간의 비율을 표시하고 DCGM_FI_PROF_GR_ENGINE_ACTIVE 메트릭을 기반으로 합니다.

- 메모리 사용률: GPU에서 사용하는 메모리를 표시하고 DCGM_FI_DEV_MEM_COPY_UTIL 메트릭을 기반으로 합니다.

- 인코더 사용률: 비디오 인코더 사용률을 표시하며 DCGM_FI_DEV_ENC_UTIL 메트릭을 기반으로 합니다.

- 디코더 사용:인코더 사용률: 사용 의 비디오 디코더 속도를 표시하고 DCGM_FI_DEV_DEC_UTIL 메트릭을 기반으로 합니다.

- 전력 소비: 와트에서 GPU의 평균 전원 사용량을 표시하고 DCGM_FI_DEV_POWER_USAGE 메트릭을 기반으로 합니다.

- GPU 온도: 현재 GPU 기온을 표시하고 DCGM_FI_DEV_GPU_TEMP 메트릭을 기반으로 합니다. 최대값은 실제 숫자가 메트릭을 통해 노출되지 않기 때문에 경험적 번호인 110 으로 설정됩니다.

- GPU 클럭 속도: GPU에서 사용하는 평균 클럭 속도를 표시하고 DCGM_FI_DEV_SM_CLOCK 메트릭을 기반으로 합니다.

- 메모리 클럭 속도: 메모리에 사용되는 평균 클럭 속도를 나타내며 DCGM_FI_DEV_MEM_CLOCK 메트릭을 기반으로 합니다.

12.3.3. GPU 지표 보기

각 GPU 하단에서 메트릭을 선택하여 지표 페이지를 볼 수 있습니다.

지표 페이지에서 다음을 수행할 수 있습니다.

- 메트릭에 대한 새로 고침 비율을 지정합니다.

- 쿼리 추가, 실행, 비활성화 및 삭제Add, run, disable, and delete queries
- 메트릭 삽입
- 확대/축소 보기 재설정

13장. 타사 모니터링 API에 액세스

OpenShift Container Platform 4.11에서는 CLI(명령줄 인터페이스)에서 일부 타사 모니터링 구성 요소의 웹 서비스 API에 액세스할 수 있습니다.

13.1. 타사 모니터링 웹 서비스 API에 액세스

Prometheus, Alertmanager, Thanos Ruler 및 Thanos Querier의 다음 모니터링 스택 구성 요소에 대해 명령줄에서 타사 웹 서비스 API에 직접 액세스할 수 있습니다.

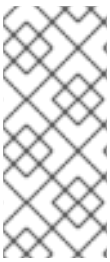
다음 예제 명령은 Alertmanager에 대한 서비스 API 수신자를 쿼리하는 방법을 보여줍니다. 이 예제에서는 관련 사용자 계정을 openshift-monitoring 네임스페이스의 monitoring-alertmanager-edit 역할에 바인딩해야 하며 계정에 경로를 볼 수 있는 권한이 있어야 합니다. 이 액세스는 인증을 위한 전달자 토큰만 지원합니다.

```
$ oc login -u <username> -p <password>
```

```
$ host=$(oc -n openshift-monitoring get route alertmanager-main -ojsonpath={.spec.host})
```

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

```
$ curl -H "Authorization: Bearer $token" -k "https://$host/api/v2/receivers"
```



참고

Thanos Ruler 및 Thanos Querier 서비스 API에 액세스하려면 요청하는 계정에 cluster-monitoring-view 클러스터 역할을 부여하여 네임스페이스 리소스에 대한 get 권한이 있어야 합니다.

13.2. PROMETHEUS의 페더레이션 끝점을 사용하여 메트릭 쿼리

페더레이션 끝점을 사용하여 클러스터 외부의 네트워크 위치에서 플랫폼 및 사용자 정의 메트릭을 스크랩할 수 있습니다. 이렇게 하려면 OpenShift Container Platform 경로를 통해 클러스터의 Prometheus /federate 끝점에 액세스합니다.

주의

지표 데이터 검색에 지연이 지연되면 페더레이션을 사용할 때 발생합니다. 이러한 지연은 스크랩된 메트릭의 정확도 및 타임라인에 영향을 미칠 수 있습니다.

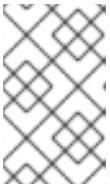
페더레이션 엔드포인트를 사용하면 특히 페더레이션 엔드포인트를 사용하여 대량의 지표 데이터를 검색하는 경우 클러스터의 성능과 확장성이 저하될 수 있습니다. 이러한 문제를 방지하려면 다음 권장 사항을 따르십시오.

- 페더레이션 엔드포인트를 통해 지표 데이터를 모두 검색하지 마십시오. 제한된 집계 데이터 세트를 검색하려는 경우에만 쿼리합니다. 예를 들어 각 요청에 대해 1,000개 미만의 샘플을 검색하면 성능 저하 위험을 최소화할 수 있습니다.
- 페더레이션 엔드 포인트를 자주 쿼리하지 마십시오. 쿼리를 30초마다 최대 1개로 제한합니다.

대량의 데이터를 클러스터 외부로 전달해야 하는 경우 대신 원격 쓰기를 사용합니다. 자세한 내용은 *원격 쓰기 스토리지 구성* 섹션을 참조하십시오.

사전 요구 사항

- **OpenShift CLI(oc)**가 설치되어 있습니다.
- **OpenShift Container Platform** 경로의 호스트 **URL**을 가져왔습니다.
- **cluster-monitoring-view** 클러스터 역할의 사용자로 클러스터에 액세스하거나 네임스페이스 리소스에 대한 **get** 권한을 사용하여 전달자 토큰을 가져올 수 있습니다.



참고

전달자 토큰 인증을 사용하여 연합 엔드포인트에만 액세스할 수 있습니다.

프로세스

1. 전달자 토큰을 검색합니다.

```
$ token=`oc whoami -t`
```

2. /federate 경로에서 지표를 쿼리합니다. 다음 예제에서는 메트릭 을 쿼리합니다.

```
$ curl -G -s -k -H "Authorization: Bearer $token" \
  'https:/<federation_host>/federate' \ 1
  --data-urlencode 'match[]=up'
```

1 <federation_host>는 페더레이션 경로의 호스트 URL을 대체합니다.

출력 예

```
# 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
...
```

13.3. 추가 리소스

- 원격 쓰기 스토리지 구성
- 메트릭 관리

- 경고 관리

14장. 모니터링 문제 조사

14.1. 사용자 정의 메트릭을 사용할 수 없는 이유 확인

ServiceMonitor 리소스를 사용하면 사용자 정의 프로젝트에서 서비스에 의해 노출되는 메트릭을 사용하는 방법을 확인할 수 있습니다. **ServiceMonitor** 리소스를 생성했지만 메트릭 **UI**에서 해당 메트릭을볼 수 없는 경우 이 프로세스에 설명된 단계를 수행하십시오.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.
- 사용자 정의 워크로드에 대한 모니터링을 활성화 및 구성하고 있어야 합니다.
- **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- **ServiceMonitor** 리소스가 생성되어 있습니다.

프로세스

1. 서비스 및 **ServiceMonitor** 리소스 구성에서 해당 라벨이 일치하는지 확인합니다.

   a. 서비스에 정의된 라벨을 가져옵니다. 다음 예제에서는 **ns1** 프로젝트의 **prometheus-example-app** 서비스를 쿼리합니다.

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

   출력 예

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

b.

ServiceMonitor 리소스의 **matchLabels app** 라벨이 이전 단계의 라벨 출력과 일치하는지 확인합니다.

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

출력 예

```
apiVersion: v1
kind: Service
# ...
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-example-app
# ...
```

참고

프로젝트 보기 권한이 있는 개발자로서 서비스 및 **ServiceMonitor** 리소스 라벨을 확인할 수 있습니다.

2.

openshift-user-workload-monitoring 프로젝트에서 **Prometheus Operator**의 로그를 검사합니다.

a.

openshift-user-workload-monitoring 프로젝트의 **Pod**를 나열합니다.

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

출력 예

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

b. prometheus-operator pod의 prometheus-operator 컨테이너에서 로그를 가져옵니다. 다음 예에서 Pod는 prometheus-operator-776fcbbd56-2nbfm입니다.

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

서비스 모니터에 문제가 있는 경우 로그에 다음과 유사한 오류가 포함될 수 있습니다.

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

3. OpenShift Container Platform 웹 콘솔 UI의 Metrics 대상 페이지에서 끝점의 대상 상태를 확인합니다.

a. OpenShift Container Platform 웹 콘솔에 로그인하고 관리자 관점에서 Observe → 대상으로 이동합니다.

b. 목록에서 지표 엔드포인트를 찾고 Status 열에서 대상의 상태를 검토합니다.

c. Status 가 Down 이면 끝점의 URL을 클릭하여 해당 메트릭 대상의 Target Details 페이지에서 자세한 정보를 확인합니다.

4. openshift-user-workload-monitoring 프로젝트에서 Prometheus Operator의 디버그 수준 로깅을 구성합니다.

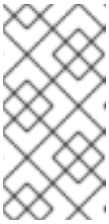
a. openshift-user-workload-monitoring 프로젝트에서 user-workload-monitoring-config ConfigMap 오브젝트를 편집합니다.

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

b. prometheusOperator의 logLevel:debug를 data / config.yaml 아래에 추가하여 로그 수준을 debug로 설정합니다.

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

c. 파일을 저장하여 변경 사항을 적용합니다.



참고

openshift-user-workload-monitoring 프로젝트의 prometheus-operator는 로그 수준 변경을 적용하면 자동으로 다시 시작됩니다.

d. openshift-user-workload-monitoring 프로젝트의 prometheus-operator 배포에 debug 로그 수준이 적용되었는지 확인합니다.

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

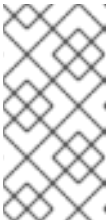
출력 예

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

디버그 수준 로깅은 Prometheus Operator가 수행한 모든 호출을 표시합니다.

e. prometheus-operator Pod가 실행되고 있는지 확인합니다.

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



참고

구성 맵에 인식할 수 없는 Prometheus Operator loglevel 값이 포함된 경우 prometheus-operator Pod가 재시작되지 않을 수 있습니다.

f. 디버그 로그를 검토하여 Prometheus Operator에서 ServiceMonitor 리소스를 사용하고 있는지 확인합니다. 기타 관련 오류에 대한 로그를 확인합니다.

추가 리소스

- 사용자 정의 워크로드 모니터링 구성 맵 생성
- ServiceMonitor 또는 PodMonitor 리소스를 만드는 방법에 대한 자세한 내용은 서비스 모니터링 방법 지정 에서 참조하십시오.
- 관리자 관점에서 메트릭 대상 액세스를참조하십시오.

14.2. PROMETHEUS가 많은 디스크 공간을 소비하는 이유 확인

개발자는 라벨을 생성하여 키-값 쌍의 형식으로 메트릭의 속성을 정의할 수 있습니다. 잠재적인 키-값 쌍의 수는 속성에 사용 가능한 값의 수에 해당합니다. 무제한의 잠재적인 값이 있는 속성을 바인딩되지 않은 속성이라고 합니다. 예를 들어, customer_id 속성은 무제한 가능한 값이 있기 때문에 바인딩되지 않은 속성입니다.

할당된 모든 키-값 쌍에는 고유한 시계열이 있습니다. 라벨에 있는 바인딩되지 않은 많은 속성을 사용하면 생성되는 시계열 수가 기하급수적으로 증가할 수 있습니다. 이는 Prometheus 성능에 영향을 미칠 수 있으며 많은 디스크 공간을 소비할 수 있습니다.

Prometheus가 많은 디스크를 사용하는 경우 다음 조치를 사용할 수 있습니다.

- 수집 중인 스크랩 샘플 수를 확인합니다.

- 가장 많은 시계열을 생성하는 라벨에 대한 자세한 내용은 **Prometheus HTTP API**를 사용하여 시계열 데이터베이스**(TSDB)** 상태를 확인합니다. 이렇게 하려면 클러스터 관리자 권한이 필요합니다.

- 사용자 정의 메트릭에 할당되는 바인딩되지 않은 속성의 수를 줄임으로써 생성되는 고유의 시계열 수를 감소합니다.

참고

사용 가능한 값의 제한된 집합에 바인딩되는 속성을 사용하면 가능한 키 **-** 값 쌍 조합의 수가 줄어듭니다.

- 사용자 정의 프로젝트에서 스크랩할 수 있는 샘플 수를 제한합니다. 여기에는 클러스터 관리자 권한이 필요합니다.

사전 요구 사항

- **cluster-admin** 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.

- **OpenShift CLI(oc)**가 설치되어 있습니다.

절차

1. 관리자 관점에서 **Observe → Metrics** 로 이동합니다.

2. **Expression** 필드에서 다음 **Prometheus Query Language (PromQL)** 쿼리를 실행합니다. 이렇게 하면 스크랩 샘플 수가 가장 많은 **10**개의 메트릭이 반환됩니다.

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

3. 예상 스크랩 샘플 수 보다 많은 메트릭에 할당된 바인딩되지 않은 라벨 값의 수를 조사합니다.

-

메트릭이 사용자 정의 프로젝트와 관련된 경우 워크로드에 할당된 메트릭의 키-값 쌍을 확인합니다. 이는 애플리케이션 수준에서 Prometheus 클라이언트 라이브러리를 통해 구현됩니다. 라벨에서 참조되는 바인딩되지 않은 속성의 수를 제한하십시오.

- 메트릭이 OpenShift Container Platform의 주요 프로젝트와 관련된 경우 Red Hat Customer Portal에서 Red Hat 지원 케이스를 생성하십시오.

4. 클러스터 관리자로 다음 명령을 실행하여 Prometheus HTTP API를 사용하여 TSDB 상태를 확인합니다.

```
$ oc login -u <username> -p <password>
```

```
$ host=$(oc -n openshift-monitoring get route prometheus-k8s -ojsonpath=
{.spec.host})
```

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

```
$ curl -H "Authorization: Bearer $token" -k "https://$host/api/v1/status/tsdb"
```

출력 예

```
"status": "success",
```

추가 리소스

- 스크랩 샘플 제한을 설정하고 관련 경고 규칙을 생성하는 방법에 대한 자세한 내용은 사용자 정의 프로젝트의 스크랩 샘플 제한 설정을 참조하십시오.

- 지원 케이스 제출

15장. CLUSTER MONITORING OPERATOR에 대한 구성 맵 참조

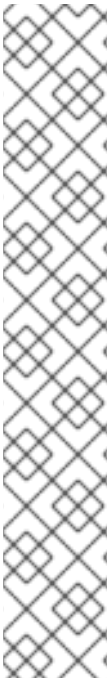
15.1. CLUSTER MONITORING OPERATOR 구성 참조

OpenShift Container Platform 클러스터 모니터링의 일부는 구성할 수 있습니다. **API**는 다양한 구성 맵에 정의된 매개변수를 설정하여 액세스할 수 있습니다.

- 모니터링 구성 요소를 구성하려면 **openshift-monitoring** 네임스페이스에서 **cluster-monitoring-config** 라는 **ConfigMap** 오브젝트를 편집합니다. 이러한 구성은 **ClusterMonitoringConfiguration** 에 의해 정의됩니다.

- 사용자 정의 프로젝트를 모니터링하는 모니터링 구성 요소를 구성하려면 **openshift-user-workload-monitoring** 네임스페이스에서 **user-workload-monitoring-config** 라는 **ConfigMap** 오브젝트를 편집합니다. 이러한 구성은 **UserWorkload Configuration**에서 정의합니다.

구성 파일은 항상 구성 맵 데이터의 **config.yaml** 키에 정의되어 있습니다.



참고

- 일부 구성 매개변수가 노출되지는 않습니다.
- 클러스터 모니터링 구성은 선택 사항입니다.
- 구성이 없거나 비어 있는 경우 기본값이 사용됩니다.
- 구성이 유효하지 않은 **YAML** 데이터인 경우 **Cluster Monitoring Operator**는 리소스 재결합을 중지하고 **Operator**의 상태 조건에서 **Degraded=True** 를 보고합니다.

15.2. ADDITIONALALERTMANAGERCONFIG

15.2.1. 설명

additional AlertmanagerConfig 리소스는 구성 요소가 추가 **Alertmanager** 인스턴스와 통신하는 방법에 대한 설정을 정의합니다.

15.2.2. 필수 항목

- **apiVersion**

PrometheusK8sConfig,PrometheusRestrictedConfig,ThanosRulerConfig

속성	유형	설명
apiVersion	string	Alertmanager의 API 버전을 정의합니다. 가능한 값은 **v1** 또는 **v2** 입니다. 기본값은 **v2** 입니다.
bearerToken	*v1.SecretKeySelector	Alertmanager에 인증할 때 사용할 전달자 토큰이 포함된 보안 키 참조를 정의합니다.
pathPrefix	string	내보내기 끝점 경로 앞에 추가할 경로 접두사를 정의합니다.
스키마	string	Alertmanager 인스턴스와 통신할 때 사용할 URL 스키마를 정의합니다. 가능한 값은 **http** 또는 **https** 입니다. 기본값은 **http** 입니다.
staticConfigs	[]string	**<hosts>:<port> 형식으로 정적으로 구성된 Alertmanager 끝점 목록입니다.**
timeout	*string	경고를 보낼 때 사용되는 시간 초과 값을 정의합니다.
tlsConfig	TLSConfig	Alertmanager 연결에 사용할 TLS 설정을 정의합니다.

15.3. ALERTMANAGERMAINCONFIG

15.3.1. 설명

AlertmanagerMainConfig 리소스는 openshift-monitoring 네임스페이스에서 Alertmanager 구성 요소에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
enabled	*bool	**openshift-monitoring** 네임스페이스에서 기본 Alertmanager 인스턴스를 활성화하거나 비활성화하는 부울 플래그입니다. 기본값은 **true**입니다.
enableUserAlertmanagerConfig	bool	**AlertmanagerConfig** 조회에 대해 사용자 정의 네임스페이스를 선택하거나 비활성화하는 부울 플래그입니다. 이 설정은 Alertmanager의 사용자 워크로드 모니터링 인스턴스가 활성화되지 않은 경우에만 적용됩니다. 기본값은 **false**입니다.
logLevel	string	Alertmanager에 대한 로그 수준 설정을 정의합니다. 가능한 값은 **error**,**warn**,**info**,**debug** 입니다. 기본값은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약된 노드를 정의합니다.
resources	*v1.ResourceRequirements	Alertmanager 컨테이너에 대한 리소스 요청 및 제한을 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	Alertmanager에 대한 영구 스토리지를 정의합니다. 이 설정을 사용하여 스토리지 클래스, 볼륨 크기, 이름을 포함하여 영구 볼륨 클레임을 구성합니다.

15.4. ALERTMANAGERUSERWORKLOADCONFIG

15.4.1. 설명

AlertmanagerUserWorkloadConfig 리소스는 사용자 정의 프로젝트에 사용되는 Alertmanager 인스턴스의 설정을 정의합니다.

UserWorkload Configuration에 표시

속성	유형	설명
enabled	bool	**openshift-user-workload-monitoring** 네임스페이스에서 사용자 정의 경고에 대해 Alertmanager 전용 인스턴스를 활성화하거나 비활성화하는 부울 플래그입니다. 기본값은 **false**입니다.
enableAlertmanagerConfig	bool	**AlertmanagerConfig** 조회에 대해 사용자 정의 네임스페이스를 활성화하거나 비활성화하는 부울 플래그입니다. 기본값은 **false**입니다.
logLevel	string	사용자 워크로드 모니터링에 대한 Alertmanager에 대한 로그 수준 설정을 정의합니다. 가능한 값은 **오류**,**경고**,**info**, **debug** 입니다. 기본값은 **info** 입니다.
resources	*v1.ResourceRequirements	Alertmanager 컨테이너에 대한 리소스 요청 및 제한을 정의합니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	Alertmanager에 대한 영구 스토리지를 정의합니다. 이 설정을 사용하여 스토리지 클래스, 볼륨 크기 및 이름을 포함하여 영구 볼륨 클레임을 구성합니다.

15.5. CLUSTERMONITORINGCONFIGURATION

15.5.1. 설명

ClusterMonitoringConfiguration 리소스는 openshift-monitoring 네임스페이스의 cluster-monitoring-config 구성 맵을 통해 기본 플랫폼 모니터링 스택을 사용자 지정하는 설정을 정의합니다.

속성	유형	설명

속성	유형	설명
alertmanagerMain	*AlertmanagerMainConfig	**AlertmanagerMainConfig** 는 **openshift-monitoring** 네임스페이스에서 Alertmanager 구성 요소에 대한 설정을 정의합니다.
enableUserWorkload	*bool	**UserWorkloadEnabled** 는 사용자 정의 프로젝트에 대한 모니터링을 활성화하는 부울 플래그입니다.
k8sPrometheusAdapter	*K8sPrometheusAdapter	**K8sPrometheusAdapter** 은 Prometheus Adapter 구성 요소에 대한 설정을 정의합니다.
kubeStateMetrics	*KubeStateMetricsConfig	**KubeStateMetricsConfig** 는 **kube-state-metrics** 에이전트에 대한 설정을 정의합니다.
prometheusK8s	*PrometheusK8sConfig	**PrometheusK8sConfig** 는 Prometheus 구성 요소에 대한 설정을 정의합니다.
prometheusOperator	*PrometheusOperatorConfig	**PrometheusOperatorConfig** 는 Prometheus Operator 구성 요소에 대한 설정을 정의합니다.
openshiftStateMetrics	*OpenShiftStateMetricsConfig	**OpenShiftMetricsConfig** 는 **openshift-state-metrics** 에이전트에 대한 설정을 정의합니다.
telemeterClient	*TelemeterClientConfig	**TelemeterClientConfig** 는 Telemeter Client 구성 요소에 대한 설정을 정의합니다.
thanosQuerier	*ThanosQuerierConfig	**ThanosQuerierConfig** 는 Thanos Querier 구성 요소에 대한 설정을 정의합니다.

15.6. DEDICATEDSERVICEMONITORS

15.6.1. 설명

DedicatedServiceMonitors 리소스를 사용하여 Prometheus Adapter에 대한 전용 서비스 모니터를 구성할 수 있습니다.

표시: K8sPrometheusAdapter

속성	유형	설명
enabled	bool	**enabled** 가 **true** 로 설정되면 CMO(Cluster Monitoring Operator)는 kubelet **/metrics/resource** 끝점을 노출하는 전용 서비스 모니터를 배포합니다. 이 서비스 모니터는 **honorTimestamps: true** 를 설정하고 Prometheus Adapter의 Pod 리소스 쿼리와 관련된 지표만 유지합니다. 또한 Prometheus Adapter는 이러한 전용 메트릭을 사용하도록 구성되어 있습니다. 전반적으로 이 기능은 **oc adm top pod** 명령 또는 Horizontal Pod Autoscaler와 같이 사용되는 Prometheus Adapter 기반 CPU 사용량 측정의 일관성을 향상시킵니다.

15.7. K8SPROMETHEUSADAPTER

15.7.1. 설명

K8sPrometheusAdapter 리소스는 Prometheus Adapter 구성 요소에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
audit	*감사	Prometheus Adapter 인스턴스에서 사용하는 감사 구성을 정의합니다. 가능한 프로필 값은 **metadata**,**요청**,**요청 응답** 및 **없음입니다**. 기본값은 **metadata** 입니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

속성	유형	설명
dedicatedServiceMonitors	*DedicatedServiceMonitors	전용 서비스 모니터를 정의합니다.

15.8. KUBESTATEMETRICSCONFIG

15.8.1. 설명

KubeStateMetricsConfig 리소스는 kube-state-metrics 에이전트에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

15.9. OPENSHIFTSTATEMETRICSCONFIG

15.9.1. 설명

OpenShiftStateMetricsConfig 리소스는 openshift-state-metrics 에이전트에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

15.10. PROMETHEUSK8SCONFIG

15.10.1. 설명

PrometheusK8sConfig 리소스는 Prometheus 구성 요소에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
additionalAlertmanagerConfigs	[]AdditionalAlertmanagerConfig	Prometheus 구성 요소에서 경고를 수신하는 추가 Alertmanager 인스턴스를 구성합니다. 기본적으로 추가 Alertmanager 인스턴스는 구성되지 않습니다.
enforcedBodySizeLimit	string	Prometheus 스크랩 지표에 대한 본문 크기 제한을 적용합니다. 스크랩된 대상의 본문 응답이 제한보다 크면 스크랩이 실패합니다. 다음 값이 유효합니다. 제한 없음, Prometheus 크기 형식(예: 64MB) 의 숫자 값(예: **64MB**) 또는 제한 용량에 따라 제한이 **자동** 으로 계산됨을 나타내는 빈 값이 유효합니다. 기본값은 비어 있으며 이는 제한이 없음을 나타냅니다.
externalLabels	map[string]string	페더레이션, 원격 스토리지, Alertmanager와 같은 외부 시스템과 통신할 때 임의의 시계열 또는 경고에 추가할 레이블을 정의합니다. 기본적으로 레이블은 추가되지 않습니다.
logLevel	string	Prometheus의 로그 수준 설정을 정의합니다. 가능한 값은 **error**,**warn**,**info**, **debug** 입니다. 기본값은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.

속성	유형	설명
queryLogFile	string	PromQL 쿼리가 기록되는 파일을 지정합니다. 이 설정은 파일 이름일 수 있습니다. 이 경우 쿼리가 **/var/log/prometheus** 에서 **emptyDir** 볼륨에 저장되는 경우 **emptyDir** 볼륨이 마운트되고 쿼리가 저장되는 위치의 전체 경로일 수 있습니다. **/dev/stderr**,**/dev/stdout** 또는 **/dev/null** 에 쓰는 것이 지원되지만 다른 **/dev/** 경로에 쓰는 것은 지원되지 않습니다. 상대 경로도 지원되지 않습니다. 기본적으로 PromQL 쿼리는 기록되지 않습니다.
remoteWrite	[]RemoteWriteSpec	URL, 인증, 재지정 설정을 포함한 원격 쓰기 구성을 정의합니다.
resources	*v1.ResourceRequirements	Prometheus 컨테이너의 리소스 요청 및 제한을 정의합니다.
보존	string	Prometheus가 데이터를 보유하는 기간을 정의합니다. 이 정의는 **[0-9]+(ms|s|m|h|d|w|y)** (ms = 밀리초, s= seconds, m = hours, d = days, w = days, y = years)를 사용하여 지정해야 합니다. 기본값은 **15d** 입니다.
retentionSize	string	데이터 블록에 사용되는 최대 디스크 공간과 WAL(Write-ahead 로그)을 정의합니다. 지원되는 값은 **B**,**KB**,**KiB**,**MB**,**MiB**,**GB**,**GiB**, TB,**TB**,**TiB**,**PB**,**PiB**,**EB**, **EiB** 입니다. 기본적으로 제한은 정의되지 않습니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	Prometheus의 영구 스토리지를 정의합니다. 이 설정을 사용하여 스토리지 클래스, 볼륨 크기 및 이름을 포함하여 영구 볼륨 클레임을 구성합니다.

15.11. PROMETHEUSOPERATORCONFIG

15.11.1. 설명

PrometheusOperatorConfig 리소스는 Prometheus Operator 구성 요소에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration,UserWorkloadConfiguration에 표시

속성	유형	설명
logLevel	string	Prometheus Operator의 로그 수준 설정을 정의합니다. 가능한 값은 **오류,경고,info, debug** 입니다. 기본값은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

15.12. PROMETHEUSRESTRICTEDCONFIG

15.12.1. 설명

PrometheusRestrictedConfig 리소스는 사용자 정의 프로젝트를 모니터링하는 Prometheus 구성 요소의 설정을 정의합니다.

UserWorkload Configuration에 표시

속성	유형	설명
additionalAlertmanagerConfigs	[]AdditionalAlertmanagerConfig	Prometheus 구성 요소에서 경고를 수신하는 추가 Alertmanager 인스턴스를 구성합니다. 기본적으로 추가 Alertmanager 인스턴스는 구성되지 않습니다.

속성	유형	설명
enforcedLabelLimit	*uint64	샘플에 허용되는 라벨 수에 대한 구조별 제한을 지정합니다. 지표 레이블을 다시 지정한 후 레이블 수가 이 제한을 초과하면 전체 스크랩이 실패로 처리됩니다. 기본값은 **0** 이며 이는 제한이 설정되지 않았음을 의미합니다.
enforcedLabelNameLengthLimit	*uint64	샘플의 레이블 이름 길이에 대한 구조별 제한을 지정합니다. 지표의 레이블을 다시 지정한 후 레이블 이름의 길이가 이 제한을 초과하면 전체 스크랩이 실패로 처리됩니다. 기본값은 **0** 이며 이는 제한이 설정되지 않았음을 의미합니다.
enforcedLabelValueLengthLimit	*uint64	샘플의 레이블 값 길이에 대한 구조별 제한을 지정합니다. 메트릭의 레이블을 다시 지정한 후 레이블 값의 길이가 이 제한을 초과하면 전체 스크랩이 실패로 처리됩니다. 기본값은 **0** 이며 이는 제한이 설정되지 않았음을 의미합니다.
enforcedSampleLimit	*uint64	허용할 스크랩 샘플 수에 대한 전역 제한을 지정합니다. 이 설정은 값이 **enforced TargetLimit** 보다 큰 경우 사용자 정의 **ServiceMonitor** 또는 **PodMonitor** 오브젝트에 설정된 SampleLimit 값을 재정의합니다. 관리자는 이 설정을 사용하여 전체 샘플 수를 제어할 수 있습니다. 기본값은 **0** 이며 이는 제한이 설정되지 않았음을 의미합니다.
enforcedTargetLimit	*uint64	스크랩 대상 수에 대한 전역 제한을 지정합니다. 이 설정은 값이 **enforcedSampleLimit** 보다 큰 경우 사용자 정의 **ServiceMonitor** 또는 **PodMonitor** 오브젝트에 설정된 **TargetLimit** 값을 재정의합니다. 관리자는 이 설정을 사용하여 전체 대상 수를 제어할 수 있습니다. 기본값은 **0**입니다.

속성	유형	설명
externalLabels	map[string]string	페더레이션, 원격 스토리지, Alertmanager와 같은 외부 시스템과 통신할 때 임의의 시계열 또는 경고에 추가할 레이블을 정의합니다. 기본적으로 레이블은 추가되지 않습니다.
logLevel	string	Prometheus의 로그 수준 설정을 정의합니다. 가능한 값은 **오류,경고,info, debug** 입니다. 기본 설정은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
queryLogFile	string	PromQL 쿼리가 기록되는 파일을 지정합니다. 이 설정은 파일 이름일 수 있습니다. 이 경우 쿼리가 **/var/log/prometheus** 에서 **emptyDir** 볼륨에 저장되는 경우 **emptyDir** 볼륨이 마운트되고 쿼리가 저장되는 위치의 전체 경로일 수 있습니다. **/dev/stderr,/dev/stdout** 또는 **/dev/null** 에 쓰는 것이 지원되지만 다른 **/dev/** 경로에 쓰는 것은 지원되지 않습니다. 상대 경로도 지원되지 않습니다. 기본적으로 PromQL 쿼리는 기록되지 않습니다.
remoteWrite	[]RemoteWriteSpec	URL, 인증, 재지정 설정을 포함한 원격 쓰기 구성을 정의합니다.
resources	*v1.ResourceRequirements	Prometheus 컨테이너의 리소스 요청 및 제한을 정의합니다.
보존	string	Prometheus가 데이터를 보유하는 기간을 정의합니다. 이 정의는 **[0-9]+(ms|s|m|h|d|w|y)** (ms = 밀리초, s= seconds, m = hours, d = days, w = days, y = years)를 사용하여 지정해야 합니다. 기본값은 **15d** 입니다.

속성	유형	설명
retentionSize	string	데이터 블록에 사용되는 최대 디스크 공간과 WAL(Write-ahead 로그)을 정의합니다. 지원되는 값은 **B**,**KB**,**KiB**,**MB**,**MiB**,**GB**,**GiB**, TB,**TB**,**TiB**,**PB**,**PiB**,**EB**, **EiB** 입니다. 기본값은 **nil** 입니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	Prometheus의 영구 스토리지를 정의합니다. 이 설정을 사용하여 볼륨의 스토리지 클래스 및 크기를 구성합니다.

15.13. REMOTEWRITESPEC

15.13.1. 설명

RemoteWriteSpec 리소스는 원격 쓰기 스토리지의 설정을 정의합니다.

15.13.2. 필수 항목

- **url**

PrometheusK8sConfig,**PrometheusRestrictedConfig**에 표시

속성	유형	설명
권한 부여	*monv1.SafeAuthorization	원격 쓰기 스토리지의 권한 부여 설정을 정의합니다.
basicAuth	*monv1.BasicAuth	원격 쓰기 끝점 URL에 대한 기본 인증 설정을 정의합니다.
bearerTokenFile	string	원격 쓰기 엔드포인트에 대한 전달자 토큰이 포함된 파일을 정의합니다. 그러나 실제로 Pod에 보안을 마운트할 수 없기 때문에 서비스 계정의 토큰만 참조할 수 있습니다.

속성	유형	설명
headers	map[string]string	각 원격 쓰기 요청과 함께 보낼 사용자 지정 HTTP 헤더를 지정합니다. Prometheus에서 설정한 헤더는 덮어쓸 수 없습니다.
metadataConfig	*monv1.MetadataConfig	시계열 메타데이터를 원격 쓰기 스토리지로 보내는 설정을 정의합니다.
name	string	원격 쓰기 대기열의 이름을 정의합니다. 이 이름은 메트릭에 사용되며 큐를 구분하기 위해 로깅에 사용됩니다. 지정하는 경우 이 이름은 고유해야 합니다.
oauth2	*monv1.OAuth2	원격 쓰기 엔드포인트에 대한 OAuth2 인증 설정을 정의합니다.
proxyUrl	string	선택적 프록시 URL을 정의합니다.
queueConfig	*monv1.QueueConfig	원격 쓰기 대기열 매개변수에 대한 튜닝 설정을 허용합니다.
remoteTimeout	string	원격 쓰기 엔드포인트에 대한 요청의 시간 제한 값을 정의합니다.
sigv4	*monv1.Sigv4	AWS Signature Version 4 인증 설정을 정의합니다.
tlsConfig	*monv1.SafeTLSConfig	원격 쓰기 끝점에 대한 TLS 인증 설정을 정의합니다.
url	string	샘플을 보낼 원격 쓰기 끝점의 URL을 정의합니다.
writeRelabelConfigs	[]monv1.RelabelConfig	원격 쓰기 레이블 구성 목록을 정의합니다.

15.14. TELEMETERCLIENTCONFIG

15.14.1. 설명

TelemeterClientConfig 리소스는 telemeter-client 구성 요소의 설정을 정의합니다.

15.14.2. 필수 항목

- **nodeSelector**
- **허용 오차**

ClusterMonitoringConfiguration에 표시

속성	유형	설명
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

15.15. THANOSQUERIERCONFIG

15.15.1. 설명

ThanosQuerierConfig 리소스는 Thanos Querier 구성 요소에 대한 설정을 정의합니다.

ClusterMonitoringConfiguration에 표시

속성	유형	설명
enableRequestLogging	bool	요청 로깅을 활성화하거나 비활성화하는 부울 플래그입니다. 기본값은 **false**입니다.
logLevel	string	Thanos Querier의 로그 수준 설정을 정의합니다. 가능한 값은 **오류,경고,info, debug** 입니다. 기본값은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약되는 노드를 정의합니다.
resources	*v1.ResourceRequirements	Thanos Querier 컨테이너에 대한 리소스 요청 및 제한을 정의합니다.

속성	유형	설명
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

15.16. THANOSRULERCONFIG

15.16.1. 설명

ThanosRulerConfig 리소스는 사용자 정의 프로젝트에 대한 Thanos Ruler 인스턴스에 대한 구성을 정의합니다.

UserWorkload Configuration에 표시

속성	유형	설명
additionalAlertmanagerConfigs	[]AdditionalAlertmanagerConfig	Thanos Ruler 구성 요소가 추가 Alertmanager 인스턴스와 통신하는 방법을 설정합니다. 기본값은 **nil** 입니다.
logLevel	string	Thanos Ruler의 로그 수준 설정을 정의합니다. 가능한 값은 **오류,경고,info, debug** 입니다. 기본값은 **info** 입니다.
nodeSelector	map[string]string	포드가 예약된 노드를 정의합니다.
resources	*v1.ResourceRequirements	Thanos Ruler 컨테이너에 대한 리소스 요청 및 제한을 정의합니다.
보존	string	Prometheus가 데이터를 보유하는 기간을 정의합니다. 이 정의는 **[0-9]+(ms|s|m|h|d|w|y)** (ms = 밀리초, s= seconds, m = hours, d = days, w = days, y = years)를 사용하여 지정해야 합니다. 기본값은 **15d** 입니다.
허용 오차	[]v1.Toleration	Pod에 대한 허용 오차를 정의합니다.

속성	유형	설명
volumeClaimTemplate	*monv1.EmbeddedPersistentVolumeClaim	Thanos Ruler의 영구 스토리지를 정의합니다. 이 설정을 사용하여 볼륨의 스토리지 클래스 및 크기를 구성합니다.

15.17. TLSCONFIG

15.17.1. 설명

TLSConfig 리소스는 TLS 연결 설정을 구성합니다.

15.17.2. 필수 항목

- **insecureSkipVerify**

에 나타납니다. additionalAlertmanagerConfig

속성	유형	설명
ca	*v1.SecretKeySelector	원격 호스트에 사용할 CA(인증 기관)가 포함된 보안 키 참조를 정의합니다.
cert	*v1.SecretKeySelector	원격 호스트에 사용할 공개 인증서가 포함된 보안 키 참조를 정의합니다.
key	*v1.SecretKeySelector	원격 호스트에 사용할 개인 키가 포함된 보안 키 참조를 정의합니다.
serverName	string	반환된 인증서의 호스트 이름을 확인하는 데 사용됩니다.
insecureSkipVerify	bool	**true** 로 설정하면 원격 호스트의 인증서 및 이름 확인이 비활성화됩니다.

15.18. USERWORKLOADCONFIGURATION

15.18.1. 설명

UserWorkloadConfiguration 리소스는 openshift-user-workload-monitoring 네임스페이스의 user-workload-monitoring-config 구성 맵에서 사용자 정의 프로젝트를 담당하는 설정을 정의합니다. openshift-monitoring 네임스페이스 아래의 cluster-monitoring-config 구성 맵에서 enableUserWorkload 를 true 로 설정한 후에 UserWorkloadConfiguration 을 활성화할 수 있습니다.

속성	유형	설명
alertmanager	*AlertmanagerUserWorkloadConfig	사용자 워크로드 모니터링에서 Alertmanager 구성 요소에 대한 설정을 정의합니다.
prometheus	*PrometheusRestrictedConfig	사용자 워크로드 모니터링에서 Prometheus 구성 요소의 설정을 정의합니다.
prometheusOperator	*PrometheusOperatorConfig	사용자 워크로드 모니터링에서 Prometheus Operator 구성 요소에 대한 설정을 정의합니다.
thanosRuler	*ThanosRulerConfig	사용자 워크로드 모니터링에서 Thanos Ruler 구성 요소에 대한 설정을 정의합니다.

16장. CLUSTER OBSERVABILITY OPERATOR

16.1. CLUSTER OBSERVABILITY OPERATOR 릴리스 노트

중요

Cluster Observability Operator는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

COO(Cluster Observability Operator)는 관리자가 다양한 서비스 및 사용자가 사용할 수 있도록 독립적으로 구성할 수 있는 독립 실행형 모니터링 스택을 생성할 수 있는 선택적 **OpenShift Container Platform Operator**입니다.

COO는 **OpenShift Container Platform**의 기본 모니터링 기능을 보완합니다. 기본 플랫폼 및 **CCMO(Cluster Monitoring Operator)**에서 관리하는 사용자 워크로드 모니터링 스택과 병렬로 배포할 수 있습니다.

이 릴리스 노트에서는 **OpenShift Container Platform**의 **Cluster Observability Operator** 개발을 추적합니다.

16.1.1. Cluster Observability Operator 0.1.1

이번 릴리스에서는 제한된 네트워크 또는 연결이 끊긴 환경에서 **Operator** 설치를 지원하도록 **Cluster Observability Operator**를 업데이트합니다.

16.1.2. Cluster Observability Operator 0.1

이번 릴리스에서는 **OperatorHub**에서 **Cluster Observability Operator**의 기술 프리뷰 버전을 사용할 수 있습니다.

16.2. CLUSTER OBSERVABILITY OPERATOR 개요

중요

Cluster Observability Operator는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(**SLA**)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

COO(Cluster Observability Operator)는 **OpenShift Container Platform**의 선택적 구성 요소입니다. 다른 서비스 및 사용자가 사용할 수 있도록 독립적으로 구성할 수 있는 독립 실행형 모니터링 스택을 생성하도록 배포할 수 있습니다.

COO는 다음 모니터링 구성 요소를 배포합니다.

- **Prometheus**
- **Thanos Querier** (선택 사항)
- **Alertmanager** (선택 사항)

COO 구성 요소는 기본 클러스터 내부 모니터링 스택과 독립적으로 작동합니다. 이 스택은 **CCMO(Cluster Monitoring Operator)**에서 배포 및 관리합니다. 두 **Operator**에서 배포한 스택 모니터링은 충돌하지 않습니다. **CMO**에서 배포한 기본 플랫폼 모니터링 구성 요소 외에도 **COO** 모니터링 스택을 사용할 수 있습니다.

16.2.1. Cluster Observability Operator 이해

COO(Cluster Observability Operator)에서 생성한 기본 모니터링 스택에는 원격 쓰기를 사용하여 외부 끝점에 지표를 보낼 수 있는 고가용성 **Prometheus** 인스턴스가 포함되어 있습니다.

각 **COO** 스택에는 중앙 위치에서 고가용성 **Prometheus** 인스턴스를 쿼리하는 데 사용할 수 있는 선택적 **Thanos Querier** 구성 요소와 다양한 서비스에 대한 경고 구성을 설정하는 데 사용할 수 있는 선택적 **Alertmanager** 구성 요소가 포함되어 있습니다.

16.2.1.1. Cluster Observability Operator 사용의 이점

COO에서 사용하는 MonitoringStack CRD는 COO 배포 모니터링 구성 요소에 대한 의견이 지정된 기본 모니터링 구성을 제공하지만 보다 복잡한 요구 사항에 맞게 사용자 지정할 수 있습니다.

COO 관리 모니터링 스택을 배포하면CMO(Cluster Monitoring Operator)에서 배포한 코어 플랫폼 모니터링 스택을 사용하여 해결하기가 어렵거나 불가능한 모니터링 요구 사항을 충족할 수 있습니다. COO를 사용하여 배포된 모니터링 스택은 코어 플랫폼 및 사용자 워크로드 모니터링에 비해 다음과 같은 이점이 있습니다.

확장성

사용자는 COO 배포 모니터링 스택에 메트릭을 추가할 수 있습니다. 이는 지원을 손실하지 않고 핵심 플랫폼 모니터링에서는 불가능합니다. 또한 COO 관리 스택은 페더레이션을 사용하여 핵심 플랫폼 모니터링에서 특정 클러스터별 메트릭을 수신할 수 있습니다.

멀티 테넌시 지원

COO는 사용자 네임스페이스당 모니터링 스택을 생성할 수 있습니다. 네임스페이스당 여러 스택을 배포하거나 여러 네임스페이스에 대해 단일 스택을 배포할 수도 있습니다. 예를 들어 클러스터 관리자, SRE 팀 및 개발 팀은 단일 공유 모니터링 구성 요소 스택을 사용하지 않고도 단일 클러스터에 자체 모니터링 스택을 배포할 수 있습니다. 다른 팀의 사용자는 애플리케이션 및 서비스에 대한 별도의 경고, 경고 라우팅 및 경고 수신자와 같은 기능을 독립적으로 구성할 수 있습니다.

확장성

필요에 따라 COO 관리 모니터링 스택을 생성할 수 있습니다. 여러 모니터링 스택을 단일 클러스터에서 실행할 수 있으므로 수동 샤딩을 사용하여 대규모 클러스터를 쉽게 모니터링할 수 있습니다. 이 기능은 지표 수가 단일 Prometheus 인스턴스의 모니터링 기능을 초과하는 경우를 해결합니다.

유연성

OLM(Operator Lifecycle Manager)을 사용하여 COO를 배포하면 OpenShift Container Platform 릴리스 사이클에서 COO 릴리스가 분리됩니다. 이 배포 방법을 사용하면 릴리스 반복이 빨라지고 변화하는 요구 사항 및 문제에 빠르게 대응할 수 있습니다. 또한 COO 관리 모니터링 스택을 배포하여 사용자는 OpenShift Container Platform 릴리스 사이클과는 별도로 경고 규칙을 관리할 수 있습니다.

고도로 사용자 정의

COO는 SSA(Server-Side Apply)를 사용하여 사용자 지정 리소스에 있는 단일 구성 가능한 필드의 소유권을 사용자에게 위임할 수 있습니다.

추가 리소스

- SSA(Server-Side Apply)에 대한 Kubernetes 문서

16.3. CLUSTER OBSERVABILITY OPERATOR 설치

중요

Cluster Observability Operator는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

클러스터 관리자는 OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 OperatorHub에서 Cluster Observability Operator (COO)를 설치할 수 있습니다. OperatorHub는 클러스터에 Operator를 설치하고 관리하는 OLM(Operator Lifecycle Manager)과 함께 작동하는 사용자 인터페이스입니다.

OperatorHub를 사용하여 COO를 설치하려면 클러스터에 Operator 추가에 설명된 절차를 따르십시오.

16.3.1. 웹 콘솔을 사용하여 Cluster Observability Operator 설치 제거

OperatorHub를 사용하여 COO(Cluster Observability Operator)를 설치한 경우 OpenShift Container Platform 웹 콘솔에서 해당 Operator를 제거할 수 있습니다.

사전 요구 사항

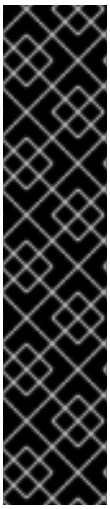
- cluster-admin 클러스터 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- OpenShift Container Platform 웹 콘솔에 로그인했습니다.

절차

1. Operator → 설치된 Operator 로 이동합니다.
2. 목록에서 Cluster Observability Operator 항목을 찾습니다.

3. 이 항목에 대해 ⋮ 를 클릭하고 **Operator** 설치 제거를 선택합니다.

16.4. 서비스를 모니터링하도록 CLUSTER OBSERVABILITY OPERATOR 구성



중요

Cluster Observability Operator는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

COO(Cluster Observability Operator)에서 관리하는 모니터링 스택을 구성하여 서비스에 대한 메트릭을 모니터링할 수 있습니다.

서비스 모니터링을 테스트하려면 다음 단계를 따르십시오.

- 서비스 엔드포인트를 정의하는 샘플 서비스를 배포합니다.
- **COO**에서 서비스를 모니터링할 방법을 지정하는 **ServiceMonitor** 오브젝트를 생성합니다.
- **MonitoringStack** 오브젝트를 생성하여 **ServiceMonitor** 오브젝트를 검색합니다.

16.4.1. Cluster Observability Operator의 샘플 서비스 배포

이 구성은 사용자 정의 **ns1-coo** 프로젝트에 **prometheus-coo-example-app** 이라는 샘플 서비스를 배포합니다. 서비스는 사용자 정의 버전 지표를 노출합니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 또는 네임스페이스에 대한 관리 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

절차

1. 네임스페이스, 배포 및 서비스에 대한 다음 구성 세부 정보가 포함된 prometheus-coo-example-app.yaml 이라는 YAML 파일을 생성합니다.

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

2. 파일을 저장합니다.

3. 다음 명령을 실행하여 클러스터에 구성을 적용합니다.

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

4. 다음 명령을 실행하고 출력을 관찰하여 포드가 실행 중인지 확인합니다.

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

출력 예

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

16.4.2. Cluster Observability Operator에서 서비스를 모니터링하는 방법 지정

"Cluster Observability Operator의 샘플 서비스 배포" 섹션에서 생성한 샘플 서비스에서 노출하는 메트릭을 사용하려면 /metrics 끝점에서 메트릭을 스크랩하도록 모니터링 구성 요소를 구성해야 합니다.

서비스 모니터링 방법을 지정하는 ServiceMonitor 오브젝트 또는 Pod를 모니터링할 방법을 지정하는 PodMonitor 오브젝트를 사용하여 이 구성을 생성할 수 있습니다. ServiceMonitor 오브젝트에는 Service 오브젝트가 필요합니다. PodMonitor 오브젝트는 MonitoringStack 오브젝트가 Pod에서 노출하는 메트릭 끝점에서 직접 메트릭을 스크랩할 수 있도록 하지 않습니다.

다음 절차에서는 ns1-coo 네임스페이스에서 prometheus-coo-example-app 이라는 샘플 서비스에 대한 ServiceMonitor 오브젝트를 생성하는 방법을 보여줍니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 또는 네임스페이스에 대한 관리 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

- Cluster Observability Operator가 설치되어 있습니다.
- prometheus-coo-example-app 샘플 서비스를 ns1-coo 네임스페이스에 배포했습니다.

참고

prometheus-coo-example-app 샘플 서비스는 TLS 인증을 지원하지 않습니다.

절차

1. 다음 ServiceMonitor 오브젝트 구성 세부 정보가 포함된 example-coo-app-service-monitor.yaml 이라는 YAML 파일을 생성합니다.

```
apiVersion: monitoring.rhobs/v1alpha1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus-coo-example-monitor
  name: prometheus-coo-example-monitor
  namespace: ns1-coo
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-coo-example-app
```

이 구성은 MonitoringStack 오브젝트가 prometheus-coo-example-app 샘플 서비스에서 노출하는 메트릭 데이터를 스크랩하기 위해 참조하는 ServiceMonitor 오브젝트를 정의합니다.

2. 다음 명령을 실행하여 클러스터에 구성을 적용합니다.

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

3. 다음 명령을 실행하고 출력을 관찰하여 ServiceMonitor 리소스가 생성되었는지 확인합니다.

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

출력 예

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

16.4.3. Cluster Observability Operator에 대한 MonitoringStack 오브젝트 생성

대상 prometheus-coo-example-app 서비스에서 노출하는 메트릭 데이터를 스크랩하려면 "Cluster Observability Operator에 대해 서비스 모니터링 방법 연결" 섹션에서 생성한 ServiceMonitor 오브젝트를 참조하는 MonitoringStack 오브젝트를 생성합니다. 그러면 이 MonitoringStack 오브젝트에서 서비스를 검색하고 노출된 지표 데이터를 스크랩할 수 있습니다.

사전 요구 사항

- cluster-admin 클러스터 역할의 사용자로 또는 네임스페이스에 대한 관리 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.
- Cluster Observability Operator가 설치되어 있습니다.
- prometheus-coo-example-app 샘플 서비스를 ns1-coo 네임스페이스에 배포했습니다.
- ns1-coo 네임스페이스에 prometheus-coo-example-monitor 라는 ServiceMonitor 오브젝트를 생성했습니다.

절차

1. MonitoringStack 오브젝트 구성에 대한 YAML 파일을 생성합니다. 이 예제에서는 example-coo-monitoring-stack.yaml 파일의 이름을 지정합니다.
2. 다음 MonitoringStack 오브젝트 구성 세부 정보를 추가합니다.

MonitoringStack 오브젝트의 예

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

3. 다음 명령을 실행하여 **MonitoringStack** 오브젝트를 적용합니다.

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

4. 다음 명령을 실행하고 출력을 검사하여 **MonitoringStack** 오브젝트를 사용할 수 있는지 확인합니다.

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

출력 예

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