OpenShift Container Platform 4.7

모니터링

OpenShift Container Platform에서 모니터링 스택 구성 및 사용

Last Updated: 2023-04-08

OpenShift Container Platform 4.7 모니터링

OpenShift Container Platform에서 모니터링 스택 구성 및 사용

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

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 핵심 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치 및 자체 업데이트 모니터링 스택이 포함되어 있습니다. 사용자 정의 프로젝트에 대한 모니터링을 활성화하는 옵션도 있습니다.

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

클러스터 관리자에게 클러스터 문제에 대해 즉시 알리는 일련의 경보가 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 메트릭을 보고 관리할 수 있으며 모니터링 대시보드를 보고 관리할 수 있습니다.https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/monitoring/#managing-alerts OpenShift Container Platform은 Prometheus, Alertmanager 및 Grafana와 같은 타사 인터페이스 에도 액세스할 수 있습니다.

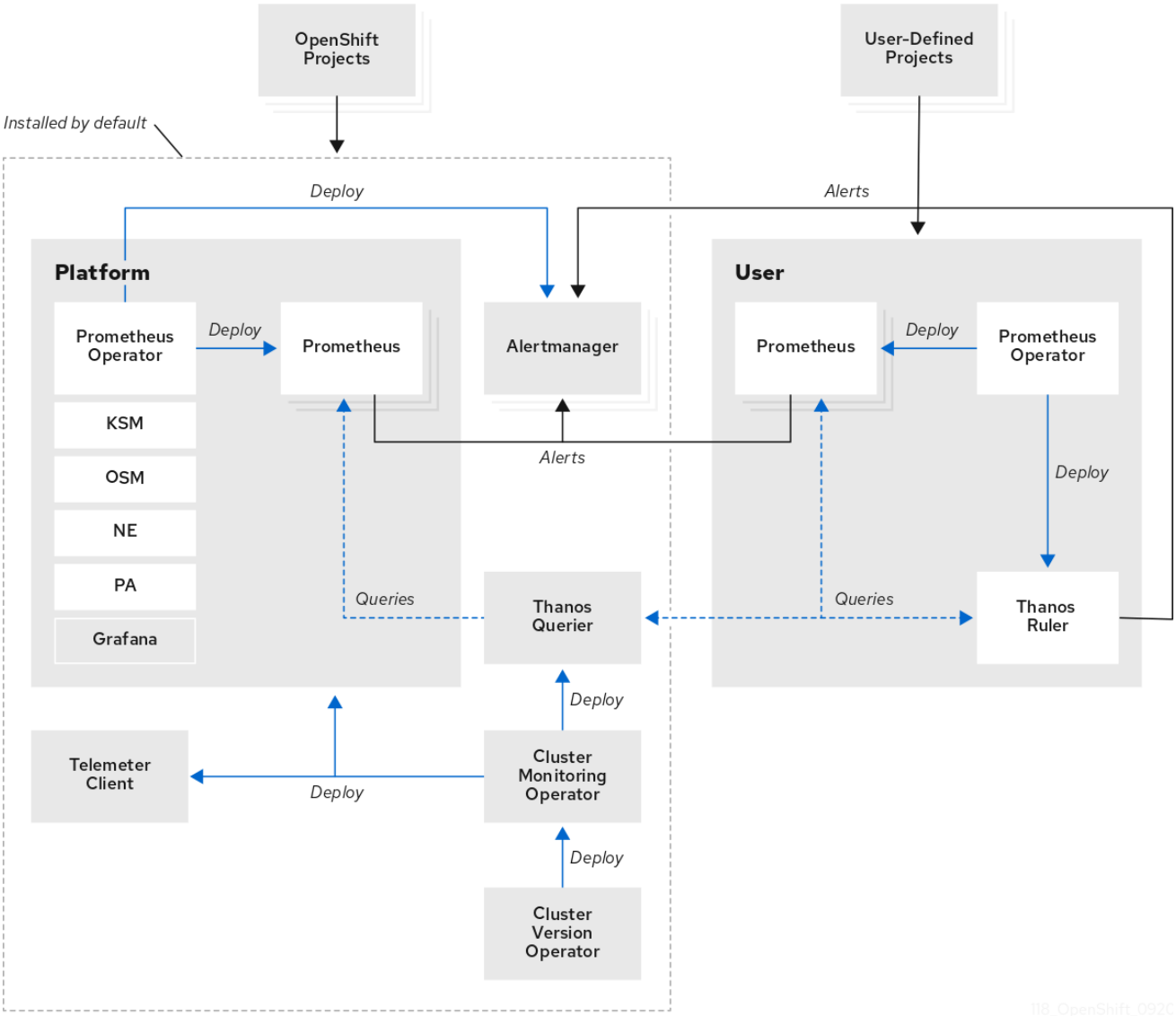
OpenShift Container Platform 4.7을 설치하면 클러스터 관리자가 선택 옵션으로 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다. 이 기능을 사용하면 클러스터 관리자, 개발자, 기타 사용자가 자신의 프로젝트에서 서비스와 Pod를 모니터링하는 방법을 지정할 수 있습니다. 수평 Pod 자동 스케일링을 위해 사용자 정의 애플리케이션 지표를 노출할 수도 있습니다. 클러스터 관리자는 모니터링 문제 해결에 많은 디스크 공간을 사용하는 사용자 지표 Unavailability 및 Prometheus와 같은 일반적인 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 4.7 모니터링 스택에는 다음과 같은 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 인스턴스, Thanos Querier, Telemeter Client 및 메트릭 대상을 배포 및 관리하고 이를 최신 상태로 유지합니다. 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 하나의 다중 테넌트 인터페이스에서 핵심 OpenShift Container Platform 메트릭과 사용자 정의 프로젝트에 대한 메트릭을 집계하고 선택적으로 중복을 제거합니다.
Grafana	Grafana 분석 플랫폼은 메트릭을 분석하고 시각화하기 위한 대시보드를 제공합니다. 대시보드와 함께 모니터링 스택을 통해 제공되는 Grafana 인스턴스는 읽기 전용입니다.
Telemeter Client	Telemeter Client는 플랫폼 Prometheus 인스턴스에서 Red Hat으로 데이터의 하위 섹션을 보내 클러스터의 원격 상태 모니터링이 용이해집니다.

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

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 apiserver
- Kubernetes 컨트롤러 관리자
- Kubernetes 스케줄러
- 미터링(미터링이 설치된 경우)
- OpenShift apiserver
- OpenShift 컨트롤러 관리자
- 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.

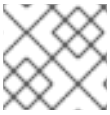
1.2.3. 사용자 정의 프로젝트를 모니터링하기 위한 구성 요소

OpenShift Container Platform 4.7에는 사용자 정의 프로젝트에서 서비스와 Pod를 모니터링할 수 있는 모니터링 스택에 선택적 기능이 포함되어 있습니다. 이 기능에는 다음과 같은 구성 요소가 포함됩니다.

표 1.2. 사용자 정의 프로젝트를 모니터링하기 위한 구성 요소

구성 요소	설명

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 4.7에서 Thanos Ruler는 사용자 정의 프로젝트의 모니터링에 대한 규칙 및 경고 평가를 제공합니다.



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. 추가 리소스

- 원격 상태 모니터링 정보
- 사용자 정의 프로젝트를 모니터링할 수 있는 사용자 권한 부여

1.4. 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.

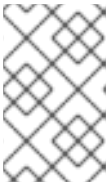
2.2. 모니터링의 유지보수 및 지원

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, 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** 프로젝트에서 시크릿 리소스로 배포됩니다. Alertmanager에 대한 추가 경로를 구성하려면 해당 시크릿을 디코딩하고 수정한 후 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.
- **모니터링 스택 Grafana 인스턴스 수정.**
- **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.**
- **Prometheus Operator에서 AlertmanagerConfig CRD를 사용하여 Alertmanager 구성 수정.**

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

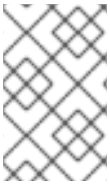
CVO 재정의를 설정하면 전체 클러스터가 지원되지 않는 상태가 되고 모니터링 스택이 의도된 상태와 조정되지 않도록 합니다. 이는 Operator에 빌드된 신뢰성 기능에 영향을 미치며 업데이트가 수신되지 않습니다. 지원을 계속하려면 재정의를 제거한 후 보고된 문제를 재현해야 합니다.

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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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.

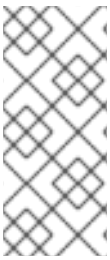
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
```

   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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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:

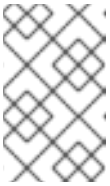
   a. 다음 YAML 매니페스트를 생성합니다. 이 예제에서는 파일을 **user-workload-monitoring-config.yaml**이라고 합니다.

```
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 4.7에서는 **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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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

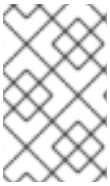
- 모니터링 구성 맵을 생성하는 단계를 위해 모니터링 스택 구성 준비 를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화

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**	
kube-state-metrics	**kubeStateMetrics**	
openshift-state-metrics	**openshiftStateMetrics**	
Grafana	**grafana**	
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. 다른 노드로 모니터링 구성 요소 이동

모니터링 스택 구성 요소 중 하나를 특정 노드로 이동할 수 있습니다.

사전 요구 사항

- **핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:**
  - **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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 요소를 이동하려면**다음을 수행합니다.
     a. **openshift-monitoring** 프로젝트에서 **cluster-monitoring-config ConfigMap** 오브젝트를 편집합니다.

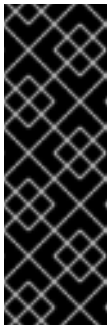
```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. **data/config.yaml**에서 구성 요소에 대한 **nodeSelector** 제약 조건을 지정합니다.

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

     이에 따라 **<component>**를 바꾸고 **<node_key>:<node_value>**를 대상 노드의 그룹을 지정하는 키-값 쌍으로 바꿉니다. 종종 단일 키-값 쌍만 사용됩니다.

구성 요소는 각 지정된 키-값 쌍을 라벨로 갖는 노드에서만 실행할 수 있습니다. 노드에도 추가 라벨이 있을 수 있습니다.



중요

대부분의 모니터링 구성 요소는 클러스터의 다른 노드에서 여러 Pod를 사용하여 고가용성을 유지함으로써 배포됩니다. 모니터링 구성 요소를 라벨된 노드로 이동할 때 구성 요소의 탄력성을 유지하기 위해 일치하는 노드를 사용할 수 있는지 확인합니다. 하나의 라벨만 지정된 경우 별도의 노드 간에 구성 요소에 대한 모든 Pod를 배포할 수 있는 충분한 노드에 해당 라벨이 포함되어 있는지 확인합니다. 또는 개별 노드와 관련된 각 라벨을 여러 개 지정할 수 있습니다.

참고

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

예를 들어 핵심 OpenShift Container Platform 프로젝트의 모니터링 구성 요소를 **nodename: controlplane1**, **nodename: worker1**, **nodename: worker2** 및 **nodename: worker2**의 라벨이 지정된 특정 노드로 이동하려면 다음을 사용합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: controlplane1
    prometheusK8s:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    alertmanagerMain:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    kubeStateMetrics:
      nodeSelector:
        nodename: worker1
    grafana:
      nodeSelector:
        nodename: worker1
    telemeterClient:
      nodeSelector:
        nodename: worker1
    k8sPrometheusAdapter:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    openshiftStateMetrics:
```

```
    nodeSelector:
      nodename: worker1
  thanosQuerier:
    nodeSelector:
      nodename: worker1
      nodename: worker2
```

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

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

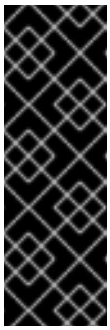
  ```
  $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
  ```

  b. **data/config.yaml**에서 구성 요소에 대한 **nodeSelector** 제약 조건을 지정합니다.

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

  이에 따라 **<component>**를 바꾸고 **<node_key>: <node_value>**를 대상 노드를 지정하는 키-값 쌍의 맵으로 바꿉니다. 종종 단일 키-값 쌍만 사용됩니다.

  구성 요소는 각 지정된 키-값 쌍을 라벨로 갖는 노드에서만 실행할 수 있습니다. 노드에도 추가 라벨이 있을 수 있습니다.



중요

대부분의 모니터링 구성 요소는 클러스터의 다른 노드에서 여러 Pod를 사용하여 고가용성을 유지함으로써 배포됩니다. 모니터링 구성 요소를 라벨된 노드로 이동할 때 구성 요소의 탄력성을 유지하기 위해 일치하는 노드를 사용할 수 있는지 확인합니다. 하나의 라벨만 지정된 경우 별도의 노드 간에 구성 요소에 대한 모든 Pod를 배포할 수 있는 충분한 노드에 해당 라벨이 포함되어 있는지 확인합니다. 또는 개별 노드와 관련된 각 라벨을 여러 개 지정할 수 있습니다.

참고

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

예를 들어 사용자 정의 프로젝트의 구성 요소를 **nodename: worker1**, **nodename: worker2** 및 **nodename: worker2**의 라벨이 지정된 특정 작업자 노드로 이동하려면 다음을 사용합니다.

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

2. 파일을 저장하여 변경 사항을 적용합니다. 새 구성의 영향을 받는 구성 요소는 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 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 단계를 위해 모니터링 스택 구성 준비 를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화
- 노드에서 라벨을 업데이트하는 방법 이해
- 노드 선택기를 사용하여 특정 노드에 Pod 배치
- **nodeSelector** 제약 조건에 대한 자세한 내용은 Kubernetes 문서 를 참조하십시오.

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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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.

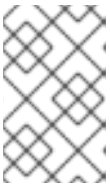
  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: user-workload-monitoring-config
    namespace: openshift-user-workload-monitoring
  data:
    config.yaml: |
      <component>:
        tolerations:
          <toleration_specification>
  ```

  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 오차가 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

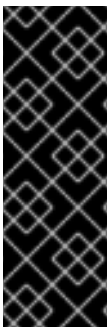
모니터링 구성 맵에 변경 사항이 저장되면 관련 프로젝트의 Pod 및 기타 리소스가 재배포될 수 있습니다. 해당 프로젝트에서 실행 중인 모니터링 프로세스도 다시 시작할 수 있습니다.

추가 리소스

- 모니터링 구성 맵을 생성하는 단계를 위해 모니터링 스택 구성 준비 를 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 문서 를 참조하십시오.

2.8. 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.



중요

Prometheus의 PVC로 연결된 PVC를 사용하여 클러스터 모니터링을 실행 중인 경우 OOM이 클러스터 업그레이드 중에 종료될 수 있습니다. Prometheus에 영구 스토리지를 사용하는 경우 클러스터 업그레이드 중 그리고 업그레이드가 완료된 후 몇 시간 동안 Prometheus 메모리 사용량이 두 배로 증가합니다. OOM 종료 문제가 발생하지 않도록 하려면 업그레이드 전에 사용 가능한 메모리 크기의 두 배인 작업자 노드를 허용합니다. 예를 들어 최소 권장 노드(8GB RAM이 있는 코어 2개)에서 모니터링을 실행 중인 경우 메모리를 16GB로 늘립니다. 자세한 내용은 BZ#1925061을 참조하십시오.

참고

권장되는 구성 가능한 스토리지 기술 을 참조하십시오.

2.8.1. 영구 스토리지 사전 요구 사항

- 디스크가 가득 차지 않도록 충분한 로컬 영구 스토리지를 전용으로 지정합니다. 필요한 스토리지의 양은 Pod 수에 따라 달라집니다. 영구 스토리지의 시스템 요구 사항에 대한 자세한 내용은 Prometheus 데이터베이스 스토리지 요구 사항 을 참조하십시오.

- 각 복제본에 대해 하나의 PV씩, PVC(영구 볼륨 클레임)가 PV(영구 볼륨)를 요청할 준비가 되어 있는지 확인합니다. Prometheus에는 두 개의 복제본이 있고 Alertmanager에는 세 개의 복제본이 있으므로 전체 모니터링 스택을 지원하기 위해 5개의 PV가 필요합니다. Local Storage Operator에서 PV를 사용할 수 있어야 합니다. 동적으로 프로비저닝된 스토리지를 활성화하면 이 문제는 적용되지 않습니다.
- 스토리지의 블록 유형을 사용합니다.
- 로컬 영구 스토리지를 구성합니다.

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.

2.8.2. 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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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 문서**를 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-storage**라고 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 문서**를 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 Operator에 의해 생성된 스토리지 클래스를 **local-storage**라고 합니다.

다음 예제에서는 Thanos Ruler의 로컬 영구 스토리지를 요청하는 PVC를 구성합니다.

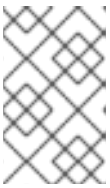
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
```

```
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 Pod가 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.

2.8.3. Prometheus 메트릭 데이터의 보존 시간 수정

기본적으로 OpenShift Container Platform 모니터링 스택은 Prometheus 데이터에 대한 보존 시간을 15일로 구성합니다. 보존 시간을 수정하여 데이터가 삭제되는 시간을 변경할 수 있습니다.

사전 요구 사항

- **핵심 OpenShift Container Platform 모니터링 구성 요소인 경우:**
  - **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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 Prometheus 인스턴스의 보존 시간을 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 구성을 추가합니다.

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

     <time_specification>을 **ms**(밀리초), **s**(초), **m**(분), **h**(시간), **d**(일), **w**(주) 또는 **y**(년)가 바로 따라오는 숫자로 바꿉니다.

     다음 예제에서는 핵심 OpenShift Container Platform 구성 요소를 모니터링하는 Prometheus 인스턴스의 보존 시간을 24시간으로 설정합니다.

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

   - **사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스의 보존 시간을 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 구성을 추가합니다.

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

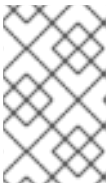
```
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: <time_specification>
```

<time_specification>을 **ms**(밀리초), **s**(초), **m**(분), **h**(시간), **d**(일), **w**(주) 또는 **y**(년)가 바로 따라오는 숫자로 바꿉니다.

다음 예제에서는 사용자 정의 프로젝트를 모니터링하는 Prometheus 인스턴스의 보존 시간을 24시간으로 설정합니다.

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

2. 파일을 저장하여 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 단계를 위해 모니터링 스택 구성 준비를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화
- 영구저장장치 이해
- 스토리지 최적화

2.9. 사용자 정의 프로젝트에서 바인딩되지 않은 메트릭 속성의 영향 제어

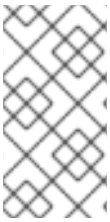
개발자는 라벨을 생성하여 키-값 쌍의 형식으로 메트릭의 속성을 정의할 수 있습니다. 잠재적인 키-값 쌍

의 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.

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

- 사용자 정의 프로젝트에서 대상 스크랩별로 **허용 가능한 샘플 수 제한**
- 스크랩 샘플 임계값에 도달하거나 대상을 스크랩할 수 없는 **경고 생성**



참고

스크랩 샘플 제한으로 인해 라벨에 많은 바인딩되지 않는 속성을 추가하여 문제가 발생하지 않도록 할 수 있습니다. 개발자는 메트릭에 대해 정의된 바인딩되지 않은 속성 수를 제한하여 기본 원인을 방지할 수도 있습니다. 사용 가능한 값의 제한된 집합에 바인딩되는 속성을 사용하면 가능한 키 - 값 쌍 조합의 수가 줄어듭니다.

2.9.1. 사용자 정의 프로젝트에 대한 스크랩 샘플 제한 설정

사용자 정의 프로젝트에서 대상 스크랩별로 허용할 수 있는 샘플 수를 제한할 수 있습니다.

주의

샘플 제한을 설정하면 제한에 도달한 후 해당 대상 스크랩에 대한 추가 샘플 데이터가 사용되지 않습니다.

사전 요구 사항

- **cluster-admin** 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. **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.yaml**에 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. 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.

2.9.2. 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** 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 true여야 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 **기간**을

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.

2.10. 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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **사용자 정의 프로젝트를 모니터링하는 구성 요소를 구성하는 경우:**
  - **cluster-admin** 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_replica**를 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_replica**를 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 **prometheus**에 대한 **externalLabels**를 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 단계를 위해 모니터링 스택 구성 준비 를 참조하십시오.
- 사용자 정의 프로젝트 모니터링 활성화
- 모니터링 구성 맵을 생성하는 단계를 위해 모니터링 스택 구성 준비 를 참조하십시오.

2.11. 모니터링 구성 요소에 대한 로그 수준 설정

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

참고

Alertmanager 구성 요소에 대한 로그 수준을 구성하려면 이 절차를 사용할 수 없습니다.

다음 로그 수준은 **cluster-monitoring-config** 및 **user-workload-monitoring-config ConfigMap** 오브젝트의 각 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 Prometheus Operator, Prometheus 또는 Thanos Querier의 로그 수준을 설정하는 경우**:
  - **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
  - **cluster-monitoring-config ConfigMap** 오브젝트를 생성하셨습니다.
- **openshift-user-workload-monitoring 프로젝트에서 Prometheus Operator, Prometheus 또는 Thanos Ruler의 로그 수준을 설정하는 경우**:
  - **cluster-admin** 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 적용하려는 모니터링 구성 요소입니다.

     2 구성 요소에 적용할 로그 수준입니다.

   - **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 적용하려는 모니터링 구성 요소입니다.

     2 구성 요소에 적용할 로그 수준입니다.

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 Pod가 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.

추가 리소스

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

2.12. 다음 단계

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

3장. 사용자 정의 프로젝트 모니터링 활성화

OpenShift Container Platform 4.7에서는 기본 플랫폼 모니터링 외에도 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다. 이제 추가 모니터링 솔루션 없이도 OpenShift Container Platform에서 자체 프로젝트를 모니터링할 수 있습니다. 이 새로운 기능을 사용하면 핵심 플랫폼 구성 요소 및 사용자 정의 프로젝트에 대한 모니터링을 한 곳에서 수행할 수 있습니다.

참고

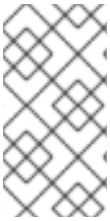
OLM(Operator Lifecycle Manager)을 사용하여 설치한 Prometheus Operator 버전은 사용자 정의 모니터링과 호환되지 않습니다. 따라서 OLM Prometheus Operator에서 관리하는 Prometheus 사용자 정의 리소스(CR)로 설치된 사용자 정의 Prometheus 인스턴스는 OpenShift Container Platform에서 지원되지 않습니다.

3.1. 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 4.7에서는 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** 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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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.yaml**에 **enableUserWorkload: true**를 추가합니다.

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

(1) **true**로 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** Pod가 **openshift-user-workload-monitoring** 프로젝트에서 실행 중인지 확인합니다. Pod를 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

3.2. 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** 역할은 프로젝트의 **PrometheusRule** 사용자 정의 리소스에 대한 읽기 액세스를 제공합니다.
- **monitoring-rules-edit** 역할은 사용자에게 프로젝트의 **PrometheusRule** 사용자 정의 리소스를 생성, 수정, 삭제할 수 있는 권한을 부여합니다.
- **monitoring-edit** 역할은 **monitoring-rules-edit** 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 통해 **ConfigMap** 오브젝트를 편집하여 사용자 정의 워크로드 모니터링을 위해 Prometheus, Prometheus Operator 및 Thanos Ruler를 구성할 수 있습니다.

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

3.2.1. 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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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 **사용자**를 선택합니다.
7. **주체 이름** 필드에 사용자 이름을 입력합니다.
8. 역할 바인딩을 적용하려면 **만들기**를 선택합니다.

3.2.2. CLI를 사용하여 사용자에게 권한 부여

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

사전 요구 사항

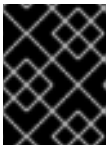
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 이미 존재하는 역할에 할당 중인 사용자 계정입니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

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-edit**로 바꿉니다.



중요

선택한 역할이 무엇이든 클러스터 관리자로 특정 프로젝트에 대해 바인딩해야 합니다.

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

3.3. 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** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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
```

3.4. 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 들면 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"
```

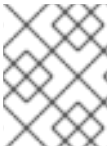
출력에 애플리케이션 pod가 가동된 기간이 표시됩니다.

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"]}]}}
```

3.5. 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
```

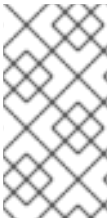
2. 파일을 저장하여 변경 사항을 적용합니다. 그런 다음 사용자 정의 프로젝트에 대한 모니터링이 자동으로 비활성화됩니다.

3. **prometheus-operator**, **prometheus-user-workload** 및 **thanos-ruler-user-workload** Pod가 **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.

3.6. 다음 단계

- 메트릭 관리

4장. 메트릭 관리

클러스터 구성 요소 및 자체 워크로드가 수행하는 방법을 모니터링하기 위해 메트릭을 수집할 수 있습니다.

4.1. 메트릭 이해

OpenShift Container Platform 4.7에서 클러스터 구성 요소는 서비스 끝점을 통해 노출된 스크랩 메트릭으로 모니터링됩니다. 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>/metrics**에 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 라이브러리에 대한 자세한 내용은 Prometheus 설명서를 참조하십시오.

4.2. 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.

4.2.1. 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:0.3.0
        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. Pod가 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
```

4.2.2. 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 Pod를 모니터링해야 하는 방법을 지정하는 **PodMonitor** CRD를 사용하여 이 작업을 수행할 수 있습니다. 전자에는 **Service** 오브젝트가 필요하지만 후자에는 필요하지 않으며 Prometheus가 Pod에서 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** 역할 또는 **monitoring-edit** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 이 예제에서는 **prometheus-example-app** 샘플 서비스를 **ns1** 프로젝트에 배포했습니다.

참고

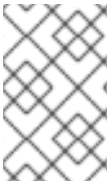
prometheus-example-app 샘플 서비스는 TLS 인증을 지원하지 않습니다.

프로세스

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.

1. 클러스터에 구성을 적용합니다.

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

ServiceMonitor 리소스를 배포하는 데 시간이 다소 걸립니다.

2. **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 TLS를 사용하여 메트릭을 스크랩하는 방법
- PodMonitor API
- ServiceMonitor API

4.3. 메트릭 쿼리

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.

4.3.1. 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.

참고

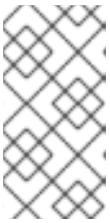
클러스터 관리자만 OpenShift Container Platform 모니터링을 통해 제공되는 타사 UI에 액세스할 수 있습니다.

사전 요구 사항

- **cluster-admin** 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 콘솔의 **관리자** 관점에서 **모니터링 → 메트릭**을 선택합니다.
2. **커서에 표시기 삽입**을 선택하여 사전 정의된 쿼리 목록을 확인합니다.
3. 사용자 정의 쿼리를 생성하려면 **표현식** 필드에 PromQL(Prometheus Query Language) 쿼리를 추가합니다.
4. 여러 쿼리를 추가하려면 **쿼리 추가**를 선택합니다.
5. 쿼리를 삭제하려면 쿼리 옆에 있는 를 선택한 다음 **쿼리 삭제**를 선택합니다.
6. 쿼리 실행을 비활성화하려면 쿼리 옆에 있는 를 선택하고 **쿼리 비활성화**를 선택합니다.
7. 생성된 쿼리를 실행하려면 **쿼리 실행**을 선택합니다. 쿼리의 메트릭은 플롯에 시각화됩니다. 쿼리가 유효하지 않으면 UI에 오류 메시지가 표시됩니다.



참고

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

8. 선택 사항: 이제 페이지 URL에 실행한 쿼리가 포함됩니다. 나중에 이 쿼리 세트를 다시 사용하려면 이 URL을 저장합니다.

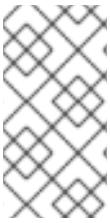
추가 리소스

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

4.3.2. 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 한 번에 하나의 프로젝트의 메트릭만 쿼리할 수 있습니다. 개발자는 코어 플랫폼 구성 요소에 대해 OpenShift Container Platform 모니터링을 통해 제공되는 타사 UI에 액세스할 수 없습니다. 대신 사용자 정의 프로젝트에 Metrics UI를 사용합니다.

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.

절차

1. OpenShift Container Platform 웹 콘솔의 **개발자** 관점에서 **모니터링 → 메트릭**을 선택합니다.
2. **Project:** 목록에서 메트릭을 보려는 프로젝트를 선택합니다.
3. **쿼리 선택** 목록에서 쿼리를 선택하거나 **PromQL 표시**를 선택하여 사용자 정의 PromQL 쿼리를 실행합니다.

참고

개발자 관점에서는 한 번에 하나의 쿼리만 실행할 수 있습니다.

추가 리소스

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

추가 리소스

- 개발자 또는 권한 있는 사용자로 비 클러스터 메트릭에 액세스하는 방법에 대한 자세한 내용은 개발자로 사용자 정의 프로젝트의 메트릭 쿼리를 참조하십시오.

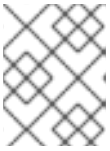
4.3.3. 시각화된 메트릭 살펴보기

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

프로세스

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 숨기기**를 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 재설정**을 선택합니다.
4. 특정 시점에서 모든 쿼리에 대한 출력을 표시하려면 해당 시점에서 플롯에 마우스 커서를 유지합니다. 쿼리 출력이 팝업 상자에 나타납니다.
5. 플롯을 숨기려면 **그래프 숨기기**를 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 재설정**을 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 대해 메트릭 쿼리 섹션을 참조하십시오.

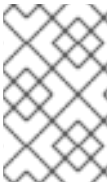
4.4. 다음 단계

- 경고 관리

5장. 경고 관리

OpenShift Container Platform 4.7에서 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-administrator** 권한으로 로그인하면 모든 경고, 음소거, 경고 규칙에 액세스할 수 있습니다.

5.1. 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 **모니터링 → 경고**를 선택합니다. 이 관점에서 경고 UI의 세 가지 주요 페이지는 **경고**, **음소거** 및 **경고 규칙** 페이지입니다.
- **개발자** 관점에서 **모니터링 → <project_name> → 경고**를 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 Platform및 사용자 정의 프로젝트에서 선택할 수 있습니다. 그러나 **cluster-admin** 권한이 없는 경우 핵심 OpenShift Container Platform 프로젝트와 관련된 경고, 음소거, 경고 규칙이 표시되지 않습니다.

5.2. 경고, 음소거, 경고 규칙 검색 및 필터링

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

경고 필터 이해

관리자 관점에서 경고 UI의 **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 **실행**되는 **플랫폼**만 표시됩니다. 다음은 각 경고 필터링 옵션을 설명합니다.

- **경고 상태** 필터:

- **실행**. 경고 조건이 true 이므로 경고가 실행되고 기간 **동안** 선택 사항이 전달됩니다. 상태가 true로 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 관찰성을 제공할 수 있습니다.

음소거 필터 이해

관리자 관점에서 경고 UI의 **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 true이면 음소거가 만료되고 알림이 전송됩니다.

경고 규칙 필터 이해

관리자 관점에서 경고 UI의 **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 true로 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. true인 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 관찰성을 제공할 수 있습니다.

개발자 관점에서 경고, 음소거, 경고 규칙 검색 및 필터링

개발자 관점에서 경고 UI의 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.

5.3. 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 **모니터링 → 경고 → 경고** 페이지로 이동합니다.
2. 선택 사항: 검색 목록의 **Name** (이름) 필드를 사용하여 이름별로 경고를 검색합니다.
3. 선택 사항: 필터 목록에서 필터를 선택하여 상태, 심각도 및 소스별로 알림을 **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. **모니터링 → 경고 → 음소거** 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 헤더 중 하나 이상을 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. **모니터링 → 경고 → 경고 규칙** 페이지로 이동합니다.
2. 선택 사항: 필터 목록에서 필터를 선택하여 상태, 심각도 및 소스별로 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 true여야 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 얻으려면다음을 수행합니다.

1. **모니터링 → <project_name> → 경고** 페이지로 이동합니다.
2. 경고, 음소거 또는 경고 규칙에 대한 세부 정보를 표시합니다.
   - **경고 세부 정보**는 경고 이름 왼쪽의 **>**를 선택한 다음 목록에 있는 경고를 선택하여 볼 수 있습니다.
   - **음소거 상세 정보**는 **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 정보**는 **경고** 페이지에 있는 경고 오른쪽의  메뉴에서 **경고 규칙 보기**를 선택하여 볼 수 있습니다.

참고

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

5.4. 경고 규칙 관리

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

OpenShift Container Platform 4.7에서는 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.

5.4.1. 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.
- **경고 라우팅을 최적화합니다.** 규칙이 기본 OpenShift Container Platform 메트릭으로 쿼리하지 않는 경우 **openshift-user-workload-monitoring** 프로젝트의 Prometheus 인스턴스에 직접 경고 규칙을 배포합니다. 이렇게 하면 경고 규칙에 대한 대기 시간이 줄어들고 모니터링 구성 요소의 부하를 최소화합니다.

주의

사용자 정의 프로젝트에 대한 기본 OpenShift Container Platform 메트릭에서는 CPU 및 메모리 사용량, 대역폭 통계 및 패킷 속도 정보에 대한 정보를 제공합니다. **openshift-user-workload-monitoring** 프로젝트에서 규칙을 직접 Prometheus 인스턴스에 라우팅하면 해당 메트릭을 경고 규칙에 포함할 수 없습니다. 경고 규칙 최적화는 문서를 읽고 모니터링 아키텍처를 포괄적으로 이해하는 경우에만 사용해야 합니다.

추가 리소스

- 경고 최적화에 대한 추가 지침은 Prometheus 경고 문서 를 참조하십시오.
- OpenShift Container Platform 4.7 모니터링 아키텍처에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.

5.4.2. 사용자 정의 프로젝트에 대한 경고 규칙 생성

사용자 정의 프로젝트에 대한 경고 규칙을 생성할 수 있습니다. 이러한 경고 규칙은 선택한 메트릭 값을 기반으로 경고가 실행됩니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 경고 규칙을 생성하려는 프로젝트에 대한 **monitoring-rules-edit** 역할이 있는 사용자로 로그인했습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

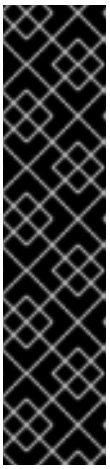
1. 경고 규칙에 사용할 YAML 파일을 생성합니다. 이 예에서는 **example-app-alerting-rule.yaml**이라고 합니다.
2. YAML 파일에 경고 규칙 구성을 추가합니다. 예를 들면 다음과 같습니다.

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.



중요

사용자 정의 경고 규칙에는 자체 프로젝트 및 클러스터 메트릭에 대한 메트릭이 포함될 수 있습니다. 다른 사용자 정의 프로젝트에 대한 메트릭은 포함할 수 없습니다.

예를 들어 사용자 정의 프로젝트 **ns1**에 대한 경고 규칙에는 CPU 및 메모리 메트릭과 같은 **ns1** 및 클러스터 메트릭에서 메트릭이 있을 수 있습니다. 그러나 규칙에는 **ns2**의 메트릭을 포함할 수 없습니다.

또한 **openshift-*** 핵심 OpenShift 프로젝트에 대한 경고 규칙을 생성할 수 없습니다. 기본적으로 OpenShift Container Platform 모니터링은 이러한 프로젝트에 대한 일련의 경고 규칙을 제공합니다.

3. 구성 파일을 리클러스터에 적용합니다.

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

경고 규칙을 생성하는 데 시간이 다소 걸립니다.

5.4.3. 플랫폼 메트리을 쿼리하지 않는 경고 규칙에 대한 대기 시간 감소

사용자 정의 프로젝트의 경고 규칙이 기본 클러스터 메트릭을 쿼리하지 않으면 **openshift-user-workload-monitoring** 프로젝트의 Prometheus 인스턴스에서 직접 규칙을 배포할 수 있습니다. 이로 인해 Thanos Ruler를 우회할 필요가 없는 경우 경고 규칙의 대기 시간이 단축됩니다. 또한 모니터링 구성 요소에 대한 전체 부하를 최소화하는 데 도움이 됩니다.

주의

사용자 정의 프로젝트에 대한 기본 OpenShift Container Platform 메트릭에서는 CPU 및 메모리 사용량, 대역폭 통계 및 패킷 속도 정보에 대한 정보를 제공합니다. **openshift-user-workload-monitoring** 프로젝트의 Prometheus 인스턴스에 직접 규칙을 배포하는 경우 해당 메트릭을 경고 규칙에 포함할 수 없습니다. 이 섹션에 설명된 프로세스는 문서를 읽고 모니터링 아키텍처를 포괄적으로 이해하는 경우에만 사용해야 합니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 경고 규칙을 생성하려는 프로젝트에 대한 **monitoring-rules-edit** 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. 키 **openshift.io/prometheus-rule-evaluation-scope** 및 값 **leaf-prometheus**가 있는 라벨을 포함한 YAML 파일에 경고 규칙 구성을 추가합니다. 예를 들면 다음과 같습니다.

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

해당 라벨이 있는 경우 경고 규칙이 **openshift-user-workload-monitoring** 프로젝트의 Prometheus 인스턴스에 배포됩니다. 라벨이 없으면 경고 규칙이 Thanos Ruler에 배포됩니다.

1. 구성 파일을 리클러스터에 적용합니다.

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

경고 규칙을 생성하는 데 시간이 다소 걸립니다.

- OpenShift Container Platform 4.7 모니터링 아키텍처에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.

5.4.4. 사용자 정의 프로젝트의 경고 규칙에 액세스

사용자 정의 프로젝트의 경고 규칙을 나열하려면 프로젝트에 대한 **monitoring-rules-view** 역할이 할당되어 있어야 합니다.

사전 요구 사항

- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.
- 프로젝트에 대한 **monitoring-rules-view** 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
```

5.4.5. 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.

프로세스

1. **관리자** 관점에서 **모니터링 → 경고 → 경고 규칙**으로 이동합니다.
2. **필터** 드롭다운 메뉴에서 **플랫폼** 및 **사용자** 소스를 선택합니다.

참고

플랫폼 소스가 기본적으로 선택됩니다.

5.4.6. 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** 역할이 있는 사용자로 로그인했습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

- 규칙 **<foo>**를 **<namespace>**에서 제거하려면 다음을 실행합니다.

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

추가 리소스

- Alertmanager 문서 참조

5.5. 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.

5.5.1. 음소거 경고

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

사전 요구 사항

- 개발자로 또는 메트릭을 보고 있는 프로젝트에 대한 **edit** 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

프로세스

특정 경고를 음소거하려면 다음을 수행합니다.

- **관리자** 관점에서:
  1. OpenShift Container Platform 웹 콘솔의 **모니터링 → 경고 → 경고** 페이지로 이동합니다.
  2. 음소거할 경고의 경우 오른쪽 열에 있는  를 선택하고 **음소거 경고**를 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 **음소거**를 선택합니다.

- **Developer** 관점에서:
  1. OpenShift Container Platform 웹 콘솔의 **모니터링 → <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 경고**를 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 **음소거**를 선택합니다.

관리자 관점에서 경고 사양을 생성하여 일련의 경고를 음소거하려면 다음을 수행합니다.

1. OpenShift Container Platform 웹 콘솔의 **모니터링 → 경고 → 음소거** 페이지로 이동합니다.
2. **음소거 상태 만들기**를 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 **음소거**를 선택합니다.

5.5.2. 음소거 편집

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

프로세스

관리자 관점에서 음소거를 편집하려면 다음을 수행합니다.

1. **모니터링 → 경고 → 음소거** 페이지로 이동합니다.
2. 수정하려는 음소거의 경우 마지막 열에서  를 선택하고 **음소거 편집**을 선택합니다.
   또는 음소거에 대한 **음소거 상세 정보** 페이지에서 **동작 → 음소거 편집**을 선택할 수 있습니다.
3. **음소거 편집** 페이지에서 변경 사항을 입력하고 **음소거**를 선택합니다. 이를 통해 기존 음소거가 만료되고 선택한 구성으로 생성됩니다.

개발자 관점에서 음소거를 편집하려면 다음을 수행합니다.

1. **모니터링 → <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 편집**을 선택합니다.

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

5.5.3. 음소거 만료

음소거를 만료할 수 있습니다. 음소거를 만료하면 영구적으로 비활성화됩니다.

프로세스

관리자 관점에서 음소거를 만료하려면 다음을 수행합니다.

1. **모니터링 → 경고 → 음소거** 페이지로 이동합니다.

2. 수정하려는 음소거의 경우 마지막 열에서 를 선택하고 **음소거 만료**를 선택합니다.
또는 음소거에 대한 **음소거 상세 정보** 페이지에서 **동작 → 음소거 만료**를 선택할 수 있습니다.

개발자 관점에서의 음소거를 만료하려면 다음을 수행합니다.

1. **모니터링 → <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 만료**를 선택합니다.

5.6. 외부 시스템에 알림 전송

OpenShift Container Platform 4.7에서는 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. 심각하지 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.

5.6.1. 경고 수신자 구성

클러스터의 중요한 문제를 파악할 수 있도록 경고 수신자를 설정할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **관리자** 관점에서 **관리 → 클러스터 설정 → 글로벌 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 **구성**을 선택합니다.

2. 이 페이지의 **수신자** 섹션에서 **수신자 만들기**를 선택합니다.
3. **수신자 만들기**에서 **수신자 이름**을 추가하고 목록에서 **수신자 유형**을 선택합니다.
4. 수신자 구성을 편집합니다.
   - PagerDuty 수신자의 경우:
     a. 통합 유형을 선택하고 PagerDuty 통합 키를 추가합니다.
     b. PagerDuty 설치의 URL을 추가합니다.
     c. 클라이언트와 인스턴스 세부 정보 또는 심각도 사양을 편집하려면 **고급 설정 표시**를 선택합니다.
   - Webhook 수신자의 경우:
     a. HTTP POST 요청이 전송되는 끝점을 추가합니다.
     b. 해결된 경보를 수신자에게 보내는 기본 옵션을 편집하려면 **고급 설정 표시**를 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. TLS가 필요한지 여부를 선택합니다.

d. 해결된 경고를 수신자에게 보내지 않도록 기본 옵션을 편집하거나 이메일 알림 구성을 편집하려면 **고급 설정 표시**를 선택합니다.

- Slack 수신자의 경우:

a. Slack Webhook의 URL을 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 표시**를 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 **정규식**을 선택합니다.

c. **라벨 추가**를 선택하여 추가 라우팅 라벨을 추가합니다.

6. **만들기**를 선택하여 수신자를 생성합니다.

5.7. 사용자 정의 ALERTMANAGER 설정 적용

openshift-monitoring 프로젝트 내에서 **alertmanager-main** 시크릿을 편집하여 기본 Alertmanager 설정을 덮어쓸 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

CLI에서 Alertmanager 설정을 변경하려면 다음을 수행합니다.

1. 현재 활성화된 Alertmanager 구성을 파일 **alertmanager.yaml**로 출력합니다.

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

2. **alertmanager.yaml**에서 설정을 편집합니다.

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - match:
      alertname: Watchdog
    repeat_interval: 5m
    receiver: watchdog
```

```
  - match:
      service: <your_service> 1
    routes:
    - match:
        <your_matching_rules> 2
      receiver: <receiver> 3
receivers:
- name: default
- name: watchdog
- name: <receiver>
#  <receiver_configuration>
```

1. **service**는 경고를 실행하는 서비스를 지정합니다.
2. **<your_matching_rules>**는 대상 경고를 지정합니다.
3. **receiver**는 경고에 사용할 수신자를 지정합니다.

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:
  - match:
      alertname: Watchdog
    repeat_interval: 5m
    receiver: watchdog
  - match: service: example-app routes: - match: 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 → 글로벌 구성 → Alertmanager → YAML** 페이지로 이동합니다.
2. YAML 설정 파일을 수정합니다.

3. **저장**을 선택합니다.

추가 리소스

- PagerDuty에 대한 자세한 내용은 PagerDuty 공식 사이트를 참조하십시오.
- **service_key** 검색 방법을 알아보려면 PagerDuty Prometheus 통합 가이드를 참조하십시오.
- 다양한 경고 수신자를 통한 경고 구성은 Alertmanager 설정을 참조하십시오.

5.8. 다음 단계

- 모니터링 대시보드 검토

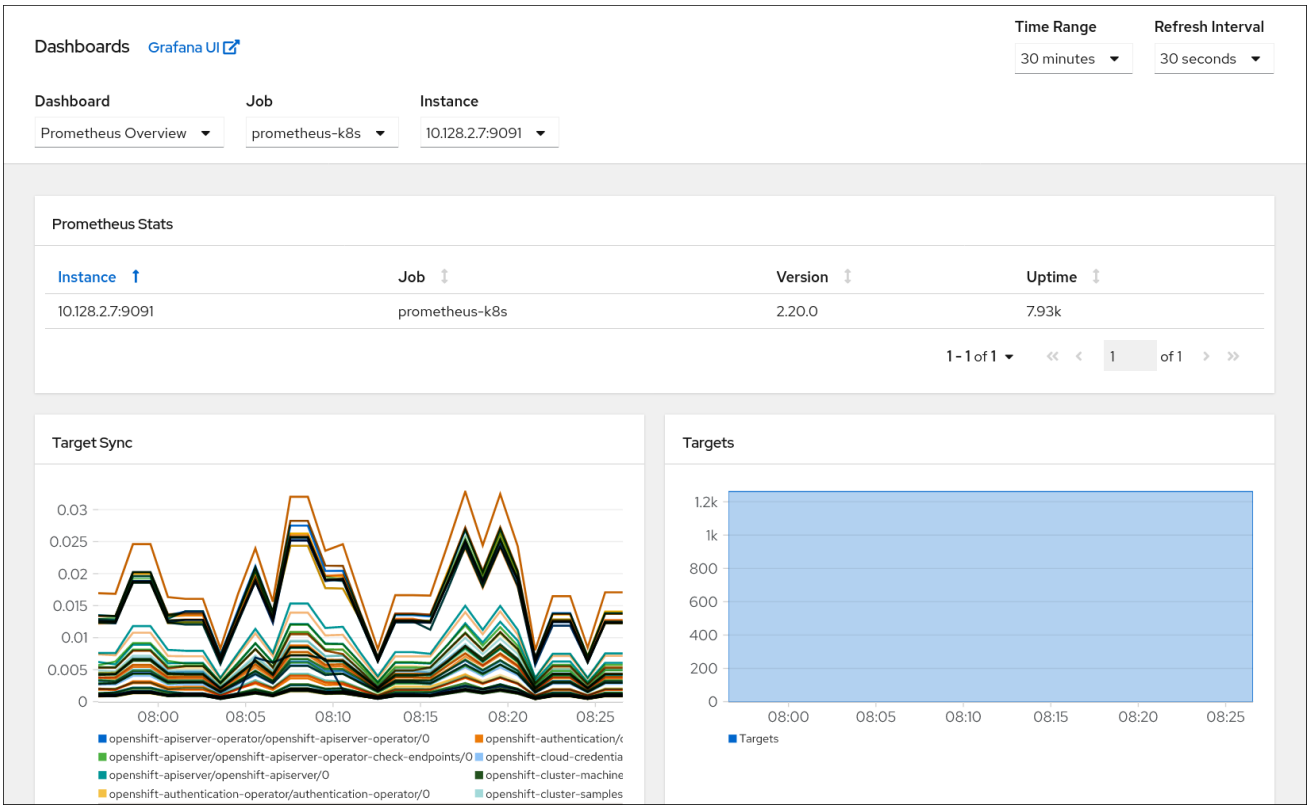
6장. 모니터링 대시보드 검토

OpenShift Container Platform 4.7은 클러스터 구성 요소 및 사용자 정의 워크로드의 상태를 이해하는 데 도움이 되는 포괄적인 모니터링 대시보드 집합을 제공합니다.

관리자 관점에서 다음을 포함하여 핵심 OpenShift Container Platform 구성 요소의 대시보드에 액세스할 수 있습니다.

- API 성능
- etcd
- Kubernetes 컴퓨팅 리소스
- Kubernetes 네트워크 리소스
- Prometheus
- 클러스터 및 노드 성능과 관련된 USE 메서드 대시보드

그림 6.1. 관리자 관점의 대시보드 예



개발자 관점에서 선택한 프로젝트에 대해 다음 통계를 제공하는 대시보드에 액세스할 수 있습니다.

- CPU 사용량
- 메모리 사용량
- 대역폭 정보
- 패킷 속도 정보

그림 6.2. 개발자 관점의 대시보드 예

참고

개발자 관점에서 한 번에 하나의 프로젝트에 대해서만 대시보드를 볼 수 있습니다.

6.1. 클러스터 관리자로 모니터링 대시보드 검토

관리자 관점에서 핵심 OpenShift Container Platform 클러스터 구성 요소와 관련된 대시보드를 볼 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. OpenShift Container Platform 웹 콘솔의 **관리자** 관점에서 **모니터링 → 대시보드**로 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.
4. 선택 사항: 새로 **고침 간격을** 선택합니다.
5. 대시보드 내에서 각각의 그래프로 마우스를 이동하여 특정 항목에 대한 세부 정보를 표시합니다.

6.2. 개발자로 모니터링 대시보드 검토

개발자 관점에서 선택한 프로젝트와 관련된 대시보드를 볼 수 있습니다. 대시보드 정보를 보기 위해 프로젝트를 모니터링하려면 액세스할 수 있어야 합니다.

사전 요구 사항

- 개발자로 또는 대시보드를 보고 있는 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

프로세스

1. OpenShift Container Platform 웹 콘솔의 개발자 관점에서 **모니터링 → 대시보드**로 이동합니다.
2. **Project:** 목록에서 프로젝트를 선택합니다.
3. **모든 워크로드** 목록에서 워크로드를 선택합니다.
4. 선택 사항: 시간 **범위 목록에서 그래프의 시간** 범위를 선택합니다.
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

6.3. 다음 단계

- 타사 UI에 액세스

7장. 타사 UI에 액세스

통합된 메트릭, 경고 및 대시보드 UI는 OpenShift Container Platform 웹 콘솔에서 제공됩니다. 이러한 통합 UI 사용에 대한 자세한 내용은 다음을 참조하십시오.

- 메트릭 관리
- 경고 관리
- 모니터링 대시보드 검토

OpenShift Container Platform은 Prometheus, Alertmanager 및 Grafana 타사 인터페이스에 대한 액세스도 제공합니다.

참고

타사 모니터링 인터페이스에 대한 기본 액세스는 향후 OpenShift Container Platform 릴리스에서 제거될 수 있습니다. 이에 따라 포트 전달을 사용하여 액세스해야 합니다.

참고

대시보드와 함께 OpenShift Container Platform 모니터링 스택을 통해 제공되는 Grafana 인스턴스는 읽기 전용입니다.

참고

Grafana 대시보드에는 Kubernetes 및 **cluster-monitoring** 메트릭만 포함됩니다. 추가 플랫폼 구성 요소는 OpenShift Container Platform 웹 콘솔의 **모니터링 → 대시보드**에 포함되어 있습니다.

7.1. 웹 콘솔을 사용하여 타사 모니터링 UI에 액세스

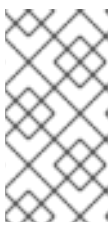
OpenShift Container Platform 웹 콘솔을 통해 Alertmanager, Grafana, Prometheus 및 Thanos Querier 웹 UI에 액세스할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **관리자** 관점에서 **네트워킹 → 라우트**로 이동합니다.



참고

타사 Alertmanager, Grafana, Prometheus 및 Thanos Querier UI에 대한 액세스는 개발자 화면에서 사용할 수 없습니다. 대신 개발자 화면에서 Metrics UI 링크를 사용합니다. 여기에는 선택한 프로젝트에 대한 사전 정의된 CPU, 메모리, 대역폭 및 네트워크 패킷 쿼리가 포함됩니다.

2. **Project:** 목록에서 **openshift-monitoring** 프로젝트를 선택합니다.
3. 타사 모니터링 UI에 액세스합니다.

- **alertmanager-main** 행에서 URL을 선택하여 Alertmanager UI에 대한 로그인 페이지를 엽니다.
- **grafana** 행에서 URL을 선택하여 Grafana UI의 로그인 페이지를 엽니다.
- **prometheus-k8s** 열에서 URL을 선택하여 Prometheus UI의 로그인 페이지를 엽니다.
- **thanos-querier** 행에서 URL을 선택하여 Thanos Querier UI의 로그인 페이지를 엽니다.

4. OpenShift Container Platform 인증 정보를 사용하여 로그인할 **OpenShift로 로그인**을 선택합니다.

7.2. CLI를 사용하여 타사 모니터링 UI에 액세스

OpenShift CLI(**oc**) 도구를 사용하여 Prometheus, Alertmanager 및 Grafana 웹 UI의 URL을 가져올 수 있습니다.

사전 요구 사항

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

프로세스

1. 다음을 실행하여 **openshift-monitoring** 프로젝트의 라우트를 나열합니다.

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

출력 예

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

2. 웹 브라우저를 사용하여 **HOST/PORT** 라우트로 이동합니다.
3. OpenShift 인증 정보를 사용하여 로그인하려면 **OpenShift를 사용하여 로그인**을 선택합니다.

중요

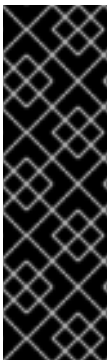
모니터링 라우트는 Cluster Monitoring Operator에 의해 관리되며 사용자가 수정할 수 없습니다.

7.3. 다음 단계

- 자동 스케일링을 위해 사용자 정의 애플리케이션 메트릭 노출

8장. 자동 스케일링을 위해 사용자 정의 애플리케이션 메트릭 노출

수평 Pod 자동 스케일러에 대한 사용자 정의 애플리케이션 메트릭을 내보낼 수 있습니다.



중요

Prometheus Adapter는 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 https://access.redhat.com/support/offerings/techpreview/를 참조하십시오.

8.1. 수평 POD 자동 스케일링을 위해 사용자 정의 애플리케이션 메트릭 노출

prometheus-adapter 리소스를 사용하여 수평 Pod 자동 스케일러에 대한 사용자 정의 애플리케이션 메트릭을 노출할 수 있습니다.

사전 요구 사항

- 사용자 정의 Prometheus 인스턴스가 배포 또는 **StatefulSet** 오브젝트에서 관리하는 Prometheus Pod로 설치되지만 Prometheus 사용자 정의 리소스(CR)에서는 설치되지 않습니다.
- 사용자 정의 **custom-prometheus** 프로젝트에 사용자 정의 Prometheus 인스턴스를 설치했습니다.

중요

OLM(Operator Lifecycle Manager)을 통해 설치된 사용자 정의 Prometheus 인스턴스 및 Prometheus Operator는 활성화된 경우 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.

- 사용자 정의 프로젝트에 애플리케이션 및 서비스를 배포했습니다. 이 예에서는 애플리케이션 및 서비스 모니터가 사용자 정의 **custom-prometheus** 프로젝트에 설치된 것으로 간주됩니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. 구성의 YAML 파일을 생성합니다. 이 예제에서는 파일을 **deploy.yaml**이라고 합니다.
2. **prometheus-adapter**에 대한 서비스 계정, 역할 및 역할 바인딩을 생성하기 위한 설정 세부 정보를 추가합니다.

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: custom-metrics-server-resources
rules:
- apiGroups:
  - custom.metrics.k8s.io
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: custom-metrics-resource-reader
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  - pods
  - services
  verbs:
  - get
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: custom-metrics:system:auth-delegator
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: custom-metrics-auth-reader
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: extension-apiserver-authentication-reader
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: custom-metrics-resource-reader
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: custom-metrics-resource-reader
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: hpa-controller-custom-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: custom-metrics-server-resources
subjects:
- kind: ServiceAccount
  name: horizontal-pod-autoscaler
  namespace: kube-system
---
```

3. **prometheus-adapter**의 사용자 정의 메트릭에 대한 설정 세부 정보를 추가합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: adapter-config
  namespace: custom-prometheus
data:
  config.yaml: |
    rules:
    - seriesQuery: 'http_requests_total{namespace!="",pod!=""}' (1)
      resources:
        overrides:
          namespace: {resource: "namespace"}
          pod: {resource: "pod"}
          service: {resource: "service"}
      name:
        matches: "^(.*)_total"
        as: "${1}_per_second" (2)
      metricsQuery: 'sum(rate(<<.Series>>{<<.LabelMatchers>>}[2m])) by (<<.GroupBy>>)'
---
```

(1) 선택한 메트릭을 HTTP 요청 수로 지정합니다.

(2) 메트릭의 빈도를 지정합니다.

4. **prometheus-adapter**를 API 서비스로 등록하기 위한 설정 세부 정보를 추가합니다.

```
apiVersion: v1
kind: Service
metadata:
```

```
  annotations:
    service.beta.openshift.io/serving-cert-secret-name: prometheus-adapter-tls
  labels:
    name: prometheus-adapter
  name: prometheus-adapter
  namespace: custom-prometheus
spec:
  ports:
  - name: https
    port: 443
    targetPort: 6443
  selector:
    app: prometheus-adapter
  type: ClusterIP
---
apiVersion: apiregistration.k8s.io/v1beta1
kind: APIService
metadata:
  name: v1beta1.custom.metrics.k8s.io
spec:
  service:
    name: prometheus-adapter
    namespace: custom-prometheus
  group: custom.metrics.k8s.io
  version: v1beta1
  insecureSkipTLSVerify: true
  groupPriorityMinimum: 100
  versionPriority: 100
---
```

5. Prometheus Adapter 이미지를 나열합니다.

```
$ oc get -n openshift-monitoring deploy/prometheus-adapter -o jsonpath="{..image}"
```

6. **prometheus-adapter** 배포에 대한 설정 세부 정보를 추가합니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-adapter
  name: prometheus-adapter
  namespace: custom-prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-adapter
  template:
    metadata:
      labels:
        app: prometheus-adapter
      name: prometheus-adapter
    spec:
      serviceAccountName: custom-metrics-apiserver
```

```
      containers:
      - name: prometheus-adapter
        image: quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:a46915a206cd7d97f240687c618dd59e8848fcc3a0f51e281f3384153a12c3e0
(1)
        args:
        - --secure-port=6443
        - --tls-cert-file=/var/run/serving-cert/tls.crt
        - --tls-private-key-file=/var/run/serving-cert/tls.key
        - --logtostderr=true
        - --prometheus-url=http://prometheus-operated.default.svc:9090/
        - --metrics-relist-interval=1m
        - --v=4
        - --config=/etc/adapter/config.yaml
        ports:
        - containerPort: 6443
        volumeMounts:
        - mountPath: /var/run/serving-cert
          name: volume-serving-cert
          readOnly: true
        - mountPath: /etc/adapter/
          name: config
          readOnly: true
        - mountPath: /tmp
          name: tmp-vol
      volumes:
      - name: volume-serving-cert
        secret:
          secretName: prometheus-adapter-tls
      - name: config
        configMap:
          name: adapter-config
      - name: tmp-vol
        emptyDir: {}
```

(1) 이전 단계에서 발견된 Prometheus Adapter 이미지를 지정합니다.

7. 클러스터에 구성을 적용합니다.

```
$ oc apply -f deploy.yaml
```

출력 예

```
serviceaccount/custom-metrics-apiserver created
clusterrole.rbac.authorization.k8s.io/custom-metrics-server-resources created
clusterrole.rbac.authorization.k8s.io/custom-metrics-resource-reader created
clusterrolebinding.rbac.authorization.k8s.io/custom-metrics:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/custom-metrics-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/custom-metrics-resource-reader created
clusterrolebinding.rbac.authorization.k8s.io/hpa-controller-custom-metrics created
configmap/adapter-config created
service/prometheus-adapter created
apiservice.apiregistration.k8s.io/v1.custom.metrics.k8s.io created
deployment.apps/prometheus-adapter created
```

8. 사용자 정의 프로젝트의 **prometheus-adapter** Pod가 **실행 중** 상태인지 확인합니다. 이 예에서 프로젝트는 **custom-prometheus**입니다.

```
$ oc -n custom-prometheus get pods prometheus-adapter-<string>
```

9. 이제 애플리케이션의 메트릭이 노출되어 수평 Pod 자동 스케일링을 구성하는 데 사용할 수 있습니다.

추가 리소스

- 수평 Pod 자동 스케일링 문서 참조
- 수평 Pod 자동 스케일러의 Kubernetes 문서 참조

8.2. 다음 단계

- 모니터링 문제 조사

9장. 모니터링 문제 조사

9.1. 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 UI에서 해당 메트릭을볼 수 없는 경우 이 프로세스에 설명된 단계를 수행하십시오.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 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 확인**합니다.
   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

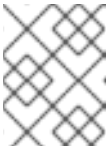
```
  labels:
    app: prometheus-example-app
```

   b. **ServiceMonitor** 리소스의 **matchLabels app** 라벨이 이전 단계의 라벨 출력과 일치하는지 확인합니다.

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

출력 예

```
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 Pod를 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-776fcbbd56-2nbfm**입니다.

   ```
   $ 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. Prometheus UI에서 **프로젝트의 대상 상태를** 직접 확인하십시오.

   a. **openshift-user-workload-monitoring** 프로젝트에서 Prometheus 인스턴스에 대한 포트 전달을 설정합니다.

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

   b. 웹 브라우저에서 http://localhost:9090/targets을 열고 Prometheus UI에서 직접 프로젝트의 대상 상태를 확인합니다. 대상과 관련된 오류 메시지를 확인합니다.

4. **openshift-user-workload-monitoring** 프로젝트에서 **Prometheus Operator의 디버그 수준 로깅을 구성**합니다.

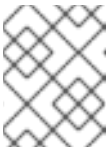
   a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 편집합니다.

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

b. **prometheusOperator**의 **logLevel:debug**를 **data / config.yaml** 아래에 추가하여 로그 수준을 **debug**로 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-operator**는 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"
```

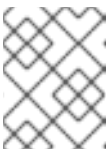
출력 예

```
        - --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 지정에서 참조하십시오.

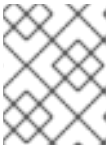
9.2. PROMETHEUS가 많은 디스크 공간을 소비하는 이유 확인

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 확인**합니다.
- 가장 많은 시계열을 생성하는 라벨에 대한 자세한 내용은 **Prometheus UI에서 시계열 데이터베이스(TSDB) 상태를 확인**합니다. 여기에는 클러스터 관리자 권한이 필요합니다.
- 사용자 정의 메트릭에 할당되는 바인딩되지 않은 속성의 수를 줄임으로써 **생성되는 고유의 시계열 수를 감소**합니다.



참고

사용 가능한 값의 제한된 집합에 바인딩되는 속성을 사용하면 가능한 키 - 값 쌍 조합의 수가 줄어듭니다.

- 사용자 정의 프로젝트에서 **스크랩할 수 있는 샘플 수를 제한**합니다. 여기에는 클러스터 관리자 권한이 필요합니다.

사전 요구 사항

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

프로세스

1. **Administrator** 관점에서 **Monitoring → Metrics**로 이동합니다..
2. **Expression** 필드에서 다음 Prometheus Query Language (PromQL) 쿼리를 실행합니다. 이렇게 하면 스크랩 샘플 수가 가장 많은 10개의 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. Prometheus UI에서 TSDB 상태를 확인합니다.
   a. **관리자** 관점에서 **네트워킹 → 라우트**로 이동합니다.
   b. **Project:** 목록에서 **openshift-monitoring** 프로젝트를 선택합니다.

c. **prometheus-k8s** 열에서 URL을 선택하여 Prometheus UI의 로그인 페이지를 엽니다.

d. OpenShift Container Platform 인증 정보를 사용하여 로그인할 **OpenShift로 로그인**을 선택합니다.

e. Prometheus UI에서 **상태 → TSDB 상태**로 이동합니다.

추가 리소스

- 스크랩 샘플 제한을 설정하고 관련 알림 규칙을 생성하는 방법에 대한 자세한 내용은 사용자 정의 프로젝트의 스크랩 샘플 제한 설정을 참조하십시오.
- 지원 케이스 제출