



Red Hat Advanced Cluster Management for Kubernetes 2.10

가시성

가시성

가시성

법적 공지

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

초록

관찰 기능 구성 요소를 활성화하여 관리 클러스터에 대한 통찰력을 얻을 수 있습니다.

차례

1장. 관찰 기능 서비스 도입	4
1.1. 관찰 기능 아키텍처	4
1.2. 관찰 기능 구성	7
2장. 관찰 기능 서비스 활성화	12
2.1. 사전 요구 사항	12
2.2. 명령줄 인터페이스에서 관찰 기능 활성화	13
2.3. RED HAT OPENSIFT CONTAINER PLATFORM 콘솔에서 관찰 기능 활성화	21
2.4. 관찰 기능 비활성화	21
2.5. 관찰 기능 제거	22
2.6. 추가 리소스	22
3장. 관찰 기능 사용	23
3.1. 관찰 기능 API를 사용하여 메트릭 쿼리	23
3.2. 메트릭을 외부 끝점으로 내보내기	23
3.3. 대시보드를 사용하여 데이터 보기 및 탐색	26
3.4. 추가 리소스	27
3.5. GRAFANA 대시보드 사용	27
4장. 관찰 기능 구성 사용자 정의	31
4.1. 사용자 정의 규칙 생성	31
4.2. 사용자 정의 메트릭 추가	32
4.3. 보존을 위한 고급 구성 추가	34
4.4. 단일 노드 OPENSIFT 클러스터에 대한 동적 메트릭	35
4.5. 콘솔에서 MULTICLUSTEROBSERVABILITY 사용자 정의 리소스 복제본 업데이트	37
4.6. 영구 볼륨 및 영구 볼륨 클레임 증가 및 감소	37
4.7. 경로 인증서 사용자 정의	38
4.8. 오브젝트 저장소에 액세스하기 위한 인증서 사용자 정의	38
4.9. 관찰 기능 애드온에 대한 프록시 설정 구성	40
4.10. 사전 요구 사항	40
4.11. 관찰 기능 애드온에 대한 프록시 설정 비활성화	41
4.12. 추가 리소스	41
5장. 관찰 기능 경고 관리	43
5.1. ALERTMANAGER 구성	43
5.2. 경고 전달	44
5.3. 음소거 경고	44
5.4. 경고 비활성화	46
5.5. 추가 리소스	47
6장. GRAFANA에서 관리되는 클러스터 레이블 사용	48
6.1. 관리형 클러스터 레이블 추가	49
6.2. 관리형 클러스터 라벨 활성화	49
6.3. 관리형 클러스터 레이블 비활성화	50
6.4. 추가 리소스	50
7장. 콘솔 소개에서 검색	52
7.1. 구성 요소 검색	52
7.2. 사용자 정의 및 구성 검색	53
7.3. 추가 리소스	56
7.4. 검색 관리	56
8장. RED HAT INSIGHTS에서 관찰 기능 사용	61

8.1. 사전 요구 사항	61
8.2. RED HAT ADVANCED CLUSTER MANAGEMENT 콘솔의 RED HAT INSIGHTS	61
8.3. 인사이트 정책 보고서 관리	62

1장. 관찰 기능 서비스 도입

관찰 기능 구성 요소는 플릿 전체의 클러스터의 상태 및 사용률을 이해하는 데 사용할 수 있는 서비스입니다. 기본적으로 다중 클러스터 관찰 기능 Operator는 Red Hat Advanced Cluster Management를 설치하는 동안 활성화됩니다.

관찰 구성 요소에 대한 자세한 내용은 다음 문서를 참조하십시오.

- [관찰 기능 아키텍처](#)
- [관찰 기능 구성](#)
- [관찰 기능 서비스 활성화](#)
- [관찰 기능 사용](#)
- [관찰 기능 사용자 정의](#)
- [관찰 기능 경고](#)
- [콘솔 소개에서 검색](#)
- [Red Hat Insights에서 관찰 기능 사용](#)

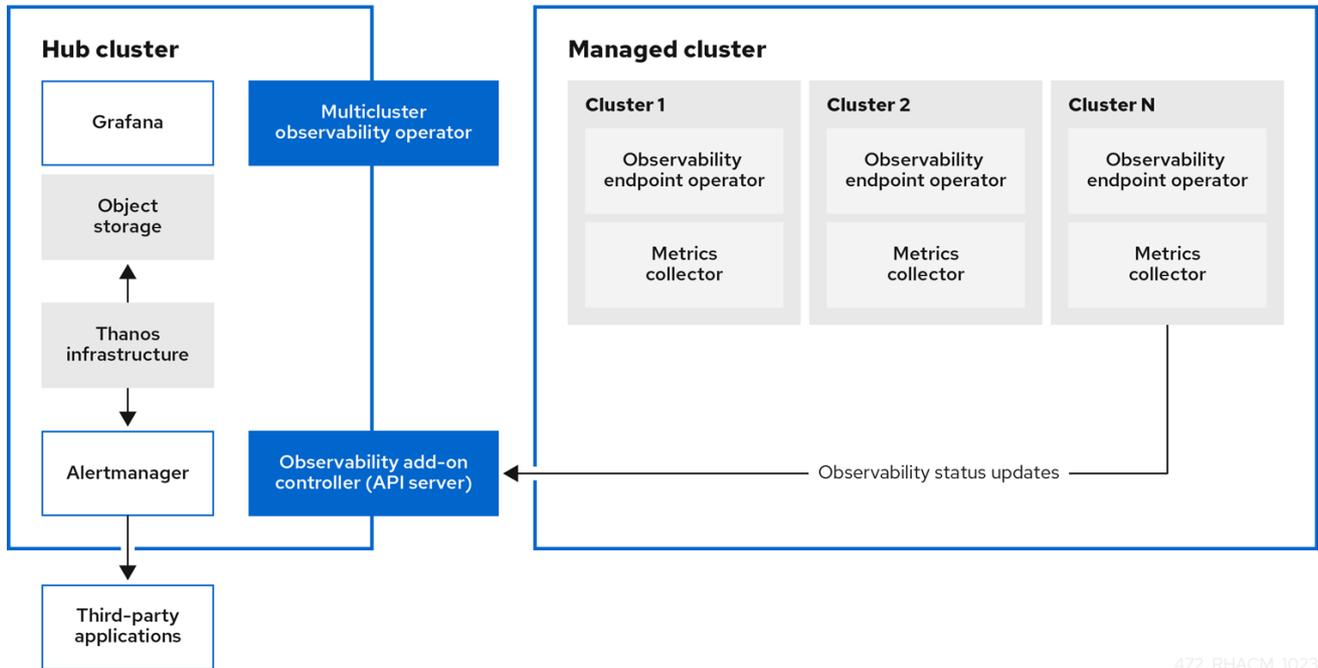
1.1. 관찰 기능 아키텍처

multiclusterhub-operator 는 기본적으로 **multicluster-observability-operator** Pod를 활성화합니다. **multicluster-observability-operator** Pod를 구성해야 합니다.

MultiClusterObservability 사용자 정의 리소스를 정의하여 Observability 서비스를 활성화하면 **observability-endpoint-operator** 가 각 가져오기 또는 생성된 관리형 클러스터에 자동으로 배포됩니다. 이 컨트롤러는 Red Hat OpenShift Container Platform Prometheus에서 데이터를 수집하는 메트릭 수집기를 시작한 다음 해당 데이터를 Red Hat Advanced Cluster Management Hub 클러스터로 전송합니다.

Observability 서비스가 활성화되면 hub 클러스터는 hub self-management가 활성화되어 있는지 여부와 관계없이 구성된 Thanos 인스턴스에 지표를 수집 및 전송하도록 항상 구성됩니다. 허브 클러스터가 자체 관리되면 **disableHubSelfManagement** 매개변수가 기본 설정인 **false** 로 설정됩니다. hub 클러스터에 대한 지표 및 경고가 **local-cluster** 네임스페이스에 표시됩니다. hub self-management가 활성화된 경우 **local-cluster** 는 클러스터 목록 드롭다운 메뉴에만 표시됩니다. Grafana Explorer에서 **local-cluster** 메트릭을 쿼리할 수 있습니다.

다음 다이어그램은 관찰 기능의 구성 요소를 보여줍니다.



472_RHACM_1023

관찰 기능 아키텍처의 구성 요소에는 다음 항목이 포함됩니다.

- **multiclusterhub-operator** Pod라고도 하는 다중 클러스터 허브 Operator는 **multicluster-observability-operator** Pod를 배포합니다. 허브 클러스터 데이터를 관리 클러스터에 보냅니다.
- **관찰 기능 애드온 컨트롤러**는 관리 클러스터의 로그를 자동으로 업데이트하는 API 서버입니다.
- Thanos 인프라에는 **multicluster-observability-operator** Pod에서 배포한 Thanos Cryostat가 포함되어 있습니다. Thanos Cryostat는 보존 구성을 사용하고 스토리지의 데이터를 압축하여 쿼리가 제대로 수행되도록 합니다. Thanos Cryostat에 문제가 발생할 때 이를 식별하는 데 도움이 되도록 상태를 모니터링하는 4개의 기본 경고를 사용합니다. 다음 기본 경고 테이블을 읽습니다.

표 1.1. 기본 Thanos 경고 테이블

경고	심각도	설명
ACMThanosCompactHaltd	심각	compactor가 중지되면 경고가 전송됩니다.
ACMThanosCompactHighCompactionFailures	경고	압축 실패 비율이 5%보다 크면 경고가 전송됩니다.
ACMThanosCompactBucketHighOperationFailures	경고	버킷 작업 실패 비율이 5%보다 크면 경고가 전송됩니다.
ACMThanosCompactHasNotRun	경고	압축기가 지난 24시간 동안 아무것도 업로드하지 않으면 경고가 전송됩니다.

- 관찰 기능 구성 요소는 **Grafana** 인스턴스를 배포하여 대시보드(static) 또는 데이터 무관을 사용하여 데이터 시각화를 활성화합니다. Red Hat Advanced Cluster Management는 Grafana의 8.5.20 버전을 지원합니다. Grafana 대시보드를 설계할 수도 있습니다. 자세한 내용은 **Grafana 대**

시보드 설계를 참조하십시오.

- *Prometheus Alertmanager* 를 사용하면 타사 애플리케이션과 함께 경고를 전달할 수 있습니다. 사용자 정의 레코딩 규칙 또는 경고 규칙을 생성하여 관찰 기능 서비스를 사용자 지정할 수 있습니다. Red Hat Advanced Cluster Management는 Prometheus Alertmanager의 버전 0.25를 지원합니다.

1.1.1. observability 서비스에 사용되는 영구 저장소

중요: 영구 스토리지에 로컬 볼륨을 사용하는 로컬 스토리지 Operator 또는 스토리지 클래스를 사용하지 마십시오. 재시작 후 Pod가 다른 노드에서 다시 시작되면 데이터가 손실될 수 있습니다. 이 경우 Pod는 더 이상 노드의 로컬 스토리지에 액세스할 수 없습니다. 데이터 손실을 방지하기 위해 수신 및 규칙 포드의 영구 볼륨에 액세스할 수 있는지 확인합니다.

Red Hat Advanced Cluster Management를 설치할 때 PVC(영구 볼륨 클레임)가 자동으로 연결할 수 있도록 다음과 같은 PV(영구 볼륨)를 생성해야 합니다. 기본 스토리지 클래스가 지정되지 않았거나 기본 스토리지 클래스를 사용하여 PV를 호스팅하려는 경우 **MultiClusterObservability** 사용자 정의 리소스에 스토리지 클래스를 정의해야 합니다. Prometheus에서 사용하는 것과 유사하게 Block Storage를 사용하는 것이 좋습니다. 또한 **alertmanager, thanos-compact, thanos-ruler, thanos-receive-default** 및 **thanos-store-shard** 의 각 복제본에는 자체 PV가 있어야 합니다. 다음 표를 참조하십시오.

표 1.2. 영구 볼륨의 테이블 목록

영구 볼륨 이름	목적
alertmanager	Alertmanager는 nflog 데이터를 저장하고 해당 스토리지에 음소거된 경고를 저장합니다. nflog 는 알림을 받는 수신자와 함께 활성 및 해결된 알림의 추가 전용 로그 및 알림이 식별된 콘텐츠의 해시 다이제스트입니다.
Thanos-compact	compact에는 처리를 위한 중간 데이터와 버킷 상태 캐시를 저장하기 위해 로컬 디스크 공간이 필요합니다. 필요한 공간은 기본 블록의 크기에 따라 다릅니다. compact는 모든 소스 블록을 다운로드할 수 있는 충분한 공간이 있어야 하고 디스크에 압축된 블록을 빌드해야 합니다. 디스크상의 데이터는 재시작 중에 삭제할 수 있으며 크래시 루프 콤팩트러를 가장 먼저 시도해야 합니다. 그러나 다시 시작 사이에 버킷 상태 캐시를 효과적으로 사용하려면 compact 영구 디스크를 제공하는 것이 좋습니다.
Thanos-rule	thanos 룰러는 고정된 간격으로 쿼리를 실행하여 선택한 쿼리 API에 대해 Prometheus 기록 및 경고 규칙을 평가합니다. 규칙 결과는 Prometheus 2.0 스토리지 형식으로 디스크에 다시 작성됩니다. 이 상태 저장 세트에 유지되는 시간 또는 일수의 데이터는 API 버전 observability.open-cluster-management.io/v1beta1 에서 수정되었습니다. observability.open-cluster-management.io/v1beta2:RetentionInLocal 에서 API 매개변수로 노출되었습니다.

Thanos-receive-default	<p>Thanos receiver는 수신 데이터(Prometheus 원격 쓰기 요청)를 수락하고 이를 Prometheus TSDB의 로컬 인스턴스에 씁니다. 주기적으로 (각 2시간), TSDB 블록은 장기 저장 및 압축을 위해 오브젝트 스토리지에 업로드됩니다. 로컬 캐시를 작동하는 이 상태 저장 세트에 유지되는 시간 또는 데이터의 양은 API 버전 observability.open-cluster-management.io/v1beta 에서 수정되었습니다. observability.open-cluster-management.io/v1beta2:RetentionInLocal에서 API 매개변수로 노출되었습니다.</p>
Thanos-store-shard	<p>기본적으로 API 게이트웨이 역할을 하므로 상당한 양의 로컬 디스크 공간이 필요하지 않습니다. 시작 시 Thanos 클러스터에 가입하고 액세스할 수 있는 데이터를 알립니다. 로컬 디스크의 모든 원격 블록에 대한 약간의 정보를 유지하고 버킷과 동기화되도록 유지합니다. 이 데이터는 일반적으로 시작 시간이 늘어남에 따라 다시 시작할 때마다 삭제하는 것이 안전합니다.</p>

참고: 시계열 기록 데이터는 오브젝트 저장소에 저장됩니다. Thanos는 오브젝트 스토리지를 지표 및 메타데이터의 기본 스토리지로 사용합니다. 오브젝트 스토리지 및 다운샘플링에 대한 자세한 내용은 [관찰 기능 활성화](#)를 참조하십시오.

1.1.2. 추가 리소스

- [Observability 서비스 소개](#) 참조
- [Observability 구성](#) 참조
- [관찰 기능 서비스 활성화](#)를 참조하십시오.

1.2. 관찰 기능 구성

관찰 기능 준수를 통해 수집할 수 있는 메트릭과 관찰 기능 Pod 용량에 대한 정보를 계속 읽고 이를 읽습니다.

1.2.1. 메트릭 유형

기본적으로 OpenShift Container Platform은 Telemetry 서비스를 사용하여 지표를 Red Hat으로 보냅니다. **acm_managed_cluster_info** 는 Red Hat Advanced Cluster Management에서 사용할 수 있으며 Telemetry에 포함되어 있지만 Red Hat Advanced Cluster Management Observe *환경 개요* 대시보드에는 표시되지 않습니다.

프레임워크에서 지원하는 다음 메트릭 유형을 확인합니다.

표 1.3. 매개변수 테이블

메트릭 이름	메트릭 유형	레이블/태그	상태
acm_managed_cluster_info	게이지	hub_cluster_id,managed_cluster_id,vendor,클라우드,버전,created_via,core_worker,socket_worker	안정적
config_policies_evaluation_duration_seconds_bucket	히스토그램	없음	안정적입니다. 자세한 내용은 <i>Governance 메트릭</i> 을 참조하십시오.
config_policies_evaluation_duration_seconds_count	히스토그램	없음	안정적입니다. 자세한 내용은 <i>Governance 메트릭</i> 을 참조하십시오.
config_policies_evaluation_duration_seconds_sum	히스토그램	없음	안정적입니다. 자세한 내용은 <i>Governance 메트릭</i> 을 참조하십시오.
policy_governance_info	게이지	type,policy,policy_namespace,cluster_namespace	안정적입니다. 자세한 내용은 <i>Governance 메트릭</i> 을 검토하십시오.
policyreport_info	게이지	managed_cluster_id,카테고리,정책,결과,심각도	안정적입니다. 자세한 내용은 <i>Managing insight _PolicyReports_</i> 를 참조하십시오.
search_api_db_connection_failed_total	카운터	없음	안정적입니다. <i>콘솔 소개 설명서의 구성 요소 검색</i> 섹션을 참조하십시오.
search_api_dbquery_duration_seconds	히스토그램	없음	안정적입니다. <i>콘솔 소개 설명서의 구성 요소 검색</i> 섹션을 참조하십시오.
search_api_requests	히스토그램	없음	안정적입니다. <i>콘솔 소개 설명서의 구성 요소 검색</i> 섹션을 참조하십시오.
search_indexer_request_count	카운터	없음	안정적입니다. <i>콘솔 소개 설명서의 구성 요소 검색</i> 섹션을 참조하십시오.

메트릭 이름	메트릭 유형	레이블/태그	상태
search_indexer_request_duration	히스토그램	없음	안정적입니다. 콘솔 소개 설명서의 구성 요소 검색 섹션을 참조하십시오.
search_indexer_requests_in_flight	게이지	없음	안정적입니다. 콘솔 소개 설명서의 구성 요소 검색 섹션을 참조하십시오.
search_indexer_request_size	히스토그램	없음	안정적입니다. 콘솔 소개 설명서의 구성 요소 검색 섹션을 참조하십시오.

1.2.2. 관찰 기능 Pod 용량 요청

관찰 기능 구성 요소에는 관찰 서비스를 설치하기 위해 Cryostat1mCPU 및 11972Mi 메모리가 필요합니다. 다음 표는 **observability-addons** 가 활성화된 5개의 관리 클러스터에 대한 Pod 용량 요청 목록입니다.

표 1.4. 관찰 기능 Pod 용량 요청

Deployment 또는 StatefulSet	컨테이너 이름	CPU (mCPU)	메모리(Mi)	replicas	Pod 총 CPU	Pod 총 메모리
observability-alertmanager	alertmanager	4	200	3	12	600
	config-reloader	4	25	3	12	75
	alertmanager-proxy	1	20	3	3	60
observability-grafana	grafana	4	100	2	8	200
	grafana-dashboard-loader	4	50	2	8	100
observability-observatorium-api	observatorium-api	20	128	2	40	256

Deployment 또는 StatefulSet	컨테이너 이름	CPU (mCPU)	메모리(Mi)	replicas	Pod 총 CPU	Pod 총 메모리
observability-observatorium-operator	observatorium-operator	100	100	1	10	50
observability-rbac-query-proxy	rbac-query-proxy	20	100	2	40	200
	oauth-proxy	1	20	2	2	40
observability-thanos-compact	Thanos-compact	100	512	1	100	512
observability-thanos-query	Thanos-query	300	1024	2	600	2048
observability-thanos-query-frontend	Thanos-query-frontend	100	256	2	200	512
observability-thanos-query-frontend-memcached	Memcached	45	128	3	135	384
	내보내기	5	50	3	15	150
observability-thanos-receive-controller	thanos-receive-controller	4	32	1	4	32
observability-thanos-receive-default	Thanos-receive	300	512	3	900	1536
observability-thanos-rule	Thanos-rule	50	512	3	150	1536
	configmap-reloader	4	25	3	12	75

Deployment 또는 StatefulSet	컨테이너 이름	CPU (mCPU)	메모리(Mi)	replicas	Pod 총 CPU	Pod 총 메모리
observability-thanos-store-memcached	Memcached	45	128	3	135	384
	내보내기	5	50	3	15	150
observability-thanos-store-shard	Thanos-store	100	1024	3	300	3072

1.2.3. 추가 리소스

- 관찰 기능 활성화에 대한 자세한 내용은 [관찰 기능 서비스 활성화를 참조하십시오.](#)
- 관찰 기능 [사용자 지정](#)을 읽고 관찰 기능을 구성하고 메트릭 및 기타 데이터를 보는 방법을 알아봅니다.
- [Grafana 대시보드](#)를 사용하여 읽습니다.
- OpenShift Container Platform 설명서에서 [Telemetry](#)를 사용하여 수집 및 전송되는 메트릭 유형을 설명합니다. 자세한 내용은 [Telemetry에서 수집하는 정보](#)를 참조하십시오.
- 자세한 내용은 [Governance 메트릭](#) 을 참조하십시오.
- Manage [insight PolicyReports](#)를 참조하십시오.
- [Prometheus 레코딩 규칙](#)을 참조하십시오.
- [Prometheus 경고 규칙](#) 도 참조하십시오.
- [Observability 서비스 소개](#) 로 돌아갑니다.

2장. 관찰 기능 서비스 활성화

허브 클러스터에서 관찰 기능 서비스를 활성화하면 **multicluster-observability-operator** 는 새 관리 클러스터를 감시하고 메트릭 및 경고 수집 서비스를 관리 클러스터에 자동으로 배포합니다. 메트릭을 사용하고 Grafana 대시보드를 구성하여 클러스터 리소스 정보를 표시하고 비용을 절감하며 서비스 중단을 방지할 수 있습니다.

multicluster-observability-operator Pod라고도 하는 관찰 기능 구성 요소를 사용하여 관리 클러스터의 상태를 모니터링합니다.

필수 액세스: 클러스터 관리자, **open-cluster-management:cluster-manager-admin** 역할 또는 S3 관리자.

- [사전 요구 사항](#)
- [명령줄 인터페이스에서 관찰 기능 활성화](#)
- [MultiClusterObservability 사용자 정의 리소스 생성](#)
- [Red Hat OpenShift Container Platform 콘솔에서 관찰 기능 활성화](#)
- [관찰 기능 비활성화](#)
- [관찰 기능 제거](#)

2.1. 사전 요구 사항

- Kubernetes용 Red Hat Advanced Cluster Management를 설치해야 합니다. 자세한 내용은 [온라인에 연결된 동안](#) 설치를 참조하십시오.
- 기본 스토리지 클래스가 지정되지 않은 경우 **MultiClusterObservability** 사용자 정의 리소스에 스토리지 클래스를 정의해야 합니다.
- 허브 클러스터에 대한 직접 네트워크 액세스가 필요합니다. 로드 밸런서 및 프록시에 대한 네트워크 액세스는 지원되지 않습니다. 자세한 내용은 [네트워킹](#) 을 참조하십시오.
- 스토리지 솔루션을 생성하려면 오브젝트 저장소를 구성해야 합니다.
 - **중요:** 오브젝트 저장소를 구성할 때 민감한 데이터가 유지될 때 필요한 암호화 요구 사항을 충족하는지 확인하십시오. 관찰 기능 서비스는 Thanos에서 지원되는 안정적인 오브젝트 저장소를 사용합니다. 여러 Red Hat Advanced Cluster Management 관찰 기능을 통해 오브젝트 저장소 버킷을 공유하지 못할 수 있습니다. 따라서 각 설치에 대해 별도의 오브젝트 저장소 버킷을 제공합니다.
 - Red Hat Advanced Cluster Management는 안정적인 오브젝트 저장소를 갖춘 다음과 같은 클라우드 공급자를 지원합니다.
 - Amazon Web Services S3 (AWS S3)
 - Red Hat Ceph (S3 호환 API)
 - Google Cloud Storage
 - Azure 스토리지
 - Red Hat OpenShift Data Foundation (이전 Red Hat OpenShift Container Storage)

- Red Hat OpenShift on IBM (ROKS)

2.2. 명령줄 인터페이스에서 관찰 기능 활성화

MultiClusterObservability 사용자 정의 리소스 인스턴스를 생성하여 관찰 기능 서비스를 활성화합니다. 관찰 기능을 활성화하기 전에 자세한 내용은 [Observability Pod 용량 요청](#)을 참조하십시오.

참고:

- Red Hat Advanced Cluster Management에서 관리하는 OpenShift Container Platform 관리 클러스터에서 관찰 기능이 활성화되거나 비활성화되면 observability 엔드포인트 Operator는 로컬 Prometheus를 자동으로 다시 시작하는 **alertmanager** 구성을 추가하여 **cluster-monitoring-config** 구성 맵을 업데이트합니다.
- observability 엔드포인트 Operator는 로컬 Prometheus를 자동으로 재시작하는 **alertmanager** 구성을 추가하여 **cluster-monitoring-config** 구성 맵을 업데이트합니다. OpenShift Container Platform 관리 클러스터에 **alertmanager** 구성을 삽입하면 구성이 Prometheus 메트릭의 retention 필드와 관련된 설정을 제거합니다.

observability 서비스를 활성화하려면 다음 단계를 완료합니다.

1. Red Hat Advanced Cluster Management Hub 클러스터에 로그인합니다.
2. 다음 명령을 사용하여 observability 서비스의 네임스페이스를 생성합니다.

```
oc create namespace open-cluster-management-observability
```

3. pull-secret을 생성합니다. Red Hat Advanced Cluster Management가 **open-cluster-management** 네임스페이스에 설치된 경우 다음 명령을 실행합니다.

```
DOCKER_CONFIG_JSON=`oc extract secret/multiclusterhub-operator-pull-secret -n open-cluster-management --to=-`
```

multiclusterhub-operator-pull-secret 이 네임스페이스에 정의되지 않은 경우 **openshift-config** 네임스페이스의 **pull-secret** 을 **open-cluster-management-observability** 네임스페이스로 복사합니다. 다음 명령을 실행합니다.

```
DOCKER_CONFIG_JSON=`oc extract secret/pull-secret -n openshift-config --to=-`
```

그런 다음 **open-cluster-management-observability** 네임스페이스에 pull-secret을 생성하고 다음 명령을 실행합니다.

```
oc create secret generic multiclusterhub-operator-pull-secret \
  -n open-cluster-management-observability \
  --from-literal=.dockerconfigjson="$DOCKER_CONFIG_JSON" \
  --type=kubernetes.io/dockerconfigjson
```

중요: OpenShift Container Platform 설명서를 사용하여 클러스터의 글로벌 풀 시크릿을 수정하는 경우 관찰 가능 네임스페이스에서 글로벌 풀 시크릿을 업데이트해야 합니다. 자세한 내용은 [글로벌 풀 시크릿](#) 업데이트를 참조하십시오.

4. 클라우드 공급자의 오브젝트 스토리지에 대한 시크릿을 생성합니다. 시크릿에는 스토리지 솔루션에 대한 인증 정보가 포함되어야 합니다. 예를 들어 다음 명령을 실행합니다.

```
oc create -f thanos-object-storage.yaml -n open-cluster-management-observability
```

지원되는 오브젝트 저장소에 대한 다음 보안 예제를 확인합니다.

- Amazon S3 또는 S3 호환의 경우 시크릿은 다음 파일과 유사할 수 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |
    type: s3
    config:
      bucket: YOUR_S3_BUCKET
      endpoint: YOUR_S3_ENDPOINT 1
      insecure: true
      access_key: YOUR_ACCESS_KEY
      secret_key: YOUR_SECRET_KEY
```

- 1** 프로토콜 없이 URL을 입력합니다. Amazon S3 끝점의 URL을 입력하십시오. **s3.us-east-1.amazonaws.com**.

자세한 내용은 [Amazon Simple Storage Service 사용자 가이드](#)를 참조하십시오 .

- Google Cloud Platform의 경우 시크릿은 다음 파일과 유사할 수 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |
    type: GCS
    config:
      bucket: YOUR_GCS_BUCKET
      service_account: YOUR_SERVICE_ACCOUNT
```

자세한 내용은 [Google Cloud Storage](#) 를 참조하십시오.

- Azure의 경우 시크릿은 다음 파일과 유사할 수 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |
```

```

type: AZURE
config:
  storage_account: YOUR_STORAGE_ACCT
  storage_account_key: YOUR_STORAGE_KEY
  container: YOUR_CONTAINER
  endpoint: blob.core.windows.net ❶
  max_retries: 0

```

- ❶ **msi_resource** 경로를 사용하는 경우 시스템에 할당된 관리 ID를 사용하여 엔드포인트 인증이 완료됩니다. 값은 <https://<storage-account-name>.blob.core.windows.net> 끝점과 유사해야 합니다.

user_assigned_id 경로를 사용하는 경우 사용자가 할당한 관리 ID를 사용하여 엔드포인트 인증이 완료됩니다. **user_assigned_id** 를 사용하는 경우 **msi_resource** 끝점 기본값은 **https:<storage_account>.<endpoint >**입니다. 자세한 내용은 [Azure Storage 설명서를 참조하십시오.](#)

참고: Azure를 Red Hat OpenShift Container Platform 클러스터의 오브젝트 스토리지로 사용하는 경우 클러스터와 연결된 스토리지 계정이 지원되지 않습니다. 새 스토리지 계정을 생성해야 합니다.

- Red Hat OpenShift Data Foundation의 경우 시크릿은 다음 파일과 유사할 수 있습니다.

```

apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |
    type: s3
    config:
      bucket: YOUR_RH_DATA_FOUNDATION_BUCKET
      endpoint: YOUR_RH_DATA_FOUNDATION_ENDPOINT ❶
      insecure: false
      access_key: YOUR_RH_DATA_FOUNDATION_ACCESS_KEY
      secret_key: YOUR_RH_DATA_FOUNDATION_SECRET_KEY

```

- ❶ 프로토콜 없이 URL을 입력합니다. 다음 URL과 유사한 Red Hat OpenShift Data Foundation 끝점의 URL을 입력합니다. **example.redhat.com:443**.

자세한 내용은 [Red Hat OpenShift Data Foundation](#) 을 참조하십시오.

- Red Hat OpenShift on IBM(ROKS)의 경우 시크릿이 다음 파일과 유사할 수 있습니다.

```

apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |

```

```

type: s3
config:
  bucket: YOUR_ROKS_S3_BUCKET
  endpoint: YOUR_ROKS_S3_ENDPOINT ❶
  insecure: true
  access_key: YOUR_ROKS_ACCESS_KEY
  secret_key: YOUR_ROKS_SECRET_KEY

```

- ❶ 프로토콜 없이 URL을 입력합니다. 다음 URL과 유사한 Red Hat OpenShift Data Foundation 끝점의 URL을 입력합니다. **example.redhat.com:443**.

자세한 내용은 IBM Cloud 설명서인 [Cloud Object Storage](#) 를 참조하십시오. 서비스 자격 증명을 사용하여 오브젝트 스토리지와 연결하십시오. 자세한 내용은 IBM Cloud 문서, [Cloud Object Store](#) 및 [서비스 인증](#) 정보를 참조하십시오.

2.2.1. AWS 보안 토큰 서비스 스토리지 구성

Amazon S3 또는 S3 호환 스토리지의 경우 AWS STS(Security Token Service)로 생성된 단기적이고 제한된 권한 인증 정보를 사용할 수도 있습니다. 자세한 내용은 [AWS Security Token Service 설명서](#) 를 참조하십시오.

AWS Security Service를 사용하여 액세스 키를 생성하려면 다음과 같은 추가 단계가 필요합니다.

1. S3 버킷에 대한 액세스를 제한하는 IAM 정책을 생성합니다.
2. 신뢰 정책으로 IAM 역할을 생성하여 OpenShift Container Platform 서비스 계정에 대한 JWT 토큰을 생성합니다.
3. S3 버킷에 액세스해야 하는 관찰 기능 서비스 계정에 대한 주석을 지정합니다. 환경 설정 단계에서 AWS(ROSA)의 Red Hat OpenShift Service를 AWS STS 토큰으로 작동하도록 구성할 수 있는 방법의 예를 확인할 수 있습니다. 자세한 내용은 ROSA와 함께 [Red Hat OpenShift Service on AWS \(ROSA\) 에서 STS와 함께](#) 요구 사항 및 STS 토큰을 사용하도록 설정에 대한 자세한 내용을 참조하십시오.

2.2.2. AWS 보안 서비스를 사용하여 액세스 키 생성

AWS Security Service를 사용하여 액세스 키를 생성하려면 다음 단계를 완료합니다.

1. AWS 환경을 설정합니다. 다음 명령을 실행합니다.

```

export POLICY_VERSION=$(date +"%m-%d-%y")
export TRUST_POLICY_VERSION=$(date +"%m-%d-%y")
export CLUSTER_NAME=<my-cluster>
export S3_BUCKET=$CLUSTER_NAME-acm-observability
export REGION=us-east-2
export NAMESPACE=open-cluster-management-observability
export SA=tbdl
export SCRATCH_DIR=/tmp/scratch
export OIDC_PROVIDER=$(oc get authentication.config.openshift.io cluster -o json | jq -r
.spec.serviceAccountIssuer| sed -e "s/^https:////")
export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
export AWS_PAGER=""
rm -rf $SCRATCH_DIR
mkdir -p $SCRATCH_DIR

```

2. 다음 명령을 사용하여 S3 버킷을 생성합니다.

```
aws s3 mb s3://$S3_BUCKET
```

3. S3 버킷에 액세스할 수 있도록 **s3-policy** JSON 파일을 생성합니다. 다음 명령을 실행합니다.

```
{
  "Version": "$POLICY_VERSION",
  "Statement": [
    {
      "Sid": "Statement",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:CreateBucket",
        "s3:DeleteBucket"
      ],
      "Resource": [
        "arn:aws:s3:::$S3_BUCKET/*",
        "arn:aws:s3:::$S3_BUCKET"
      ]
    }
  ]
}
```

4. 다음 명령을 사용하여 정책을 적용합니다.

```
S3_POLICY=$(aws iam create-policy --policy-name $CLUSTER_NAME-acm-obs \
--policy-document file://$SCRATCH_DIR/s3-policy.json \
--query 'Policy.Arn' --output text)
echo $S3_POLICY
```

5. **TrustPolicy** JSON 파일을 생성합니다. 다음 명령을 실행합니다.

```
{
  "Version": "$TRUST_POLICY_VERSION",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/${OIDC_PROVIDER}"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "${OIDC_PROVIDER}:sub": [
            "system:serviceaccount:${NAMESPACE}:observability-thanos-query",
            "system:serviceaccount:${NAMESPACE}:observability-thanos-store-shard",
            "system:serviceaccount:${NAMESPACE}:observability-thanos-compact",
            "system:serviceaccount:${NAMESPACE}:observability-thanos-rule",
            "system:serviceaccount:${NAMESPACE}:observability-thanos-receive",

```

```

    ]
  }
}
]
}

```

6. 다음 명령을 사용하여 AWS Prometheus 및 CloudMonitor에 대한 역할을 생성합니다.

```

S3_ROLE=$(aws iam create-role \
  --role-name "$CLUSTER_NAME-acm-obs-s3" \
  --assume-role-policy-document file://$SCRATCH_DIR/TrustPolicy.json \
  --query "Role.Arn" --output text)
echo $S3_ROLE

```

7. 역할에 정책을 연결합니다. 다음 명령을 실행합니다.

```

aws iam attach-role-policy \
  --role-name "$CLUSTER_NAME-acm-obs-s3" \
  --policy-arn $S3_POLICY

```

시크릿은 다음 파일과 유사할 수 있습니다. **config** 섹션은 **signature_version2: false** 를 지정하고 **access_key** 및 **secret_key**:을 지정하지 않습니다.

```

apiVersion: v1
kind: Secret
metadata:
  name: thanos-object-storage
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  thanos.yaml: |
    type: s3
    config:
      bucket: $S3_BUCKET
      endpoint: s3.$REGION.amazonaws.com
      signature_version2: false

```

8. MultiClusterObservability 사용자 정의 리소스 섹션에 설명된 대로 **MultiClusterObservability** 사용자 정의 리소스 를 사용할 때 서비스 계정 주석을 지정합니다.
9. 다음 명령을 사용하여 클라우드 공급자의 S3 액세스 키 및 시크릿 키를 검색할 수 있습니다. **base64** 문자열을 시크릿에 디코딩, 편집 및 인코딩해야 합니다.

```

YOUR_CLOUD_PROVIDER_ACCESS_KEY=$(oc -n open-cluster-management-observability get secret <object-storage-secret> -o jsonpath="{.data.thanos\.yaml}" | base64 -decode | grep access_key | awk '{print $2}')

```

```

echo $ACCESS_KEY

```

```

YOUR_CLOUD_PROVIDER_SECRET_KEY=$(oc -n open-cluster-management-observability get secret <object-storage-secret> -o jsonpath="{.data.thanos\.yaml}" | base64 -decode | grep secret_key | awk '{print $2}')

```

```

echo $SECRET_KEY

```

- 10. Pod에서 다음 배포 및 상태 저장 세트가 있는지 확인하여 관찰 기능이 활성화되어 있는지 확인합니다. 다음 정보를 받을 수 있습니다.

```

observability-thanos-query (deployment)
observability-thanos-compact (statefulset)
observability-thanos-receive-default (statefulset)
observability-thanos-rule (statefulset)
observability-thanos-store-shard-x (statefulsets)

```

2.2.3. MultiClusterObservability 사용자 정의 리소스 생성

MultiClusterObservability 사용자 정의 리소스를 사용하여 다양한 구성 요소의 영구 볼륨 스토리지 크기를 지정합니다. **MultiClusterObservability** 사용자 정의 리소스를 처음 생성하는 동안 스토리지 크기를 설정해야 합니다. 스토리지 크기 값을 배포 후 업데이트할 때 스토리지 클래스가 동적 볼륨 확장을 지원하는 경우에만 변경 사항이 적용됩니다. 자세한 내용은 [Red Hat OpenShift Container Platform 설명서에서 영구 볼륨 확장을 참조하십시오](#).

hub 클러스터에서 **MultiClusterObservability** 사용자 정의 리소스를 생성하려면 다음 단계를 완료합니다.

1. **multiclusterobservability_cr.yaml** 이라는 **MultiClusterObservability** 사용자 정의 리소스 YAML 파일을 생성합니다.
관찰을 위해 다음과 같은 기본 YAML 파일을 확인합니다.

```

apiVersion: observability.open-cluster-management.io/v1beta2
kind: MultiClusterObservability
metadata:
  name: observability
spec:
  observabilityAddonSpec: {}
  storageConfig:
    metricObjectStorage:
      name: thanos-object-storage
      key: thanos.yaml

```

advanced 섹션에서 **retentionConfig** 매개변수 값을 수정할 수 있습니다. 자세한 내용은 [Thanos Downsampling resolution and retention](#) 에서 참조하십시오. 관리 클러스터 수에 따라 상태 저장 세트의 스토리지 양을 업데이트할 수 있습니다. S3 버킷이 STS 토큰을 사용하도록 구성된 경우 S3 역할과 함께 STS를 사용하도록 서비스 계정에 주석을 둡니다. 다음 구성을 확인합니다.

```

spec:
  advanced:
    compact:
      serviceAccountAnnotations:
        eks.amazonaws.com/role-arn: $S3_ROLE
    store:
      serviceAccountAnnotations:
        eks.amazonaws.com/role-arn: $S3_ROLE
    rule:
      serviceAccountAnnotations:
        eks.amazonaws.com/role-arn: $S3_ROLE
  receive:
    serviceAccountAnnotations:

```

```
eks.amazonaws.com/role-arn: $S3_ROLE
query:
serviceAccountAnnotations:
eks.amazonaws.com/role-arn: $S3_ROLE
```

자세한 내용은 [Observability API](#) 를 참조하십시오.

- 인프라 머신 세트에 배포하려면 **MultiClusterObservability** YAML에서 **nodeSelector** 를 업데이트하여 설정된 레이블을 설정해야 합니다. YAML은 다음 콘텐츠와 유사할 수 있습니다.

```
nodeSelector:
node-role.kubernetes.io/infra:
```

자세한 내용은 [인프라 머신 세트 생성](#) 을 참조하십시오.

- 다음 명령을 실행하여 관찰 기능 YAML을 클러스터에 적용합니다.

```
oc apply -f multiclusterobservability_cr.yaml
```

Thanos, Grafana 및 Alertmanager의 **open-cluster-management-observability** 네임스페이스의 모든 Pod가 생성됩니다. Red Hat Advanced Cluster Management Hub 클러스터에 연결된 모든 관리형 클러스터는 지표를 Red Hat Advanced Cluster Management Observability 서비스로 다시 보낼 수 있습니다.

- Grafana 대시보드를 시작하여 observability 서비스가 활성화되고 데이터가 입력되었는지 확인합니다.
- 콘솔 개요 페이지 또는 *Clusters* 페이지에서 콘솔 헤더 옆에 있는 **Grafana 링크**를 클릭합니다.
 - 또는 [https://\\$ACM_URL/grafana/dashboards](https://$ACM_URL/grafana/dashboards) 를 사용하여 OpenShift Container Platform 3.11 Grafana 대시보드에 액세스합니다.
 - OpenShift Container Platform 3.11 대시 보드를 보려면 *OCP 3.11* 이라는 폴더를 선택합니다.
- multicluster-observability-operator** 배포에 액세스하여 **multicluster-observability-operator** Pod가 **multiclusterhub-operator** 배포에 의해 배포되고 있는지 확인합니다. 다음 명령을 실행합니다.

```
oc get deploy multicluster-observability-operator -n open-cluster-management --show-labels

NAME                                READY UP-TO-DATE AVAILABLE AGE LABELS
multicluster-observability-operator 1/1   1         1         35m
installer.name=multiclusterhub,installer.namespace=open-cluster-management
```

- 리소스와 연결된 **라벨** 에 대한 **multicluster-observability-operator** 배포의 labels 섹션을 확인합니다. **labels** 섹션에는 다음 세부 정보가 포함될 수 있습니다.

```
labels:
installer.name: multiclusterhub
installer.namespace: open-cluster-management
```

.. 선택 사항: 특정 관리 클러스터에서 관찰 데이터 수집에서 제외하려면 **observability: disabled** 라는 클러스터에 다음 클러스터 레이블을 추가합니다.

observability 서비스가 활성화되어 있습니다. 관찰 기능 서비스를 활성화하면 다음 기능이 시작됩니다.

- 관리 클러스터의 모든 경고 관리자는 Red Hat Advanced Cluster Management Hub 클러스터로 전달됩니다.
- Red Hat Advanced Cluster Management Hub 클러스터에 연결된 모든 관리형 클러스터는 Red Hat Advanced Cluster Management observability 서비스로 경고를 다시 보낼 수 있습니다. 이메일, PagerDuty 또는 OpsGenie와 같은 올바른 수신자 통합으로 경고를 중복, 그룹화 및 라우팅하도록 Red Hat Advanced Cluster Management Alertmanager를 구성할 수 있습니다. 또한 경고의 실크 및 억제를 처리할 수 있습니다.

참고: Red Hat Advanced Cluster Management Hub 클러스터 기능에 대한 경고는 Red Hat OpenShift Container Platform 버전 4.13 이상이 있는 관리형 클러스터에서만 지원됩니다. 관찰 기능이 활성화된 Red Hat Advanced Cluster Management를 설치하면 OpenShift Container Platform 4.13 이상에서 경고가 자동으로 hub 클러스터로 전달됩니다. 자세한 내용은 [경고 전달을 참조하십시오](#).

2.3. RED HAT OPENSIFT CONTAINER PLATFORM 콘솔에서 관찰 기능 활성화

선택적으로 Red Hat OpenShift Container Platform 콘솔에서 관찰 기능을 활성화하고 **open-cluster-management-observability** 라는 프로젝트를 생성할 수 있습니다. **open-cluster-management-observability** 프로젝트에서 **multiclusterhub-operator-pull-secret** 이라는 이미지 풀 시크릿을 생성해야 합니다.

open-cluster-management-observability 프로젝트에서 **thanos-object-storage** 라는 오브젝트 스토리지 시크릿을 생성합니다. 오브젝트 스토리지 시크릿 세부 정보를 입력하고 **생성** 을 클릭합니다. 시크릿 예제를 보려면 *Enabling observability* 섹션의 4단계를 참조하십시오.

MultiClusterObservability 사용자 정의 리소스 인스턴스를 생성합니다. 다음 메시지가 표시되면 OpenShift Container Platform에서 관찰 기능 서비스가 성공적으로 활성화됩니다. **Observability 구성 요소가 배포되고 실행 중입니다.**

2.3.1. Thanos 버전 확인

Thanos가 클러스터에 배포된 후 CLI(명령줄 인터페이스)에서 Thanos 버전을 확인합니다.

hub 클러스터에 로그인한 후 관찰 가능 Pod에서 다음 명령을 실행하여 Thanos 버전을 수신합니다.

```
thanos --version
```

Thanos 버전이 표시됩니다.

2.4. 관찰 기능 비활성화

Red Hat Advanced Cluster Management Hub 클러스터에서 데이터 수집을 중지하는 관찰 기능을 비활성화할 수 있습니다.

2.4.1. 모든 클러스터에서 관찰 기능 비활성화

모든 관리 클러스터에서 관찰 가능 구성 요소를 제거하여 관찰 기능을 비활성화합니다. **enableMetrics** 를 **false** 로 설정하여 **multicluster-observability-operator** 리소스를 업데이트합니다. 업데이트된 리소스는 다음 변경 사항과 유사할 수 있습니다.

```
spec:
  imagePullPolicy: Always
```

```
imagePullSecret: multiclusterhub-operator-pull-secret
observabilityAddonSpec: # The ObservabilityAddonSpec defines the global settings for all managed
clusters which have observability add-on enabled
enableMetrics: false #indicates the observability addon push metrics to hub server
```

2.4.2. 단일 클러스터에서 관찰 기능 비활성화

특정 관리 클러스터에서 관찰 구성 요소를 제거하여 관찰 기능을 비활성화합니다.

managedclusters.cluster.open-cluster-management.io 사용자 정의 리소스에 **observability: disabled** 레이블을 추가합니다. Red Hat Advanced Cluster Management 콘솔 *Clusters* 페이지에서 **observability=disabled** 라벨을 지정된 클러스터에 추가합니다.

참고: 관찰 기능 구성 요소가 있는 관리 클러스터가 분리되면 **메트릭 수집** 배포가 제거됩니다.

2.5. 관찰 기능 제거

MultiClusterObservability 사용자 정의 리소스를 제거하면 관찰 기능 서비스를 비활성화하고 설치 제거합니다. OpenShift Container Platform 콘솔 탐색에서 **Operator > 설치된 Operator > Advanced Cluster Manager for Kubernetes** 를 선택합니다. **MultiClusterObservability** 사용자 정의 리소스를 제거합니다.

2.6. 추가 리소스

- 오브젝트 스토리지 정보에 대한 클라우드 공급자 설명서 링크:
 - [Amazon Web Services S3 \(AWS S3\)](#)
 - [Red Hat Ceph \(S3 호환 API\)](#)
 - [Google Cloud Storage](#)
 - [Azure 스토리지](#)
 - [Red Hat OpenShift Data Foundation \(이전 Red Hat OpenShift Container Storage\)](#)
 - [Red Hat OpenShift on IBM \(ROKS\)](#)
- [관찰 기능 사용](#)을 참조하십시오.
- 관찰 기능 서비스 사용자 지정에 대한 자세한 내용은 [사용자 지정 관찰 기능](#)을 참조하십시오.
- 더 많은 관련 주제를 보려면 [Observability 서비스 도입](#)으로 돌아가십시오.

3장. 관찰 기능 사용

관찰 기능 서비스를 사용하여 플릿의 클러스터 사용률을 확인합니다.

- [관찰 기능 API를 사용하여 메트릭 쿼리](#)
- [메트릭을 외부 끝점으로 내보내기](#)
- [데이터 보기 및 탐색](#)

3.1. 관찰 기능 API를 사용하여 메트릭 쿼리

관찰 기능은 OpenShift 경로 **rbac-query-proxy** 를 통해 쿼리할 메트릭에 대한 외부 API를 제공합니다. **rbac-query-proxy** 경로에 대한 쿼리를 가져오려면 다음 옵션을 참조하십시오.

- 다음 명령을 사용하여 경로 세부 정보를 가져올 수 있습니다.

```
oc get route rbac-query-proxy -n open-cluster-management-observability
```

- OpenShift OAuth 액세스 토큰을 사용하여 **rbac-query-proxy** 경로에 액세스할 수도 있습니다. 토큰은 네임스페이스를 가져올 수 있는 권한이 있는 사용자 또는 서비스 계정과 연결되어야 합니다. 자세한 내용은 [사용자 소유 OAuth 액세스 토큰 관리](#)를 참조하십시오.

관찰을 위해 **proxy-byo-cert** 시크릿을 생성하려면 다음 단계를 완료합니다.

1. 기본 CA 인증서를 가져오고 키 **tls.crt** 의 콘텐츠를 로컬 파일에 저장합니다. 다음 명령을 실행합니다.

```
oc -n openshift-ingress get secret router-certs-default -o jsonpath="{.data.tls\.crt}" | base64 -d > ca.crt
```

2. 다음 명령을 실행하여 메트릭을 쿼리합니다.

```
curl --cacert ./ca.crt -H "Authorization: Bearer {TOKEN}" https://{PROXY_ROUTE_URL}/api/v1/query?query={QUERY_EXPRESSION}
```

참고: **QUERY_EXPRESSION** 은 표준 Prometheus 쿼리 표현식입니다. 예를 들어 이전에 언급한 명령의 URL을 https://{PROXY_ROUTE_URL}/api/v1/query?query=cluster_infrastructure_provider 로 교체하여 메트릭 **cluster_infrastructure_provider** 를 쿼리합니다. 자세한 내용은 [Prometheus 쿼리](#)를 참조하십시오.

3. 생성된 인증서를 사용하여 **proxy-byo-ca** 시크릿을 생성하려면 다음 명령을 실행합니다.

```
oc -n open-cluster-management-observability create secret tls proxy-byo-ca --cert ./ca.crt --key ./ca.key
```

4. 다음 명령을 사용하여 생성된 인증서를 사용하여 **proxy-byo-cert** 시크릿을 생성합니다.

```
oc -n open-cluster-management-observability create secret tls proxy-byo-cert --cert ./ingress.crt --key ./ingress.key
```

3.2. 메트릭을 외부 끝점으로 내보내기

Prometheus Remote-Write 사양을 실시간으로 지원하는 외부 끝점으로 지표를 내보냅니다. 메트릭을 외부 끝점으로 내보내려면 다음 단계를 완료합니다.

1. **open-cluster-management-observability** 네임스페이스에서 외부 끝점의 액세스 정보를 사용하여 외부 끝점에 대한 Kubernetes 시크릿을 생성합니다. 다음 예제 시크릿을 확인합니다.

```

apiVersion: v1
kind: Secret
metadata:
  name: victoriametrics
  namespace: open-cluster-management-observability
type: Opaque
stringData:
  ep.yaml: |
    url: http://victoriametrics:8428/api/v1/write
    http_client_config:
      basic_auth:
        username: test
        password: test
    
```

ep.yaml 은 콘텐츠의 키이며 다음 단계의 **MultiClusterObservability** 사용자 정의 리소스에서 사용됩니다. 현재 관찰 기능은 기본 인증 또는 **tls** 활성화를 사용하여 보안 검사 없이 엔드포인트로 메트릭을 내보낼 수 있습니다. 지원되는 매개변수의 전체 목록은 다음 표를 확인하십시오.

이름	설명	스키마
URL <i>필요</i>	외부 엔드포인트의 URL입니다.	string
http_client_c onfig <i>optional</i>	HTTP 클라이언트를 위한 고급 구성입니다.	HttpClientConfig

HttpClientConfig

이름	설명	스키마
basic_auth <i>선택 사항</i>	기본 인증을 위한 HTTP 클라이언트 구성	basicauth
tls_config <i>optional</i>	TLS에 대한 HTTP 클라이언트 구성	TLSConfig

basicauth

이름	설명	스키마
사용자 이름 <i>선택 사항</i>	기본 승인을 위한 사용자 이름입니다.	string

이름	설명	스키마
암호 선택 사항	기본 승인을 위한 암호입니다.	string

TLSConfig

이름	설명	스키마
secret_name <i>required</i>	인증서가 포함된 보안의 이름입니다.	string
ca_file_key <i>optional</i>	시크릿의 CA 인증서 키(insecure_skip_verify 가 true로 설정된 경우에만 <i>선택 사항</i>)	string
cert_file_key <i>required</i>	시크릿에 있는 클라이언트 인증서의 키입니다.	string
key_file_key <i>필요</i>	시크릿에 있는 클라이언트 키의 키입니다.	string
insecure_skip_verify <i>optional</i>	대상 인증서에 대한 확인을 건너 뛰는 매개변수입니다.	bool

- 내보낼 외부 엔드 포인트 목록을 추가하기 위해 **MultiClusterObservability** 사용자 정의 리소스에 **writeStorage** 매개변수를 추가합니다. 다음 예제를 확인합니다.

```
spec:
  storageConfig:
    writeStorage: ❶
    - key: ep.yaml
      name: victoriametrics
```

- ❶ 각 항목에는 **name** 과 **key** 의 두 가지 속성이 있습니다. **name**은 엔드포인트 액세스 정보가 포함된 Kubernetes 시크릿의 이름이며 **key** 는 시크릿에 있는 콘텐츠의 키입니다. 목록에 항목을 두 개 이상 추가하면 메트릭이 여러 외부 엔드포인트로 내보내집니다.

- acm_remote_write_requests_total** 메트릭을 확인하여 메트릭 내보내기를 활성화한 후 지표 내보내기의 상태를 확인합니다.
 - hub 클러스터의 OpenShift Container Platform 콘솔에서 Observe 섹션에서 Metrics 를 클릭하여 **Metrics** 페이지로 이동합니다.
 - 그런 다음 **acm_remote_write_requests_total** 메트릭을 쿼리합니다. 해당 메트릭의 값은 하나의 관찰자 API 인스턴스에서 하나의 외부 끝점에 대한 특정 응답이 있는 총 요청 수입니다. **name** 레이블은 외부 끝점의 이름입니다. 코드 레이블은 메트릭 내보내기에 대한 HTTP 요청의 반환 코드입니다.

3.3. 대시보드를 사용하여 데이터 보기 및 탐색

hub 클러스터에서 Grafana에 액세스하여 관리 클러스터의 데이터를 확인합니다. 특정 경고를 쿼리하고 쿼리에 대한 필터를 추가할 수 있습니다.

예를 들어 단일 노드 OpenShift 클러스터에서 `cluster_infrastructure_provider` 경고를 탐색하려면 `cluster_infrastructure_provider{clusterType="SNO"}` 쿼리 표현식을 사용합니다.

참고: 단일 노드 관리 클러스터에서 관찰 기능이 활성화된 경우 `ObservabilitySpec.resources.CPU.limits` 매개변수를 설정하지 마십시오. CPU 제한을 설정하면 관리 클러스터의 용량에 대해 관찰 가능 Pod가 계산됩니다. 추가 리소스 섹션에서 Management Workload Partitioning에 대한 참조를 참조하십시오.

3.3.1. 기록 데이터 보기

기록 데이터를 쿼리할 때 쿼리 매개변수 옵션을 수동으로 설정하여 대시보드에서 표시되는 데이터 양을 제어합니다. 다음 단계를 완료합니다.

1. 허브 클러스터에서 콘솔 헤더에 있는 **Grafana 링크**를 선택합니다.
2. **패널 편집을 선택하여 클러스터 대시보드를 편집합니다**
3. Grafana의 쿼리 프론트 엔드 데이터 소스에서 쿼리 탭을 클릭합니다.
4. **\$datasource** 를 선택합니다.
5. 더 많은 데이터를 보려면 Step 매개변수 섹션의 값을 늘립니다. Step 매개변수 섹션이 비어 있으면 자동으로 계산됩니다.
6. Custom query parameters 필드를 찾아 **max_source_resolution=auto** 를 선택합니다.
7. 데이터가 표시되는지 확인하려면 Grafana 페이지를 새로 고칩니다.

쿼리 데이터는 Grafana 대시보드에서 표시됩니다.

3.3.2. Red Hat Advanced Cluster Management 대시보드 보기

Red Hat Advanced Cluster Management observability 서비스를 활성화하면 대시보드 세 개를 사용할 수 있습니다. 다음 대시보드 설명:

- **경고 분석:** 관리 클러스터 플릿 내에서 생성되는 경고의 개요입니다.
- **Alert별 클러스터:** 경고 이름으로 필터링할 수 있는 경고 대시보드입니다.
- **클러스터별 경고 대시보드:** 클러스터별로 필터링하고 클러스터 환경 내에서 시작되거나 보류 중인 경고에 대한 실시간 데이터를 볼 수 있습니다.

3.3.3. etcd 테이블 보기

Grafana의 허브 클러스터 대시보드에서 etcd 테이블을 보고 데이터 저장소로 etcd의 안정성을 확인할 수도 있습니다. hub 클러스터에서 Grafana 링크를 선택하여 hub 클러스터에서 수집된 etcd 테이블 데이터를 확인합니다. 관리 클러스터에서 리더 선택 변경 사항이 표시됩니다.

3.3.4. Kubernetes API 서버 대시보드 보기

Kubernetes API 서버 대시보드를 보려면 다음 옵션을 확인합니다.

- Grafana의 허브 클러스터 대시보드에서 클러스터 플릿 Kubernetes API 서비스 수준 개요를 확인합니다.
 1. Grafana 대시보드로 이동합니다.
 2. **Kubernetes > 서비스 수준 개요 > API 서버** 를 선택하여 관리 대시보드 메뉴에 액세스합니다. Fleet 개요 및 상위 클러스터 세부 정보가 표시됩니다. 지난 7일 또는 30일 동안 대상 서비스 수준 목표 (SLO) 값을 초과하거나 충족하는 총 클러스터 수, 차단 및 비활성 클러스터, API 서버 요청 기간이 표시됩니다.
- Grafana의 허브 클러스터 대시보드에서 Kubernetes API 서비스 수준 개요 테이블을 확인합니다.
 1. hub 클러스터에서 Grafana 대시보드로 이동합니다.
 2. **Kubernetes > 서비스 수준 개요 > API 서버** 를 선택하여 관리 대시보드 메뉴에 액세스합니다. Fleet 개요 및 상위 클러스터 세부 정보가 표시됩니다. 지난 7일 또는 30일 동안의 오류 예산으로 나머지 다운타임 및 추세가 표시됩니다.

3.4. 추가 리소스

- 자세한 내용은 [Prometheus Remote-Write 사양](#)을 참조하십시오.
- [관찰 기능 서비스 활성화](#)를 읽습니다.
- 더 많은 주제를 보려면 [Observability 서비스 소개](#) 로 돌아가십시오.

3.5. GRAFANA 대시보드 사용

Grafana 대시보드를 사용하여 허브 클러스터 및 관리 클러스터 지표를 확인합니다. Grafana 경고 대시보드에 표시되는 데이터는 관리 클러스터에서 발생하는 **경고 메트릭**을 사용합니다. **경고 메트릭**은 hub 클러스터의 Red Hat Advanced Cluster Management 경고 관리자에게 경고를 전달하는 관리 클러스터에 영향을 미치지 않습니다. 따라서 지표 및 경고에는 별도의 전파 메커니즘이 있으며 별도의 코드 경로를 따릅니다.

Grafana 경고 대시보드에 데이터가 표시되면 관리 클러스터 경고가 Red Hat Advanced Cluster Management Hub 클러스터 경고 관리자로 성공적으로 전달된다고 보장하지 않습니다. 메트릭이 관리 클러스터에서 전파되는 경우 Grafana 경고 대시보드에 표시된 데이터를 볼 수 있습니다.

개발 요구에 Grafana 대시보드를 사용하려면 다음을 완료합니다.

- [Grafana 개발자 인스턴스 설정](#)
- [Grafana 대시보드 설계](#)
- [Grafana 개발자 인스턴스 설치 제거](#)

3.5.1. Grafana 개발자 인스턴스 설정

grafana-dev 인스턴스를 생성하여 Grafana 대시보드를 설계할 수 있습니다. 최신 **grafana-dev** 인스턴스를 사용해야 합니다.

Grafana 개발자 인스턴스를 설정하려면 다음 단계를 완료합니다.

1. **툴 폴더에 있는 스크립트를 실행할 수 있도록 `open-cluster-management/multicluster-observability-operator/` 리포지토리를 복제합니다.**
2. **`setup-grafana-dev.sh` 를 실행하여 Grafana 인스턴스를 설정합니다. 스크립트를 실행하면 `secret/grafana-dev-config,deployment.apps/grafana-dev,service/grafana-dev,ingress.extensions/grafana-dev,persistentvolumeclaim/grafana-dev`.**

```
./setup-grafana-dev.sh --deploy
secret/grafana-dev-config created
deployment.apps/grafana-dev created
service/grafana-dev created
serviceaccount/grafana-dev created
clusterrolebinding.rbac.authorization.k8s.io/open-cluster-management:grafana-crb-dev
created
route.route.openshift.io/grafana-dev created
persistentvolumeclaim/grafana-dev created
oauthclient.oauth.openshift.io/grafana-proxy-client-dev created
deployment.apps/grafana-dev patched
service/grafana-dev patched
route.route.openshift.io/grafana-dev patched
oauthclient.oauth.openshift.io/grafana-proxy-client-dev patched
clusterrolebinding.rbac.authorization.k8s.io/open-cluster-management:grafana-crb-dev
patched
```

3. **`switch-to-grafana-admin.sh` 스크립트를 사용하여 사용자 역할을 Grafana 관리자로 전환합니다.**
 - a. Grafana URL, **`https://grafana-dev-open-cluster-management-observability.{OPENSIFT_INGRESS_DOMAIN}`** 을 선택하고 로그인합니다.
 - b. 그런 다음 다음 명령을 실행하여 전환된 사용자를 Grafana 관리자로 추가합니다. 예를 들어 **`kubeadmin`** 을 사용하여 로그인한 후 다음 명령을 실행합니다.

```
./switch-to-grafana-admin.sh kube:admin
User <kube:admin> switched to be grafana admin
```

Grafana 개발자 인스턴스가 설정됩니다.

3.5.1.1. Grafana 버전 확인

CLI(명령줄 인터페이스) 또는 Grafana 사용자 인터페이스에서 Grafana 버전을 확인합니다.

hub 클러스터에 로그인한 후 **`observability-grafana`** Pod 터미널에 액세스합니다. 다음 명령을 실행합니다.

```
grafana-cli
```

현재 클러스터 환경 내에 배포된 Grafana 버전이 표시됩니다.

또는 Grafana 대시보드에서 Manage (관리) 탭으로 이동할 수 있습니다. 버전이 나열된 페이지 끝으로 스크롤합니다.

3.5.2. Grafana 대시보드 설계

Grafana 인스턴스를 설정한 후 대시보드를 설계할 수 있습니다. Grafana 콘솔을 새로 고치고 대시보드를 설계하려면 다음 단계를 완료합니다.

1. Grafana 콘솔에서 탐색 패널에서 **Create** 아이콘을 선택하여 대시보드를 만듭니다. 대시보드를 선택한 다음 **새 패널 추가**를 클릭합니다.
2. 새 대시보드/편집 패널 에서 쿼리 탭으로 이동합니다.
3. 데이터 소스 선택기에서 **Observatorium** 을 선택하고 PromQL 쿼리를 입력하여 쿼리를 구성합니다.
4. Grafana 대시보드 헤더에서 대시보드 헤더에 있는 **저장** 아이콘을 클릭합니다.
5. 설명이 포함된 이름을 추가하고 **저장**을 클릭합니다.

3.5.2.1. ConfigMap 을 사용하여 Grafana 대시보드 설계

ConfigMap을 사용하여 Grafana 대시보드를 설계합니다. **generate-dashboard-configmap-yaml.sh** 스크립트를 사용하여 대시보드 ConfigMap을 생성하고 ConfigMap을 로컬에 저장할 수 있습니다.

```
./generate-dashboard-configmap-yaml.sh "Your Dashboard Name"
Save dashboard <your-dashboard-name> to ./your-dashboard-name.yaml
```

이전에 언급한 스크립트를 실행할 수 있는 권한이 없는 경우 다음 단계를 완료합니다.

1. 대시보드를 선택하고 **대시보드 설정** 아이콘을 클릭합니다.
2. 탐색 패널에서 **JSON 모델** 아이콘을 클릭합니다.
3. 대시보드 JSON 데이터를 복사하여 **data** 섹션에 붙여넣습니다.
4. 이름을 수정하고 **\$your-dashboard-name** 을 바꿉니다. **data.\$your-dashboard-name.json.\$your_dashboard_json.json** 의 **uid** 필드에 UUID(Universally unique identifier)를 입력합니다. **uidegen** 과 같은 프로그램을 사용하여 UUID를 생성할 수 있습니다. ConfigMap은 다음 파일과 유사할 수 있습니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: $your-dashboard-name
  namespace: open-cluster-management-observability
  labels:
    grafana-custom-dashboard: "true"
data:
  $your-dashboard-name.json: |-
    $your_dashboard_json
```

참고:

- 대시보드가 **grafana-dev** 인스턴스 내에 생성되는 경우 대시보드 이름을 가져와서 스크립트에서 인수로 전달할 수 있습니다. 예를 들어 Demo Dashboard 라는 대시보드는 **grafana-dev** 인스턴스에 생성됩니다. CLI에서 다음 스크립트를 실행할 수 있습니다.

```
./generate-dashboard-configmap-yaml.sh "Demo Dashboard"
```

스크립트를 실행한 후 다음 메시지가 표시될 수 있습니다.

```
Save dashboard <demo-dashboard> to ./demo-dashboard.yaml
```

- 대시보드가 일반 폴더에 없는 경우 이 ConfigMap의 **annotations** 섹션에 폴더 이름을 지정할 수 있습니다.

```
annotations:
  observability.open-cluster-management.io/dashboard-folder: Custom
```

ConfigMap에 대한 업데이트를 완료한 후 이를 설치하여 대시보드를 Grafana 인스턴스로 가져올 수 있습니다.

CLI 또는 OpenShift Container Platform 콘솔에서 YAML을 적용하여 YAML 파일이 생성되었는지 확인합니다. **open-cluster-management-observability** 네임스페이스 내의 ConfigMap이 생성됩니다. CLI에서 다음 명령을 실행합니다.

```
oc apply -f demo-dashboard.yaml
```

OpenShift Container Platform 콘솔에서 **demo-dashboard.yaml** 파일을 사용하여 ConfigMap을 생성합니다. 대시보드는 사용자 지정 폴더에 있습니다.

3.5.3. Grafana 개발자 인스턴스 설치 제거

인스턴스를 제거하면 관련 리소스도 삭제됩니다. 다음 명령을 실행합니다.

```
./setup-grafana-dev.sh --clean
secret "grafana-dev-config" deleted
deployment.apps "grafana-dev" deleted
serviceaccount "grafana-dev" deleted
route.route.openshift.io "grafana-dev" deleted
persistentvolumeclaim "grafana-dev" deleted
oauthclient.oauth.openshift.io "grafana-proxy-client-dev" deleted
clusterrolebinding.rbac.authorization.k8s.io "open-cluster-management:grafana-crb-dev" deleted
```

3.5.4. 추가 리소스

- [외부 엔드포인트로 메트릭 내보내기를 참조하십시오.](#)
- UUID를 생성하는 방법은 [uudegen](#) 을 참조하십시오.
- 자세한 내용은 [Grafana에서 관리되는 클러스터 레이블 사용](#)을 참조하십시오.
- [Grafana 대시보드를 사용하여 페이지 시작 부분으로 돌아갑니다.](#)
- 자세한 내용은 [Observing 환경 소개](#) 를 참조하십시오.

4장. 관찰 기능 구성 사용자 정의

관찰 기능을 활성화한 후 환경의 특정 요구에 맞게 관찰 기능 구성을 사용자 지정합니다.

관찰 서비스에서 수집하는 클러스터 플릿 데이터를 관리하고 확인하는 방법에 대한 자세한 내용은 다음 섹션을 참조하십시오.

필수 액세스: 클러스터 관리자

- 사용자 정의 규칙 생성
- 사용자 정의 메트릭 추가
- 보존을 위한 고급 구성 추가
- 콘솔에서 MultiClusterObservability 사용자 정의 리소스 복제본 업데이트
- 영구 볼륨 및 영구 볼륨 클레임 증가 및 감소
- 경로 인증 사용자 정의
- 오브젝트 저장소에 액세스하기 위한 인증서 사용자 정의
- 관찰 기능 애드온에 대한 프록시 설정 구성
- 관찰 기능 애드온에 대한 프록시 설정 비활성화

4.1. 사용자 정의 규칙 생성

관찰 가능성 리소스에 Prometheus 기록 규칙 및 경고 규칙을 추가하여 관찰 기능 설치에 대한 사용자 정의 규칙을 생성합니다.

비용이 많이 드는 표현식을 미리 계산하려면 레코딩 규칙 기능을 사용합니다. 결과는 새로운 시계열로 저장됩니다. 경고 규칙을 사용하면 경고를 외부 서비스로 보내는 방법에 따라 경고 조건을 지정할 수 있습니다.

참고: 사용자 지정 규칙을 업데이트하면 **observability-thanos-rule** Pod가 자동으로 다시 시작됩니다.

Prometheus를 사용하여 사용자 정의 규칙을 정의하여 경고 조건을 생성하고 외부 메시징 서비스에 알림을 보냅니다. 사용자 지정 규칙의 다음 예제를 확인합니다.

- 사용자 정의 경고 규칙을 만듭니다. **open-cluster-management-observability** 네임스페이스에 **thanos-ruler-custom-rules** 라는 구성 맵을 생성합니다. 다음 예와 같이 키 이름을 **custom_rules.yaml** 로 지정해야 합니다. 구성에 여러 규칙을 생성할 수 있습니다.
 - CPU 사용량이 정의된 값을 전달할 때 알리는 사용자 지정 경고 규칙을 만듭니다. YAML은 다음 콘텐츠와 유사할 수 있습니다.

```
data:
  custom_rules.yaml: |
    groups:
      - name: cluster-health
        rules:
          - alert: ClusterCPUHealth-jb
            annotations:
              summary: Notify when CPU utilization on a cluster is greater than the defined
```

```
utilization limit
  description: "The cluster has a high CPU usage: {{ $value }} core for {{
$labels.cluster }} {{ $labels.clusterID }}."
  expr: |
    max(cluster:cpu_usage_cores:sum) by (clusterID, cluster, prometheus) > 0
  for: 5s
  labels:
    cluster: "{{ $labels.cluster }}"
    prometheus: "{{ $labels.prometheus }}"
    severity: critical
```

- 기본 경고 규칙은 **open-cluster-management-observability** 네임스페이스의 **thanos-ruler-default-rules** 구성 맵에 있습니다.
- **thanos-ruler-custom-rules** 구성 맵에 사용자 지정 레코딩 규칙을 생성합니다. Pod의 컨테이너 메모리 캐시 합계를 가져오는 기능을 제공하는 레코딩 규칙을 생성합니다. YAML은 다음 콘텐츠와 유사할 수 있습니다.

```
data:
  custom_rules.yaml: |
    groups:
      - name: container-memory
        rules:
          - record: pod:container_memory_cache:sum
            expr: sum(container_memory_cache{pod!=""}) BY (pod, container)
```

참고: 구성 맵을 변경한 후 구성이 자동으로 다시 로드됩니다. **observability-thanos-ruler** 사이트 카 내의 **config-reload** 로 인해 구성이 다시 로드됩니다.

- 경고 규칙이 올바르게 작동하는지 확인하려면 Grafana 대시보드로 이동하여 **탐색** 페이지를 선택하고 **ALERTS** 를 쿼리합니다. 경고를 생성한 경우에만 Grafana에서 경고를 사용할 수 있습니다.

4.2. 사용자 정의 메트릭 추가

모든 관리 클러스터에서 메트릭 수집을 시작하려면 **metrics_list.yaml** 파일에 메트릭을 추가합니다. 다음 단계를 완료합니다.

1. 사용자 지정 지표를 추가하기 전에 다음 명령을 사용하여 **mco observability** 가 활성화되어 있는지 확인합니다.

```
oc get mco observability -o yaml
```

2. **status.conditions.message** 섹션에서 다음 메시지가 표시되는지 확인합니다.

```
Observability components are deployed and running
```

3. 다음 명령을 사용하여 **open-cluster-management-observability** 네임스페이스에 **observability-metrics-custom-allowlist** 구성 맵을 생성합니다.

```
oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml
```

4. 사용자 지정 지표의 이름을 **metrics_list.yaml** 매개변수에 추가합니다. 구성 맵의 YAML은 다음 내용과 유사할 수 있습니다.

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: observability-metrics-custom-allowlist
data:
  metrics_list.yaml: |
    names: ❶
    - node_memory_MemTotal_bytes
    rules: ❷
    - record: apiserver_request_duration_seconds:histogram_quantile_90
      expr:
        histogram_quantile(0.90,sum(rate(apiserver_request_duration_seconds_bucket{job="apiserver",
        verb!="WATCH"}[5m])) by (verb,le))

```

- ❶ **선택 사항:** 관리 클러스터에서 수집할 사용자 정의 메트릭의 이름을 추가합니다.
- ❷ **선택 사항:** **expr** 및 **record** 매개변수 쌍에 대해 하나의 값만 입력하여 쿼리 표현식을 정의합니다. 메트릭은 관리 클러스터의 **record** 매개변수에 정의된 이름으로 수집됩니다. 반환된 메트릭 값은 쿼리 표현식을 실행한 후 결과입니다.

하나 또는 두 개의 섹션을 사용할 수 있습니다. 사용자 워크로드 메트릭은 사용자 워크로드 메트릭 추가 섹션을 참조하십시오.

참고: 전체 함대에 적용하는 대신 사용자 지정 메트릭 허용 목록에서 각 관리 클러스터를 개별적으로 사용자 지정할 수도 있습니다. 관리 클러스터에서 직접 동일한 YAML 을 생성하여 사용자 지정할 수 있습니다.

5. Grafana 대시보드 **탐색** 페이지에서 지표를 쿼리하여 사용자 지정 지표에서 데이터 수집을 확인합니다. 자체 대시보드에서 사용자 지정 메트릭을 사용할 수도 있습니다.

4.2.1. 사용자 워크로드 메트릭 추가

OpenShift Container Platform의 워크로드에서 OpenShift Container Platform 사용자 정의 메트릭을 수집하여 Grafana 대시보드의 지표를 표시합니다. 다음 단계를 완료합니다.

1. OpenShift Container Platform 클러스터에서 모니터링을 활성화합니다. 추가 리소스 섹션에서 사용자 정의 프로젝트에 대한 모니터링 활성화를 참조하십시오.
사용자 정의 워크로드가 활성화된 관리형 클러스터가 있는 경우 사용자 워크로드는 **테스트** 네임스페이스에 있으며 메트릭을 생성합니다. 이러한 메트릭은 OpenShift Container Platform 사용자 워크로드에서 Prometheus에 의해 수집됩니다.
2. **observability-metrics-custom-allowlist** 구성 맵에 사용자 워크로드 지표를 추가하여 **테스트** 네임스페이스에서 지표를 수집합니다. 다음 예제를 확인합니다.

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: observability-metrics-custom-allowlist
  namespace: test
data:
  uwl_metrics_list.yaml: ❶
  names: ❷
  - sample_metrics

```

- 1 구성 맵 데이터의 키를 입력합니다.
- 2 YAML 형식으로 구성 맵 데이터의 값을 입력합니다. **names** 섹션에는 **테스트 네임스페이스**에서 수집할 지표 이름 목록이 포함되어 있습니다. 구성 맵을 생성한 후 관찰 가능 수집기는 대상 네임스페이스에서 허브 클러스터로 지표를 수집하고 푸시합니다.

4.2.2. 기본 메트릭 제거

관리 클러스터에서 특정 메트릭에 대한 데이터를 수집하지 않으려면 **observability-metrics-custom-allowlist.yaml** 파일에서 지표를 제거합니다. 지표를 제거하면 관리 클러스터에서 지표 데이터가 수집되지 않습니다. 기본 메트릭을 제거하려면 다음 단계를 완료합니다.

1. 다음 명령을 사용하여 **mco observability** 가 활성화되어 있는지 확인합니다.

```
oc get mco observability -o yaml
```

2. 메트릭 이름 시작 시 하이픈 - 을 사용하여 기본 메트릭의 이름을 **metrics_list.yaml** 매개변수에 추가합니다. 다음 메트릭 예제를 확인합니다.

```
-cluster_infrastructure_provider
```

3. 다음 명령을 사용하여 **open-cluster-management-observability** 네임스페이스에 **observability-metrics-custom-allowlist** 구성 맵을 생성합니다.

```
oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml
```

4. 관찰 기능 서비스가 관리 클러스터에서 특정 지표를 수집하지 않는지 확인합니다. Grafana 대시보드에서 지표를 쿼리하면 지표가 표시되지 않습니다.

4.3. 보존을 위한 고급 구성 추가

필요에 따라 각 관찰 가능성 구성 요소의 보존을 업데이트하려면 **고급** 구성 섹션을 추가합니다. 다음 단계를 완료합니다.

1. 다음 명령을 사용하여 **MultiClusterObservability** 사용자 정의 리소스를 편집합니다.

```
oc edit mco observability -o yaml
```

2. **고급** 섹션을 파일에 추가합니다. YAML 파일은 다음 내용과 유사할 수 있습니다.

```
spec:
  advanced:
    retentionConfig:
      blockDuration: 2h
      deleteDelay: 48h
      retentionInLocal: 24h
      retentionResolutionRaw: 365d
      retentionResolution5m: 365d
      retentionResolution1h: 365d
    receive:
      resources:
```

```
limits:
  memory: 4096Gi
  replicas: 3
```

참고:

- 고급 구성에 추가할 수 있는 모든 매개변수에 대한 설명은 **Observability API 설명서**를 참조하십시오.
 - **retentionResolutionRaw, retentionResolution5m** 또는 **retentionResolution1h** 와 같은 모든 해결 수준에 대한 기본 보장은 365 일 (**365d**)입니다. **MultiClusterObservability spec.advanced.retentionConfig** 매개변수에서 해결 보장에 대한 명시적 값을 설정해야 합니다.
3. 이전 버전에서 업그레이드하고 해당 버전 보장 구성을 유지하려면 이전에 언급한 구성을 추가합니다. 다음 단계를 완료합니다.
- a. 다음 명령을 실행하여 **MultiClusterObservability** 리소스로 이동합니다.

```
edit mco observability
```

- b. **spec.advanced.retentionConfig** 매개변수에서 다음 구성을 적용합니다.

```
spec:
  advanced:
    retentionConfig:
      retentionResolutionRaw: 365d
      retentionResolution5m: 365d
      retentionResolution1h: 365d
```

4.4. 단일 노드 OPENSIFT 클러스터에 대한 동적 메트릭

동적 메트릭 컬렉션은 특정 조건에 따라 자동 메트릭 컬렉션을 지원합니다. 기본적으로 단일 노드 OpenShift 클러스터는 포드 및 컨테이너 리소스 지표를 수집하지 않습니다. 단일 노드 OpenShift 클러스터가 특정 리소스 소비 수준에 도달하면 정의된 세분화된 메트릭이 동적으로 수집됩니다. 일정 기간 동안 클러스터 리소스 사용량이 임계값보다 일관되게 작으면 세분화된 메트릭 수집이 중지됩니다.

메트릭은 컬렉션 규칙에 의해 지정된 관리 클러스터의 조건에 따라 동적으로 수집됩니다. 이러한 메트릭은 동적으로 수집되므로 다음 Red Hat Advanced Cluster Management Grafana 대시보드는 데이터를 표시하지 않습니다. 컬렉션 규칙이 활성화되고 해당 메트릭이 수집되면 다음 패널은 수집 규칙이 시작되는 기간 동안 데이터를 표시합니다.

- Kubernetes/Compute 리소스/네임스페이스(Pods)
- Kubernetes/Compute 리소스/네임스페이스(워크로드)
- Kubernetes/Compute 리소스/노드(Pods)
- Kubernetes/Compute 리소스/Pod
- Kubernetes/Compute Resources/Workload 컬렉션 규칙에는 다음 조건이 포함됩니다.
- 동적으로 수집할 지표 세트입니다.
- PromQL 표현식으로 작성된 조건입니다.

- 컬렉션의 시간 간격을 **true** 로 설정해야 합니다.
- 수집 규칙을 평가해야 하는 클러스터를 선택하는 일치 표현식입니다.

기본적으로 컬렉션 규칙은 관리 클러스터에서 30초마다 또는 특정 시간 간격으로 지속적으로 평가됩니다. 컬렉션 간격과 시간 간격 사이의 가장 낮은 값이 우선합니다. **for** 속성에서 지정한 기간 동안 컬렉션 규칙 조건이 지속되면 컬렉션 규칙이 시작되고 규칙에서 지정한 메트릭이 관리 클러스터에서 자동으로 수집됩니다. 메트릭 컬렉션은 컬렉션 규칙 조건이 더 이상 관리 클러스터에 없는 후 15분 이상 시작된 후 자동으로 중지됩니다.

컬렉션 규칙은 **collect_rules** 라는 매개 변수 섹션으로 그룹화됩니다. 여기서 그룹으로 활성화하거나 비활성화할 수 있습니다. Red Hat Advanced Cluster Management 설치에는 **HighCPUUsage** 및 **HighMemoryUsage** 의 두 가지 기본 컬렉션 규칙이 있는 **SNOResourceUsage** SNOResourceUsage 가 포함되어 있습니다. 노드 CPU 사용량이 70%를 초과하면 **HighCPUUsage** 컬렉션 규칙이 시작됩니다. **HighMemoryUsage** 컬렉션 규칙은 단일 노드 OpenShift 클러스터의 전체 메모리 사용률이 사용 가능한 노드 메모리의 70%를 초과하면 시작됩니다. 현재는 앞서 언급한 임계값이 고정되어 있으며 변경할 수 없습니다. **for** 속성에 의해 지정된 간격보다 많은 컬렉션 규칙이 시작되면 시스템은 **dynamic_metrics** 섹션에 지정된 메트릭 수집이 자동으로 시작됩니다.

다음 YAML 파일에서 **collect_rules** 섹션의 동적 메트릭 목록을 확인합니다.

```
collect_rules:
  - group: SNOResourceUsage
    annotations:
      description: >
        By default, a {sno} cluster does not collect pod and container resource metrics. Once a {sno}
        cluster
        reaches a level of resource consumption, these granular metrics are collected dynamically.
        When the cluster resource consumption is consistently less than the threshold for a period of
        time,
        collection of the granular metrics stops.
    selector:
      matchExpressions:
        - key: clusterType
          operator: In
          values: ["{sno}"]
    rules:
      - collect: SNOHighCPUUsage
        annotations:
          description: >
            Collects the dynamic metrics specified if the cluster cpu usage is constantly more than 70% for
            2 minutes
          expr: (1 - avg(rate(node_cpu_seconds_total{mode="idle"}[5m]))) * 100 > 70
          for: 2m
        dynamic_metrics:
          names:
            - container_cpu_cfs_periods_total
            - container_cpu_cfs_throttled_periods_total
            - kube_pod_container_resource_limits
            - kube_pod_container_resource_requests
            - namespace_workload_pod:kube_pod_owner:relabel
            - node_namespace_pod_container:container_cpu_usage_seconds_total:sum_irate
            - node_namespace_pod_container:container_cpu_usage_seconds_total:sum_rate
      - collect: SNOHighMemoryUsage
        annotations:
          description: >
```

```

Collects the dynamic metrics specified if the cluster memory usage is constantly more than 70%
for 2 minutes
expr: (1 - sum(:node_memory_MemAvailable_bytes:sum) /
sum(kube_node_status_allocatable{resource="memory"})) * 100 > 70
for: 2m
dynamic_metrics:
  names:
    - kube_pod_container_resource_limits
    - kube_pod_container_resource_requests
    - namespace_workload_pod:kube_pod_owner:relabel
  matches:
    - __name__="container_memory_cache",container!=""
    - __name__="container_memory_rss",container!=""
    - __name__="container_memory_swap",container!=""
    - __name__="container_memory_working_set_bytes",container!=""

```

다음 예와 같이 **collect_rules.group** 을 **custom-allowlist** 에서 비활성화할 수 있습니다. **collect_rules.group** 을 비활성화하면 메트릭 컬렉션이 이전 동작으로 되돌아갑니다. 이러한 메트릭은 지정된 간격으로 수집됩니다.

```

collect_rules:
  - group: -SNOResourceUsage

```

이 데이터는 규칙이 시작될 때 Grafana 에만 표시됩니다.

4.5. 콘솔에서 MULTICLUSTEROBSERVABILITY 사용자 정의 리소스 복제본 업데이트

워크로드가 증가하면 관찰 기능 Pod의 복제본 수를 늘립니다. hub 클러스터에서 Red Hat OpenShift Container Platform 콘솔로 이동합니다. **MultiClusterObservability** 사용자 정의 리소스를 찾고 복제본을 변경하려는 구성 요소의 **replicas** 매개변수 값을 업데이트합니다. 업데이트된 YAML은 다음 콘텐츠와 유사할 수 있습니다.

```

spec:
  advanced:
    receive:
      replicas: 6

```

mco observability 사용자 정의 리소스 내의 매개변수에 대한 자세한 내용은 Observability API 설명서를 참조하십시오.

4.6. 영구 볼륨 및 영구 볼륨 클레임 증가 및 감소

영구 볼륨 및 영구 볼륨 클레임을 늘리고 줄여 스토리지 클래스의 스토리지 양을 변경합니다. 다음 단계를 완료합니다.

1. 스토리지 클래스가 볼륨 확장을 지원하는 경우 영구 볼륨의 크기를 늘리려면 **MultiClusterObservability** 사용자 정의 리소스를 업데이트합니다.
2. 영구 볼륨의 크기를 줄이려면 영구 볼륨을 사용하여 Pod를 제거한 후 영구 볼륨을 삭제하고 다시 생성합니다. 영구 볼륨에서 데이터 손실이 발생할 수 있습니다. 다음 단계를 완료합니다.
 - a. **MultiClusterObservability** 사용자 정의 리소스에 주석 **mco-pause: "true"** 주석을 추가하여 **MultiClusterObservability** Operator를 일시 중지합니다.

- b. 원하는 구성 요소의 상태 저장 세트 또는 배포를 찾습니다. 복제본 수를 **0** 으로 변경합니다. 이렇게 하면 종료가 시작되어 데이터 손실을 방지하는 데 적용 가능한 경우 로컬 데이터를 업로드해야 합니다. 예를 들어 Thanos **Receive** stateful 세트의 이름은 **observability-thanos-receive-default** 이며 기본적으로 세 개의 복제본이 있습니다. 따라서 다음과 같은 영구 볼륨 클레임을 찾고 있습니다.
 - **data-observability-thanos-receive-default-0**
 - **data-observability-thanos-receive-default-1**
 - **data-observability-thanos-receive-default-2**
 - c. 원하는 구성 요소에서 사용하는 영구 볼륨 및 영구 볼륨 클레임을 삭제합니다.
 - d. **MultiClusterObservability** 사용자 정의 리소스에서 구성 요소의 스토리지 크기를 스토리지 크기 필드에 원하는 양으로 편집합니다. 구성 요소 이름을 접두사로 지정합니다.
 - e. 이전에 추가한 주석을 제거하여 **MultiClusterObservability** Operator의 일시 중지를 해제합니다.
 - f. Operator를 일시 중지한 후 재조정을 시작하려면 **multicluster-observability-operator** 및 **observatorium-operator** Pod를 삭제합니다. Pod가 즉시 다시 생성되고 조정됩니다.
3. **MultiClusterObservability** 사용자 정의 리소스를 확인하여 영구 볼륨 및 볼륨 클레임이 업데이트되었는지 확인합니다.

4.7. 경로 인증서 사용자 정의

OpenShift Container Platform 경로 인증을 사용자 지정하려면 **alt_names** 섹션에 경로를 추가해야 합니다. OpenShift Container Platform 경로에 액세스할 수 있도록 하려면 **alertmanager.apps.<domainname>**, **observatorium-api.apps.<domainname>**, **rbac-query-proxy.apps.<domainname>** 이라는 정보를 추가합니다.

자세한 내용은 Governance 문서의 alertmanager 경로에 대한 인증서 교체 를 참조하십시오.

참고: 사용자는 인증서 교체 및 업데이트를 담당합니다.

4.8. 오브젝트 저장소에 액세스하기 위한 인증서 사용자 정의

인증 기관이 포함된 **Secret** 리소스를 생성하고 **MultiClusterObservability** 사용자 정의 리소스를 구성하여 **observability** 오브젝트 저장소를 사용하여 보안 연결을 구성할 수 있습니다. 다음 단계를 완료합니다.

1. 오브젝트 저장소 연결을 검증하려면 다음 명령을 사용하여 인증 기관이 포함된 파일에 **Secret** 오브젝트를 생성합니다.

```
oc create secret generic <tls_secret_name> --from-file=ca.crt=<path_to_file> -n open-cluster-management-observability
```

- a. 또는 다음 YAML 을 적용하여 보안을 생성할 수 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <tls_secret_name>
  namespace: open-cluster-management-observability
```

```

type: Opaque
data:
  ca.crt: <base64_encoded_ca_certificate>

```

선택 사항: 상호 TLS를 활성화하려면 이전 시크릿에 **public.crt** 및 **private.key** 키를 추가해야 합니다.

- 다음 명령을 사용하여 **metricObjectStorage** 섹션에 TLS 시크릿 세부 정보를 추가합니다.

```
oc edit mco observability -o yaml
```

파일은 다음 YAML 과 유사할 수 있습니다.

```

metricObjectStorage:
  key: thanos.yaml
  name: thanos-object-storage
  tlsSecretName: tls-certs-secret ❶
  tlsSecretMountPath: /etc/minio/certs ❷

```

- ❶ **tlsSecretName** 의 값은 이전에 생성한 **Secret** 오브젝트의 이름입니다.
- ❷ **tlsSecretMountPath** 매개변수에 정의된 **/etc/minio/certs/** 경로는 인증서가 Observability 구성 요소에 마운트된 위치를 지정합니다. 이 경로는 다음 단계에 필요합니다.

- 인증서 세부 정보와 함께 **http_config.tls_config** 섹션을 추가하여 **thanos-object-storage** 시크릿에서 **thanos.yaml** 정의를 업데이트합니다. 다음 예제를 확인합니다.

```

thanos.yaml: |
  type: s3
  config:
    bucket: "thanos"
    endpoint: "minio:9000"
    insecure: false ❶
    access_key: "minio"
    secret_key: "minio123"
    http_config:
      tls_config:
        ca_file: /etc/minio/certs/ca.crt ❷
        insecure_skip_verify: false

```

- ❶ HTTPS를 활성화하려면 **insecure** 매개변수를 **false** 로 설정합니다.
- ❷ **ca_file** 매개변수의 경로는 **MultiClusterObservability** 사용자 정의 리소스의 **tlsSecretMountPath** 와 일치해야 합니다. **ca.crt** 는 < **tls_secret_name** > 시크릿 리소스의 키와 일치해야 합니다.

선택 사항: 상호 TLS를 활성화하려면 **cert_file** 및 **key_file** 키를 **tls_config** 섹션에 추가해야 합니다. 다음 예제를 참조하십시오.

```

thanos.yaml: |
  type: s3
  config:
    bucket: "thanos"

```

```

endpoint: "minio:9000"
insecure: false
access_key: "minio"
secret_key: "minio123"
http_config:
  tls_config:
    ca_file: /etc/minio/certs/ca.crt 1
    cert_file: /etc/minio/certs/public.crt
    key_file: /etc/minio/certs/private.key
    insecure_skip_verify: false
    
```

1 **ca_file, cert_file, key_file**의 경로는 **MultiClusterObservability** 사용자 정의 리소스의 **tlsSecretMountPath**와 일치해야 합니다. **ca.crt, public.crt, private.crt**는 **tls_secret_name** > **Secret** 리소스의 각 키와 일치해야 합니다.

4. 오브젝트 저장소에 액세스할 수 있는지 확인하려면 Pod가 배포되었는지 확인합니다. 다음 명령을 실행합니다.

```

oc -n open-cluster-management-observability get pods -l app.kubernetes.io/name=thanos-store
    
```

4.9. 관찰 기능 애드온에 대한 프록시 설정 구성

관리 클러스터의 통신이 HTTP 및 HTTPS 프록시 서버를 통해 hub 클러스터에 액세스할 수 있도록 프록시 설정을 구성합니다. 일반적으로 애드온은 허브 클러스터와 관리 클러스터 간에 HTTP 및 HTTPS 프록시 서버를 지원하기 위해 특별한 구성이 필요하지 않습니다. 그러나 관찰 기능 애드온을 활성화한 경우 프록시 구성을 완료해야 합니다.

4.10. 사전 요구 사항

- 허브 클러스터가 있어야 합니다.
- hub 클러스터와 관리 클러스터 간에 프록시 설정을 활성화했습니다.

관찰 기능 애드온에 대한 프록시 설정을 구성하려면 다음 단계를 완료합니다.

1. hub 클러스터의 클러스터 네임스페이스로 이동합니다.
2. **spec.proxyConfig** 매개변수를 추가하여 프록시 설정으로 **AddOnDeploymentConfig** 리소스를 생성합니다. 다음 YAML 예제를 확인합니다.

```

apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: <addon-deploy-config-name>
  namespace: <managed-cluster-name>
spec:
  agentInstallNamespace: open-cluster-managment-addon-observability
  proxyConfig:
    httpsProxy: "http://<username>:<password>@<ip>:<port>" 1
    noProxy: ".cluster.local, .svc, 172.30.0.1" 2
    
```

1 이 필드의 경우 HTTP 프록시 또는 HTTPS 프록시를 지정할 수 있습니다.

2 kube-apiserver 의 IP 주소를 포함합니다.

- IP 주소를 가져오려면 관리 클러스터에서 다음 명령을 실행합니다.

```
oc -n default describe svc kubernetes | grep IP:
```

- ManagedClusterAddOn** 리소스로 이동하여 수행한 **AddOnDeploymentConfig** 리소스를 참조하여 업데이트합니다. 다음 YAML 예제를 확인합니다.

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: observability-controller
  namespace: <managed-cluster-name>
spec:
  installNamespace: open-cluster-managment-addon-observability
  configs:
    - group: addon.open-cluster-management.io
      resource: AddonDeploymentConfig
      name: <addon-deploy-config-name>
      namespace: <managed-cluster-name>
```

- 프록시 설정을 확인합니다. 프록시 설정을 성공적으로 구성한 경우 관리 클러스터의 관찰 기능에 이진트에 의해 배포된 지표 수집기는 허브 클러스터로 데이터를 보냅니다. 다음 단계를 완료합니다.
 - hub 클러스터로 이동한 다음 Grafana 대시보드의 관리 클러스터로 이동합니다.
 - 프록시 설정에 대한 메트릭을 확인합니다.

4.11. 관찰 기능 애드온에 대한 프록시 설정 비활성화

개발이 필요한 경우 hub 클러스터 및 관리 클러스터에 대해 구성된 관찰 기능 애드온에 대한 프록시 설정을 비활성화해야 할 수 있습니다. 언제든지 관찰 기능 애드온의 프록시 설정을 비활성화할 수 있습니다. 다음 단계를 완료합니다.

- ManagedClusterAddOn** 리소스로 이동합니다.
- 참조된 **AddOnDeploymentConfig** 리소스를 제거합니다.

4.12. 추가 리소스

- 자세한 내용은 [Prometheus 구성](#) 을 참조하십시오. 기록 규칙 및 경고 규칙에 대한 자세한 내용은 [Prometheus 문서의 레코딩 규칙 및 경고 규칙](#) 을 참조하십시오.
- 대시보드 보기에 대한 자세한 내용은 [Grafana 대시보드 사용](#) 을 참조하십시오.
- [외부 엔드포인트로 메트릭 내보내기](#) 를 참조하십시오.
- [사용자 정의 프로젝트에 대한 모니터링 활성화](#) 를 참조하십시오.
- [Observability API](#) 를 참조하십시오.

- [alertmanager](#) 경로의 인증서를 업데이트하는 방법에 대한 자세한 내용은 [alertmanager 에 대한 인증서 교체를 참조하십시오.](#)
- 관찰성 경고에 대한 자세한 내용은 [Observability](#) 경고를 참조하십시오.
- 경고 전달에 대한 자세한 내용은 [Prometheus Alertmanager 설명서](#) 를 참조하십시오.
- 자세한 내용은 [Observability](#) 경고를 참조하십시오.
- 관찰 가능성 서비스에 대한 자세한 내용은 [Observability 서비스 도입](#) 을 참조하십시오.
- 자세한 내용은 [Management Workload Partitioning](#) 을 참조하십시오.
- 이 항목의 시작 부분으로 돌아가서 [관찰 기능을 사용자 정의합니다.](#)

5장. 관찰 기능 경고 관리

hub 클러스터 및 관리 클러스터 변경에 대한 알림을 받을 관찰 기능 서비스에 대한 경고를 수신하고 정의합니다.

- [Alertmanager 구성](#)
- [경고 전달](#)
- [음소거 경고](#)
- [경고 비활성화](#)

5.1. ALERTMANAGER 구성

이메일, Slack, PagerDuty와 같은 외부 메시징 도구를 통합하여 Alertmanager에서 알림을 수신합니다. 통합을 추가하려면 **open-cluster-management-observability** 네임스페이스에서 **alertmanager-config** 시크릿을 재정의하고 Alertmanager의 경로를 구성해야 합니다. 사용자 정의 수신자 규칙을 업데이트하려면 다음 단계를 완료합니다.

1. **alertmanager-config** 시크릿에서 데이터를 추출합니다. 다음 명령을 실행합니다.

```
oc -n open-cluster-management-observability get secret alertmanager-config --template='{{ index .data "alertmanager.yaml" }}' |base64 -d > alertmanager.yaml
```

2. 다음 명령을 실행하여 **alertmanager.yaml** 파일 구성을 편집하고 저장합니다.

```
oc -n open-cluster-management-observability create secret generic alertmanager-config --from-file=alertmanager.yaml --dry-run -o=yaml | oc -n open-cluster-management-observability replace secret --filename=-
```

업데이트된 시크릿은 다음 콘텐츠와 유사할 수 있습니다.

```
global
  smtp_smarthost: 'localhost:25'
  smtp_from: 'alertmanager@example.org'
  smtp_auth_username: 'alertmanager'
  smtp_auth_password: 'password'
templates:
- '/etc/alertmanager/template/*.tmpl'
route:
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: team-X-mails
routes:
- match_re:
  service: ^(foo1|foo2|baz)$
  receiver: team-X-mails
```

변경 사항이 수정 후 즉시 적용됩니다. Alertmanager의 예는 [prometheus/alertmanager](#) 를 참조하십시오.

5.2. 경고 전달

관찰 기능을 활성화하면 OpenShift Container Platform 관리 클러스터의 경고가 허브 클러스터로 자동으로 전송됩니다. **alertmanager-config** YAML 파일을 사용하여 외부 알림 시스템으로 경고를 구성할 수 있습니다.

alertmanager-config YAML 파일의 다음 예제를 확인합니다.

```
global:
  slack_api_url: '<slack_webhook_url>'

route:
  receiver: 'slack-notifications'
  group_by: [alertname, datacenter, app]

receivers:
- name: 'slack-notifications'
  slack_configs:
  - channel: '#alerts'
    text: 'https://internal.myorg.net/wiki/alerts/{{ .GroupLabels.app }}/{{ .GroupLabels.alertname }}'
```

경고 전달을 위해 프록시를 구성하려면 다음 글로벌 항목을 **alertmanager-config** YAML 파일에 추가합니다.

```
global:
  slack_api_url: '<slack_webhook_url>'
  http_config:
    proxy_url: http://****
```

5.2.1. 관리 클러스터에 대한 경고 전달 비활성화

관리 클러스터에 대한 경고 전달을 비활성화하려면 **MultiClusterObservability** 사용자 정의 리소스에 다음 주석을 추가합니다.

```
metadata:
  annotations:
    mco-disable-alerting: "true"
```

주석을 설정하면 관리 클러스터에서 경고 전달 구성이 복원됩니다. **openshift-monitoring** 네임스페이스의 **ocp-monitoring-config** 구성 맵에 대한 변경 사항도 복원됩니다. 주석을 설정하면 **ocp-monitoring-config** 구성 맵이 더 이상 observability Operator 끝점에 의해 관리되거나 업데이트되지 않습니다. 구성을 업데이트하면 관리형 클러스터의 Prometheus 인스턴스가 다시 시작됩니다.

중요: 메트릭에 영구 불륨이 있는 Prometheus 인스턴스가 있고 Prometheus 인스턴스가 다시 시작되면 관리 클러스터의 지표가 손실됩니다. hub 클러스터의 메트릭은 영향을 받지 않습니다.

변경 사항을 되돌리면 **open-cluster-management-addon-observability** 네임스페이스에 **cluster-monitoring-reverted** 라는 ConfigMap이 생성됩니다. ConfigMap에서 수동으로 추가한 새 경고 전달 구성이 복원되지 않습니다.

허브 클러스터 경고 관리자가 더 이상 관리된 클러스터 경고를 타사 메시징 툴에 전파하지 않는지 확인합니다. 이전 섹션, Alertmanager 구성을 참조하십시오.

5.3. 음소거 경고

수신하지 않으려는 경고를 추가합니다. 경고 이름, 일치 레이블 또는 시간 기간으로 경고를 음소거할 수 있습니다. 음소거하려는 경고를 추가하면 ID가 생성됩니다. 음소거 경고의 ID는 **d839aca9-ed46-40be-84c4-dca8773671da** 문자열과 유사할 수 있습니다.

경고를 음소거하는 방법에 대해 계속 읽습니다.

- Red Hat Advanced Cluster Management 경고를 음소거하려면 **open-cluster-management-observability** 네임스페이스의 **alertmanager-main** Pod에 액세스할 수 있어야 합니다. 예를 들어 Pod 터미널에 다음 명령을 입력하여 음소거 **SampleAlert**

```
amtool silence add --alertmanager.url="http://localhost:9093" --author="user" --comment="Silencing sample alert" alertname="SampleAlert"
```

- 여러 일치 레이블을 사용하여 경고를 음소거합니다. 다음 명령은 **match-label-1** 및 **match-label-2**를 사용합니다.

```
amtool silence add --alertmanager.url="http://localhost:9093" --author="user" --comment="Silencing sample alert" <match-label-1>=<match-value-1> <match-label-2>=<match-value-2>
```

- 특정 기간 동안 경고를 음소거하려면 **--duration** 플래그를 사용합니다. 다음 명령을 실행하여 **SampleAlert**를 1시간 동안 음소거합니다.

```
amtool silence add --alertmanager.url="http://localhost:9093" --author="user" --comment="Silencing sample alert" --duration="1h" alertname="SampleAlert"
```

음소거된 경고의 시작 또는 종료 시간을 지정할 수도 있습니다. 다음 명령을 입력하여 특정 시작 시 **SampleAlert**를 음소거합니다.

```
amtool silence add --alertmanager.url="http://localhost:9093" --author="user" --comment="Silencing sample alert" --start="2023-04-14T15:04:05-07:00" alertname="SampleAlert"
```

- 생성된 모든 음소거 경고를 보려면 다음 명령을 실행합니다.

```
amtool silence --alertmanager.url="http://localhost:9093"
```

- 더 이상 경고를 음소거하지 않으려면 다음 명령을 실행하여 경고의 실침을 종료합니다.

```
amtool silence expire --alertmanager.url="http://localhost:9093" "d839aca9-ed46-40be-84c4-dca8773671da"
```

- 모든 경고의 silencing을 종료하려면 다음 명령을 실행합니다.

```
amtool silence expire --alertmanager.url="http://localhost:9093" $(amtool silence query --alertmanager.url="http://localhost:9093" -q)
```

5.3.1. 관찰성 스토리지 마이그레이션

경고 음소거를 사용하는 경우 음소거를 이전 상태에서 유지하면서 관찰 기능을 마이그레이션할 수 있습니다. 이렇게 하려면 선택한 **StorageClass** 리소스를 사용하는 새 **StatefulSets** 및 **PersistentVolumes (PV)** 리소스를 생성하여 Red Hat Advanced Cluster Management observability 스토리지를 마이그레이션합니다.

참고: PV의 스토리지는 클러스터에서 수집된 지표를 저장하는 데 사용되는 오브젝트 스토리지와 다릅니다.

StatefulSets 및 PV를 사용하여 관찰 기능 데이터를 새 스토리지로 마이그레이션하는 경우 다음 데이터 구성 요소가 저장됩니다.

- **Observatorium** 또는 **Thanos**: 데이터를 수신한 다음 오브젝트 스토리지에 업로드합니다. 일부 구성 요소는 PV에 데이터를 저장합니다. 이 데이터의 경우 Observatorium 또는 Thanos는 시작 시 오브젝트 스토리지를 자동으로 다시 생성하므로 이 데이터를 손실해도 결과가 발생하지 않습니다.
- **Alertmanager**: 음소거된 경고만 저장합니다. 이러한 음소거된 경고를 유지하려면 해당 데이터를 새 PV로 마이그레이션해야 합니다.

관찰성 스토리지를 마이그레이션하려면 다음 단계를 완료합니다.

1. **MultiClusterObservability** 에서 **.spec.storageConfig.storageClass** 필드를 새 스토리지 클래스로 설정합니다.
2. **PersistentVolumeClaim** 을 삭제할 때에도 이전 **PersistentVolumes** 의 데이터를 유지하려면 기존의 모든 **PersistentVolumes** 로 이동합니다.
3. **reclaimPolicy** 를 "Retain": 'oc patch pv <your-pv-name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}' 로 변경합니다.
4. **선택 사항:** 데이터 손실을 방지하려면 [OCP 4의 DG 8 Operator의 다른 스토리지 클래스로 영구 데이터 마이그레이션](#) 을 참조하십시오.
5. 다음 **StatefulSet** 리소스에서 **StatefulSet** 및 **PersistentVolumeClaim** 을 모두 삭제합니다.
 - a. **alertmanager-db-observability-alertmanager**
 - b. **data-observability-thanos-compact**
 - c. **data-observability-thanos-receive-default**
 - d. **data-observability-thanos-store-shard**
 - e. **중요:** 새 **StatefulSet** 을 생성할 수 있도록 **MultiClusterObservability** Operator Pod를 삭제한 후 다시 생성해야 할 수 있습니다.
6. 이름이 동일하지만 올바른 **StorageClass** 를 사용하여 새 **PersistentVolumeClaim** 을 다시 생성합니다.
7. 이전 **PersistentVolume** 을 참조하는 새 **PersistentVolumeClaim** 을 생성합니다.
8. 새 **StatefulSet** 및 **PersistentVolumes** 에서 선택한 새 **StorageClass** 를 사용하는지 확인합니다.

5.4. 경고 비활성화

전 세계적으로 클러스터 전체에서 Red Hat Advanced Cluster Management 경고를 표시하지 않도록 합니다. **open-cluster-management-observability** 네임스페이스에서 **alertmanager-config** 에서 억제 규칙을 정의하여 경고를 비활성화합니다.

억제 규칙은 기존의 다른 일치 항목 집합과 일치하는 매개 변수 집합이 있는 경우 경고를 음소거합니다. 규칙을 적용하려면 대상 및 소스 경고 모두 동일한 목록의 라벨 이름에 대해 동일한 레이블 값이 있어야 합니다. **inhibit_rules** 는 다음과 유사할 수 있습니다.

```

global:
  resolve_timeout: 1h
inhibit_rules: 1
  - equal:
    - namespace
  source_match: 2
    severity: critical
  target_match_re:
    severity: warning|info

```

1

inhibit_rules 매개 변수 섹션은 동일한 네임스페이스에서 경고를 찾도록 정의됩니다. 네임스페이스 내에서 위험 경고가 시작되고 해당 네임스페이스에 심각도 수준 경고 또는 정보가 포함된 기타 경고가 있는 경우 중요한 경고만 **Alertmanager** 수신자로 라우팅됩니다. 일치하는 경우 다음 경고가 표시될 수 있습니다.

```

ALERTS{alertname="foo", namespace="ns-1", severity="critical"}
ALERTS{alertname="foo", namespace="ns-1", severity="warning"}

```

2

source_match 및 **target_match_re** 매개 변수의 값이 일치하지 않으면 경고가 수신자로 라우팅됩니다.

```

ALERTS{alertname="foo", namespace="ns-1", severity="critical"}
ALERTS{alertname="foo", namespace="ns-2", severity="warning"}

```

•

Red Hat Advanced Cluster Management에서 억제된 경고를 보려면 다음 명령을 입력합니다.

```
amtool alert --alertmanager.url="http://localhost:9093" --inhibited
```

5.5. 추가 리소스

•

자세한 내용은 [관찰 기능 사용자 지정](#)을 참조하십시오.

•

자세한 관찰 기능 항목은 [Observability 서비스 도입](#)을 참조하십시오.

6장. GRAFANA에서 관리되는 클러스터 레이블 사용

관리 클러스터 레이블을 활성화하여 Grafana 대시보드와 함께 사용합니다. hub 클러스터에서 관찰 기능이 활성화되면 `open-cluster-management-observability` 네임스페이스에 `observability-managed-cluster-label-allowlist` ConfigMap이 생성됩니다. ConfigMap에는 ACM - Cluster Overview Grafana 대시보드 내에서 필터링할 레이블 이름 목록을 채울 `observability-rbac-query-proxy` Pod에서 관리하는 관리 클러스터 레이블 목록이 포함되어 있습니다. 기본적으로 관찰 기능은 `observability-managed-cluster-label-allowlist` ConfigMap의 레이블 서브 세트를 무시합니다.

클러스터를 관리형 클러스터 플릿 또는 수정으로 가져올 때 `observability-rbac-query-proxy` Pod는 관리 클러스터 라벨에 대한 참조의 변경 사항을 감시하고 변경 사항을 반영하도록 `observability-managed-cluster-label-allowlist` ConfigMap을 자동으로 업데이트합니다. ConfigMap에는 `ignore_labels` 또는 레이블 목록에 포함된 고유한 레이블 이름만 포함됩니다. `observability-managed-cluster-label-allowlist` ConfigMap은 다음 YAML 파일과 유사할 수 있습니다.

```
data:
  managed_cluster.yaml: |
    ignore_labels: 1
      - clusterID
      - cluster.open-cluster-management.io/clusterset
      - feature.open-cluster-management.io/addon-application-manager
      - feature.open-cluster-management.io/addon-cert-policy-controller
      - feature.open-cluster-management.io/addon-cluster-proxy
      - feature.open-cluster-management.io/addon-config-policy-controller
      - feature.open-cluster-management.io/addon-governance-policy-framework
      - feature.open-cluster-management.io/addon-iam-policy-controller
      - feature.open-cluster-management.io/addon-observability-controller
      - feature.open-cluster-management.io/addon-search-collector
      - feature.open-cluster-management.io/addon-work-manager
      - installer.name
      - installer.namespace
      - local-cluster
      - name
    labels: 2
      - cloud
      - vendor
```

+ <1> ConfigMap의 `ignore_labels` 키 목록에 나열된 모든 레이블은 ACM - Clusters Overview Grafana 대시보드의 드롭다운 필터에서 제거됩니다. <2> 활성화된 레이블은 ACM - Clusters Overview Grafana 대시보드의 드롭다운 필터에 표시됩니다. 선택한 레이블 키 값에 따라 값을 `acm_managed_cluster_labels` 지표에서 가져옵니다.

Grafana에서 관리 클러스터 레이블을 사용하는 방법을 계속 읽습니다.

- [관리형 클러스터 레이블 추가](#)

- [관리형 클러스터 라벨 활성화](#)
- [관리형 클러스터 레이블 비활성화](#)

6.1. 관리형 클러스터 레이블 추가

observability-managed-cluster-label-allowlist ConfigMap에 관리 클러스터 레이블을 추가하면 Grafana에서 라벨을 필터 옵션으로 사용할 수 있습니다. 허브 클러스터 또는 관리 클러스터 플릿과 연결된 관리형 클러스터 오브젝트에 고유한 레이블을 추가합니다. 예를 들어 레이블 **department=finance** 를 관리 클러스터에 추가하면 **ConfigMap**이 업데이트되어 다음 변경 사항과 유사할 수 있습니다.

```
data:
  managed_cluster.yaml: |
    ignore_labels:
      - clusterID
      - cluster.open-cluster-management.io/clusterset
      - feature.open-cluster-management.io/addon-application-manager
      - feature.open-cluster-management.io/addon-cert-policy-controller
      - feature.open-cluster-management.io/addon-cluster-proxy
      - feature.open-cluster-management.io/addon-config-policy-controller
      - feature.open-cluster-management.io/addon-governance-policy-framework
      - feature.open-cluster-management.io/addon-iam-policy-controller
      - feature.open-cluster-management.io/addon-observability-controller
      - feature.open-cluster-management.io/addon-search-collector
      - feature.open-cluster-management.io/addon-work-manager
      - installer.name
      - installer.namespace
      - local-cluster
      - name
    labels:
      - cloud
      - department
      - vendor
```

6.2. 관리형 클러스터 라벨 활성화

observability-managed-cluster-label-allowlist ConfigMap의 **ignore_labels** 목록에서 레이블을 제거하여 이미 비활성화된 관리형 클러스터 레이블을 활성화합니다.

예를 들어 **local-cluster** 및 **name** 레이블을 활성화합니다. **observability-managed-cluster-label-allowlist ConfigMap**은 다음 콘텐츠와 유사할 수 있습니다.

```
data:
  managed_cluster.yaml: |
```

```

ignore_labels:
  - clusterID
  - installer.name
  - installer.namespace
labels:
  - cloud
  - vendor
  - local-cluster
  - name

```

ConfigMap이 30초 후에 다시 동기화되어 클러스터 라벨이 업데이트되었는지 확인합니다. **ConfigMap**을 업데이트한 후 **open-cluster-management-observability** 네임스페이스에서 **observability-rbac-query-proxy Pod** 로그를 확인하여 라벨이 나열된 위치를 확인합니다. **Pod** 로그에 다음 정보가 표시될 수 있습니다.

```
enabled managedcluster labels: <label>
```

Grafana 대시보드에서 레이블이 **Label** 드롭다운 메뉴에 값으로 나열되어 있는지 확인합니다.

6.3. 관리형 클러스터 레이블 비활성화

Label 드롭다운 필터에 관리 클러스터 레이블이 나열되지 않도록 제외합니다. **ignore_labels** 목록에 레이블 이름을 추가합니다. 예를 들어 **local-cluster** 및 **name** 을 **ignore_labels** 목록에 다시 추가하는 경우 **YAML**은 다음 파일과 유사할 수 있습니다.

```

data:
  managed_cluster.yaml: |
    ignore_labels:
      - clusterID
      - installer.name
      - installer.namespace
      - local-cluster
      - name
    labels:
      - cloud
      - vendor

```

open-cluster-management-observability 네임스페이스에서 **observability-rbac-query-proxy Pod** 로그를 확인하여 라벨이 나열된 위치를 확인합니다. **Pod** 로그에 다음 정보가 표시될 수 있습니다.

```
disabled managedcluster label: <label>
```

6.4. 추가 리소스

- [Grafana 대시보드 사용을 참조하십시오.](#)
- [Grafana에서 관리 클러스터 레이블을 사용하여 페이지 시작 부분으로 돌아갑니다.](#)

7장. 콘솔 소개에서 검색

Red Hat Advanced Cluster Management for Kubernetes의 경우 모든 클러스터에서 Kubernetes 리소스에 대한 가시성을 제공합니다. 또한 검색에서는 Kubernetes 리소스 및 다른 리소스와의 관계를 인덱싱합니다.

- [구성 요소 검색](#)
- [사용자 정의 및 구성 검색](#)
- [추가 리소스](#)

7.1. 구성 요소 검색

검색 아키텍처는 다음 구성 요소로 구성됩니다.

표 7.1. 구성 요소 테이블 검색

구성 요소 이름	지표	메트릭 유형	설명
search-collector			Kubernetes 리소스를 감시하고, 리소스 메타데이터를 수집하고, 모든 관리 클러스터에서 리소스에 대한 관계를 계산하고, 수집된 데이터를 search-indexer 로 보냅니다. 관리 클러스터의 search-collector 는 klusterlet-addon-search 라는 Pod로 실행됩니다.
search-indexer 수집기에서 리소스 메타데이터를 수신하고 PostgreSQL 데이터베이스에 씁니다. search-indexer 는 허브 클러스터의 리소스를 감시하여 활성 관리 클러스터를 추적합니다.	search_indexer_request_duration	히스토그램	시간(초)은 관리 클러스터에서 요청을 처리하는 데 걸리는 시간(초)입니다.
	search_indexer_request_size	히스토그램	검색 인덱서 요청의 총 변경 사항(관리된 클러스터에서) 총 변경 사항(추가, 업데이트, 삭제)입니다.

구성 요소 이름	지표	메트릭 유형	설명
	search_indexer_request_count	카운터	검색 인덱서에서 수신한 총 요청(관리된 클러스터에서)
	search_indexer_requests_in_flight	게이지	검색 인덱서가 지정된 시간에 처리 중인 총 요청입니다.
search-api GraphQL을 통해 검색 인덱스의 모든 클러스터 데이터에 액세스하고 RBAC(역할 기반 액세스 제어)를 적용합니다.	search_api_requests	히스토그램	HTTP 요청의 히스토그램(초)입니다.
	search_dbquery_duration_seconds	히스토그램	데이터베이스 요청 대기 시간(초)입니다.
	search_api_db_connection_failed_total	카운터	실패한 총 데이터베이스 연결 시도 수입니다.
search-postgres			PostgreSQL 데이터베이스 인스턴스의 모든 관리 클러스터에서 수집된 데이터를 저장합니다.

검색은 허브 클러스터에서 기본적으로 구성됩니다. 관리 클러스터를 프로비저닝하거나 수동으로 가져올 때 **klusterlet-addon-search**가 활성화됩니다. 관리 클러스터에서 검색을 비활성화하려면 클러스터의 **klusterlet** 애드온 설정 수정을 참조하십시오.

7.2. 사용자 정의 및 구성 검색

search-v2-operator 사용자 정의 리소스에서 기본값을 수정할 수 있습니다. 사용자 정의 리소스의 세부 정보를 보려면 다음 명령을 실행합니다.

```
oc get search search-v2-operator -o yaml
```

검색 Operator는 **search-v2-operator** 사용자 정의 리소스를 감시하고 변경 사항을 조정하고 활성 Pod를 업데이트합니다. 구성에 대한 다음 설명을 확인합니다.

- **PostgreSQL 데이터베이스 스토리지:**

Red Hat Advanced Cluster Management를 설치하면 **PostgreSQL** 데이터를 빈 디렉터리(**empty directory**) 볼륨에 저장하도록 **PostgreSQL** 데이터베이스가 구성됩니다. 빈 디렉터리 크

기가 제한된 경우 PostgreSQL 데이터를 PVC(영구 볼륨 클레임)에 저장하여 검색 성능을 향상시킬 수 있습니다. Red Hat Advanced Cluster Management Hub 클러스터에서 스토리지 클래스를 선택하여 검색 데이터를 백업할 수 있습니다. 예를 들어 gp2 스토리지 클래스를 선택하는 경우 구성은 다음 예와 유사할 수 있습니다.

```
apiVersion: search.open-cluster-management.io/v1alpha1
kind: Search
metadata:
  name: search-v2-operator
  namespace: open-cluster-management
  labels:
    cluster.open-cluster-management.io/backup: ""
spec:
  dbStorage:
    size: 10Gi
    storageClassName: gp2
```

이 구성은 gp2-search 라는 PVC를 생성하고 search-postgres Pod에 마운트됩니다. 기본적으로 스토리지 크기는 10Gi 입니다. 스토리지 크기를 수정할 수 있습니다. 예를 들어 20Gi 는 약 200개의 관리 클러스터에 충분할 수 있습니다.

- Pod 메모리 또는 CPU 요구 사항, 복제본 수 및 4개의 검색 Pod(indexer,database,queryapi 또는 수집기 Pod)에 대한 로그 수준을 업데이트하여 비용을 최적화합니다. search-v2-operator 사용자 정의 리소스의 배포 섹션을 업데이트합니다. search-v2-operator 에서 관리하는 배포의 4가지는 개별적으로 업데이트할 수 있습니다. search-v2-operator 사용자 정의 리소스는 다음 파일과 유사할 수 있습니다.

```
apiVersion: search.open-cluster-management.io/v1alpha1
kind: Search
metadata:
  name: search-v2-operator
  namespace: open-cluster-management
spec:
  deployments:
    collector:
      resources: ①
      limits:
        cpu: 500m
        memory: 128Mi
      requests:
        cpu: 250m
        memory: 64Mi
    indexer:
      replicaCount: 3
    database: ②
      envVar:
        - name: POSTGRESQL_EFFECTIVE_CACHE_SIZE
          value: 1024MB
        - name: POSTGRESQL_SHARED_BUFFERS
          value: 512MB
```

```

- name: WORK_MEM
  value: 128MB
queryapi:
arguments: 3
- -v=3

```

1 1

인덱서, 데이터베이스, queryapi 또는 수집기 Pod에 리소스를 적용할 수 있습니다.

2 2

envVar 섹션에 여러 환경 변수를 추가하여 이름을 지정하는 각 변수의 값을 지정할 수 있습니다.

3

-v=3 인수를 추가하여 이전 4개의 Pod 중 하나에 대한 로그 수준 상세 정보 표시를 제어할 수 있습니다.

메모리 리소스가 인덱서 Pod에 적용되는 다음 예제를 참조하십시오.

```

indexer:
resources:
limits:
memory: 5Gi
requests:
memory: 1Gi

```

검색 Pod에 대한 노드 배치:

nodeSelector 매개변수 또는 tolerations 매개변수를 사용하여 검색 Pod 배치를 업데이트할 수 있습니다. 다음 예제 구성을 확인합니다.

```

spec:
dbStorage:
size: 10Gi
deployments:
collector: {}
database: {}
indexer: {}
queryapi: {}
nodeSelector:
node-role.kubernetes.io/infra: ""
tolerations:

```

- effect: NoSchedule
key: node-role.kubernetes.io/infra
operator: Exists

7.3. 추가 리소스

- [검색을 관리하는 방법에 대한 지침은 검색 관리를 참조하십시오. ???](#)
- [Kubernetes 콘솔용 Red Hat Advanced Cluster Management에 대한 자세한 내용은 웹 콘솔을 참조하십시오.](#)

7.4. 검색 관리

클러스터에서 리소스 데이터를 쿼리하려면 **search**를 사용합니다.

필수 액세스: 클러스터 관리자

다음 항목을 계속 읽습니다.

- [구성 가능한 검색 컬렉션 생성](#)
- [검색 콘솔 사용자 정의](#)
- [콘솔에서 쿼리](#)
- [관리 클러스터에서 **klusterlet-addon-search** 배포 업데이트](#)

7.4.1. 구성 가능한 검색 컬렉션 생성

allow 및 **deny list** 섹션에 리소스를 나열하여 클러스터에서 수집하는 **Kubernetes** 리소스를 정의하기 위해 **search-collector-config ConfigMap**을 생성합니다. **ConfigMap** 내의 **data.AllowedResources** 및 **data.DeniedResources** 섹션에 있는 리소스를 나열합니다. 다음 명령을 실행하여 리소스를 생성합니다.

```
oc apply -f yourconfigMapFile.yaml
```

ConfigMap은 다음 **YAML** 파일과 유사할 수 있습니다.

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: search-collector-config
  namespace: <namespace where search-collector add-on is deployed>
data:
  AllowedResources: |-
    - apiGroups:
      - "*"
      resources:
        - services
        - pods
    - apiGroups:
      - admission.k8s.io
      - authentication.k8s.io
      resources:
        - "*"
  DeniedResources: |-
    - apiGroups:
      - "*"
      resources:
        - secrets
    - apiGroups:
      - admission.k8s.io
      resources:
        - policies
        - iampolicies
        - certificatepolicies
  
```

이전 **ConfigMap** 예제에서는 모든 **apiGroups** 에서 서비스와 **Pod**를 수집할 수 있도록 허용하는 동시에 **admission.k8s.io** 및 **authentication.k8s.io** **apiGroups** 에서 모든 리소스를 수집할 수 있습니다. 동시에 **ConfigMap** 예제에서는 정책, **iam policies** 및 **certificatepolicies** 의 **apiGroup admission.k8s.io** 의 수집을 방지하는 동안 모든 **apiGroups** 의 시크릿 중앙 컬렉션을 방지합니다.

참고: **ConfigMap**을 제공하지 않으면 기본적으로 모든 리소스가 수집됩니다. **AllowedResources** 만 제공하는 경우 **AllowedResources** 에 나열되지 않은 모든 리소스가 자동으로 제외됩니다. **AllowedResources** 및 **DeniedResources** 에 동시에 나열된 리소스도 제외됩니다.

7.4.2. 검색 콘솔 사용자 정의

OpenShift Container Platform 콘솔에서 검색 결과 제한을 사용자 지정할 수 있습니다. **multicluster-engine** 네임스페이스에서 **console-mce-config** 를 업데이트합니다. 이러한 설정은 모든 사용자에게 적용되며 성능에 영향을 미칠 수 있습니다. 다음 성능 매개변수 설명을 확인합니다.

-

192.0.2.D_SEARCH_LIMIT - 사용자당 저장된 최대 검색 양입니다. 기본적으로 각 사용자에게 대해 저장된 검색 수가 10개로 제한됩니다. 기본값은 10입니다. 제한을 업데이트하려면 **console-config ConfigMap: 192.0.2. D_SEARCH_LIMIT: x** 에 다음 키 값을 추가합니다.

- **SEARCH_RESULT_LIMIT** - 콘솔에 표시되는 검색 결과의 최대 양입니다. 기본값은 1000입니다. 이 제한을 제거하려면 -1로 설정합니다.
- **SEARCH_AUTOCOMPLETE_LIMIT** - 검색 표시줄 **typeahead**에 대해 검색된 최대 제안 수입니다. 기본값은 10,000입니다. 이 제한을 제거하려면 -1로 설정합니다.

OpenShift Container Platform 콘솔에서 다음 **patch** 명령을 실행하여 검색 결과를 100개의 항목으로 변경합니다.

```
oc patch configmap console-mce-config -n multicluster-engine --type merge -p '{"data": {"SEARCH_RESULT_LIMIT": "100"}}'
```

7.4.3. 콘솔에서 쿼리

검색 상자에 텍스트 값을 입력하면 이름 또는 네임스페이스와 같은 속성의 값이 있는 모든 항목이 포함됩니다. 사용자는 빈 공간이 포함된 값을 검색할 수 없습니다.

보다 구체적인 검색 결과를 보려면 검색에 속성 선택기를 포함합니다. 더 정확한 검색 범위를 위해 속성에 관련 값을 결합할 수 있습니다. 예를 들어 **dev** 클러스터에서 **"red"** 문자열과 일치하는 결과를 수신하려면 **cluster:dev red** 를 검색합니다.

검색을 사용하여 쿼리를 만들려면 다음 단계를 완료합니다.

1. 탐색 메뉴에서 검색을 클릭합니다.
2. 검색 상자에 단어를 입력한 다음 검색에서 해당 값이 포함된 리소스를 찾습니다.
 - 리소스를 검색할 때 원래 검색 결과와 관련된 다른 리소스가 수신되므로 리소스가 시스템의 다른 리소스와 상호 작용하는 방식을 시각화하는 데 도움이 됩니다.
 - 검색에서는 각 클러스터를 반환하고 검색한 리소스로 나열합니다. 허브 클러스터의 리소스의 경우 클러스터 이름이 **local-cluster** 로 표시됩니다.

- 검색 결과는 종류별로 그룹화되며 각 리소스 유형은 테이블에 그룹화됩니다.
- 검색 옵션은 클러스터 오브젝트에 따라 다릅니다.
- 특정 라벨을 사용하여 결과를 구체화할 수 있습니다. 레이블을 쿼리할 때 검색은 대소문자를 구분합니다. 필터링은 선택할 수 있는 다음 예제(이름,네임스페이스,상태 및 기타 리소스 필드)를 참조하십시오. 자동 완성은 검색을 구체화하기 위한 제안을 제공합니다. 다음 예제를 참조하십시오.
- 모든 Pod 리소스를 찾으려면 `kind:pod` 와 같은 단일 필드를 검색합니다.
- `kind:pod namespace:default` 와 같은 여러 필드를 검색하여 기본 네임스페이스에서 Pod를 찾습니다.

참고:

- `>`, `>=`, `<`, `<=`, `!=` 와 같은 문자를 사용하여 조건으로 검색할 수도 있습니다.
- 여러 값을 사용하여 둘 이상의 속성 선택기를 검색하면 검색에서 쿼리된 값 중 하나를 반환합니다. 다음 예제를 확인합니다.
- `kind:pod name:a` 를 검색하면 `a` 라는 모든 Pod가 반환됩니다.
- `kind:pod name:a,b` 를 검색하면 `a` 또는 `b` 라는 모든 Pod가 반환됩니다.
- `kind:pod status:!Running` 을 검색하여 상태가 `Running` 이 아닌 모든 Pod 리소스를 찾습니다.
- `kind:pod restarts:>1` 을 검색하여 최소 두 번 다시 시작한 모든 Pod를 찾습니다.

3.

검색을 저장하려면 검색 저장 아이콘을 클릭합니다.

7.4.4. 관리 클러스터에서 `klusterlet-addon-search` 배포 업데이트

관리 클러스터에서 **Kubernetes** 오브젝트를 수집하기 위해 `klusterlet-addon-search Pod`는 검색이 활성화된 모든 관리 클러스터에서 실행됩니다. 이 배포는 `open-cluster-management-agent-addon` 네임스페이스에서 실행됩니다. 리소스가 많은 관리형 클러스터에는 `klusterlet-addon-search` 배포를 위해 더 많은 메모리가 필요할 수 있습니다.

관리 클러스터의 `klusterlet-addon-search Pod`의 리소스 요구 사항은 **Red Hat Advanced Cluster Management Hub** 클러스터의 `ManagedClusterAddon` 사용자 정의 리소스에 지정할 수 있습니다. 관리 클러스터 이름이 있는 각 관리 클러스터의 네임스페이스가 있습니다. 관리 클러스터 이름과 일치하는 네임스페이스에서 `ManagedClusterAddon` 사용자 정의 리소스를 편집합니다. 다음 명령을 실행하여 `xyz` 관리 클러스터에서 리소스 요구 사항을 업데이트합니다.

```
oc edit managedclusteraddon search-collector -n xyz
```

리소스 요구 사항을 주석으로 추가합니다. 다음 예제를 확인합니다.

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  annotations:
    addon.open-cluster-management.io/search_memory_limit: 2048Mi
    addon.open-cluster-management.io/search_memory_request: 512Mi
```

이 주석은 관리 클러스터의 리소스 요구 사항을 재정의하고 새 리소스 요구 사항을 사용하여 `Pod`를 자동으로 다시 시작합니다.

참고: 콘솔에서 **API Explorer**를 사용하여 관리 클러스터에 정의된 모든 리소스를 검색할 수 있습니다. 또는 다음 명령을 실행하여 모든 리소스를 검색할 수도 있습니다. `oc api-resources`

[Observing 환경 소개](#) 로 돌아갑니다.

8장. RED HAT INSIGHTS에서 관찰 기능 사용

Red Hat Insights는 **Red Hat Advanced Cluster Management** 관찰 기능과 통합되어 있으며 클러스터의 기존 또는 잠재적인 문제를 식별하는 데 도움이 됩니다. **Red Hat Insights**는 안정성, 성능, 네트워크 및 보안 위협을 식별, 우선 순위 및 해결하는 데 도움이 됩니다. **Red Hat OpenShift Container Platform**은 **OpenShift Cluster Manager**를 통해 클러스터 상태 모니터링을 제공합니다. **OpenShift Cluster Manager**는 클러스터의 상태, 사용량 및 크기에 대한 익명화되고 집계된 정보를 수집합니다. 자세한 내용은 **Red Hat Insights** 제품 설명서를 참조하십시오.

OpenShift 클러스터를 생성하거나 가져올 때 관리형 클러스터에서 익명화된 데이터가 **Red Hat**으로 자동 전송됩니다. 이 정보는 클러스터 상태 정보를 제공하는 인사이트를 생성하는 데 사용됩니다. **Red Hat Advanced Cluster Management** 관리자는 이 상태 정보를 사용하여 심각도에 따라 경고를 생성할 수 있습니다.

필수 액세스: 클러스터 관리자

8.1. 사전 요구 사항

- **Red Hat Insights**가 활성화되어 있는지 확인합니다. 자세한 내용은 [원격 상태 보고를 비활성화하려면 글로벌 클러스터 풀 시크릿 수정을 참조하십시오](#).
- **OpenShift Container Platform** 버전 4.0 이상을 설치합니다.
- **OpenShift Cluster Manager**에 등록된 Hub 클러스터 사용자는 **OpenShift Cluster Manager**의 모든 **Red Hat Advanced Cluster Management** 관리 클러스터를 관리할 수 있어야 합니다.

8.2. RED HAT ADVANCED CLUSTER MANAGEMENT 콘솔의 RED HAT INSIGHTS

계속 읽고 통합에 대한 기능 설명을 확인합니다.

- 클러스터 페이지에서 클러스터를 선택하는 경우 상태 카드에서 확인된 문제 수를 선택할 수 있습니다. 상태 카드에는 노드, 애플리케이션, 정책 위반 및 확인된 문제에 대한 정보가 표시됩니다. 확인된 문제 카드는 **Red Hat Insights**의 정보를 나타냅니다. 확인된 문제 상태에는 심각도 별 문제 수가 표시됩니다. 문제에 사용되는 분류 수준은 심각, 주요, 낮음 및 경고와 같은 심각도 범주입니다.
- 번호를 클릭하면 잠재적인 문제 측면 패널이 표시됩니다. 총 문제에 대한 요약 및 차트가 페

널에 표시됩니다. 검색 기능을 사용하여 권장 수정 사항을 검색할 수도 있습니다. 수정 옵션은 취약점에 대한 설명, 취약점이 연결된 카테고리 및 총 위험 요소를 표시합니다.

-

설명 섹션에서 취약점 링크를 선택할 수 있습니다. **How to fix** 탭을 선택하여 취약점을 해결하는 단계를 확인합니다. **Reason** 탭을 클릭하여 취약점이 발생한 이유를 확인할 수도 있습니다.

자세한 내용은 [정보 정책 보고서 관리](#)를 참조하십시오.

8.3. 인사이트 정책 보고서 관리

Red Hat Advanced Cluster Management for Kubernetes PolicyReports 는 *insights-client* 가 생성한 위반입니다. **PolicyReports** 는 인시던트 관리 시스템으로 전송되는 경고를 정의하고 구성하는 데 사용됩니다. 위반이 발생하면 **PolicyReport** 의 경고가 인시던트 관리 시스템으로 전송됩니다.

8.3.1. 인사이트 정책 보고서 검색

관리 클러스터에서 위반이 있는 특정 **Insights PolicyReport** 를 검색할 수 있습니다. 다음 단계를 완료합니다.

- 1.

Red Hat Advanced Cluster Management Hub 클러스터에 로그인합니다.

- 2.

콘솔 헤더에서 검색 아이콘을 클릭하여 검색 페이지로 이동합니다.

- 3.

kind:policyreport 쿼리를 입력합니다.

참고: **PolicyReport** 이름은 클러스터 이름과 일치합니다.

- 4.

Insights 정책 위반 및 카테고리를 사용하여 쿼리를 지정할 수 있습니다. **PolicyReport** 이름을 선택하면 연결된 클러스터의 세부 정보 페이지로 리디렉션됩니다. **Insights** 사이드바가 자동으로 표시됩니다.

- 5.

검색 서비스가 비활성화되고 인사이트를 검색하려면 **hub** 클러스터에서 다음 명령을 실행합니다.

```
oc get policyreport --all-namespaces
```

8.3.2. 콘솔에서 확인된 문제 보기

특정 클러스터에서 확인된 문제를 볼 수 있습니다. 다음 단계를 완료합니다.

1. **Red Hat Advanced Cluster Management** 클러스터에 로그인합니다.
2. 탐색 메뉴에서 개요 를 선택합니다.
3. 심각도를 선택하여 해당 심각도와 연결된 **PolicyReports** 를 확인합니다. 클러스터 문제 및 심각도에 대한 세부 정보는 클러스터 문제 요약 카드에서 표시됩니다.
 - a. 또는 탐색 메뉴에서 클러스터를 선택할 수 있습니다.
 - b. 자세한 내용을 보려면 표에서 관리 클러스터를 선택합니다.
 - c. 상태 카드에서 확인된 문제 수를 확인합니다.
4. 잠재적인 문제 수를 선택하여 문제에 대한 심각도 차트 및 권장 수정 사항을 확인합니다.
5. 취약점의 링크를 클릭하여 해결 방법 및 취약점 의 이유를 확인합니다.

참고: 문제가 해결된 후 30분마다 Red Hat Advanced Cluster Management에서 Red Hat Insights를 수신하며 Red Hat Insights는 2시간마다 업데이트됩니다.
6. **PolicyReport** 에서 경고 메시지를 보낸 구성 요소가 있는지 확인하십시오.
 - a. **Governance** 페이지로 이동하여 특정 **PolicyReport** 를 선택합니다.
 - b. **Status** 탭을 선택하고 세부 정보 보기를 클릭하여 **PolicyReport YAML** 파일을 확인합니다.
 - c.

위반을 보낸 구성 요소를 알려주는 **source** 매개 변수를 찾습니다. **value** 옵션은 **grc** 및 **insights** 입니다.

8.3.3. 추가 리소스

- **PolicyReports** 에 대한 사용자 정의 경고 규칙을 만드는 방법을 알아보려면 자세한 내용은 [Alertmanager 구성](#) 을 참조하십시오.