



Red Hat Advanced Cluster Management for Kubernetes 2.8

可觀測性

可觀測性

可觀測性

法律上の通知

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.

概要

可観測性サービスを有効にしてカスタマイズし、マネージドクラスターを最適化する方法

目次

第1章 可観測性サービスについて	3
1.1. 環境の監視	3
1.2. 可観測性サービスの有効化	10
第2章 可観測性のカスタマイズ	23
2.1. カスタムルールの作成	23
2.2. カスタムメトリックの追加	24
2.3. 外部エンドポイントへのメトリクスのエクスポート	26
2.4. 詳細 設定の追加	28
2.5. コンソールからの MULTICLUSTEROBSERVABILITY カスタムリソースレプリカの更新	29
2.6. ルート証明書のカスタマイズ	29
2.7. オブジェクトストアにアクセスするための証明書のカスタマイズ	29
2.8. データの表示およびデプロイメント	30
2.9. 可観測性の無効化	31
2.10. 関連情報	32
第3章 可観測性アラート	33
3.1. ALERTMANAGER の設定	33
3.2. アラートの転送	34
3.3. アラートをサイレントにする	35
3.4. アラートの抑制	36
3.5. 関連情報	36
第4章 GRAFANA ダッシュボードの使用	37
4.1. RED HAT ADVANCED CLUSTER MANAGEMENT FOR KUBERNETES ダッシュボード	37
4.2. GRAFANA 開発者インスタンスの設定	37
4.3. GRAFANA ダッシュボードの設計	38
4.4. GRAFANA 開発者インスタンスのアンインストール	40
4.5. 関連情報	40
4.6. GRAFANA でマネージドクラスターラベルを使用する	40
第5章 コンソールでの検索の概要	44
5.1. 検索コンポーネント	44
5.2. 検索のカスタマイズと設定	45
5.3. 関連情報	47
5.4. 検索の管理	47
第6章 RED HAT INSIGHTS での可観測性の使用	51
6.1. 前提条件	51
6.2. RED HAT ADVANCED CLUSTER MANAGEMENT コンソールからの RED HAT INSIGHTS	51
6.3. INSIGHTS POLICYREPORTS の管理	51

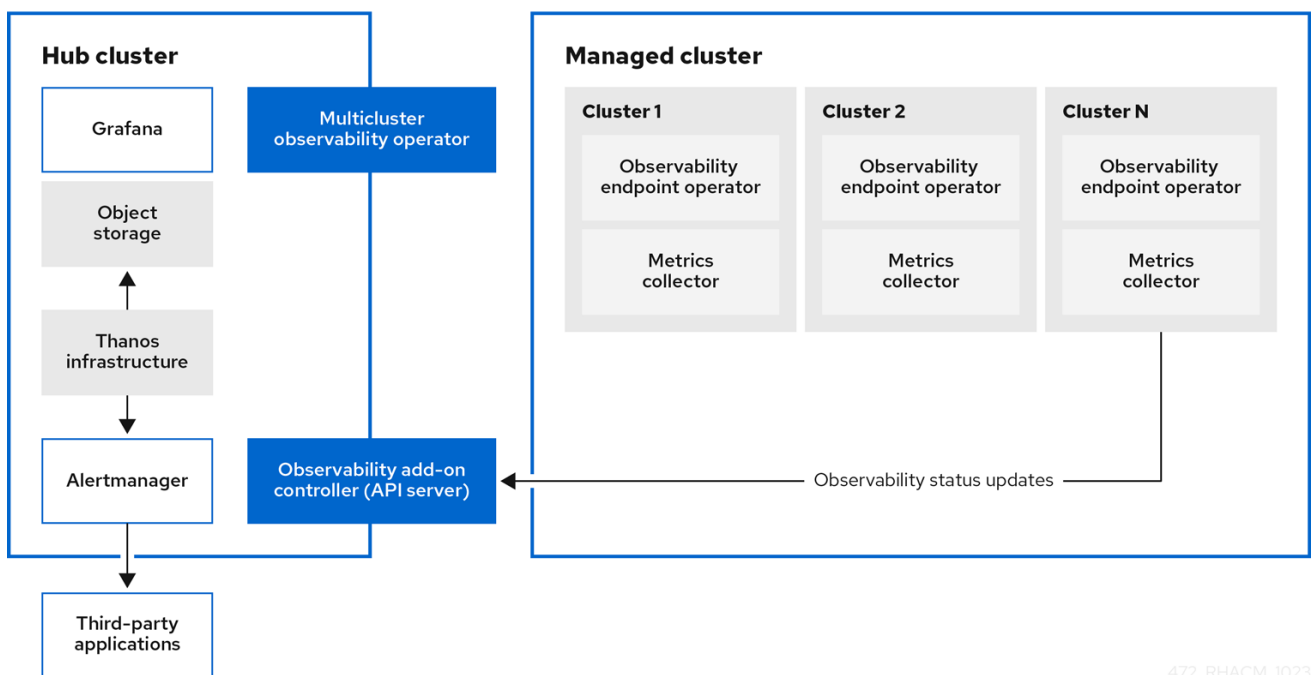
第1章 可観測性サービスについて

可観測性サービスを有効にすると、Red Hat Advanced Cluster Management for Kubernetes を使用して、マネージドクラスターに関する理解を深め、最適化することができます。この情報は、コストを節約し、不要なイベントを防ぐことができます。

- [環境の監視](#)
- [コンソールでの検索の概要](#)
- [Red Hat Insights での可観測性の使用](#)

1.1. 環境の監視

Red Hat Advanced Cluster Management for Kubernetes を使用して、マネージドクラスターに関する理解を深め、最適化することができます。ハブクラスターで可観測性サービス Operator (**multicluster-observability-operator**) を有効にして、マネージドクラスターの状態を監視します。以下のセクションでは、マルチクラスター可観測性サービスのアーキテクチャーについて説明します。



472_RHACM_1023

- [可観測性サービス](#)
- [サポート](#)
- [メトリックのタイプ](#)
- [可観測性 Pod の容量要求](#)
- [可観測性サービスで使用される永続ストア](#)
- [関連情報](#)

1.1.1. 可観測性サービス

デフォルトでは可観測性は、製品のインストール時に追加されますが、有効にはなっていません。永続ストレージの要件により、可観測性サービスはデフォルトで有効にはなりません。可観測性については、[サポート セクション](#)を参照してください。

サービスを有効にすると、**observability-endpoint-operator** はインポートまたは作成された各クラスターに自動的にデプロイされます。このコントローラーは、Red Hat OpenShift Container Platform Prometheus からデータを収集してから、Red Hat Advanced Cluster Management ハブクラスターに送信します。ハブクラスターが **local-cluster** として自己インポートする場合は、可観測性もそのクラスターで有効になり、メトリックがハブクラスターから収集されます。

可観測性サービスは、Prometheus Alertmanager のインスタンスをデプロイすることで、サードパーティーのアプリケーションでのアラートの転送が可能になります。また、ダッシュボード (静的) またはデータ検索を使用してデータの可視化を有効にする Grafana のインスタンスも含まれます。Red Hat Advanced Cluster Management は、Grafana のバージョン 8.5.20 をサポートします。Grafana ダッシュボードを設計することもできます。詳細は、[Grafana ダッシュボードの設計](#)を参照してください。カスタムのレコーディングルールまたはアラートルールを作成して、可観測性サービスをカスタマイズできます。

1.1.2. サポート

- Red Hat Advanced Cluster Management は、Red Hat OpenShift Data Foundation (以前の Red Hat OpenShift Container Storage) によってテストされ、完全にサポートされています。
- Red Hat Advanced Cluster Management は、S3 API と互換性のあるユーザー提供のオブジェクトストレージにおけるマルチクラスター可観測性 Operator の機能をサポートします。可観測性サービスは、Thanos がサポートする安定したオブジェクトストアを使用します。
- Red Hat Advanced Cluster Management はビジネス的に妥当な範囲内で、根本原因の特定を支援します。サポートチケットが発行され、根本的な原因がお客様が提供した S3 互換オブジェクトストレージの結果であると判断された場合は、カスタマーサポートチャネルを使用して問題を解決する必要があります。
- Red Hat Advanced Cluster Management は、お客様が起票したサポートチケットの根本的な原因が S3 互換性のあるオブジェクトストレージプロバイダーである場合に、問題修正サポートの確約はありません。

1.1.3. メトリックのタイプ

デフォルトで、OpenShift Container Platform は Telemetry サービスを使用してメトリックを Red Hat に送信します。**acm_managed_cluster_info** は、Red Hat Advanced Cluster Management で利用でき、Telemetry に含まれていますが、Red Hat Advanced Cluster Management **Observe 環境の概要** ダッシュボードには表示されません。

フレームワークでサポートされているメトリックタイプの次の表を参照してください。

表1.1パラメーターの表

メトリック名	メトリックのタイプ	ラベル/タグ	ステータス
acm_managed_cluster_info	ゲージ	hub_cluster_id、managed_cluster_id、vendor、cloud、version、available、created_via、core_worker、socket_worker	Stable

メトリック名	メトリックのタイプ	ラベル/タグ	ステータス
<code>config_policies_evaluation_duration_seconds_bucket</code>	ヒストグラム	なし	Stable。詳細は、 ガバナンスメトリック を参照してください。
<code>config_policies_evaluation_duration_seconds_count</code>	ヒストグラム	なし	Stable。詳細は、 ガバナンスメトリック を参照してください。
<code>config_policies_evaluation_duration_seconds_sum</code>	ヒストグラム	なし	Stable。詳細は、 ガバナンスメトリック を参照してください。
<code>policy_governance_info</code>	ゲージ	<code>type</code> 、 <code>policy</code> 、 <code>policy_namespace</code> 、 <code>cluster_namespace</code>	Stable。詳細は、 ガバナンスメトリック を参照してください。
<code>policyreport_info</code>	ゲージ	<code>managed_cluster_id</code> 、 <code>category</code> 、 <code>policy</code> 、 <code>result</code> 、 <code>severity</code>	Stable。詳細は、 insight_PolicyReports_の管理 を参照してください。
<code>search_api_db_connection_failed_total</code>	カウンター	なし	Stable。コンソールでの 検索についてのコンポーネントの検索 セクションを参照してください。
<code>search_api_dbquery_duration_seconds</code>	ヒストグラム	なし	Stable。コンソールでの 検索についてのコンポーネントの検索 セクションを参照してください。
<code>search_api_requests</code>	ヒストグラム	なし	Stable。コンソールでの 検索についてのコンポーネントの検索 セクションを参照してください。
<code>search_indexer_request_count</code>	カウンター	なし	Stable。コンソールでの 検索についてのコンポーネントの検索 セクションを参照してください。

メトリック名	メトリックのタイプ	ラベル/タグ	ステータス
search_indexer_request_duration	ヒストグラム	なし	Stable。コンソールでの検索についてのコンポーネントの検索セクションを参照してください。
search_indexer_requests_in_flight	ゲージ	なし	Stable。コンソールでの検索についてのコンポーネントの検索セクションを参照してください。
search_indexer_request_size	ヒストグラム	なし	Stable。コンソールでの検索についてのコンポーネントの検索セクションを参照してください。

1.1.4. 可観測性 Pod の容量要求

可観測性サービスをインストールするには、可観測性コンポーネントで 2701mCPU および 11972Mi のメモリーが必要です。以下の表は、**observability-addons** が有効なマネージドクラスター 5 台の Pod 容量要求のリストです。

表1.2 可観測性 Pod の容量要求

デプロイメントまたは StatefulSet	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計メモリー
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

デプロイメントまたは StatefulSet	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計メモリー
observability-observatorium-api	observatorium-api	20	128	2	40	256
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
	exporter	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

デプロイメントまたは StatefulSet	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計メモリー
observability-thanos-rule	thanos-rule	50	512	3	150	1536
	configmap-reloader	4	25	3	12	75
observability-thanos-store-memcached	memcached	45	128	3	135	384
	exporter	5	50	3	15	150
observability-thanos-store-shard	thanos-store	100	1024	3	300	3072

1.1.5. 可観測性サービスで使用される永続ストア

重要: 永続ストレージにローカルボリュームを使用するローカルストレージ Operator またはストレージクラスを使用しないでください。再起動後に Pod が別のノードで再起動されると、データが失われる可能性があります。これが発生すると、Pod はノード上のローカルストレージにアクセスできなくなります。データの損失を回避するために、**receive** Pod および **receive** Pod の永続ボリュームにアクセスできることを確認してください。

Red Hat Advanced Cluster Management をインストールするときは、次の永続ボリューム (PV) を作成して、Persistent Volume Claims (PVC) を自動的にアタッチできるようにする必要があります。デフォルトのストレージクラスが指定されていない場合、またはデフォルト以外のストレージクラスを使用して PV をホストする場合は、**MultiClusterObservability** カスタムリソースでストレージクラスを定義する必要があります。Prometheus が使用するものと同様の、ブロックストレージを使用することを推奨します。また、**alertmanager**、**thanos-compact**、**thanos-ruler**、**thanos-receive-default**、および **thanos-store-shard** の各レプリカには、独自の PV が必要です。次の表を参照します。

表1.3 永続ボリュームの表リスト

永続ボリューム名	目的
alertmanager	alertmanager は nflog データおよび通知なしのアラートをストレージに保存します。 nflog は、通知されたレシーバーおよび、アクティブな通知と解決済みの通知、通知により特定されたコンテンツのハッシュダイジェストについての追記専用のログです。

thanos-compact	コンパクターは、処理の中間データとバケット状態キャッシュの保存にローカルのディスク領域が必要です。必要な領域は、下層にあるブロックサイズにより異なります。コンパクターには、すべてのソースブロックをダウンロードして、ディスクで圧縮ブロックを構築するのに十分な領域が必要です。ディスク上のデータは、次の再起動までに安全に削除でき、最初の試行でクラッシュループコンパクターの停止が解決されるはずですが、次の再起動までにバケットの状態キャッシュを効果的に使用するには、コンパクターの永続ディスクを用意することが推奨されます。
thanos-rule	thanos ruler は、固定の間隔でクエリーを発行して、選択したクエリー API に対して Prometheus 記録およびアラートルールを評価します。ルールの結果は、Prometheus 2.0 ストレージ形式でディスクに書き込まれます。このステートフルセットで保持されるデータの期間 (時間または日) は、API バージョンの observability.open-cluster-management.io/v1beta1 で修正されました。 observability.open-cluster-management.io/v1beta2: RetentionInLocal の API パラメーターとして公開されました。
thanos-receive-default	Thanos receiver は、受信データ (Prometheus リモート書き込みリクエスト) を受け入れて Prometheus TSDB のローカルインスタンスに書き込みます。TSDB ブロックは定期的 (2 時間) に、長期的に保存および圧縮するためにオブジェクトストレージにアップロードされます。ローカルキャッシュを実行するこのステートフルセットで保持される期間 (時間または日) は、API バージョン observability.open-cluster-management.io/v1beta で修正されました。 observability.open-cluster-management.io/v1beta2: RetentionInLocal の API パラメーターとして公開されました。
thanos-store-shard	これは、主に API ゲートウェイとして機能するため、大量のローカルディスク容量は必要ありません。これは、起動時に Thanos クラスタに参加して、アクセスできるデータを広告します。ローカルディスク上のすべてのリモートブロックに関する情報のサイズを小さく保ち、バケットと同期させます。このデータは通常、起動時間が長くなると、再起動時に安全に削除できます。

注記: 時系列の履歴データはオブジェクトストアに保存されます。Thanos は、オブジェクトストレージをメトリクスおよび関連するメタデータのプライマリストレージとして使用します。オブジェクトストレージおよび downsampling 機能の詳細は、**可観測性サービスの有効化** を参照してください。

1.1.6. 関連情報

- 可観測性有効化の詳細は、[可観測性サービスの有効化](#) を参照してください。
- 可観測性サービスの設定、メトリクスおよびその他のデータの表示方法は、[可観測性のカスタマイズ](#) を参照してください。
- [Grafana ダッシュボードの使用](#) を参照してください。
- OpenShift Container Platform ドキュメントで、Telemetry を使用して収集されて送信されるメトリクスのタイプについて確認します。詳細は、[Telemetry で収集される情報](#) を参照してください。
- 詳細は、[ガバナンスメトリック](#) を参照してください。
- [insight PolicyReports の管理](#) を参照してください。
- [Prometheus 記録ルール](#) を参照してください。
- [Prometheus アラートルール](#) も参照してください。
- [コンソールでの検索の概要](#) に戻って確認してください。

1.2. 可観測性サービスの有効化

可観測性サービス (**multicluster-observability-operator**) でマネージドクラスターの状態を監視します。

必要なアクセス権: クラスター管理者、**open-cluster-management:cluster-manager-admin** ロール、または S3 管理者。

- [前提条件](#)
- [コマンドラインからの可観測性の有効化](#)
- [MultiClusterObservability カスタムリソースの作成](#)
- [Red Hat OpenShift Container Platform コンソールからの可観測性の有効化](#)
- [外部メトリッククエリーの使用](#)
- [可観測性の無効化](#)

1.2.1. 前提条件

- Red Hat Advanced Cluster Management for Kubernetes がインストールされている。詳細は、[ネットワーク接続時のオンラインインストール](#) を参照してください。
- デフォルトのストレージクラスが指定されていない場合は、**MultiClusterObservability** カスタムリソースでストレージクラスを定義する必要があります。
- ハブクラスターへの直接的なネットワークアクセスが必要です。ロードバランサーおよびプロキシへのネットワークアクセスはサポートされていません。詳細は、[Networking](#) を参照してください。
- ストレージソリューションを作成するようにオブジェクトストアが設定されている。Red Hat Advanced Cluster Management は、安定したオブジェクトストアで以下のクラウドプロバイダーをサポートします。

- [Amazon Web Services S3 \(AWS S3\)](#)
- [Red Hat Ceph \(S3 互換 API\)](#)
- [Google Cloud Storage](#)
- [Azure ストレージ](#)
- Red Hat OpenShift Data Foundation (旧称: Red Hat OpenShift Container Storage)
- [Red Hat OpenShift on IBM\(ROKS\)](#)
重要: オブジェクトストアを設定する場合は、機密データを永続化する時に必要な暗号化要件を満たすようにしてください。可観測性サービスは、Thanos がサポートする安定したオブジェクトストアを使用します。

1.2.2. コマンドラインからの可観測性の有効化

MultiClusterObservability カスタムリソースを作成して可観測性サービスを有効にします。可観測性を有効にする前に、[可観測性 Pod の容量要求](#) を参照してください。

注記:

- Red Hat Advanced Cluster Management が管理する OpenShift Container Platform マネージドクラスターで可観測性を有効または無効にすると、可観測性エンドポイント Operator は、ローカル Prometheus を自動的に再起動する **alertmanager** 設定を追加して **cluster-monitoring-config** ConfigMap を更新します。
- 可観測性エンドポイント Operator は、ローカル Prometheus を自動的に再起動する **アラートマネージャー** 設定を別途追加して、**cluster-monitoring-config** config map を更新します。したがって、OpenShift Container Platform マネージドクラスターに **alertmanager** 設定を挿入すると、Prometheus メトリクスの保持に関連する設定が削除されます。

可観測性サービスを有効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management ハブクラスターにログインします。
2. 以下のコマンドを使用して可観測性サービスの namespace を作成します。

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

3. プルシークレットを生成します。Red Hat Advanced Cluster Management が **open-cluster-management** namespace にインストールされている場合は、以下のコマンドを実行します。

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

multiclusterhub-operator-pull-secret が namespace に定義されていない場合には、**pull-secret** を **openshift-config** namespace から **open-cluster-management-observability** namespace にコピーします。以下のコマンドを実行します。

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

次に **open-cluster-management-observability** namespace でプルリクエストを作成して、以下のコマンドを実行します。

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

重要: OpenShift Container Platform ドキュメントを使用してクラスターのグローバルプルシークレットを変更する場合は、必ず可観測性 namespace のグローバルプルシークレットも更新してください。詳細は、[Updating the global pull secret](#) を参照してください。

- お使いのクラウドプロバイダーのオブジェクトストレージのシークレットを作成します。シークレットには、ストレージソリューションへの認証情報を追加する必要があります。たとえば、以下のコマンドを実行します。

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

サポートされるオブジェクトストアのシークレットの例を以下に示します。

- Amazon S3 または S3 と互換性のある場合、シークレットは以下のファイルのようになります。

```
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 プロトコルなしで URL を入力します。次の URL のような Amazon S3 エンドポイントの URL を入力します: **example.redhat.com:443**。

詳細は、[Amazon Simple Storage Service ユーザーガイド](#) を参照してください。

- Google の場合は、以下のファイルのようになります。

```
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

```

詳細は、[Google Cloud Storage とは](#) を参照してください。

- Azure の場合は、以下のファイルのようになります。

```

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** パスを使用する場合、エンドポイント認証はシステム割り当てのマネージド ID を使用して完了します。値はエンドポイント <https://<storage-account-name>.blob.core.windows.net> のようにする必要があります。

user_assigned_id パスを使用する場合は、ユーザー割り当てマネージド ID を使用してエンドポイント認証が完了します。**user_assigned_id** を使用する場合、**msi_resource** エンドポイントのデフォルト値は https:<storage_account>.<endpoint> です。詳細は、[Azure Storage のドキュメント](#) を参照してください。

注記: Azure を Red Hat OpenShift Container Platform クラスターのオブジェクトストレージとして使用する場合には、クラスターに関連付けられたストレージアカウントはサポートされません。新規ストレージアカウントを作成する必要があります。

- Red Hat OpenShift Data Foundation では、シークレットは以下のファイルのようになります。

```

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

```

- 1 プロトコルなしで URL を入力します。次の URL のような Red Hat OpenShift Data Foundation エンドポイントの URL を入力します: **example.redhat.com:443**。

詳細は、[Red Hat OpenShift Data Foundation](#) を参照してください。

- Red Hat OpenShift on IBM (ROKS) では、シークレットは以下のファイルのようになります。

```
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 1
      insecure: true
      access_key: YOUR_ROKS_ACCESS_KEY
      secret_key: YOUR_ROKS_SECRET_KEY
```

- 1 プロトコルなしで URL を入力します。次の URL のような Red Hat OpenShift Data Foundation エンドポイントの URL を入力します: **example.redhat.com:443**。

詳細は、IBM Cloud のドキュメント [Cloud Object Storage](#) を参照してください。サービスの認証情報を使用してオブジェクトストレージに接続するようにしてください。詳細は、IBM Cloud のドキュメント、[Cloud Object Store](#) および [Service Credentials](#) を参照してください。

- Amazon S3 または S3 と互換性のあるストレージの場合、AWS Security Token Service (AWS STS) で生成された短期間の限定特権認証情報を使用することもできます。詳細については、[AWS Security Token Service ドキュメント](#) を参照してください。AWS Security Service を使用してアクセスキーを生成するには、次の追加の手順が必要です。
 - S3 バケットへのアクセスを制限する IAM ポリシーを作成します。
 - OpenShift Container Platform サービスアカウントの JWT トークンを生成するための信頼ポリシーを持つ IAM ロールを作成します。
 - S3 バケットへのアクセスが必要な可観測性サービスアカウントのアノテーションを指定します。Red Hat OpenShift Service on AWS (ROSA) クラスタで可観測性を設定して AWS STS トークンを使用する方法の例は [環境の設定](#) ステップで確認できます。詳細は、[Red Hat OpenShift Service on AWS \(ROSA\)](#) を参照してください。また、STS トークンを使用するための要件とセットアップの詳細な説明については、[ROSA with STS の説明](#) を参照してください。

1.2.2.1. AWS Security Service を使用したアクセスキーの生成

AWS Security Service を使用してアクセスキーを生成するには、次の手順を実行します。

1. AWS 環境をセットアップします。以下のコマンドを実行します。

```
export POLICY_VERSION=$(date +"%m-%d-%y")
export TRUST_POLICY_VERSION=$(date +"%m-%d-%y")
export CLUSTER_NAME=<my-cluster>
export S3_BUCKET=$CLUSTER_NAME-acm-observability
export REGION=us-east-2
export NAMESPACE=open-cluster-management-observability
export SA=tbdc
export SCRATCH_DIR=/tmp/scratch
export OIDC_PROVIDER=$(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. 次のコマンドで S3 バケットを作成します。

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

3. S3 バケットにアクセスするための **s3-policy** JSON ファイルを作成します。以下のコマンドを実行します。

```
{
  "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. 次のコマンドでポリシーを適用します。

```
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 ファイルを作成します。以下のコマンドを実行します。

```
{
  "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. 次のコマンドを使用して、AWS Prometheus と CloudWatch のロールを作成します。

```
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. ポリシーをロールにアタッチします。以下のコマンドを実行します。

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

シークレットは、次のファイルのようになる場合があります。**config** セクションでは **signature_version2: false** が指定されており、**access_key** と **secret_key** は指定されていません。

```
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 カスタムリソースの作成** セクションで説明されているように、**MultiClusterObservability** カスタムリソースを使用するときに、サービスアカウントアノテーションを指定します。
9. 以下のコマンドを使用して、クラウドプロバイダーの S3 アクセスキーおよびシークレットキーを取得できます。シークレットの **base64** 文字列のデコード、編集、エンコードが必要です。

```
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. 次のデプロイメントとステートフルセットの Pod をチェックして、可観測性が有効になっていることを確認します。次の情報が表示される場合があります。

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

1.2.3. MultiClusterObservability カスタムリソースの作成

MultiClusterObservability カスタムリソースを使用して、さまざまなコンポーネントの永続ボリュームのストレージサイズを指定します。**MultiClusterObservability** カスタムリソースの最初の作成時にストレージサイズを設定する必要があります。デプロイ後にストレージサイズ値を更新すると、ストレージクラスが動的ボリューム拡張をサポートしている場合にのみ変更が反映されます。詳細は、Red Hat OpenShift Container Platform ドキュメントの [永続ボリュームの拡張](#) を参照してください。

次の手順を実行して、ハブクラスターに **MultiClusterObservability** カスタムリソースを作成します。

1. **multiclusterobservability_cr.yaml** という名前の **MultiClusterObservability** カスタムリソースの YAML ファイルを作成します。
可観測性については、以下のデフォルト YAML ファイルを確認してください。

```
apiVersion: observability.open-cluster-management.io/v1beta2
kind: MultiClusterObservability
metadata:
  name: observability
spec:
  observabilityAddonSpec: {}
  storageConfig:
```

```
metricObjectStorage:
  name: thanos-object-storage
  key: thanos.yaml
```

advanced セクションで **retentionConfig** パラメーターの値を変更する必要がある場合があります。詳細は、[Thanos Downsampling resolution and retention](#) を参照してください。マネージドクラスターの数によっては、ステートフルセットのストレージの量を更新する必要がある場合があります。S3 バケットが STS トークンを使用するように設定されている場合は、S3 ロールで STS を使用するようにサービスアカウントにアノテーションを付けます。次の設定を表示します。

```
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
```

詳細は、[可観測性 API](#) を参照してください。

- インフラストラクチャーマシンセットにデプロイするには、**MultiClusterObservability** YAML の **nodeSelector** を更新して、セットのラベルを設定する必要があります。YAML の内容は以下ようになります。

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

詳細は、[インフラストラクチャーマシンセットの作成](#) を参照してください。

- 以下のコマンドを実行して可観測性 YAML をクラスターに適用します。

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

Thanos、Grafana および Alertmanager の **open-cluster-management-observability** namespace にすべての Pod が作成されます。Red Hat Advanced Cluster Management ハブクラスターに接続されたマネージドクラスターはすべて、メトリクスを Red Hat Advanced Cluster Management の可観測性サービスに送信できます。

- Grafana ダッシュボードを起動して可観測性サービスが有効になっていることを検証し、データが入力されていることを確認します。コンソールの **概要** ページまたは **クラスター** ページから、コンソールヘッダーの近くにある **Grafana リンク** をクリックします。
注記: 可観測性データを収集しないように特定のマネージドクラスターを除外するには、クラスターに **observability: disabled** クラスターラベルを追加します。

可観測性サービスを有効化します。可観測性サービスを有効にすると、次の機能が開始されます。

- マネージドクラスターからのアラートマネージャーはすべて、Red Hat Advanced Cluster Management ハブクラスターに転送されます。
- Red Hat Advanced Cluster Management ハブクラスターに接続されたマネージドクラスターはすべて、アラートを Red Hat Advanced Cluster Management の可観測性サービスに送信できます。Red Hat Advanced Cluster Management Alertmanager を設定して、重複を排除してグループ化し、アラートをメール、PagerDuty、または OpsGenie などの適切なレシーバー統合にルーティングすることができます。アラートの通知解除や抑制にも対応できます。

注記: Red Hat Advanced Cluster Management ハブクラスター機能へのアラート転送は、Red Hat OpenShift Container Platform バージョン 4.8 以降のマネージドクラスターでのみサポートされます。可観測性を有効にして Red Hat Advanced Cluster Management をインストールすると、OpenShift Container Platform v4.8 以降のアラートは自動的にハブクラスターに転送されます。詳細は、[送信アラート](#) を参照してください。

- 次の URL を使用して OpenShift Container Platform 3.11 Grafana ダッシュボードにアクセスします: [https://\\$ACM_URL/grafana/dashboards](https://$ACM_URL/grafana/dashboards)。OCP 3.11 という名前のフォルダーを選択して、OpenShift Container Platform 3.11 ダッシュボードを表示します。

1.2.4. Red Hat OpenShift Container Platform コンソールからの可観測性の有効化

オプションで、Red Hat OpenShift Container Platform コンソールから可観測性を有効にし、**open-cluster-management-observability** という名前のプロジェクトを作成します。**open-cluster-management-observability** プロジェクトに、**multiclusterhub-operator-pull-secret** という名前のイメージプルシークレットを作成してください。

open-cluster-management-observability プロジェクトに **thanos-object-storage** という名前のオブジェクトストレージシークレットを作成します。オブジェクトストレージシークレットの詳細を入力し、**Create** をクリックします。シークレットの例を表示するには、**可観測性の有効化** セクションの手順 4 を参照してください。

MultiClusterObservability カスタムリソースインスタンスを作成します。**Observability components are deployed and running** のメッセージが表示されると、OpenShift Container Platform から可観測性サービスが正常に有効化されています。

1.2.4.1. Thanos バージョンの検証

コマンドラインインターフェイス (CLI) で Thanos のバージョンを確認します。Thanos がクラスターにデプロイされたら、コマンドラインインターフェイス (CLI) から Thanos のバージョンを確認します。

ハブクラスターにログインした後、可観測性 Pod で次のコマンドを実行して Thanos バージョンを受け取ります。

```
thanos --version
```

Thanos のバージョンが表示されます。

1.2.4.2. 外部メトリッククエリーの使用

可観測性には、外部 API があり、OpenShift ルート (**rbac-query-proxy**) を使用してメトリックをクエリーできます。以下のタスクを確認して、**rbac-query-proxy** ルートを使用します。

- 以下のコマンドを使用して、ルートの詳細を取得できます。

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

- **rbac-query-proxy** ルートにアクセスするには、OpenShift OAuth アクセストークンが必要です。トークンは、namespace 取得のパーミッションがあるユーザーまたはサービスアカウントと関連付ける必要があります。詳細は、[ユーザーが所有する OAuth アクセストークンの管理](#) について参照してください。
- デフォルトの CA 証明書を取得し、**tls.crt** キーの内容をローカルファイルに保存します。以下のコマンドを実行します。

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

- 以下のコマンドを実行してメトリックのクエリーを実行します。

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

注記: **QUERY_EXPRESSION** は標準の Prometheus クエリー式です。たとえば、前述のコマンドの URL を https://{PROXY_ROUTE_URL}/api/v1/query?query=cluster_infrastructure_provider に置き換えて、メトリクス **cluster_infrastructure_provider** をクエリーします。詳細は、[Prometheus のクエリー](#) を参照してください。

- 以下のコマンドを実行し、生成された証明書を使用して **proxy-byo-ca** シークレットおよび **proxy-byo-cert** シークレットを作成します。

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

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

1.2.4.3. 単一ノード OpenShift クラスターの動的メトリクス

動的メトリックコレクションは、特定の条件に基づく自動メトリック収集をサポートします。デフォルトで、SNO クラスターは Pod およびコンテナのリソースメトリックを収集しません。SNO クラスターが特定のリソース消費レベルに達すると、定義された詳細なメトリックが動的に収集されます。クラスターリソースの消費量が一定期間いきい値を一貫して下回ると、詳細なメトリック収集が停止します。

メトリックは、コレクションルールで指定されたマネージドクラスターの状態に基づいて動的に収集されます。これらのメトリックは動的に収集されるため、以下の Red Hat Advanced Cluster Management Grafana ダッシュボードではデータは表示されません。コレクションルールがアクティブになり、対応するメトリックが収集されると、以下のパネルには、コレクションルールが開始される期間のデータが表示されます。

- Kubernetes/コンピューティングリソース/namespace (Pod)
- Kubernetes/コンピューティングリソース/namespace (ワークロード)
- Kubernetes/コンピューティングリソース/ノード (Pod)
- Kubernetes/コンピューティングリソース/Pod

- Kubernetes/コンピューティングリソース/ワークロード

コレクションルールには、以下の条件が含まれます。

- 動的に収集するメトリックのセット。
- PromQL 式として記述された条件。
- コレクションの間隔。 **true** に設定する必要があります。
- 収集ルールを評価する必要があるクラスターを選択するための一致式。

デフォルトでは、コレクションルールは、30 秒ごとにマネージドクラスターで継続的に評価されるか、特定の間隔で評価されます。コレクションの間隔と時間間隔の最小値が優先されます。収集ルールの条件が **for** 属性で指定された期間持続すると、収集ルールが開始され、ルールで指定されたメトリクスがマネージドクラスターに自動的に収集されます。メトリクスの収集は、収集ルールの条件がマネージドクラスターに存在しなくなった後、開始してから少なくとも 15 分後に自動的に停止します。

収集ルールは、**collect_rules** という名前のパラメーターセクションとしてグループ化され、グループとして有効または無効にできます。Red Hat Advanced Cluster Management インストールには、コレクションルールグループ (**HighCPUUsage** および **HighMemoryUsage**) のデフォルトコレクションルール **SNOResourceUsage** が含まれます。**HighCPUUsage** コレクションルールは、ノードの CPU 使用率が 70% を超えると開始されます。**HighMemoryUsage** コレクションルールは、SNO クラスターの全体的なメモリー使用率が利用可能なノードメモリーの合計 70% を超えると開始されます。現在、上記のしきい値は固定されており、変更できません。コレクションルールが **for** 属性で指定された間隔を超えて開始すると、システムは **dynamic_metrics** セクションに指定されたメトリックの収集を自動的に開始します。

以下の YAML ファイルで、**collect_rules** セクションからの動的メトリックのリストを表示します。

```

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!=""

```

以下の例のように、**collect_rules.group** は **custom-allowlist** で無効にできます。**collect_rules.group** を無効にすると、メトリックコレクションは以前の動作に戻ります。これらのメトリックは定期的に、指定された間隔で収集されます。

```

collect_rules:
  - group: -SNOResourceUsage

```

データは、ルールを開始時のみ Grafana に表示されます。

1.2.5. 可観測性の無効化

observability リソースをアンインストールして、可観測性サービスを無効にします。OpenShift Container Platform コンソールナビゲーションから、**Operators > Installed Operators > Advanced Cluster Manager for Kubernetes** の順に選択します。**MultiClusterObservability** カスタムリソースを削除します。

可観測性サービスのカスタマイズ方法の詳細は、[可観測性のカスタマイズ](#) を参照してください。

第2章 可観測性のカスタマイズ

可観測性サービスが収集するデータのカスタマイズ、管理、および表示については、以下のセクションを参照してください。

must-gather コマンドで可観測性リソース用に作成される新規情報についてのログを収集します。詳細は、**トラブルシューティング** ドキュメントの **Must-gather** セクションを参照してください。

- [カスタムルールの作成](#)
- [カスタムメトリックの追加](#)
- [外部エンドポイントへのメトリクスのエクスポート](#)
- [詳細 設定の追加](#)
- [コンソールからの MultiClusterObservability カスタムリソースレプリカの更新](#)
- [ルート認定のカスタマイズ](#)
- [オブジェクトストアにアクセスするための証明書のカスタマイズ](#)
- [データの表示およびデプロイメント](#)
- [可観測性の無効化](#)

2.1. カスタムルールの作成

可観測性リソースに、Prometheus [レコードルール](#) および [アラートルール](#) を追加して、可観測性インストールのカスタムルールを作成します。

- レコードルールでは、必要に応じてコストの掛かる式を事前に計算するか、コンピュートできます。結果は新たな時系列のセットとして保存されます。
- アラートルールでは、アラートを外部サービスに送信する方法に基づいてアラート条件を指定する機能を提供します。
Prometheus でカスタムルールを定義してアラート条件を作成し、通知を外部メッセージングサービスに送信します。

注記: カスタムルールを更新すると、**observability-thanos-rule** Pod は自動的に再起動されます。

open-cluster-management-observability namespace に **thanos-ruler-custom-rules** という名前の ConfigMap を作成します。以下の例のように、キーは **custom_rules.yaml** という名前を指定する必要があります。設定には、複数のルールを作成できます。

- デフォルトでは、同梱のアラートルールは **open-cluster-management-observability** namespace の **thanos-ruler-default-rules** ConfigMap に定義されます。
たとえば、CPU の使用状況が定義値を超えた場合に通知するカスタムのアラートルールを作成できます。YAML の内容は以下のようになります。

```
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

```

- **thanos-ruler-custom-rules** ConfigMap 内にカスタムの録画ルールを作成することもできます。たとえば、Pod のコンテナメモリーキャッシュの合計を取得できるようにする記録ルールを作成することができます。YAML の内容は以下のようになります。

```

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

```

注記: これが最初の新規カスタムルールである場合には、すぐに作成されます。ConfigMap に変更が加えられると、設定は自動的に再読み込みされます。この設定は、**observability-thanos-ruler** サイドカー内の **config-reload** により再読み込みされます。

アラートルールが正しく機能していることを確認するには、Grafana ダッシュボードを起動し、**Explore** ページに移動し、**ALERTS** にクエリーを実行します。アラートは、アラートが開始された場合に Grafana でのみ利用できます。

2.2. カスタムメトリックの追加

metrics_list.yaml ファイルにメトリックを追加して、マネージドクラスターから収集されるようになります。

カスタムメトリックを追加する前に、**oc get mco observability -o yaml** コマンドで、**mco observability** が有効になっていることを確認します。**status.conditions.message** のメッセージが **Observability components are deployed and running** となっていることを確認します。

observability-metrics-custom-allowlist.yaml という名前のファイルを作成し、**metrics_list.yaml** パラメーターにカスタムメトリックの名前を追加します。ConfigMap の YAML は、以下の内容のようになります。

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: observability-metrics-custom-allowlist
data:
  metrics_list.yaml: |
    names:

```

```

- node_memory_MemTotal_bytes
rules:
- 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))

```

ユーザーワークロードメトリクスについては、[ユーザーワークロードメトリクスの追加](#) セクションを参照してください。

- **names** セクションで、マネージドクラスターから収集されるカスタムメトリックの名前を追加します。
- **rules** セクションで、パラメーターペア **expr** と **record** に値を1つだけ入力し、クエリー式を定義します。メトリックは、マネージドクラスターの **record** パラメーターで定義される名前で収集されます。クエリー式の実行後の結果が、メトリックの値として返されます。
- **names** と **rules** セクションはオプションです。セクションのいずれかまたは両方を使用できます。

oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml のコマンドで、**open-cluster-management-observability** namespace に **observability-metrics-custom-allowlist** ConfigMap を作成します。

Grafana ダッシュボードから **Explore** ページからメトリックをクエリーし、カスタムメトリックからのデータが収集されていることを確認します。独自のダッシュボードでカスタムメトリックを使用することもできます。ダッシュボードの表示について、詳しくは [Grafana ダッシュボードの使用](#) を参照してください。

2.2.1. ユーザーワークロードメトリクスの追加

OpenShift Container Platform のワークロードから OpenShift Container Platform ユーザー定義メトリクスを収集できます。監視を有効にする必要があります。[ユーザー定義プロジェクトのモニタリングの有効化](#) を参照してください。

ユーザー定義のワークロードの監視が有効になっているマネージドクラスターがある場合、ユーザーのワークロードは **test** namespace に配置され、メトリクスを生成します。これらのメトリクスは、OpenShift Container Platform ユーザーワークロードから Prometheus によって収集されます。

test namespace に **observability-metrics-custom-allowlist** という名前の ConfigMap を作成して、ユーザーワークロードからメトリクスを収集します。以下の例を参照してください。

```

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

```

- **uwl_metrics_list.yaml** は、ConfigMap データのキーです。
- ConfigMap データの値は YAML 形式です。**names** セクションには、**test** namespace から収集するメトリック名のリストが含まれます。ConfigMap を作成すると、ターゲット namespace

から指定されたメトリクスが可観測性コレクターによって収集され、ハブクラスターにプッシュされます。

2.2.2. デフォルトメトリックの削除

マネージドクラスターで特定のメトリック用にデータを収集しない場合は、**observability-metrics-custom-allowlist.yaml** ファイルからメトリックを削除します。メトリックを削除すると、メトリックデータはマネージドクラスターでは収集されません。前述したように、**mco observability** が有効になっていることを確認します。

メトリック名の先頭にハイフン - を指定して **metrics_list.yaml** パラメーターにデフォルトのメトリック名を追加します。例: **-cluster_infrastructure_provider**

oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml のコマンドで、**open-cluster-management-observability** namespace に **observability-metrics-custom-allowlist** ConfigMap を作成します。

特定のメトリックがマネージドクラスターから収集されていないことを確認します。Grafana ダッシュボードからメトリックをクエリーしても、メトリックは表示されません。

2.3. 外部エンドポイントへのメトリクスのエクスポート

可観測性をカスタマイズして、Prometheus Remote-Write 仕様をリアルタイムでサポートする外部エンドポイントにメトリクスをエクスポートできます。詳細は、[Prometheus Remote-Write 仕様](#) を参照してください。

2.3.1. 外部エンドポイントの Kubernetes シークレットの作成

open-cluster-management-observability namespace の外部エンドポイントのアクセス情報を使用して Kubernetes シークレットを作成する必要があります。次のシークレットの例を表示します。

```
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 はコンテンツのキーであり、次のステップで **MultiClusterObservability** カスタムリソースで使用されます。現在、可観測性では、セキュリティーチェックは使用せず、Basic 認証または **tls** を使用する場合に、エンドポイントへのメトリクスのエクスポートをサポートしています。サポートされているパラメーターの完全なリストについては、次の表を参照してください。

名前	説明	スキーマ
url required	外部エンドポイントの URL。	string
http_client_config optional	HTTP クライアントの高度な設定。	HttpClientConfig

HttpClientConfig

名前	説明	スキーマ
basic_auth (任意)	基本認証用の HTTP クライアント設定。	BasicAuth
tls_config (任意)	TLS の HTTP クライアント設定。	TLSConfig

BasicAuth

名前	説明	スキーマ
username (任意)	基本認証のユーザー名。	string
password (任意)	基本認証用のパスワード。	string

TLSConfig

名前	説明	スキーマ
secret_name (必須)	証明書を含むシークレットの名前。	string
ca_file_key (任意)	シークレットの CA 証明書のキー (<code>insecure_skip_verify</code> が true に設定されている場合のみオプション)。	string
cert_file_key (必須)	シークレット内のクライアント証明書のキー。	string
key_file_key (必須)	シークレットのクライアントキーのキー。	string

<code>insecure_skip_verify</code> (任意)	ターゲット証明書の検証をスキップするパラメーター。	bool
---	---------------------------	------

2.3.2. MultiClusterObservability カスタムリソースの更新

Kubernetes シークレットを作成したら、**MultiClusterObservability** カスタムリソースを更新して、**spec.storageConfig** パラメーターに **writeStorage** を追加する必要があります。以下の例を参照してください。

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

writeStorage の値はリストです。メトリクスを1つの外部エンドポイントにエクスポートする場合は、リストにアイテムを追加できます。リストに複数のアイテムを追加すると、メトリクスは複数の外部エンドポイントにエクスポートされます。各アイテムには、**name** と **key** の2つの属性が含まれています。**Name** は、エンドポイントアクセス情報を含む Kubernetes シークレットの名前であり、**key** はシークレット内のコンテンツのキーです。次の説明表を参照してください

2.3.3. メトリックエクスポートのステータスの表示

メトリクスのエクスポートを有効にした後、**acm_remote_write_requests_total** メトリクスをチェックすることにより、メトリクスのエクスポートのステータスを表示できます。ハブクラスターの OpenShift コンソールから、**Observe** セクションの **Metrics** をクリックして、**Metrics** ページに移動します。

次に、**acm_remote_write_requests_total** メトリックをクエリーします。そのメトリックの値は、1つの observatorium API インスタンスで、1つの外部エンドポイントに対する特定の応答を持つリクエストの総数です。**name** ラベルは、外部エンドポイントの名前です。**code** ラベルは、メトリクスエクスポートの HTTP リクエストのリターンコードです。

2.4. 詳細 設定の追加

advanced 設定セクションを追加して、必要に応じて可観測性コンポーネントごとに保持内容を更新します。

MultiClusterObservability カスタムリソースを編集し、コマンド **oc edit mco observability -o yaml** で **advanced** セクションを追加します。YAML ファイルは以下の内容のようになります。

```
spec:
  advanced:
    retentionConfig:
      blockDuration: 2h
      deleteDelay: 48h
      retentionInLocal: 24h
      retentionResolutionRaw: 30d
      retentionResolution5m: 180d
      retentionResolution1h: 0d
    receive:
      resources:
```



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

advanced 設定に追加できるすべてのパラメーターの説明は、[Observability API](#) を参照してください。

2.5. コンソールからの MULTICLUSTEROBSERVABILITY カスタムリソースレプリカの更新

ワークロードが増加する場合は、可観測性 Pod のレプリカ数を増やします。ハブクラスターから Red Hat OpenShift Container Platform コンソールに移動します。**MultiClusterObservability** カスタムリソースを見つけて、レプリカを変更するコンポーネントの **replicas** パラメーター値を更新します。更新した YAML は以下のようになります。

```
spec:
  advanced:
    receive:
      replicas: 6
```

mco observability カスタムリソース内のパラメーターの詳細は、[可観測性 API](#) を参照してください。

2.6. ルート証明書のカスタマイズ

OpenShift Container Platform ルート認証をカスタマイズする場合は、ルートを実名 **alt_names** セクションに追加する必要があります。OpenShift Container Platform ルートにアクセスできるようにするには、**alertmanager.apps.<domainname>**、**observatorium-api.apps.<domainname>**、**rbac-query-proxy.apps.<domainname>** の情報を追加します。

注記: ユーザーは証明書のローテーションおよび更新を行います。

2.7. オブジェクトストアにアクセスするための証明書のカスタマイズ

オブジェクトストアにアクセスするための証明書をカスタマイズするには、次の手順を実行します。

1. オブジェクトストアシークレットに証明書を追加して、**http_config** セクションを編集します。以下の例を参照してください。

```
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
      insecure_skip_verify: false
```

1. オブジェクトストアシークレットを **open-cluster-management-observability** namespace に追加します。シークレットには、前のシークレットの例で定義した **ca.crt** が含まれている必要があります。相互 TLS を有効にする場合は、前のシークレットで **public.crt** および

private.key を提供する必要があります。以下の例を参照してください。

```

thanos.yaml: |
  type: s3
  config:
    ...
  http_config:
  tls_config:
    ca_file: /etc/minio/certs/ca.crt ❶
    cert_file: /etc/minio/certs/public.crt
    key_file: /etc/minio/certs/private.key
    insecure_skip_verify: false

```

❶ 証明書と、**thanos-object-storage** シークレットのキー値へのパス。

2. **MultiClusterObservability** カスタムリソースの **TLSecretName** パラメーターでシークレット名を設定することもできます。シークレット名が **tls-certs-secret** である次の例を表示します。

```

metricObjectStorage:
  key: thanos.yaml
  name: thanos-object-storage
  tlsSecretName: tls-certs-secret

```

オブジェクトストアにアクセスする必要があるすべてのコンポーネントの **tlsSecretMountPath** リソースにシークレットをマウントします。これには、**receiver**, **store**, **ruler**, **compact** のコンポーネントが含まれます。

2.8. データの表示およびデプロイメント

ハブクラスターから Grafana にアクセスして、マネージドクラスターからデータを表示します。特定のアラートを照会して、そのクエリーのフィルターを追加できます。

たとえば、シングルノードクラスターから **cluster_infrastructure_provider** をクエリーするには、以下のクエリー式 **cluster_infrastructure_provider{clusterType="SNO"}** を使用します。

注記: シングルノードのマネージドクラスターで可観測性が有効になっている場合は、**ObservabilitySpec.resources.CPU.limits** パラメーターを設定しないでください。CPU 制限を設定すると、可観測性 Pod がマネージドクラスターの容量にカウントされます。**追加リソース** セクションの **管理ワークロードのパーティショニング** を参照してください。

2.8.1. 履歴データの表示

履歴データをクエリーする場合は、クエリーパラメーターオプションを手動で設定して、ダッシュボードから表示されるデータの量を制御します。以下の手順を実行します。

1. ハブクラスターから、コンソールヘッダーにある **Grafana link** を選択します。
2. **Edit Panel** を選択して、クラスターダッシュボードを編集します。
3. Grafana のクエリーフロントエンドデータソースから、**Query** タブをクリックします。
4. **\$datasource** を選択します。

5. より多くのデータを表示する場合は、**Step** パラメーターセクションの値を増やします。**Step** パラメーターセクションが空の場合は、自動的に計算されます。
6. **Custom query parameters** フィールドを見つけて、**max_source_resolution=auto** を選択します。
7. データが表示されていることを確認するには、Grafana ページを更新します。

Grafana ダッシュボードからクエリデータが表示されます。

2.8.2. etcd テーブルの表示

Grafana のハブクラスターダッシュボードから etcd テーブルを表示し、データストアとしての etcd の安定性を確認します。

ハブクラスターから Grafana リンクを選択して、ハブクラスターから収集された **etcd** テーブルデータを表示します。マネージドクラスターの **Leader election changes** が表示されます。

2.8.3. Kubernetes API サーバーダッシュボードのクラスターフリートサービスレベルの概要の表示

Grafana のハブクラスターダッシュボードから、Kubernetes API サービスレベルの概要を表示します。

Grafana ダッシュボードに移動した後に、**Kubernetes > Service-Level Overview > API Server** を選択して管理ダッシュボードメニューにアクセスします。**Fleet Overview** および **Top Cluster** の詳細が表示されます。

過去7日間または30日間のターゲットとする **サービスレベル目標 (SLO)** 値を超えるか、満たしているクラスターの合計数、オフラインクラスター、および API サーバー要求の期間を表示します。

2.8.4. Kubernetes API サーバーダッシュボードのクラスターサービスレベルの概要の表示

Grafana のハブクラスターダッシュボードから Kubernetes API サービスレベルの概要テーブルを表示します。

Grafana ダッシュボードに移動した後に、**Kubernetes > Service-Level Overview > API Server** を選択して管理ダッシュボードメニューにアクセスします。**Fleet Overview** および **Top Cluster** の詳細が表示されます。

過去7日間または30日間のエラーとなっている予算、残りのダウンタイム、および傾向を表示します。

2.9. 可観測性の無効化

可観測性を無効にして、Red Hat Advanced Cluster Management ハブクラスターでデータ収集を停止します。

2.9.1. すべてのクラスターで可観測性を無効にする

すべてのマネージドクラスターで可観測性コンポーネントを削除して、可観測性を無効にします。

enableMetrics を **false** に設定して、**multicluster-observability-operator** リソースを更新します。更新されたリソースは、以下のような変更内容になります。

■

```
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
```

2.9.2. 単一クラスターで可観測性を無効にする

特定のマネージドクラスターの可観測性コンポーネントを削除して可観測性を無効にします。 **managedclusters.cluster.open-cluster-management.io** のカスタムリソースに **observability: disabled** ラベルを追加します。

Red Hat Advanced Cluster Management コンソールの **Clusters** ページから、指定したクラスターに **observability=disabled** ラベルを追加します。

注記: 可観測性コンポーネントが含まれるマネージドクラスターをデタッチすると、**metrics-collector** デプロイメントが削除されます。

2.10. 関連情報

- 可観測性アラートの詳細は、[可観測性アラート](#) を参照してください。
- アラート転送の詳細は、[Prometheus Alertmanager ドキュメント](#) を参照してください。
- 詳細は、[可観測性アラート](#) を参照してください。
- 可観測性サービスの詳細は、[可観測性サービスの概要](#) を参照してください。
- 詳細は、[Prometheus の設定](#) を参照してください。
- 詳細は、[管理ワークロードのパーティショニング](#) を参照してください。
- このトピックで最初に説明した [可観測性のカスタマイズ](#) を参照してください。

第3章 可観測性アラート

ハブクラスターとマネージドクラスターの変更が通知されるように、可観測性サービスのアラートを受信および定義します。

- [Alertmanager の設定](#)
- [アラートの転送](#)
 - [マネージドクラスターのアラート転送の無効化](#)
- [アラートをサイレントにする](#)
- [アラートの抑制](#)

3.1. ALERTMANAGER の設定

メール、Slack、PagerDuty などの外部メッセージングツールを統合し、Alertmanager から通知を受信します。**open-cluster-management-observability** namespace で **alertmanager-config** シークレットを上書きして、統合を追加し、Alertmanager のルートを設定します。以下の手順を実行して、カスタムのレシーバールールを更新します。

1. **alertmanager-config** シークレットからデータを抽出します。以下のコマンドを実行します。

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

2. 以下のコマンドを実行し、**alertmanager.yaml** ファイル設定を編集して保存します。

```
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=
```

更新したシークレットは以下の内容のようになります。

```
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
```

変更内容は、変更後すぐに適用されます。Alertmanager の例は、[prometheus/alertmanager](#) を参照してください。

3.2. アラートの転送

可観測性を有効にした後には、OpenShift Container Platform マネージドクラスターからのアラートは自動的にハブクラスターに送信されます。**alertmanager-config** YAML ファイルを使用して、外部通知システムでアラートを設定できます。

alertmanager-config YAML ファイルの例を以下に示します。

```
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 }}'
```

アラート転送用のプロキシを設定する場合は、**alertmanager-config** YAML ファイルに次の **global** エントリーを追加します。

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

3.2.1. マネージドクラスターのアラート転送の無効化

マネージドクラスターのアラート転送を無効にするには、次のアノテーションを **MultiClusterObservability** カスタムリソースに追加します。

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

アノテーションを設定すると、マネージドクラスターのアラート転送設定が元に戻ります。**openshift-monitoring** namespace の **ocp-monitoring-config** ConfigMap に加えられた変更も元に戻ります。アノテーションを設定すると、**ocp-monitoring-config** ConfigMap が可観測性オペレーターのエンドポイントによって管理または更新されなくなります。設定を更新すると、マネージドクラスターの Prometheus インスタンスが再起動します。

重要: メトリクス用の永続ボリュームを持つ Prometheus インスタンスがある場合、マネージドクラスターからのメトリクスは失われ、Prometheus インスタンスが再起動されます。ハブクラスターからのメトリクスは影響を受けません。

変更が元に戻ると、**cluster-monitoring-reverted** という名前の ConfigMap が **open-cluster-management-addon-observability** namespace に作成されます。手動で追加された新しいアラート転送設定は、ConfigMap から元に戻りません。

ハブクラスターアラートマネージャーがマネージドクラスターアラートをサードパーティーのメッセージングツールに伝達していないことを確認します。前のセクション **Alertmanager の設定** を参照してください。

3.3. アラートをサイレントにする

受信したくないアラートを追加します。アラート名、一致ラベル、または期間によってアラートをサイレントにすることができます。サイレントにしたいアラートを追加すると、ID が作成されます。サイレントにしたアラートの ID は、文字列 **d839aca9-ed46-40be-84c4-dca8773671da** のようになります。

アラートをサイレントにする方法は、引き続きお読みください。

- Red Hat Advanced Cluster Management アラートをサイレントにするには、**open-cluster-management-observability** namespace の **alertmanager-main** Pod にアクセスする必要があります。たとえば、Pod ターミナルに次のコマンドを入力して、**SampleAlert** をサイレントにします。

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

- 複数の一致ラベルを使用してアラートをサイレントにします。次のコマンドは **match-label-1** と **match-label-2** を使用します。

```
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>
```

- 特定の期間アラートをサイレントにする場合は、**--duration** フラグを使用します。次のコマンドを実行して、**SampleAlert** を1時間サイレントにします。

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

消音アラートの開始時刻または終了時刻を指定することもできます。次のコマンドを入力して、特定の開始時刻に **SampleAlert** をサイレントにします。

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

- 作成されたサイレント化されたアラートをすべて表示するには、次のコマンドを実行します。

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

- アラートをサイレントにしたい場合は、次のコマンドを実行してアラートのサイレントを終了します。

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

- すべてのアラートをサイレントにするのを終了するには、次のコマンドを実行します。

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

3.4. アラートの抑制

重大度の低い Red Hat Advanced Cluster Management アラートをクラスター全体でグローバルに抑制します。アラートを抑制するには、**open-cluster-management-observability** namespace の **alertmanager-config** で抑制ルールを定義します。

抑制ルールは、既存のマッチャーの別のセットと一致する一連のパラメーター一致がある場合にアラートをミュートします。ルールを有効にするには、ターゲットアラートとソースアラートの両方で、**equal** リスト内のラベル名のラベル値が同じである必要があります。**Inhibit_rules** は次のようになります。

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

- ❶ **hibit_rules** パラメーターセクションは、同じ namespace のアラートを検索するために定義されています。**critical** アラートがネームスペース内で開始し、その namespace に重大度レベルの **warning** または **info** を含む他のアラートがある場合は、**critical** アラートのみが Alertmanager レシーバーにルーティングされます。一致するものがあつた場合、次のアラートが表示される場合があります。

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

- ❷ **source_match** パラメーターと **target_match_re** パラメーターの値が一致しない場合、アラートはレシーバーにルーティングされます。

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

- Red Hat Advanced Cluster Management で抑制されたアラートを表示するには、次のコマンドを入力します。

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

3.5. 関連情報

- このトピックの最初で説明した [可観測性アラート](#) を参照してください。
- 詳細は、[可観測性のカスタマイズ](#) を参照してください。
- 可観測性に関するその他のトピックは、[可観測性サービスの概要](#) を参照してください。

第4章 GRAFANA ダッシュボードの使用

Grafana ダッシュボードを使用して、ハブクラスターとマネージドクラスターのメトリクスを表示します。

- [Red Hat Advanced Cluster Management for Kubernetes ダッシュボード](#)
- [Grafana 開発者インスタンスの設定](#)
 - [Grafana のバージョン検証](#)
- [Grafana ダッシュボードの設計](#)
 - [ConfigMap での Grafana ダッシュボードの設計](#)
- [Grafana 開発者インスタンスのアンインストール](#)
- [関連情報](#)

4.1. RED HAT ADVANCED CLUSTER MANAGEMENT FOR KUBERNETES ダッシュボード

可観測性サービスを有効にすると、3つのダッシュボードが使用可能になります。次のダッシュボードの説明を確認してください。

- **Alert Analysis:** マネージドクラスターフリート内で生成されているアラートの概要を示すダッシュボード。
- **Clusters by Alert:** アラート名でフィルタリングできるアラートダッシュボード。
- **Alerts by Cluster:** クラスターでフィルタリングし、クラスター環境内で発生したアラート、または保留中のアラートのリアルタイムデータを表示できるアラートダッシュボード。

4.2. GRAFANA 開発者インスタンスの設定

grafana-dev インスタンスを作成して、Grafana ダッシュボードを設計できます。まず、[stolostron/multicluster-observability-operator/](#) リポジトリのクローンを作成し、**tools** フォルダにあるスクリプトを実行できるようにします。必ず最新の **grafana-dev** インスタンスを使用してください。

Grafana 開発者インスタンスを設定するには、以下の手順を実行します。

1. **setup-grafana-dev.sh** を実行して、Grafana インスタンスを設定します。スクリプトを実行すると、**secret/grafana-dev-config**、**deployment.apps/grafana-dev**、**service/grafana-dev**、**ingress.extensions/grafana-dev**、**persistentvolumeclaim/grafana-dev** のリソースが作成されます。

```
./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

```

2. **switch-to-grafana-admin.sh** スクリプトを使用して、ユーザーロールを Grafana 管理者に切り替えます。
 - a. Grafana の URL https://grafana-dev-open-cluster-management-observability.{OPENSHIFT_INGRESS_DOMAIN} を選択し、ログインします。
 - b. 次に、以下のコマンドを実行して、切り替えユーザーを Grafana 管理者として追加します。たとえば、**kubeadmin** を使用してログインしたら、以下のコマンドを実行します。

```

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

```

Grafana 開発者インスタンを設定します。

4.2.1. Grafana のバージョン検証

コマンドラインインターフェイス (CLI) または Grafana ユーザーインターフェイスから Grafana のバージョンを検証します。

ハブクラスターにログインした後、**observability-grafana** Pod ターミナルにアクセスします。以下のコマンドを実行します。

```
grafana-cli
```

現在クラスター環境内にデプロイされている Grafana のバージョンが表示されます。

Grafana ダッシュボードの **Manage** タブに移動することもできます。ページの最後までスクロールすると、バージョンリストがあります。

4.3. GRAFANA ダッシュボードの設計

Grafana インスタンを設定したら、ダッシュボードを設計できます。Grafana コンソールを更新し、ダッシュボードを設計するには、以下の手順を実行します。

1. Grafana コンソールのナビゲーションパネルから **Create** アイコンを選択してダッシュボードを作成します。 **Dashboard** を選択し、 **Add new panel** をクリックします。
2. **New Dashboard/Edit Panel** ビューで、 **Query** タブを選択します。
3. データソースセクターから **Observatorium** を選択し、PromQL クエリーを入力してクエリーを設定します。
4. Grafana ダッシュボードヘッダーから、ダッシュボードヘッダーにある **Save** アイコンをクリックします。
5. 説明的な名前を追加し、 **Save** をクリックします。

4.3.1. ConfigMap での Grafana ダッシュボードの設計

ConfigMap を使用して、Grafana ダッシュボードを設計します。**generate-dashboard-configmap-yaml.sh** スクリプトを使用してダッシュボードの ConfigMap を生成し、ローカルで ConfigMap を保存できます。

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

前述のスクリプトを実行するパーミッションがない場合は、以下の手順を実行します。

1. ダッシュボードを選択し、**Dashboard 設定** アイコンをクリックします。
2. ナビゲーションパネルから **JSON Model** アイコンをクリックします。
3. ダッシュボード JSON データをコピーし、**data** セクションに貼り付けます。
4. **name** を、**\$your-dashboard-name** に置き換えます。**data.\$your-dashboard-name.json.\$\$your_dashboard_json** の **uid** フィールドに Universally Unique Identifier (UUID) を入力します。**uuirogen** などのプログラムを使用して UUID を作成できます。ConfigMap は、以下のファイルのようになります。

```
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
```

注記:

- ダッシュボードが **grafana-dev** インスタンス内に作成されている場合は、ダッシュボードの名前を取得して、スクリプトで引数として渡すことができます。たとえば、**Demo Dashboard** という名前のダッシュボードが **grafana-dev** インスタンスに作成されます。CLI から、次のスクリプトを実行できます。

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

スクリプトを実行すると、次のメッセージが表示される場合があります。

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

- ダッシュボードが **General** フォルダーにない場合は、この ConfigMap の **annotations** セクションでフォルダー名を指定できます。

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

ConfigMap の更新が完了したら、インストールしてダッシュボードを Grafana インスタンスにインポートできます。

CLI または OpenShift Container Platform コンソールから YAML を適用して、YAML ファイルが作成されていることを確認します。**open-cluster-management-observability** namespace 内に ConfigMap が作成されます。CLI から次のコマンドを実行します。

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

OpenShift Container Platform コンソールから、**demo-dashboard.yaml** ファイルを使用して、ConfigMap を作成します。ダッシュボードは **Custom** フォルダーにあります。

4.4. GRAFANA 開発者インスタンスのアンインストール

インスタンスをアンインストールすると、関連するリソースも削除されます。以下のコマンドを実行します。

```
./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
```

4.5. 関連情報

- UUID の作成手順は、[uuidegen](#) を参照してください。
- 詳細は、[Grafana でのマネージドクラスターラベルの使用](#) を参照してください。
- [Grafana ダッシュボードの使用](#) ページの先頭に戻ります。
- [環境の監視の概要](#) に戻ります。

4.6. GRAFANA でマネージドクラスターラベルを使用する

マネージドクラスターラベルを有効にして、Grafana ダッシュボードで使用できるようにします。ハブクラスターで可観測性が有効になっている場合は、**observability-managed-cluster-label-allowlist** ConfigMap が **open-cluster-management-observability** namespace に作成されます。ConfigMap には、**observability-rbac-query-proxy** Pod によって維持されるマネージドクラスターラベルのリストが含まれており、**ACM - Cluster Overview** Grafana ダッシュボード内からフィルタリングするラベル名のリストを入力します。デフォルトでは、可観測性は **observability-managed-cluster-label-allowlist** ConfigMap のラベルのサブセットを無視します。

クラスターがマネージドクラスターフリートにインポートされるか、変更されると、**observability-rbac-query-proxy** Pod は、マネージドクラスターラベルを参照して変更を監視し、**observability-managed-cluster-label-allowlist** ConfigMap を自動的に更新して、変化します。ConfigMap には、**ignore_labels** または **labels** リストに含まれる一意のラベル名のみが含まれます。**observability-managed-cluster-label-allowlist** ConfigMap は次の YAML ファイルのようになる場合があります。

```
data:
  managed_cluster.yaml: |
    ignore_labels: ❶
    - 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> ConfigMap の **ignore_labels** キーリストにリストされているラベルはすべて、**ACM - Clusters Summary** Grafana ダッシュボードのドロップダウンフィルターから削除されます。<2> 有効になっているラベルは **ACM - Clusters Overview** Grafana ダッシュボードのドロップダウンフィルターに表示されます。値は、選択した **label** キーの値に応じて、**acm_managed_cluster_labels** メトリックから取得されます。

引き続き Grafana でのマネージドクラスターラベルの使用方法を確認してください。

- [マネージドクラスターラベルの追加](#)
- [マネージドクラスターラベルの有効化](#)
- [マネージドクラスターラベルの無効化](#)

4.6.1. マネージドクラスターラベルの追加

マネージドクラスターラベルを **observability-managed-cluster-label-allowlist** ConfigMap に追加すると、そのラベルは Grafana のフィルターオプションとして使用できるようになります。ハブクラスター、またはマネージドクラスターフリートに関連付けられているマネージドクラスターオブジェクトに一意的なラベルを追加します。たとえば、ラベル **department=finance** をマネージドクラスターに追加すると、ConfigMap が更新され、次のように変更されます。

```

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

```

4.6.2. マネージドクラスターラベルの有効化

observability-managed-cluster-label-allowlist ConfigMap の **ignore_labels** リストからラベルを削除して、すでに無効になっているマネージドクラスターラベルを有効にします。

たとえば、**local-cluster** および **name** ラベルを有効にします。**observability-managed-cluster-label-allowlist** ConfigMap は、次の内容のようになる場合があります。

```

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

```

クラスターラベルが確実に更新されるように、ConfigMap は 30 秒後に再同期します。ConfigMap を更新した後、**open-cluster-management-observability** namespace の **observability-rbac-query-proxy** Pod ログをチェックして、ラベルがリストされている場所を確認します。次の情報が Pod ログに表示される場合があります。

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

Grafana ダッシュボードから、ラベルが **Label** ドロップダウンメニューの値としてリストされていることを確認します。

4.6.3. マネージドクラスターラベルの無効化

Label ドロップダウンフィルターのリストからマネージドクラスターラベルを除外します。ラベル名を **ignore_labels** リストに追加します。たとえば、**local-cluster** と **name** を **ignore_labels** リストに戻すと、YAML は次のファイルのようになります。

```

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

```

open-cluster-management-observability namespace の **observability-rbac-query-proxy** Pod ログをチェックして、ラベルがどこにリストされているかを確認します。次の情報が Pod ログに表示される場合があります。

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

4.6.4. 関連情報

- [Grafana ダッシュボードの使用](#) を参照してください。
- ページの最初の [Grafana でのマネージドクラスターラベルの使用](#) に戻ります。

第5章 コンソールでの検索の概要

Red Hat Advanced Cluster Management for Kubernetes では、検索機能でクラスター全体の Kubernetes リソースを視認できるようにします。検索では、Kubernetes リソースや他のリソースとの関係もインデックス化されます。

- [検索コンポーネント](#)
- [検索のカスタマイズと設定](#)
- [関連情報](#)

5.1. 検索コンポーネント

検索アーキテクチャーは、以下のコンポーネントで設定されています。

表5.1 検索コンポーネントテーブル

コンポーネント名	メトリクス	メトリックのタイプ	説明
search-collector			Kubernetes リソースを監視し、リソースメタデータを収集し、すべてのマネージドクラスターにわたるリソースの関係を計算し、収集したデータを search-indexer に送信します。マネージドクラスターの search-collector は、 klusterlet-addon-search という名前の Pod として実行されます。
search-indexer コレクターからリソースのメタデータを受信し、PostgreSQL データベースに書き込みます。 search-indexer はハブクラスターのリソースを監視し、アクティブなマネージドクラスターを追跡します。	search_indexer_request_duration	ヒストグラム	検索インデクサーが (マネージドクラスターからの) 要求を処理するのにかかる時間 (秒)。
	search_indexer_request_size	ヒストグラム	(マネージドクラスターからの) 検索インデクサー要求における変更の合計 (追加、更新、削除)。
	search_indexer_request_count	カウンター	検索インデクサーが (マネージドクラスターから) 受信したリクエストの合計。

コンポーネント名	メトリクス	メトリックのタイプ	説明
	search_indexer_requests_in_flight	ゲージ	検索インデクサーが指定された時間で処理する要求の合計数。
search-api GraphQL を介して search-indexer 内のすべてのクラスターデータへのアクセスを提供し、ロールベースのアクセス制御 (RBAC) を適用します。	search_api_requests	ヒストグラム	HTTP 要求の継続時間のヒストグラム (秒単位)。
	search_dbquery_duration_seconds	ヒストグラム	データベース要求のレイテンシー (秒単位)。
	search_api_db_connection_failed_total	カウンター	失敗したデータベース接続試行の合計数。
search-postgres			すべてのマネージドクラスターから収集されたデータを PostgreSQL データベースのインスタンスに保存します。

デフォルトでは、検索はハブクラスターで設定されます。マネージドクラスターをプロビジョニングするか、手動でインポートすると、**klusterlet-addon-search** が有効になります。マネージドクラスターの検索を無効にする場合は、[クラスターの klusterlet アドオン設定の変更](#) を参照してください。

5.2. 検索のカスタマイズと設定

search-v2-operator カスタムリソースのデフォルト値を変更できます。カスタムリソースの詳細を表示するには、次のコマンドを実行します。

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

検索オペレーターは、**search-v2-operator** カスタムリソースを監視し、変更を調整して、アクティブな Pod を更新します。次の設定の説明を参照してください。

- PostgreSQL データベースストレージ:
Red Hat Advanced Cluster Management をインストールすると、PostgreSQL データベースは、PostgreSQL データを空のディレクトリー (**emptyDir**) ボリュームに保存するように設定されます。空のディレクトリーサイズが制限されている場合は、PostgreSQL データを永続ボリューム要求 (PVC) に保存して、検索パフォーマンスを向上させることができます。Red Hat Advanced Cluster Management ハブクラスターからストレージクラスを選択して、検索データをバックアップできます。たとえば、**gp2** ストレージクラスを選択した場合、設定は次の例のようになります。

```
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
```

この設定により、**gp2-search** という名前の PVC が作成され、**search-postgres** Pod にマウントされます。デフォルトでは、ストレージサイズは **10Gi** です。ストレージサイズを変更できます。たとえば、約 200 のマネージドクラスターには **20Gi** で十分な場合があります。

- 4つの検索 Pod (**indexer**、**database**、**queryapi**、**collector**) のPod メモリーまたは CPU 要件、レプリカ数、更新ログレベルを調整することでコストを最適化します。**search-v2-operator** カスタムリソースの **deployment** セクションを更新します。**search-v2-operator** によって管理される 4つのデプロイメントがあり、個別に更新できます。**search-v2-operator** カスタムリソースは、次のファイルのようになる場合があります。

```
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 リソースを、**indexer**、**database**、**queryapi**、**collector** のPod に適用できます。

② **envVar** セクションに複数の環境変数を追加して、名前を付けた各変数の値を指定できます。

③ **--v=3** 引数を追加することで、前述した 4つの Pod のいずれかでログの詳細レベルを制御できます。

以下は、メモリーリソースがインデクサー Pod に適用される例です。

```
indexer:
```

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

- 検索 Pod のノード配置:
nodeSelector パラメーターまたは **tolerations** パラメーターを使用して、検索 Pod の **Placement** を更新できます。次の設定例を表示します。

```
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
```

5.3. 関連情報

- 検索の管理方法は、[検索の管理](#) を参照してください。
- Red Hat Advanced Cluster Management for Kubernetes コンソールに関するその他のトピックについては、[Web コンソール](#) を参照してください。

5.4. 検索の管理

検索を使用して、クラスターからリソースデータをクエリーします。

必要なアクセス権限: クラスターの管理者

次のトピックを引き続きお読みください。

- [検索設定可能コレクションの作成](#)
- [サーチコンソールのカスタマイズ](#)
- [コンソールでのクエリー](#)
 - [ArgoCD アプリケーションのクエリー](#)
- [マネージドクラスターでの klusterlet-addon-search デプロイメントの更新](#)

5.4.1. 検索設定可能コレクションの作成

search-collector-config ConfigMap を作成して、許可リストと拒否リストのセクションにリソースをリストすることにより、どの Kubernetes リソースをクラスターから収集するかを定義します。

ConfigMap 内の **data.AllowedResources** および **data.DeniedResources** セクションにリソースを一覧表示します。以下のコマンドを実行してリソースを作成します。

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

ConfigMap は、次の YAML ファイルのようになります。

```
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
```

上記の ConfigMap の例では、すべての **apiGroups** から **services** と **pods** を収集でき、**admission.k8s.io** および **authentication.k8s.io** **apiGroups** からすべてのリソースも収集できます。同時に、この ConfigMap の例では、すべての **apiGroup** から **secrets** を一元的に収集するのを防止し、**apiGroup admission.k8s.io** からの **policy**、**iampolicies**、**certificatepolicies** の収集も防止します。

注記: ConfigMap を指定しない場合、デフォルトですべてのリソースが収集されます。**AllowedResources** のみを指定した場合、**AllowedResources** にリストされていないすべてのリソースは自動的に除外されます。**AllowedResources** と **DeniedResources** に同時にリストされているリソースも除外されます。

5.4.2. サーチコンソールのカスタマイズ

OpenShift Container Platform コンソールから検索結果の制限をカスタマイズできます。**multicluster-engine** namespace の **console-mce-config** を更新します。これらの設定はすべてのユーザーに適用され、パフォーマンスに影響を与える可能性があります。次のパフォーマンスパラメーターの説明を表示します。

- **SAVED_SEARCH_LIMIT**: ユーザーごとに保存された検索の最大量。デフォルトでは、ユーザーごとに10個の保存済み検索の制限があります。デフォルト値は **10** です。制限を更新するには、**console-config** ConfigMap にキー値 **SAVED_SEARCH_LIMIT: x** を追加します。
- **SEARCH_RESULT_LIMIT**: コンソールに表示される検索結果の最大量。デフォルト値は **1000** です。この制限を削除するには、**-1** に設定します。
- **SEARCH_AUTOCOMPLETE_LIMIT**: 検索バーの先行入力に対して取得される候補の最大数。デフォルト値は **10,000** です。この制限を削除するには、**-1** に設定します。

OpenShift Container Platform コンソールから次の **patch** コマンドを実行して、検索結果を100項目に変更します。

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

5.4.3. コンソールでのクエリー

検索ボックスにテキスト値を入力すると、名前や namespace などのプロパティからのその値が含まれる結果が表示されます。空白のスペースを含む値の検索はできません。

検索結果をさらに絞り込むには、検索にプロパティセレクターを追加します。プロパティに関連する値を組み合わせ、検索範囲をより正確に指定できます。たとえば、**cluster:dev red** と検索すると、**dev** クラスター内で "red" の文字列と一致する結果が返されます。

検索でクエリーを作成するには、次の手順を実行します。

1. ナビゲーションメニューの **検索** をクリックします。
2. **Search box** に単語を入力すると、検索機能で、対象の値が含まれたリソースを見つけ出します。
 - リソースを検索すると、元の検索結果に関連する他のリソースが表示されるので、リソースがシステム内にある他のリソースとどのように対話するのかを視覚的に確認できます。
 - 検索すると、各クラスターと、検索したリソースが返され、リスト表示されます。ハブクラスターのリソースの場合には、クラスター名は **local-cluster** として表示されます。
 - 検索結果は、**kind** でグループ化され、リソースの **kind** ごとに表でグループ化されます。
 - 検索オプションはクラスターオブジェクトにより異なります。
 - 特定のラベルで結果を絞り込むことができます。ラベルのクエリー時の検索は、大文字と小文字が区別されます。フィルタリング用に選択できる例 **name**、**namespace**、**status**、およびその他のリソースフィールドを参照してください。自動補完では、補完候補を表示して検索を絞り込むことができます。以下の例を参照してください。
 - **kind:pod** など、フィールド1つを検索すると、すべての Pod リソースが返されます。
 - **kind:pod namespace:default** など、複数のフィールドを検索すると、デフォルトの namespace にある Pod が返されます。

注記:

- **>**, **>=**, **<**, **<=**, **!=** などの文字を使用して、条件を指定した検索も可能です。

- 複数の値を含む複数のプロパティセレクターを検索すると、クエリーされた値のいずれかを返します。以下の例を参照してください。
- **kind:pod name:a** と検索すると、**a** という名前の Pod が返されます。
- **kind:pod name:a,b** と検索すると、**a** または **b** という名前の Pod が返されます。
- **kind:pod status:!Running** を検索すると、ステータスが **Running** ではないすべての Pod リソースが返されます。
- **kind:pod restarts:>1** を検索すると、最低でも 2 回再起動した全 Pod が返されます。

3. 検索を保存する場合は、**Save search** アイコンをクリックします。

5.4.3.1. ArgoCD アプリケーションのクエリー

ArgoCD アプリケーションを検索すると、**Applications** ページに移動します。**Search** ページから ArgoCD アプリケーションにアクセスするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management ハブクラスターにログインします。
2. コンソールヘッダーから **Search** アイコンを選択します。
3. **kind:application** および **apigroup:argoproj.io** の値でクエリーをフィルターします。
4. 表示するアプリケーションを選択します。**アプリケーション** ページでは、アプリケーションに関する情報の概要が表示されます。

5.4.4. マネージドクラスターでの **klusterlet-addon-search** デプロイメントの更新

マネージドクラスターから Kubernetes オブジェクトを収集するために、検索が有効になっているすべてのマネージドクラスターで **klusterlet-addon-search** Pod が実行されます。このデプロイメントは、**open-cluster-management-agent-addon** namespace で実行されます。多数のリソースを持つマネージドクラスターでは、**klusterlet-addon-search** デプロイメントが機能するために、より多くのメモリーが必要になる場合があります。

マネージドクラスター内の **klusterlet-addon-search** Pod のリソース要件は、Red Hat Advanced Cluster Management ハブクラスター内の **ManagedClusterAddon** カスタムリソースで指定できます。マネージドクラスターごとに、マネージドクラスター名を持つ namespace があります。マネージドクラスター名と一致する namespace から **ManagedClusterAddon** カスタムリソースを編集します。次のコマンドを実行して、**xyz** マネージドクラスターのリソース要件を更新します。

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

リソース要件をアノテーションとして追加します。以下の例を参照してください。

```
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
```

アノテーションはマネージドクラスターのリソース要件をオーバーライドし、新しいリソース要件で Pod を自動的に再起動します。

[環境の監視の概要](#) に戻ります。

第6章 RED HAT INSIGHTS での可観測性の使用

Red Hat Insights は、Red Hat Advanced Cluster Management 可観測性と統合されており、クラスター内の既存の問題や発生しうる問題を特定できるように有効化されています。Red Hat Insights は、安定性、パフォーマンス、ネットワーク、およびセキュリティリスクの特定、優先順位付け、および解決に役立ちます。Red Hat OpenShift Container Platform は、OpenShift Cluster Manager を使用してクラスターのヘルスマonitoringを提供します。OpenShift Cluster Manager は、クラスターのヘルス、使用状況、サイズの情報を匿名で累積して収集します。詳細は、[Red Hat Insights の製品ドキュメント](#) を参照してください。

OpenShift クラスターを作成またはインポートすると、マネージドクラスターからの匿名データは自動的に Red Hat に送信されます。この情報を使用してクラスターのヘルス情報を提供する insights を作成します。Red Hat Advanced Cluster Management 管理者は、このヘルス情報を使用して重大度に基づいてアラートを作成できます。

必要なアクセス権限: クラスターの管理者

6.1. 前提条件

- Red Hat Insights が有効になっていることを確認する。詳細は、[グローバルクラスタープルシークレットの変更によるリモートヘルスレポートの無効化](#) を参照してください。
- OpenShift Container Platform バージョン 4.0 以降がインストールされている。
- OpenShift Cluster Manager に登録されているハブクラスターユーザーが OpenShift Cluster Manager の全 Red Hat Advanced Cluster Management マネージドクラスターを管理できる。

6.2. RED HAT ADVANCED CLUSTER MANAGEMENT コンソールからの RED HAT INSIGHTS

以下で、統合に関する機能の説明を確認します。

- **Clusters** ページからクラスターを選択すると、**Status** カードから **特定された問題の数** を選択できます。**Status** カードには、**ノード**、**アプリケーション**、**ポリシー違反** および **特定された問題** に関する情報が表示されます。**Identified issues** カードは、Red Hat Insights からの情報を表します。**Identified issues** のステータスには、重大度による問題数が表示されます。問題の対応レベルは、**Critical**、**Major**、**Low**、および **Warning** の重大度に分類されます。
- 数字をクリックすると、**Potential issue** のサイドパネルが表示されます。パネルにすべての問題の概要およびチャートが表示されます。検索機能を使用して、推奨される修復を検索することもできます。修復オプションは、脆弱性の **説明**、脆弱性に関連する **カテゴリー**、および **全体的なリスク** を表示します。
- **Description** セクションから、脆弱性へのリンクを選択できます。**How to remediate** タブを選択して脆弱性を解決するための手順を表示します。**Reason** タブをクリックすると、脆弱性が発生した理由を確認することもできます。

詳細は、[Insight PolicyReports の管理](#) を参照してください。

6.3. INSIGHTS POLICYREPORTS の管理

Red Hat Advanced Cluster Management for Kubernetes **PolicyReports** は、**insights-client** で生成される違反です。**PolicyReports** は、インシデント管理システムに送信されるアラートの定義および設定に使用されます。違反がある場合には、**PolicyReport** からのアラートはインシデント管理システムに

送信されます。

Insight **PolicyReports** の管理および表示方法については、以下のセクションを参照してください。

- [Insight ポリシーレポートの検索](#)
- [コンソールから特定された問題の表示](#)

6.3.1. Insight ポリシーレポートの検索

マネージドクラスター全体で、違反した特定の insight **PolicyReport** を検索できます。

Red Hat Advanced Cluster Management ハブクラスターにログインしたら、コンソールヘッダーの **Search** アイコンをクリックして **Search** ページに移動します。 **kind:policyreport** のクエリーを入力します。

注記: **PolicyReport** 名はクラスターの名前と同じになります。

また、クエリーは、insight ポリシー違反およびカテゴリ別にさらに指定することもできます。 **PolicyReport** 名を選択すると、関連付けられたクラスターの **Details** ページにリダイレクトされます。 **Insights** サイドバーが自動的に表示されます。

検索サービスが無効になり、insight を検索する必要がある場合は、ハブクラスターから以下のコマンドを実行します。

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

6.3.2. コンソールから特定された問題の表示

特定のクラスターで特定された問題を表示できます。

Red Hat Advanced Cluster Management クラスターにログインしたら、ナビゲーションメニューから **Overview** を選択します。重大度を選択して、対象の重大度に関連付けられた **PolicyReports** を表示します。 **クラスターの問題** の概要カードから、クラスターの問題と重要性の詳細を表示します。

または、ナビゲーションメニューから **Clusters** を選択できます。テーブルからマネージドクラスターを選択して、詳細情報を表示します。 **Status** カードから、特定された問題の数を表示します。

発生する可能性のある問題数を選択して、重大度チャートと、その問題に対して推奨される修復を表示します。脆弱性へのリンクをクリックすると、**修復する方法** と脆弱性の **理由** の手順を表示します。

注記: 問題の解決後には、Red Hat Advanced Cluster Management で Red Hat Insights の情報を 30 分ごとに受信し、Red Hat Insights は 2 時間ごとに更新されます。

PolicyReport からアラートメッセージを送信したコンポーネントを確認してください。 **ガバナンス** ページに移動し、特定の **ポリシーレポート** を選択します。 **Status** タブを選択し、 **View details** リンクをクリックして **PolicyReport** YAML ファイルを表示します。

source パラメーターを見つけます。このパラメーターにより、違反を送信したコンポーネントが通知されます。値オプションは **grc** および **insights** です。

PolicyReports にカスタムアラートルールを作成する方法は、 [Alertmanager の設定](#) を参照してください。

