Red Hat Advanced Cluster Management for Kubernetes 2.10

可観測性

可観測性

Last Updated: 2024-05-24

Red Hat Advanced Cluster Management for Kubernetes 2.10 可観測性

可観測性

法律上の通知

概要

可観測性コンポーネントを有効にして、マネージドクラスターに関する洞察を取得します。

目次

第1章 可観測性サービスについて

可観測性コンポーネントは、フリート全体のクラスターの正常性と使用率を理解するために使用できるサービスです。デフォルトでは、マルチクラスター可観測性 Operator は Red Hat Advanced Cluster Management のインストール中に有効になります。

可観測性コンポーネントの詳細は、次のドキュメントを参照してください。

- 可観測性アーキテクチャー
- 可観測性の設定
- 可観測性サービスの有効化
- 可観測性の使用
- 可観測性のカスタマイズ
- 可観測性アラート
- コンソールでの検索の概要
- Red Hat Insights での可観測性の使用

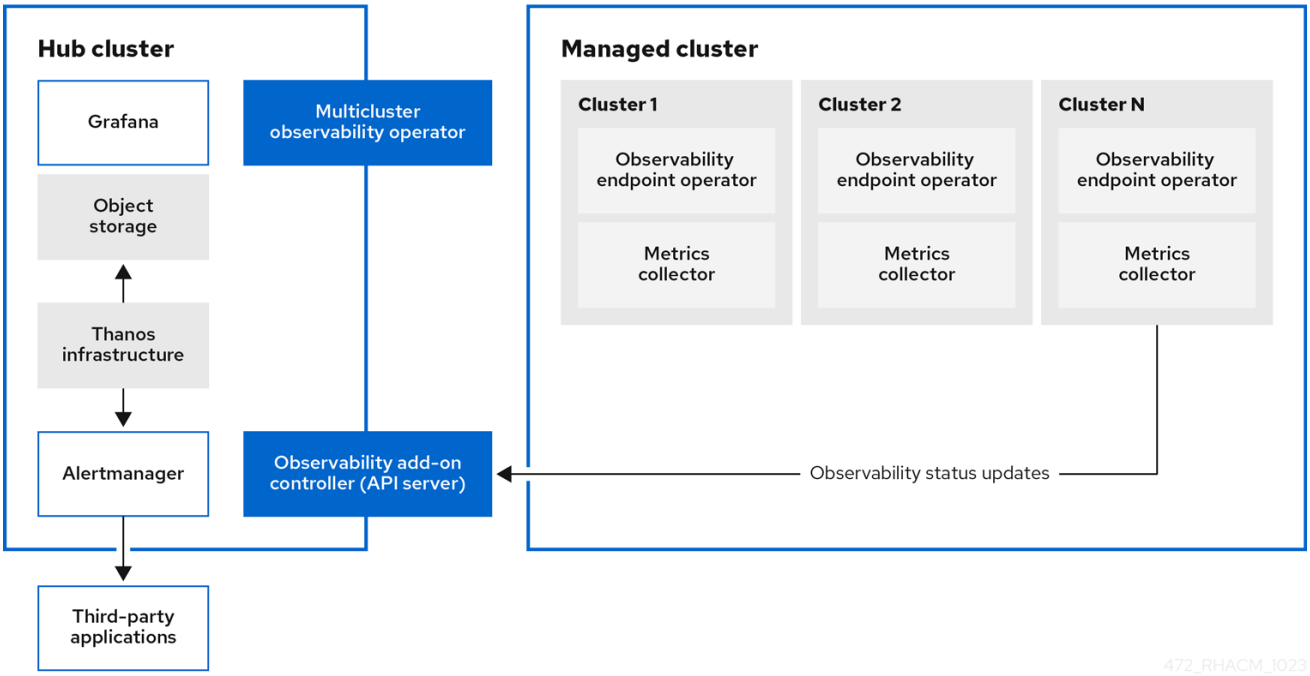
1.1. 可観測性アーキテクチャー

multiclusterhub-operator は、デフォルトで **multicluster-observability-operator** Pod を有効にします。**multicluster-observability-operator** Pod を設定する必要があります。

MultiClusterObservability カスタムリソースを定義して Observability サービスを有効にすると、インポートまたは作成された各マネージドクラスターに **observability-endpoint-operator** が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 からデータを収集するメトリクスコレクターを起動し、そのデータを Red Hat Advanced Cluster Management ハブクラスターに送信します。

Observability サービスが有効になっている場合、ハブクラスターは、ハブの自己管理が有効になっているかどうかに関係なく、設定された Thanos インスタンスにメトリクスを収集して送信するように常に設定されます。ハブクラスターがセルフマネージドの場合、**disableHubSelfManagement** パラメーターはデフォルトの設定である **false** に設定されます。ハブクラスターのメトリクスとアラートは、**local-cluster** namespace に表示されます。ハブの自己管理が有効になっている場合にのみ、**local-cluster** がクラスターリストのドロップダウンメニューに表示されます。Grafana エクスプローラーで **local-cluster** メトリクスをクエリーできます。

次の図は、可観測性の設定要素を示しています。



可観測性アーキテクチャーのコンポーネントには次の項目が含まれます。

- マルチクラスターハブオペレーター (**multiclusterhub-operator** Pod とも呼ばれます) は、**multicluster-observability-operator** Pod をデプロイします。ハブクラスターデータをマネージドクラスターに送信します。
- **可観測性アドオンコントローラー** は、マネージドクラスターのログを自動的に更新する API サーバーです。
- Thanos インフラストラクチャーには、**multicluster-observability-operator** Pod によってデプロイされる Thanos Compactor が含まれます。Thanos Compactor は、保持設定とストレージ内のデータの圧縮を使用して、クエリーが適切に実行されることを保証します。
  Thanos Compactor でいつ問題が発生しているかを特定するには、その正常性を監視する 4 つのデフォルトのアラートを使用します。次のデフォルトアラートの表を確認してください。

表1.1 デフォルトの Thanos アラートの表

アラート	重大度	説明
ACMThanosCompactHalted	critical	コンパクターが停止するとアラートが送信されます。
ACMThanosCompactHighCompactionFailures	warning	圧縮失敗率が 5% を超えると、アラートが送信されます。
ACMThanosCompactBucketHighOperationFailures	warning	バケット操作の失敗率が 5% を超えると、アラートが送信されます。
ACMThanosCompactHasNotRun	warning	コンパクターが過去 24 時間以内に何もアップロードしなかった場合、アラートが送信されます。

- 可観測性コンポーネントは、**Grafana** のインスタンスをデプロイして、ダッシュボード (静的) またはデータ探索によるデータの視覚化を可能にします。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。

- **Prometheus Alertmanager** を使用すると、サードパーティーアプリケーションでアラートを転送できます。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。Red Hat Advanced Cluster Management は、Prometheus Alertmanager のバージョン 0.25 をサポートします。

1.1.1. 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-compactor**、**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1.2 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、次回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。ただし、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 クラスターに参加して、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ー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。

1.1.2. 関連情報

- 可観測性サービスについて を参照してください。
- 可観測性の設定 を参照してください。
- 可観測性サービスの有効化 を参照してください。

1.2. 可観測性の設定

可観測性コンポーネントで収集できるメトリックと可観測性 Pod の容量について理解するには、読み続けてください。

1.2.1. 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1.3 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
config_policies_evaluation_duration_seconds_bucket	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。
config_policies_evaluation_duration_seconds_count	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。
config_policies_evaluation_duration_seconds_sum	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。
policy_governance_info	ゲージ	**type**、**policy**、**policy_namespace**、**cluster_namespace**	Stable。詳細は、**ガバナンスメトリック** を参照してください。
policyreport_info	ゲージ	**managed_cluster_id**、**category**、**policy**、**result**、**severity**	Stable。詳細は、**insight_PolicyReports_ の管理** を参照してください。
search_api_db_connection_failed_total	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。
search_api_dbquery_duration_seconds	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。

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_api_requests	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_request_count	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。
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。

1.2.2. 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1.4 可観測性 Pod の容量要求

デプロイメントまたは StatefulSet	コンテナー名	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

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

デプロイメントまたは StatefulSet	コンテナー名	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-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
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

1.2.3. 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。
- 可観測性サービスの紹介 に戻ります。

第2章 可観測性サービスの有効化

ハブクラスターで可観測性サービスを有効にすると、**multicluster-observability-operator** が新しいマネージドクラスターを監視し、メトリックおよびアラート収集サービスをマネージドクラスターに自動的にデプロイします。メトリックを使用して Grafana ダッシュボードを設定すると、クラスターリソース情報が表示され、コストを節約し、サービスの中断を防ぐことができます。

multicluster-observability-operator Pod とも呼ばれる可観測性コンポーネントを使用して、マネージドクラスターのステータスを監視します。

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
- コマンドラインインターフェイス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
- 可観測性の無効化
- 可観測性の削除

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ー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。
  - **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。複数の Red Hat Advanced Cluster Management 可観測性インストールでオブジェクトストアバケットを共有できない場合があります。したがって、インストールごとに個別のオブジェクトストアバケットを提供します。
  - 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)

2.2. コマンドラインインターフェイス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 、グローバルプルシークレットの更新 を参照してください。

4. 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。**s3.us-east-1.amazonaws.com** の URL のような Amazon S3 エンドポイントの URL を入力します。

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

- Google Cloud Platform 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 (1)
      max_retries: 0
```

(1) **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 (1)
      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。

2.2.1. AWS Security Token Service のストレージの設定

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。

1. 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。
2. 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。
3. 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) クラスター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。

2.2.2. 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=$CLUSTER_NAME-acm-observability
export REGION=us-east-2
export NAMESPACE=open-cluster-management-observability
export SA=tbd
export SCRATCH_DIR=/tmp/scratch
export OIDC_PROVIDER=$(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)
```

2.2.3. 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。

2. 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 の内容は以下のようになります。

```
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。

3. 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。

4. 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。

5. 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。

   a. あるいは、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**。

   b. OpenShift Container Platform 3.11 ダッシュボードを表示するには、**OCP 3.11** という名前のフォルダーを選択します。

6. **multicluster-observability-operator** デプロイメントにアクセスして、**multicluster-observability-operator** Pod が **multiclusterhub-operator** デプロイメントによってデプロイされていることを確認します。以下のコマンドを実行します。

```
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
```

7. リソースに関連付けられているラベルについて **multicluster-observability-operator** デプロイメントの **labels** セクションを表示します。**labels** セクションには次の詳細が含まれる場合があります。

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

. **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 4.13 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 4.13 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。

2.3. 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。

2.3.1. Thanos バージョンの検証

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。

2.4. 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。

2.4.1. 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
```

2.4.2. 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。

2.5. 可観測性の削除

MultiClusterObservability カスタムリソースを削除すると、可観測性サービスが無効化され、アンインストールされます。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。

2.6. 関連情報

- オブジェクトストレージ情報に関するクラウドプロバイダーのドキュメントへのリンク:
  - 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)
- 可観測性の使用 を参照してください。

- 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。
- その他の関連トピックについては、Observability サービスの概要 に戻ってください。

第3章 可観測性の使用

可観測性サービスを使用して、フリート全体のクラスターの使用率を表示します。

- 可観測性 API を使用したメトリックのクエリー
- 外部エンドポイントへのメトリクスのエクスポート
- データの表示およびデプロイメント

3.1. 可観測性 API を使用したメトリックのクエリー

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。**rbac-query-proxy** ルートのクエリーを取得するには、以下のオプションを参照してください。

- 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
```

- OpenShift OAuth アクセストークンを使用して **rbac-query-proxy** ルートにアクセスすることもできます。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 を参照してください。

可観測性用の **proxy-byo-cert** シークレットを作成するには、次の手順を実行します。

1. 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
```

2. 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。

3. 以下のコマンドを実行して、生成された証明書を使用して **proxy-byo-ca** 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
```

4. 以下のコマンドを使用して、生成された証明書を使用して **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-cert --cert
./ingress.crt --key ./ingress.key
```

3.2. 外部エンドポイントへのメトリクスのエクスポート

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ックをエクスポートします。メトリックを外部エンドポイントにエクスポートするには、次の手順を実行します。

1. **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、外部エンドポイントの Kubernetes シークレットを作成します。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

名前	説明	スキーマ
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 (**insecure_skip_verify** 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
insecure_skip_verify (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. エクスポートする外部エンドポイントのリストを追加するには、**writeStorage** パラメーターを **MultiClusterObservability** カスタムリソースに追加します。以下の例を参照してください。

```
spec:
  storageConfig:
    writeStorage: 1
    - key: ep.yaml
      name: victoriametrics
```

1 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。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。

3. メトリックのエクスポートが有効になった後、**acm_remote_write_requests_total** メトリックを確認して、メトリックのエクスポートのステータスを表示します。

   a. ハブクラスターの OpenShift Container Platform 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。

   b. 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。

3.3. ダッシュボードを使用した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、単一ノードの OpenShift クラスターから **cluster_infrastructure_provider** アラートを確認するには、**cluster_infrastructure_provider{clusterType="SNO"}** のクエリー式を使用します。

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。

3.3.1. 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ーデータが表示されます。

3.3.2. Red Hat Advanced Cluster Management ダッシュボードの表示

Red Hat Advanced Cluster Management 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利用可能になります。以下は、ダッシュボードの説明です。

- **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。

3.3.3. 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 テーブルを表示して、データストアとしての etcd の安定性を確認することもできます。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。

3.3.4. Kubernetes API サーバーダッシュボードの表示

以下のオプションを表示して、Kubernetes API サーバーダッシュボードを表示します。

- 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。
  1. Grafana ダッシュボードに移動します。
  2. **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が表示されます。
- 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。
  1. ハブクラスターから Grafana ダッシュボードに移動します。
  2. **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、および傾向が表示されます。

3.4. 関連情報

- 詳細は、Prometheus Remote-Write 仕様 を参照してください。
- 可観測性サービスの有効化 を参照してください。
- その他のトピックは、可観測性サービスの概要 を確認してください。

3.5. 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。Grafana アラートダッシュボードに表示されるデータは、マネージドクラスターから発信される **alerts** メトリクスに依存します。**alerts** メトリクスは、ハブクラスター上の Red Hat Advanced Cluster

Management アラートマネージャーにアラートを転送するマネージドクラスターには影響しません。したがって、メトリクスとアラートには異なる伝播メカニズムがあり、それぞれ別のコードパスに従います。

Grafana アラートダッシュボードにデータが表示されている場合でも、マネージドクラスターアラートが Red Hat Advanced Cluster Management ハブクラスターアラートマネージャーに正常に転送されているという保証はありません。メトリクスがマネージドクラスターから伝播されている場合は、Grafana アラートダッシュボードにデータが表示されます。

開発ニーズに合わせて Grafana ダッシュボードを使用するには、以下を実行します。

- Grafana 開発者インスタンスの設定
- 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

3.5.1. 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。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. **open-cluster-management/multicluster-observability-operator/** リポジトリーのクローンを作成し、**tools** フォルダーにあるスクリプトを実行できるようにします。
2. **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
```

3. **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 開発者インスタンスを設定します。

3.5.1.1. 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。

3.5.2. Grafana ダッシュボードの設計

Grafana インスタンスを設定したら、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. データソースセレクターから **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。

3.5.2.1. 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。**uuidegen** 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。

3.5.3. 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
```

3.5.4. 関連情報

- 外部エンドポイントへのメトリクスのエクスポート を参照してください。
- 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。
- トピックについては、環境の監視の紹介 を参照してください。

第4章 可観測性設定のカスタマイズ

可観測性を有効にした後、環境の特定のニーズに合わせて可観測性設定をカスタマイズします。

可観測性サービスが収集するクラスターフリートデータを管理および表示する方法の詳細は、次のセクションをお読みください。

必要なアクセス権限: クラスターの管理者

- カスタムルールの作成
- カスタムメトリックの追加
- **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
- 永続ボリュームおよび永続ボリューム要求 (PVC) の増減
- 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
- 可観測性アドオンのプロキシー設定
- 可観測性アドオンのプロキシー設定の無効化

4.1. 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。

負荷の高い式を事前計算するには、記録ルール機能を使用します。結果は新たな時系列のセットとして保存されます。アラートルールを使用すると、外部サービスにアラートを送信する方法に基づいてアラート条件を指定できます。

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

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。以下のカスタムルールの例を確認してください。

- カスタムアラートルールを作成します。**open-cluster-management-observability** 名前空間に **thanos-ruler-custom-rules** という名前の config map を作成します。以下の例のように、**custom_rules.yaml** キーに名前を付ける必要があります。設定には、複数のルールを作成できます。
  - 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します。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
```

  - デフォルトのアラートルールは、**open-cluster-management-observability** namespace の **thanos-ruler-default-rules** config map にあります。

- **thanos-ruler-custom-rules** config map 内にカスタム記録ルールを作成します。Pod のコンテナー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します。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: config map に変更を加えた後、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、設定が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 ダッシュボードに移動し、**Explore** ページを選択して、**ALERTS** にクエリーを実行します。アラートを作成した場合、アラートは Grafana でのみ使用できます。

4.2. カスタムメトリックの追加

metrics_list.yaml ファイルにメトリクスを追加して、マネージドクラスターから収集します。以下の手順を実行します。

1. カスタムメトリックを追加する前に、次のコマンドを使用して **mco observability** が有効になっていることを確認します。

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

2. **status.conditions.message** セクションで、以下のメッセージを確認します。

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

3. 以下のコマンドで、**open-cluster-management-observability** namespace に **observability-metrics-custom-allowlist** ConfigMap を作成します。

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

4. **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: 1
      - node_memory_MemTotal_bytes
    rules: 2
    - 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))
```

1. **オプション:** 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。

2. **オプション: expr** と **record** パラメーターのペアに値を 1 つだけ入力して、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。

セクションのいずれかまたは両方を使用できます。

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。

5. Grafana ダッシュボードの **Explore** ページからメトリクスをクエリーして、カスタムメトリクスからのデータコレクションを確認します。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。

4.2.1. 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 ユーザー定義のメトリクスを収集し、Grafana ダッシュボードからメトリクスを表示します。以下の手順を実行します。

1. OpenShift Container Platform クラスターでモニタリングを有効にします。**関連情報** セクションの **ユーザー定義プロジェクトの監視の有効化** 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。

2. ユーザーワークロードメトリックを **observability-metrics-custom-allowlist** Config Map に追加して、**test** namespace のメトリックを収集します。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: 1
 names: 2
   - sample_metrics
```

1 config map データのキーを入力します。

2 Config Map データの値を YAML 形式で入力します。**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。config map を作成すると、可観測性コレクターはメトリクスを収集し、ターゲット namespace からハブクラスターにプッシュします。

4.2.2. デフォルトメトリックの削除

マネージドクラスターから特定のメトリクスのデータを収集したくない場合は、**observability-metrics-custom-allowlist.yaml** ファイルからメトリクスを削除します。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。デフォルトメトリックを削除するには、以下の手順を実施します。

1. 以下のコマンドを使用して、**mco observability** が有効になっていることを確認します。

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

2. 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。次のメトリックの例を確認してください。

```
-cluster_infrastructure_provider`
```

3. 以下のコマンドで、**open-cluster-management-observability** namespace に **observability-metrics-custom-allowlist** ConfigMap を作成します。

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

4. 可観測性サービスがマネージドクラスターから特定のメトリクスを収集し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。

4.3. 保持の詳細設定の追加

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、以下のコマンドで **advanced** セクションを追加します。

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

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** ドキュメントを参照してください。

4.4. シングルノード 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。デフォルトでは、シングルノードの OpenShift クラスターは Pod およびコンテナーのリソースメトリクスを収集しません。シングルノードの OpenShift クラスターが特定のレベルのリソース消費に達すると、定義された詳細なメトリクス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し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/ワークロード収集ルールには次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の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** 収集ルールは、シングルノード OpenShift クラスターの全体的なメモリー使用率が使用可能なノードメモリーの 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、ルールの開始時のみ Grafana に表示されます。

4.5. 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** ドキュメントを参照してください。

4.6. 永続ボリュームおよび永続ボリューム要求 (PVC) の増減

永続ボリュームと永続ボリューム要求を増減して、ストレージクラス内のストレージの量を変更します。以下の手順を実行します。

1. ストレージクラスがボリュームの拡張をサポートしている場合は、**MultiClusterObservability** カスタムリソースを更新して、永続ボリュームのサイズを増やします。

2. 永続ボリュームのサイズを小さくするには、永続ボリュームを使用している Pod を削除し、永続ボリュームを削除して再作成します。永続ボリュームでデータが失われる可能性があります。以下の手順を実行します。

    a. **MultiClusterObservability** カスタムリソースにアノテーション **mco-pause: "true"** を追加して、**MultiClusterObservability** Operator を一時停止します。

    b. 目的のコンポーネントのステートフルセットまたはデプロイメントを探します。レプリカ数を **0** に変更します。これによりシャットダウンが開始され、データの損失を避けるために、該当する場合はローカルデータがアップロードされます。たとえば、Thanos **Receive** ステートフルセットの名前は **observability-thanos-receive-default** で、デフォルトでは 3 つのレプリカがあります。したがって、次の永続ボリューム要求を探します。

        - **data-observability-thanos-receive-default-0**
        - **data-observability-thanos-receive-default-1**

- **data-observability-thanos-receive-default-2**

c. 必要なコンポーネントによって使用される永続ボリュームおよび永続ボリューム要求を削除します。

d. **MultiClusterObservability** カスタムリソースで、コンポーネントの設定のストレージサイズを、ストレージサイズフィールドで必要な量に編集します。接頭辞にはコンポーネントの名前が付いています。

e. 以前に追加したアノテーションを削除して **MultiClusterObservability** Operator の一時停止を解除します。

f. Operator を一時停止した後に調整を開始するには、**multicluster-observability-operator** および **observatorium- operator** Pod を削除します。Pod はすぐに再作成され、調整されます。

3. **MultiClusterObservability** カスタムリソースをチェックして、永続ボリュームとボリューム要求が更新されていることを確認します。

4.7. 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 **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。

詳細は、ガバナンスドキュメントの **alertmanager ルートの証明書の置き換え** を参照してください。

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。

4.8. 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
```

2. 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、以前のシークレットに **public.crt** キーと **private.key** キー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。

```
thanos.yaml: |
  type: s3
  config:
    ...
    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) 証明書と、**thanos-object-storage** シークレットのキー値へのパス。

3. **MultiClusterObservability** カスタムリソースの **tlsSecretName** パラメーターと **tlsSecretMountPath** パラメーターを更新して、シークレット名とマウントパスを設定します。以下の例を参照してください。この例では、シークレット名が **tls-certs-secret** で、証明書とキー値へのパスは、前の例で使用したディレクトリーを使用しています。

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

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。

4.9. 可観測性アドオンのプロキシー設定

マネージドクラスターからの通信が HTTP および HTTPS プロキシーサーバー経由でハブクラスターにアクセスできるようにプロキシー設定を指定します。通常、アドオンでは、ハブクラスターとマネージドクラスターの間で HTTP および HTTPS プロキシーサーバーをサポートする特別な設定は必要ありません。ただし、可観測性アドオンを有効にしている場合は、プロキシー設定を完了する必要があります。

4.10. 前提条件

- ハブクラスターがある。
- ハブクラスターとマネージドクラスター間のプロキシー設定が有効にしている。

可観測性アドオンのプロキシー設定を指定するには、以下の手順を実行します。

1. ハブクラスターのクラスター namespace に移動します。
2. **spec.proxyConfig** パラメーターを追加して、プロキシー設定を使用して **AddOnDeploymentConfig** リソースを作成します。以下は、YAML の例です。

```
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. このフィールドには、HTTP プロキシーまたは HTTPS プロキシーのいずれかを指定できます。
2. **kube-apiserver** の IP アドレスを含めます。

3. マネージドクラスターで IP アドレスを取得するには、以下のコマンドを実行します。

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

4. **ManagedClusterAddOn** リソースに移動し、作成した **AddOnDeploymentConfig** リソースを参照して更新します。以下は、YAML の例です。

```
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>
```

5. プロキシー設定を確認してください。プロキシー設定が正常に指定されている場合に、マネージドクラスター上の可観測性アドオンエージェントによってデプロイされたメトリックコレクターはデータをハブクラスターに送信します。以下の手順を実行します。
   a. ハブクラスターに移動し、Grafana ダッシュボードでマネージドクラスターに移動します。
   b. プロキシー設定のメトリクスを表示します。

4.11. 可観測性アドオンのプロキシー設定の無効化

開発に必要な変更がある場合は、ハブクラスターとマネージドクラスターに設定した可観測性アドオンのプロキシー設定を無効にすることが必要な場合があります。可観測性アドオンのプロキシー設定はいつでも無効にできます。以下の手順を実行します。

1. **ManagedClusterAddOn** リソースに移動します。
2. 参照される **AddOnDeploymentConfig** リソースを削除します。

4.12. 関連情報

- 詳細は 、Prometheus の設定 を参照してください。記録ルールとアラートルールの詳細は、Prometheus ドキュメント の記録ルールとアラートルールを参照してください。
- 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。
- 外部エンドポイントへのメトリクスのエクスポート を参照してください。
- 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。
- 可観測性 API を参照してください。
- alertmanager ルートの証明書の更新に関する詳細は、alertmanager の 証明書の置き換え を参照してください。
- 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、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5章 可観測性アラートの管理

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
- アラートをサイレンスにする
- アラートの抑制

5.1. 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。

5.2. 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ーを設定する場合は、**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://****
```

5.2.1. 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。

5.3. 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 にアクセスできる必要があります。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くない場合は、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)
```

5.4. 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: (1)
  - equal:
      - namespace
    source_match: (2)
      severity: critical
    target_match_re:
      severity: warning|info
```

(1) **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。一致するものがあった場合、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"}
```

(2) **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
```

5.5. 関連情報

- 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6章 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、**observabilty-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: 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> 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

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の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
```

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。

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>
```

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7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

7.1. 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7.1 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。

7.2. 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 1 リソースを、**indexer**、**database**、**queryapi**、**collector** のPod に適用できます。

2 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
```

7.3. 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。

7.4. 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
- 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

7.4.1. 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。

7.4.2. 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"}}'
```

7.4.3. 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 などのプロパティー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、検索にプロパティーセレクターを追加します。プロパティーに関連する値を組み合わせて、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ー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。

7.4.4. 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。

注記: コンソールで API Explorer を使用して、マネージドクラスターに定義されているすべてのリソースを検出できます。**oc api-resources**コマンドを実行しても、すべてのリソースを検出できます。

環境の監視の概要 に戻ります。

第8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ーリスクの特定、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モニタリングを提供します。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

8.1. 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。

8.2. 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。

8.3. 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。

8.3.1. 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。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の **Search** アイコンをクリックして **Search** ページに移動します。
3. **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。
4. Insights ポリシー違反とカテゴリーを使用してクエリーを指定できます。**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。
5. 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
```

8.3.2. 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。以下の手順を実行します。

1. Red Hat Advanced Cluster Management クラスターにログインします。
2. ナビゲーションメニューから **Overview** を選択します。
3. 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。
   a. または、ナビゲーションメニューから **Clusters** を選択できます。
   b. 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。
   c. **Status** カードから、特定された問題の数を表示します。
4. 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。
5. 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。
6. **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。
   a. **Governance** ページに移動し、特定の **PolicyReport** を選択します。
   b. **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。
   c. **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。

8.3.3. 関連情報

- **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。