Red Hat Advanced Cluster Management for Kubernetes 2.6

可観測性

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

Last Updated: 2023-11-01

Red Hat Advanced Cluster Management for Kubernetes 2.6 可観測性

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

法律上の通知

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

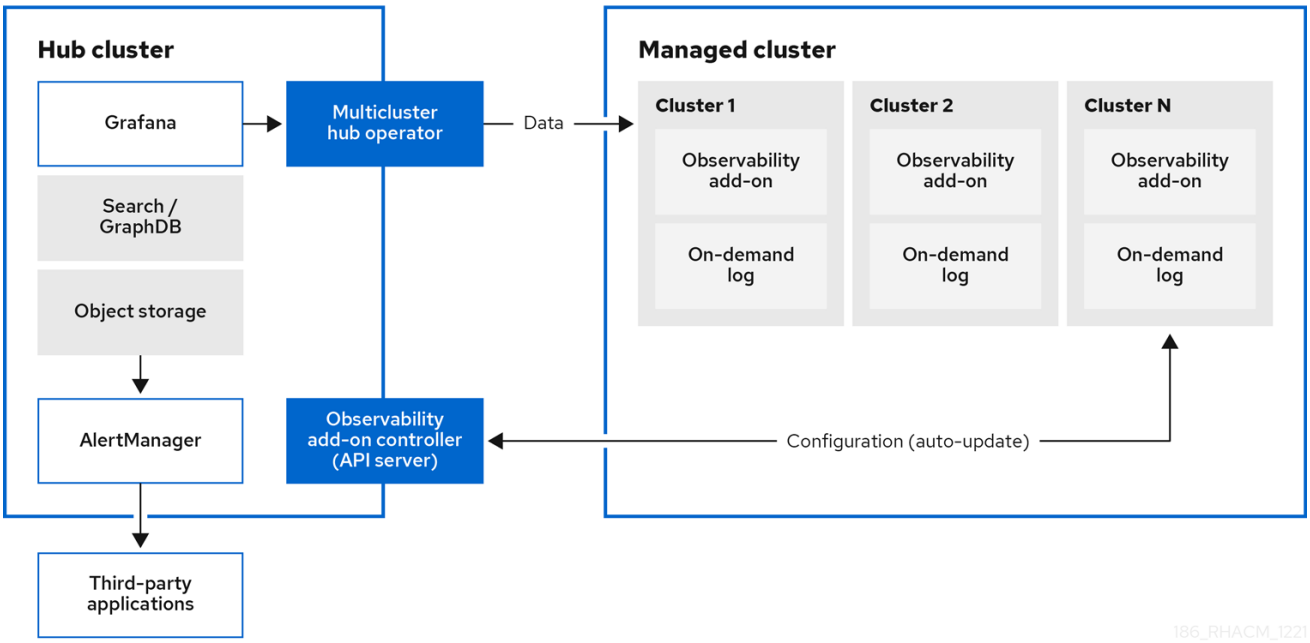
第1章 環境の監視の紹介

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
- 可観測性サービスの有効化
- コンソールでの検索
- 可観測性のカスタマイズ
- Grafana ダッシュボードの設計
- Red Hat Insights の可観測性
- Insights PolicyReports の管理

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。



注記: **オンデマンドログ** は、特定 の Pod のログをリアルタイムで取得するエンジニア用のアクセスを提供します。ハブクラスターからのログは集約されません。これらのログは、検索サービスとコンソールの他の部分を使用してアクセスできます。

- 可観測性サービス
- メトリクスのタイプ
- 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
- サポート

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。Red Hat Advanced Cluster Management は、以下の S3 互換のある、安定したオブジェクトストアをサポートします。

- Amazon S3
  注記: Thanos のオブジェクトストアインターフェイスは、AWS S3 RESTFUL API 互換の API、または Minio や Ceph などのその他の S3 と互換のあるオブジェクトストアをサポートします。
- Google Cloud Storage
- Azure ストレージ
- Red Hat OpenShift Data Foundation
  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。サポートされるオブジェクトストアの全一覧は、Thanos のドキュメント 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、メトリクス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 8.1.3 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。

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。

1.1.2. メトリクスのタイプ

デフォルトで、OpenShift Container Platform は Telemetry サービスを使用してメトリクスを 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	Status
acm_managed_cluster_info	ゲージ	**hub_cluster_id**、**managed_cluster_id**、**vendor**、**cloud**、**version**、**available**、**created_via**、**core_worker**、**socket_worker**	安定
policy_governance_info	ゲージ	**type**、**policy**、**policy_namespace**、**cluster_namespace**	安定。詳細は、ガバナンスのメトリクス を参照してください。
policyreport_info	ゲージ	**managed_cluster_id**、**category**、**policy**、**result**、**severity**	安定。詳細は、インサイト PolicyReports の管理 を参照してください。
config_policies_evaluation_duration_seconds_bucket	ヒストグラム	なし。	安定。詳細は、ガバナンスのメトリクス を参照してください。
config_policies_evaluation_duration_seconds_count	ヒストグラム	なし。	安定。詳細は、ガバナンスのメトリクス を参照してください。
config_policies_evaluation_duration_seconds_sum	ヒストグラム	なし。	安定。詳細は、ガバナンスのメトリクス 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。

1.1.3. 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 容量要求の一覧です。

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 CRD	45	128	3	135	384
	exporter	5	50	3	15	150

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-controller	thanos-receive-controller	4	32	1	4	32
observability-thanos-receive-default	thanos-receive	300	512	3	900	1536
observability-thanos-rule	thanos-rule	50	512	3	150	1536
	configmap-reloader	4	25	3	12	75
observability-thanos-store-memcached	Memcached CRD	45	128	3	135	384
	exporter	5	50	3	15	150
observability-thanos-store-shard	thanos-store	100	1024	3	300	3072

1.1.4. 可観測性サービスで使用される永続ストア

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** 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に、ブロックストレージを使用することをお勧めします。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1.3 永続ボリュームの表一覧

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。

1.1.5. 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。

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。

可観測性サービスの設定、メトリック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。

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
- 可観測性の有効化
- MultiClusterObservability CR 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、**MultiClusterObservability** CR 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。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。Thanos がサポートするオブジェクトストアの詳細は、Thanos のドキュメント を参照してください。

1.2.2. 可観測性の有効化

MultiClusterObservability カスタムリソース (CR) 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、必ず可観測性名前空間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。

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
      insecure: true
      access_key: YOUR_ACCESS_KEY
      secret_key: YOUR_SECRET_KEY
```

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
      max_retries: 0
```

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
      insecure: false
      access_key: YOUR_RH_DATA_FOUNDATION_ACCESS_KEY
      secret_key: YOUR_RH_DATA_FOUNDATION_SECRET_KEY
```

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
```

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

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) クラスターでオブザーバビリティを設定して 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。詳細については、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。

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 CR 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 (CR) 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
```

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。

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)
```

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

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:
      eks.amazonaws.com/role-arn=$S3_ROLE
    store:
      eks.amazonaws.com/role-arn=$S3_ROLE
    rule:
      eks.amazonaws.com/role-arn=$S3_ROLE
    receive:
      eks.amazonaws.com/role-arn=$S3_ROLE
    query:
      eks.amazonaws.com/role-arn=$S3_ROLE
```

   詳細については、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 に全 Pod を作成します。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。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**。**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。

1.2.3. 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 CR 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。

1.2.3.1. 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**) を使用してメトリクス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クス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、**cluster_infrastructure_provider** メトリクスのクエリーには、前述したコマンドの URL を **https://{PROXY_ROUTE_URL}/api/v1/query?query=cluster_infrastructure_provider** の URL に置き換えます。詳細は、prometheus のクエリー を参照してください。

- **rbac-query-proxy** ルートの証明書を置き換えることもできます。証明書を生成するための OpenSSL コマンド を参照して、証明書を作成します。**csr.cnf** をカスタマイズする時に、**DNS.1** を **rbac-query-proxy** ルートのホスト名に更新します。
  - 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
```

1.2.3.2. 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 およびコンテナーのリソースメトリクスを収集しません。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。

メトリクスは、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。これらのメトリクス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、対応するメトリクスが収集されると、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クスのセット。
- 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** 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クス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クスの一覧を表示します。

```
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、メトリクスコレクションは以前の動作に戻ります。これらのメトリクスは定期的に、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。

1.2.4. 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。

1.3. コンソールでの検索

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。検索すると、Kubernetes リソースや関係を他のリソースにインデックス化します。ストレージクラスとストレージサイズを変更する場合

は、**searchcustomization** カスタムリソースを作成して、検索永続性のストレージ設定を定義できます。

- 検索コンポーネント
- 検索カスタマイズ
  - 再ディスグラフメモリーを増やすためのオプション
  - 保存済み検索制限の更新
- 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

1.3.1. 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。

- **collector**: Kubernetes リソースを監視し、インデックスを作成します。**search-collector** は、マネージドクラスター内のリソースの関係を計算します。
- **aggregator**: コレクターからデータを受け取り、データベースに書き込みます。**search-aggregator** は、ハブクラスターのリソースを監視し、マルチクラスターの関係を計算して、接続されたコレクターからのアクティビティーを追跡します。
- **Search API**: 検索インデックスでデータにアクセスできるようにして、ロールベースのアクセス制御を有効にします。

検索はデフォルトで有効となっています。また、マネージドクラスターのプロビジョニングまたは手動でのインポート時にも検索が有効です。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。

1.3.2. 検索カスタマイズ

Red Hat Advanced Cluster Management をインストールすると、インメモリーデータをファイルシステムに永続化するように製品が設定されます。StatefulSet **search-redisgraph** は Redisgraph Pod をデプロイし、これは **persist** という名前の永続ボリュームをマウントします。クラスターにデフォルトのストレージクラスが定義されている場合、検索コンポーネントはデフォルトのストレージクラスに 10Gi の Persistent Volume Claims (PVC) を作成します。デフォルトのストレージクラスがクラスターに存在しない場合は、検索によりインデックスが空のディレクトリー (**emptyDir**) に保存されます。

searchcustomization カスタムリソースを作成することで、検索のストレージ設定をカスタマイズできます。検索カスタマイズは namespace にスコープ指定され、検索がハブクラスターにインストールされている場所にあります。次の検索カスタマイズのカスタムリソースの例をご覧ください。

```
apiVersion: search.open-cluster-management.io/v1alpha1
kind: SearchCustomization
metadata:
  name: searchcustomization
  namespace: open-cluster-management
  labels:
    cluster.open-cluster-management.io/backup: ""
spec:
```

```
  persistence: true
  storageClass: gp2
  storageSize: 12Gi
```

次のコマンドを実行して、検索カスタマイズのカスタムリソース定義を表示します。

```
oc get crd searchcustomizations.search.open-cluster-management.io -o yaml
```

カスタマイズカスタムリソースで **persistence** フラグを **false** に更新することで、永続性を無効にすることができます。これにより、ファイルシステムへの検索インデックスの保存がオフになります。永続性のステータスは、検索演算子 (**searchoperator**) カスタムリソースから取得できます。コマンド **oc get searchoperator searchoperator -o yaml** を実行して、検索演算子のカスタムリソースを表示します。

1.3.2.1. 再ディスグラフメモリーを増やすためのオプション

再ディスグラフは、オブジェクトの数がキャッシュされるにつれてメモリーを直線的に増やす必要があるインメモリーデータベースです。多くのマネージドクラスターまたは多数の Kubernetes オブジェクトを含む Red Hat Advanced Cluster Management クラスターでは、redisgraph Pod のメモリー更新を制限する必要があります (**search-redisgraph-0**)。

デフォルトでは redisgraph Pod (**search-redisgraph-0**) は、メモリーの上限が **4Gi** としてデプロイされます。サイズの大きいクラスターを管理する場合には、ハブクラスターの namespace で **searchoperator** の **redisgraph_resource.limit_memory** を編集して、この上限を増やす必要があります。たとえば、次のコマンドを使用して上限を **8Gi** に更新できます。

```
oc patch searchoperator searchoperator --type='merge' -p '{"spec":{"redisgraph_resource":
{"limit_memory":"8Gi"}}}'
```

変更が行われると、**search-redisgraph** Pod は更新された設定で自動的に再起動します。

1.3.2.2. 保存済み検索制限の更新

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。**administrator** ロールを持つユーザーのみが、キー値 **key:value: SAVED_SEARCH_LIMIT: x** を **console-config** ConfigMap に追加して制限を更新できます。

1.3.3. 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。

以下の手順に従って、検索でクエリーを実行します。

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、一覧表示されます。**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。以下の名前、namespace、ステータス、その他のリソースフィールドの例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。
- デフォルトでは、検索ドロップダウンメニューには 2500 イメージという制限があります。イメージ制限を引き上げるには、**defaultImageQueryLimit** 環境変数を追加して **search-api** デプロイメントを編集します。検索を使用してデプロイメントを検索できます。以下の例を参照してください。

```
name: defaultImageQueryLimit
 value: x (1)
```

(1) **X** は、検索ドロップダウンメニューから表示するイメージの数を表します。

  - または、以下のコマンドを使用してデプロイメントにパッチを適用できます。

```
oc patch deployment search-api -n open-cluster-management -p '{"spec":
{"template": {"spec": {"containers": [{"name": "search-api","env": [{"name":
"defaultImageQueryLimit","value": "X"}]}]}}}}'
```

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。

1.3.3.1. 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。

1.3.4. 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。

Red Hat Advanced Cluster Management for Kubernetes コンソールの詳細は、Web console を参照してください。

1.4. 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
- AlertManager の設定
- カスタムメトリクスの追加
- デフォルトメトリクスの削除

- 外部エンドポイントへのメトリクスのエクスポート
  - 外部エンドポイントの Kubernetes シークレットの作成
  - MultiClusterObservability CR の更新
  - メトリックエクスポートのステータスの表示
- **詳細** 設定の追加
- コンソールからの **multiclusterobservability** CR レプリカの更新
- 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
- 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展開
  - etcd テーブルの表示
  - 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
  - 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
- 可観測性の無効化

1.4.1. 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。詳細は、Prometheus configuration を参照してください。

- 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
        recording_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。

アラートルールが適切に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。

1.4.2. 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 の例については、prometheus/alertmanager を参照してください。

1.4.3. カスタムメトリクスの追加

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されるようにします。

カスタムメトリクスを追加する前に、**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))
```

- **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クス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。メトリクスは、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クスを使用することもできます。ダッシュボードの表示に関する詳細は、Grafana ダッシュボードの設計 を参照してください。

1.4.4. デフォルトメトリクスの削除

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。

1.4.5. 外部エンドポイントへのメトリックのエクスポート

可観測性をカスタマイズして、Prometheus Remote Write プロトコル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できます。詳細は、Prometheus Remote Write プロトコル を参照してください。

1.4.5.1. 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、次のステップの **multiclusterobservability** CR で使用されます。現在、可観測性は、セキュリティーチェックなし、基本認証、または **tls** の有効化を使用して、エンドポイントへのメトリック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 **optional**	基本認証用の HTTP クライアント設定。	BasicAuth
tls_config **optional**	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 **optional**	基本認証用のパスワード。	string

TLSConfig

名前	説明	スキーマ
secret_name **required**	証明書を含むシークレットの名前。	string

ca_file_key **optional**	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 **required**	シークレット内のクライアント証明書のキー。	string
key_file_key **required**	シークレットのクライアントキーのキー。	string
insecure_skip_verify **optional**	ターゲット証明書の検証をスキップするパラメーター。	ブール型

1.4.5.2. MultiClusterObservability CR の更新

Kubernetes シークレットを作成した後、**multiclusterobservability** CR 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

1.4.5.3. 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。

1.4.6. 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 CR を編集し、**oc edit mco observability -o yaml** コマンドで **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。

1.4.7. コンソールからの multiclusterobservability CR 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 (CR) を見つけ、レプリカを変更するコンポーネントの **replicas** パラメーターの値を更新します。更新した YAML は以下のようになります。

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

observability CR 内のパラメーターの詳細は、Observability API を参照してください。

1.4.8. 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://****
```

詳細は、Prometheus Alertmanager のドキュメント を参照してください。

1.4.8.1. マネージドクラスターの転送アラートの無効化

マネージドクラスターのアラート転送を無効にします。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 に加えられた変更は元に戻ります。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: メトリック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ックは失われ、Prometheus インスタンスが再起動されます。ただし、ハブクラスターからのメトリック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。

1.4.9. 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)
```

1.4.10. 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
```

1.4.11. ルート認定のカスタマイズ

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。

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。

1.4.11.1. 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できます。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
```

open-cluster-management-observability namespace にシークレットを指定する必要があります。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
          cert_file: /etc/minio/certs/public.crt
          key_file: /etc/minio/certs/private.key
          insecure_skip_verify: false
```

MultiClusterObservability CR でシークレット名、**TLSSecretName** パラメーター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
```

このシークレットは、オブジェクトストアにアクセスする必要のあるすべてのコンポーネントにマウントでき、次のコンポーネントが含まれます: **receiver**、**store**、**ruler**、**compact**。

1.4.12. データの表示および展開

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: 単一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。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。

1.4.12.1. 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。

1.4.12.2. 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。

1.4.12.3. 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。

1.4.13. 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。

1.4.13.1. 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
```

1.4.13.2. 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。

可観測性サービスを使用したコンソールでのデータの監視に関する詳細は、環境の監視の紹介 を参照してください。

1.5. GRAFANA ダッシュボードの設計

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。

- 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

1.5.1. Grafana 開発者インスタンスの設定

まず、**stolostron/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。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。

1.5.2. 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。

1.5.2.1. 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。

1.5.3. 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
```

1.6. RED HAT INSIGHTS の可観測性

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

1.6.1. 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。

1.6.2. 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。

- **説明** 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。

1.7. 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

1.7.1. 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ー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
```

1.7.2. 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。