



Red Hat Advanced Cluster Management for Kubernetes 2.2

環境の監視

可観測性サービスを有効にしてカスタマイズしてマネージドクラスターを最適化する方法については、[こちらを参照してください](#)。

Red Hat Advanced Cluster Management for Kubernetes 2.2 環境の監視

可観測性サービスを有効にしてカスタマイズしてマネージドクラスターを最適化する方法については、こちらを参照してください。

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Observing_environments.ent file |.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

可観測性サービスを有効にしてカスタマイズしてマネージドクラスターを最適化する方法については、こちらを参照してください。

目次

第1章 環境の監視の紹介	3
1.1. 環境の監視	3
1.1.1. 可観測性サービス	4
1.1.1.1. 可観測性の証明書	4
1.1.1.2. メトリクスのタイプ	5
1.1.1.3. 可観測性 Pod の容量要求	5
1.1.2. 可観測性サービスで使用する永続ストア	7
1.2. 可観測性サービスの有効化	8
1.2.1. 前提条件	8
1.2.2. 可観測性の有効化	9
1.2.2.1. MultiClusterObservability CR の作成	11
1.2.3. 可観測性の無効化	13
1.3. 可観測性のカスタマイズ	13
1.3.1. カスタムルールの作成	13
1.3.2. AlertManager ルールの設定	14
1.3.3. カスタムメトリクスの追加	15
1.3.4. データの表示および展開	16
1.3.5. 可観測性の無効化	16
1.3.5.1. 全クラスターの可観測性の無効化	16
1.3.5.2. 単一クラスターの可観測性の無効化	17
1.4. GRAFANA ダッシュボードの設計	17
1.4.1. Grafana 開発者インスタンスの設定	17
1.4.2. Grafana ダッシュボードの設計	18
1.4.2.1. ConfigMap での Grafana ダッシュボードの設計	18
1.4.3. Grafana 開発者インスタンスのアンインストール	19

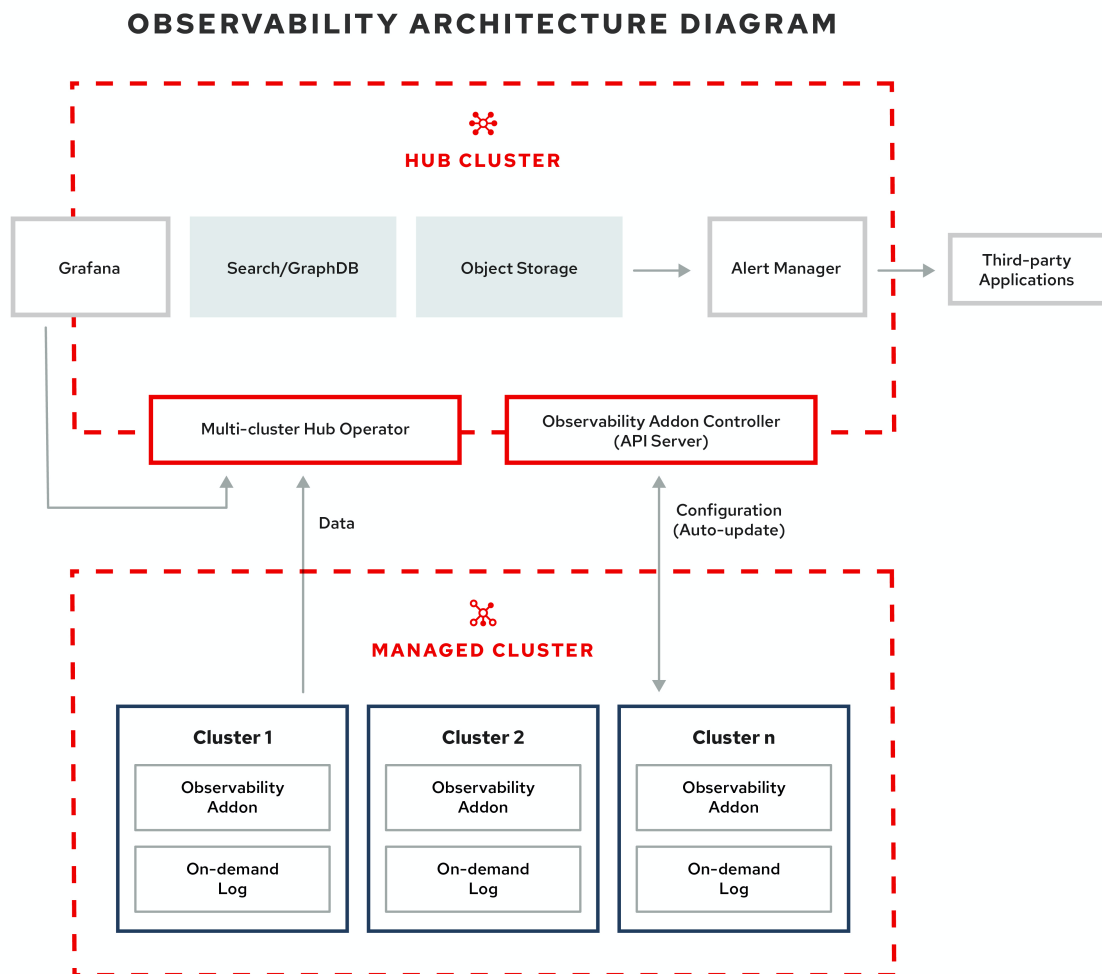
第1章 環境の監視の紹介

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
- 可観測性のカスタマイズ
- Grafana ダッシュボードの設計

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。可観測性サービス Operator (**multicluster-observability-operator**) を有効化して、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](#)
- [可観測性の証明書](#)
- [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する永続ストア](#)

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、有効化されません。永続ストレージの要件で、可観測性サービスはデフォルトで有効化されていません。Red Hat Advanced Cluster Management は、以下の安定したオブジェクトストアをサポートします。

- Amazon S3 (または Ceph など、他の S3 と互換性のあるオブジェクトストア)
- Google Cloud Storage
- Azure ストレージ
- Red Hat OpenShift Container Storage

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。

ハブクラスターで可観測性を有効にすると、ハブクラスターを **local-cluster** というマネージドクラスターとして処理してメトリクスを収集します。

注記: Red Hat Advanced Cluster Management では、**metrics-collector** は Red Hat OpenShift Container Platform 4.x クラスターでのみサポートされます。

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 6.4.x 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。

可観測性の有効化に関する詳細は、「[可観測性サービスの有効化](#)」を参照してください。

1.1.1.1. 可観測性の証明書

可観測性証明書は、期限が切れると自動的に更新されます。以下の一覧を表示し、証明書が自動更新される場合の影響について確認します。

- ハブクラスターのコンポーネントは自動的に再起動し、更新された証明書を取得します。

- Red Hat Advanced Cluster Management は、更新された証明書マネージャークラスターに送信します。
- **metrics-collector** は再起動して、更新された証明書をマウントします。
注記: **metrics-collector** は、証明書の削除前および削除後にメトリクスをハブクラスターにプッシュできます。クラスター全体での証明書の更新に関する詳細は、「[管理対象証明書の更新](#)」を参照してください。

1.1.1.2. 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。以下のメトリクスが Red Hat Advanced Cluster Management に追加され、テレメトリに同梱されていますが、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。

- **visual_web_terminal_sessions_total** はハブクラスターで収集されます。
- **acm_managed_cluster_info** は、各マネージャークラスターで収集され、ハブクラスターに送信されます。

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。

1.1.1.3. 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 2636 mCPU の CPU および 11388 Mi のメモリが必要です。以下の表は、**observability-addons** が有効なマネージャークラスター 5 台の Pod 容量要求です。

表1.1 可観測性 Pod の容量要求

デプロイメントまたは StatefulSet	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計メモリー
Alertmanager	Alertmanager	4	200	3	12	600
	config-reloader	4	25	3	12	75
grafana	grafana	4	100	2	8	200
	grafana-dashboard-loader	4	50	2	8	100
observability-observatorium-observatorium-api	observatorium-api	20	128	2	40	256

デプロイメントまたは StatefulSet	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計 メモリー
observability-observatorium-thanos-compact	thanos-compact	100	512	1	100	512
observability-observatorium-thanos-query	thanos-query	300	1024	2	600	2048
observability-observatorium-thanos-query-frontend	thanos-query-frontend	100	256	2	200	512
observability-observatorium-thanos-receive-controller	thanos-receive-controller	4	32	1	4	32
observability-observatorium-thanos-receive-default	thanos-receive	300	512	3	900	1536
observability-observatorium-thanos-rule	thanos-rule	50	512	3	150	1536
	configmap-reloader	4	25	3	12	75
observability-observatorium-thanos-store-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	コンテナ名	CPU (mCPU)	メモリー (Mi)	レプリカ	Pod の合計 CPU	Pod の合計 メモリー
observability-observatorium-thanos-store-shard	thanos-store	100	1024	3	300	3072
observatorium-operator	observatorium-operator	100	100	1	100	100
rbac-query-proxy	rbac-query-proxy	20	100	2	40	200

1.1.2. 可観測性サービスで使用する永続ストア

Red Hat Advanced Cluster Management のインストール時に、以下の永続ボリュームを作成します。

表1.2 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ために十分な領域が必要です。ディスク上のデータは、次の再起動までに安全に削除でき、最初の試行でクラッシュループコンパクターの停止が解決されるはずです。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ためには、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。

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 ブロックは定期的に (2 時間)、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。
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ダウンサンリングの詳細は、「[可観測性サービスの有効化](#)」を参照してください。

1.2. 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。

必要なアクセス権限: クラスターの管理者または **open-cluster-management:cluster-manager-admin** ロール。

- [前提条件](#)
- [可観測性の有効化](#)
- [MultiClusterObservability CR の作成](#)
- [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。
- 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 Container Storage](#)

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。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. 以下のコンマ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
```

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
```

- a. 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。
 - Red Hat Advanced Cluster Management では、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

```

- Amazon S3 または S3 と互換性のある場合には、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_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。

- OpenShift Container Storage 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: s3
  config:
    bucket: YOUR_OCS_BUCKET
    endpoint: YOUR_OCS_ENDPOINT
    insecure: true
    access_key: YOUR_OCS_ACCESS_KEY
    secret_key: YOUR_OCS_SECRET_KEY
```

詳細は、「[OpenShift Container Storage のインストール](#)」を参照してください。

- 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。

```
ACCESS_KEY=$(oc -n <your-object-storage> get secret <object-storage-secret> -o yaml
| grep AccessKey | awk '{print $2}' | base64 --decode)

echo $ACCESS_KEY

SECRET_KEY=$(oc -n <your-object-storage> get secret <object-storage-secret> -o yaml
| grep SecretKey | awk '{print $2}' | base64 --decode)

echo $SECRET_KEY
```

1.2.2.1. MultiClusterObservability CR の作成

以下の手順を実行して **MultiClusterObservability** カスタムリソース (CR) を削除します。

- 以下の手順を実行して、マネージドクラスターの **MultiClusterObservability** カスタムリソース (mco CR) を作成します。
 - 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/v1beta1
kind: MultiClusterObservability
metadata:
  name: observability #Your customized name of MulticlusterObservability CR
spec:
  availabilityConfig: High # Available values are High or Basic
  enableDownSampling: false # The default value is false. This is not recommended as
  querying long-time ranges without non-downsampled data is not efficient and useful.
  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: true # EnableMetrics indicates the observability addon push metrics to
  hub server
  interval: 30 # Interval for the observability addon push metrics to hub server
  retentionResolution1h: 30d # How long to retain samples of 1 hour in bucket
  retentionResolution5m: 14d
  retentionResolutionRaw: 5d
  storageConfigObject: # Specifies the storage to be used by Observability
  metricObjectStorage:
    name: thanos-object-storage
    key: thanos.yaml
  statefulSetSize: 10Gi # The amount of storage applied to the Observability
  StatefulSets, i.e. Amazon S3 store, Rule, compact and receiver.
  statefulSetStorageClass: gp2

```

retentionResolution 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。デフォルトでは、downsampling は無効になっています。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、**statefulSetSize** を更新する必要がある場合があります。詳細は、「[Observability API](#)」を参照してください。

- b. インフラストラクチャ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マシンセットの作成](#)」を参照してください。

- c. 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。

2. 可観測性サービスが有効になっていることを確認するには、Grafana ダッシュボードを起動して、データが追加されていることを確認します。以下の手順を実行します。
 - a. Red Hat Advanced Cluster Management コンソールにログインします。
 - b. ナビゲーションメニューから **Observe environments > Overview** を選択します。

- c. 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て、マネージドクラスターからメトリクスを表示します。
- 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** のクラスターラベルを追加します。

1.2.3. 可観測性の無効化

可観測性 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。手順については、「[コマンドを使用した MultiClusterHub インスタンスの削除](#)」のステップ1を参照してください。

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

1.3. 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**」の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 ルールの設定](#)
- [カスタムメトリクスの追加](#)
- [データの表示および展開](#)
- [可観測性の無効化](#)

1.3.1. カスタムルールの作成

Prometheus [レコードルール](#) および [アラートルール](#) を可観測性リソースに追加して、可観測性のインストール時のカスタムルールを作成できます。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-observatorium-thanos-rule** Pod は自動的に再起動されます。

カスタムルールを作成するには、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. **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。

```
data:
  custom_rules.yaml: |
    groups:
      - name: cluster-health
        rules:
          - alert: ClusterCPUHealth-jb
            annotations:
              summary: Notify when CPU utilization on a cluster is greater than the defined
              utilization limit
            description: "The cluster has a high CPU usage: {{ $value }} core for {{
              $labels.cluster }} {{ $labels.clusterID }}."
            expr: |
              max(cluster:cpu_usage_cores:sum) by (clusterID, cluster, prometheus) > 0
            for: 5s
            labels:
              cluster: "{{ $labels.cluster }}"
              prometheus: "{{ $labels.prometheus }}"
              severity: critical
```

- **thanos-ruler-custom-rules** ConfigMap 内にカスタムの録画ルールを作成することもできます。

たとえば、Pod のコンテナメモリーキャッシュの合計を取得できるようにする記録ルールを作成することができます。YAML の内容は以下のようになります。

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

注記: これが最初の新規カスタムルールである場合には、すぐに作成されます。ConfigMap に変更を加える場合には、**kubectl rollout restart statefulset observability-observatorium-thanos-rule -n open-cluster-management-observability** のコマンドを使用して、可観測性 Pod を再起動する必要があります。

3. 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、以下の手順を実行します。

- a. Grafana ダッシュボードにアクセスし、**Explore** アイコンをクリックします。
- b. メトリクスの検索バーで、「ALERTS」と入力してクエリーを実行します。システムにある現在保留中または実行状態にある ALERTS がすべて表示されます。
- c. アラートが表示されない場合は、ルールを再度表示して、式が正しいかどうかを確認します。

カスタムルールが作成されました。

1.3.2. AlertManager ルールの設定

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。

1.3.3. 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には、以下の手順を実行します。

1. クラスターにログインします。
2. **mco observability** が有効化され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。以下のコマンドを実行します。

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

3. 以下の内容で、**observability-metrics-custom-allowlist.yaml** の新規ファイルを作成しま

す。**metrics_list.yaml** パラメーターにカスタムメトリクスの名前を追加します。たとえば、**node_memory_MemTotal_bytes** をメトリクスの一覧に追加します。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
```

4. 以下のコマンドを実行して、**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
```

5. Grafana ダッシュボードでメトリクスを表示して、カスタムメトリクスがマネージドクラスターから収集されていることを確認します。ハブクラスターから、**Grafana ダッシュボード** のリンクを選択します。
6. Grafana 検索バーから、表示するメトリクスを入力します。

カスタムメトリクスからデータを収集します。

1.3.4. データの表示および展開

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。コンソールから Grafana ダッシュボードを表示するには、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. ナビゲーションメニューから **Observe environments > Overview > Grafana link** を選択します。
また、**Clusters** ページから Grafana ダッシュボードにアクセスすることもできます。ナビゲーションメニューで **Automate infrastructure > Clusters > Grafana** を選択します。
3. Grafana ナビゲーションメニューから **Explore** アイコンを選択して、Prometheus メトリクスエクスプローラーにアクセスします。

1.3.5. 可観測性の無効化

Red Hat Advanced Cluster Management ハブクラスターでデータ収集を停止する可観測性を無効にできます。

1.3.5.1. 全クラスターの可観測性の無効化

全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。

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:
  availabilityConfig: High # Available values are High or Basic
  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
```

1.3.5.2. 単一クラスターの可観測性の無効化

以下のいずれかの手順を実行して、特定のマネージドクラスターの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、以下の手順を実行し、**observability: disabled** ラベルを追加します。
 1. Red Hat Advanced Cluster Management コンソールで、**Automate infrastructure > Clusters** を選択します。
 2. 可観測性に送信されるデータ収集を無効にするクラスターの名前を選択します。
 3. **Labels** を選択します。
 4. 以下のラベルを追加して、可観測性コレクションを無効にするラベルを作成します。

```
observability=disabled
```

5. **Add** を選択してラベルを追加します。
6. **Done** を選択してラベルの一覧を閉じます。

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。

1.4. 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。

1.4.1. Grafana 開発者インスタンスの設定

まず、[stolostron/multicluster-observability-operator/](#) リポジトリのクローンを作成し、tools フォルダにある スクリプトを実行できるようにします。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
ingress.extensions/grafana-dev created
persistentvolumeclaim/grafana-dev created
```

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://\\$ACM_URL/grafana-dev/](https://$ACM_URL/grafana-dev/) を選択して、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。

1.4.2. Grafana ダッシュボードの設計

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

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

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

ConfigMap で Grafana 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. **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。

- a. 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。
- b. 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。
- c. ダッシュボード JSON データをコピーし、**metadata** セクションに貼り付けます。

- d. **name** を、**\$your-dashboard-name** に置き換えます。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 **General** フォルダーにない場合は、この ConfigMap の **annotations** セクションに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。

1.4.3. 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
service "grafana-dev" deleted
ingress.extensions "grafana-dev" deleted
persistentvolumeclaim "grafana-dev" deleted
```