OpenShift Dedicated 4

ユーザー定義プロジェクトのモニタリング

モニタリングスタックの概要および設定

Last Updated: 2021-09-26

OpenShift Dedicated 4 ユーザー定義プロジェクトのモニタリング

モニタリングスタックの概要および設定

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

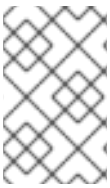
概要

OpenShift Dedicated クラスターの監視

目次

第1章 モニタリングスタックについて

OpenShift Dedicated では、Red Hat Site Reliability Engineer (SRE) プラットフォームメトリクスから切り離して独自のプロジェクトをモニターできます。追加のモニタリングソリューションなしに独自のプロジェクトをモニターできます。



注記

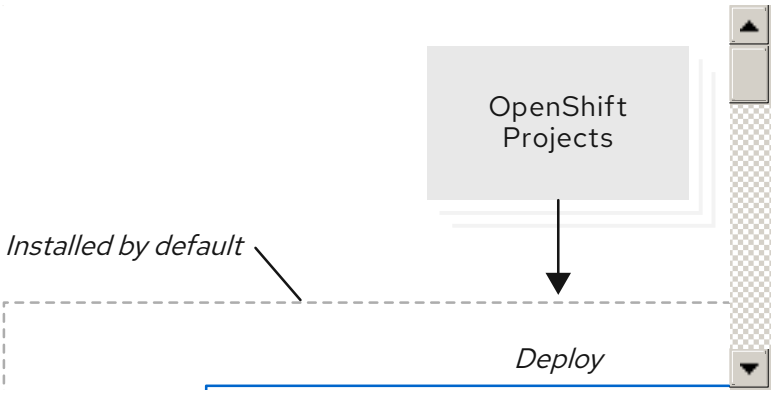
本書の手順に従って、ユーザー定義プロジェクトをモニターするためにサポートされる Prometheus インスタンスを設定します。カスタム Prometheus インスタンスは OpenShift Dedicated ではサポートされません。

1.1. モニタリングスタックについて

OpenShift Dedicated モニタリングスタックは、Prometheus オープンソースプロジェクトおよびその幅広いエコシステムをベースとしています。モニタリングスタックには、以下のコンポーネントが含まれます。

- **デフォルトのプラットフォームモニタリングコンポーネント**。プラットフォームモニタリングコンポーネントのセットは、OpenShift Dedicated のインストール時にデフォルトで **openshift-monitoring** プロジェクトにインストールされます。これにより、OpenShift Dedicated のコアのモニタリング機能が提供されます。デフォルトのモニタリングスタックは、クラスターのリモートのヘルスモニタリングも有効にします。CPU やメモリーなどの重要なメトリクスは、すべての namespace のすべてのワークロードから収集され、利用可能になります。
  これらのコンポーネントは、以下の図の **Installed by default** (デフォルトのインストール) セクションで説明されています。

- **ユーザー定義のプロジェクトをモニターするためのコンポーネント**。この機能はデフォルトで有効にされており、ユーザー定義プロジェクトのモニタリングを提供します。これらのコンポーネントは、以下の図の **User** (ユーザー) セクションで説明されています。



1.1.1. ユーザー定義プロジェクトをモニターするためのコンポーネント

OpenShift Dedicated には、ユーザー定義プロジェクトでサービスおよび Pod をモニターできるモニタリングスタックのオプションの拡張機能が含まれています。この機能には、以下のコンポーネントが含まれます。

表1.1 ユーザー定義プロジェクトをモニターするためのコンポーネント

コンポーネント	説明
Prometheus Operator	**openshift-user-workload-monitoring** プロジェクトの Prometheus Operator は、同じプロジェクトで Prometheus および Thanos Ruler インスタンスを作成し、設定し、管理します。
Prometheus	Prometheus は、ユーザー定義のプロジェクト用にモニタリング機能が提供されるモニタリングシステムです。Prometheus は処理のためにアラートを Alertmanager に送信します。ただし、アラートのルーティングは現在サポートされていません。
Thanos Ruler	Thanos Ruler は、別のプロセスとしてデプロイされる Prometheus のルール評価エンジンです。OpenShift Dedicated 4 では、Thanos Ruler はユーザー定義プロジェクトのモニタリングについてのルールおよびアラート評価を提供します。

これらのすべてのコンポーネントはスタックによってモニターされ、OpenShift Dedicated の更新時に自動的に更新されます。

1.1.2. ユーザー定義プロジェクトのターゲットのモニタリング

モニタリングは、OpenShift Dedicated のユーザー定義プロジェクトについてデフォルトで有効にされます。以下をモニターできます。

- ユーザー定義プロジェクトのサービスエンドポイント経由で提供されるメトリクス。
- ユーザー定義プロジェクトで実行される Pod。

1.2. 関連情報

- ユーザー定義プロジェクトのモニタリングのアクセス
- デフォルトのモニタリングコンポーネント
- デフォルトのモニタリングターゲット

1.3. 次のステップ

- モニタリングスタックの設定

第2章 モニタリングスタックの設定

本書では、ユーザー定義プロジェクトのモニタリングでサポートされる内容について説明します。また、モニタリングスタックの設定方法を示し、いくつかの一般的な設定シナリオを示します。

2.1. モニタリングのメンテナンスおよびサポート

OpenShift Dedicated Monitoring の設定のサポートされる方法として、本書で説明されているオプションを使用できます。**サポートされていない他の設定は使用しないでください。**

重要

別の Prometheus インスタンスのインストールは、Red Hat Site Reliability Engineers (SRE) ではサポートされていません。

設定のパラダイムが Prometheus リリース間で変更される可能性があり、このような変更には、設定のすべての可能性が制御されている場合のみ適切に対応できます。本セクションで説明されている設定以外の設定を使用する場合、**cluster-monitoring-operator** が差分を調整するため、変更内容は失われます。Operator はデフォルトで定義された状態へすべてをリセットします。

2.1.1. ユーザー定義プロジェクトをモニターするためのサポートについての考慮事項

以下の変更は明示的にサポートされていません。

- カスタム Prometheus インスタンスの OpenShift Dedicated へのインストール

2.2. モニタリングスタックの設定

OpenShift Dedicated では、**user-workload-monitoring-config ConfigMap** オブジェクトを使用してユーザー定義プロジェクトのワークロードをモニターするスタックを設定できます。設定マップはクラスターモニタリング Operator (CMO) を設定し、その後にスタックのコンポーネントが設定されます。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

      ```
      $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
      ```

   b. 設定を、**data.config.yaml** の下に値とキーのペア **<component_name>: <component_configuration>** として追加します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>:
      <configuration_for_the_component>
```

<component> および **<configuration_for_the_component>** を随時置き換えます。

以下の **ConfigMap** オブジェクトの例は、Prometheus のデータ保持期間および最小コンテナーリソース要求を設定します。これは、ユーザー定義のプロジェクトのみをモニターする Prometheus インスタンスに関連します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus: 1
      retention: 24h 2
      resources:
        requests:
          cpu: 200m 3
          memory: 2Gi 4
```

1. Prometheus コンポーネントを定義し、後続の行はその設定を定義します。
2. ユーザー定義プロジェクトをモニターする Prometheus インスタンスについて 24 時間のデータ保持期間を設定します。
3. Prometheus コンテナーの 200 ミリコアの最小リソース要求を定義します。
4. Prometheus コンテナーのメモリーの 2 GiB の最小 Pod リソース要求を定義します。

2. ファイルを保存して、変更を **ConfigMap** オブジェクトに適用します。新規設定の影響を受けた Pod は自動的に再起動されます。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

2.3. 設定可能なモニタリングコンポーネント

以下の表は、設定可能なモニタリングコンポーネントと、**user-workload-monitoring-config ConfigMap** オブジェクトでコンポーネントを指定するために使用されるキーを示しています。

表2.1 設定可能なモニタリングコンポーネント

コンポーネント	user-workload-monitoring-config 設定マップキー
Prometheus Operator	**prometheusOperator**
Prometheus	**prometheus**
Thanos Ruler	**thanosRuler**

2.4. モニタリングコンポーネントの異なるノードへの移動

ユーザー定義プロジェクトのワークロードを監視する任意のコンポーネントを特定のワーカーノードに移動できます。コンポーネントをマスターノードまたはインフラストラクチャーノードに移動することは許可されません。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ユーザー定義プロジェクトをモニターするコンポーネントを移動するには、**ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. コンポーネントの **nodeSelector** 制約を **data.config.yaml** に指定します。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       <component>:
         nodeSelector:
   ```

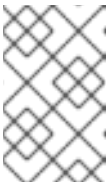
```
    <node_key>: <node_value>
    <node_key>: <node_value>
    <...>
```

<component> を適宜置き換え、**<node_key>: <node_value>** を、宛先ノードを指定するキーと値のペアのマップに置き換えます。通常は、単一のキーと値のペアのみが使用されます。

コンポーネントは、指定されたキーと値のペアのそれぞれをラベルとして持つノードでのみ実行できます。ノードには追加のラベルを持たせることもできます。

重要

モニタリングコンポーネントの多くは、高可用性を維持するために、クラスターの異なるノード間で複数の Pod を使用してデプロイされます。モニタリングコンポーネントをラベル付きノードに移動する際には、コンポーネントの耐障害性を維持するために十分な数の一致するノードが利用可能であることを確認します。1 つのラベルのみが指定されている場合、複数の別々のノードにコンポーネントに関連するすべての Pod を分散するために、十分な数のノードにそのラベルが含まれていることを確認します。または、複数のラベルを指定することもできます。その場合、それぞれのラベルを個々のノードに関連付けます。



注記

nodeSelector の制約を設定した後にモニタリングコンポーネントが **Pending** 状態のままである場合、Pod ログでテイントおよび容認に関連するエラーの有無を確認します。

たとえば、ユーザー定義プロジェクトのモニタリングコンポーネントを **nodename: worker1**、**nodename: worker2**、および **nodename: worker2** のラベルが付けられた特定のワーカーノードに移行するには、以下を使用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: worker1
    prometheus:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    thanosRuler:
      nodeSelector:
        nodename: worker1
        nodename: worker2
```

2. 変更を適用するためにファイルを保存します。新しい設定の影響を受けるコンポーネントは新しいノードに自動的に移動します。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

関連情報

- ノードでラベルを更新する方法について
- ノードセレクターの使用による特定ノードへの Pod の配置
- **nodeSelector** 制約についての詳細は、Kubernetes ドキュメント を参照してください。

2.5. ユーザー定義プロジェクトをモニターするコンポーネントへの容認の割り当て

ユーザー定義プロジェクトをモニターするコンポーネントに容認を割り当て、それらをテイントされたワーカーノードに移行できるようにします。スケジューリングは、マスターまたはインフラストラクチャーノードでは許可されません。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを **openshift-user-workload-monitoring** namespace に作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

   b. コンポーネントの **tolerations** を指定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
```

```
<component>:
  tolerations:
    <toleration_specification>
```

<component> および **<toleration_specification>** を随時置き換えます。

たとえば、**oc adm taint nodes node1 key1=value1:NoSchedule** は、テイントをキーが **key1** で、値が **value1** の **node1** に追加します。これにより、容認がテイントに設定されていない限り、モニタリングコンポーネントが **node1** に Pod をデプロイするのを防ぎます。以下の例では、サンプルのテイントを容認するように **thanosRuler** コンポーネントを設定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      tolerations:
      - key: "key1"
        operator: "Equal"
        value: "value1"
        effect: "NoSchedule"
```

2. 変更を適用するためにファイルを保存します。新しいコンポーネントの配置設定が自動的に適用されます。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

関連情報

- テイントおよび容認 (Toleration) については、OpenShift Container Platform ドキュメント を参照してください。
- テイントおよび容認 (Toleration) については、Kubernetes ドキュメントを参照してください。

2.6. 永続ストレージの設定

クラスターモニタリングを永続ストレージと共に実行すると、メトリクスは永続ボリューム (PV) に保存され、Pod の再起動または再作成後も維持されます。これは、メトリクスデータをデータ損失から保護する必要がある場合に適しています。実稼働環境では、永続ストレージを設定することを強く推奨します。IO デマンドが高いため、ローカルストレージを使用することが有利になります。

重要

「設定可能な推奨のストレージ技術」を参照してください。

2.6.1. 永続ストレージの前提条件

- ストレージのブロックタイプを使用します。

2.6.2. ローカル永続ボリューム要求 (PVC) の設定

モニタリングコンポーネントが永続ボリューム (PV) を使用できるようにするには、永続ボリューム要求 (PVC) を設定する必要があります。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ユーザー定義プロジェクトをモニターするコンポーネントの PVC を設定するには、**ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. コンポーネントの PVC 設定を **data.config.yaml** の下に追加します。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       <component>:
         volumeClaimTemplate:
           spec:
             storageClassName: <storage_class>
             resources:
               requests:
                 storage: <amount_of_storage>
   ```

   volumeClaimTemplate の指定方法については、PersistentVolumeClaims についての Kubernetes ドキュメントを参照してください。

   以下の例では、ユーザー定義プロジェクトをモニターする Prometheus インスタンスのローカル永続ストレージを要求する PVC を設定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

上記の例では、ローカルストレージ Operator によって作成されるストレージクラスは **local-storage** と呼ばれます。

以下の例では、Thanos Ruler のローカル永続ストレージを要求する PVC を設定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

2. 変更を適用するためにファイルを保存します。新規設定の影響を受けた Pod は自動的に再起動され、新規ストレージ設定が適用されます。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

2.6.3. Prometheus メトリクスデータの保持期間の変更

デフォルトで、OpenShift Dedicated モニタリングスタックは、Prometheus データの保持期間を 15 日間に設定します。データを削除するタイミングを変更するために、ユーザー定義のプロジェクトをモニターする Prometheus インスタンスの保持時間を変更できます。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ユーザー定義プロジェクトをモニターする Prometheus インスタンスの保持時間を変更するには、**ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. 保持期間の設定を **data.config.yaml** に追加します。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       prometheus:
         retention: <time_specification>
   ```

   <time_specification> を、**ms** (ミリ秒)、**s** (秒)、**m** (分)、**h** (時間)、**d** (日)、**w** (週)、または **y** (年) が直後に続く数字に置き換えます。

   以下の例では、ユーザー定義プロジェクトをモニターする Prometheus インスタンスの保持期間を 24 時間に設定します。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       prometheus:
         retention: 24h
   ```

2. 変更を適用するためにファイルを保存します。新規設定の影響を受けた Pod は自動的に再起動されます。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

関連情報

- 永続ストレージについて
- Optimizing storage

2.7. ユーザー定義プロジェクトでのバインドされていないメトリクス属性の影響の制御

開発者は、キーと値のペアの形式でメトリクスの属性を定義するためにラベルを作成できます。使用できる可能性のあるキーと値のペアの数は、属性について使用できる可能性のある値の数に対応します。数が無制限の値を持つ属性は、バインドされていない属性と呼ばれます。たとえば、**customer_id** 属性は、使用できる値が無限にあるため、バインドされていない属性になります。

割り当てられるキーと値のペアにはすべて、一意の時系列があります。ラベルに多数のバインドされていない値を使用すると、作成される時系列の数が指数関数的に増加する可能性があります。これは Prometheus のパフォーマンスに影響する可能性があり、多くのディスク領域を消費する可能性があります。

dedicated-admin は、以下の手段を使用して、ユーザー定義プロジェクトでのバインドされていないメトリクス属性の影響を制御できます。

- ユーザー定義プロジェクトで、ターゲット収集ごとに **受け入れ可能なサンプル数を制限** します。

注記

収集サンプルを制限すると、多くのバインドされていない属性をラベルに追加して問題が発生するのを防ぐことができます。さらに開発者は、メトリクスに定義するバインドされていない 属性の数を制限することにより、根本的な原因を防ぐことができます。使用可能な値の制限されたセットにバインドされる属性を使用すると、可能なキーと値のペアの組み合わせの数が減ります。

2.7.1. ユーザー定義プロジェクトの収集サンプル制限の設定

ユーザー定義プロジェクトで、ターゲット収集ごとに受け入れ可能なサンプル数を制限できます。

警告

サンプル制限を設定すると、制限に達した後にそのターゲット収集についての追加のサンプルデータは取り込まれません。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

2. **enforcedSampleLimit** 設定を **data.config.yaml** に追加し、ユーザー定義プロジェクトのターゲットの収集ごとに受け入れ可能なサンプルの数を制限できます。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      enforcedSampleLimit: 50000 1
```

1 このパラメーターが指定されている場合は、値が必要です。この **enforcedSampleLimit** の例では、ユーザー定義プロジェクトのターゲット収集ごとに受け入れ可能なサンプル数を 50,000 に制限します。

3. 変更を適用するためにファイルを保存します。制限は自動的に適用されます。

警告

変更が **user-workload-monitoring-config ConfigMap** オブジェクトに保存されると、**openshift-user-workload-monitoring** プロジェクトの Pod および他のリソースは再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

関連情報

- 最高数の収集サンプルを持つメトリクスをクエリーする手順: Prometheus が大量のディスク領域を消費している理由の特定

2.8. モニタリングコンポーネントのログレベルの設定

Prometheus Operator、Prometheus、および Thanos Ruler のログレベルを設定できます。

以下のログレベルは、**user-workload-monitoring-config ConfigMap** オブジェクトのそれらのコンポーネントのそれぞれに適用できます。

- **debug**デバッグ、情報、警告、およびエラーメッセージをログに記録します。
- **info**情報、警告およびエラーメッセージをログに記録します。
- **warn**警告およびエラーメッセージのみをログに記録します。
- **error**エラーメッセージのみをログに記録します。

デフォルトのログレベルは **info** です。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **ConfigMap** オブジェクトを編集します。
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

   b. コンポーネントの **logLevel: <log_level>** を **data.config.yaml** の下に追加します。

```
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: 1
      logLevel: <log_level> 2
```

1. ログレベルを適用するモニタリングコンポーネント。
2. コンポーネントに適用するログレベル。

2. 変更を適用するためにファイルを保存します。ログレベルの変更を適用する際に、コンポーネントの Pod は自動的に再起動します。

警告

変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

3. 関連するプロジェクトでデプロイメントまたは Pod 設定を確認し、ログレベルが適用されていることを確認します。以下の例では、**openshift-user-workload-monitoring** プロジェクトの **prometheus-operator** デプロイメントでログレベルを確認します。

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |  grep "log-level"
```

出力例

```
        - --log-level=debug
```

4. コンポーネントの Pod が実行中であることを確認します。以下の例は、**openshift-user-workload-monitoring** プロジェクトの Pod のステータスを一覧表示します。

```
$ oc -n openshift-user-workload-monitoring get pods
```

注記

認識されない **loglevel** 値が **ConfigMap** オブジェクトに含まれる場合、コンポーネントの Pod が正常に再起動されない可能性があります。

2.9. 次のステップ

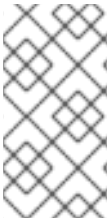
- ユーザー定義プロジェクトのモニタリングのアクセス

第3章 ユーザー定義プロジェクトのモニタリングのアクセス

デフォルトで、ユーザー定義プロジェクトおよびプラットフォームモニタリングの一元的なモニタリングが有効にされます。追加のモニタリングソリューションなしに、OpenShift Dedicated で独自のプロジェクトをモニターできます。

ユーザー定義プロジェクトのモニタリングを無効にすることはできません。

dedicated-admin ユーザーには、ユーザー定義プロジェクトのモニタリングを設定し、アクセスするためのデフォルトのパーミッションがあります。



注記

カスタム Prometheus インスタンスおよび Operator Lifecycle Manager (OLM) でインストールされる Prometheus Operator では、ユーザー定義のプロジェクトモニタリングが有効である場合にこれに関する問題が生じる可能性があります。カスタム Prometheus インスタンスはサポートされません。

3.1. 次のステップ

- メトリクスの管理

第4章 メトリクスの管理

本書では、OpenShift Dedicated メトリクスの収集、クエリー、および可視化方法を説明します。

4.1. メトリクスについて

OpenShift Dedicated では、クラスターコンポーネントはサービスエンドポイントで公開されるメトリクスを収集することによりモニターされます。ユーザー定義プロジェクトのメトリクスのコレクションを設定することもできます。メトリクスを使用すると、クラスターコンポーネントおよび独自のワークロードの実行方法をモニターできます。

Prometheus クライアントライブラリーをアプリケーションレベルで使用することで、独自のワークロードに指定するメトリクスを定義できます。

OpenShift Dedicated では、メトリクスは **/metrics** の正規名の下に HTTP サービスエンドポイント経由で公開されます。**curl** クエリーを **http://<endpoint>/metrics** に対して実行して、サービスの利用可能なすべてのメトリクスを一覧表示できます。たとえば、**prometheus-example-app** サンプルアプリケーションへのルートを公開し、以下のコマンドを実行して利用可能なすべてのメトリクスを表示できます。

```
$ curl http://<example_app_endpoint>/metrics
```

出力例

```
# HELP http_requests_total Count of all HTTP requests
# TYPE http_requests_total counter
http_requests_total{code="200",method="get"} 4
http_requests_total{code="404",method="get"} 2
# HELP version Version information about this binary
# TYPE version gauge
version{version="v0.1.0"} 1
```

関連情報

- Prometheus クライアントライブラリーについての詳細は、Prometheus ドキュメントを参照してください。

4.2. ユーザー定義プロジェクトのメトリクスコレクションの設定

ServiceMonitor リソースを作成して、ユーザー定義プロジェクトのサービスエンドポイントからメトリクスを収集できます。これは、アプリケーションが Prometheus クライアントライブラリーを使用してメトリクスを **/metrics** の正規名に公開していることを前提としています。

このセクションでは、ユーザー定義のプロジェクトでサンプルサービスをデプロイし、次にサービスのモニター方法を定義する **ServiceMonitor** リソースを作成する方法を説明します。

4.2.1. サンプルサービスのデプロイ

ユーザー定義のプロジェクトでサービスのモニタリングをテストするには、サンプルサービスをデプロイすることができます。

手順

1. サービス設定の YAML ファイルを作成します。この例では、**prometheus-example-app.yaml** という名前です。

2. 以下のデプロイメントおよびサービス設定の詳細をファイルに追加します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-example-app
  name: prometheus-example-app
  namespace: ns1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-example-app
  template:
    metadata:
      labels:
        app: prometheus-example-app
    spec:
      containers:
      - image: quay.io/brancz/prometheus-example-app:v0.2.0
        imagePullPolicy: IfNotPresent
        name: prometheus-example-app
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: prometheus-example-app
  name: prometheus-example-app
  namespace: ns1
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-example-app
  type: ClusterIP
```

この設定は、**prometheus-example-app** という名前のサービスをユーザー定義の **ns1** プロジェクトにデプロイします。このサービスは、カスタム **version** メトリクスを公開します。

3. 設定をクラスターに適用します。

```
$ oc apply -f prometheus-example-app.yaml
```

サービスをデプロイするには多少時間がかかります。

4. Pod が実行中であることを確認できます。

```
$ oc -n ns1 get pod
```

出力例

```
NAME                                  READY     STATUS    RESTARTS   AGE
prometheus-example-app-7857545cb7-sbgwq   1/1       Running   0          81m
```

4.2.2. サービスのモニター方法の指定

サービスが公開するメトリクスを使用するには、OpenShift Dedicated モニタリングを、**/metrics** エンドポイントからメトリクスを収集できるように設定する必要があります。これは、サービスのモニタリング方法を指定する **ServiceMonitor** カスタムリソース定義、または Pod のモニタリング方法を指定する **PodMonitor** CRD を使用して実行できます。前者の場合は **Service** オブジェクトが必要ですが、後者の場合は不要です。これにより、Prometheus は Pod によって公開されるメトリクスエンドポイントからメトリクスを直接収集することができます。

注記

OpenShift Dedicated では、**ServiceMonitor** リソースの **tlsConfig** プロパティーを使用し、エンドポイントからメトリクスを収集する際に使用する TLS 設定を指定できます。**tlsConfig** プロパティーは **PodMonitor** リソースではまだ利用できません。メトリクスの収集時に TLS 設定を使用する必要がある場合は、**ServiceMonitor** リソースを使用する必要があります。

この手順では、ユーザー定義プロジェクトでサービスの **ServiceMonitor** リソースを作成する方法を説明します。

前提条件

- **dedicated-admin** ロールまたは **monitoring-edit** ロールを持つユーザーとしてクラスターにアクセスできる。
- この例では、**prometheus-example-app** サンプルサービスを **ns1** プロジェクトにデプロイしている。

手順

1. **ServiceMonitor** リソース設定の YAML ファイルを作成します。この例では、ファイルは **example-app-service-monitor.yaml** という名前です。
2. 以下の **ServiceMonitor** リソース設定の詳細を追加します。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus-example-monitor
  name: prometheus-example-monitor
  namespace: ns1
```

```
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-example-app
```

これは、**prometheus-example-app** サンプルサービスによって公開されるメトリクスを収集する **ServiceMonitor** リソースを定義します。これには **version** メトリクスが含まれます。

3. 設定をクラスターに適用します。

```
$ oc apply -f example-app-service-monitor.yaml
```

ServiceMonitor をデプロイするのに多少時間がかかります。

4. **ServiceMonitor** リソースが実行中であることを確認できます。

```
$ oc -n ns1 get servicemonitor
```

出力例

```
NAME                         AGE
prometheus-example-monitor   81m
```

関連情報

- **ServiceMonitor** および **PodMonitor** リソースについての詳細は、Prometheus Operator API のドキュメン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へのアクセス

4.3. メトリクスのクエリー

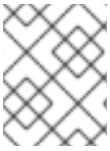
OpenShift モニタリングダッシュボードでは、Prometheus のクエリー言語 (PromQL) クエリーを実行し、プロットに可視化されるメトリクスを検査できます。この機能により、クラスターの状態と、モニターしているユーザー定義のプロジェクトに関する情報が提供されます。

dedicated-admin として、ユーザー定義プロジェクトに関するメトリクスに対して、一度に 1 つ以上の namespace をクエリーできます。

開発者として、メトリクスのクエリー時にプロジェクト名を指定する必要があります。選択したプロジェクトのメトリクスを表示するには、必要な権限が必要です。

4.3.1. 管理者としてのすべてのプロジェクトのメトリクスのクエリー

dedicated-admin またはすべてのプロジェクトの表示パーミッションを持つユーザーとして、メトリクス UI ですべてのデフォルト OpenShift Dedicated およびユーザー定義プロジェクトのメトリクスにアクセスできます。



注記

専任の管理者のみが、OpenShift Dedicated で提供されるサードパーティーの UI にアクセスできます。

前提条件

- **dedicated-admin** ロールまたはすべてのプロジェクトの表示パーミッションを持つユーザーとしてクラスターにアクセスできる。

手順

1. OpenShift Web コンソールの **Administrator** パースペクティブで、**Monitoring → Metrics** を選択します。
2. **Insert Metric at Cursor** を選択し、事前に定義されたクエリーの一覧を表示します。
3. カスタムクエリーを作成するには、Prometheus クエリー言語 (PromQL) のクエリーを **Expression** フィールドに追加します。
4. 複数のクエリーを追加するには、**Add Query** を選択します。
5. クエリーを削除するには、クエリーの横にある  を選択してから **Delete query** を選択します。
6. クエリーの実行を無効にするには、クエリーの横にある  を選択してから **Disable query** を選択します。
7. **Run Queries** を選択し、作成したクエリーを実行します。クエリーからのメトリクスはプロットで可視化されます。クエリーが無効な場合、UI にはエラーメッセージが表示されます。

注記

大量のデータで動作するクエリーは、時系列グラフの描画時にタイムアウトするか、またはブラウザーをオーバーロードする可能性があります。これを回避するには、**Hide graph** を選択し、メトリクステーブルのみを使用してクエリーを調整します。次に、使用できるクエリーを確認した後に、グラフを描画できるようにプロットを有効にします。

8. オプション: ページ URL には、実行したクエリーが含まれます。このクエリーのセットを再度使用できるようにするには、この URL を保存します。

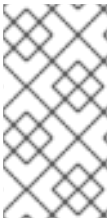
関連情報

- PromQL クエリーの作成に関する詳細は、Prometheus クエリーについてのドキュメントを参照してください。

4.3.2. 開発者としてのユーザー定義プロジェクトのメトリクスのクエリー

ユーザー定義のプロジェクトのメトリクスには、開発者またはプロジェクトの表示パーミッションを持つユーザーとしてアクセスできます。

Developer パースペクティブには、選択したプロジェクトの事前に定義された CPU、メモリー、帯域幅、およびネットワークパケットのクエリーが含まれます。また、プロジェクトの CPU、メモリー、帯域幅、ネットワークパケットおよびアプリケーションメトリクスについてカスタム Prometheus Query Language (PromQL) クエリーを実行することもできます。



注記

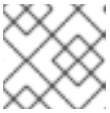
開発者は **Developer** パースペクティブのみを使用でき、**Administrator** パースペクティブは使用できません。開発者は、1 度に 1 つのプロジェクトのメトリクスのみをクエリーできます。開発者は OpenShift Dedicated モニタリングで提供されるサードパーティーの UI にアクセスできません。

前提条件

- 開発者として、またはメトリクスで表示しているプロジェクトの表示パーミッションを持つユーザーとしてクラスターへのアクセスがある。
- ユーザー定義プロジェクトのモニタリングを有効にしている。
- ユーザー定義プロジェクトにサービスをデプロイしている。
- サービスのモニター方法を定義するために、サービスの **ServiceMonitor** カスタムリソース定義 (CRD) を作成している。

手順

1. OpenShift Dedicate Web コンソールの **Developer** パースペクティブから、**Monitoring → Metrics** を選択します。
2. **Project:** 一覧でメトリクスで表示するプロジェクトを選択します。
3. **Select Query** 一覧からクエリーを選択するか、または **Show PromQL** を選択してカスタム PromQL クエリーを実行します。



注記

Developer パースペクティブでは、1 度に 1 つのクエリーのみを実行できます。

関連情報

- PromQL クエリーの作成に関する詳細は、Prometheus クエリーについてのドキュメントを参照してください。
- 開発者または特権のあるユーザーとしてクラスター以外のメトリクスにアクセスする方法についての詳細は、「開発者としてのユーザー定義プロジェクトのメトリクスのクエリー」を参照してください。

4.3.3. 視覚化されたメトリクスの使用

クエリーの実行後に、メトリクスが対話式プロットに表示されます。プロットの X 軸は時間を表し、Y 軸はメトリクスの値を表します。各メトリクスは、グラフ上の色付きの線で表示されます。プロットを対話的に操作し、メトリクスを参照できます。

手順

Administrator パースペクティブ:

1. 最初に、有効なすべてのクエリーからのすべてのメトリクスがプロットに表示されます。表示されるメトリクスを選択できます。

注記

デフォルトでは、クエリーテーブルに、すべてのメトリクスとその現在の値を一覧表示する拡張ビューが表示されます。˅ を選択すると、クエリーの拡張ビューを最小にすることができます。

   - クエリーからすべてのメトリクスを非表示にするには、クエリーの  をクリックし、**Hide all series** をクリックします。
   - 特定のメトリクスを非表示にするには、クエリーテーブルに移動し、メトリクス名の横にある色の付いた四角をクリックします。
2. プロットをズームアップし、時間範囲を変更するには、以下のいずれかを行います。
   - プロットを水平にクリックし、ドラッグして、時間範囲を視覚的に選択します。
   - 左上隅のメニューを使用して、時間範囲を選択します。
3. 時間の範囲をリセットするには、**Reset Zoom** を選択します。
4. 特定の時点のすべてのクエリーの出力を表示するには、その時点のプロットの上にカーソルを合わせます。クエリーの出力はポップアップに表示されます。
5. プロットを非表示にするには、**Hide Graph** を選択します。

Developer パースペクティブ:

1. プロットをズームアップし、時間範囲を変更するには、以下のいずれかを行います。
   - プロットを水平にクリックし、ドラッグして、時間範囲を視覚的に選択します。
   - 左上隅のメニューを使用して、時間範囲を選択します。
2. 時間の範囲をリセットするには、**Reset Zoom** を選択します。
3. 特定の時点のすべてのクエリーの出力を表示するには、その時点のプロットの上にカーソルを合わせます。クエリーの出力はポップアップに表示されます。

関連情報

- PromQL インターフェースの使用について「メトリクスのクエリー」セクションを参照してください。
- モニタリング関連の問題のトラブルシューティング

4.4. 次のステップ

- アラート

第5章 アラート

ユーザー定義プロジェクトでのワークロードのモニタリングについてのアラートは、現時点ではこの製品ではサポートされていません。

5.1. 次のステップ

- モニタリングダッシュボードの確認

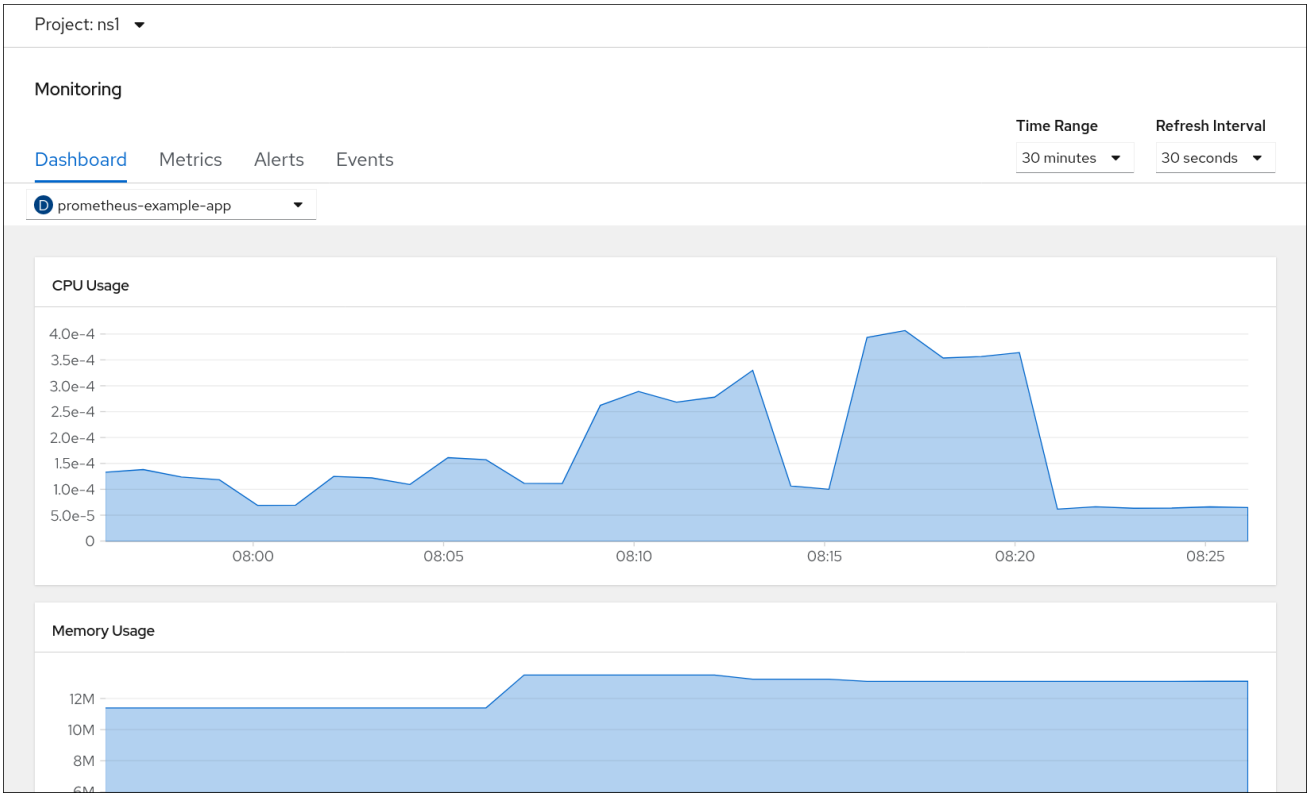
第6章 モニタリングダッシュボードの確認

OpenShift Dedicated は、ユーザー定義プロジェクトの状態を理解するのに役立つモニタリングダッシュボードを提供します。

Developer パースペクティブでは、選択されたプロジェクトの以下の統計を提供するダッシュボードにアクセスできます。

- CPU usage (CPU の使用率)
- メモリー使用量
- 帯域幅に関する情報
- パケットレート情報

図6.1 Developer パースペクティブのダッシュボードの例



注記

Developer パースペクティブでは、1度に1つのプロジェクトのみのダッシュボードを表示できます。

6.1. 開発者としてのモニタリングダッシュボードの確認

Developer パースペクティブでは、選択されたプロジェクトに関連するダッシュボードを表示できます。ダッシュボード情報を表示するには、プロジェクトをモニターするためのアクセスが必要になります。

前提条件

- **dedicated-admin** として、またはダッシュボードで表示しているプロジェクトの表示パーミッションを持つユーザーとしてクラスターにアクセスできる必要があります。

手順

1. OpenShift Dedicated Web コンソールの **Developer** パースペクティブで、**Monitoring → Dashboard** に移動します。
2. **Project:** 一覧でプロジェクトを選択します。
3. **All Workloads** 一覧でワークロードを選択します。
4. 必要に応じて、**Time Range** 一覧でグラフの時間範囲を選択します。
5. オプション:、**Refresh Interval** を選択します。
6. 特定の項目についての詳細情報を表示するには、ダッシュボードの各グラフにカーソルを合わせます。

6.2. 次のステップ

- モニタリング関連の問題のトラブルシューティング

第7章 モニタリング関連の問題のトラブルシューティング

本書では、ユーザー定義プロジェクトの一般的なモニタリング問題のトラブルシューティング方法を説明します。

7.1. ユーザー定義プロジェクトのメトリクスが利用できない理由の判別

ユーザー定義プロジェクトのモニタリング時にメトリクスが表示されない場合は、以下の手順を実行して問題のトラブルシューティングを実行します。

手順

1. メトリクス名に対してクエリーを実行し、プロジェクトが正しいことを確認します。

   a. OpenShift Container Platform Web コンソールの **Developer** パースペクティブから、**Monitoring → Metrics** を選択します。

   b. **Project:** 一覧でメトリクスで表示するプロジェクトを選択します。

   c. **Select Query** 一覧からクエリーを選択するか、または **Show PromQL** を選択してカスタム PromQL クエリーを実行します。
   Select Query ペインには、メトリクス名が表示されます。

   クエリーはプロジェクトごとに実行される必要があります。表示されるメトリクスは、選択したプロジェクトに関連するメトリクスです。

2. メトリックが必要な Pod がアクティブにメトリックを提供していることを確認します。以下の **oc exec** コマンドを Pod で実行し、 **podIP**、**port**、 および **/metrics** をターゲットにします。

```
$ oc exec <sample_pod> -n <sample_namespace> -- curl <target_pod_IP>:<port>/metrics
```

注記

curl がインストールされている Pod でコマンドを実行する必要があります。

以下の出力例は、有効なバージョンのメトリクスを含む結果を示しています。

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
# HELP version Version information about this binary-- --:--:-- --:--:--     0
# TYPE version gauge
version{version="v0.1.0"} 1
100   102  100   102    0     0  51000      0 --:--:-- --:--:-- --:--:-- 51000
```

無効な出力は、対応するアプリケーションに問題があることを示しています。

3. **PodMonitor** CRD を使用している場合は、 **PodMonitor** CRDがラベル一致を使用して適切な Pod を参照するよう設定されていることを確認します。詳細は、Prometheus Operator のドキュメントを参照してください。

4. **ServiceMonitor** CRD を使用し、Pod の **/metrics** エンドポイントがメトリクスデータを表示している場合は、以下の手順を実行して設定を確認します。

a. サービスが正しい **/metrics** エンドポイントを参照していることを確認します。この出力のサービス **labels** は、後続の手順でサービスが定義するサービスモニターの **labels** および **/metrics** エンドポイントと一致する必要があります。

```
$ oc get service
```

出力例

```
apiVersion: v1
kind: Service (1)
metadata:
  labels: (2)
    app: prometheus-example-app
  name: prometheus-example-app
  namespace: ns1
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-example-app
  type: ClusterIP
```

(1) これがサービス API であることを指定します。

(2) このサービスに使用されるラベルを指定します。

b. **serviceIP**、**port**、および **/metrics** エンドポイントをクエリーし、以前に Pod で実行した **curl** コマンドと同じメトリクスがあるかどうかを確認します。

i. 以下のコマンドを実行してサービス IP を見つけます。

```
$ oc get service -n <target_namespace>
```

ii. **/metrics** エンドポイントをクエリーします。

```
$ oc exec <sample_pod> -n <sample_namespace> -- curl <service_IP>:
<port>/metrics
```

以下の例では、有効なメトリクスが返されます。

出力例

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   102  100   102    0     0  51000      0 --:--:-- --:--:-- --:--:--   99k
# HELP version Version information about this binary
# TYPE version gauge
version{version="v0.1.0"} 1
```

c. ラベルのマッチングを使用して、**ServiceMonitor** オブジェクトが必要なサービスを参照す

るように設定されていることを確認します。これを実行するには、**oc get service** 出力の **Service** オブジェクトを **oc get servicemonitor** 出力の **ServiceMonitor** オブジェクトと比較します。ラベルは、表示されるメトリクスについて一致している必要があります。
たとえば、直前の手順の **Service** オブジェクトに **app: prometheus-example-app** ラベルがあり、**ServiceMonitor** オブジェクトに同じ **app: prometheus-example-app** 一致ラベルがある点に注意してください。

5. すべて有効で、メトリクスがまだ利用できない場合は、サポートチームにお問い合わせください。