OpenShift Container Platform 4.7

モニタリング

OpenShift Container Platform でのモニタリングスタックの設定および使用

Last Updated: 2021-10-10

OpenShift Container Platform 4.7 モニタリング

OpenShift Container Platform でのモニタリングスタックの設定および使用

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

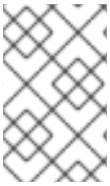
本書では、OpenShift Container Platform で Prometheus モニタリングスタックを設定し、使用する方法について説明します。

目次

第1章 モニタリングスタックについて

OpenShift Container Platform には、**コアプラットフォームコンポーネントのモニタリング** を提供する事前に設定され、事前にインストールされた自己更新型のモニタリングスタックが含まれます。OpenShift Container Platform は、追加設定が不要のモニタリングのベストプラクティスを提供します。クラスター管理者にクラスターの問題について即時に通知するアラートのセットがデフォルトで含まれます。OpenShift Container Platform Web コンソールのデフォルトのダッシュボードには、クラスターの状態をすぐに理解できるようにするクラスターのメトリクスの視覚的な表示が含まれます。

OpenShift Container Platform 4.7 のインストール後に、クラスター管理者はオプションで **ユーザー定義プロジェクトのモニタリング** を有効にできます。この機能を使用することで、クラスター管理者、開発者、および他のユーザーは、サービスと Pod を独自のプロジェクトでモニターする方法を指定できます。次に、OpenShift Container Platform Web コンソールでメトリクスのクエリー、ダッシュボードの確認、および独自のプロジェクトのアラートルールおよびサイレンスを管理できます。



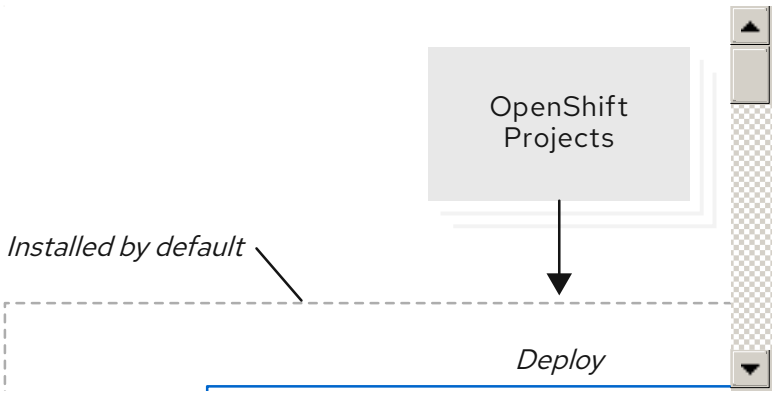
注記

クラスター管理者は、開発者およびその他のユーザーに、独自のプロジェクトをモニターするパーミッションを付与できます。事前に定義されたモニタリングロールのいずれかを割り当てると、特権が付与されます。

1.1. モニタリングスタックについて

OpenShift Container Platform 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 Container Platform 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、Kubernetes サービスを含む OpenShift Container Platform のコアコンポーネント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。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**。オプションでユーザー定義プロジェクトのモニタリングを有効にした後に、追加のモニタリングコンポーネントは **openshift-user-workload-monitoring** プロジェクトにインストールされます。これにより、ユーザー定義プロジェクトのモニタリング機能が提供されます。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. デフォルトのモニタリングコンポーネント

デフォルトで、OpenShift Container Platform 4.7 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。

表1.1 デフォルトのモニタリングスタックコンポーネント

コンポーネント	説明
クラスターモニタリング Operator	Cluster Monitoring Operator (CMO) は、モニタリングスタックの中心的なコンポーネントです。Prometheus インスタンスである Thanos Querier、Telemeter Client、およびメトリクスターゲットをデプロイおよび管理し、それらが最新の状態であることを確認します。CMO は Cluster Version Operator (CVO) によってデプロイされます。
Prometheus Operator	**openshift-monitoring** プロジェクトの Prometheus Operator(PO)は、プラットフォーム Prometheus インスタンスおよび Alertmanager インスタンスを作成し、設定し、管理します。また、Kubernetes ラベルのクエリーに基づいてモニタリングターゲットの設定を自動生成します。
Prometheus	Prometheus は、OpenShift Container Platform モニタリングスタックのベースとなるモニタリングシステムです。Prometheus は Time Series を使用するデータベースであり、メトリクスのルール評価エンジンです。Prometheus は処理のためにアラートを Alertmanager に送信します。
Prometheus アダプター	Prometheus アダプター (上記の図の PA) は、Prometheus で使用する Kubernetes ノードおよび Pod クエリーを変換します。変換されるリソースメトリクスには、CPU およびメモリーの使用率メトリクスが含まれます。Prometheus アダプターは、Horizontal Pod Autoscaling のクラスターリソースメトリクス API を公開します。Prometheus アダプターは **oc adm top nodes** および **oc adm top pods** コマンドでも使用されます。
Alertmanager	Alertmanager サービスは、Prometheus から送信されるアラートを処理します。また、Alertmanager は外部の通知システムにアラートを送信します。
kube-state-metrics エージェント	**kube-state-metrics** エクスポーターエージェント (上記の図の KSM) は、Kubernetes オブジェクトを Prometheus が使用できるメトリクスに変換します。
openshift-state-metrics エージェント	OpenShift Container Platform 固有のリソースのメトリクスを追加すると、**openshift-state-metrics** エクスポーター (上記の図の OSM) は **kube-state-metrics** に対して拡張します。

コンポーネント	説明
node-exporter エージェント	**node-exporter** エージェント (上記の図の NE) はクラスター内のすべてのノードについてのメトリクスを収集します。**node-exporter** エージェントはすべてのノードにデプロイされます。
Thanos Querier	Thanos Querier は、単一のマルチテナントインターフェースで、OpenShift Container Platform のコアメトリクスおよびユーザー定義プロジェクトのメトリクスを集約し、これらの重複を排除します。
Grafana	Grafana 解析プラットフォームは、メトリクスの分析および可視化のためのダッシュボードを提供します。モニタリングスタックおよびダッシュボードと共に提供される Grafana インスタンスは読み取り専用です。
Telemeter クライアント	Telemeter Client は、クラスターのリモートヘルスモニタリングを容易にするために、プラットフォーム Prometheus インスタンスから Red Hat にデータのサブセクションを送信します。

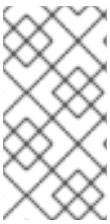
モニタリグスタックのすべてのコンポーネントはスタックによってモニターされ、OpenShift Container Platform の更新時に自動的に更新されます。

1.1.2. デフォルトのモニタリングターゲット

スタック自体のコンポーネントに加え、デフォルトのモニタリングスタックは以下をモニターします。

- CoreDNS
- Elasticsearch（ロギングがインストールされている場合）
- etcd
- Fluentd（ロギングがインストールされている場合）
- HAProxy
- イメージレジストリー
- Kubelets
- Kubernetes apiserver
- Kubernetes controller manager
- Kubernetes scheduler
- Metering（メータリングがインストールされている場合）

- OpenShift apiserver
- OpenShift コントロールマネージャー
- Operator Lifecycle Manager (OLM)



注記

各 OpenShift Container Platform コンポーネントはそれぞれのモニタリング設定を行います。OpenShift Container Platform コンポーネントのモニタリングに関する問題については、Bugzilla で一般的なモニタリングコンポーネントについてではなく、特定のコンポーネントに対してバグを報告してください。

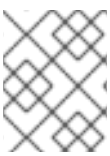
他の OpenShift Container Platform フレームワークのコンポーネントもメトリクスを公開する場合があります。詳細については、それぞれのドキュメントを参照してください。

1.1.3. 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 Container Platform 4.7 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1.2 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 (PO) 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。
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 Container Platform 4.7 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。



注記

上記の表のコンポーネントは、モニタリングがユーザー定義のプロジェクトに対して有効にされた後にデプロイされます。

モニタリグスタック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 Container Platform の更新時に自動的に更新されます。

1.1.4. 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

モニタリングがユーザー定義プロジェクトについて有効にされている場合には、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. 追加リソース

- リモートヘルスモニタリングについて
- ユーザーに対するユーザー定義のプロジェクトをモニターするパーミッションの付与

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

OpenShift Container Platform 4 インストールプログラムは、インストール前の少数の設定オプションのみを提供します。ほとんどの OpenShift Container Platform フレームワークコンポーネント (クラスターモニタリングスタックを含む) の設定はインストール後に行われます。

このセクションでは、サポートされている設定内容を説明し、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. 前提条件

- モニタリングスタックには、追加のリソース要件があります。コンピューティングリソースの推奨事項については、「Cluster Monitoring Operator のスケーリング」を参照し、十分なリソースがあることを確認してください。

2.2. モニタリングのメンテナンスおよびサポート

OpenShift Container Platform モニタリングの設定は、本書で説明されているオプションを使用して行う方法がサポートされている方法です。**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。**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。

2.2.1. モニタリングのサポートに関する考慮事項

以下の変更は明示的にサポートされていません。

- **追加の ServiceMonitor、 PodMonitor、および PrometheusRule オブジェクトの openshift-*、および kube-* プロジェクトでの作成。**
- **openshift-monitoring または openshift-user-workload-monitoring プロジェクトにデプロイされるリソースまたはオブジェクト変更**OpenShift Container Platform モニタリングスタックによって作成されるリソースは、後方互換性の保証がないために他のリソースで使用されることは意図されていません。

  注記

  Alertmanager 設定は、**openshift-monitoring** プロジェクトにシークレットリソースとしてデプロイされます。Alertmanager の追加ルートを設定するには、そのシークレットをデコードし、変更し、その後にエンコードする必要があります。この手順は、前述のステートメントに対してサポートされる例外です。

- **スタックのリソースの変更。**OpenShift Container Platform モニタリングスタックは、そのリソースが常に期待される状態にあることを確認します。これらが変更される場合、スタックはこれらをリセットします。
- **ユーザー定義ワークロードの openshift-*、および kube-* プロジェクトへのデプロイ。**これらのプロジェクトは Red Hat が提供するコンポーネント用に予約され、ユーザー定義のワークロードに使用することはできません。
- **モニタリングスタック Grafana インスタンスの変更。**
- **カスタム Prometheus インスタンスの OpenShift Container Platform へのインストール。**

- Prometheus Operator での **Probe** カスタムリソース定義 (CRD) による現象ベースのモニタリングの有効化。
- Prometheus Operator での **AlertmanagerConfig** CRD を使用した Alertmanager 設定の変更。

注記

メトリクス、記録ルールまたはアラートルールの後方互換性を保証されません。

2.2.2. Operator のモニタリングについてのサポートポリシー

モニタリング Operator により、OpenShift Container Platform モニタリングリソースの設定およびテスト通りに機能することを確認できます。Operator の Cluster Version Operator (CVO) コントロールがオーバーライドされる場合、Operator は設定の変更に対応せず、クラスターオブジェクトの意図される状態を調整したり、更新を受信したりしません。

Operator の CVO コントロールのオーバーライドはデバッグ時に役立ちますが、これはサポートされず、クラスター管理者は個々のコンポーネントの設定およびアップグレードを完全に制御することを前提としています。

Cluster Version Operator のオーバーライド

spec.overrides パラメーターを CVO の設定に追加すると、管理者はコンポーネントについての CVO の動作にオーバーライドの一覧を追加できます。コンポーネントについて **spec.overrides[].unmanaged** パラメーターを **true** に設定すると、クラスターのアップグレードがブロックされ、CVO のオーバーライドが設定された後に管理者にアラートが送信されます。

> Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

警告

CVO のオーバーライドを設定すると、クラスター全体がサポートされていない状態になり、モニタリングスタックをその意図された状態に調整されなくなります。これは Operator に組み込まれた信頼性の機能に影響を与え、更新が受信されなくなります。サポートを継続するには、オーバーライドを削除した後に、報告された問題を再現する必要があります。

2.3. モニタリングスタックの設定の準備

モニタリング設定マップを作成し、更新してモニタリングスタックを設定できます。

2.3.1. クラスターモニタリング設定マップの作成

OpenShift Container Platform のコアモニタリングコンポーネントを設定するには、**cluster-monitoring-config ConfigMap** オブジェクトを **openshift-monitoring** プロジェクトに作成する必要があります。

注記

変更を **cluster-monitoring-config ConfigMap** オブジェクトに保存すると、**openshift-monitoring** プロジェクトの Pod の一部またはすべてが再デプロイされる可能性があります。これらのコンポーネントが再デプロイするまで時間がかかる場合があります。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **cluster-monitoring-config ConfigMap** オブジェクトが存在するかどうかを確認します。

```
$ oc -n openshift-monitoring get configmap cluster-monitoring-config
```

2. **ConfigMap** オブジェクトが存在しない場合:

   a. 以下の YAML マニフェストを作成します。以下の例では、このファイルは **cluster-monitoring-config.yaml** という名前です。

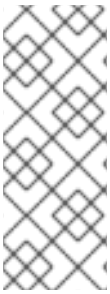
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
```

   b. 設定を適用して **ConfigMap** を作成します。

```
$ oc apply -f cluster-monitoring-config.yaml
```

2.3.2. ユーザー定義のワークロードモニタリング設定マップの作成

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設定するには、**user-workload-monitoring-config ConfigMap** オブジェクトを **openshift-user-workload-monitoring** プロジェクトに作成する必要があります。



注記

変更を **user-workload-monitoring-config ConfigMap** オブジェクトに保存すると、**openshift-user-workload-monitoring** プロジェクトの Pod の一部またはすべてが再デプロイされる可能性があります。これらのコンポーネントが再デプロイするまで時間がかかる場合があります。ユーザー定義プロジェクトのモニタリングを最初に有効にする前に設定マップを作成し、設定することができます。これにより、Pod を頻繁に再デプロイする必要がなくなります。

前提条件

- **cluster-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。

- OpenShift CLI (**oc**) がインストールされている。

手順

1. **user-workload-monitoring-config ConfigMap** オブジェクトが存在するかどうかを確認します。

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

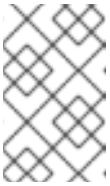
2. **user-workload-monitoring-config ConfigMap** オブジェクトが存在しない場合：

   a. 以下の YAML マニフェストを作成します。以下の例では、このファイルは **user-workload-monitoring-config.yaml** という名前です。

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

   b. 設定を適用して **ConfigMap** を作成します。

```
$ oc apply -f user-workload-monitoring-config.yaml
```



注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

追加リソース

- ユーザー定義プロジェクトのモニタリングの有効化

2.4. モニタリングスタックの設定

OpenShift Container Platform 4.7 では、 **cluster-monitoring-config** または **user-workload-monitoring-config ConfigMap** オブジェクトを使用してモニタリング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

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:

- **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
- **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。
   - **OpenShift Container Platform のコアモニタリングコンポーネントを設定するには**、以下を実行します。
     a. **openshift-monitoring** プロジェクトで **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-monitoring edit configmap cluster-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: cluster-monitoring-config
  namespace: openshift-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 の永続ボリューム要求 (PVC) を設定します。これは、OpenShift Container Platform のコアコンポーネン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: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s: 1
      volumeClaimTemplate:
        spec:
          storageClassName: fast
          volumeMode: Filesystem
          resources:
            requests:
              storage: 40Gi
```

```
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。
```

- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定するには**、以下を実行します。

  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。

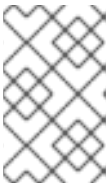
  3 Prometheus コンテナーの 200 ミリコアの最小リソース要求を定義します。

  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。

注記

Prometheus 設定マップコンポーネントは、 **cluster-monitoring-config ConfigMap** オブジェクトで **prometheusK8s** と呼ばれ、 **user-workload-monitoring-config ConfigMap** オブジェクトで **prometheus** と呼ばれます。

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。



注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化

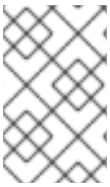
2.5. 設定可能なモニタリングコンポーネント

以下の表は、設定可能なモニタリングコンポーネントと、**cluster-monitoring-config** および **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	cluster-monitoring-config 設定マップキー	user-workload-monitoring-config 設定マップキー
Prometheus Operator	**prometheusOperator**	**prometheusOperator**
Prometheus	**prometheusK8s**	**prometheus**
Alertmanager	**alertmanagerMain**	
kube-state-metrics	**kubeStateMetrics**	

コンポーネント	cluster-monitoring-config 設定マップキー	user-workload-monitoring-config 設定マップキー
openshift-state-metrics	**openshiftStateMetrics**	
Grafana	**grafana**	
Telemeter クライアント	**telemeterClient**	
Prometheus アダプター	**k8sPrometheusAdapter**	
Thanos Querier	**thanosQuerier**	
Thanos Ruler		**thanosRuler**



注記

Prometheus キーは、**cluster-monitoring-config ConfigMap** で **prometheusK8s** と呼ばれ、**user-workload-monitoring-config ConfigMap** オブジェクトで **prometheus** と呼ばれています。

2.6. モニタリングコンポーネントの異なるノードへの移動

モニタリングスタックコンポーネントのいずれかを指定されたノードに移動できます。

前提条件

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
  - **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。
   - **OpenShift Container Platform のコアプロジェクトをモニターするコンポーネントを移行するには**、以下を実行します。

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

```
$ oc -n openshift-monitoring edit configmap cluster-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: cluster-monitoring-config
  namespace: openshift-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、宛先ノードのグループ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、OpenShift Container Platform のコアプロジェクトのモニタリングコンポーネントを、 **nodename: controlplane1**、 **nodename: worker1**、**nodename: worker2**、および **nodename: worker2** のラベルが付けられた特定のノードに移行するには、以下を使用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
```

```
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: controlplane1
    prometheusK8s:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    alertmanagerMain:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    kubeStateMetrics:
      nodeSelector:
        nodename: worker1
    grafana:
      nodeSelector:
        nodename: worker1
    telemeterClient:
      nodeSelector:
        nodename: worker1
    k8sPrometheusAdapter:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    openshiftStateMetrics:
      nodeSelector:
        nodename: worker1
    thanosQuerier:
      nodeSelector:
        nodename: worker1
        nodename: worker2
```

- **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**、以下を実行します。

  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。

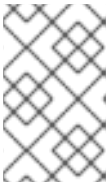
注記

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。



注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化
- 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。

2.7. モニタリングコンポーネントへの容認 (TOLERATION) の割り当て

容認をモニタリングスタックのコンポーネントに割り当て、それらをテイントされたノードに移動することができます。

前提条件

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
  - **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。

- **容認をコア OpenShift Container Platform プロジェクトをモニターするコンポーネントに割り当てるには**、以下を実行します。

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

  ```
  $ oc -n openshift-monitoring edit configmap cluster-monitoring-config
  ```

  b. コンポーネントの **tolerations** を指定します。

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
    name: cluster-monitoring-config
    namespace: openshift-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。以下の例は、サンプルのテイントを容認するように **alertmanagerMain** コンポーネントを設定します。

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

- **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るには**、以下を実行します。

  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。

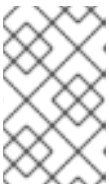
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>:
      tolerations:
        <toleration_specification>
```

<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。



注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

警告

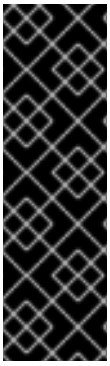
変更がモニタリング設定マップに保存されると、関連するプロジェクトの Pod およびその他のリソースが再デプロイされる可能性があります。該当するプロジェクトの実行中のモニタリングプロセスも再起動する可能性があります。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化
- 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。

2.8. 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、メトリクスデータまたはアラート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

Prometheus の割り当てられた PVC を使用してクラスターモニタリングを実行している場合、クラスターのアップグレード時に OOM による強制終了が生じる可能性があります。永続ストレージが Prometheus 用に使用される場合、Prometheus のメモリー使用量はクラスターのアップグレード時、およびアップグレードの完了後の数時間で 2 倍になります。OOM による強制終了の問題を回避するには、ワーカーノードで、アップグレード前に利用可能なメモリーのサイズを 2 倍にできるようにします。たとえば、推奨される最小ノード（RAM が 8 GB の 2 コア）でモニタリングを実行している場合は、メモリーを 16 GB に増やします。詳細は、BZ#1925061 を参照してください。

注記

「設定可能な推奨のストレージ技術」を参照してください。

2.8.1. 永続ストレージの前提条件

- ディスクが一杯にならないように、十分なローカル永続ストレージを確保します。必要な永続ストレージは Pod 数によって異なります。永続ストレージのシステム要件については、「Prometheus データベースのストレージ要件」を参照してください。
- 永続ボリューム要求 (PVC) で要求される永続ボリューム (PV) が利用できる状態にあることを確認する必要があります。各レプリカに 1 つの PV が必要です。Prometheus には 2 つのレプリカがあり、Alertmanager には 3 つのレプリカがあるため、モニタリングスタック全体をサポートするには、合計で 5 つの PV が必要になります。PV は、ローカルストレージ Operator で利用できる必要があります。動的にプロビジョニングされるストレージを有効にすると、この設定は適用されません。
- ストレージのブロックタイプを使用します。
- ローカル永続ストレージを設定します。

注記

永続ストレージにローカルボリュームを使用する場合は、**LocalVolume** オブジェクトの **volumeMode: block** で記述される raw ブロックボリュームを使用しないでください。Elasticsearch は raw ブロックボリュームを使用できません。

2.8.2. 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

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
  - **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。
   - **OpenShift Container Platform のコア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**、以下を実行します。
     a. **openshift-monitoring** プロジェクトで **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

     ```
     $ oc -n openshift-monitoring edit configmap cluster-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: cluster-monitoring-config
       namespace: openshift-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、OpenShift Container Platform のコアコンポーネントをモニターする 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: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      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、Alertmanager 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: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 10Gi
```

- **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**、以下を実行します。

  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。

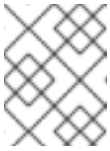
以下の例では、ユーザー定義プロジェクトをモニターする Prometheus インスタンスのローカル永続ストレージを要求する PVC を設定します。

```
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: 10Gi
```



注記

thanosRuler コンポーネントのストレージ要件は、評価されるルールの数と、各ルールによって生成されるサンプル数によって異なります。

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。

注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

2.8.3. Prometheus メトリクスデータの保持期間の変更

デフォルトで、OpenShift Container Platform クラスター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。この保持期間は、データ削除のタイミングを調整するために変更できます。

前提条件

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
  - **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。
   - **OpenShift Container Platform のコア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、以下を実行します。

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

```
$ oc -n openshift-monitoring edit configmap cluster-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: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      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、OpenShift Container Platform のコアコンポーネントをモニターする 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: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: 24h
```

- **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するには**、以下を実行します。

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。

注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化
- 永続ストレージについて
- Optimizing storage

2.9. 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。

クラスター管理者は、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。
- ターゲットを収集できない場合や、収集サンプルのしきい値に達する際に実行される **アラートを作成** 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。

2.9.1. 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

- **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
- **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。

注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

2.9.2. 収集サンプルアラートの作成

以下の場合に通知するアラートを作成できます。

- ターゲットを収集できず、指定された **for** の期間利用できない
- 指定された **for** の期間、収集サンプルのしきい値に達するか、またはこの値を上回る

前提条件

- **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
- 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。

- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- **enforcedSampleLimit** を使用して、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制限している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ターゲットがダウンし、実行されたサンプル制限に近づく際に通知するアラートを指定して YAML ファイルを作成します。この例のファイルは **monitoring-stack-alerts.yaml** という名前です。

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    prometheus: k8s
    role: alert-rules
  name: monitoring-stack-alerts (1)
  namespace: ns1 (2)
spec:
  groups:
  - name: general.rules
    rules:
    - alert: TargetDown (3)
      annotations:
        message: '{{ printf "%.4g" $value }}% of the {{ $labels.job }}/{{ $labels.service
          }} targets in {{ $labels.namespace }} namespace are down.' (4)
      expr: 100 * (count(up == 0) BY (job, namespace, service) / count(up) BY (job,
        namespace, service)) > 10
      for: 10m (5)
      labels:
        severity: warning (6)
    - alert: ApproachingEnforcedSamplesLimit (7)
      annotations:
        message: '{{ $labels.container }} container of the {{ $labels.pod }} pod in the {{
$labels.namespace }} namespace consumes {{ $value | humanizePercentage }} of the
samples limit budget.' (8)
      expr: scrape_samples_scraped/50000 > 0.8 (9)
      for: 10m (10)
      labels:
        severity: warning (11)
```

(1) アラートルールの名前を定義します。

(2) アラートルールをデプロイするユーザー定義のプロジェクトを指定します。

(3) **TargetDown** アラートは、 **for** の期間にターゲットを収集できないか、または利用できない場合に実行されます。

(4) **TargetDown** アラートが実行される場合に出力されるメッセージ。

(5)

アラートが実行される前に、**TargetDown** アラートの条件がこの期間中 true である必要があります。

6 **TargetDown** アラートの重大度を定義します。

7 **ApproachingEnforcedSamplesLimit** アラートは、指定された **for** の期間に定義された収集サンプルのしきい値に達するか、またはこの値を上回る場合に実行されます。

8 **ApproachingEnforcedSamplesLimit** アラートの実行時に出力されるメッセージ。

9 **ApproachingEnforcedSamplesLimit** アラートのしきい値。この例では、ターゲット収集ごとのサンプル数が実行されたサンプル制限 **50000** の 80% を超えるとアラートが実行されます。アラートが実行される前に、**for** の期間も経過している必要があります。式 **scrape_samples_scraped/<number> > <threshold>** の **<number>** は **user-workload-monitoring-config ConfigMap** オブジェクトで定義される **enforcedSampleLimit** 値に一致する必要があります。

10 アラートが実行される前に、**ApproachingEnforcedSamplesLimit** アラートの条件がこの期間中 true である必要があります。

11 **ApproachingEnforcedSamplesLimit** アラートの重大度を定義します。

2. 設定をユーザー定義プロジェクトに適用します。

```
$ oc apply -f monitoring-stack-alerts.yaml
```

追加リソース

- ユーザー定義のワークロードモニタリング設定マップの作成
- ユーザー定義プロジェクトのモニタリングの有効化
- 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については、「Prometheus が大量のディスク領域を消費している理由の特定」を参照してください。

2.10. 追加ラベルの時系列 (TIME SERIES) およびアラートへの割り当て

Prometheus の外部ラベル機能を使用して、カスタムラベルを、Prometheus から出るすべての時系列およびアラートに割り当てることができます。

前提条件

- **OpenShift Container Platform のコアモニタリングコンポーネントを設定する場合**、以下を実行します。
  - **cluster-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ます。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **ユーザー定義のプロジェクトをモニターするコンポーネントを設定する場合**:
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。

- **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。
   - **カスタムラベルを、OpenShift Container Platform のコアプロジェクトをモニターする Prometheus インスタンスから出るすべての時系列およびアラートに割り当てるには**、以下を実行します。
     a. **openshift-monitoring** プロジェクトで **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

     b. **data/config.yaml** の下にすべてのメトリクスについて追加する必要のあるラベルのマップを定義します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      externalLabels:
        <key>: <value> 1
```

     1 **<key>: <value>** をキーと値のペアのマップに置き換えます。ここで、**<key>** は新規ラベルの一意の名前で、**<value>** はその値になります。

警告

prometheus または **prometheus_replica** は予約され、上書きされるため、これらをキー名として使用しないでください。

たとえば、リージョンおよび環境に関するメタデータをすべての時系列およびアラートに追加するには、以下を使用します。

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

```
    prometheusK8s:
      externalLabels:
        region: eu
        environment: prod
```

- **カスタムラベルを、ユーザー定義のプロジェクトをモニターする Prometheus インスタンスから出るすべての時系列およびアラートに割り当てるには**、以下を実行します。

  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. **data/config.yaml** の下にすべてのメトリクスについて追加する必要のあるラベルのマップを定義します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      externalLabels:
        <key>: <value> 1
```

1 **<key>: <value>** をキーと値のペアのマップに置き換えます。ここで、**<key>** は新規ラベルの一意の名前で、**<value>** はその値になります。

警告

prometheus または **prometheus_replica** は予約され、上書きされるため、これらをキー名として使用しないでください。

注記

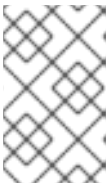
openshift-user-workload-monitoring プロジェクトでは、Prometheus はメトリクスを処理し、Thanos Ruler はアラートおよび記録ルールを処理します。**user-workload-monitoring-config ConfigMap** オブジェクトで **prometheus** の **externalLabels** を設定すると、すべてのルールではなく、メトリクスの外部ラベルのみが設定されます。

たとえば、リージョンおよび環境に関するメタデータをすべての時系列およびユーザー定義プロジェクトに関連するアラートに追加するには、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: |
    prometheus:
      externalLabels:
        region: eu
        environment: prod
```

2. 変更を適用するためにファイルを保存します。新しい設定は自動的に適用されます。



注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化
- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。

2.11. 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、Alertmanager および Thanos Querier 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、**cluster-monitoring-config** および **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

- **openshift-monitoring プロジェクトで Prometheus Operator、Prometheus、または Thanos Querier のログレベルを設定する場合には、以下を実行します。**
  - **cluster-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。
  - **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- **openshift-user-workload-monitoring プロジェクトで Prometheus Operator、Prometheus、または Thanos Ruler のログレベルを設定する場合には、**以下を実行します。
  - **cluster-admin** ロールを持つユーザーとして、または **openshift-user-workload-monitoring** プロジェクトの **user-workload-monitoring-config-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。
  - **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。
   - **openshift-monitoring プロジェクトのコンポーネントのログレベルを設定するには、**以下を実行します。
     a. **openshift-monitoring** プロジェクトで **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

     ```
     $ oc -n openshift-monitoring edit configmap cluster-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: cluster-monitoring-config
       namespace: openshift-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。

   - **openshift-user-workload-monitoring プロジェクトのコンポーネントのログレベルを設定するには、**以下を実行します。

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。

注記

user-workload-monitoring-config ConfigMap オブジェクトに適用される設定は、クラスター管理者がユーザー定義プロジェクトのモニタリングを有効にしない限りアクティブにされません。

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。

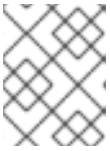
```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |  grep "log-level"
```

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。

追加リソース

- モニタリング設定マップを作成する手順は、「モニタリングスタックの設定の準備」を参照してください。
- ユーザー定義プロジェクトのモニタリングの有効化

2.12. 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有効化
- リモート正常性レポートを確認し、必要な場合はこれをオプトアウトします。

第3章 ユーザー定義プロジェクトのモニタリングの有効化

OpenShift Container Platform 4.7 では、デフォルトのプラットフォームのモニタリングに加えて、ユーザー定義プロジェクトのモニタリングを有効にできます。追加のモニタリングソリューションなしに、OpenShift Container Platform で独自のプロジェクトをモニターできるようになりました。この新機能を使用することで、コアプラットフォームコンポーネントとユーザー定義プロジェクトのモニタリングが一元化されます。

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、ユーザー定義のワークロード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 OpenShift Container Platform ではサポートされません。

3.1. ユーザー定義プロジェクトのモニタリングの有効化

クラスター管理者は、クラスターモニタリング **ConfigMap** オブジェクト に **enableUserWorkload: true**フィールドを設定し、ユーザー定義プロジェクトのモニタリングを有効にできます。

重要

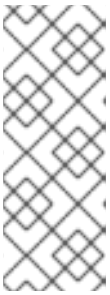
OpenShift Container Platform 4.7 では、ユーザー定義プロジェクトのモニタリングを有効にする前に、カスタム Prometheus インスタンスを削除する必要があります。

注記

OpenShift Container Platform のユーザー定義プロジェクトのモニタリングを有効にするには、**cluster-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必要があります。これにより、クラスター管理者は任意で、ユーザー定義のプロジェクトをモニターするコンポーネントを設定するパーミッションをユーザーに付与できます。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-monitoring-configConfigMap** オブジェクトを作成している。
- オプションで **user-workload-monitoring-config ConfigMap** を **openshift-user-workload-monitoring** プロジェクトに作成している。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 **ConfigMap** に設定オプションを追加できます。



注記

設定の変更を **user-workload-monitoring-config ConfigMap** に保存するたびに、**openshift-user-workload-monitoring** プロジェクトの Pod が再デプロイされます。これらのコンポーネントが再デプロイするまで時間がかかる場合があります。ユーザー定義プロジェクトのモニタリングを最初に有効にする前に **ConfigMap** オブジェクトを作成し、設定することができます。これにより、Pod を頻繁に再デプロイする必要がなくなります。

手順

1. **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

2. **enableUserWorkload: true** を **data/config.yaml** の下に追加します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true 1
```

1 **true** に設定すると、**enableUserWorkload** パラメーターはクラスター内のユーザー定義プロジェクトのモニタリングを有効にします。

3. 変更を適用するためにファイルを保存します。ユーザー定義プロジェクトのモニタリングは自動的に有効になります。

警告

変更が **cluster-monitoring-config ConfigMap** オブジェクトに保存されると、**openshift-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。

4. **prometheus-operator**、 **prometheus-user-workload** および **thanos-ruler-user-workload** Pod が **openshift-user-workload-monitoring** プロジェクトで実行中であることを確認します。Pod が起動するまでに少し時間がかかる場合があります。

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

出力例

```
NAME                                   READY   STATUS    RESTARTS   AGE
prometheus-operator-6f7b748d5b-t7nbg   2/2     Running   0          3h
prometheus-user-workload-0             4/4     Running   1          3h
prometheus-user-workload-1             4/4     Running   1          3h
thanos-ruler-user-workload-0           3/3     Running   0          3h
thanos-ruler-user-workload-1           3/3     Running   0          3h
```

追加リソース

- クラスターモニタリング設定マップの作成

- モニタリングスタックの設定
- ユーザーに対するユーザー定義プロジェクトのモニタリングを設定するためのパーミッションの付与

3.2. ユーザーに対するユーザー定義のプロジェクトをモニターするパーミッションの付与

クラスター管理者は、すべての OpenShift Container Platform のコアプロジェクトおよびユーザー定義プロジェクトをモニターできます。

クラスター管理者は、開発者およびその他のユーザーに、独自のプロジェクトをモニターするパーミッションを付与できます。特権は、以下のモニタリングロールのいずれかを割り当てることで付与されます。

- **monitoring-rules-view** ロールは、プロジェクトの **PrometheusRule** カスタムリソースへの読み取りアクセスを提供します。
- **monitoring-rules-edit** ロールは、プロジェクトの **PrometheusRule** カスタムリソースを作成し、変更し、削除するパーミッションをユーザーに付与します。
- **monitoring-edit** ロールは、**monitoring-rules-edit** ロールと同じ特権を付与します。さらに、ユーザーはサービスまたは Pod の新規の収集 ターゲットを作成できます。このロールを使用すると、**ServiceMonitor** および **PodMonitor** リソースを作成し、変更し、削除することもできます。

また、ユーザー定義のプロジェクトをモニターするコンポーネントを設定するパーミッションをユーザーに付与することもできます。

- **openshift-user-workload-monitoring** プロジェクトの user-workload-monitoring-config-edit ロールにより、**user-workload-monitoring-config ConfigMap** オブジェクトを編集できます。このロールを使用して、**ConfigMap** オブジェクトを編集し、ユーザー定義のワークロードのモニター用に Prometheus、Prometheus Operator および Thanos Ruler を設定できます。

このセクションでは、OpenShift Container Platform Web コンソールまたは CLI を使用してこれらのロールを割り当てる方法について説明します。

3.2.1. Web コンソールを使用したユーザーパーミッションの付与

OpenShift Container Platform Web コンソールを使用して、独自のプロジェクトをモニターするパーミッションをユーザーに付与できます。

前提条件

- **cluster-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。
- ロールを割り当てるユーザーアカウントがすでに存在している。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**User Management → Role Bindings → Create Binding** に移動します。
2. **Binding Type**で、「Namespace Role Binding」タイプを選択します。

3. Name フィールドに、ロールバインディングの名前を入力します。
4. Namespace フィールドで、アクセスを付与するユーザー定義プロジェクトを選択します。

重要

モニタリングロールは、Namespace フィールドで適用するプロジェクトにバインドされます。この手順を使用してユーザーに付与するパーミッションは、選択されたプロジェクトにのみ適用されます。

5. Role Name 一覧で、**monitoring-rules-view**、 **monitoring-rules-edit**、または **monitoring-edit** を選択します。
6. Subject セクションで、User を選択します。
7. Subject Name フィールドにユーザーの名前を入力します。
8. Create を選択して、ロールバインディングを適用します。

3.2.2. CLI を使用したユーザーパーミッションの付与

OpenShift CLI (**oc**) を使用して、独自のプロジェクトをモニターするパーミッションをユーザーに付与できます。

前提条件

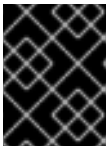
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ロールを割り当てるユーザーアカウントがすでに存在している。
- OpenShift CLI (**oc**) がインストールされている。

手順

- プロジェクトのユーザーにモニタリングロールを割り当てます。

```
$ oc policy add-role-to-user <role> <user> -n <namespace>
```
(1)

(1) **<role>** を **monitoring-rules-view**、**monitoring-rules-edit**、または **monitoring-edit** に置き換えます。



重要

選択したすべてのロールは、クラスター管理者が特定のプロジェクトにバインドする必要があります。

たとえば、**<role>** を **monitoring-edit** に、**<user>** を **johnsmith** に、**<namespace>** を **ns1** に置き換えます。これにより、ユーザー **johnsmith** に、メトリクスコレクションをセットアップし、**ns1** namespace にアラートルールを作成するパーミッションが割り当てられます。

3.3. ユーザーに対するユーザー定義プロジェクトのモニタリングを設定するためのパーミッションの付与

ユーザーに対して、ユーザー定義プロジェクトのモニタリングを設定するためのパーミッションを付与できます。

前提条件

- **cluster-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。
- ロールを割り当てるユーザーアカウントがすでに存在している。
- OpenShift CLI (**oc**) がインストールされている。

手順

- **user-workload-monitoring-config-edit** ロールを **openshift-user-workload-monitoring** プロジェクトのユーザーに割り当てます。

```
$ oc -n openshift-user-workload-monitoring adm policy add-role-to-user \
  user-workload-monitoring-config-edit <user> \
  --role-namespace openshift-user-workload-monitoring
```

3.4. カスタムアプリケーションについてのクラスター外からのメトリクスへのアクセス

独自のサービスをモニタリングする際に、コマンドラインから Prometheus 統計をクエリーする方法を説明します。クラスター外からモニタリングデータにアクセスするには、**thanos-querier** ルートを使用します。

前提条件

- 独自のサービスをデプロイしている。**ユーザー定義プロジェクトのモニタリングの有効化**手順に従ってください。

手順

1. トークンを展開して Prometheus に接続します。

```
$ SECRET=`oc get secret -n openshift-user-workload-monitoring | grep  prometheus-user-workload-token | head -n 1 | awk '{print $1 }'`
```

```
$ TOKEN=`echo $(oc get secret $SECRET -n openshift-user-workload-monitoring -o json | jq -r '.data.token') | base64 -d`
```

2. ルートホストを展開します。

```
$ THANOS_QUERIER_HOST=`oc get route thanos-querier -n openshift-monitoring -o json | jq -r '.spec.host'`
```

3. コマンドラインで独自のサービスのメトリクスをクエリーします。以下は例になります。

```
$ NAMESPACE=ns1
```

```
$ curl -X GET -kG "https://$THANOS_QUERIER_HOST/api/v1/query?" --data-urlencode
"query=up{namespace='$NAMESPACE'}" -H "Authorization: Bearer $TOKEN"
```

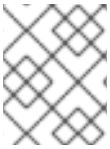
出力には、アプリケーション Pod が起動していた期間が表示されます。

出力例

```
{"status":"success","data":{"resultType":"vector","result":[{"metric":
{"__name__":"up","endpoint":"web","instance":"10.129.0.46:8080","job":"prometheus-
example-app","namespace":"ns1","pod":"prometheus-example-app-68d47c4fb6-
jztp2","service":"prometheus-example-app"},"value":[1591881154.748,"1"]}]}}
```

3.5. ユーザー定義プロジェクトのモニタリングの無効化

ユーザー定義プロジェクトのモニタリングを有効にした後に、クラスターモニタリング **ConfigMap** オブジェクトに **enableUserWorkload: false** を設定してこれを再度無効にできます。



注記

または、**enableUserWorkload: true** を削除して、ユーザー定義プロジェクトのモニタリングを無効にできます。

手順

1. **cluster-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

   a. **data/config.yaml** で **enableUserWorkload:** を **false** に設定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: false
```

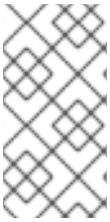
2. 変更を適用するためにファイルを保存します。ユーザー定義プロジェクトのモニタリングは自動的に無効になります。

3. **prometheus-operator**、 **prometheus-user-workload** および **thanos-ruler-user-workload** Pod が **openshift-user-workload-monitoring** プロジェクトで終了していることを確認します。これには少し時間がかかる場合があります。

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

出力例

```
No resources found in openshift-user-workload-monitoring project.
```



注記

openshift-user-workload-monitoring プロジェクトの **user-workload-monitoring-config ConfigMap** オブジェクトは、ユーザー定義プロジェクトのモニタリングが無効にされている場合は自動的に削除されません。これにより、**ConfigMap** で作成した可能性のあるカスタム設定を保持されます。

3.6. 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

4.1. メトリクスについて

OpenShift Container Platform 4.7 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 Container Platform 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: ghcr.io/rhobs/prometheus-example-app:0.3.0
        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。

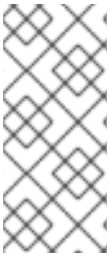
```
$ oc -n ns1 get pod
```

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 Container 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 Container Platform 4.7 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

- **cluster-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。
- 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。
- 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。

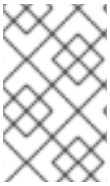
手順

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。



注記

ユーザー定義の namespace の **ServiceMonitor** リソースは、同じ namespace のサービスのみを検出できます。つまり、**ServiceMonitor** リソースの **namespaceSelector** フィールドは常に無視されます。

1. 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。

2. **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の有効化

4.3. メトリクスのクエリー

OpenShift Container Platform 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ワークロードに関する情報が提供されます。

クラスター管理者 は、すべての OpenShift Container Platform のコアプロジェクトおよびユーザー定義プロジェクトのメトリクス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. クラスター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

クラスター管理者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 Container Platform 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

クラスター管理者のみが、OpenShift Container Platform Monitoring 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

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされていること。

手順

1. OpenShift Container Platform Web コンソール内の **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。開発者はコアプラットフォームコンポーネント用の OpenShift Container Platform 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。その代わりとして、ユーザー定義プロジェクトにメトリクス UI を使用します。

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 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。
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。

追加リソース

- 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にてマウスのカーソルを保持します。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にてマウスのカーソルを保持します。クエリーの出力はポップアップに表示されます。

追加リソース

- 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。

4.4. 次のステップ

- アラートの管理

第5章 アラートの管理

OpenShift Container Platform 4.7 では、アラート UI を使用してアラート、サイレンス、およびアラートルールを管理できます。

- **アラートルール**アラートルールには、クラスター内の特定の状態を示す一連の条件が含まれます。アラートは、これらの条件が true の場合にトリガーされます。アラートルールには、アラートのルーティング方法を定義する重大度を割り当てることができます。
- **Alerts**アラートは、アラートルールで定義された条件が true の場合に発生します。アラートは、一連の状況が OpenShift Container Platform クラスター内で明確であることを示す通知を提供します。
- **サイレンス**。サイレンスをアラートに適用し、アラートの条件が true の場合に通知が送信されることを防ぐことができます。初期通知後はアラートをミュートにして、根本的な問題の解決に取り組むことができます。

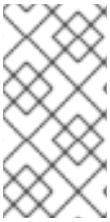
注記

アラート UI で利用可能なアラート、サイレンス、およびアラートルールは、アクセス可能なプロジェクトに関連付けられます。たとえば、**cluster-administrator** 権限でログインしている場合、アラート、サイレンス、およびアラートルールすべてにアクセスできます。

5.1. ADMINISTRATOR および DEVELOPER パースペクティブでのアラート UI へのアクセス

アラート UI は、OpenShift Container Platform Web コンソールの Administrator パースペクティブおよび Developer パースペクティブからアクセスできます。

- **Administrator** パースペクティブで、**Monitoring → Alerting** を選択します。このパースペクティブのアラート UI の主なページには、**Alerts**、**Silences**、および **Alerting Rules** という 3 つのページがあります。
- **Developer** パースペクティブで、**Monitoring → <project_name> → Alerts** を選択します。このパースペクティブのアラートでは、サイレンスおよびアラートルールはすべて **Alerts** ページで管理されます。**Alerts** ページに表示される結果は、選択されたプロジェクトに固有のものです。



注記

Developer パースペクティブでは、**Project:** 一覧からアクセスできる OpenShift Container Platform のコアプロジェクトおよびユーザー定義プロジェクトを選択できます。ただし、**cluster-admin** 権限がない場合、OpenShift Container Platform のコアプロジェクトに関連するアラート、サイレンス、およびアラートルールは表示されません。

5.2. アラート、サイレンスおよびアラートルールの検索およびフィルター

アラート UI に表示されるアラート、サイレンス、およびアラートルールをフィルターできます。このセクションでは、利用可能なフィルターオプションのそれぞれについて説明します。

アラートフィルターについて
Administrator パースペクティブでは、アラート UI の **Alerts** ページに、デフォルトの OpenShift Container Platform プロジェクトおよびユーザー定義プロジェクトに関連するアラートの詳細が提供さ

れます。このページには、各アラートの重大度、状態、およびソースの概要が含まれます。アラートが現在の状態に切り替わった時間も表示されます。

アラートの状態、重大度、およびソースでフィルターできます。デフォルトでは、**Firing** の **Platform** アラートのみが表示されます。以下では、それぞれのアラートフィルターオプションについて説明します。

- **Alert State** フィルター:
  - **Firing**アラート条件が true で、オプションの **for** の期間を経過しているためにアラートが実行されます。アラートは、条件が true である限り継続して実行されます。
  - **Pending**アラートはアクティブですが、アラート実行前のアラートルールに指定される期間待機します。
  - **Silenced**アラートは定義された期間についてサイレンスにされるようになりました。定義するラベルセレクターのセットに基づいてアラートを一時的にミュートします。一覧表示される値または正規表現のすべてに一致するアラートについては通知は送信されません。
- **Severity** フィルター:
  - **Critical**アラートをトリガーした状態は重大な影響を与える可能性があります。このアラートには、実行時に早急な対応が必要となり、通常は個人または緊急対策チーム (Critical Response Team) に送信先が設定されます。
  - **Warning**アラートは、問題の発生を防ぐために注意が必要になる可能性のある問題についての警告通知を提供します。通常、警告は早急な対応を要さないレビュー用にチケットシステムにルート指定されます。
  - **Info**アラートは情報提供のみを目的として提供されます。
  - **None**アラートには重大度が定義されていません。
  - また、ユーザー定義プロジェクトに関連するアラートの重大度の定義を作成することもできます。
- **Source** フィルター:
  - **Platform**プラットフォームレベルのアラートは、デフォルトの OpenShift Container Platform プロジェクトにのみ関連します。これらのプロジェクトは OpenShift Container Platform のコア機能を提供します。
  - **User**ユーザーアラートはユーザー定義のプロジェクトに関連します。これらのアラートはユーザーによって作成され、カスタマイズ可能です。ユーザー定義のワークロードモニタリングはインストール後に有効にでき、独自のワークロードへの可観測性を提供します。

サイレンスフィルターについて

Administrator パースペクティブでは、アラート UI の **Silences** ページには、デフォルトの OpenShift Container Platform およびユーザー定義プロジェクトのアラートに適用されるサイレンスについての詳細が示されます。このページには、それぞれのサイレンスの状態の概要とサイレンスが終了する時間の概要が含まれます。

サイレンス状態でフィルターを実行できます。デフォルトでは、**Active** および **Pending** のサイレンスのみが表示されます。以下は、それぞれのサイレンス状態のフィルターオプションについて説明しています。

- **Silence State** フィルター:

- **Active**サイレンスはアクティブで、アラートはサイレンスが期限切れになるまでミュートされます。
- **Pending**サイレンスがスケジュールされており、アクティブな状態ではありません。
- **Expired**アラートの条件が true の場合、サイレンスが期限切れになり、通知が送信されます。

アラートルールフィルターについて

Administrator パースペクティブでは、アラート UI の **Alerting Rules** ページには、デフォルトの OpenShift Container Platform およびユーザー定義プロジェクトに関連するアラートルールの詳細が示されます。このページには、各アラートルールの状態、重大度およびソースの概要が含まれます。

アラート状態、重大度、およびソースを使用してアラートルールをフィルターできます。デフォルトでは、**プラットフォーム**のアラートルールのみが表示されます。以下では、それぞれのアラートルールのフィルターオプションを説明します。

- **Alert State** フィルター:
  - **Firing**アラート条件が true で、オプションの **for** の期間を経過しているためにアラートが実行されます。アラートは、条件が true である限り継続して実行されます。
  - **Pending**アラートはアクティブですが、アラート実行前のアラートルールに指定される期間待機します。
  - **Silenced**アラートは定義された期間についてサイレンスにされるようになりました。定義するラベルセレクターのセットに基づいてアラートを一時的にミュートします。一覧表示される値または正規表現のすべてに一致するアラートについては通知は送信されません。
  - **Not Firing**アラートは実行されません。
- **Severity** フィルター:
  - **Critical**アラートルールで定義される状態は重大な影響を与える可能性があります。true の場合、これらの状態には早急な対応が必要です。通常、ルールに関連するアラートは個別または緊急対策チーム (Critical Response Team) に送信先が設定されます。
  - **Warning**アラートルールで定義される状態は、問題の発生を防ぐために注意を要する場合があります。通常、ルールに関連するアラートは早急な対応を要さないレビュー用にチケットシステムにルート指定されます。
  - **Info**アラートルールは情報アラートのみを提供します。
  - **None**アラートルールには重大度が定義されていません。
  - ユーザー定義プロジェクトに関連するアラートルールのカスタム重大度定義を作成することもできます。
- **Source** フィルター:
  - **Platform**プラットフォームレベルのアラートルールは、デフォルトの OpenShift Container Platform プロジェクトにのみ関連します。これらのプロジェクトは OpenShift Container Platform のコア機能を提供します。
  - **User**ユーザー定義のワークロードアラートルールは、ユーザー定義プロジェクトに関連します。これらのアラートルールはユーザーによって作成され、カスタマイズ可能です。ユーザー定義のワークロードモニタリングはインストール後に有効にでき、独自のワークロードへの可観測性を提供します。

Developer パースペクティブでのアラート、サイレンスおよびアラートルールの検索およびフィルター

Developer パースペクティブのアラート UI の「Alerts」ページでは、選択されたプロジェクトに関連するアラートとサイレンスを組み合わせたビューを提供します。規定するアラートルールへのリンクが表示されるアラートごとに提供されます。

このビューでは、アラートの状態と重大度でフィルターを実行できます。デフォルトで、プロジェクトへのアクセスパーミッションがある場合は、選択されたプロジェクトのすべてのアラートが表示されます。これらのフィルターは Administrator パースペクティブについて記載されているフィルターと同じです。

5.3. アラート、サイレンスおよびアラートルールについての情報の取得

アラート UI は、アラートおよびそれらを規定するアラートルールおよびサイレンスについての詳細情報を提供します。

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。

手順

Administrator パースペクティブでアラートについての情報を取得するには、以下を実行します。

1. OpenShift Container Platform Web コンソールを開き、**Monitoring → Alerting → Alerts** ページに移動します。
2. オプション: 検索一覧で **Name** フィールドを使用し、アラートを名前で検索します。
3. オプション: **Filter** 一覧でフィルターを選択し、アラートを状態、重大度およびソースでフィルターします。
4. オプション: 1 つ以上の **Name**、**Severity**、**State**、および **Source** 列ヘッダーをクリックし、アラートを並べ替えます。
5. **Alert Details** ページに移動するためにアラートの名前を選択します。このページには、アラートの時系列データを示すグラフが含まれます。また、以下を含むアラートについての情報も含まれます。
   - アラートの説明
   - アラートに関連付けられたメッセージ
   - アラートに割り当てられるラベル
   - アラートを規定するアラートルールへのリンク
   - アラートが存在する場合のアラートのサイレンス

Administrator パースペクティブでサイレンスについての情報を取得するには、以下を実行します。

1. **Monitoring → Alerting → Silences** ページに移動します。
2. オプション: **Search by name** フィールドを使用し、サイレンスを名前でフィルターします。

3. オプション: **Filter** 一覧でフィルターを選択し、サイレンスをフィルターします。デフォルトでは、**Active** および **Pending** フィルターが適用されます。

4. オプション: 1 つ以上の **Name**、**Firing Alerts**、および **State** 列ヘッダーをクリックしてサイレンスを並べ替えます。

5. **Silence Details** ページに移動するサイレンスの名前を選択します。このページには、以下の詳細が含まれます。

    - アラート仕様
    - 開始時間
    - 終了時間
    - サイレンス状態
    - 発生するアラートの数および一覧

Administrator パースペクティブでアラートルールについての情報を取得するには、以下を実行します。

1. **Monitoring → Alerting → Alerting Rules** ページに移動します。

2. オプション: **Filter** 一覧でフィルターを選択し、アラートルールを状態、重大度およびソースでフィルターします。

3. オプション: 1 つ以上の **Name**、**Severity**、**Alert State**、および **Source** 列ヘッダーをクリックし、アラートルールを並べ替えます。

4. アラートルールの名前を選択し、**Alerting Rule Details** ページに移動します。このページには、アラートルールに関する以下の情報が含まれます。

    - アラートルール名、重大度、および説明
    - アラートを発生させるための条件を定義する式
    - アラートを発生させるための条件が true である期間
    - アラートルールに規定される各アラートのグラフ。アラートを発生させる際に使用する値が表示されます。
    - アラートルールで規定されるすべてのアラートについての表

Developer パースペクティブでアラート、サイレンス、およびアラートルールについての情報を取得するには、以下を実行します。

1. **Monitoring → <project_name> → Alerts** ページに移動します。

2. アラート、サイレンス、またはアラートルールの詳細を表示します。

    - **Alert Details** を表示するには、アラート名の左側で **>** を選択し、一覧でアラートを選択します。
    - **Silence Details** を表示するには、**Alert Details** ページの **Silenced By** セクションでサイレンスを選択します。**Silence Details** ページには、以下の情報が含まれます。
        - アラート仕様

- 開始時間
- 終了時間
- サイレンス状態
- 発生するアラートの数および一覧

- **アラートルールの詳細を表示するには、** Alerts ページのアラートの右側にある メニューで **View Alerting Rule** を選択します。

注記

選択したプロジェクトに関連するアラート、サイレンスおよびアラートルールのみが **Developer** パースペクティブに表示されます。

5.4. アラートルールの管理

OpenShift Container Platform モニタリングには、デフォルトのアラートルールのセットが同梱されます。クラスター管理者は、デフォルトのアラートルールを表示できます。

OpenShift Container Platform 4.7 では、ユーザー定義プロジェクトでアラートルールを作成し、表示し、編集し、削除することができます。

アラートルールについての考慮事項

- デフォルトのアラートルールは OpenShift Container Platform クラスター用に使用され、それ以外の目的では使用されません。
- 一部のアラートルールには、複数の意図的に同じ名前が含まれます。それらは同じイベントについてのアラートを送信しますが、それぞれ異なるしきい値、重大度およびそれらの両方が設定されます。
- 抑制 (inhibition) ルールは、高い重大度のアラートが実行される際に実行される低い重大度のアラートの通知を防ぎます。

5.4.1. ユーザー定義プロジェクトのアラートの最適化

アラートルールの作成時に以下の推奨事項を考慮して、独自のプロジェクトのアラートを最適化できます。

- **プロジェクト用に作成するアラートルールの数を最小限にします**。影響を与える条件についてユーザーに通知するアラートルールを作成します。影響を与えない条件について数多くのアラートを生成すると、関連性のあるアラートを認識することはより困難になります。
- **原因ではなく現象についてのアラートルールを作成します**。根本的な原因に関係なく状態について通知するアラートルールを作成します。次に、原因を調査できます。アラートルールのそれぞれが特定の原因にのみ関連する場合に、さらに多くのアラートルールが必要になります。そのため、いくつかの原因は見落される可能性があります。
- **アラートルールを作成する前にプランニングを行います**。重要な現象と、その発生時に実行するアクションを決定します。次に、現象別にアラートルールをビルドします。

- **クリアなアラートメッセージングを提供します**。アラートメッセージに現象および推奨されるアクションを記載します。
- **アラートルールに重大度レベルを含めます**。アラートの重大度は、報告される現象が生じた場合に取るべき対応によって異なります。たとえば、現象に個人または緊急対策チーム (Critical Response Team) による早急な対応が必要な場合、重大アラートをトリガーする必要があります。
- **アラートルーティングを最適化します**。ルールがデフォルトの OpenShift Container Platform メトリクスをクエリーしない場合には、**openshift-user-workload-monitoring** プロジェクトの Prometheus インスタンスにアラートルールを直接デプロイします。これにより、アラートルールの待ち時間が短縮され、モニタリングコンポーネントへの負荷が最小限に抑えられます。

警告

ユーザー定義プロジェクトのデフォルトの OpenShift Container Platform メトリクスは、CPU およびメモリーの使用状況、帯域幅の統計、およびパケットレートについての情報を提供します。ルールを **openshift-user-workload-monitoring** プロジェクトの Prometheus インスタンスに直接ロート指定する場合、これらのメトリクスをアラートルールに含めることはできません。アラートルールの最適化は、ドキュメントを参照し、モニタリング用のアーキテクチャーの全体像を把握している場合にのみ使用してください。

追加リソース

- アラートの最適化に関する追加のガイドラインについては、Prometheus アラートのドキュメントを参照してください。
- OpenShift Container Platform 4.7 モニタリングアーキテクチャーについての詳細は、「モニタリングスタックについて」を参照してください。

5.4.2. ユーザー定義プロジェクトのアラートルールの作成

ユーザー定義のプロジェクトについてアラートルールを作成できます。これらのアラートルールは、選択したメトリクスの値に基づいてアラートを実行します。

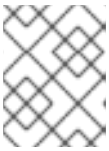
前提条件

- 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。
- アラートルールを作成する必要のある namespace の **monitoring-rules-edit** ロールを持つユーザーとしてログインします。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. アラートルールの YAML ファイルを作成します。この例では、**example-app-alerting-rule.yaml** という名前です。

2. アラートルール設定を YAML ファイルに追加します。以下は例になります。



注記

アラートルールの作成時に、同じ名前のルールが別のプロジェクトにある場合に、プロジェクトのラベルがこのアラートルールに対して適用されます。

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert
      expr: version{job="prometheus-example-app"} == 0
```

この設定により、**example-alert** という名前のアラートルールが作成されます。アラートルールは、サンプルサービスによって公開される **version** メトリクスが **0** になるとアラートを実行します。

重要

ユーザー定義のアラートルールには、独自のプロジェクトおよびクラスターメトリクスのメトリクスを含めることができます。別のユーザー定義プロジェクトのメトリクスを含めることはできません。

たとえば、ユーザー定義プロジェクトの **ns1** のアラートルールには、**ns1**、および CPU およびメモリーメトリクスなどのクラスターメトリクスなどを含めることができます。ただし、ルールには **ns2** からのメトリクスを含めることはできません。

さらに、**openshift-*** コア OpenShift プロジェクトのアラートルールを作成することはできません。デフォルトで OpenShift Container Platform モニタリングはこれらのプロジェクトのアラートルールのセットを提供します。

3. 設定ファイルをクラスターに適用します。

```
$ oc apply -f example-app-alerting-rule.yaml
```

アラートルールの作成には多少時間がかかります。

5.4.3. プラットフォームメトリクスをクエリーしないアラートルールの待ち時間の短縮

ユーザー定義プロジェクトのアラートルールがデフォルトのクラスターメトリクスをクエリーしない場合、**openshift-user-workload-monitoring** プロジェクトの Prometheus インスタンスにルールを直接デプロイすることができます。これにより、Thanos Ruler が必要でない場合にこれをバイパスすることで、アラートルールの待ち時間が短縮されます。これは、モニタリングコンポーネントの全体的な負荷を最小限に抑えるのに役立ちます。

警告

ユーザー定義プロジェクトのデフォルトの OpenShift Container Platform メトリクスは、CPU およびメモリーの使用状況、帯域幅の統計、およびパケットレートについての情報を提供します。ルールを **openshift-user-workload-monitoring** プロジェクトの Prometheus インスタンスに直接デプロイする場合、これらのメトリクスをアラートルールに含めることはできません。本セクションで説明した手順は、ドキュメントを参照し、モニタリング用のアーキテクチャーの全体像を把握している場合にのみ使用してください。

前提条件

- 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。
- アラートルールを作成する必要のある namespace の **monitoring-rules-edit** ロールを持つユーザーとしてログインします。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. アラートルールの YAML ファイルを作成します。この例では、**example-app-alerting-rule.yaml** という名前です。
2. キーが **openshift.io/prometheus-rule-evaluation-scope** で、値が **leaf-prometheus** のラベルが含まれる YAML ファイルにアラートルール設定を追加します。以下は例になります。

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-alert
  namespace: ns1
  labels:
    openshift.io/prometheus-rule-evaluation-scope: leaf-prometheus
spec:
  groups:
  - name: example
    rules:
    - alert: VersionAlert
      expr: version{job="prometheus-example-app"} == 0
```

そのラベルがある場合、アラートルールは **openshift-user-workload-monitoring** プロジェクトの Prometheus インスタンスにデプロイされます。ラベルが存在しない場合、アラートルールは Theanos Ruler にデプロイされます。

1. 設定ファイルをクラスターに適用します。

```
$ oc apply -f example-app-alerting-rule.yaml
```

アラートルールの作成には多少時間がかかります。

- OpenShift Container Platform 4.7 モニタリングアーキテクチャーについての詳細は、「モニタリングスタックについて」を参照してください。

5.4.4. ユーザー定義プロジェクトのアラートルールへのアクセス

ユーザー定義プロジェクトのアラートルールを一覧表示するには、プロジェクトの **monitoring-rules-view** ロールが割り当てられている必要があります。

前提条件

- 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。
- プロジェクトの **monitoring-rules-view** ロールを持つユーザーとしてログインしている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **<project>** でアラートルールを一覧表示できます。

```
$ oc -n <project> get prometheusrule
```

2. アラートルールの設定を一覧表示するには、以下を実行します。

```
$ oc -n <project> get prometheusrule <rule> -o yaml
```

5.4.5. 単一ビューでのすべてのプロジェクトのアラートルールの一覧表示

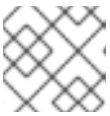
クラスター管理者は、OpenShift Container Platform のコアプロジェクトおよびユーザー定義プロジェクトのアラートルールを単一ビューで一覧表示できます。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **Administrator** パースペクティブで、**Monitoring** → **Alerting** → **Alerting Rules** に移動します。
2. **Filter** ドロップダウンメニューで、**Platform** および **User** ソースを選択します。



注記

Platform ソースはデフォルトで選択されます。

5.4.6. ユーザー定義プロジェクトのアラートルールの削除

ユーザー定義プロジェクトのアラートルールを削除できます。

前提条件

- 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。
- アラートルールを作成する必要のある namespace の **monitoring-rules-edit** ロールを持つユーザーとしてログインします。
- OpenShift CLI (**oc**) がインストールされている。

手順

- **<namespace>** のルール **<foo>** を削除するには、以下を実行します。

```
$ oc -n <namespace> delete prometheusrule <foo>
```

追加リソース

- Alertmanager ドキュメントを参照してください。

5.5. サイレンスの管理

アラートの発生時にアラートについての通知の受信を停止するためにサイレンスを作成できます。根本的な問題を解決する際に、初回の通知後にアラートをサイレンスにすることが役に立つ場合があります。

サイレンスの作成時に、サイレンスをすぐにアクティブにするか、または後にアクティブにするかを指定する必要があります。また、サイレンスの有効期限を設定する必要もあります。

既存のサイレンスを表示し、編集し、期限切れにすることができます。

5.5.1. アラートをサイレンスにする

特定のアラート、または定義する仕様に一致するアラートのいずれかをサイレンスにすることができます。

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 **edit** 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。

手順

特定のアラートをサイレンスにするには、以下を実行します。

- **Administrator** パースペクティブ:
  1. OpenShift Container Platform Web コンソールの **Monitoring → Alerting → Alerts** ページに移動します。
  2. サイレンスにする必要のあるアラートについて、右側の列で  を選択し、**Silence Alert** を選択します。**Silence Alert** フォームは、選択したアラートの事前に設定された仕様と共に表示されます。
  3. オプション: サイレンスを変更します。
  4. サイレンスを作成する前にコメントを追加する必要があります。

5. サイレンスを作成するには、**Silence** を選択します。

- **Developer** パースペクティブ:

1. OpenShift Container Platform Web コンソールで、**Monitoring → <project_name> → Alerts** ページに移動します。
2. アラート名の左側にある **>** を選択して、アラートの詳細を展開します。拡張されたビューでアラートの名前を選択し、アラートの **Alert Details** ページを開きます。
3. **Silence Alert** を選択します。**Silence Alert** フォームが、選択したアラートの事前に設定された仕様と共に表示されます。
4. オプション: サイレンスを変更します。
5. サイレンスを作成する前にコメントを追加する必要があります。
6. サイレンスを作成するには、**Silence** を選択します。

Administrator パースペクティブにアラート仕様を作成してアラートのセットをサイレンスにするには、以下を実行します。

1. OpenShift Container Platform Web コンソールの **Monitoring → Alerting → Silences** ページに移動します。
2. **Create Silence** を選択します。
3. **Create Silence** フォームで、アラートのスケジュール、期間、およびラベルの詳細を設定します。また、サイレンスのコメントを追加する必要もあります。
4. 直前の手順で入力したラベルセクターに一致するアラートのサイレンスを作成するには、**Silence** を選択します。

5.5.2. サイレンスの編集

サイレンスは編集することができます。 これにより、既存のサイレンスが期限切れとなり、変更された設定で新規のサイレンスが作成されます。

手順

Administrator パースペクティブでサイレンスを編集するには、以下を実行します。

1. **Monitoring → Alerting → Silences** ページに移動します。
2. 変更するサイレンスについて、最後の列の  を選択し、**Edit silence** を選択します。または、サイレンスについて **Silence Details** ページで **Actions → Edit Silence** を選択できます。
3. **Edit Silence** ページで変更を入力し、**Silence** を選択します。これにより、既存のサイレンスが期限切れとなり、選択された設定でサイレンスが作成されます。

Developer パースペクティブでサイレンスを編集するには、以下を実行します。

1. **Monitoring → <project_name> → Alerts** ページに移動します。

2. アラート名の左側にある > を選択して、アラートの詳細を展開します。拡張されたビューでアラートの名前を選択し、アラートの **Alert Details** ページを開きます。

3. そのページの **Silenced By** セクションでサイレンスの名前を選択し、サイレンスの **Silence Details** ページに移動します。

4. **Silence Details** ページに移動するサイレンスの名前を選択します。

5. サイレンスについて、**Silence Details** ページで **Actions → Edit Silence** を選択します。

6. **Edit Silence** ページで変更を入力し、**Silence** を選択します。これにより、既存のサイレンスが期限切れとなり、選択された設定でサイレンスが作成されます。

5.5.3. 有効期限切れにするサイレンス

サイレンスは有効期限切れにすることができます。サイレンスはいったん期限切れになると、永久に無効にされます。

手順

Administrator パースペクティブでサイレンスを期限切れにするには、以下を実行します。

1. **Monitoring → Alerting → Silences** ページに移動します。

2. 変更するサイレンスについて、最後の列の  を選択し、**Expire silence** を選択します。または、サイレンスの **Silence Details** ページで **Actions → Expire Silence** を選択できます。

Developer パースペクティブでサイレンスを期限切れにするには、以下を実行します。

1. **Monitoring → <project_name> → Alerts** ページに移動します。

2. アラート名の左側にある > を選択して、アラートの詳細を展開します。拡張されたビューでアラートの名前を選択し、アラートの **Alert Details** ページを開きます。

3. そのページの **Silenced By** セクションでサイレンスの名前を選択し、サイレンスの **Silence Details** ページに移動します。

4. **Silence Details** ページに移動するサイレンスの名前を選択します。

5. サイレンスの **Silence Details** ページで **Actions → Expire Silence** を選択します。

5.6. 外部システムへの通知の送信

OpenShift Container Platform 4.7 では、実行するアラートをアラート UI で表示できます。アラートは、デフォルトでは通知システムに送信されるように設定されません。以下のレシーバータイプにアラートを送信するように OpenShift Container Platform を設定できます。

- PagerDuty
- Webhook
- Email
- Slack

レシーバーへのアラートのルートを指定することにより、障害が発生する際に適切なチームに通知をタイムリーに送信できます。たとえば、重大なアラートには早急な対応が必要となり、通常は個人または緊急対策チーム (Critical Response Team) に送信先が設定されます。重大ではない警告通知を提供するアラートは、早急な対応を要さないレビュー用にチケットシステムにルート指定される可能性があります。

Watchdog アラートの使用によるアラートが機能することの確認

OpenShift Container Platform モニタリングには、継続的に実行される Watchdog アラートが含まれます。Alertmanager は、Watchdog のアラート通知を設定された通知プロバイダーに繰り返し送信します。通常、プロバイダーは Watchdog アラートの受信を停止する際に管理者に通知するように設定されます。このメカニズムは、Alertmanager と通知プロバイダー間の通信に関連する問題を迅速に特定するのに役立ちます。

5.6.1. アラートレシーバーの設定

アラートレシーバーを設定して、クラスターについての重要な問題について把握できるようにします。

前提条件

- **cluster-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。

手順

1. **Administrator** パースペクティブで、**Administration → Cluster Settings → Global Configuration → Alertmanager** に移動します。

注記

または、通知ドロワーで同じページに移動することもできます。OpenShift Container Platform Web コンソールの右上にあるベルのアイコンを選択し、**AlertmanagerReceiverNotConfigured** アラートで **Configure** を選択します。

2. ページの **Receivers** セクションで **Create Receiver** を選択します。
3. **Create Receiver** フォームで、**Receiver Name** を追加し、一覧から **Receiver Type** を選択します。
4. レシーバー設定を編集します。
   - PagerDuty receiver の場合:
     a. 統合のタイプを選択し、PagerDuty 統合キーを追加します。
     b. PagerDuty インストールの URL を追加します。
     c. クライアントおよびインシデントの詳細または重大度の指定を編集する場合は、**Show advanced configuration** を選択します。
   - Webhook receiver の場合:
     a. HTTP POST リクエストを送信するエンドポイントを追加します。
     b. デフォルトオプションを編集して解決したアラートを receiver に送信する場合は、**Show advanced configuration** を選択します。

- メール receiver の場合:

  a. 通知を送信するメールアドレスを追加します。

  b. SMTP 設定の詳細を追加します。これには、通知の送信先のアドレス、メールの送信に使用する smarthost およびポート番号、SMTP サーバーのホスト名、および認証情報を含む詳細情報が含まれます。

  c. TLS が必要であるかどうかについて選択します。

  d. デフォルトオプションを編集して解決済みのアラートが receiver に送信されないようにしたり、メール通知設定の本体を編集する必要がある場合は、**Show advanced configuration** を選択します。

- Slack receiver の場合:

  a. Slack Webhook の URL を追加します。

  b. 通知を送信する Slack チャネルまたはユーザー名を追加します。

  c. デフォルトオプションを編集して解決済みのアラートが receiver に送信されないようにしたり、アイコンおよびユーザー設定を編集する必要がある場合は、**Show advanced configuration** を選択します。チャネル名とユーザー名を検索し、これらをリンクするかどうかについて選択することもできます。

5. デフォルトでは、実行されるすべてのセレクターに一致するラベルを持つアラートは receiver に送信されます。実行されるアラートのラベルの値を、それらが receiver に送信される前の状態に完全に一致させる必要がある場合は、以下を実行します。

   a. ルーティングラベル名と値をフォームの **Routing Labels** セクションに追加します。

   b. 正規表現を使用する場合は **Regular Expression** を選択します。

   c. **Add Label** を選択して、さらにルーティングラベルを追加します。

6. **Create** を選択して receiver を作成します。

5.7. カスタム ALERTMANAGER 設定の適用

alertmanager-main シークレットを **openshift-monitoring** プロジェクト内で編集して、デフォルトの Alertmanager 設定を上書きできます。

前提条件

- **cluster-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。

手順

CLI で Alertmanager 設定を変更するには、以下を実行します。

1. 現在アクティブな Alertmanager 設定をファイル **alertmanager.yaml** に出力します。

```
$ oc -n openshift-monitoring get secret alertmanager-main --template='{{ index .data "alertmanager.yaml" }}' | base64 --decode > alertmanager.yaml
```

2. **alertmanager.yaml** で設定を編集します。

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - match:
      alertname: Watchdog
    repeat_interval: 5m
    receiver: watchdog
  - match:
      service: <your_service> 1
    routes:
    - match:
        <your_matching_rules> 2
      receiver: <receiver> 3
receivers:
- name: default
- name: watchdog
- name: <receiver>
#  <receiver_configuration>
```

1 **service** は、アラートを発生させるサービスを指定します。

2 **<your_matching_rules>** はターゲットアラートを指定します。

3 **receiver** は、アラートに使用する受信手段を指定します。

以下の Alertmanager 設定例は、PagerDuty をアラートレシーバーとして設定します。

```
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - match:
      alertname: Watchdog
    repeat_interval: 5m
    receiver: watchdog
  - match: service: example-app routes: - match: severity: critical receiver: team-frontend-page
receivers:
- name: default
- name: watchdog
- name: team-frontend-page pagerduty_configs: - service_key: "your-key"
```

この設定では、**example-app** サービスで実行される重大度が **critical** のアラートは、**team-frontend-page** receiver を使用して送信されます。通常、これらのタイプのアラートは、個別または緊急対策チーム (Critical Response Team) に送信先が設定されます。

3. 新規設定をファイルで適用します。

```
$ oc -n openshift-monitoring create secret generic alertmanager-main --from-
file=alertmanager.yaml --dry-run -o=yaml |  oc -n openshift-monitoring replace secret --
filename=-
```

OpenShift Container Platform Web コンソールから Alertmanager 設定を変更するには、以下を実行します。

1. Web コンソールの **Administration → Cluster Settings → Global Configuration → Alertmanager → YAML** ページに移動します。
2. YAML 設定ファイルを変更します。
3. **Save** を選択します。

追加リソース

- PagerDuty についての詳細は、PagerDuty の公式サイトを参照してください。
- **service_key** を取得する方法については、『PagerDuty Prometheus Integration Guide 』を参照してください。
- 各種のアラートレシーバー経由でアラートを設定する方法については、「Alertmanager configuration」を参照してください。

5.8. 次のステップ

- モニタリングダッシュボードの確認

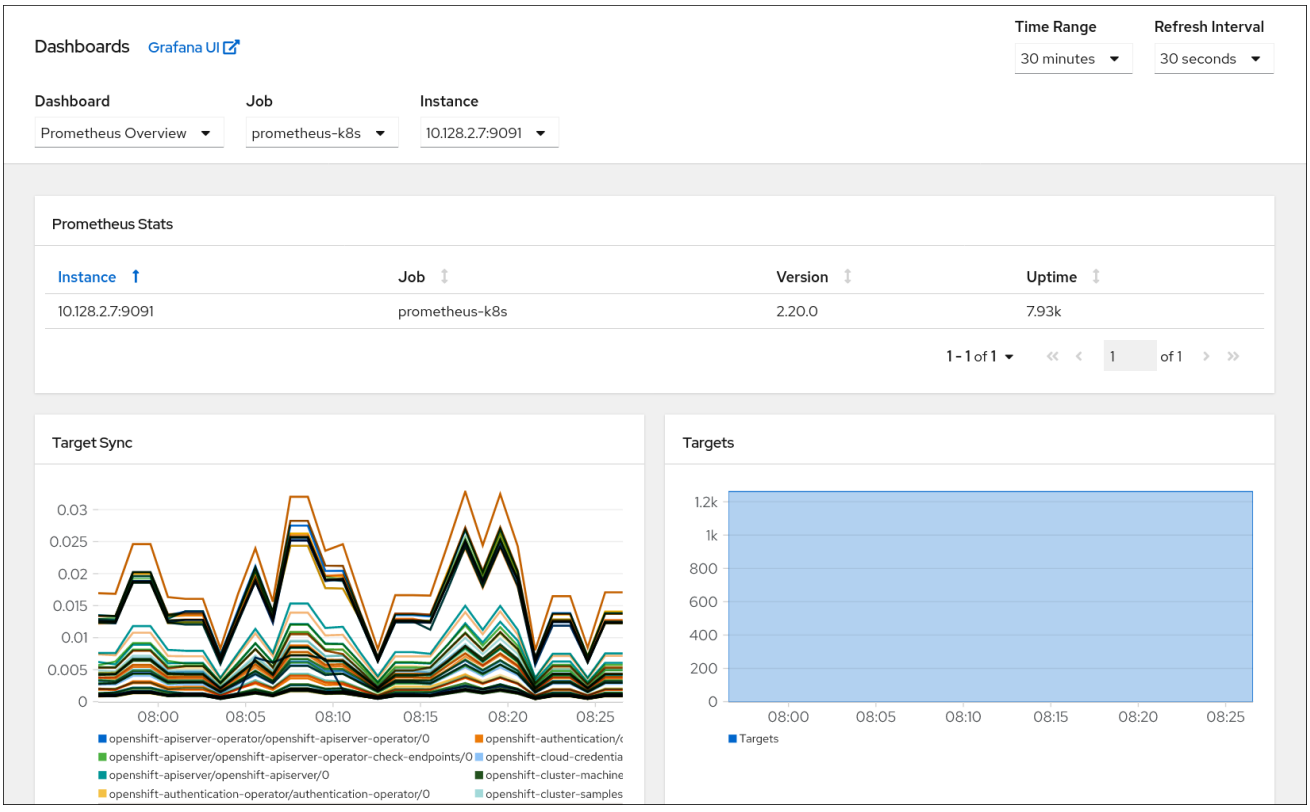
第6章 モニタリングダッシュボードの確認

OpenShift Container Platform 4.7 は、クラスターコンポーネントおよびユーザー定義のワークロードの状態を理解するのに役立つ包括的なモニタリングダッシュボードのセットを提供します。

Administrator パースペクティブでは、以下を含む OpenShift Container Platform のコアコンポーネントのダッシュボードにアクセスできます。

- API パフォーマンス
- etcd
- Kubernetes コンピュートリソース
- Kubernetes ネットワークリソース
- Prometheus
- クラスターおよびノードのパフォーマンスに関連する USE メソッドダッシュボード

図6.1 Administrator パースペクティブのダッシュボードの例



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 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6.2 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. クラスター管理者としてのモニタリングダッシュボードの確認

Administrator パースペクティブでは、OpenShift Container Platform クラスターのコアコンポーネントに関連するダッシュボードを表示できます。

前提条件

- **cluster-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。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Monitoring → Dashboards** に移動します。
2. **Dashboard** 一覧でダッシュボードを選択します。**etcd** や **Prometheus** ダッシュボードなどの一部のダッシュボードは、選択時に追加のサブメニューを生成します。
3. 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。
4. オプション:、**Refresh Interval** を選択します。
5. 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. 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

- 開発者として、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 Container Platform 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。

追加リソース

- Developer パースペクティブを使用したプロジェクトおよびアプリケーションメトリクスのモニタリング

6.3. 次のステップ

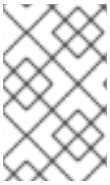
- サードパーティーの UI へのアクセス

第7章 サードパーティーの UI へのアクセス

統合メトリクス、アラート、およびダッシュボード UI は OpenShift Container Platform Web コンソールで提供されます。これらの統合 UI の使用方法についての詳細は、以下のドキュメントを参照してください。

- メトリクスの管理
- アラートの管理
- モニタリングダッシュボードの確認

OpenShift Container Platform は、Prometheus、Alertmanager、および Grafana サードパーティーインターフェースへのアクセスも提供します。



注記

サードパーティーのモニタリングインターフェースへのデフォルトのアクセスは今後の OpenShift Container Platform リリースで削除される可能性があります。その場合、ポート転送を使用してそれらにアクセスする必要があります。

注記

OpenShift Container Platform モニタリングスタックおよびダッシュボードと共に提供される Grafana インスタンスは読み取り専用です。

注記

Grafana ダッシュボードには、Kubernetes および **cluster-monitoring** メトリクスのみが含まれます。追加のプラットフォームコンポーネントは、OpenShift Container Platform Web コンソールの **Monitoring → Dashboards** に含まれます。

7.1. WEB コンソールを使用したサードパーティーのモニタリング UI へのアクセス

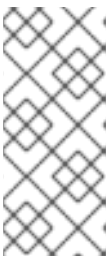
OpenShift Container Platform Web コンソールを使用して Alertmanager、Grafana、Prometheus、および Thanos Querier Web UI にアクセスできます。

前提条件

- **cluster-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。

手順

1. **Administrator** パースペクティブで、**Networking → Routes** に移動します。



注記

Developer パースペクティブから、サードパーティーの Alertmanager、Grafana、Prometheus、および Thanos Querier UI にアクセスすることはできません。代わりに Developer パースペクティブの Metrics UI リンクを使用します。これには、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。

2. **Project:** 一覧で **openshift-monitoring** プロジェクトを選択します。
3. サードパーティーのモニタリング UI にアクセスします。
   - **alertmanager-main** 行の URL を選択し、Alertmanager UI のログインページを開きます。
   - **grafana** 行の URL を選択し、Grafana UI のログインページを開きます。
   - **prometheus-k8s** 行の URL を選択し、Prometheus UI のログインページを開きます。
   - **thanos-querier** 行の URL を選択し、Thanos Querier UI のログインページを開きます。
4. **Log in with OpenShift** を選択し、penShift Container Platform 認証情報を使用してログインします。

7.2. CLI の使用によるサードパーティーのモニタリング UI へのアクセス

OpenShift CLI (**oc**) ツールを使用して、Prometheus、Alertmanager、および Grafana Web UI の URL を取得できます。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下を実行して、**openshift-monitoring** プロジェクトのルートを一覧表示します。

```
$ oc -n openshift-monitoring get routes
```

出力例

```
NAME                HOST/PORT                                                  ...
alertmanager-main   alertmanager-main-openshift-monitoring.apps._url_.openshift.com ...
grafana             grafana-openshift-monitoring.apps._url_.openshift.com      ...
prometheus-k8s      prometheus-k8s-openshift-monitoring.apps._url_.openshift.com    ...
thanos-querier      thanos-querier-openshift-monitoring.apps._url_.openshift.com    ...
```

2. Web ブラウザーを使用して **HOST/PORT** ルートに移動します。
3. **Log in with OpenShift** を選択し、OpenShift 認証情報を使用してログインします。

重要

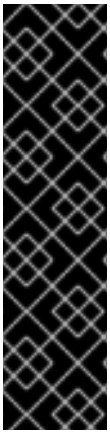
モニタリングルートは Cluster Monitoring Operator によって管理され、ユーザーが変更することはできません。

7.3. 次のステップ

- 自動スケーリングのカスタムアプリケーションメトリクスの公開

第8章 自動スケーリングのカスタムアプリケーションメトリクスの公開

Horizontal Pod Autoscaler のカスタムアプリケーションメトリクスをエクスポートできます。



重要

Prometheus アダプターはテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

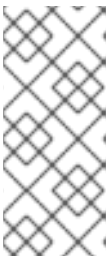
Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

8.1. HORIZONTAL POD AUTOSCALING のカスタムアプリケーションメトリクスの公開

prometheus-adapter リソースを使用して、Horizontal Pod Autoscaler のカスタムアプリケーションメトリクスを公開できます。

前提条件

- カスタム Prometheus インスタンスがインストールされていること。この例では、Prometheus がユーザー定義の **custom-prometheus** プロジェクトにインストールされていることが前提になります。



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、ユーザー定義のワークロード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 OpenShift Container Platform ではサポートされません。

- ユーザー定義のプロジェクトにアプリケーションとサービスをデプロイしている。この例では、アプリケーションとそのサービスモニターがユーザー定義の **custom-prometheus** プロジェクトにインストールされていることが前提になります。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 設定の YAML ファイルを作成します。この例では、これは **deploy.yaml** というファイルになります。
2. **prometheus-adapter** のサービスアカウント、必要なロールおよびロールバインディングを作成するための設定の詳細を追加します。

```
kind: ServiceAccount
```

```
apiVersion: v1
metadata:
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: custom-metrics-server-resources
rules:
- apiGroups:
  - custom.metrics.k8s.io
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: custom-metrics-resource-reader
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  - pods
  - services
  verbs:
  - get
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: custom-metrics:system:auth-delegator
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: custom-metrics-auth-reader
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: extension-apiserver-authentication-reader
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: custom-metrics-resource-reader
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: custom-metrics-resource-reader
subjects:
- kind: ServiceAccount
  name: custom-metrics-apiserver
  namespace: custom-prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: hpa-controller-custom-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: custom-metrics-server-resources
subjects:
- kind: ServiceAccount
  name: horizontal-pod-autoscaler
  namespace: kube-system
---
```

3. **prometheus-adapter** のカスタムメトリクスの設定の詳細を追加します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: adapter-config
  namespace: custom-prometheus
data:
  config.yaml: |
    rules:
    - seriesQuery: 'http_requests_total{namespace!="",pod!=""}' (1)
      resources:
        overrides:
          namespace: {resource: "namespace"}
          pod: {resource: "pod"}
          service: {resource: "service"}
      name:
        matches: "^(.*)_total"
        as: "${1}_per_second" (2)
      metricsQuery: 'sum(rate(<<.Series>>{<<.LabelMatchers>>}[2m])) by (<<.GroupBy>>)'
---
```

(1) 選択したメトリクスが HTTP 要求の数になるように指定します。

(2) メトリクスの頻度を指定します。

4. **prometheus-adapter** を API サービスとして登録するための設定の詳細を追加します。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.alpha.openshift.io/serving-cert-secret-name: prometheus-adapter-tls
  labels:
    name: prometheus-adapter
  name: prometheus-adapter
  namespace: custom-prometheus
spec:
  ports:
  - name: https
    port: 443
    targetPort: 6443
  selector:
    app: prometheus-adapter
  type: ClusterIP
---
apiVersion: apiregistration.k8s.io/v1beta1
kind: APIService
metadata:
  name: v1beta1.custom.metrics.k8s.io
spec:
  service:
    name: prometheus-adapter
    namespace: custom-prometheus
  group: custom.metrics.k8s.io
  version: v1beta1
  insecureSkipTLSVerify: true
  groupPriorityMinimum: 100
  versionPriority: 100
---
```

5. Prometheus アダプターイメージを一覧表示します。

```
$ oc get -n openshift-monitoring deploy/prometheus-adapter -o jsonpath="{..image}"
```

6. **prometheus-adapter** をデプロイするための設定の詳細を追加します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-adapter
  name: prometheus-adapter
  namespace: custom-prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-adapter
  template:
    metadata:
      labels:
        app: prometheus-adapter
```

```
      name: prometheus-adapter
    spec:
      serviceAccountName: custom-metrics-apiserver
      containers:
      - name: prometheus-adapter
        image: quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:a46915a206cd7d97f240687c618dd59e8848fcc3a0f51e281f3384153a12c3e0
(1)
        args:
        - --secure-port=6443
        - --tls-cert-file=/var/run/serving-cert/tls.crt
        - --tls-private-key-file=/var/run/serving-cert/tls.key
        - --logtostderr=true
        - --prometheus-url=http://prometheus-operated.default.svc:9090/
        - --metrics-relist-interval=1m
        - --v=4
        - --config=/etc/adapter/config.yaml
        ports:
        - containerPort: 6443
        volumeMounts:
        - mountPath: /var/run/serving-cert
          name: volume-serving-cert
          readOnly: true
        - mountPath: /etc/adapter/
          name: config
          readOnly: true
        - mountPath: /tmp
          name: tmp-vol
      volumes:
      - name: volume-serving-cert
        secret:
          secretName: prometheus-adapter-tls
      - name: config
        configMap:
          name: adapter-config
      - name: tmp-vol
        emptyDir: {}
```

(1) 直前の手順にある Prometheus Adapter イメージを指定します。

7. 設定をクラスターに適用します。

```
$ oc apply -f deploy.yaml
```

出力例

```
serviceaccount/custom-metrics-apiserver created
clusterrole.rbac.authorization.k8s.io/custom-metrics-server-resources created
clusterrole.rbac.authorization.k8s.io/custom-metrics-resource-reader created
clusterrolebinding.rbac.authorization.k8s.io/custom-metrics:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/custom-metrics-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/custom-metrics-resource-reader created
clusterrolebinding.rbac.authorization.k8s.io/hpa-controller-custom-metrics created
configmap/adapter-config created
```

```
service/prometheus-adapter created
apiservice.apiregistration.k8s.io/v1.custom.metrics.k8s.io created
deployment.apps/prometheus-adapter created
```

8. ユーザー定義プロジェクトの **prometheus-adapter** Pod が **Running** 状態にあることを確認します。この例では、プロジェクトは **custom-prometheus** です。

```
$ oc -n custom-prometheus get pods prometheus-adapter-<string>
```

9. アプリケーションのメトリクスが公開され、Horizontal Pod Autoscaling を設定するために使用できます。

追加リソース

- Horizontal Pod Autoscaling についてのドキュメント を参照してください。
- Horizontal Pod Autoscaler についての Kubernetes ドキュメント を参照してください。

8.2. 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9章 モニタリング関連の問題のトラブルシューティング

9.1. ユーザー定義のメトリクスが利用できない理由の調査

ServiceMonitor リソースを使用すると、ユーザー定義プロジェクトでサービスによって公開されるメトリクスの使用方法を判別できます。**ServiceMonitor** リソースを作成している場合で、メトリクス UI に対応するメトリクスが表示されない場合は、この手順で説明されるステップを実行します。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。
- ユーザー定義のワークロードのモニタリングを有効にし、設定している。
- **user-workload-monitoring-config ConfigMap** オブジェクトを作成している。
- **ServiceMonitor** リソースを作成している。

手順

1. サービスおよび **ServiceMonitor** リソース設定で、**対応するラベルの一致を確認** します。
   a. サービスに定義されたラベルを取得します。以下の例では、**ns1** プロジェクトの **prometheus-example-app** サービスをクエリーします。

```
$ oc -n ns1 get service prometheus-example-app -o yaml
```

出力例

```
  labels:
    app: prometheus-example-app
```

   b. **ServiceMonitor** リソース設定の **matchLabels app** ラベルが直前のステップのラベルの出力と一致することを確認します。

```
$ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
```

出力例

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

注記

プロジェクトの表示パーミッションを持つ開発者として、サービスおよび **ServiceMonitor** リソースラベルを確認できます。

2. **openshift-user-workload-monitoring** プロジェクトの **Prometheus Operator のログを検査**します。

   a. **openshift-user-workload-monitoring** プロジェクトの Pod を一覧表示します。

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

   出力例

   ```
   NAME                                   READY   STATUS    RESTARTS   AGE
   prometheus-operator-776fcbbd56-2nbfm   2/2     Running   0          132m
   prometheus-user-workload-0             5/5     Running   1          132m
   prometheus-user-workload-1             5/5     Running   1          132m
   thanos-ruler-user-workload-0           3/3     Running   0          132m
   thanos-ruler-user-workload-1           3/3     Running   0          132m
   ```

   b. **prometheus-operator** Pod の **prometheus-operator** コンテナーからログを取得します。以下の例では、Pod は **prometheus-operator-776fcbbd56-2nbfm** になります。

   ```
   $ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-
   2nbfm -c prometheus-operator
   ```

   サービスモニターに問題がある場合、ログには以下のようなエラーが含まれる可能性があります。

   ```
   level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829
   component=prometheusoperator msg="skipping servicemonitor" error="it accesses file
   system via bearer token file which Prometheus specification prohibits"
   servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring
   prometheus=user-workload
   ```

3. Prometheus UI で**プロジェクトのターゲットステータスを**直接確認します。

   a. **openshift-user-workload-monitoring** プロジェクトで Prometheus インスタンスへのポート転送を確立します。

   ```
   $ oc port-forward -n openshift-user-workload-monitoring pod/prometheus-user-workload-
   0 9090
   ```

   b. Web ブラウザーで http://localhost:9090/targets を開き、 Prometheus UI でプロジェクトのターゲットのステータスを直接確認します。ターゲットに関連するエラーメッセージがあるかどうかを確認します。

4. **openshift-user-workload-monitoring** プロジェクトで **Prometheus Operator のデバッグレベルのロギングを設定** します。

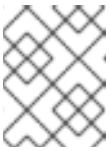
   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

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

b. **prometheusOperator** の **logLevel: debug** を **data/config.yaml** に追加し、ログレベルを **debug** に設定します。

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

c. 変更を適用するためにファイルを保存します。



注記

openshift-user-workload-monitoring プロジェクトの **prometheus-operator** は、ログレベルの変更時に自動的に再起動します。

d. **debug** ログレベルが **openshift-user-workload-monitoring** プロジェクトの **prometheus-operator** デプロイメントに適用されていることを確認します。

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

出力例

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

debug レベルのロギングにより、Prometheus Operator によって行われるすべての呼び出しが表示されます。

e. **prometheus-operator** Pod が実行されていることを確認します。

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

注記

認識されない Prometheus Operator の **loglevel** 値が設定マップに含まれる場合、**prometheus-operator** Pod が正常に再起動されない可能性があります。

f. デバッグログを確認し、Prometheus Operator が **ServiceMonitor** リソースを使用しているかどうかを確認します。ログで他の関連するエラーの有無を確認します。

追加リソース

- ユーザー定義のワークロードモニタリング設定マップの作成

- **ServiceMonitor** または **PodMonitor** の作成方法についての詳細は、「サービスのモニター法の指定」を参照してください。

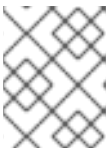
9.2. PROMETHEUS が大量のディスク領域を消費している理由の特定

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。

Prometheus が多くのディスクを消費する場合、以下の手段を使用できます。

- 収集される **収集サンプルの数を確認** します。
- **Prometheus UI での時系列データベース (TSDB) のステータスを確認**して、最も多くの時系列ラベルを作成するラベルについて確認できます。これには、クラスター管理者の権限が必要です。
- ユーザー定義メトリクスに割り当てられるバインドされていない属性の数を減らすことで、**作成される一意の時系列の数を減らします**。



注記

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。

- ユーザー定義プロジェクト間で **収集可能なサンプル数の数に制限を適用します**。これには、クラスター管理者の権限が必要です。

前提条件

- **cluster-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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **Administrator** パースペクティブで、**Monitoring** → **Metrics** に移動します。
2. **Expression** フィールドで、以下の Prometheus Query Language (PromQL)クエリーを実行します。これにより、収集サンプルの数が最も多い 10 メトリクスが返されます。

```
topk(10,count by (job)({__name__=~".+"}))
```

3. 予想されるよりも多くの収集サンプルを持つメトリクスに割り当てられたバインドされていないラベル値の数を調査します。
   - **メトリクスがユーザー定義のプロジェクトに関連する場合**、ワークロードに割り当てられたメトリクスのキーと値のペアを確認します。これらのライブラリーは、アプリケーションレベルで Prometheus クライアントライブラリーを使用して実装されます。ラベルで参

照されるバインドされていない属性の数の制限を試行します。

- **メトリクスが OpenShift Container Platform のコアプロジェクトに関連する場合**、Red Hat サポートケースを Red Hat カスタマーポータル で作成してください。

4. Prometheus UI で TSDB ステータスを確認します。
   a. **Administrator** パースペクティブで、**Networking → Routes** に移動します。
   b. **Project:** 一覧で **openshift-monitoring** プロジェクトを選択します。
   c. **prometheus-k8s** 行の URL を選択し、Prometheus UI のログインページを開きます。
   d. **Log in with OpenShift** を選択し、OpenShift Container Platform 認証情報を使用してログインします。
   e. Prometheus UI で、**Status → TSDB Status** に移動します。

関連情報

- 収集サンプルの制限を設定し、関連するアラートルールを作成する方法についての詳細は、「ユーザー定義プロジェクトの収集サンプル制限の設定」を参照してください。
- サポートケースの送信