OpenShift Container Platform 4.13

Cluster Observability Operator

OpenShift Container Platform での Cluster Observability Operator の設定と使用

Last Updated: 2024-09-12

OpenShift Container Platform 4.13 Cluster Observability Operator

OpenShift Container Platform での Cluster Observability Operator の設定と使用

法律上の通知

概要

Cluster Observability Operator を使用して、OpenShift Container Platform で可観測性コンポーネントをデプロイして設定します。

目次

第1章 CLUSTER OBSERVABILITY OPERATOR リリースノート

重要

Cluster Observability Operator はテクノロジープレビュー機能としてのみ使用できます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Cluster Observability Operator (COO) は、オプションの OpenShift Container Platform Operator です。管理者はこれを使用して、さまざまなサービスやユーザーが使用できるように個別に設定できる、スタンドアロンのモニタリングスタックを作成できます。

COO は、OpenShift Container Platform のビルトインモニタリング機能を補完します。これは、Cluster Monitoring Operator (CMO) で管理されるデフォルトのプラットフォームおよびユーザーワークロードモニタリングスタックと並行してデプロイできます。

これらのリリースノートは、OpenShift Container Platform での Cluster Observability Operator の開発を追跡します。

1.1. CLUSTER OBSERVABILITY OPERATOR 0.3.0

Cluster Observability Operator 0.3.0 では、次のアドバイザリーを利用できます。

- RHEA-2024:4399 Cluster Observability Operator 0.3.0

1.1.1. 新機能および機能拡張

- 今回のリリースにより、**MonitoringStack** コンポーネントで容認とノードセレクターを使用できるようになりました。

1.1.2. バグ修正

- 以前のバージョンでは、logging UIPlugin は **Available** 状態にならず、特定のバージョンの OpenShift Container Platform にインストールされている場合、ロギング Pod は作成されませんでした。このリリースでこの問題が解決されました。(COO-34)

1.2. CLUSTER OBSERVABILITY OPERATOR 0.3.0

Cluster Observability Operator 0.3.0 では、次のアドバイザリーを利用できます。

- RHEA-2024:4399 Cluster Observability Operator 0.3.0

1.2.1. 新機能および機能拡張

- このリリースでは、Cluster Observability Operator は、今後の OpenShift Container Platform 可観測性 Web コンソール UI プラグインと可観測性コンポーネントのバックエンドサポートを追加します。

1.3. CLUSTER OBSERVABILITY OPERATOR 0.2.0

Cluster Observability Operator 0.2.0 では、次のアドバイザリーを利用できます。

- RHEA-2024:2662 Cluster Observability Operator 0.2.0

1.3.1. 新機能および機能拡張

- このリリースでは、Cluster Observability Operator は、OpenShift Container Platform Web コンソールユーザーインターフェイス (UI) の可観測性関連プラグインのインストールと管理をサポートします。(COO-58)

1.4. CLUSTER OBSERVABILITY OPERATOR 0.1.3

Cluster Observability Operator 0.1.3 では、次のアドバイザリーを利用できます。

- RHEA-2024:1744 Cluster Observability Operator 0.1.3

1.4.1. バグ修正

- 以前は、**http://<prometheus_url>:9090/graph** で Prometheus Web ユーザーインターフェイス (UI) にアクセスしようとすると、**Error opening React index.html: open web/ui/static/react/index.html: no such file or directory** エラーメッセージが表示されていました。このリリースでは問題が解決され、Prometheus Web UI が正しく表示されるようになりました。(COO-34)

1.5. CLUSTER OBSERVABILITY OPERATOR 0.1.2

Cluster Observability Operator 0.1.2 では、次のアドバイザリーを利用できます。

- RHEA-2024:1534 Cluster Observability Operator 0.1.2

1.5.1. CVE

- CVE-2023-45142

1.5.2. バグ修正

- 以前は、特定のクラスターサービスバージョン (CSV) アノテーションが COO のメタデータに含まれていませんでした。これらのアノテーションが欠落していたため、COO の一部の特長と機能がパッケージマニフェストまたは OperatorHub ユーザーインターフェイスに表示されませんでした。このリリースで、欠落していたアノテーションが追加され、この問題が解決されました。(COO-11)

- 以前は、COO の自動更新が機能せず、OperatorHub で新しいバージョンが利用可能であっても、Operator の新しいバージョンによって古いバージョンが自動的に置き換えられませんでした。このリリースでこの問題が解決されました。(COO-12)

- 以前は、Thanos Querier が 127.0.0.1 (**localhost**) のポート 9090 でネットワークトラフィックのみをリッスンしていたため、Thanos Querier サービスにアクセスしようとすると **502 Bad Gateway** エラーが発生しました。このリリースで、Thanos Querier 設定が更新され、コンポーネントがデフォルトポート (10902) でリッスンするようになり、問題が解決されました。この変更の結果、必要に応じて、SSA (Server Side Apply) を使用してポートを変更し、プロキシーチェーンを追加することもできるようになりました。(COO-14)

1.6. CLUSTER OBSERVABILITY OPERATOR 0.1.1

Cluster Observability Operator 0.1.1 では、次のアドバイザリーを利用できます。

- 2024:0550 Cluster Observability Operator 0.1.1

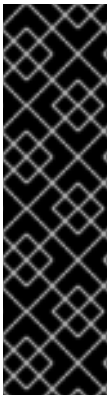
1.6.1. 新機能および機能拡張

このリリースでは、Cluster Observability Operator が更新され、制限されたネットワークまたは非接続環境での Operator のインストールがサポートされるようになりました。

1.7. CLUSTER OBSERVABILITY OPERATOR 0.1

このリリースでは、Cluster Observability Operator のテクノロジープレビューバージョンが OperatorHub で利用できるようになります。

第2章 CLUSTER OBSERVABILITY OPERATOR の概要



重要

Cluster Observability Operator はテクノロジープレビュー機能としてのみ使用できます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Cluster Observability Operator (COO) は、オプションの OpenShift Container Platform コンポーネントです。これをデプロイして、さまざまなサービスやユーザーが使用できるように個別に設定可能なスタンドアロンのモニタリングスタックを作成できます。

COO は、次のモニタリングコンポーネントをデプロイします。

- Prometheus
- Thanos Querier (オプション)
- Alertmanager (オプション)

COO コンポーネントは、Cluster Monitoring Operator (CMO) でデプロイおよび管理されるデフォルトのクラスター内モニタリングスタックとは独立して機能します。2 つの Operator でデプロイされたモニタリングスタックは競合しません。CMO でデプロイされた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に加え、COO モニタリングスタックを使用できます。

2.1. CLUSTER OBSERVABILITY OPERATOR について

Cluster Observability Operator (COO) で作成されたデフォルトのモニタリングスタックには、リモート書き込みを使用して外部エンドポイントにメトリクスを送信できる可用性の高い Prometheus インスタンスが含まれています。

各 COO スタックには、中央の場所から高可用性 Prometheus インスタンスをクエリーするために使用できるオプションの Thanos Querier コンポーネントと、さまざまなサービスのアラート設定をセットアップするために使用できるオプションの Alertmanager コンポーネントも含まれています。

2.1.1. Cluster Observability Operator を使用する利点

COO が使用する **MonitoringStack** CRD は、COO でデプロイされたモニタリングコンポーネントに対して独自のデフォルトモニタリング設定を提供しますが、より複雑な要件に合わせてカスタマイズすることも可能です。

COO で管理されるモニタリングスタックを導入すると、Cluster Monitoring Operator (CMO) でデプロイされたコアプラットフォームモニタリングスタックでは対応が難しい、または不可能なモニタリングニーズを満たすことができます。COO を使用して導入されたモニタリングスタックには、コアプラットフォームとユーザーワークロードのモニタリングに比べて次の利点があります。

拡張性

ユーザーは、COO でデプロイされたモニタリングスタックにさらに多くのメトリクスを追加できま

すが、これをコアプラットフォームモニタリングで行った場合はサポートされません。さらに、COO で管理されるスタックは、フェデレーションを使用して、コアプラットフォームのモニタリングから特定のクラスター固有のメトリクスを受け取ることができます。

マルチテナンシーのサポート

COO は、ユーザー namespace ごとにモニタリングスタックを作成できます。namespace ごとに複数のスタックをデプロイしたり、複数の namespace に単一のスタックをデプロイしたりすることもできます。たとえば、クラスター管理者、SRE チーム、開発チームは、モニタリングコンポーネントの単一の共有スタックを使用するのではなく、独自のモニタリングスタックを単一のクラスターにデプロイできます。その後、各チームのユーザーは、アプリケーションやサービスのアラート、アラートルーティング、アラートレシーバーなどの機能を個別に設定できます。

スケーラビリティー

必要に応じて、COO で管理されるモニタリングスタックを作成できます。単一のクラスター上で複数のモニタリングスタックを実行できるため、手動シャーディングを使用することで非常に大規模なクラスターを容易に監視できます。この機能は、メトリクスの数が単一の Prometheus インスタンスのモニタリング能力を超える場合に対処します。

柔軟性

Operator Lifecycle Manager (OLM) を使用して COO をデプロイすると、COO リリースが OpenShift Container Platform リリースサイクルから切り離されます。このデプロイメント方法により、リリースイテレーションが短縮され、変化する要件や問題に迅速に対応できるようになります。さらに、COO で管理されるモニタリングスタックをデプロイすることで、ユーザーは OpenShift Container Platform のリリースサイクルとは別にアラートルールを管理できます。

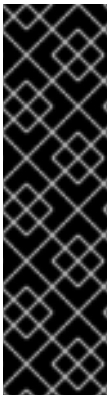
高度なカスタマイズが可能

COO は、カスタマイズを強化する Server-Side Apply (SSA) を使用して、カスタムリソース内にある 1 つの設定可能フィールドの所有権をユーザーに委譲できます。

関連情報

- Server-Side Apply (SSA) に関する Kubernetes ドキュメント

第3章 CLUSTER OBSERVABILITY OPERATOR のインストール



重要

Cluster Observability Operator はテクノロジープレビュー機能としてのみ使用できます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

クラスター管理者は、OpenShift Container Platform Web コンソールまたは CLI を使用して、OperatorHub から Cluster Observability Operator (COO) をインストールまたは削除できます。OperatorHub は、クラスター上に Operator をインストールして管理する Operator Lifecycle Manager (OLM) と連動して動作するユーザーインターフェイスです。

3.1. WEB コンソールに CLUSTER OBSERVABILITY OPERATOR のインストール

OpenShift Container Platform Web コンソールを使用して、OperatorHub から Cluster Observability Operator (COO) をインストールします。

前提条件

- **cluster-admin** クラスター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 Container Platform Web コンソールにログインしている。

手順

1. OpenShift Container Platform Web コンソールで、**Operators → OperatorHub** をクリックします。
2. **Filter by keyword** ボックスに **cluster observability operator** と入力します。
3. 結果リストで **Cluster Observability Operator** をクリックします。
4. Operator に関する情報を読み、次のデフォルトのインストール設定を確認します。
   - **Update channel → development**
   - **Version →** <most_recent_version>
   - **Installation mode → All namespaces on the cluster (default)**
   - **Installed Namespace → openshift-operators**
   - **Update approval → Automatic**

5. オプション: 要件に合わせてデフォルトのインストール設定を変更します。たとえば、別の更新チャネルをサブスクライブしたり、Operator の古いリリースバージョンをインストールしたり、Operator の新しいバージョンへの更新に手動の承認を必要とするように選択できます。

6. **Install** をクリックします。

検証

- **Operators → Installed Operators** に移動し、リストに **Cluster Observability Operator** エントリーが表示されていることを確認します。

関連情報

Operator のクラスターへの追加

3.2. WEB コンソールを使用して CLUSTER OBSERVABILITY OPERATOR をアンインストールする

OperatorHub を使用して Cluster Observability Operator (COO) をインストールした場合は、OpenShift Container Platform Web コンソールでそれをアンインストールできます。

前提条件

- **cluster-admin** クラスター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 Container Platform Web コンソールにログインしている。

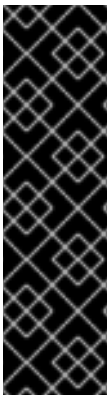
手順

1. **Operators → Installed Operators** に移動します。
2. リスト内で **Cluster Observability Operator** エントリーを見つけます。
3. このエントリーの  をクリックし、**Uninstall Operator** を選択します。

検証

- **Operator → Installed Operator** に移動し、**Cluster Observability Operator** エントリーがリストに表示されなくなったことを確認します。

第4章 サービスを関しするための CLUSTER OBSERVABILITY OPERATOR 設定



重要

Cluster Observability Operator はテクノロジープレビュー機能としてのみ使用できます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Cluster Observability Operator (COO) で管理されるモニタリングスタックを設定することで、サービスのメトリクスを監視できます。

サービスのモニタリングをテストするには、次の手順に従います。

- サービスエンドポイントを定義するサンプルサービスをデプロイします。
- COO によるサービスのモニタリング方法を指定する **ServiceMonitor** オブジェクトを作成します。
- **ServiceMonitor** オブジェクトを検出するための **MonitoringStack** オブジェクトを作成します。

4.1. CLUSTER OBSERVABILITY OPERATOR のサンプルサービスをデプロイする

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

前提条件

- **cluster-admin** クラスターロールを持つユーザーとして、または namespace の管理権限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. **prometheus-coo-example-app.yaml** という名前の YAML ファイルを作成します。このファイルには、namespace、デプロイメント、およびサービスに関する次の設定の詳細が含まれます。

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1-coo
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  labels:
    app: prometheus-coo-example-app
  name: prometheus-coo-example-app
  namespace: ns1-coo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-coo-example-app
  template:
    metadata:
      labels:
        app: prometheus-coo-example-app
    spec:
      containers:
      - image: ghcr.io/rhobs/prometheus-example-app:0.4.2
        imagePullPolicy: IfNotPresent
        name: prometheus-coo-example-app
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: prometheus-coo-example-app
  name: prometheus-coo-example-app
  namespace: ns1-coo
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-coo-example-app
  type: ClusterIP
```

2. ファイルを保存します。
3. 次のコマンドを実行して、設定をクラスターに適用します。

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

4. 次のコマンドを実行して出力を確認し、Pod が実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS    RESTARTS   AGE
prometheus-coo-example-app-0927545cb7-anskj   1/1     Running   0          81m
```

4.2. CLUSTER OBSERVABILITY OPERATOR によるサービスのモニタリング方法の指定

「Cluster Observability Operator のサンプルサービスをデプロイする」セクションで作成したサンプルサービスが公開するメトリクスを使用するには、**/metrics** エンドポイントからメトリクスを取得するようにモニタリングコンポーネントを設定する必要があります。

この設定は、サービスのモニタリング方法を指定する **ServiceMonitor** オブジェクト、または Pod のモニタリング方法を指定する **PodMonitor** オブジェクトを使用して作成できます。**ServiceMonitor** オブジェクトには **Service** オブジェクトが必要です。**PodMonitor** オブジェクトには必要ないため、**MonitoringStack** オブジェクトは Pod が公開するメトリクスエンドポイントから直接メトリクスを取得できます。

この手順は、**ns1-coo** namespace に **prometheus-coo-example-app** という名前のサンプルサービスの **ServiceMonitor** オブジェクトを作成する方法を示しています。

前提条件

- **cluster-admin** クラスターロールを持つユーザーとして、または namespace の管理権限を持つユーザーとして、クラスターにアクセスできる。
- Cluster Observability Operator がインストールされている。
- **prometheus-coo-example-app** サンプルサービスを **ns1-coo** namespace にデプロイしている。

注記

prometheus-example-app サンプルサービスは、TLS 認証をサポートしていません。

手順

1. 次の **ServiceMonitor** オブジェクト設定の詳細を含む YAML ファイルを、**example-coo-app-service-monitor.yaml** という名前で作成します。

```
apiVersion: monitoring.rhobs/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus-coo-example-monitor
  name: prometheus-coo-example-monitor
  namespace: ns1-coo
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-coo-example-app
```

この設定は、**prometheus-coo-example-app** サンプルサービスが公開するメトリクスデータを収集するために **MonitoringStack** オブジェクトが参照する **ServiceMonitor** オブジェクトを定義します。

2. 次のコマンドを実行して、設定をクラスターに適用します。

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

3. 次のコマンドを実行して出力を観察し、**ServiceMonitor** リソースが作成されたことを確認します。

```
$ oc -n ns1-coo get servicemonitors.monitoring.rhobs
```

出力例

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

4.3. CLUSTER OBSERVABILITY OPERATOR への MONITORINGSTACK オブジェクト作成

ターゲット **prometheus-coo-example-app** サービスが公開するメトリクスデータを収集するには、「Cluster Observability Operator でサービスを監視する方法を指定する」セクションで作成した **ServiceMonitor** オブジェクトを参照する **MonitoringStack** オブジェクトを作成します。この **MonitoringStack** オブジェクトはサービスを検出し、そこから公開されているメトリクスデータを収集できます。

前提条件

- **cluster-admin** クラスターロールを持つユーザーとして、または namespace の管理権限を持つユーザーとして、クラスターにアクセスできる。
- Cluster Observability Operator がインストールされている。
- **prometheus-coo-example-app** サンプルサービスを **ns1-coo** namespace にデプロイしている。
- **ns1-coo** namespace に、**prometheus-coo-example-monitor** という名前の **ServiceMonitor** オブジェクトを作成している。

手順

1. **MonitoringStack** オブジェクト設定の YAML ファイルを作成します。この例では、ファイル名を **example-coo-monitoring-stack.yaml** にします。
2. 以下の **MonitoringStack** オブジェクト設定の詳細を追加します。

MonitoringStack オブジェクトの例

```
apiVersion: monitoring.rhobs/v1alpha1
kind: MonitoringStack
metadata:
  name: example-coo-monitoring-stack
  namespace: ns1-coo
spec:
  logLevel: debug
  retention: 1d
```

```
  resourceSelector:
    matchLabels:
      k8s-app: prometheus-coo-example-monitor
```

3. 次のコマンドを実行して、**MonitoringStack** オブジェクトを適用します。

```
$ oc apply -f example-coo-monitoring-stack.yaml
```

4. 次のコマンドを実行し、出力で **MonitoringStack** オブジェクトが利用可能であることを確認します。

```
$ oc -n ns1-coo get monitoringstack
```

出力例

```
NAME                        AGE
example-coo-monitoring-stack   81m
```