



OpenShift Container Platform 4.3

ロギング

OpenShift Container Platform でのクラスターロギングの設定

OpenShift Container Platform 4.3 ロギング

OpenShift Container Platform でのクラスターロギングの設定

法律上の通知

Copyright © 2020 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本書では、クラスターロギング機能のインストール、設定および使用方法について説明します。クラスターロギングは、各種の OpenShift Container Platform サービスについてのログを集計します。

目次

第1章 クラスターロギングおよび OPENSIFT CONTAINER PLATFORM について	4
1.1. クラスターロギング	4
第2章 クラスターロギングのデプロイについて	8
2.1. クラスターロギングのデプロイおよび設定について	8
2.2. クラスターロギングおよび OPENSIFT CONTAINER PLATFORM のストレージについての考慮事項	11
2.3. 追加リソース	12
第3章 クラスターロギングのデプロイ	13
3.1. CLI を使用した ELASTICSEARCH OPERATOR のインストール	13
3.2. WEB コンソールを使用した CLUSTER LOGGING OPERATOR のインストール	16
3.3. CLI を使用した CLUSTER LOGGING OPERATOR のインストール	20
3.4. 追加リソース	24
第4章 クラスターロギングの更新	26
4.1. クラスターロギングの更新	26
第5章 クラスターログの表示	29
5.1. クラスターログの表示	29
5.2. OPENSIFT CONTAINER PLATFORM WEB コンソールでのクラスターログの表示	29
第6章 KIBANA を使用したクラスターログの表示	31
6.1. KIBANA の起動	31
第7章 クラスターロギングデプロイメントの設定	33
7.1. クラスターロギングの設定について	33
7.2. クラスターロギングの管理状態の変更	36
7.3. クラスターロギングの設定	39
7.4. ログデータを保存し、整理するための ELASTICSEARCH の設定	40
7.5. KIBANA の設定	48
7.6. ELASTICSEARCH データのキューレーション	52
7.7. ロギングコレクターの設定	59
7.8. KUBERNETES イベントの収集および保存	63
7.9. 容認を使用した クラスターロギング POD 配置の制御	67
7.10. ログのサードパーティーシステムへの転送	73
7.11. SYSTEMD-JOURNALD および FLUENTD の設定	89
第8章 ELASTICSEARCH ステータスの表示	93
8.1. ELASTICSEARCH ステータスの表示	93
8.2. ELASTICSEARCH コンポーネントのステータスの表示	96
第9章 クラスターロギングのステータス表示	100
9.1. CLUSTER LOGGING OPERATOR のステータス表示	100
9.2. クラスターロギングコンポーネントのステータスの表示	104
第10章 ノードセレクターを使用したクラスターロギングリソースの移動	106
10.1. クラスターロギングリソースの移動	106
第11章 ELASTICSEARCH の手動によるロールアウト	110
11.1. ELASTICSEARCH クラスターのローリング再起動の実行	110
第12章 KIBANA のトラブルシューティング	114
12.1. KUBERNETES ログインループのトラブルシューティング	114
12.2. KIBANA コンソール表示時の KUBERNETES の不明なエラーのトラブルシューティング	114

12.3. KIBANA コンソール表示時の KUBERNETES の 503 エラーのトラブルシューティング	115
第13章 エクスポートされるフィールド	116
13.1. デフォルトのエクスポートされるフィールド	116
13.2. SYSTEMD のエクスポートされるフィールド	131
13.3. KUBERNETES のエクスポートされるフィールド	134
13.4. コンテナのエクスポートされるフィールド	135
13.5. OVIRT のエクスポートされるフィールド	136
13.6. AUSHAPE のエクスポートされるフィールド	136
13.7. TLOG のエクスポートされるフィールド	137
第14章 クラスターロギングのアンインストール	139
14.1. OPENSIFT CONTAINER PLATFORM からのクラスターロギングのアンインストール	139

第1章 クラスターロギングおよび OPENSIFT CONTAINER PLATFORM について

クラスター管理者は、クラスターロギングをデプロイし、ノードシステムログ、アプリケーションコンテナログなどの OpenShift Container Platform クラスターからのすべてのログを集計できます。

1.1. クラスターロギング

OpenShift Container Platform クラスター管理者は、いくつかの CLI コマンドを使用してクラスターロギングをデプロイでき、OpenShift Container Platform Web コンソールを使用して Elasticsearch Operator および Cluster Logging Operator をインストールできます。Operator がインストールされている場合、クラスターロギングのカスタムリソース (Custom Resource、CR) を作成してクラスターロギング Pod およびクラスターロギングのサポートに必要な他のリソースをスケジュールします。Operator はクラスターロギングのデプロイ、アップグレード、および維持を行います。

クラスターロギングは、**instance** という名前のクラスターロギングのカスタムリソース (CR) を変更することで設定できます。CR は、ログを収集し、保存し、視覚化するために必要なロギングスタックのすべてのコンポーネントを含む完全なクラスターロギングデプロイメントを定義します。Cluster Logging Operator は **ClusterLogging** カスタムリソースを監視し、ロギングデプロイメントを適宜調整します。

管理者およびアプリケーション開発者は、表示アクセスを持つプロジェクトのログを表示できます。

1.1.1. クラスターロギングコンポーネント

クラスターロギングコンポーネントは Elasticsearch、Fluentd、Kibana (EFK) に基づいています。コレクターの **Fluentd** は、OpenShift Container Platform クラスターの各ノードにデプロイされます。これはすべてのノードおよびコンテナのログを収集し、それらを **Elasticsearch** (ES) に書き込みます。**Kibana** は、ユーザーおよび管理者が集計されたデータを使って高度な視覚化およびダッシュボードを作成できる中央の Web UI です。

現時点で、5種類のクラスターロギングコンポーネントがあります。

- **logStore**: これはログが保存される場所です。現在の実装は Elasticsearch です。
- **collection**: これは、ノードからログを収集し、それらをフォーマットし、logStore に保存するコンポーネントです。現在の実装は Fluentd です。
- **visualization**: これは、ログ、グラフ、チャートなどを表示するために使用される UI コンポーネントです。現在の実装は Kibana です。
- **curation**: これは期間に基づいてログをトリミングするコンポーネントです。現在の実装は Curator です。

本書では、特筆されない限り、logStore と Elasticsearch、visualization と Kibana、curation と Curator、collection と Fluentd を区別せずに使用することがあります。

1.1.2. ログストアについて

OpenShift Container Platform は **Elasticsearch** (ES) を使用して Fluentd からのログデータを、データストアまたは **インデックス** に編成します。

Elasticsearch は、各インデックスを **シャード** と呼ばれる複数の部分に細分化し、Elasticsearch クラスター内の Elasticsearch ノードセット全体に分散します。Elasticsearch を **レプリカ** というシャードのコピーを作成するように設定できます。さらに、Elasticsearch はレプリカを Elasticsearch ノード全体に分

散します。**ClusterLogging** カスタムリソースにより、カスタムリソース定義 (Custom Resource Definition、CRD) にレプリケーションポリシーを指定して、データの冗長性および耐障害性を提供することができます。

クラスターロギング Elasticsearch インスタンスは、短期 (約 7 日間) の保存について最適化され、テストされています。長期間ログを保持する必要がある場合は、データをサードパーティーのストレージシステムに移動することが推奨されます。



注記

インデックステンプレートのプライマリーシャードの数は Elasticsearch データノードの数と等しくなります。

Cluster Logging Operator および Elasticsearch Operator は、各 Elasticsearch ノードが独自のストレージボリュームを含む一意の Deployment を使用してデプロイされるようにします。クラスターロギングのカスタムリソース (CR) を使用して Elasticsearch ノードの数を増やすことができます。以下に示すように、ストレージおよびネットワークの場所を選択する際の留意事項については、[Elastic のドキュメント](#) を参照してください。



注記

可用性の高い Elasticsearch 環境には 3 つ以上の Elasticsearch ノードが必要で、それぞれが別のホストに置かれる必要があります。

Elasticsearch インデックスに適用されているロールベースアクセス制御 (RBAC) は、開発者のログの制御アクセスを可能にします。**project.{project_name}.{project_uuid}.*** 形式を使用したインデックスへのアクセスは、特定プロジェクトのユーザーのパーミッションに基づいて制限されます。

詳細は「[Elasticsearch \(ES\)](#)」を参照してください。

1.1.3. ロギングコレクターについて

OpenShift Container Platform は Fluentd を使用してクラスターについてのデータを収集します。

ロギングコレクターは、Pod を各 OpenShift Container Platform ノードにデプロイする OpenShift Container Platform の daemonset としてデプロイされます。**journald** は、オペレーティングシステム、コンテナランタイム、および OpenShift Container Platform からのログメッセージを提供するシステムログソースです。

コンテナランタイムは、プロジェクト、Pod 名、およびコンテナ ID などのログメッセージのソースを特定するための最小限の情報を提供します。この情報だけでは、ログのソースを一意に特定することはできません。ログコレクターがログを処理する前に、指定された名前およびプロジェクトを持つ Pod が削除される場合、ラベルやアノテーションなどの API サーバーからの情報は利用できない可能性があります。そのため、似たような名前の Pod やプロジェクトからログメッセージを区別したり、ログのソースを追跡することができない場合があります。この制限により、ログの収集および正規化はベストエフォート ベースであると見なされます。



重要

利用可能なコンテナランタイムは、ログメッセージのソースを特定するための最小限の情報を提供し、個別のログメッセージが一意となる確証はなく、これらのメッセージにより、そのソースを追跡できる訳ではありません。

詳細情報は、「[Fluentd](#)」を参照してください。

1.1.4. ロギングの視覚化について

OpenShift Container Platform は Kibana を使用して、Fluentd によって収集され、Elasticsearch によってインデックス化されるログデータを表示します。

Kibana は、ヒストグラム、線グラフ、円グラフ、ヒートマップ、ビルトインされた地理空間サポート、その他の視覚化機能を使用して Elasticsearch データをクエリーし、検出し、視覚化するためのブラウザベースのコンソールインターフェースです。

詳細は、「[Kibana](#)」を参照してください。

1.1.5. ロギング curation について

Elasticsearch Curator ツールは、グローバルに、またはプロジェクトごとにスケジュールされたメンテナンス操作を実行します。Curator はその設定に基づいて動作を実行します。Elasticsearch クラスターあたり 1 つの Curator Pod のみの使用が推奨されます。

```
spec:
  curation:
    type: "curator"
  resources:
    curator:
      schedule: "30 3 * * *" 1
```

1 Curator スケジュールを [cron 形式](#) で指定します。

詳細は「[Curator](#)」を参照してください。

1.1.6. イベントのルーティングについて

イベントルーターは、クラスターロギングで収集できるように OpenShift Container Platform イベントを監視する Pod です。イベントルーターはすべてのプロジェクトからイベントを収集し、それらを **STDOUT** に書き込みます。Fluentd はそれらのイベントを収集し、それらを OpenShift Container Platform Elasticsearch インスタンスに転送します。Elasticsearch はイベントを **infra** インデックスにインデックス化します。

イベントルーターは手動でデプロイする必要があります。

1.1.7. クラスターロギングカスタムリソースについて

クラスターロギングデプロイメントを変更するには、クラスターロギングカスタムリソース (CR) を作成し、変更します。CR の作成または変更方法については、このドキュメントで適宜説明されます。

以下は、クラスターロギングの通常のクラスターリソースの例です。

クラスターロギング CR の例

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
```

```
logStore:
  type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
    resources:
      limits:
        memory: 16Gi
      requests:
        cpu: 500m
        memory: 16Gi
    storage:
      storageClassName: "gp2"
      size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
    proxy:
      resources:
        limits:
          memory: 100Mi
        requests:
          cpu: 100m
          memory: 100Mi
    replicas: 2
  curation:
    type: "curator"
    curator:
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 200m
          memory: 200Mi
    schedule: "*/10 * * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd:
        resources:
          limits:
            memory: 1Gi
          requests:
            cpu: 200m
            memory: 1Gi
```

第2章 クラスターロギングのデプロイについて

クラスターロギングを OpenShift Container Platform クラスターにインストールする前に、以下のセクションを確認します。

2.1. クラスターロギングのデプロイおよび設定について

OpenShift Container Platform クラスターロギングは、小規模および中規模の OpenShift Container Platform クラスター用に調整されたデフォルト設定で使用されるように設計されています。

以下のインストール方法には、サンプルのクラスターロギングのカスタムリソース (CR) が含まれます。これを使用して、クラスターロギングインスタンスを作成し、クラスターロギングのデプロイメントを設定することができます。

デフォルトのクラスターロギングインストールを使用する必要がある場合は、サンプル CR を直接使用できます。

デプロイメントをカスタマイズする必要がある場合、必要に応じてサンプル CR に変更を加えます。以下では、クラスターロギングのインスタンスをインストール時に実行し、インストール後に変更する設定について説明します。クラスターロギングのカスタムリソース外で加える変更を含む、各コンポーネントの使用方法については、設定についてのセクションを参照してください。

2.1.1. クラスターロギングの設定およびチューニング

クラスターロギング環境は、**openshift-logging** プロジェクトにデプロイされるクラスターロギングのカスタムリソースを変更することによって設定できます。

インストール時またはインストール後に、以下のコンポーネントのいずれかを変更することができます。

メモリーおよび CPU

resources ブロックを有効なメモリーおよび CPU 値で変更することにより、各コンポーネントの CPU およびメモリーの両方の制限を調整することができます。

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
            memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
        resources:
          limits:
            cpu:
              memory:
            requests:
              cpu:
                memory:
```

```

    type: "fluentd"
  visualization:
    kibana:
      resources:
        limits:
          cpu:
          memory:
        requests:
          cpu:
          memory:
      type: kibana
  curation:
    curator:
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 200m
          memory: 200Mi
      type: "curator"

```

Elasticsearch ストレージ

storageClass name および **size** パラメーターを使用し、Elasticsearch クラスターの永続ストレージのクラスおよびサイズを設定できます。Cluster Logging Operator は、これらのパラメーターに基づいて、Elasticsearch クラスターの各データノードについて **PersistentVolumeClaim** を作成します。

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
    storage:
      storageClassName: "gp2"
      size: "200G"

```

この例では、クラスターの各データノードが「gp2」ストレージの「200G」を要求する **PersistentVolumeClaim** にバインドされるように指定します。それぞれのプライマリーシャードは単一のレプリカによってサポートされます。

注記

storage ブロックを省略すると、一時ストレージのみを含むデプロイメントになります。

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
    storage: {}

```

Elasticsearch レプリケーションポリシー

Elasticsearch シャードをクラスター内のデータノードにレプリケートする方法を定義するポリシーを設定できます。

- **FullRedundancy**:各インデックスのシャードはすべてのデータノードに完全にレプリケートされます。
- **MultipleRedundancy**:各インデックスのシャードはデータノードの半分に分散します。
- **SingleRedundancy**:各シャードの単一コピー。2つ以上のデータノードが存在する限り、ログは常に利用可能かつ回復可能です。
- **ZeroRedundancy**:シャードのコピーはありません。ログは、ノードの停止または失敗時に利用不可になる(または失われる)可能性があります。

Curator スケジュール

Curator のスケジュールを [cron 形式](#) で指定します。

```
spec:
  curation:
    type: "curator"
  resources:
    curator:
      schedule: "30 3 * * *"
```

2.1.2. 変更されたクラスターロギングカスタムリソースのサンプル

以下は、前述のオプションを使用して変更されたクラスターロギングのカスタムリソースの例です。

変更されたクラスターロギングカスタムリソースのサンプル

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
  resources:
    limits:
      memory: 32Gi
    requests:
      cpu: 3
      memory: 32Gi
  storage: {}
  redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    resources:
      limits:
```

```

    memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
    replicas: 1
  curation:
    type: "curator"
  curator:
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 200Mi
    schedule: "*/5 * * * *"
  collection:
    logs:
      type: "fluentd"
    fluentd:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 200m
          memory: 1Gi

```

2.2. クラスターロギングおよび OPENSIFT CONTAINER PLATFORM のストレージについての考慮事項

永続ボリュームは、それぞれの Elasticsearch デプロイメントに1つのデータノードに対して1つのデータボリュームを持たせるために必要です。OpenShift Container Platform では、これは Persistent Volume Claim (永続ボリューム要求、PVC) を使用して実行されます。

Elasticsearch Operator は Elasticsearch リソース名を使って PVC に名前を付けます。詳細は、永続 Elasticsearch ストレージを参照してください。

Fluentd は **systemd ジャーナル** および `/var/log/containers/` から Elasticsearch にログを送信します。

このため、必要なデータ量とアプリケーションログデータの集計方法を事前に検討しておく必要があります。一部の Elasticsearch ユーザーは、**絶対的なストレージ使用率をおよそ 50% に維持し、常に 70% 未満にする必要があることを確認** しています。これは、大規模なマージ操作を実行する際に Elasticsearch が応答しなくなる状態を避けるのに役立ちます。

デフォルトでは、85% になると Elasticsearch は新規データのノードへの割り当てを停止し、90% になると Elasticsearch は可能な場合に、該当ノードから別のノードへ既存シャードの移動を試行します。ただし、85% 未満の空き容量を持つノードがない場合、Elasticsearch は新規インデックスの作成を拒否し、ステータスは RED になります。



注記

これらの基準値 (高い値および低い値を含む) は現行リリースにおける Elasticsearch のデフォルト値です。これらの値を変更することはできますが、いずれの変更もアラートにも適用する必要があります。アラートはこれらのデフォルト値に基づくものです。

2.3. 追加リソース

Operator のインストールについての詳細は、[「Installing Operators from the OperatorHub」](#) を参照してください。

第3章 クラスターロギングのデプロイ

クラスターロギングは、Elasticsearch および Cluster Logging Operator をデプロイしてインストールできます。Elasticsearch Operator は、クラスターロギングによって使用される Elasticsearch クラスターを作成し、管理します。Cluster Logging Operator はロギングスタックのコンポーネントを作成し、管理します。

クラスターロギングを OpenShift Container Platform にデプロイするプロセスには以下が関係します。

- 「[クラスターロギングのデプロイについて](#)」でインストールオプションを確認します。
- [クラスターロギングストレージについての考慮事項](#)を確認します。
- Elasticsearch Operator および Cluster Logging Operator をインストールします。

3.1. CLI を使用した ELASTICSEARCH OPERATOR のインストール

以下の指示に従って CLI を使い、Elasticsearch Operator のインストールを **実行する必要があります**。

前提条件

Elasticsearch の必要な永続ストレージがあることを確認します。各 Elasticsearch ノードには独自のストレージボリュームが必要であることに注意してください。

Elasticsearch はメモリー集約型アプリケーションです。デフォルトで、OpenShift Container Platform はメモリー要求および 16 GB の制限を持つ 3 つの Elasticsearch ノードをインストールします。OpenShift Container Platform ノードの最初の 3 つのセットには、Elasticsearch をクラスター内で実行するのに十分なメモリーがない可能性があります。Elasticsearch に関連するメモリーの問題が発生した場合、既存ノードのメモリーを増やすのではなく、Elasticsearch ノードをクラスターにさらに追加する必要があります。

手順

CLI を使用して Elasticsearch Operator をインストールするには、以下を実行します。

1. Elasticsearch Operator の namespace を作成します。
 - a. Elasticsearch Operator の namespace オブジェクト YAML ファイル (`eo-namespace.yaml` など) を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-operators-redhat 1
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true" 2
```

- 1** **openshift-operators-redhat** namespace を指定する必要があります。メトリクスとの競合が発生する可能性を防ぐには、Prometheus のクラスターモニタリングスタックを、**openshift-operators** namespace からではなく、**openshift-operators-redhat** namespace からメトリクスを収集するように設定する必要があります。**openshift-operators** namespace には信頼されていないコミュニティー Operator が含まれる可能性があり、OpenShift Container Platform メトリクスと同じ名前でもトリクスを公開する可能性があるため、これによって競合が生じる可能性があります。

- 2 クラスターモニタリングが **openshift-operators-redhat** namespace を収集できるように、このラベルを上記のように指定する必要があります。

b. namespace を作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f eo-namespace.yaml
```

2. 以下のオブジェクトを作成して Elasticsearch Operator をインストールします。

- a. Elasticsearch Operator の Operator グループオブジェクトの YAML ファイル (**eo-og.yaml** など) を作成します。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-operators-redhat
  namespace: openshift-operators-redhat 1
spec: {}
```

- 1 **openshift-operators-redhat** namespace を指定する必要があります。

b. Operator グループオブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f eo-og.yaml
```

- c. Subscription オブジェクト YAML ファイル(**eo-sub.yaml** など) を作成し、namespace を Operator にサブスクライブします。

サブスクリプションの例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat" 1
spec:
  channel: "4.3" 2
  installPlanApproval: "Automatic"
  source: "redhat-operators" 3
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

- 1 **openshift-operators-redhat** namespace を指定する必要があります。

- 2 4.3 をチャンネルとして指定します。
- 3 **redhat-operators** を指定します。OpenShift Container Platform クラスターが、非接続クラスターとも呼ばれるネットワークが制限された環境でインストールされている場合、Operator Lifecycle Manager (OLM) の設定時に作成される CatalogSource オブジェクトの名前を指定します。

d. Subscription オブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f eo-sub.yaml
```

e. **openshift-operators-redhat** プロジェクトに切り替えます。

```
$ oc project openshift-operators-redhat
```

```
Now using project "openshift-operators-redhat"
```

f. ロールベースアクセス制御 (RBAC) オブジェクトファイル (**eo-rbac.yaml** など) を使用して、**openshift-operators-redhat** namespace にアクセスするための Prometheus パーミッションを付与します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s
  namespace: openshift-operators-redhat
rules:
- apiGroups:
  - ""
  resources:
  - services
  - endpoints
  - pods
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prometheus-k8s
  namespace: openshift-operators-redhat
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: prometheus-k8s
subjects:
```

```
- kind: ServiceAccount
  name: prometheus-k8s
  namespace: openshift-operators-redhat
```

- g. RBAC オブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f eo-rbac.yaml
```

Elasticsearch Operator は **openshift-operators-redhat** namespace にインストールされ、クラスター内の各プロジェクトにコピーされます。

3. Operator のインストールを確認します。

```
oc get csv --all-namespaces
```

NAMESPACE	VERSION	REPLACES	PHASE	NAME	DISPLAY
default				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
kube-node-lease				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
kube-public				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
kube-system				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
openshift-apiserver-operator				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
openshift-apiserver				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
openshift-authentication-operator				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
openshift-authentication				elasticsearch-operator.4.3.1-202002032140	
Elasticsearch Operator	4.3.1-202002032140		Succeeded		
...					

それぞれの namespace には Elasticsearch Operator がなければなりません。バージョン番号が表示されるものと異なる場合があります。

次のステップ

以下のセクションの手順に従って、コンソールまたは CLI を使用して Cluster Logging Operator をインストールします。

3.2. WEB コンソールを使用した CLUSTER LOGGING OPERATOR のインストール

OpenShift Container Platform Web コンソールを使って Cluster Logging Operator をインストールすることができます。



注記

Web コンソールまたは **oc new-project** コマンドを使用して、**openshift-**で始まる Project を作成することはできません。YAML オブジェクトファイルを使用して namespace を作成し、以下のように **oc create -f <file-name>.yaml** コマンドを実行する必要があります。

手順

OpenShift Container Platform Web コンソールを使って Cluster Logging Operator をインストールするには、以下を実行します。

1. Cluster Logging Operator の namespace を作成します。CLI を使用して namespace を作成する必要があります。
 - a. Cluster Logging Operator の namespace オブジェクト YAML ファイル (**clo-namespace.yaml** など) を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging 1
  annotations:
    openshift.io/node-selector: "" 2
  labels:
    openshift.io/cluster-monitoring: "true" 3
```

1 2 3 これらのラベルを以下のように指定します。

- b. namespace を作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f clo-namespace.yaml
```

2. Cluster Logging Operator をインストールします。
 - a. OpenShift Container Platform Web コンソールで、**Operators** → **OperatorHub** をクリックします。
 - b. 利用可能な Operator の一覧から **Cluster Logging** を選択し、**Install** をクリックします。
 - c. **Create Operator Subscription** ページの **A specific Namespace on the cluster** の下で **openshift-logging** を選択します。次に、**Subscribe** をクリックします。
3. Cluster Logging Operator がインストールされていることを確認します。
 - a. **Operators** → **Installed Operators** ページに切り替えます。
 - b. **Cluster Logging** が **Status** が **InstallSucceeded** の状態で **openshift-logging** プロジェクトに一覧表示されていることを確認します。



注記

インストール時に、Operator は **Failed** ステータスを表示する可能性があります。その後、Operator が **InstallSucceeded** メッセージと共にインストールされる場合、**Failed** メッセージを安全に無視することができます。

Operator がインストール済みとして表示されない場合に、さらにトラブルシューティングを実行します。

- **Operators** → **Installed Operators** ページに切り替え、**Status** 列でエラーまたは失敗の有無を確認します。
- **Workloads** → **Pods** ページに切り替え、**openshift-logging** および **openshift-operators-redhat** プロジェクトの Pod で問題を報告しているログの有無を確認します。

4. クラスターロギングのインスタンスを作成します。

- Administration** → **Custom Resource Definitions** ページに切り替えます。
- Custom Resource Definitions** ページで、**ClusterLogging** をクリックします。
- Custom Resource Definition Overview** ページで、**Actions** メニューから **View Instances** を選択します。
- Cluster Loggings** ページで、**Create Cluster Logging** をクリックします。
データを読み込むためにページを更新する必要がある場合があります。
- YAML フィールドで、コードを以下に置き換えます。



注記

このデフォルトのクラスターロギング設定は各種の環境をサポートすることが予想されます。クラスターロギングのクラスターに加えることのできる変更についての詳細は、クラスターロギングコンポーネントのチューニングおよび設定についてのトピックを確認してください。

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" ①
  namespace: "openshift-logging"
spec:
  managementState: "Managed" ②
  logStore:
    type: "elasticsearch" ③
  elasticsearch:
    nodeCount: 3 ④
    storage:
      storageClassName: "<storage-class-name>" ⑤
      size: 200G
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" ⑥
```

```

kibana:
  replicas: 1
curation:
  type: "curator" 7
  curator:
    schedule: "30 3 * * *"
collection:
logs:
  type: "fluentd" 8
  fluentd: {}

```

- 1 名前は **instance** である必要があります。
- 2 クラスターロギングの管理状態。ほとんどの場合、クラスターロギングのデフォルト値を変更する場合は、これを **Unmanaged** (管理外) に設定する必要があります。ただし、管理外のデプロイメントはクラスターロギングが管理対象の状態に戻されるまで更新を受信しません。詳細は「[クラスターロギングの管理状態の変更](#)」を参照してください。
- 3 Elasticsearch の設定に必要な設定。CR を使用してシャードのレプリケーションポリシーおよび永続ストレージを設定できます。詳細は「[Elasticsearch の設定](#)」を参照してください。
- 4 Elasticsearch ノードの数を指定します。この一覧に続く注記を確認してください。
- 5 Elasticsearch ストレージの既存の StorageClass の名前を入力します。最適なパフォーマンスを得るには、ブロックストレージを割り当てる StorageClass を指定します。
- 6 Kibana の設定に必要な設定。CR を使用して、冗長性を確保するために Kibana をスケリングし、Kibana ノードの CPU およびメモリーを設定できます。詳細は「[Kibana の設定](#)」を参照してください。
- 7 Curator の設定に必要な設定。CR を使用して Curator スケジュールを設定できます。詳細は「[Curator の設定](#)」を参照してください。
- 8 Fluentd の設定に必要な設定。CR を使用して Fluentd の CPU およびメモリー制限を設定できます。詳細は「[Fluentd の設定](#)」を参照してください。



注記

Elasticsearch マスターノードの最大数は 3 です。3 を超える **nodeCount** を指定する場合は、OpenShift Container Platform は、マスター、クライアントおよびデータロールを使用して、3 つのマスターとしての適格性のあるノードである Elasticsearch ノードを作成します。追加の Elasticsearch ノードは、クライアントおよびデータロールを使用してデータのみのノードとして作成されます。マスターノードは、インデックスの作成および削除、シャードの割り当て、およびノードの追跡などのクラスター全体でのアクションを実行します。データノードはシャードを保持し、CRUD、検索、および集計などのデータ関連の操作を実行します。データ関連の操作は、I/O、メモリーおよび CPU 集約型の操作です。これらのリソースを監視し、現行ノードがオーバーロードする場合にデータノード追加することが重要です。

たとえば、**nodeCount=4** の場合に、以下のノードが作成されます。

```
$ oc get deployment
```

```
cluster-logging-operator 1/1 1 1 18h
elasticsearch-cd-x6kdekli-1 0/1 1 0 6m54s
elasticsearch-cdm-x6kdekli-1 1/1 1 1 18h
elasticsearch-cdm-x6kdekli-2 0/1 1 0 6m49s
elasticsearch-cdm-x6kdekli-3 0/1 1 0 6m44s
```

インデックステンプレートのプライマリーシャードの数は Elasticsearch データノードの数と等しくなります。

- f. **Create** をクリックします。これにより、クラスターロギングのカスタムリソースおよび Elasticsearch のカスタムリソースが作成されます。これらを編集すると、クラスターロギングのクラスターに変更を加えることができます。
5. インストールを確認します。
 - a. **Workloads** → **Pods** ページに切り替えます。
 - b. **openshift-logging** プロジェクトを選択します。
以下の一覧のようなクラスターロギング、Elasticsearch、Fluentd、および Kibana のいくつかの Pod が表示されるはずです。
 - cluster-logging-operator-cb795f8dc-xkckc
 - elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
 - elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv
 - elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
 - fluentd-2c7dg
 - fluentd-9z7kk
 - fluentd-br7r2
 - fluentd-fn2sb
 - fluentd-pb2f8
 - fluentd-zqgqx
 - kibana-7fb4fd4cc9-bvt4p

3.3. CLI を使用した CLUSTER LOGGING OPERATOR のインストール

OpenShift Container Platform CLI を使用して、Cluster Logging Operator をインストールできます。Cluster Logging Operator はロギングスタックのコンポーネントを作成し、管理します。

手順

CLI を使用して Cluster Logging Operator をインストールするには、以下を実行します。

1. Cluster Logging Operator の namespace を作成します。
 - a. Cluster Logging Operator の namespace オブジェクト YAML ファイル (`cluster-logging-operator.yaml`) を作成します。

namespace.yaml (下記) を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true"
```

b. namespace を作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f clo-namespace.yaml
```

2. 以下のオブジェクトを作成して Elasticsearch Logging Operator をインストールします。

a. Cluster Logging Operator の OperatorGroup オブジェクトの YAML ファイル (**eo-og.yaml** など) を作成します。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: cluster-logging
  namespace: openshift-logging 1
spec:
  targetNamespaces:
    - openshift-logging 2
```

1 **2** **openshift-logging** namespace を指定する必要があります。

b. OperatorGroup オブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f clo-og.yaml
```

c. Subscription オブジェクト YAML ファイル (**clo-sub.yaml** など) を作成し、namespace を Operator にサブスクライブします。

サブスクリプションの例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging 1
```

```
spec:
  channel: "4.3" 2
  name: cluster-logging
  source: redhat-operators 3
  sourceNamespace: openshift-marketplace
```

- 1 **openshift-logging** namespace を指定する必要があります。
- 2 **4.3** をチャンネルとして指定します。
- 3 **redhat-operators** を指定します。OpenShift Container Platform クラスターが、非接続クラスターとも呼ばれる制限されたネットワークにインストールされている場合、Operator Lifecycle Manager (OLM) の設定時に作成した CatalogSource オブジェクトの名前を指定します。

d. Subscription オブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f clo-sub.yaml
```

Cluster Logging Operator は **openshift-logging** namespace にインストールされます。

3. Operator のインストールを確認します。

openshift-logging namespace には Cluster Logging Operator がなければなりません。バージョン番号が表示されるものと異なる場合があります。

```
oc get csv --all-namespaces
```

NAMESPACE	VERSION	REPLACES	PHASE	NAME	DISPLAY
...					
openshift-logging	Logging	4.3.1-202002032140	Succeeded	clusterlogging.4.3.1-202002032140	Cluster
...					

4. クラスターロギングのインスタンスを作成します。

a. Cluster Logging Operator のインスタンスオブジェクト YAML ファイル (**clo-instance.yaml** など) を作成します。



注記

このデフォルトのクラスターロギング設定は各種の環境をサポートすることが予想されます。クラスターロギングのクラスターに加えることのできる変更についての詳細は、クラスターロギングコンポーネントのチューニングおよび設定についてのトピックを確認してください。

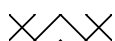
```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
```

```

name: "instance" ❶
namespace: "openshift-logging"
spec:
  managementState: "Managed" ❷
  logStore:
    type: "elasticsearch" ❸
    elasticsearch:
      nodeCount: 3 ❹
      storage:
        storageClassName: "<storage-class-name>" ❺
        size: 200G
        redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" ❻
    kibana:
      replicas: 1
  curation:
    type: "curator" ❼
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd" ❽
      fluentd: {}

```

- ❶ 名前は **instance** である必要があります。
- ❷ クラスターロギングの管理状態。ほとんどの場合、クラスターロギングのデフォルト値を変更する場合は、これを **Unmanaged** (管理外) に設定する必要があります。ただし、管理外のデプロイメントはクラスターロギングが **Managed** の状態に戻されるまで更新を受信しません。詳細は「[クラスターロギングの管理状態の変更](#)」を参照してください。
- ❸ Elasticsearch の設定に必要な設定。CR (Custom Resource) を使用してシャードのレプリケーションポリシーおよび永続ストレージを設定できます。詳細は「[Elasticsearch の設定](#)」を参照してください。
- ❹ Elasticsearch ノードの数を指定します。この一覧に続く注記を確認してください。
- ❺ Elasticsearch ストレージの既存の StorageClass の名前を入力します。最適なパフォーマンスを得るには、ブロックストレージを割り当てる StorageClass を指定します。
- ❻ Kibana の設定に必要な設定。CR を使用して、冗長性を確保するために Kibana をスケーリングし、Kibana ノードの CPU およびメモリーを設定できます。詳細は「[Kibana の設定](#)」を参照してください。
- ❼ Curator の設定に必要な設定。CR を使用して Curator スケジュールを設定できます。詳細は「[Curator の設定](#)」を参照してください。
- ❽ Fluentd の設定に必要な設定。CR を使用して Fluentd の CPU およびメモリー制限を設定できます。詳細は「[Fluentd の設定](#)」を参照してください。



注記

Elasticsearch マスターノードの最大数は 3 です。3 を超える **nodeCount** を指定する場合、OpenShift Container Platform は、マスター、クライアントおよびデータロールを使用して、3 つのマスターとしての適格性のあるノードである Elasticsearch ノードを作成します。追加の Elasticsearch ノードは、クライアントおよびデータロールを使用してデータのみノードとして作成されます。マスターノードは、インデックスの作成および削除、シャードの割り当て、およびノードの追跡などのクラスター全体でのアクションを実行します。データノードはシャードを保持し、CRUD、検索、および集計などのデータ関連の操作を実行します。データ関連の操作は、I/O、メモリーおよび CPU 集約型の操作です。これらのリソースを監視し、現行ノードがオーバーロードする場合にデータノード追加することが重要です。

たとえば、**nodeCount=4** の場合に、以下のノードが作成されます。

```
$ oc get deployment
```

```
cluster-logging-operator 1/1 1 1 18h
elasticsearch-cd-x6kdekli-1 1/1 1 0 6m54s
elasticsearch-cdm-x6kdekli-1 1/1 1 1 18h
elasticsearch-cdm-x6kdekli-2 1/1 1 0 6m49s
elasticsearch-cdm-x6kdekli-3 1/1 1 0 6m44s
```

インデックステンプレートのプライマリーシャードの数は Elasticsearch データノードの数と等しくなります。

- b. インスタンスを作成します。

```
$ oc create -f <file-name>.yaml
```

以下は例になります。

```
$ oc create -f clo-instance.yaml
```

5. **openshift-logging** プロジェクトに Pod を一覧表示して、インストールを検証します。以下の一覧のようなクラスターロギング、Elasticsearch、Fluentd、および Kibana のいくつかの Pod が表示されるはずですが、

```
oc get pods -n openshift-logging
```

```
NAME                                READY STATUS RESTARTS AGE
cluster-logging-operator-66f77fccb-ppzbg 1/1 Running 0 7m
elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp 2/2 Running 0 2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc 2/2 Running 0 2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2 2/2 Running 0 2m4s
fluentd-587vb                          1/1 Running 0 2m26s
fluentd-7mpb9                           1/1 Running 0 2m30s
fluentd-flm6j                           1/1 Running 0 2m33s
fluentd-gn4rn                           1/1 Running 0 2m26s
fluentd-nlgb6                           1/1 Running 0 2m30s
fluentd-snpkt                           1/1 Running 0 2m28s
kibana-d6d5668c5-rppqm                  2/2 Running 0 2m39s
```

3.4. 追加リソース

Operator のインストールについての詳細は、「[Installing Operators from the OperatorHub](#)」を参照してください。

第4章 クラスターロギングの更新

OpenShift Container Platform クラスターを 4.2 から 4.3 に更新したら、クラスターロギングを 4.2 から 4.3 にアップグレードする必要があります。

4.1. クラスターロギングの更新

OpenShift Container Platform クラスターの更新後に、Elasticsearch Operator および Cluster Logging Operator のサブスクリプションを更新して、クラスターロギングを 4.2 から 4.3 に更新できます。



重要

新規のログ転送機能によって導入された変更により、OpenShift Container Platform 4.3 リリース以降の `out_forward` のサポートが変更されました。OpenShift Container Platform 4.3 では、ConfigMap を作成して `out_forward` を設定します。Fluentd ConfigMap の `secure-forward.conf` セクションに対するすべての更新は削除されます。

更新する前に `out_forward` プラグインを使用する場合、Fluentd ConfigMap から現在の `secure-forward.conf` セクションをコピーし、`secure-forward` ConfigMap の作成時にコピーされるデータを使用できます。

前提条件

- クラスターを 4.2 から 4.3 に更新します。
- クラスターロギングのステータスが正常であることを確認します。
 - すべての Pod が **Ready** 状態にある。
 - Elasticsearch クラスターが正常である。
- オプションで、`secure-forward` ConfigMap を作成する必要がある場合に Fluentd ConfigMap から現在の `secure-forward.conf` セクションをコピーします。上記の注記を参照してください。

手順

1. Elasticsearch Operator を更新します。
 - a. Web コンソールで **Operators** → **Installed Operators** をクリックします。
 - b. **openshift-logging** プロジェクトを選択します。
 - c. **Elasticsearch Operator** をクリックします。
 - d. **Subscription** → **Channel** をクリックします。
 - e. **Change Subscription Update Channel** ウィンドウで 4.3 を選択し、**Save** をクリックします。
 - f. 数秒待ってから **Operators** → **Installed Operators** をクリックします。Elasticsearch Operator が 4.3 と表示されます。以下は例になります。

```
Elasticsearch Operator
4.3.0-201909201915 provided
by Red Hat, Inc
```

2. Cluster Logging Operator を更新します。
 - a. Web コンソールで **Operators** → **Installed Operators** をクリックします。
 - b. **openshift-logging** プロジェクトを選択します。
 - c. **Cluster Logging Operator** をクリックします。
 - d. **Subscription** → **Channel** をクリックします。
 - e. **Change Subscription Update Channel** ウィンドウで **4.3** を選択し、**Save** をクリックします。
 - f. 数秒待ってから **Operators** → **Installed Operators** をクリックします。
Cluster Logging Operator は 4.3 として表示されます。以下は例になります。

```
Cluster Logging
4.3.0-201909201915 provided
by Red Hat, Inc
```

3. ロギングコンポーネントを確認します。
 - a. Elasticsearch Pod が 4.3 イメージを使用していることを確認します。

```
$ oc get pod -o yaml -n openshift-logging --selector component=elasticsearch |grep
'image:'
```

```
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
```

- b. すべての Elasticsearch Pod が **Ready** ステータスであることを確認します。

```
$ oc get pod -n openshift-logging --selector component=elasticsearch
```

```
NAME                                READY STATUS RESTARTS AGE
elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk 2/2   Running 0      31m
elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk 2/2   Running 0      30m
elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc 2/2   Running 0      29m
```

- c. Elasticsearch クラスターが正常であることを確認します。

```
oc exec -n openshift-logging -c elasticsearch elasticsearch-cdm-1pbrl44l-1-55b7546f4c-
mshhk -- es_cluster_health

{
  "cluster_name" : "elasticsearch",
  "status" : "green",
  ....
}
```

- d. ログインコレクター Pod が 4.3 イメージを使用していることを確認します。

```
$ oc get pod -n openshift-logging --selector logging-infra=fluentd -o yaml |grep 'image:'

image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
```

- e. Kibana Pod が 4.3 イメージを使用していることを確認します。

```
$ oc get pod -n openshift-logging --selector logging-infra=kibana -o yaml |grep 'image:'

image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
```

- f. Curator CronJob が 4.3 イメージを使用していることを確認します。

```
$$ oc get CronJob curator -n openshift-logging -o yaml |grep 'image:'

image: registry.redhat.io/openshift4/ose-logging-curator5:v4.3.0-202001081344
```


第5章 クラスターログの表示

CLI または OpenShift Container Platform Web コンソールで OpenShift Container Platform クラスターのログを表示できます。

5.1. クラスターログの表示

CLI でクラスターのログを表示できます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

クラスターログを表示するには、以下を実行します。

oc logs [-f] <pod_name> コマンドを使用します。ここで、**-f** はオプションになります。

```
$ oc logs -f <any-fluentd-pod> -n openshift-logging 1
```

- 1** ログコレクター Pod の名前を指定します。**-f** オプションを使用してログに書き込まれている内容をフォローします。

以下は例になります。

```
$ oc logs -f fluentd-ht42r -n openshift-logging
```

ログファイルの内容が出力されます。

デフォルトで、Fluentd はログの末尾または終了部分からログを読み取ります。

5.2. OPENSIFT CONTAINER PLATFORM WEB コンソールでのクラスターログの表示

OpenShift Container Platform Web コンソールでクラスターのログを表示できます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

クラスターログを表示するには、以下を実行します。

1. OpenShift Container Platform コンソールで **Workloads** → **Pods** に移動します。
2. ドロップダウンメニューから **openshift-logging** プロジェクトを選択します。
3. **fluentd** プレフィックスを使ってロギングコレクター Pod のいずれかをクリックします。
4. **Logs** をクリックします。

デフォルトで、Fluentd はログの末尾または終了部分からログを読み取ります。

第6章 KIBANA を使用したクラスターログの表示

クラスターロギングのインストールは Kibana Web コンソールをデプロイします。

6.1. KIBANA の起動

Kibana は、ヒストグラム、線グラフ、円グラフ、ヒートマップ、ビルトインされた地理空間サポート、その他の可視化機能を使用してログをクエリーし、検出し、可視化するためのブラウザベースのコンソールです。

前提条件

プロキシを使用して OpenShift Container Platform をインストールしている場合、**.apps.<cluster_name>.<base_domain>** をクラスター全体のプロキシオブジェクトの **noProxy** 一覧に追加する必要があります。

以下は例になります。

```
$ oc edit proxy/cluster

apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  creationTimestamp: "2020-03-30T00:45:44Z"
  generation: 3
  name: cluster
  resourceVersion: "26654"
  selfLink: /apis/config.openshift.io/v1/proxies/cluster
  uid: 2213b41b-0721-4c9f-9586-0678c0058f85
spec:
  httpProxy: http://proxy.com
  httpsProxy: https://proxy.com
  noProxy: .apps.mycluster.example.com ①
  trustedCA:
    name: user-ca-bundle
```

① **.apps.<cluster_name>.<base_domain>** を **noProxy** 一覧に追加します。これは、プロキシを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のカンマ区切りの一覧です。

手順

Kibana を起動するには、以下を実行します。

1. OpenShift Container Platform コンソールで、**Monitoring** → **Logging** をクリックします。
2. OpenShift Container Platform コンソールにログインするために使用するものと同じ認証情報を使用してログインします。
Kibana インターフェースが起動します。以下を実行することができます。
 - Discover ページを使用してデータを検索し、参照します。
 - Visualize ページを使用してデータのグラフを表示し、データをマップします。
 - Dashboard ページを使用してカスタムダッシュボードを作成し、表示します。

Kibana インターフェースの使用および設定については、本書では扱いません。詳細については、[Kibana のドキュメント](#)を参照してください。



注記

Kibana コンソールで **security_exception** エラーが出され、Kibana インデックスにアクセスできない場合には、OAuth トークンの期限が切れている可能性があります。このエラーが表示された場合、Kibana コンソールからログアウトしてから再度ログインします。これにより OAuth トークンが更新され、インデックスにアクセスできるはずです。

第7章 クラスターロギングデプロイメントの設定

7.1. クラスターロギングの設定について

クラスターロギングを OpenShift Container Platform クラスターにインストールした後に、以下の設定を行うことができます。



注記

特に指示がない場合は、これらの設定を実行する前にクラスターロギングを管理外の状態に設定する必要があります。詳細は、「[クラスターロギングの管理状態の変更](#)」を参照してください。

管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

詳細は、[管理外の Operator のサポートポリシー](#)について参照してください。

7.1.1. クラスターロギングのデプロイおよび設定について

OpenShift Container Platform クラスターロギングは、小規模および中規模の OpenShift Container Platform クラスター用に調整されたデフォルト設定で使用されるように設計されています。

以下のインストール方法には、サンプルのクラスターロギングのカスタムリソース (CR) が含まれます。これを使用して、クラスターロギングインスタンスを作成し、クラスターロギングのデプロイメントを設定することができます。

デフォルトのクラスターロギングインストールを使用する必要がある場合は、サンプル CR を直接使用できます。

デプロイメントをカスタマイズする必要がある場合、必要に応じてサンプル CR に変更を加えます。以下では、クラスターロギングのインスタンスをインストール時に実行し、インストール後に変更する設定について説明します。クラスターロギングのカスタムリソース外で加える変更を含む、各コンポーネントの使用方法については、設定についてのセクションを参照してください。

7.1.1.1. クラスターロギングの設定およびチューニング

クラスターロギング環境は、**openshift-logging** プロジェクトにデプロイされるクラスターロギングのカスタムリソースを変更することによって設定できます。

インストール時またはインストール後に、以下のコンポーネントのいずれかを変更することができます。

メモリーおよび CPU

resources ブロックを有効なメモリーおよび CPU 値で変更することにより、各コンポーネントの CPU およびメモリーの両方の制限を調整することができます。

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
          memory: 16Gi
```

```

    requests:
      cpu: 500m
      memory: 16Gi
    type: "elasticsearch"
  collection:
    logs:
      fluentd:
        resources:
          limits:
            cpu:
            memory:
          requests:
            cpu:
            memory:
        type: "fluentd"
  visualization:
    kibana:
      resources:
        limits:
          cpu:
          memory:
        requests:
          cpu:
          memory:
      type: kibana
  curation:
    curator:
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 200m
          memory: 200Mi
      type: "curator"

```

Elasticsearch ストレージ

storageClass name および **size** パラメーターを使用し、Elasticsearch クラスターの永続ストレージのクラスおよびサイズを設定できます。Cluster Logging Operator は、これらのパラメーターに基づいて、Elasticsearch クラスターの各データノードについて **PersistentVolumeClaim** を作成します。

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
    storage:
      storageClassName: "gp2"
      size: "200G"

```

この例では、クラスターの各データノードが「gp2」ストレージの「200G」を要求する **PersistentVolumeClaim** にバインドされるように指定します。それぞれのプライマリーシャードは単一のレプリカによってサポートされます。



注記

storage ブロックを省略すると、一時ストレージのみを含むデプロイメントになります。

```
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage: {}
```

Elasticsearch レプリケーションポリシー

Elasticsearch シャードをクラスター内のデータノードにレプリケートする方法を定義するポリシーを設定できます。

- **FullRedundancy**:各インデックスのシャードはすべてのデータノードに完全にレプリケートされます。
- **MultipleRedundancy**:各インデックスのシャードはデータノードの半分に分散します。
- **SingleRedundancy**:各シャードの単一コピー。2つ以上のデータノードが存在する限り、ログは常に利用可能かつ回復可能です。
- **ZeroRedundancy**:シャードのコピーはありません。ログは、ノードの停止または失敗時に利用不可になる(または失われる)可能性があります。

Curator スケジュール

Curator のスケジュールを [cron 形式](#) で指定します。

```
spec:
  curation:
    type: "curator"
  resources:
    curator:
      schedule: "30 3 * * *"
```

7.1.1.2. 変更されたクラスターロギングカスタムリソースのサンプル

以下は、前述のオプションを使用して変更されたクラスターロギングのカスタムリソースの例です。

変更されたクラスターロギングカスタムリソースのサンプル

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
```

```

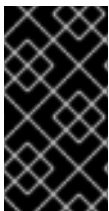
nodeCount: 3
resources:
  limits:
    memory: 32Gi
  requests:
    cpu: 3
    memory: 32Gi
  storage: {}
  redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    replicas: 1
curation:
  type: "curator"
  curator:
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 200Mi
    schedule: "*/5 * * * *"
collection:
  logs:
    type: "fluentd"
    fluentd:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 200m
          memory: 1Gi

```

7.2. クラスターログインの管理状態の変更

Cluster Logging Operator または Elasticsearch Operator によって管理される特定のコンポーネントを変更するには、Operator を **Unmanaged** (管理外) 状態に設定する必要があります。

Unmanaged (管理外) 状態では、Operator は CR の変更に対応しません。Unmanaged (管理外) 状態の場合、管理者が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できることが前提となります。



重要

管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

詳細は、[管理外の Operator のサポートポリシー](#) について参照してください。

Managed (管理対象) 状態では、Cluster Logging Operator (CLO) は クラスターロギングのカスタムリソース (CR) の変更に対応し、ロギングデプロイメントを適宜調整します。

OpenShift Container Platform ドキュメントでは、前提条件の段階で OpenShift Container Platform クラスターを Unmanaged (管理外) 状態に設定する必要があることを示しています。



注記

Elasticsearch Operator (EO) を管理外の状態に設定し、Cluster Logging Operator (CLO) を管理対象のままにする場合、CLO は EO に加えた変更を元に戻します。EO は CLO によって管理されているためです。

7.2.1. クラスターロギングの管理状態の変更

Cluster Logging Operator によって管理されるコンポーネントを変更するには、Operator を **Unmanaged (管理外)** の状態に設定する必要があります。

- Curator CronJob
- Elasticsearch CR
- Kibana Deployment
- ログコレクター DaemonSet

管理対象の状態ではこれらのコンポーネントを変更する場合、Cluster Logging Operator はそれらの変更を元に戻します。



注記

管理外のクラスターロギング環境は、Cluster Logging Operator を管理対象の状態に戻すまで更新を受信しません。

前提条件

- Cluster Logging Operator がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
```

```
....
```

```
spec:
  managementState: "Managed" ❶
```

- ❶ 管理状態を **Managed** (管理対象) または **Unmanaged** (管理外) として指定します。

7.2.2. Elasticsearch 管理状態の変更

Elasticsearch Operator によって管理される Elasticsearch デプロイメントファイルを変更するには、Operator を **Unmanaged** (管理外) の状態に設定する必要があります。

管理対象の状態ではこれらのコンポーネントを変更する場合、Elasticsearch Operator はこれらの変更を元に戻します。



注記

Elasticsearch Operator が管理対象の状態に戻されるまで、管理外の Elasticsearch クラスタは更新を受信しません。

前提条件

- Elasticsearch Operator がインストールされていること。
- **openshift-logging** プロジェクトに Elasticsearch CR の名前があること。

```
$ oc get -n openshift-logging Elasticsearch
NAME      AGE
elasticsearch 28h
```

手順

openshift-logging プロジェクトで Elasticsearch のカスタムリソース (CR) を編集します。

```
$ oc edit Elasticsearch elasticsearch

apiVersion: logging.openshift.io/v1
kind: Elasticsearch
metadata:
  name: elasticsearch

....

spec:
  managementState: "Managed" ❶
```

- ❶ 管理状態を **Managed** (管理対象) または **Unmanaged** (管理外) として指定します。



注記

Elasticsearch Operator (EO) を管理外の状態に設定し、Cluster Logging Operator (CLO) を管理対象のままにする場合、CLO は EO に加えた変更を元に戻します。EO は CLO によって管理されているためです。

7.3. クラスターロギングの設定

クラスターロギングは、**openshift-logging** プロジェクトにデプロイされるクラスターロギングのカスタムリソース (CR) を使用して設定できます。

Cluster Logging Operator は、クラスターロギング CR への変更の有無を監視し、欠落しているロギングコンポーネントを作成し、ロギングデプロイメントを適宜調整します。

クラスターロギング CR はクラスターロギングのカスタムリソース定義 (CRD) に基づいており、これは完全なクラスターロギングデプロイメントを定義し、ログを収集し、保存し、視覚化するために必要なロギングスタックのすべてのコンポーネントが含まれます。

クラスターロギングのカスタムリソース (CRD) のサンプル

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  creationTimestamp: '2019-03-20T18:07:02Z'
  generation: 1
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  curation:
    curator:
      resources: null
      schedule: 30 3 * * *
      type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu:
          memory:
        requests:
          cpu:
          memory:
        storage: {}
      type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      proxy:
        resources: null

```

```
replicas: 1
resources: null
type: kibana
```

クラスターロギングについて以下を設定できます。

- クラスターロギングを管理外の状態にします。これにより、管理者は個別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が想定されます。
- **cluster-logging-operator** デプロイメントで該当する環境変数を変更し、各クラスターロギングコンポーネントのイメージを上書きすることができます。
- ノードセレクターを使用してロギングコンポーネントの特定のノードを指定できます。

7.3.1. クラスターロギングコンポーネントイメージについて

クラスターロギングには、それぞれが1つ以上のイメージで実装されている複数のコンポーネントがあります。各イメージは **openshift-logging** プロジェクトの **cluster-logging-operator** デプロイメントで定義される環境変数で指定されており、これを変更することはできません。

以下のコマンドを実行してイメージを表示できます。

```
$ oc -n openshift-logging set env deployment/cluster-logging-operator --list | grep _IMAGE
```

```
ELASTICSEARCH_IMAGE=registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3 1
FLUENTD_IMAGE=registry.redhat.io/openshift4/ose-logging-fluentd:v4.3 2
KIBANA_IMAGE=registry.redhat.io/openshift4/ose-logging-kibana5:v4.3 3
CURATOR_IMAGE=registry.redhat.io/openshift4/ose-logging-curator5:v4.3 4
OAUTH_PROXY_IMAGE=registry.redhat.io/openshift4/ose-oauth-proxy:v4.3 5
```

- 1 **ELASTICSEARCH_IMAGE** は Elasticsearch をデプロイします。
- 2 **FLUENTD_IMAGE** は Fluentd をデプロイします。
- 3 **KIBANA_IMAGE** は Kibana をデプロイします。
- 4 **CURATOR_IMAGE** は Curator をデプロイします。
- 5 **OAUTH_PROXY_IMAGE** は OpenShift Container Platform の OAUTH を定義します。

値は、環境によって異なる可能性があります。



重要

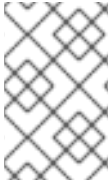
ロギングルートは Cluster Logging Operator によって管理され、ユーザーが変更することはできません。

7.4. ログデータを保存し、整理するための ELASTICSEARCH の設定

OpenShift Container Platform は Elasticsearch (ES) を使用してログデータを保存し、整理します。

ログストアに加えることのできる変更には、以下が含まれます。

- Elasticsearch クラスターのストレージ。
- シャードをクラスター内の複数のデータノードにレプリケートする方法 (完全なレプリケーションからレプリケーションなしを含む)。
- Elasticsearch データへの外部アクセスを許可する。



注記

Elasticsearch ノードのスケールダウンはサポートされていません。スケールダウン時に Elasticsearch Pod が誤って削除される場合があり、その場合にはシャードが割り当てられず、レプリカシャードが失われる可能性があります。

Elasticsearch はメモリー集約型アプリケーションです。それぞれの Elasticsearch ノードには、クラスターロギングのカスタムリソースで指定しない限り、メモリー要求および制限の両方に 16G のメモリーが必要です。初期設定の OpenShift Container Platform ノードのセットは、Elasticsearch クラスターをサポートするのに十分な大きさではない場合があります。その場合、推奨されるサイズ以上のメモリーを使用して実行できるようにノードを OpenShift Container Platform クラスターに追加する必要があります。

各 Elasticsearch ノードはこれより低い値のメモリー設定でも動作しますが、これは実稼働環境でのデプロイメントには推奨されません。



注記

Elasticsearch Operator (EO) を管理外の状態に設定し、Cluster Logging Operator (CLO) を管理対象のままにする場合、CLO は EO に加えた変更を元に戻します。EO は CLO によって管理されているためです。

7.4.1. Elasticsearch CPU およびメモリー制限の設定

それぞれのコンポーネント仕様は、CPU とメモリーの両方への調整を許可します。Elasticsearch Operator は環境に適した値を設定するため、これらの値を手動で調整する必要はありません。

各 Elasticsearch ノードはこれより低い値のメモリー設定でも動作しますが、これは実稼働環境でのデプロイメントには推奨されません。実稼働環境での使用の場合には、デフォルトの 16Gi よりも小さい値を各 Pod に割り当てることはできません。Pod ごとに割り当て可能な最大値は 64Gi であり、この範囲の中で、できるだけ多くのメモリーを割り当てることを推奨します。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
```

```
....
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      resources: ❶
      limits:
        memory: 16Gi
      requests:
        cpu: 500m
        memory: 16Gi
```

- ❶ 必要に応じて CPU およびメモリー制限を指定します。これらの値を空のままにすると、Elasticsearch Operator はデフォルト値を設定します。これらのデフォルト値はほとんどのデプロイメントでは問題なく使用できるはずです。

Elasticsearch の CPU およびメモリーの容量を調整する場合、要求値と制限値の両方を変更する必要があります。

以下は例になります。

```
resources:
  limits:
    cpu: "8"
    memory: "32Gi"
  requests:
    cpu: "8"
    memory: "32Gi"
```

Kubernetes は一般的にはノードの CPU 設定に従い、Elasticsearch が指定された制限を使用することを許可しません。**requests** と **limits** に同じ値を設定することにより、Elasticsearch は必要な CPU およびメモリーを確実に使用できるようにします (利用可能な CPU およびメモリーがノードにあることを前提とします)。

7.4.2. Elasticsearch レプリケーションポリシーの設定

Elasticsearch シャードをクラスター内の複数のデータノードにレプリケートする方法を定義できます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
oc edit clusterlogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
```

```
....
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy" ❶
```

❶ シャードの冗長性ポリシーを指定します。変更の保存時に変更が適用されます。

- **FullRedundancy**:Elasticsearch は、各インデックスのプライマリーシャードをすべてのデータノードに完全にレプリケートします。これは最高レベルの安全性を提供しますが、最大量のディスクが必要となり、パフォーマンスは最低レベルになります。
- **MultipleRedundancy**:Elasticsearch は、各インデックスのプライマリーシャードをデータノードの半分に完全にレプリケートします。これは、安全性とパフォーマンス間の適切なトレードオフを提供します。
- **SingleRedundancy**:Elasticsearch は、各インデックスのプライマリーシャードのコピーを1つ作成します。2つ以上のデータノードが存在する限り、ログは常に利用可能かつ回復可能です。5以上のノードを使用する場合には、MultipleRedundancyよりもパフォーマンスが良くなります。このポリシーは、単一 Elasticsearch ノードのデプロイメントには適用できません。
- **ZeroRedundancy**:Elasticsearch は、プライマリーシャードのコピーを作成しません。ノードが停止または失敗した場合、ログは利用不可となるか、失われる可能性があります。安全性よりもパフォーマンスを重視する場合や、独自のディスク/PVC バックアップ/復元戦略を実装している場合は、このモードを使用できます。



注記

インデックステンプレートのプライマリーシャードの数は Elasticsearch データノードの数と等しくなります。

7.4.3. Elasticsearch ストレージの設定

Elasticsearch には永続ストレージが必要です。ストレージが高速になると、Elasticsearch のパフォーマンスも高速になります。



警告

NFS ストレージをボリュームまたは永続ボリュームを使用 (または Gluster などの NAS を使用する) ことは Elasticsearch ストレージではサポートされません。Lucene は NFS が指定しないファイルシステムの動作に依存するためです。データの破損およびその他の問題が発生する可能性があります。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. クラスターロギング CR を編集してクラスターの各データノードが Persistent Volume Claim (永続ボリューム要求、PVC) にバインドされるように指定します。

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
  storage:
    storageClassName: "gp2"
    size: "200G"
```

この例では、クラスターの各データノードが、「200G」の AWS General Purpose SSD (gp2) ストレージを要求する Persistent Volume Claim (永続ボリューム要求、PVC) にバインドされるように指定します。

7.4.4. Elasticsearch での emptyDir ストレージの設定

Elasticsearch で emptyDir を使用することができます。これは、Pod のデータすべてが再起動時に失われる一時デプロイメントを作成します。



注記

emptyDir を使用する場合、Elasticsearch が再起動するか、または再デプロイされる場合にデータが失われます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. クラスターロギング CR を編集して emptyDir を指定します。

```
spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
  storage: {}
```

7.4.5. Elasticsearch のルートとしての公開

デフォルトでは、クラスターロギングでデプロイされた Elasticsearch はロギングクラスターの外部からアクセスできません。データにアクセスするツールについては、Elasticsearch へ外部アクセスするために re-encryption termination でルートを有効にすることができます。

re-encrypt ルート、OpenShift Container Platform トークンおよびインストールされた Elasticsearch CA 証明書を作成して Elasticsearch に外部からアクセスすることができます。次に、以下を含む cURL 要求を持つ Elasticsearch ノードにアクセスします。

- **Authorization: Bearer \${token}**
- Elasticsearch reencrypt ルートおよび [Elasticsearch API 要求](#)

内部からは、Elasticsearch クラスター IP を使用して Elasticsearch にアクセスすることができます。

以下のコマンドのいずれかを使用して、Elasticsearch クラスター IP を取得できます。

```
$ oc get service elasticsearch -o jsonpath={.spec.clusterIP} -n openshift-logging
```

```
172.30.183.229
```

```
oc get service elasticsearch -n openshift-logging
```

```
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
elasticsearch ClusterIP   172.30.183.229 <none>      9200/TCP   22h
```

```
$ oc exec elasticsearch-cdm-oplnhinv-1-5746475887-fj2f8 -n openshift-logging -- curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}" "https://172.30.183.229:9200/_cat/health"
```

```
% Total    % Received % Xferd Average Speed   Time    Time     Time Current
           Dload  Upload Total   Spent   Left  Speed
100  29 100  29  0  0  108  0 --:--:-- --:--:-- --:--:-- 108
```

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。
- ログにアクセスできるようになるには、プロジェクトへのアクセスが必要です。

手順

Elasticsearch を外部に公開するには、以下を実行します。

1. **openshift-logging** プロジェクトに切り替えます。

```
$ oc project openshift-logging
```

2. Elasticsearch から CA 証明書を抽出し、**admin-ca** ファイルに書き込みます。

```
$ oc extract secret/elasticsearch --to=. --keys=admin-ca
```

```
admin-ca
```

3. Elasticsearch サービスのルートを YAML ファイルとして作成します。

- a. 以下のように YAML ファイルを作成します。

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: elasticsearch
  namespace: openshift-logging
spec:
  host:
  to:
    kind: Service
    name: elasticsearch
  tls:
    termination: reencrypt
    destinationCACertificate: | ❶

```

- ❶ 次の手順で Elasticsearch CA 証明書を追加するか、またはコマンドを使用します。一部の re-encrypt ルートで必要とされる **spec.tls.key**、**spec.tls.certificate**、および **spec.tls.caCertificate** パラメーターを設定する必要はありません。

- b. 以下のコマンドを実行して作成したルート YAML に Elasticsearch CA 証明書を追加します。

```
cat ./admin-ca | sed -e "s/^  /" >> <file-name>.yaml
```

- c. ルートを作成します。

```

$ oc create -f <file-name>.yaml

route.route.openshift.io/elasticsearch created

```

4. Elasticsearch サービスが公開されていることを確認します。

- a. 要求に使用されるこの ServiceAccount のトークンを取得します。

```
$ token=$(oc whoami -t)
```

- b. 作成した **elasticsearch** ルートを環境変数として設定します。

```
$ routeES=`oc get route elasticsearch -o jsonpath={.spec.host}`
```

- c. ルートが正常に作成されていることを確認するには、公開されたルート経由で Elasticsearch にアクセスする以下のコマンドを実行します。

```
curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}"
"https://${routeES}/.operations.*/_search?size=1" | jq
```

以下のような出力が表示されます。

```

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100  944  100  944    0    0    62    0  0:00:15  0:00:15  --:--:--  204
{
  "took": 441,
  "timed_out": false,

```

```
"_shards": {
  "total": 3,
  "successful": 3,
  "skipped": 0,
  "failed": 0
},
"hits": {
  "total": 89157,
  "max_score": 1,
  "hits": [
    {
      "_index": ".operations.2019.03.15",
      "_type": "com.example.viaq.common",
      "_id": "ODdiNWlyYzAtMjg5Ni0TAtNWE3MDY1MjMzNTc3",
      "_score": 1,
      "_source": {
        "_SOURCE_MONOTONIC_TIMESTAMP": "673396",
        "systemd": {
          "t": {
            "BOOT_ID": "246c34ee9cdeecb41a608e94",
            "MACHINE_ID": "e904a0bb5efd3e36badee0c",
            "TRANSPORT": "kernel"
          },
          "u": {
            "SYSLOG_FACILITY": "0",
            "SYSLOG_IDENTIFIER": "kernel"
          }
        },
        "level": "info",
        "message": "acpiphp: Slot [30] registered",
        "hostname": "localhost.localdomain",
        "pipeline_metadata": {
          "collector": {
            "ipaddr4": "10.128.2.12",
            "ipaddr6": "fe80::xx:xxx:fe4c:5b09",
            "inputname": "fluent-plugin-systemd",
            "name": "fluentd",
            "received_at": "2019-03-15T20:25:06.273017+00:00",
            "version": "1.3.2 1.6.0"
          }
        },
        "@timestamp": "2019-03-15T20:00:13.808226+00:00",
        "viaq_msg_id": "ODdiNWlyYzAtMYTAtNWE3MDY1MjMzNTc3"
      }
    }
  ]
}
```

7.4.6. Elasticsearch アラートルール

これらのアラートルールを Prometheus に表示できます。

アラート	説明	重大度
ElasticsearchClusterNotHealthy	クラスターのヘルスステータスは少なくとも 2m の間 RED になります。クラスターは書き込みを受け入れず、シャードが見つからない可能性があるか、またはマスターノードがまだ選択されていません。	critical
ElasticsearchClusterNotHealthy	クラスターのヘルスステータスは少なくとも 20m の間 YELLOW になります。一部のシャードレプリカは割り当てられません。	warning
ElasticsearchBulkRequestsRejectionJumps	クラスターのノードにおける高いバルク除去率 (High Bulk Rejection Ratio) を示します。このノードはインデックスの速度に追いついていない可能性があります。	warning
ElasticsearchNodeDiskWatermarkReached	クラスターのノードでディスクの低い基準値に達しています。シャードをこのノードに割り当てることはできません。ノードにディスク領域を追加することを検討する必要があります。	alert
ElasticsearchNodeDiskWatermarkReached	クラスターのノードでディスクの高い基準値に達しています。一部のシャードは可能な場合に別のノードに再度割り当てられる可能性があります。ノードにディスク領域が追加されるか、またはこのノードに割り当てられる古いインデックスをドロップします。	high
ElasticsearchJVMHeapUseHigh	クラスター内のノード上での JVM ヒープの使用量は <value> です。	alert
AggregatedLoggingSystemCPUHigh	クラスター内のノード上でのシステム CPU の使用量は <value> です。	alert
ElasticsearchProcessCPUHigh	クラスター内のノード上での ES プロセス CPU の使用量は <value> です。	alert

7.5. KIBANA の設定

OpenShift Container Platform は Kibana を使用して、Fluentd によって収集され、Elasticsearch によってインデックス化されるログデータを表示します。

冗長性を確保するために Kibana をスケーリングし、Kibana ノードの CPU およびメモリーを設定することができます。



注記

特に指示がない場合は、これらの設定を実行する前にクラスターロギングを管理外の状態に設定する必要があります。詳細は、「[クラスターロギングの管理状態の変更](#)」を参照してください。

管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

詳細は、[管理外の Operator のサポートポリシー](#)について参照してください。

7.5.1. Kibana CPU およびメモリー制限の設定

それぞれのコンポーネント仕様は、CPU とメモリーの両方への調整を許可します。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...
spec:
  visualization:
    type: "kibana"
    kibana:
      replicas:
      resources: ①
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy: ②
      resources:
        limits:
          memory: 100Mi
        requests:
          cpu: 100m
          memory: 100Mi
```

- ① 各ノードに割り当てる CPU およびメモリー制限を指定します。
- ② Kibana プロキシに割り当てる CPU およびメモリー制限を指定します。

7.5.2. 冗長性を確保するための Kibana のスケーリング

冗長性を確保するために Kibana デプロイメントをスケーリングできます。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...
spec:
  visualization:
    type: "kibana"
  kibana:
    replicas: 1 ①
```

- ① Kibana ノードの数を指定します。

7.5.3. 容認 (Toleration) による Kibana Pod の配置の制御

Kibana Pod が実行するノードを制御し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

容認をクラスターロギングのカスタムリソース (CR) を利用して Kibana Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペアです。他の Pod にはない特定の **key:value** ペアを使用することで、Kibana Pod のみがそのノード上で実行されます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. 以下のコマンドを使用して、Kibana Pod をスケジュールするノードにテイントを追加します。

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

以下は例になります。

```
$ oc adm taint nodes node1 kibana=node:NoExecute
```

この例では、テイントをキー **kibana**、値 **node**、およびテイントの効果 **NoExecute** のある **node1** に配置します。**NoExecute** テイント effect を使用する必要があります。**NoExecute** は、テイントに一致する Pod のみをスケジュールし、一致しない既存の Pod を削除します。

2. クラスターロギングのカスタムリソース (CR) の **visualization** セクションを編集し、Kibana Pod の容認を設定します。

```
visualization:
  type: "kibana"
  kibana:
    tolerations:
      - key: "kibana" ①
        operator: "Exists" ②
        effect: "NoExecute" ③
        tolerationSeconds: 6000 ④
```

- ① ノードに追加したキーを指定します。
- ② **Exists** Operator を指定して、**key/value/effect** パラメーターが一致するようにします。
- ③ **NoExecute** effect を指定します。
- ④ オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

この容認は、**oc adm taint** コマンドで作成されたテイントと一致します。この容認のある Pod は、**node1** にスケジュールできます。

7.5.4. Kibana Visualize ツールのインストール

Kibana の **Visualize** タブを使用すると、コンテナログの監視用に視覚化やダッシュボードを作成でき、管理者ユーザー (**cluster-admin** または **cluster-reader**) はデプロイメント、namespace、Pod、およびコンテナごとにログを表示することができます。

手順

ダッシュボードおよび他の Kibana UI オブジェクトを読み込むには、以下を実行します。

1. 必要な場合は、Cluster Logging Operator のインストール時にデフォルトで作成される Kibana ルートを取得します。

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

NAMESPACE	NAME	HOST/PORT
PATH SERVICES	PORT	TERMINATION WILDCARD
openshift-logging	kibana	kibana-openshift-logging.apps.openshift.com
kibana	<all> reencrypt/Redirect	None

2. Elasticsearch Pod の名前を取得します。

```
$ oc get pods -l component=elasticsearch
```

NAME	READY	STATUS	RESTARTS	AGE
elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k	2/2	Running	0	22h
elasticsearch-cdm-5ceex6ts-2-f799564cb-19mj7	2/2	Running	0	22h
elasticsearch-cdm-5ceex6ts-3-585968dc68-k7kjr	2/2	Running	0	22h

3. この手順で必要とされるユーザーごとに必要な設定を作成します。

- a. ダッシュボードを追加する必要のあるユーザーとして Kibana ダッシュボードにログインします。

```
https://kibana-openshift-logging.apps.openshift.com 1
```

- 1 URL は Kibana ルートです。

- b. **Authorize Access** ページが表示される場合、すべてのパーミッションを選択してから **Allow selected permissions** をクリックします。

- c. Kibana ダッシュボードからログアウトします。

4. Elasticsearch Pod のいずれかの名前を使用して、Pod が置かれているプロジェクトで以下のコマンドを実行します。

```
$ oc exec <es-pod> -- es_load_kibana_ui_objects <user-name>
```

以下は例になります。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k -- es_load_kibana_ui_objects <user-name>
```

注記

視覚化やダッシュボードなどの Kibana オブジェクトのメタデータは、`.kibana` `{user_hash}` インデックス形式で Elasticsearch に保存されます。 `userhash=$(echo -n $username | sha1sum | awk '{print $1}')` コマンドを使用して、`user_hash` を取得できます。デフォルトで、Kibana `shared_ops` インデックスモードでは、クラスター管理者ロールを持つすべてのユーザーがインデックスを共有でき、この Kibana オブジェクトメタデータは `.kibana` インデックスに保存されます。

インポート/エクスポート機能を使用するか、`curl` コマンドを使用してメタデータを Elasticsearch インデックスに挿入して、特定のユーザーに対してカスタムダッシュボードをインポートできます。

7.6. ELASTICSEARCH データのキュレーション

Elasticsearch Curator ツールは、グローバルに、またはプロジェクトごとにスケジュールされたメンテナンス操作を実行します。Curator はその設定に基づいて動作を実行します。

Cluster Logging Operator は Curator とその設定をインストールします。Curator [cron スケジュール](#) は、クラスターロギングのカスタムリソースを使用して設定でき、追加の設定オプションは **openshift-logging** プロジェクトの Curator ConfigMap、**curator** にあります。これには、Curator 設定ファイル `curator5.yaml` および OpenShift Container Platform カスタム設定ファイル `config.yaml` が組み込まれています。

OpenShift Container Platform は `config.yaml` を内部で使用し、Curator の [Action ファイル](#) を生成します。

オプションで、**action** ファイルを直接使用できます。このファイルを編集すると、定期的に行うことができるように Curator で利用できるアクションを使用できます。ただし、これによりファイルの変更によりクラスターに破壊的な影響が及ぶ可能性があり、必要なインデックス/設定が Elasticsearch から削除され

る可能性があるため、上級ユーザーのみがこれを実行することが推奨されます。ほとんどのユーザーは Curator 設定マップのみを変更するだけでよく、**action** ファイルを編集することはできません。

7.6.1. Curator スケジュールの設定

クラスターロギングインストールで作成されたクラスターロギングのカスタムリソースを使用して、Curator のスケジュールを指定できます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

Curator スケジュールを設定するには、以下を実行します。

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソースを編集します。

```
$ oc edit clusterlogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...

curation:
  curator:
    schedule: 30 3 * * * 1
    type: curator
```

- 1 Curator のスケジュールを **cron 形式** で指定します。



注記

タイムゾーンは Curator Pod が実行されるホストノードに基づいて設定されます。

7.6.2. Curator インデックス削除の設定

Curator を、保持設定に基づいて Elasticsearch データを削除するように設定できます。プロジェクトごとの設定およびグローバル設定を行うことができます。グローバル設定は、指定されていないプロジェクトに適用されます。プロジェクトごとの設定はグローバル設定を上書きします。

前提条件

- クラスターロギングがインストールされている必要があります。

手順

インデックスを削除するには、以下を実行します。

1. OpenShift Container Platform カスタム Curator 設定ファイルを編集します。

```
$ oc edit configmap/curator
```

2. 必要に応じて以下のパラメーターを設定します。

```
config.yaml: |
  project_name:
    action
    unit:value
```

利用可能なパラメーターを以下に示します。

表7.1 プロジェクトオプション

変数名	説明
project_name	プロジェクトの実際の名前 (myapp-devel など)。OpenShift Container Platform の操作ログについては、名前 .operations をプロジェクト名として使用します。
action	実行するアクション。現在許可されているのは delete のみです。
unit	削除に使用する期間 (days 、 weeks 、または months)。
value	単位数。

表7.2 フィルターオプション

変数名	説明
.defaults	.defaults を project_name として使用し、指定されていないプロジェクトのデフォルトを設定します。
.regex	プロジェクト名に一致する正規表現の一覧。
pattern	適切にエスケープされた有効な正規表現パターン。一重引用符で囲まれています。

たとえば、以下のように Curator を設定します。

- **1 day** を経過した **myapp-dev** プロジェクトのインデックスを削除する
- **1 week** を経過した **myapp-qa** プロジェクトのインデックスを削除する
- **8 weeks** を経過した **operations** ログを削除する
- **31 days** を経過したその他すべてのプロジェクトのインデックスを削除する
- **^project\..+\.dev.*\$** 正規表現と一致する、1日を経過したインデックスを削除する
- **^project\..+\.test.*\$** 正規表現と一致する、2日を経過したインデックスを削除する

以下を使用します。

```
config.yaml: |
  .defaults:
    delete:
      days: 31

  .operations:
    delete:
      weeks: 8

  myapp-dev:
    delete:
      days: 1

  myapp-qe:
    delete:
      weeks: 1

  .regex:
    - pattern: '^project\..+\-dev\..*$'
      delete:
        days: 1
    - pattern: '^project\..+\-test\..*$'
      delete:
        days: 2
```

重要

months を操作の **\$UNIT** として使用する場合、Curator は今月の当日ではなく、今月の最初の日からカウントを開始します。たとえば、今日が 4 月 15 日であり、現時点で 2 カ月を経過したインデックスを削除する場合 (`delete: months: 2`)、Curator は 2 月 15 日より古い日付のインデックスを削除するのではなく、2 月 1 日より古いインデックスを削除します。つまり、今月の最初の日付まで遡り、そこから 2 カ月遡ります。Curator で厳密な設定をする必要がある場合、最も適切な方法として日数 (例: **delete: days: 30**) を使用することができます。

7.6.3. Curator のトラブルシューティング

本セクションの情報を使用して Curator のデバッグを実行できます。たとえば、Curator が失敗状態にあり、ログメッセージで理由が示されていない場合、次にスケジュールされている cron ジョブの実行を待機する代わりに、ログレベルを引き上げ、新規ジョブをトリガーできます。

前提条件

クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

Curator のデバッグログを有効にし、次の Curator の反復を手動でトリガーします。

1. Curator のデバッグログを有効にします。

```
$ oc set env cronjob/curator CURATOR_LOG_LEVEL=DEBUG
CURATOR_SCRIPT_LOG_LEVEL=DEBUG
```

ログレベルを指定します。

- **CRITICAL**。Curator は重大なメッセージのみを表示します。
- **ERROR**。Curator はエラーおよび重大なメッセージのみを表示します。
- **WARNING**。Curator はエラー、警告、および重大なメッセージのみを表示します。
- **INFO**。Curator は情報、エラー、警告、および重大なメッセージのみを表示します。
- **DEBUG**。Curator は上記のすべてに加えてデバッグメッセージのみを表示します。デフォルト値は INFO です。



注記

クラスターロギングは、OpenShift Container Platform ラッパースクリプト (`run.sh` および `convert.py`) で OpenShift Container Platform カスタム環境変数 **CURATOR_SCRIPT_LOG_LEVEL** を使用します。この環境変数は、必要に応じてスクリプトのデバッグ用に **CURATOR_LOG_LEVEL** と同じ値を取ります。

2. 次の Curator の反復をトリガーします。

```
$ oc create job --from=cronjob/curator <job_name>
```

3. 以下のコマンドを使用して CronJob を制御します。

- CronJob を一時停止します。

```
$ oc patch cronjob curator -p '{"spec":{"suspend":true}}'
```

- CronJob を再開します。

```
$ oc patch cronjob curator -p '{"spec":{"suspend":false}}'
```

- CronJob スケジュールを変更します。

```
$ oc patch cronjob curator -p '{"spec":{"schedule":"0 0 * * *"}}' 1
```

1 **schedule** オプションは、[cron 形式](#)のスケジュールを受け入れます。

7.6.4. スクリプト化されたデプロイメントでの Curator の設定

Curator をスクリプト化されたデプロイメントで設定する必要がある場合に、本セクションの情報を 사용합니다。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされていること。
- クラスターロギングを管理外の状態に設定する。

手順

以下のスニペットを使用し、スクリプトで Curator を設定します。

- スクリプト化されたデプロイメントの場合

1. 設定を作成し、変更します。

- a. Curator 設定ファイルおよび OpenShift Container Platform カスタム設定ファイルを Curator 設定マップからコピーし、それぞれについて別個のファイルをマップし、作成します。

```
$ oc extract configmap/curator --keys=curator5.yaml,config.yaml --to=/my/config
```

- b. `/my/config/curator5.yaml` および `/my/config/config.yaml` ファイルを編集します。

2. 既存の Curator 設定マップを削除し、編集された YAML ファイルを新規の Curator 設定マップに追加します。

```
$ oc delete configmap curator ; sleep 1
$ oc create configmap curator \
  --from-file=curator5.yaml=/my/config/curator5.yaml \
  --from-file=config.yaml=/my/config/config.yaml \
  ; sleep 1
```

次の反復でこの設定を使用します。

- `action` ファイルを使用している場合

1. 設定を作成し、変更します。

- a. Curator 設定ファイルおよび `action` ファイルを Curator 設定マップからコピーし、それぞれについて別個のファイルをマップし、作成します。

```
$ oc extract configmap/curator --keys=curator5.yaml,actions.yaml --to=/my/config
```

- b. `/my/config/curator5.yaml` および `/my/config/actions.yaml` ファイルを編集します。

2. 既存の Curator 設定マップを削除し、編集された YAML ファイルを新規の Curator 設定マップに追加します。

```
$ oc delete configmap curator ; sleep 1
$ oc create configmap curator \
  --from-file=curator5.yaml=/my/config/curator5.yaml \
  --from-file=actions.yaml=/my/config/actions.yaml \
  ; sleep 1
```

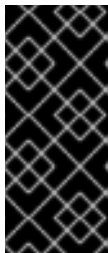
次の反復でこの設定を使用します。

7.6.5. Curator Action ファイルの使用

`openshift-logging` プロジェクトの Curator ConfigMap には、[Curator Action ファイル](#) が含まれます。このファイルで、Curator Action が定期的に行われるように設定します。

ただし、`action` ファイルを使用する場合、OpenShift Container Platform は、重要な内部インデックスが間違えて削除されないように設定されている `curator` ConfigMap の `config.yaml` セクションを無視します。`action` ファイルを使用するには、除外ルールを設定に加えてこれらのインデックスを保持する

必要があります。さらに、本トピックの手順に従う他のすべてのパターンも手動で追加する必要があります。



重要

actions および **config.yaml** は相互に排他的な設定ファイルです。**actions** ファイルがある場合、OpenShift Container Platform は **config.yaml** ファイルを無視します。**action** ファイルの使用は、クラスターに破壊的な影響を与え、必要なインデックス/設定が Elasticsearch から削除される可能性があるために、状況ユーザーのみがこれを実行することが推奨されます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされていること。
- クラスターロギングを管理外の状態に設定する。管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

手順

Curator をインデックスを削除するように設定するには、以下を実行します。

1. Curator ConfigMap を編集します。

```
oc edit cm/curator -n openshift-logging
```

2. **action** ファイルに以下の変更を加えます。

```
actions:
1:
  action: delete_indices 1
  description: >-
    Delete .operations indices older than 30 days.
    Ignore the error if the filter does not
    result in an actionable list of indices (ignore_empty_list).
    See
https://www.elastic.co/guide/en/elasticsearch/client/curator/5.2/ex\_delete\_indices.html
  options:
    # Swallow curator.exception.NoIndices exception
    ignore_empty_list: True
    # In seconds, default is 300
    timeout_override: ${CURATOR_TIMEOUT}
    # Don't swallow any other exceptions
    continue_if_exception: False
    # Optionally disable action, useful for debugging
    disable_action: False
    # All filters are bound by logical AND
  filters: 2
  - filtertype: pattern
    kind: regex
    value: '^\.operations\..*$'
    exclude: False 3
  - filtertype: age
    # Parse timestamp from index name
```

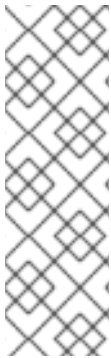
```
source: name
direction: older
timestring: '%Y.%m.%d'
unit: days
unit_count: 30
exclude: False
```

- 1 **delete_indices** を指定して指定されたインデックスを削除します。
- 2 **filers** パラメーターを使用して削除されるインデックスを指定します。これらのパラメーターについての詳細は [Elastic Search Curator のドキュメント](#) を参照してください。
- 3 インデックスが削除されるように **false** を指定します。

7.7. ロギングコレクターの設定

OpenShift Container Platform は Fluentd を使用して、クラスターから操作およびアプリケーションログを収集し、Kubernetes Pod および Namespace メタデータでデータを拡充します。

ログローテーション、ログの位置を設定し、外部のログアグリゲーターを使用し、ログコレクターの他の設定を行うことができます。



注記

特に指示がない場合は、これらの設定を実行する前にクラスターロギングを管理外の状態に設定する必要があります。詳細は、「[クラスターロギングの管理状態の変更](#)」を参照してください。

管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

詳細は、[管理外の Operator のサポートポリシー](#) について参照してください。

7.7.1. ロギングコレクター Pod の表示

oc get pods --all-namespaces -o wide コマンドを使用して、Fluentd がデプロイされるノードを表示できます。

手順

openshift-logging プロジェクトで以下のコマンドを実行します。

```
$ oc get pods --all-namespaces -o wide | grep fluentd
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE	READINESS	GATES				
fluentd-5mr28	1/1	Running	0	4m56s	10.129.2.12	ip-10-0-164-233.ec2.internal
<none>	<none>					
fluentd-cnc4c	1/1	Running	0	4m56s	10.128.2.13	ip-10-0-155-142.ec2.internal
<none>	<none>					
fluentd-nlp8z	1/1	Running	0	4m56s	10.131.0.13	ip-10-0-138-77.ec2.internal
<none>	<none>					
fluentd-rknlk	1/1	Running	0	4m56s	10.128.0.33	ip-10-0-128-130.ec2.internal
<none>	<none>					

```

fluentd-rsm49      1/1    Running 0      4m56s 10.129.0.37 ip-10-0-163-191.ec2.internal
<none>           <none>
fluentd-wjt8s     1/1    Running 0      4m56s 10.130.0.42 ip-10-0-156-251.ec2.internal
<none>           <none>

```

7.7.2. ログコレクター CPU およびメモリー制限の設定

ログコレクターは、CPU とメモリー制限の両方への調整を許可します。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```

$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...

spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: ①
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi

```

- ① 必要に応じて CPU、メモリー制限および要求を指定します。表示される値はデフォルト値です。

7.7.3. Fluentd のバッファークチャンク制限の設定

Fluentd ログコレクターが多数のログを処理できない場合、Fluentd はメモリーの使用量を減らし、データ損失を防ぐためにファイルバッファリングを実行します。

Fluentd ファイルバッファリングは、記録を **chunks** に保管します。チャンクは **buffers** に保管されません。

Fluentd Daemonset で環境変数を編集して、クラスターでファイルバッファリングを調整できます。



注記

Fluentd Daemonset で **FILE_BUFFER_LIMIT** または **BUFFER_SIZE_LIMIT** パラメーターを変更するには、クラスターロギングを管理外 (unmanaged) の状態に設定する必要があります。管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

- **BUFFER_SIZE_LIMIT** このパラメーターは、Fluentd が新規チャンクを作成する前に各チャンクファイルの最大サイズを決定します。デフォルトは **8M** です。このパラメーターは、Fluentd **chunk_limit_size** 変数を設定します。

高い値の **BUFFER_SIZE_LIMIT** の場合、チャンクファイルごとにより多くのレコードを収集できます。ただし、レコードのサイズが大きくなると、ログストアに送信されるまでにより長い時間がかかります。

- **FILE_BUFFER_LIMIT** このパラメーターは、ログ出力ごとにファイルバッファサイズを決定します。この値は、Fluentd Pod がスケジューラされるノードの利用可能な領域をベースとする要求のみになります。OpenShift Container Platform では、Fluentd がノードの容量を超えることを許可しません。デフォルトは **256Mi** です。

高い値の **FILE_BUFFER_LIMIT** は出力数に基づいてより高い値の **BUFFER_QUEUE_LIMIT** に変換される可能性があります。ただし、ノードの領域が不足すると、Fluentd は失敗する可能性があります。

デフォルトで、**number_of_outputs** は、すべてのログが単一リソースに送信され、追加のリソースごとに **1** つずつ増分する場合に **1** になります。ログ転送 API、Fluentd **Forward** プロトコル、または syslog プロトコルを使用してログを外部の場所に転送する場合、複数の出力がある可能性があります。

永続的なボリュームサイズは、**FILE_BUFFER_LIMIT** に出力数を乗算した結果よりも大きくなければなりません。

- **BUFFER_QUEUE_LIMIT** このパラメーターは、許可されるバッファチャンクの最大数です。**BUFFER_QUEUE_LIMIT** パラメーターは直接調整できません。OpenShift Container Platform は、この値を利用可能なロギングの出力数、チャンクサイズ、およびファイルシステム領域に基づいて計算します。デフォルトは **32** チャンクです。**BUFFER_QUEUE_LIMIT** を変更するには、**FILE_BUFFER_LIMIT** の値を変更する必要があります。**BUFFER_QUEUE_LIMIT** パラメーターは Fluentd **queue_limit_length** パラメーターを設定します。

OpenShift Container Platform は **BUFFER_QUEUE_LIMIT** を $(\text{FILE_BUFFER_LIMIT} / (\text{number_of_outputs} * \text{BUFFER_SIZE_LIMIT}))$ として計算します。

デフォルトの値のセットを使用すると、**BUFFER_QUEUE_LIMIT** の値は **32** になります。

- **FILE_BUFFER_LIMIT = 256Mi**
- **number_of_outputs = 1**
- **BUFFER_SIZE_LIMIT = 8Mi**

OpenShift Container Platform は Fluentd [ファイルバッファプラグイン](#) を使用してチャンクを保存する方法を設定します。以下のコマンドを使用してバッファファイルの場所を確認できます。

```
$ oc get cm fluentd -o json | jq -r '.data."fluent.conf"'
```

```
<buffer>
  @type file 1
  path '/var/lib/flunetd/retry-elasticsearch' 2
```

- 1** Fluentd ファイル バッファプラグイン。この値は変更しないでください。
- 2** バッファチャンクが格納されるパス。

前提条件

- クラスターロギングを管理外の状態に設定する。管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

手順

バッファチャンク制限を設定するには、以下を実行します。

1. **fluentd** Daemonset で以下のいずれかのパラメーターを編集します。

```
spec:
  template:
    spec:
      containers:
        env:
          - name: FILE_BUFFER_LIMIT 1
            value: "256"
          - name: BUFFER_SIZE_LIMIT 2
            value: 8Mi
```

- 1** 出力ごとに Fluentd ファイルバッファサイズを指定します。
- 2** 各 Fluentd バッファチャンクの最大サイズを指定します。

7.7.4. 環境変数を使用したロギングコレクターの設定

環境変数を使用して Fluentd ログコレクターの設定を変更することができます。

利用可能な環境変数の一覧については、Github の [Fluentd README](#) を参照してください。

前提条件

- クラスターロギングを管理外の状態に設定する。管理外の状態の Operator 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設定およびアップグレードを完全に制御していることを前提としています。

手順

必要に応じて Fluentd 環境変数のいずれかを設定します。

```
oc set env ds/fluentd <env-var>=<value>
```

以下は例になります。

```
oc set env ds/fluentd LOGGING_FILE_AGE=30
```

7.7.5. ロギングコレクターのアラートについて

以下のアラートはロギングコレクターによって生成され、Prometheus UI の **Alerts** タブに表示できます。

すべてのロギングコレクターアラートは、OpenShift Container Platform Web コンソールの **Monitoring** → **Alerts** ページに一覧表示されます。アラートは以下の状態のいずれかになります。

- **Firing**アラートの状態はタイムアウトの期間は true になります。Firing アラートの末尾の **Option** メニューをクリックし、詳細情報を表示するか、アラートを非通知 (silence) にします。
- **Pending**: このアラート状態は現時点で true ですが、タイムアウトに達していません。
- **Not Firing**アラートは現時点でトリガーされていません。

表7.3 Fluentd Prometheus アラート

アラート	メッセージ	説明	重大度
FluentdErrorsHigh	In the last minute, <value> errors reported by fluentd <instance>.	Fluentd は指定した数 (デフォルトでは 10) よりも多くの問題を報告しています。	Critical
FluentdNodeDown	Prometheus could not scrape fluentd <instance> for more than 10m.	Fluentd は Prometheus が特定の Fluentd インスタンスを収集できなかったことを報告します。	Critical
FluentdQueueLengthBurst	In the last minute, fluentd <instance> buffer queue length increased more than 32.Current value is <value>.	Fluentd は値が大きすぎることを報告しています。	Warning
FluentdQueueLengthIncreasing	In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1.Current value is <value>.	Fluentd はキューの使用についての問題を報告しています。	Critical

7.8. KUBERNETES イベントの収集および保存

OpenShift Container Platform イベントルーターは、Kubernetes イベントを監視し、それらをクラスターロギングによって収集できるようにログに記録する Pod です。イベントルーターは手動でデプロイする必要があります。

イベントルーターはすべてのプロジェクトからイベントを収集し、それらを **STDOUT** に書き込みます。Fluentd はそれらのイベントを収集し、それらを OpenShift Container Platform Elasticsearch インスタンスに転送します。Elasticsearch はイベントを **infra** インデックスにインデックス化します。



重要

イベントルーターは追加の負荷を Fluentd に追加し、処理できる他のログメッセージの数に影響を与える可能性があります。

7.8.1. イベントルーターのデプロイおよび設定

以下の手順を使用してイベントルーターをクラスターにデプロイします。イベントルーターを **openshift-logging** プロジェクトに常にデプロイし、クラスター全体でイベントが収集されるようにする必要があります。

以下のテンプレートオブジェクトは、イベントルーターに必要なサービスアカウント、クラスターロールおよびクラスターロールバインディングを作成します。テンプレートはイベントルーター Pod も設定し、デプロイします。このテンプレートは変更せずに使用するか、Deployment オブジェクトの CPU およびメモリー要求を変更することができます。

前提条件

- サービスアカウントを作成し、クラスターロールバインディングを更新するには、適切なパーミッションが必要です。たとえば、以下のテンプレートを、**cluster-admin** ロールを持つユーザーで実行できます。
- クラスターロギングがインストールされている必要があります。

手順

1. イベントルーターのテンプレートを作成します。

```
kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to cluster logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount 1
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole 2
    apiVersion: v1
    metadata:
      name: event-reader
    rules:
      - apiGroups: [""]
        resources: ["events"]
        verbs: ["get", "watch", "list"]
  - kind: ClusterRoleBinding 3
    apiVersion: v1
```

```
metadata:
  name: event-reader-binding
subjects:
- kind: ServiceAccount
  name: eventrouter
  namespace: ${NAMESPACE}
roleRef:
  kind: ClusterRole
  name: event-reader
- kind: ConfigMap 4
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment 5
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  labels:
    component: eventrouter
    logging-infra: eventrouter
    provider: openshift
  spec:
    selector:
      matchLabels:
        component: eventrouter
        logging-infra: eventrouter
        provider: openshift
    replicas: 1
    template:
      metadata:
        labels:
          component: eventrouter
          logging-infra: eventrouter
          provider: openshift
        name: eventrouter
      spec:
        serviceAccount: eventrouter
        containers:
        - name: kube-eventrouter
          image: ${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
          volumeMounts:
          - name: config-volume
            mountPath: /etc/eventrouter
        volumes:
```

```

- name: config-volume
  configMap:
    name: eventrouter
parameters:
- name: IMAGE
  displayName: Image
  value: "registry.redhat.io/openshift4/ose-logging-eventrouter:latest"
- name: CPU 6
  displayName: CPU
  value: "100m"
- name: MEMORY 7
  displayName: Memory
  value: "128Mi"
- name: NAMESPACE
  displayName: Namespace
  value: "openshift-logging" 8

```

- 1** イベントルーターの **openshift-logging** プロジェクトでサービスアカウントを作成します。
- 2** ClusterRole を作成し、クラスター内のイベントを監視します。
- 3** ClusterRoleBinding を作成し、ClusterRole を ServiceAccount にバインドします。
- 4** **openshift-logging** プロジェクトで ConfigMap を作成し、必要な **config.json** ファイルを生成します。
- 5** **openshift-logging** プロジェクトで Deployment を作成し、イベントルーター Pod を生成し、設定します。
- 6** イベントルーター Pod に割り当てるメモリの最小量を指定します。デフォルトは **128Mi** に設定されます。
- 7** イベントルーター Pod に割り当てる CPU の最小量を指定します。デフォルトは **100m** に設定されます。
- 8** オブジェクトをインストールする **openshift-logging** プロジェクトを指定します。

2. 以下のコマンドを使用してテンプレートを処理し、これを適用します。

```
$ oc process -f <templatefile> | oc apply -f -
```

以下は例になります。

```
$ oc process -f eventrouter.yaml | oc apply -f -
```

出力例

```

serviceaccount/logging-eventrouter created
clusterrole.authorization.openshift.io/event-reader created
clusterrolebinding.authorization.openshift.io/event-reader-binding created
configmap/logging-eventrouter created
deployment.apps/logging-eventrouter created

```

3. イベントルーターが **openshift-logging** プロジェクトにインストールされていることを確認します。
 - a. 新規イベントルーター Pod を表示します。

```
$ oc get pods --selector component=eventrouter -o name -n openshift-logging
```

出力例

```
pod/cluster-logging-eventrouter-d649f97c8-qvv8r
```

- b. イベントルーターによって収集されるイベントを表示します。

```
$ oc logs <cluster_logging_eventrouter_pod> -n openshift-logging
```

以下は例になります。

```
$ oc logs cluster-logging-eventrouter-d649f97c8-qvv8r -n openshift-logging
```

出力例

```
{
  "verb": "ADDED",
  "event": {
    "metadata": {
      "name": "openshift-service-catalog-controller-manager-remover.1632d931e88fcd8f",
      "namespace": "openshift-service-catalog-removed",
      "selfLink": "/api/v1/namespaces/openshift-service-catalog-removed/events/openshift-service-catalog-controller-manager-remover.1632d931e88fcd8f",
      "uid": "787d7b26-3d2f-4017-b0b0-420db4ae62c0",
      "resourceVersion": "21399",
      "creationTimestamp": "2020-09-08T15:40:26Z"
    },
    "involvedObject": {
      "kind": "Job",
      "namespace": "openshift-service-catalog-removed",
      "name": "openshift-service-catalog-controller-manager-remover",
      "uid": "fac9f479-4ad5-4a57-8adc-cb25d3d9cf8f",
      "apiVersion": "batch/v1",
      "resourceVersion": "21280"
    },
    "reason": "Completed",
    "message": "Job completed",
    "source": {
      "component": "job-controller"
    },
    "firstTimestamp": "2020-09-08T15:40:26Z",
    "lastTimestamp": "2020-09-08T15:40:26Z",
    "count": 1,
    "type": "Normal"
  }
}
```

また、Elasticsearch **infra** インデックスを使用してインデックスパターンを作成し、Kibanaを使用してイベントを表示することもできます。

7.9. 容認を使用した クラスターロギング POD 配置の制御

テイントおよび容認を使用することで、クラスターロギング Pod が特定のノードで実行され、その他のワークロードがそれらのノードで実行されないようにします。

テイントおよび容認は、単純な **key:value** のペアです。ノードのテイントはノードに対し、テイントを容認しないすべての Pod を拒否するよう指示します。

key は最大 253 文字までの文字列で、**value** は最大 63 文字までの文字列になります。文字列は文字または数字で開始する必要があり、文字、数字、ハイフン、ドットおよびアンダースコアを含めることができます。

容認を使用したクラスターロギング CR のサンプル

```
apiVersion: "logging.openshift.io/v1"
```

```
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 1
      tolerations: ❶
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 8Gi
      requests:
        cpu: 100m
        memory: 1Gi
      storage: {}
      redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: ❷
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi
    replicas: 1
  curation:
    type: "curator"
    curator:
      tolerations: ❸
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 100Mi
      schedule: "*/5 * * * *"
  collection:
    logs:
      type: "fluentd"
```



```

fluentd:
  tolerations: ④
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
  resources:
    limits:
      memory: 2Gi
    requests:
      cpu: 100m
      memory: 1Gi

```

- ① この容認は Elasticsearch Pod に追加されます。
- ② この容認は Kibana Pod に追加されます。
- ③ この容認は Curator Pod に追加されます。
- ④ この容認はロギングコレクター Pod に追加されます。

7.9.1. 容認を使用した Elasticsearch Pod 配置の制御

Elasticsearch Pod が実行するノードを制御し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

容認をクラスターロギングのカスタムリソース (CR) で Elasticsearch Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペアです。他の Pod にはない特定の **key:value** ペアを使用することで、Elasticsearch Pod のみがそのノード上で実行されるようにできます。

デフォルトで、Elasticsearch Pod には以下の容認があります。

```

tolerations:
  - effect: "NoExecute"
    key: "node.kubernetes.io/disk-pressure"
    operator: "Exists"

```

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. 以下のコマンドを使用して、クラスターロギング Pod をスケジュールするノードにテイントを追加します。

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

以下は例になります。

```
$ oc adm taint nodes node1 elasticsearch=node:NoExecute
```

この例では、テイントをキー **elasticsearch**、値 **node**、およびテイントの効果 **NoExecute** のある **node1** に配置します。**NoExecute** 効果のノードは、テイントに一致する Pod のみをスケジューリングし、一致しない既存の Pod を削除します。

2. クラスタロギングのカスタムリソース (CR) の **logStore** セクションを編集し、Elasticsearch Pod の容認を設定します。

```
logStore:
  type: "elasticsearch"
  elasticsearch:
    nodeCount: 1
    tolerations:
      - key: "elasticsearch" ①
        operator: "Exists" ②
        effect: "NoExecute" ③
        tolerationSeconds: 6000 ④
```

- ① ノードに追加したキーを指定します。
- ② **Exists** Operator を指定し、キー **elasticsearch** のあるテイントがノードに存在する必要があるようにします。
- ③ **NoExecute** effect を指定します。
- ④ オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

この容認は、**oc adm taint** コマンドで作成されたテイントと一致します。この容認のある Pod は **node1** にスケジューリングできます。

7.9.2. 容認 (Toleration) による Kibana Pod の配置の制御

Kibana Pod が実行するノードを制御し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

容認をクラスタロギングのカスタムリソース (CR) を利用して Kibana Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペアです。他の Pod にはない特定の **key:value** ペアを使用することで、Kibana Pod のみがそのノード上で実行されます。

前提条件

- クラスタロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. 以下のコマンドを使用して、Kibana Pod をスケジューリングするノードにテイントを追加します。

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

以下は例になります。

```
$ oc adm taint nodes node1 kibana=node:NoExecute
```

この例では、テイントをキー **kibana**、値 **node**、およびテイントの効果 **NoExecute** のある **node1** に配置します。**NoExecute** テイント effect を使用する必要があります。**NoExecute** は、テイントに一致する Pod のみをスケジュールし、一致しない既存の Pod を削除します。

2. クラスターロギングのカスタムリソース (CR) の **visualization** セクションを編集し、Kibana Pod の容認を設定します。

```
visualization:
  type: "kibana"
  kibana:
    tolerations:
      - key: "kibana" ①
        operator: "Exists" ②
        effect: "NoExecute" ③
        tolerationSeconds: 6000 ④
```

- ① ノードに追加したキーを指定します。
- ② **Exists** Operator を指定して、**key/value/effect** パラメーターが一致するようにします。
- ③ **NoExecute** effect を指定します。
- ④ オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

この容認は、**oc adm taint** コマンドで作成されたテイントと一致します。この容認のある Pod は、**node1** にスケジュールできます。

7.9.3. 容認を使用した Curator Pod 配置の制御

Curator Pod が実行するノードを制御し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

容認をクラスターロギングのカスタムリソース (CR) で Curator Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペア です。他の Pod にはない特定の **key:value** ペアを使用することで、Curator Pod のみがそのノード上で実行されるようになります。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. 以下のコマンドを使用して、Curator Pod をスケジュールする必要のあるノードにテイントを追加します。

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

以下は例になります。

```
$ oc adm taint nodes node1 curator=node:NoExecute
```

この例では、テイントをキー **curator**、値 **node**、およびテイント effect **NoExecute** のある **node1** に配置します。**NoExecute** テイント effect を使用する必要があります。**NoExecute** は、テイントに一致する Pod のみをスケジュールし、一致しない既存の Pod を削除します。

2. クラスターログインのカスタムリソース (CR) の **curation** セクションを編集し、Curator Pod の容認を設定します。

```
curation:
  type: "curator"
  curator:
    tolerations:
      - key: "curator" ①
        operator: "Exists" ②
        effect: "NoExecute" ③
        tolerationSeconds: 6000 ④
```

- ① ノードに追加したキーを指定します。
- ② **Exists** Operator を指定して、**key/value/effect** パラメーターが一致するようにします。
- ③ **NoExecute** effect を指定します。
- ④ オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

この容認は、**oc adm taint** コマンドで作成されるテイントと一致します。この容認のある Pod は、**node1** にスケジュールできます。

7.9.4. 容認を使用したログコレクター Pod 配置の制御

ログインコレクター Pod が実行するノードを確認し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

容認をクラスターログインのカスタムリソース (CR) でログインコレクター Pod に適用し、テイントをノード仕様でノードに適用します。テイントおよび容認を使用すると、Pod がメモリーや CPU などの問題によってエビクトされないようにすることができます。

デフォルトで、ログインコレクター Pod には以下の容認があります。

```
tolerations:
  - key: "node-role.kubernetes.io/master"
    operator: "Exists"
    effect: "NoExecute"
```

前提条件

- クラスターログインおよび Elasticsearch がインストールされている必要があります。

手順

1. 以下のコマンドを使用して、ログインコレクター Pod がログインコレクター Pod をスケジュールする必要のあるノードにテイントを追加します。

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

以下は例になります。

```
$ oc adm taint nodes node1 collector=node:NoExecute
```

この例では、テイントをキー **collector**、値 **node**、およびテイント effect **NoExecute** のある **node1** に配置します。**NoExecute** テイント effect を使用する必要があります。**NoExecute** は、テイントに一致する Pod のみをスケジュールし、一致しない既存の Pod を削除します。

2. クラスターロギングのカスタムリソース (CR) の **collection** セクションを編集して、ロギングコレクター Pod の容認を設定します。

```
collection:
  logs:
    type: "fluentd"
  rsyslog:
    tolerations:
      - key: "collector" ①
        operator: "Exists" ②
        effect: "NoExecute" ③
        tolerationSeconds: 6000 ④
```

- ① ノードに追加したキーを指定します。
- ② **Exists** Operator を指定して、**key/value/effect** パラメーターが一致するようにします。
- ③ **NoExecute** effect を指定します。
- ④ オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

この容認は、**oc adm taint** コマンドで作成されたテイントと一致します。この容認のある Pod は、**node1** にスケジュールできます。

7.9.5. 追加リソース

テイントおよび容認についての詳細は、「[ノードテイントを使用した Pod 配置の制御](#)」を参照してください。

7.10. ログのサードパーティーシステムへの転送

デフォルトで、OpenShift Container Platform クラスターロギングはクラスターロギングカスタムリソースに定義されるデフォルトの内部 Elasticsearch ログストアにログを送信します。

- **Fluentd 転送プロトコルを使用したログの送信**。Configmap を使用して [Fluentd 転送プロトコル](#) を使用し、Fluent 転送プロトコルを受け入れる外部ロギングアグリゲーターにログを安全に送信できます。
- **syslog を使用したログの送信**。Configmap を作成して、[syslog プロトコル](#) を使用してログを外部 syslog (RFC 3164) サーバーに送信できます。

または、現在テクノロジープレビューとして**ログ転送 API**を使用できます。Fluentd プロトコルおよび syslog よりも設定が簡単なログ転送 API は、ログを内部 Elasticsearch ログストアおよび外部の Fluentd ログ集計ソリューションに送信するための設定を公開します。



重要

ログ転送 API はテクノロジープレビュー機能としてのみご利用いただけます。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。

ConfigMap を使用してログを転送する方法は非推奨となり、今後のリリースではログ転送 API に置き換えられます。

7.10.1. Fluentd 転送プロトコルを使用したログの転送

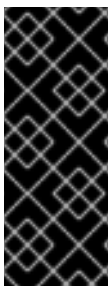
Fluentd **転送** プロトコルを使用して、デフォルトの Elasticsearch ログストアではなく外部のログアグリゲーターにログのコピーを送信できます。OpenShift Container Platform クラスターでは、Fluentd **転送** プロトコルを使用して、このプロトコルを受け入れるように設定されたサーバーにログを送信します。外部ログアグリゲーターを OpenShift Container Platform からログを受信するように設定する必要があります。



注記

ログ転送のこの方法は OpenShift Container Platform では非推奨となり、今後のリリースではログ転送 API に置き換えられます。

Fluentd **転送** プロトコルを使用して OpenShift Container Platform をログを送信するように設定するには、外部ログアグリゲーターを参照する **openshift-logging** namespace の **secure-forward** という ConfigMap を作成します。



重要

OpenShift Container Platform 4.3 以降では、Fluentd **転送** プロトコルを使用するプロセスは変更されています。以下で説明されているように ConfigMap を作成する必要があります。

さらに、設定で必要になる証明書を、Fluentd Pod にマウントされる **secure-forward** という名前のシークレットに追加できます。

secure-forward.conf のサンプル

```
<store>
  @type forward
  <security>
    self_hostname ${hostname} # ${hostname} is a placeholder.
    shared_key "fluent-receiver"
```

```

</security>
transport tls
tls_verify_hostname false      # Set false to ignore server cert hostname.

tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
<buffer>
  @type file
  path '/var/lib/fluentd/secureforwardlegacy'
  queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }"
  chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }"
  flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
  flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
  flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
  retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
  retry_forever true
  # the systemd journald 0.0.8 input plugin will just throw away records if the buffer
  # queue limit is hit - 'block' will halt further reads and keep retrying to flush the
  # buffer to the remote - default is 'exception' because in_tail handles that case
  overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'exception'}"
</buffer>
<server>
  host fluent-receiver.openshift-logging.svc # or IP
  port 24224
</server>
</store>

```

設定に基づく secure-forward ConfigMap のサンプル

```

apiVersion: v1
data:
  secure-forward.conf: "<store>
    \ @type forward
    \ <security>
    \   self_hostname ${hostname} # ${hostname} is a placeholder.
    \   shared_key \"fluent-receiver\"
    \ </security>
    \ transport tls
    \ tls_verify_hostname false      # Set false to ignore server cert hostname.
    \ tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
    \ <buffer>
    \   @type file
    \   path '/var/lib/fluentd/secureforwardlegacy'
    \   queued_chunks_limit_size \"#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }\"
    \   chunk_limit_size \"#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }\"
    \   flush_interval \"#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}\"
    \   flush_at_shutdown \"#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}\"
    \   flush_thread_count \"#{ENV['FLUSH_THREAD_COUNT'] || 2}\"
    \   retry_max_interval \"#{ENV['FORWARD_RETRY_WAIT'] || '300'}\"
    \   retry_forever true
    \   # the systemd journald 0.0.8 input plugin will just throw away records if the buffer
    \   # queue limit is hit - 'block' will halt further reads and keep retrying to flush the
    \   # buffer to the remote - default is 'exception' because in_tail handles that case
    \   overflow_action \"#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'exception'}\"
    \ </buffer>
    \ <server>

```

```

\ host fluent-receiver.openshift-logging.svc # or IP
\ port 24224
\ </server>
</store>"
kind: ConfigMap
metadata:
  creationTimestamp: "2020-01-15T18:56:04Z"
  name: secure-forward
  namespace: openshift-logging
  resourceVersion: "19148"
  selfLink: /api/v1/namespaces/openshift-logging/configmaps/secure-forward
  uid: 6fd83202-93ab-d851b1d0f3e8

```

手順

OpenShift Container Platform を Fluentd **転送** プロトコルを使用してログを転送できるように設定するには、以下を実行します。

1. **転送** パラメーターについて **secure-forward.conf** という名前の設定ファイルを作成します。
 - a. シークレットおよび TLS 情報を設定します。

```

<store>
  @type forward

  self_hostname ${hostname} ①
  shared_key <SECRET_STRING> ②

  transport tls ③

  tls_verify_hostname true ④
  tls_cert_path <path_to_file> ⑤

```

- ① 自動生成される証明書の共通名 (CN) のデフォルト値を指定します。
- ② ノード間で共有キーを入力します。
- ③ **tls** を指定して TLS 検証を有効にします。
- ④ サーバー証明書のホスト名を確認するには **true** に設定します。サーバー証明書のホスト名を無視するには、**false** に設定します。
- ⑤ プライベート CA 証明書ファイルへのパスを **/etc/ocp-forward/ca_cert.pem** として指定します。

mTLS を使用するには、クライアント証明書およびキーパラメーターなどの設定に関する情報として [Fluentd のドキュメント](#) を参照してください。

- b. 外部 Fluentd サーバーの名前、ホスト、およびポートを設定します。

```

<server>
  name ①
  host ②
  hostlabel ③
  port ④

```



```

</server>
<server> 5
  name
  host
</server>

```

- 1 オプションで、このサーバーの名前を入力します。
- 2 サーバーのホスト名または IP を指定します。
- 3 サーバーのホストラベルを指定します。
- 4 サーバーのポートを指定します。
- 5 オプションで、サーバーを追加します。2つ以上のサーバーを指定する場合、**forward** はこれらのサーバーノードをラウンドロビン順で使用します。

以下は例になります。

```

<server>
  name externalserver1
  host 192.168.1.1
  hostlabel externalserver1.example.com
  port 24224
</server>
<server>
  name externalserver2
  host externalserver2.example.com
  port 24224
</server>
</store>

```

2. 設定ファイルから **openshift-logging** namespace に **secure-forward** という名前の ConfigMap を作成します。

```
$ oc create configmap secure-forward --from-file=secure-forward.conf -n openshift-logging
```

3. オプション: レシーバーに必要なシークレットをインポートします。

```
$ oc create secret generic secure-forward --from-file=<arbitrary-name-of-
key1>=cert_file_from_fluentd_receiver --from-
literal=shared_key=value_from_fluentd_receiver
```

以下は例になります。

```
$ oc create secret generic secure-forward --from-file=ca-bundle.crt=ca-for-fluentd-
receiver/ca.crt --from-literal=shared_key=fluentd-receiver
```

4. **fluentd** Pod を更新し、**secure-forward** シークレットおよび **secure-forward** ConfigMap を適用します。

```
$ oc delete pod --selector logging-infra=fluentd
```

- 外部ログアグリゲーターを OpenShift Container Platform からメッセージを安全に受信できるように設定します。

7.10.2. syslog プロトコルを使用したログの転送

syslog プロトコルを使用して、デフォルトの Elasticsearch ログストアではなく外部の syslog サーバーにログのコピーを送信できます。この **syslog** プロトコルについては、以下の点に注意してください。

- RFC 5424 ではなく、syslog プロトコル (RFC 3164) を使用する
- TLS に対応していないため、安全ではない
- Kubernetes メタデータ、systemd データその他のメタデータを提供しない



注記

ログ転送のこの方法は OpenShift Container Platform では非推奨となり、今後のリリースではログ転送 API に置き換えられます。

syslog プロトコルには、以下の 2 つのバージョンがあります。

- **out_syslog**: UDP で通信するバッファなしの実装は、データをバッファせずに結果を即時に書き込みます。
- **out_syslog_buffered**: TCP で通信するバッファの実装は、[データをいくつかのチャンクにバッファリングします](#)。

syslog プロトコルを使用してログ転送を設定するには、ログを転送するために必要な情報を使って **syslog.conf** という設定ファイルを作成します。次に、そのファイルを使用して OpenShift Container Platform がログの転送時に使用する **openshift-logging** namespace の **syslog** という ConfigMap を作成します。syslog サーバーを OpenShift Container Platform からログを受信するように設定する必要があります。



重要

OpenShift Container Platform 4.3 以降では、**syslog** プロトコルを使用するプロセスは変更されています。以下で説明されているように ConfigMap を作成する必要があります。

設定ファイルに別個の **<store>** スタンザを指定して、ログを複数の syslog サーバーに転送できます。

サンプル **syslog.conf**

```
<store>
@type syslog_buffered 1
remote_syslog rsyslogserver.openshift-logging.svc.cluster.local 2
port 514 3
hostname fluentd-4nzfz 4
remove_tag_prefix tag 5
tag_key ident,systemd.u.SYSLOG_IDENTIFIER 6
facility local0 7
severity info 8
```

```
use_record true 9
payload_key message 10
</store>
```

- 1 syslog プロトコル (**syslog** または **syslog_buffered** のいずれか)。
- 2 syslog サーバーの完全修飾ドメイン名 (FQDN) または IP アドレス。
- 3 接続先のポート番号。デフォルトは **514** です。
- 4 syslog サーバーの名前。
- 5 タグからプレフィックスを削除します。デフォルトは "" (空) です。
- 6 syslog キーを設定するためのフィールド。
- 7 syslog ログファシリティまたはソース。
- 8 syslog ログの重大度。
- 9 レコードの重大度とファシリティを使用するかどうかを決定する (ある場合)。
- 10 syslog メッセージのペイロードを設定するためのキー。デフォルトは **message** に設定されます。

サンプル `syslog.conf` をベースとするサンプル `syslog ConfigMap`

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: syslog
  namespace: openshift-logging
data:
  syslog.conf: |
    <store>
    @type syslog_buffered
    remote_syslog syslogserver.openshift-logging.svc.cluster.local
    port 514
    hostname fluentd-4nzfz
    remove_tag_prefix tag
    tag_key ident,systemd.u.SYSLOG_IDENTIFIER
    facility local0
    severity info
    use_record true
    payload_key message
    </store>
```

手順

OpenShift Container Platform が **syslog** プロトコルを使用してログを転送するように設定するには、以下を実行します。

1. **<store>** スタンザ内に以下のパラメーターが含まれる **syslog.conf** という名前の設定ファイルを作成します。
 - a. **syslog** プロトコルタイプを指定します。

-

```
@type syslog_buffered 1
```

- 1 使用するプロトコル (**syslog** または **syslog_buffered** のいずれか) を指定します。

- b. 外部 syslog サーバーの名前、ホスト、およびポートを設定します。

```
remote_syslog <remote> 1  
port <number> 2  
hostname <name> 3
```

- 1 syslog サーバーの FQDN または IP アドレスを指定します。
- 2 syslog サーバーのポートを指定します。
- 3 この syslog サーバーの名前を指定します。

以下は例になります。

```
remote_syslog syslogserver.openshift-logging.svc.cluster.local  
port 514  
hostname fluentd-server
```

- c. 必要に応じて他の syslog 変数を設定します。

```
remove_tag_prefix 1  
tag_key <key> 2  
facility <value> 3  
severity <value> 4  
use_record <value> 5  
payload_key message 6
```

- 1 このパラメーターを追加して、**tag** フィールドを syslog プレフィックスから削除します。
- 2 syslog キーを設定するためのフィールドを指定します。
- 3 syslog ログファシリティまたはソースを指定します。値については、[RTF 3164](#) を参照してください。
- 4 syslog ログの重大度を指定します。値については、[RTF 3164](#) リンクを参照してください。
- 5 **true** を指定して、レコードの重大度およびファシリティを使用します (ある場合)。**true** の場合、**container_name**、**namespace_name**、および **pod_name** は、出力の内容に組み込まれます。
- 6 syslog メッセージのペイロードを設定するためにキーを指定します。デフォルトは **message** に設定されます。

以下は例になります。

```
facility local0
severity info
```

設定ファイルは以下のように表示されます。

```
<store>
@type syslog_buffered
remote_syslog syslogserver.openshift-logging.svc.cluster.local
port 514
hostname fluentd-4nzhz
tag_key ident,systemd.u.SYSLOG_IDENTIFIER
facility local0
severity info
use_record false
</store>
```

2. 設定ファイルから **openshift-logging** namespace に **syslog** という名前の ConfigMap を作成します。

```
$ oc create configmap syslog --from-file=syslog.conf -n openshift-logging
```

Cluster Logging Operator は Fluentd Pod を再デプロイします。Pod が再デプロイされない場合、強制的に再デプロイするために Fluentd Pod を削除できます。

```
$ oc delete pod --selector logging-infra=fluentd
```

7.10.3. ログ転送 API を使用したログの転送

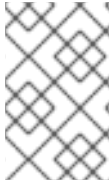
ログ転送 API により、コンテナおよびノードログをクラスター内外の特定のエンドポイントに送信できるようにカスタムパイプラインを設定できます。既存のロギングサービス、外部 Elasticsearch クラスター、外部ログ集計ソリューション、またはセキュリティ情報およびイベント管理 (SIEM) システムなどの OpenShift Container Platform クラスターロギングで管理されていないリモート宛先および内部 OpenShift Container Platform Elasticsearch ログストアにログをタイプ別に送信することができます。

重要

ログ転送 API は現時点で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。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。

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。

異なるタイプのログを異なるシステムに送信して、組織の誰がそれぞれのタイプにアクセスできるかを制御できます。オプションの TLS サポートにより、組織の必要に応じてセキュアな通信を使用してログを送信することができます。

**注記**

ログ転送 API の使用はオプションです。ログを内部の OpenShift Container Platform Elasticsearch インスタンスのみに転送する必要がある場合は、ログ転送 API を設定しないようにしてください。

7.10.3.1. ログ転送 API について

ログ転送 API を使用してクラスターログを転送するには、**outputs** および **pipelines** の組み合わせを使用して、OpenShift Container Platform クラスター内外の特定のエンドポイントにログを送信する必要があります。

**注記**

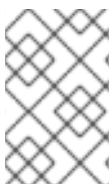
デフォルトの内部 OpenShift Container Platform Elasticsearch ログストアのみを使用する必要がある場合は、ログ転送機能は設定しません。

デフォルトで、クラスターロギング Operator はクラスターロギングカスタムリソースに定義されるデフォルトの内部 Elasticsearch ログストアにログを送信します。ログ転送機能を使用するには、カスタム **logforwarding** 設定ファイルを作成し、指定したエンドポイントにログを送信します。

output はログデータの宛先で、**パイプライン** は単一のソースまたは1つまたは複数の出力の単純なルーティングを定義します。

出力には、以下のいずれかが該当します。

- **elasticsearch** を使用して、外部の Elasticsearch v5.x クラスターおよび/または内部の OpenShift Container Platform Elasticsearch インスタンスにログを転送します。
- **forward** を使用して、ログを外部のログ集計ソリューションに転送します。このオプションは Fluentd **forward** プロトコルを使用します。

**注記**

CIDR アノテーションを使用するクラスター全体のプロキシが有効になっている場合、エンドポイントは IP アドレスではなくサーバー名または FQDN である必要があります。

パイプライン は、データのソースを出力に関連付けます。データのソースは以下のいずれかになります。

- **logs.app**: クラスターで実行される、インフラストラクチャーコンテナアプリケーションを除くユーザーアプリケーションによって生成されるコンテナログ。
- **logs.infra**: ジャーナルログなどの、クラスターで実行されるインフラストラクチャーコンポーネントおよび OpenShift Container Platform ノードで生成されるログ。インフラストラクチャーコンポーネントは、**openshift***、**kube***、または **default** プロジェクトで実行される Pod です。
- **logs.audit**: ノード監査システム (auditd) で生成されるログ (`/var/log/audit/audit.log` ファイルに保存される)、および Kubernetes apiserver および OpenShift apiserver の監査ログ。

以下の点に注意してください。

- 内部 OpenShift Container Platform Elasticsearch インスタンスは、監査ログのセキュアなスト

レージを提供しません。監査ログを転送するシステムが組織および政府の規制に準拠しており、適切にセキュリティーが保護されていることを確認することが推奨されています。OpenShift Container Platform クラスターロギングはこれらの規制に準拠しません。

- 出力は、シークレットを使用する TLS 通信をサポートします。シークレットには、`tls.crt`、`tls.key`、および `ca-bundler.crt` のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。転送機能を安全な方法で使用するには、シークレットにキー `shared_key` が含まれる必要があります。
- キーやシークレット、サービスアカウント、ポートのオープン、またはグローバルプロキシ設定など、外部の宛先で必要となる可能性のある追加設定を作成し、維持する必要があります。

以下の例は、3つの出力を作成します。

- 内部 OpenShift Container Platform Elasticsearch インスタンス
- 非セキュアな外部で管理される Elasticsearch インスタンス
- セキュアな外部ログアグリゲーター (`forward` プロトコルを使用)。

3つのパイプラインは以下を送信します。

- 内部 OpenShift Container Platform Elasticsearch へのアプリケーションログ
- 外部 Elasticsearch インスタンスへのインフラストラクチャーログ
- セキュリティーが保護されたデバイスへの監査ログ (`forward` プロトコルを使用)

ログ転送の出力とパイプラインのサンプル

```
apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance ❶
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true ❷
  outputs: ❸
  - name: elasticsearch ❹
    type: "elasticsearch" ❺
    endpoint: elasticsearch.openshift-logging.svc:9200 ❻
    secret: ❼
      name: fluentd
  - name: elasticsearch-insecure
    type: "elasticsearch"
    endpoint: elasticsearch-insecure.svc.messaging.cluster.local
    insecure: true ❽
  - name: secureforward-offcluster
    type: "forward"
    endpoint: https://secureforward.offcluster.com:24224
    secret:
      name: secureforward
  pipelines: ❾
  - name: container-logs ❿
```

```

inputSource: logs.app 11
outputRefs: 12
- elasticsearch
- secureforward-offcluster
- name: infra-logs
inputSource: logs.infra
outputRefs:
- elasticsearch-insecure
- name: audit-logs
inputSource: logs.audit
outputRefs:
- secureforward-offcluster

```

- 1** ログ転送 CR の名前は **instance** である必要があります。
- 2** デフォルトのログ転送動作を無効にするパラメーター。
- 3** 出力の設定。
- 4** 出力を記述する名前。
- 5** **elasticsearch** または **forward** のいずれかの出力タイプ。
- 6** サーバー名、FQDN、または IP アドレスのいずれかのエンドポイントを入力します。CIDR アノテーションを使用するクラスター全体のプロキシが有効になっている場合、エンドポイントは IP アドレスではなくサーバー名または FQDN である必要があります。内部 OpenShift Container Platform Elasticsearch インスタンスの場合は、**elasticsearch.openshift-logging.svc:9200** を指定します。
- 7** TLS 通信のエンドポイントに必要なシークレットのオプションの名前。シークレットは **openshift-logging** プロジェクトに存在する必要があります。
- 8** エンドポイントがシークレットを使用しない場合のオプションの設定 (これにより、非セキュアな通信が発生します)。
- 9** パイプラインの設定。
- 10** パイプラインを説明する名前。
- 11** データソース: **logs.app**、**logs.infra**、または **logs.audit**。
- 12** CR に設定される単一または複数の出力の名前。

7.10.3.2. ログ転送 API の有効化

API を使用してログを転送する前に、ログ転送 API を有効にする必要があります。

手順

ログ転送 API を有効にするには、以下を実行します。

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```


2. **clusterlogging.openshift.io/logforwardingtechpreview** アノテーションを追加し、**enabled** に設定します。

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  annotations:
    clusterlogging.openshift.io/logforwardingtechpreview: enabled ❶
  name: "instance"
  namespace: "openshift-logging"
spec:
  ...

  collection: ❷
    logs:
      type: "fluentd"
      fluentd: {}

```

- ❶ ログ転送 API を有効および無効にします。ログ転送を使用するには、**enabled** に設定します。OpenShift Container Platform Elasticsearch インスタンスのみを使用するには、**disable** に設定するか、またはアノテーションを追加しません。
- ❷ Fluentd を使用できるように、**spec.collection** ブロックをクラスターロギング CR で定義する必要があります。

7.10.3.3. ログ転送 API を使用したログ転送の設定

ログ転送を設定するには、クラスターロギングのカスタムリソース (CR) を編集して、**clusterlogging.openshift.io/logforwardingtechpreview: enabled** アノテーションを追加し、ログ転送カスタムリソースを作成して出力、パイプラインを指定し、ログ転送を有効にします。

ログ転送を有効にする場合、次の3つのソースタイプのすべてのパイプラインを定義する必要があります (**logs.app**、**logs.infra**、および **logs.audit**)。未定義のソースタイプのログはすべてドロップされます。たとえば、**logs.app** および **log-audit** タイプのパイプラインを指定するものの、**logs.infra** タイプのパイプラインを指定していない場合、**logs.infra** ログがドロップされます。

手順

API を使用してログ転送を設定するには、以下を実行します。

1. 以下のようなログ転送 CR YAML ファイルを作成します。

```

apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  disableDefaultForwarding: true ❸
  outputs: ❹
  - name: elasticsearch
    type: "elasticsearch"
    endpoint: elasticsearch.openshift-logging.svc:9200

```

```

secret:
  name: elasticsearch
- name: elasticsearch-insecure
  type: "elasticsearch"
  endpoint: elasticsearch-insecure.svc.messaging.cluster.local
  insecure: true
- name: secureforward-offcluster
  type: "forward"
  endpoint: https://secureforward.offcluster.com:24224
secret:
  name: secureforward
pipelines: 5
- name: container-logs
  inputSource: logs.app
  outputRefs:
  - elasticsearch
  - secureforward-offcluster
- name: infra-logs
  inputSource: logs.infra
  outputRefs:
  - elasticsearch-insecure
- name: audit-logs
  inputSource: logs.audit
  outputRefs:
  - secureforward-offcluster

```

- 1 ログ転送 CR の名前は **instance** である必要があります。
- 2 ログ転送 CR の namespace は **openshift-logging** である必要があります。
- 3 **true** に設定されると、デフォルトのログ転送動作が無効になります。
- 4 1つ以上のエンドポイントを追加するには、以下を実行します。
 - **elasticsearch** または **forward** のいずれかの出力タイプを指定します。
 - 出力の名前を入力します。
 - サーバー名、FQDN、または IP アドレスのいずれかのエンドポイントを入力します。CIDR アノテーションを使用するクラスター全体のプロキシが有効になっている場合、エンドポイントは IP アドレスではなくサーバー名または FQDN である必要があります。内部 OpenShift Container Platform Elasticsearch インスタンスの場合は、**elasticsearch.openshift-logging.svc:9200** を指定します。
 - オプション: TLS 通信のエンドポイントに必要なシークレットの名前を入力します。シークレットは **openshift-logging** プロジェクトに存在する必要があります。
 - エンドポイントがシークレットを使用しない場合に **insecure: true** を指定します (これにより、非セキュアな通信が発生します)。
- 5 1つ以上のパイプラインを追加します。
 - パイプラインの名前を入力します。
 - ソースタイプ (**logs.app**、**logs.infra**、または **logs.audit**) を指定します。
 - CR に設定された 1つ以上の出力の名前を指定します。



注記

disableDefaultForwarding: true を設定する場合、アプリケーション、インフラストラクチャーおよび監査の3つの種類のログすべてのパイプラインおよび出力を設定する必要があります。ログの種類に対応するパイプラインおよび出力を指定しない場合、それらのログは保存されず、失われます。

2. CR オブジェクトを作成します。

```
$ oc create -f <file-name>.yaml
```

7.10.3.3.1. ログ転送カスタムリソースのサンプル

通常のログ転送設定は以下の例のようになります。

以下のログ転送カスタムリソースは、すべてのログをセキュアな Elasticsearch ログストアに送信します。

Elasticsearch ログストアに転送するカスタムリソースのサンプル

```
apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
  - name: user-created-es
    type: elasticsearch
    endpoint: 'elasticsearch-server.openshift-logging.svc:9200'
    secret:
      name: piplinesecret
  pipelines:
  - name: app-pipeline
    inputSource: logs.app
    outputRefs:
    - user-created-es
  - name: infra-pipeline
    inputSource: logs.infra
    outputRefs:
    - user-created-es
  - name: audit-pipeline
    inputSource: logs.audit
    outputRefs:
    - user-created-es
```

以下のログ転送カスタムリソースは、Fluentd **forward** プロトコルを使用してすべてのログをセキュアな Fluentd インスタンスに送信します。

forward プロトコルを使用するためのサンプルカスタムリソース

```

apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
  - name: fluentd-created-by-user
    type: forward
    endpoint: 'fluentdserver.openshift-logging.svc:24224'
    secret:
      name: fluentdserver
  pipelines:
  - name: app-pipeline
    inputSource: logs.app
    outputRefs:
    - fluentd-created-by-user
  - name: infra-pipeline
    inputSource: logs.infra
    outputRefs:
    - fluentd-created-by-user
  - name: clo-default-audit-pipeline
    inputSource: logs.audit
    outputRefs:
    - fluentd-created-by-user

```

7.10.3.4. ログ転送 API の無効化

ログ転送 API を無効にし、指定されたエンドポイントへのログの転送を停止するには、**metadata.annotations.clusterlogging.openshift.io/logforwardingtechpreview:enabled** パラメーターをクラスターロギング CR から削除してからログ転送 CR を削除します。コンテナおよびノードログは内部 OpenShift Container Platform Elasticsearch インスタンスに転送されます。



注記

disableDefaultForwarding=false を設定すると、クラスターロギングがログを指定されたエンドポイント および デフォルトの内部 OpenShift Container Platform Elasticsearch インスタンスに送信できなくなります。

手順

ログ転送 API を無効にするには、以下を実行します。

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```

2. **clusterlogging.openshift.io/logforwardingtechpreview** アノテーションを削除します。

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:

```

```

annotations:
  clusterlogging.openshift.io/logforwardingtechpreview: enabled ❶
name: "instance"
namespace: "openshift-logging"
....

```

- ❶ このアノテーションを削除します。

3. ログ転送カスタムリソースを削除します。

```
$ oc delete LogForwarding instance -n openshift-logging
```

7.11. SYSTEMD-JOURNALD および FLUENTD の設定

Fluentd のジャーナルからの読み取りや、ジャーナルのデフォルト設定値は非常に低く、ジャーナルがシステムサービスからのロギング速度に付いていくことができないためにジャーナルエントリーが失われる可能性があります。

ジャーナルでエントリーが失われるのを防ぐことができるように **RateLimitInterval=1s** および **RateLimitBurst=10000** (必要な場合はさらに高い値) を設定することが推奨されます。

7.11.1. クラスターロギング用の systemd-journald の設定

プロジェクトのスケールアップ時に、デフォルトのロギング環境にはいくらかの調整が必要になる場合があります。

たとえば、ログが見つからない場合は、journald の速度制限を引き上げる必要がある場合があります。一定期間保持するメッセージ数を調整して、クラスターロギングがログをドロップせずに過剰なリソースを使用しないようにすることができます。

また、ログを圧縮する必要があるかどうか、ログを保持する期間、ログを保存する方法、ログを保存するかどうかやその他の設定を決定することもできます。

手順

1. 必要な設定で **journald.conf** ファイルを作成します。

```

Compress=yes ❶
ForwardToConsole=no ❷
ForwardToSyslog=no
MaxRetentionSec=1month ❸
RateLimitBurst=10000 ❹
RateLimitInterval=1s
Storage=persistent ❺
SyncIntervalSec=1s ❻
SystemMaxUse=8g ❼
SystemKeepFree=20% ❽
SystemMaxFileSize=10M ❾

```

- ❶ ログがファイルシステムに書き込まれる前にそれらのログを圧縮するかどうかを指定します。 **yes** を指定してメッセージを圧縮するか、または **no** を指定して圧縮しないようにします。デフォルトは **yes** です。

- 2 ログメッセージを転送するかどうかを設定します。それぞれについて、デフォルトで **no** に設定されます。以下を指定します。
 - **ForwardToConsole**: ログをシステムコンソールに転送します。
 - **ForwardToKsmg**: ログをカーネルログバッファに転送します。
 - **ForwardToSyslog**: syslog デーモンに転送します。
 - **ForwardToWall**: メッセージを wall メッセージとしてすべてのログインしているユーザーに転送します。
- 3 ジャーナルエントリを保存する最大時間を指定します。数字を入力して秒数を指定します。または、「year」、「month」、「week」、「day」、「h」または「m」などの単位を含めます。無効にするには **0** を入力します。デフォルトは **1month** です。
- 4 レート制限を設定します。**RateLimitIntervalSec** で定義される期間に、**RateLimitBurst** で指定される以上のログが受信される場合、この期間内の追加のメッセージすべてはこの期間が終了するまでにドロップされます。デフォルト値である **RateLimitInterval=1s** および **RateLimitBurst=10000** を設定することが推奨されます。
- 5 ログの保存方法を指定します。デフォルトは **persistent** です。
 - **volatile**: ログを **/var/log/journal/** のメモリーに保存します。
 - **persistent**: ログを **/var/log/journal/** のディスクに保存します。systemd は存在しない場合はディレクトリを作成します。
 - **auto**: ディレクトリが存在する場合に、ログを **/var/log/journal/** に保存します。存在しない場合は、systemd はログを **/run/systemd/journal** に一時的に保存します。
 - **none**: ログを保存しません。systemd はすべてのログをドロップします。
- 6 **ERR**、**WARNING**、**NOTICE**、**INFO**、および **DEBUG** ログについてジャーナルファイルをディスクに同期させるまでのタイムアウトを指定します。systemd は、**CRIT**、**ALERT**、または **EMERG** ログの受信後すぐに同期を開始します。デフォルトは **1s** です。
- 7 ジャーナルが使用できる最大サイズを指定します。デフォルトは **8g** です。
- 8 systemd が残す必要のあるディスク領域のサイズを指定します。デフォルトは **20%** です。
- 9 **/var/log/journal** に永続的に保存される個別のジャーナルファイルの最大サイズを指定します。デフォルトは **10M** です。



注記

レート制限を削除する場合、システムロギングデーモンの CPU 使用率が高くなる可能性があります。以前はスロットルされていた可能性のあるメッセージが処理されるためです。

systemd 設定の詳細について

は、<https://www.freedesktop.org/software/systemd/man/journald.conf.html> を参照してください。このページに一覧表示されるデフォルト設定は OpenShift Container Platform には適用されない可能性があります。

2. **journal.conf** ファイルを base64 に変換します。

```
$ export jrnl_cnf=$( cat /journald.conf | base64 -w0 )
```

3. マスターまたはワーカー用に新規の MachineConfig を作成し、**journal.conf** パラメーターを追加します。
以下は例になります。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 50-corp-journald
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,${jrnl_cnf}
          mode: 0644 ①
          overwrite: true
          path: /etc/systemd/journald.conf ②
```

- ① **journal.conf** ファイルのパーミッションを設定します。0644 パーミッションを設定することが推奨されます。

- ② base64 でエンコードされた **journal.conf** ファイルへのパスを指定します。

4. MachineConfig を作成します。

```
$ oc apply -f <filename>.yaml
```

コントローラーは新規の MachineConfig を検出し、新規の **rendered-worker-<hash>** バージョンを生成します。

5. 新規のレンダリングされた設定の各ノードへのロールアウトのステータスをモニターします。

```
$ oc describe machineconfigpool/worker
```

```
Name:      worker
Namespace:
Labels:    machineconfiguration.openshift.io/mco-built-in=
Annotations: <none>
API Version: machineconfiguration.openshift.io/v1
Kind:      MachineConfigPool
...
Conditions:
```

Message:

Reason: All nodes are updating to rendered-worker-913514517bcea7c93bd446f4830bc64e

第8章 ELASTICSEARCH ステータスの表示

Elasticsearch Operator のステータスや、数多くの Elasticsearch コンポーネントを表示できます。

8.1. ELASTICSEARCH ステータスの表示

Elasticsearch クラスターのステータスを表示することができます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトに切り替えます。

```
$ oc project openshift-logging
```

2. Elasticsearch クラスターのステータスを表示するには、以下を実行します。

- a. Elasticsearch インスタンスの名前を取得します。

```
$ oc get Elasticsearch
```

```
NAME          AGE
elasticsearch 5h9m
```

- b. Elasticsearch のステータスを取得します。

```
$ oc get Elasticsearch <Elasticsearch-instance> -o yaml
```

以下は例になります。

```
$ oc get Elasticsearch elasticsearch -n openshift-logging -o yaml
```

出力には、以下のような情報が含まれます。

```
status: 1
cluster: 2
  activePrimaryShards: 30
  activeShards: 60
  initializingShards: 0
  numDataNodes: 3
  numNodes: 3
  pendingTasks: 0
  relocatingShards: 0
  status: green
  unassignedShards: 0
  clusterHealth: ""
  conditions: [] 3
  nodes: 4
- deploymentName: elasticsearch-cdm-zjf34ved-1
  upgradeStatus: {}
```

```

- deploymentName: elasticsearch-cdm-zjf34ved-2
  upgradeStatus: {}
- deploymentName: elasticsearch-cdm-zjf34ved-3
  upgradeStatus: {}
pods: 5
client:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
data:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
master:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
shardAllocationEnabled: all

```

- 1 出力の **status** スタンザに、クラスターステータスのフィールドが表示されます。
- 2 Elasticsearch クラスターのステータス:
 - アクティブなプライマリーシャードの数
 - アクティブなシャードの数
 - 初期化されるシャードの数
 - Elasticsearch データノードの数
 - Elasticsearch ノードの合計数
 - 保留中のタスクの数
 - Elasticsearch のステータス: **green**、**red**、**yellow**
 - 未割り当てのシャードの数
- 3 ステータスの状況 (ある場合)。Elasticsearch クラスターのステータスは、Pod が配置されていない場合にスケジューラーからの理由を示します。以下の状況に関連したイベントが表示されます。
 - Elasticsearch およびプロキシコンテナの両方についてコンテナが待機中。
 - Elasticsearch およびプロキシコンテナの両方についてコンテナが終了している。

- Pod がスケジューリング対象外である。さらに多数の問題についての状況が表示され
ます。詳細は、**状況メッセージのサンプル** を参照してください。

- 4 **upgradeStatus** のクラスター内の Elasticsearch ノード。
- 5 'failed'、**notReady** または **ready** 状態で一覧表示される、クラスターの Elasticsearch クライアント、データおよびマスター Pod。

8.1.1. 状況メッセージのサンプル

以下は、Elasticsearch インスタンスの **Status** セクションからの一部の状態メッセージの例になります。

このステータスメッセージは、ノードが設定された低基準値を超えており、シャードがこのノードに割り当てられないことを示します。

```
status:
  nodes:
    - conditions:
      - lastTransitionTime: 2019-03-15T15:57:22Z
        message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
          be allocated on this node.
        reason: Disk Watermark Low
        status: "True"
        type: NodeStorage
      deploymentName: example-elasticsearch-cdm-0-1
      upgradeStatus: {}
```

このステータスメッセージは、ノードが設定された高基準値を超えており、シャードが他のノードに移動させられることを示します。

```
status:
  nodes:
    - conditions:
      - lastTransitionTime: 2019-03-15T16:04:45Z
        message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated
          from this node.
        reason: Disk Watermark High
        status: "True"
        type: NodeStorage
      deploymentName: example-elasticsearch-cdm-0-1
      upgradeStatus: {}
```

このステータスメッセージは、CR の Elasticsearch ノードセレクターがクラスターのいずれのノードにも一致しないことを示します。

```
status:
  nodes:
    - conditions:
      - lastTransitionTime: 2019-04-10T02:26:24Z
        message: '0/8 nodes are available: 8 node(s) didn't match node selector.'
        reason: Unschedulable
        status: "True"
        type: Unschedulable
```

このステータスメッセージは、Elasticsearch CR が存在しない PVC を使用することを示します。

```
status:
  nodes:
  - conditions:
    - last Transition Time: 2019-04-10T05:55:51Z
      message: pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
      reason: Unschedulable
      status: True
      type: Unschedulable
```

このステータスメッセージは、Elasticsearch クラスターには Elasticsearch の冗長性ポリシーをサポートするための十分なノードがないことを示します。

```
status:
  clusterHealth: ""
  conditions:
  - lastTransitionTime: 2019-04-17T20:01:31Z
    message: Wrong RedundancyPolicy selected. Choose different RedundancyPolicy or
      add more nodes with data roles
    reason: Invalid Settings
    status: "True"
    type: InvalidRedundancy
```

このステータスメッセージは、クラスター内のマスターノードの数が多過ぎることを示します。

```
status:
  clusterHealth: green
  conditions:
  - lastTransitionTime: '2019-04-17T20:12:34Z'
    message: >-
      Invalid master nodes count. Please ensure there are no more than 3 total
      nodes with master roles
    reason: Invalid Settings
    status: 'True'
    type: InvalidMasters
```

8.2. ELASTICSEARCH コンポーネントのステータスの表示

数多くの Elasticsearch コンポーネントのステータスを表示することができます。

Elasticsearch インデックス

Elasticsearch インデックスのステータスを表示することができます。

1. Elasticsearch Pod の名前を取得します。

```
$ oc get pods --selector component=elasticsearch -o name

pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. インデックスのステータスを取得します。

```
$ oc exec elasticsearch-cdm-1godmszn-1-6f8495-vp4lw -- indices

Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw -n openshift-logging'
to see all of the containers in this pod.
Wed Apr 10 05:42:12 UTC 2019
health status index                uuid                pri rep docs.count
docs.deleted store.size pri.store.size
red open .kibana.647a750f1787408bf50088234ec0edd5a6a9b2ac
N7iCbRjSSc2bGhn8Cpc7Jg 2 1
green open .operations.2019.04.10          GTewEJEzQjaus9QjvBBnGg 3 1
2176114 0 3929 1956
green open .operations.2019.04.11          ausZHoKxTNOoBvv9RIXfrw 3 1
1494624 0 2947 1475
green open .kibana                      9Fltn1D0QHsnFMXpphZ--Q 1 1 1
0 0 0
green open .searchguard                  chOwDnQlSsqhfSPcot1Yiw 1 1
5 1 0 0
```

Elasticsearch Pod

Elasticsearch Pod のステータスを表示することができます。

1. Pod の名前を取得します。

```
$ oc get pods --selector component=elasticsearch -o name

pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Pod のステータスを取得します。

```
oc describe pod elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
```

出力には、以下のようなステータス情報が含まれます。

```
....
Status:      Running
....

Containers:
  elasticsearch:
    Container ID:  cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
    State:          Running
    Started:        Mon, 08 Apr 2019 10:17:56 -0400
    Ready:          True
    Restart Count:  0
    Readiness:      exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
                   period=5s #success=1 #failure=3
....

proxy:
```

```

Container ID: cri-
o://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
State:      Running
  Started:   Mon, 08 Apr 2019 10:18:38 -0400
Ready:      True
Restart Count: 0

```

....

```

Conditions:
Type      Status
Initialized True
Ready     True
ContainersReady True
PodScheduled True

```

....

```
Events:      <none>
```

Elasticsearch デプロイメント設定

Elasticsearch デプロイメント設定のステータスを表示することができます。

1. デプロイメント設定の名前を取得します。

```

$ oc get deployment --selector component=elasticsearch -o name

deployment.extensions/elasticsearch-cdm-1gon-1
deployment.extensions/elasticsearch-cdm-1gon-2
deployment.extensions/elasticsearch-cdm-1gon-3

```

2. デプロイメント設定のステータスを取得します。

```
$ oc describe deployment elasticsearch-cdm-1gon-1
```

出力には、以下のようなステータス情報が含まれます。

```

....
Containers:
  elasticsearch:
    Image:      registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
                period=5s #success=1 #failure=3

```

....

```

Conditions:
Type      Status Reason
----      -
Progressing Unknown DeploymentPaused
Available True MinimumReplicasAvailable

```

```
....
```

```
Events:      <none>
```

Elasticsearch ReplicaSet

Elasticsearch ReplicaSet のステータスを表示することができます。

1. レプリカセットの名前を取得します。

```
$ oc get replicaSet --selector component=elasticsearch -o name
```

```
replicaset.extensions/elasticsearch-cdm-1gon-1-6f8495  
replicaset.extensions/elasticsearch-cdm-1gon-2-5769cf  
replicaset.extensions/elasticsearch-cdm-1gon-3-f66f7d
```

2. レプリカセットのステータスを取得します。

```
$ oc describe replicaSet elasticsearch-cdm-1gon-1-6f8495
```

出力には、以下のようなステータス情報が含まれます。

```
....
```

```
Containers:
```

```
  elasticsearch:
```

```
    Image:   registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3
```

```
    Readiness:  exec ["/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s  
    period=5s #success=1 #failure=3
```

```
....
```

```
Events:      <none>
```

第9章 クラスタロギングのステータス表示

Cluster Logging Operator のステータスや、数多くのクラスタロギングコンポーネントを表示できます。

9.1. CLUSTER LOGGING OPERATOR のステータス表示

Cluster Logging Operator のステータスを表示することができます。

前提条件

- クラスタロギングおよび Elasticsearch がインストールされている必要があります。

手順

- openshift-logging** プロジェクトに切り替えます。

```
$ oc project openshift-logging
```

- クラスタロギングのステータスを表示するには、以下を実行します。
 - クラスタロギングのステータスを取得します。

```
$ oc get clusterlogging instance -o yaml
```

出力には、以下のような情報が含まれます。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
...
status: ❶
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd ❷
        nodes:
          fluentd-2rhqp: ip-10-0-169-13.ec2.internal
          fluentd-6fgjh: ip-10-0-165-244.ec2.internal
          fluentd-6l2ff: ip-10-0-128-218.ec2.internal
          fluentd-54nx5: ip-10-0-139-30.ec2.internal
          fluentd-flpnn: ip-10-0-147-228.ec2.internal
          fluentd-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
            - fluentd-2rhqp
            - fluentd-54nx5
            - fluentd-6fgjh
            - fluentd-6l2ff
            - fluentd-flpnn
            - fluentd-n2frh
```



```
curation: ③
  curatorStatus:
    - cronJobs: curator
      schedules: 30 3 * * *
      suspended: false
logstore: ④
  elasticsearchStatus:
    - ShardAllocationEnabled: all
      cluster:
        activePrimaryShards: 5
        activeShards: 5
        initializingShards: 0
        numDataNodes: 1
        numNodes: 1
        pendingTasks: 0
        relocatingShards: 0
        status: green
        unassignedShards: 0
      clusterName: elasticsearch
      nodeConditions:
        elasticsearch-cdm-mkkdys93-1:
          nodeCount: 1
      pods:
        client:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        data:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        master:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  visualization: ⑤
    kibanaStatus:
      - deployment: kibana
      pods:
        failed: []
        notReady: []
        ready:
          - kibana-7fb4fd4cc9-f2nls
      replicaSets:
        - kibana-7fb4fd4cc9
      replicas: 1
```

- ① 出力の **status** スタンザに、クラスターステータスのフィールドが表示されます。
- ② Fluentd Pod についての情報
- ③ Curator Pod についての情報

- 4 Elasticsearch クラスターの健全性 (**green**、**yellow**、または **red**) などの Elasticsearch Pod についての情報
- 5 Kibana Pod についての情報

9.1.1. 状態メッセージ (condition message) のサンプル

以下は、クラスターロギングインスタンスの **Status.Nodes** セクションからの一部の状態メッセージの例です。

以下のようなステータスメッセージは、ノードが設定された低基準値を超えており、シャードがこのノードに割り当てられないことを示します。

```
nodes:
- conditions:
- lastTransitionTime: 2019-03-15T15:57:22Z
  message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
    be allocated on this node.
  reason: Disk Watermark Low
  status: "True"
  type: NodeStorage
  deploymentName: example-elasticsearch-clientdatamaster-0-1
  upgradeStatus: {}
```

以下のようなステータスメッセージは、ノードが設定された高基準値を超えており、シャードが他のノードに移動させられることを示します。

```
nodes:
- conditions:
- lastTransitionTime: 2019-03-15T16:04:45Z
  message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated
    from this node.
  reason: Disk Watermark High
  status: "True"
  type: NodeStorage
  deploymentName: cluster-logging-operator
  upgradeStatus: {}
```

以下のようなステータスメッセージは、CR の Elasticsearch ノードセクターがクラスターのいずれのノードにも一致しないことを示します。

```
Elasticsearch Status:
Shard Allocation Enabled: shard allocation unknown
Cluster:
  Active Primary Shards: 0
  Active Shards:        0
  Initializing Shards:  0
  Num Data Nodes:      0
  Num Nodes:           0
  Pending Tasks:       0
  Relocating Shards:   0
  Status:               cluster health unknown
  Unassigned Shards:   0
Cluster Name:          elasticsearch
```

```

Node Conditions:
elasticsearch-cdm-mkkdys93-1:
  Last Transition Time: 2019-06-26T03:37:32Z
  Message:          0/5 nodes are available: 5 node(s) didn't match node selector.
  Reason:           Unschedulable
  Status:           True
  Type:             Unschedulable
elasticsearch-cdm-mkkdys93-2:
Node Count: 2
Pods:
Client:
Failed:
Not Ready:
  elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
  elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
Ready:
Data:
Failed:
Not Ready:
  elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
  elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
Ready:
Master:
Failed:
Not Ready:
  elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
  elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
Ready:

```

以下のようなステータスメッセージは、要求された PVC が PV にバインドされないことを示します。

```

Node Conditions:
elasticsearch-cdm-mkkdys93-1:
  Last Transition Time: 2019-06-26T03:37:32Z
  Message:          pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
  Reason:           Unschedulable
  Status:           True
  Type:             Unschedulable

```

以下のようなステータスメッセージは、ノードセクターがいずれのノードにも一致しないため、Curator Pod をスケジュールできないことを示します。

```

Curation:
Curator Status:
Cluster Condition:
curator-1561518900-cjx8d:
  Last Transition Time: 2019-06-26T03:20:08Z
  Reason:           Completed
  Status:           True
  Type:             ContainerTerminated
curator-1561519200-zqxxj:
  Last Transition Time: 2019-06-26T03:20:01Z
  Message:          0/5 nodes are available: 1 Insufficient cpu, 5 node(s) didn't match node
selector.
  Reason:           Unschedulable

```

```
Status:      True
Type:        Unschedulable
Cron Jobs:   curator
Schedules:   */5 * * * *
Suspended:   false
```

以下のようなステータスメッセージは、ノードセクターがいずれのノードにも一致しないため、Fluentd Pod をスケジュールできないことを示します。

```
Status:
Collection:
Logs:
Fluentd Status:
  Daemon Set: fluentd
Nodes:
Pods:
  Failed:
  Not Ready:
  Ready:
```

9.2. クラスターロギングコンポーネントのステータスの表示

数多くのクラスターロギングコンポーネントのステータスを表示することができます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトに切り替えます。

```
$ oc project openshift-logging
```

2. クラスターロギングデプロイメントのステータスを表示します。

```
$ oc describe deployment cluster-logging-operator
```

出力には、以下のようなステータス情報が含まれます。

```
Name:          cluster-logging-operator
...

Conditions:
  Type          Status Reason
  ----          -
  Available     True   MinimumReplicasAvailable
  Progressing   True   NewReplicaSetAvailable
...

Events:
```

```

Type Reason      Age From      Message
---- -
Normal ScalingReplicaSet 62m deployment-controller Scaled up replica set cluster-logging-operator-574b8987df to 1----

```

3. クラスターロギング ReplicaSet のステータスを表示します。

- a. ReplicaSet の名前を取得します。

```

$ oc get replicaset
NAME                                DESIRED CURRENT READY AGE
cluster-logging-operator-574b8987df 1        1        1    159m
elasticsearch-cdm-uhr537yu-1-6869694fb 1        1        1    157m
elasticsearch-cdm-uhr537yu-2-857b6d676f 1        1        1    156m
elasticsearch-cdm-uhr537yu-3-5b6fdd8cfd 1        1        1    155m
kibana-5bd5544f87                    1        1        1    157m

```

- b. ReplicaSet のステータスを取得します。

```
$ oc describe replicaset cluster-logging-operator-574b8987df
```

出力には、以下のようなステータス情報が含まれます。

```

Name:      cluster-logging-operator-574b8987df
...

Replicas:  1 current / 1 desired
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
...

Events:
Type Reason      Age From      Message
---- -
Normal SuccessfulCreate 66m replicaset-controller Created pod: cluster-logging-operator-574b8987df-qjhhq-----

```

第10章 ノードセレクターを使用したクラスターロギングリソースの移動

ノードセレクターを使用して Elasticsearch、Kibana、Curator Pod を異なるノードにデプロイします。

10.1. クラスターロギングリソースの移動

すべてのクラスターロギングコンポーネント、Elasticsearch、Kibana、および Curator の Pod を異なるノードにデプロイするように Cluster Logging Operator を設定できます。Cluster Logging Operator Pod については、インストールされた場所から移動することはできません。

たとえば、Elasticsearch Pod の CPU、メモリーおよびディスクの要件が高いために、この Pod を別のノードに移動できます。



注記

MachineSet を 6 つ以上のレプリカを使用するように設定する必要があります。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされています。これらの機能はデフォルトでインストールされません。

手順

1. **openshift-logging** プロジェクトでクラスターロギングのカスタムリソースを編集します。

```
$ oc edit ClusterLogging instance

apiVersion: logging.openshift.io/v1
kind: ClusterLogging

....

spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: "
      resources: null
      schedule: 30 3 * * *
      type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: "
      redundancyPolicy: SingleRedundancy
```

```

resources:
  limits:
    cpu: 500m
    memory: 16Gi
  requests:
    cpu: 500m
    memory: 16Gi
  storage: {}
type: elasticsearch
managementState: Managed
visualization:
  kibana:
    nodeSelector: ③
      node-role.kubernetes.io/infra: " ④
    proxy:
      resources: null
      replicas: 1
      resources: null
      type: kibana
....

```

- ① ② ③ ④ 適切な値が設定された **nodeSelector** パラメーターを、移動する必要のあるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。

検証手順

コンポーネントが移動したことを確認するには、**oc get pod -o wide** コマンドを使用できます。

以下は例になります。

- Kibana Pod を **ip-10-0-147-79.us-east-2.compute.internal** ノードから移動する必要がある場合、以下を実行します。

```

$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
NAME                                READY STATUS RESTARTS AGE IP          NODE
NOMINATED NODE READINESS GATES
kibana-5b8bdf44f9-ccpq9 2/2   Running 0      27s 10.129.2.18 ip-10-0-147-79.us-east-2.compute.internal <none> <none>

```

- Kibana Pod を、専用インフラストラクチャーノードである **ip-10-0-139-48.us-east-2.compute.internal** ノードに移動する必要がある場合、以下を実行します。

```

$ oc get nodes
NAME                                STATUS ROLES    AGE VERSION
ip-10-0-133-216.us-east-2.compute.internal Ready  master    60m v1.16.2
ip-10-0-139-146.us-east-2.compute.internal Ready  master    60m v1.16.2
ip-10-0-139-192.us-east-2.compute.internal Ready  worker    51m v1.16.2
ip-10-0-139-241.us-east-2.compute.internal Ready  worker    51m v1.16.2
ip-10-0-147-79.us-east-2.compute.internal Ready  worker    51m v1.16.2
ip-10-0-152-241.us-east-2.compute.internal Ready  master    60m v1.16.2
ip-10-0-139-48.us-east-2.compute.internal Ready  infra     51m v1.16.2

```

ノードには **node-role.kubernetes.io/infra: "** ラベルがあることに注意してください。

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml

kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: "
  ...
```

- Kibana Pod を移動するには、クラスターロギング CR を編集してノードセレクターを追加します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

visualization:
  kibana:
    nodeSelector: ❶
    node-role.kubernetes.io/infra: " ❷
  proxy:
    resources: null
  replicas: 1
  resources: null
  type: kibana
```

- ❶ ❷ ノード仕様のラベルに一致するノードセレクターを追加します。

- CR を保存した後に、現在の Kibana Pod は終了し、新規 Pod がデプロイされます。

```
$ oc get pods

NAME                                READY STATUS    RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7  1/1 Running      0      29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg  2/2 Running      0      28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj  2/2 Running      0      28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78  2/2 Running      0      28m
fluentd-42dzz                            1/1 Running      0      28m
fluentd-d74rq                             1/1 Running      0      28m
fluentd-m5vr9                             1/1 Running      0      28m
fluentd-nkx17                             1/1 Running      0      28m
fluentd-pdvqb                             1/1 Running      0      28m
fluentd-tflh6                             1/1 Running      0      28m
kibana-5b8bdf44f9-ccpq9                   2/2 Terminating 0      4m11s
kibana-7d85dcffc8-bfpfp                   2/2 Running      0      33s
```


- 新規 Pod が **ip-10-0-139-48.us-east-2.compute.internal** ノードに置かれます。

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
NAME                READY STATUS   RESTARTS AGE IP          NODE
NOMINATED NODE READINESS GATES
kibana-7d85dcffc8-bfpfp 2/2 Running 0      43s 10.131.0.22 ip-10-0-139-48.us-east-2.compute.internal <none> <none>
```

- しばらくすると、元の Kibana Pod が削除されます。

```
$ oc get pods
NAME                                READY STATUS   RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7 1/1 Running 0      30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg 2/2 Running 0      29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj 2/2 Running 0      29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78 2/2 Running 0      29m
fluentd-42dzz                          1/1 Running 0      29m
fluentd-d74rq                          1/1 Running 0      29m
fluentd-m5vr9                          1/1 Running 0      29m
fluentd-nkx17                          1/1 Running 0      29m
fluentd-pdvqb                          1/1 Running 0      29m
fluentd-tflh6                          1/1 Running 0      29m
kibana-7d85dcffc8-bfpfp                2/2 Running 0      62s
```

第11章 ELASTICSEARCH の手動によるロールアウト

OpenShift Container Platform は、Elasticsearch クラスターのローリング再起動をサポートします。ローリング再起動は、ダウンタイムなしに適切な変更を Elasticsearch クラスターに適用します (3つのマスターが設定される場合)。Elasticsearch クラスターは、ノードが1回に1度ずつオフラインにされる間オンラインのままとなり、運用可能な状態になります。

11.1. ELASTICSEARCH クラスターのローリング再起動の実行

elasticsearch configmap または **elasticsearch-*** デプロイメント設定のいずれかを変更する際にローリング再起動を実行します。

さらにローリング再起動は、Elasticsearch Pod が実行されるノードで再起動が必要な場合に推奨されます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

クラスターのローリング再起動を実行するには、以下を実行します。

1. **openshift-logging** プロジェクトに切り替えます。

```
$ oc project openshift-logging
```

2. 以下のコマンドを使用して Elasticsearch から CA 証明書を抽出し、**admin-ca** ファイルに書き込みます。

```
$ oc extract secret/elasticsearch --to=. --keys=admin-ca
admin-ca
```

3. シャードの同期フラッシュを実行して、シャットダウン前にディスクへの書き込みを待機している保留中の操作がないようにします。

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- curl -s --cacert
/etc/elasticsearch/secret/admin-ca --cert /etc/elasticsearch/secret/admin-cert --key
/etc/elasticsearch/secret/admin-key -XPOST 'https://localhost:9200/_flush/synced'
```

以下は例になります。

```
oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -- curl -s --cacert
/etc/elasticsearch/secret/admin-ca --cert /etc/elasticsearch/secret/admin-cert --key
/etc/elasticsearch/secret/admin-key -XPOST 'https://localhost:9200/_flush/synced'
```

4. OpenShift Container Platform **es_util** ツールを使用して、ノードを意図的に停止する際のシャードのバランスを防ぎます。

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query=_cluster/settings -
XPUT 'https://localhost:9200/_cluster/settings' -d '{ "transient": {
"cluster.routing.allocation.enable" : "none" } }'
```

以下は例になります。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/settings?pretty=true -XPUT 'https://localhost:9200/_cluster/settings' -d '{
"transient": { "cluster.routing.allocation.enable" : "none" } }'

{
  "acknowledged" : true,
  "persistent" : { },
  "transient" : {
    "cluster" : {
      "routing" : {
        "allocation" : {
          "enable" : "none"
        }
      }
    }
  }
}
```

5. 完了したら、ES クラスターのそれぞれのデプロイメントについて、以下を実行します。

- a. デフォルトで、OpenShift Container Platform Elasticsearch クラスターはノードのロールアウトをブロックします。以下のコマンドを使用してロールアウトを許可し、Pod が変更を取得できるようにします。

```
$ oc rollout resume deployment/<deployment-name>
```

以下は例になります。

```
$ oc rollout resume deployment/elasticsearch-cdm-0-1
deployment.extensions/elasticsearch-cdm-0-1 resumed
```

新規 Pod がデプロイされます。Pod が準備状態のコンテナを持つと、次のデプロイメントに進むことができます。

```
$ oc get pods | grep elasticsearch-*

NAME                                READY STATUS RESTARTS AGE
elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k  2/2   Running 0      22h
elasticsearch-cdm-5ceex6ts-2-f799564cb-l9mj7  2/2   Running 0      22h
elasticsearch-cdm-5ceex6ts-3-585968dc68-k7kjr  2/2   Running 0      22h
```

- b. 完了したら、ロールアウトを許可しないように Pod をリセットします。

```
$ oc rollout pause deployment/<deployment-name>
```

以下は例になります。

```
$ oc rollout pause deployment/elasticsearch-cdm-0-1
deployment.extensions/elasticsearch-cdm-0-1 paused
```

- c. Elasticsearch クラスターが **green** 状態にあることを確認します。

■

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query=_cluster/health?pretty=true
```



注記

直前のコマンドで使用した Elasticsearch Pod でロールアウトを実行した場合、Pod は存在しなくなっているため、ここで新規 Pod 名が必要になります。

以下は例になります。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/health?pretty=true

{
  "cluster_name" : "elasticsearch",
  "status" : "green", ①
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 8,
  "active_shards" : 16,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 1,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

① 次に進む前に、このパラメーターが **green** であることを確認します。

- Elasticsearch 設定マップを変更した場合、それぞれの Elasticsearch Pod についてこれらの手順を繰り返します。
- クラスターのすべてのデプロイメントがロールアウトされたら、シャードのバランシングを再度有効にします。

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query=_cluster/settings -
XPUT 'https://localhost:9200/_cluster/settings' -d '{"transient": {
"cluster.routing.allocation.enable" : "none" } }'
```

以下は例になります。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/settings?pretty=true -XPUT 'https://localhost:9200/_cluster/settings' -d '{
"transient": { "cluster.routing.allocation.enable" : "all" } }'

{
  "acknowledged" : true,
  "persistent" : { },
}
```

```
"transient" : {  
  "cluster" : {  
    "routing" : {  
      "allocation" : {  
        "enable" : "all"  
      }  
    }  
  }  
}  
}
```

第12章 KIBANA のトラブルシューティング

OpenShift Container Platform で Kibana コンソールを使用することで発生する可能性がある問題は簡単に解決できますが、この場合役に立つエラーメッセージは表示されません。OpenShift Container Platform に Kibana をデプロイする際に問題が発生した場合は、以下のトラブルシューティングセクションを確認してください。

12.1. KUBERNETES ログインループのトラブルシューティング

Kibana コンソールの OAuth2 プロキシはマスターホストの OAuth2 サーバーとシークレットを共有する必要があります。シークレットが両方のサーバーで同一でない場合はログインループが発生する可能性があります、Kibana のログインページに継続的にリダイレクトされます。

手順

この問題を修正するには、以下を実行します。

1. 以下のコマンドを実行し、現在の OAuthClient を削除します。

```
$ oc delete oauthclient/kibana-proxy
```

12.2. KIBANA コンソール表示時の KUBERNETES の不明なエラーのトラブルシューティング

Kibana コンソールにアクセスしようとする際に、以下のブラウザーエラーが表示される場合があります。

```
{"error":"invalid_request","error_description":"The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."}
```

このエラーは、OAuth2 クライアントとサーバー間の不一致が原因で発生します。ログイン後にサーバーが安全にリダイレクトできるように、クライアントのリターンアドレスがホワイトリストで指定されている必要があります。

この問題を修正するには、OAuthClient エントリーを置き換えます。

手順

OAuthClient エントリーを置き換えるには、以下を実行します。

1. 以下のコマンドを実行し、現在の OAuthClient を削除します。

```
$ oc delete oauthclient/kibana-proxy
```

問題が解決しない場合は、OAuth クライアントに一覧表示されている URL の Kibana にアクセスしていることを確認してください。この問題は、転送先ポート (標準の 443 HTTPS ポートではなく 1443 など) の URL にアクセスすることで発生する可能性があります。OAuth クライアントを編集することで、サーバーのホワイトリストを調整できます。

```
$ oc edit oauthclient/kibana-proxy
```

12.3. KIBANA コンソール表示時の KUBERNETES の 503 エラーのトラブルシューティング

Kibana コンソールを表示する時にプロキシーエラーが発生する場合は、2つの問題のうちのいずれかが原因である可能性があります。

- Kibana が Pod を認識していない可能性があります。Elasticsearch の起動が遅い場合、Kibana はそれに到達しようとしてタイムアウトする場合があります。関連サービスにエンドポイントがあるかどうか確認してください。

```
$ oc describe service kibana
Name:          kibana
[...]
Endpoints:    <none>
```

Kibana Pod が実行中である場合、エンドポイントが一覧表示されます。実行中でない場合は、Kibana Pod とデプロイメントの状態を確認してください。デプロイメントをスケールダウンして、再び元に戻さなければならない場合があります。

- Kibana サービスにアクセスするためのルートがマスクされています。これは、あるプロジェクトでテストデプロイメントを実行し、次に最初のデプロイメントを完全に削除することなく別のプロジェクトでデプロイした場合に発生する可能性があります。複数のルートが同じ宛先に送信される場合、デフォルトルーターは最初に作成されたルートにのみルーティングします。問題が発生するルートをチェックして、そのルートが複数の場所で定義されているかどうかを確認してください。

```
$ oc get route --all-namespaces --selector logging-infra=support
```

第13章 エクスポートされるフィールド

ログインシステムによってエクスポートされ、Elasticsearch および Kibana での検索に利用できるフィールドがあります。検索時には、ドットの付いたフィールドのフルネームを使用します。たとえば、Elasticsearch /_search URL の場合、Kubernetes Pod 名を検索するには、/_search/q=kubernetes.pod_name:name-of-my-pod を使用します。

以下のセクションでは、ログインストアに存在しない可能性のあるフィールドについて説明します。これらのフィールドのすべてがすべてのレコードにある訳ではありません。フィールドは以下のカテゴリーに分類されます。

- **exported-fields-Default**
- **exported-fields-systemd**
- **exported-fields-kubernetes**
- **exported-fields-pipeline_metadata**
- **exported-fields-ovirt**
- **exported-fields-aushape**
- **exported-fields-tlog**

13.1. デフォルトのエクスポートされるフィールド

これらは、ログインシステムによってエクスポートされるデフォルトフィールドであり、Elasticsearch および Kibana での検索に利用できます。デフォルトフィールドは最上位および **collectd*** フィールドです。

最上位フィールド

最上位フィールドは、すべてのアプリケーションに共通であり、すべてのレコードに存在する可能性があります。Elasticsearch テンプレートの場合、最上位フィールドは、テンプレートのマッピングセクションで **default** の実際のマッピングを設定します。

パラメーター	説明
@timestamp	ログペイロードが作成された時点か、作成時間が不明な場合はログペイロードが最初に収集された時点の UTC 値のマーキング。これは、ログを処理するパイプラインのログペイロードの生成時についてのベストエフォートベースの判別に基づきます。フィールドを特定の目的のために予約されることを示す @ プレフィックスを追加します。Elasticsearch の場合、ほとんどのツールはデフォルトで @timestamp を検索します。形式は 2015-01-24 14:06:05.071000 などのようになります。
geoip	これはマシンの geo IP です。
hostname	hostname は、元のペイロードを生成するエンティティの完全修飾ドメイン名 (FQDN) です。このフィールドでは、このコンテキストの取得が試行されます。これを生成するエンティティがコンテキストを認識している場合があります。また、エンティティには、コレクターまたはノーマライザーによって認識される制限された namespace がある場合もあります。

パラメーター	説明
ipaddr4	ソースサーバーの IP アドレス V4。配列である場合があります。
ipaddr6	ソースサーバーの IP アドレス V6(ある場合)。
level	<p>rsyslog (severitytext プロパティ)、python のロギングモジュールによって提供されるロギングレベル。使用できる値は misc/sys/syslog.h に一覧表示される値と、trace および unknown です。たとえば "alert crit debug emerg err info notice trace unknown warning" などがあります。trace は syslog.h 一覧にはありませんが、数多くのアプリケーションがこれを使用することに注意してください。</p> <p>をクリックします。unknown は、ロギングシステムが認識しない値を取得した時のみ使用します。この値は最も高い値であることに注意してください。trace は debug よりも高レベルであるか、詳細度が高いと見なされます。error は非推奨となっているため、err を使用してください。panic を emerg に変換します。warn を warning に変換します。</p> <p>syslog/journal PRIORITY からの数値は通常、misc/sys/syslog.h に記載されている優先順位の値を使用してマップされます。</p> <p>他のロギングシステムからのログレベルおよび優先順位は、最も近い一致にマップされる必要があります。例については、python logging を参照してください。</p>
message	通常のログエントリーメッセージまたはペイロードです。これはコレクターまたはノーマライザーによってプルされるメタデータから削除される可能性があります。これは UTF-8 でエンコーディングされます。
pid	ロギングエンティティのプロセス ID です (ある場合)。
service	ロギングエンティティに関連付けられたサービスの名前です (ある場合)。たとえば、 syslog APP-NAME プロパティは、サービスフィールドにマップされます。
tags	コレクターまたはノーマライザーによって各ログに配置される、オプションで指定される Operator 定義のタグの一覧です。ペイロードには、ホワイトスペースで区切られた文字列トークンまたは文字列トークンの JSON 一覧を使用した文字列を指定できます。
file	ファイルパスのコレクター TODO アナライザーに対してローカルのログエントリーを含むファイルへのオプションのパスです。

パラメーター	説明
offset	オフセット値では、値が単一ログファイルで単調に増加する場合に、バイトの値をファイルのログ行 (ゼロまたは1ベース) またはログ行の番号 (ゼロまたは1ベース) の開始地点に表示できます。この値はラップでき、ログファイルの新規バージョンを表示できます (ローテーション)。
namespace_name	このレコードを、その名前を共有する namespace に関連付けます。この値は保存されませんが、アクセス制御および可視化のためにレコードを適切な namespace に関連付けるために使用されます。通常、この値はタグに指定されますが、プロトコルがタグの送信をサポートしない場合、このフィールドを使用できます。このフィールドがある場合、これはタグまたは kubernetes.namespace_name に指定される namespace を上書きします。
namespace_uuid	これは、 namespace_name に関連付けられる uuid です。この値は保存されませんが、アクセス制御および可視化のためにレコードを適切な namespace に関連付けるために使用されます。このフィールドがある場合、これは kubernetes.namespace_uuid に指定される uuid を上書きします。また、これにより、このログレコードの Kubernetes メタデータ検索はスキップされます。

collectd フィールド

以下のフィールドは namespace メトリクスのメタデータを表します。

パラメーター	説明
collectd.interval	type: float collectd の間隔。
collectd.plugin	type: string collectd プラグイン。
collectd.plugin_instance	type: string collectd plugin_instance
collectd.type_instance	type: string collectd type_instance 。
collectd.type	type: string collectd タイプ。

パラメーター	説明
collectd.dtypes	type: string collectd データセットタイプ。

collectd.processes フィールド

以下のフィールドは **collectd** プロセスのプラグインに対応します。

パラメーター	説明
collectd.processes.ps_state	type: integer collectd ps_state タイプのプロセスプラグイン。

collectd.processes.ps_disk_ops フィールド

collectd ps_disk_ops タイプのプロセスプラグイン。

パラメーター	説明
collectd.processes.ps_disk_ops.read	type: float TODO
collectd.processes.ps_disk_ops.write	type: float TODO
collectd.processes.ps_vm	type: integer collectd ps_vm タイプのプロセスプラグイン。
collectd.processes.ps_rss	type: integer collectd ps_rss タイプのプロセスプラグイン。
collectd.processes.ps_data	type: integer collectd ps_data タイプのプロセスプラグイン。
collectd.processes.ps_code	type: integer collectd ps_code タイプのプロセスプラグイン。
collectd.processes.ps_stacksize	type: integer collectd ps_stacksize タイプのプロセスプラグイン。

collectd.processes.ps_cputime フィールド

collectd ps_cputime タイプのプロセスプラグイン。

パラメーター	説明
collectd.processes.ps_cp utime.user	type: float TODO
collectd.processes.ps_cp utime.syst	type: float TODO

collectd.processes.ps_count フィールド
collectd ps_count タイプのプロセスプラグイン。

パラメーター	説明
collectd.processes.ps_co unt.processes	type: integer TODO
collectd.processes.ps_co unt.threads	type: integer TODO

collectd.processes.ps_pagefaults フィールド
collectd ps_pagefaults タイプのプロセスプラグイン。

パラメーター	説明
collectd.processes.ps_pa gefaults.majflt	type: float TODO
collectd.processes.ps_pa gefaults.minflt	type: float TODO

collectd.processes.ps_disk_octets フィールド
collectd ps_disk_octets タイプのプロセスプラグイン。

パラメーター	説明
collectd.processes.ps_di sk_octets.read	type: float TODO
collectd.processes.ps_di sk_octets.write	type: float TODO

パラメーター	説明
collectd.processes.fork_rate	type: float collectd fork_rate タイプのプロセスプラグイン。

collectd.disk フィールド

collectd ディスクプラグインに対応します。

collectd.disk.disk_merged フィールド

collectd disk_merged タイプのディスクプラグイン。

パラメーター	説明
collectd.disk.disk_merge_d.read	type: float TODO
collectd.disk.disk_merge_d.write	type: float TODO

collectd.disk.disk_octets フィールド

collectd disk_octets タイプのディスクプラグイン。

パラメーター	説明
collectd.disk.disk_octets.read	type: float TODO
collectd.disk.disk_octets.write	type: float TODO

collectd.disk.disk_time フィールド

collectd disk_time タイプのディスクプラグイン。

パラメーター	説明
collectd.disk.disk_time.read	type: float TODO
collectd.disk.disk_time.write	type: float TODO

collectd.disk.disk_ops フィールド

collectd disk_ops タイプのディスクプラグインです。

パラメーター	説明
collectd.disk.disk_ops.read	type: float TODO
collectd.disk.disk_ops.write	type: float TODO
collectd.disk.pending_operations	type: integer collectd pending_operations タイプのディスクプラグイン。

collectd.disk.disk_io_time フィールド

collectd disk_io_time タイプのディスクプラグイン。

パラメーター	説明
collectd.disk.disk_io_time.io_time	type: float TODO
collectd.disk.disk_io_time.weighted_io_time	type: float TODO

collectd.interface フィールド

collectd インターフェースプラグインに対応します。

collectd.interface.if_octets フィールド

collectd if_octets タイプのインターフェースプラグイン。

パラメーター	説明
collectd.interface.if_octets.rx	type: float TODO
collectd.interface.if_octets.tx	type: float TODO

collectd.interface.if_packets フィールド

collectd if_packets タイプのインターフェースプラグイン。

パラメーター	説明
collectd.interface.if_packets.rx	type: float TODO
collectd.interface.if_packets.tx	type: float TODO

collectd.interface.if_errors フィールド
collectd if_errors タイプのインターフェースプラグイン。

パラメーター	説明
collectd.interface.if_errors.rx	type: float TODO
collectd.interface.if_errors.tx	type: float TODO

collectd.interface.if_dropped フィールド
collectd if_dropped タイプのインターフェースプラグイン。

パラメーター	説明
collectd.interface.if_dropped.rx	type: float TODO
collectd.interface.if_dropped.tx	type: float TODO

collectd.virt フィールド
collectd 仮想プラグインに対応します。

collectd.virt.if_octets フィールド
collectd if_octets タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.if_octets.rx	type: float TODO

パラメーター	説明
collectd.virt.if_octets.tx	type: float TODO

collectd.virt.if_packets フィールド
collectd if_packets タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.if_packets.rx	type: float TODO
collectd.virt.if_packets.tx	type: float TODO

collectd.virt.if_errors フィールド
collectd if_errors タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.if_errors.rx	type: float TODO
collectd.virt.if_errors.tx	type: float TODO

collectd.virt.if_dropped フィールド
collectd if_dropped タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.if_dropped.rx	type: float TODO
collectd.virt.if_dropped.tx	type: float TODO

collectd.virt.disk_ops フィールド
collectd disk_ops タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.disk_ops.read	type: float TODO
collectd.virt.disk_ops.write	type: float TODO

collectd.virt.disk_octets フィールド
collectd disk_octets タイプの仮想プラグイン。

パラメーター	説明
collectd.virt.disk_octets.read	type: float TODO
collectd.virt.disk_octets.write	type: float TODO
collectd.virt.memory	type: float collectd メモリータイプの仮想プラグイン。
collectd.virt.virt_vcpu	type: float collectd virt_vcpu タイプの仮想プラグイン。
collectd.virt.virt_cpu_total	type: float collectd virt_cpu_total タイプの仮想プラグイン。

collectd.CPU フィールド
collectd CPU プラグインに対応します。

パラメーター	説明
collectd.CPU.percent	type: float collectd percent タイプの CPU プラグイン。

collectd.df フィールド
collectd df プラグインに対応します。

パラメーター	説明
collectd.df.df_complex	type: float collectd df_complex タイプの df プラグイン。
collectd.df.percent_bytes	type: float collectd percent_bytes タイプの df プラグイン。

collectd.entropy フィールド

collectd エントロピープラグインに対応します。

パラメーター	説明
collectd.entropy.entropy	type: integer collectd エントロピータイプのエントロピー プラグイン。

collectd.memory フィールド

collectd メモリープラグインに対応します。

パラメーター	説明
collectd.memory.memory	type: float collectd メモリータイプのメモリープラグイン。
collectd.memory.percent	type: float collectd パーセントタイプのメモリープラグイン。

collectd.swap フィールド

collectd swap プラグインに対応します。

パラメーター	説明
collectd.swap.swap	type: integer collectd swap タイプの swap プラグイン。
collectd.swap.swap_io	type: integer collectd swap_io タイプの swap プラグイン。

collectd.load フィールド

collectd ロードプラグインに対応します。

collectd.load.load フィールド**collectd** ロードタイプのロードプラグイン

パラメーター	説明
collectd.load.load.shortterm	type: float TODO
collectd.load.load.midterm	type: float TODO
collectd.load.load.longterm	type: float TODO

collectd.aggregation フィールド**collectd** 集計プラグインに対応します。

パラメーター	説明
collectd.aggregation.percent	type: float TODO

collectd.statsd フィールド**collectd statsd** プラグインに対応します。

パラメーター	説明
collectd.statsd.host_cpu	type: integer collectd CPU タイプの statsd プラグイン。
collectd.statsd.host_elapsed_time	type: integer collectd elapsed_time タイプの statsd プラグイン。
collectd.statsd.host_memory	type: integer collectd メモリータイプの statsd プラグイン。
collectd.statsd.host_nic_speed	type: integer collectd nic_speed タイプの statsd プラグイン。
collectd.statsd.host_nic_rx	type: integer collectd nic_rx タイプの statsd プラグイン。

パラメーター	説明
<code>collectd.statsd.host_nic_tx</code>	type: integer collectd nic_tx タイプの statsd プラグイン。
<code>collectd.statsd.host_nic_rx_dropped</code>	type: integer collectd nic_rx_dropped タイプの statsd プラグイン。
<code>collectd.statsd.host_nic_tx_dropped</code>	type: integer collectd nic_tx_dropped タイプの statsd プラグイン。
<code>collectd.statsd.host_nic_rx_errors</code>	type: integer collectd nic_rx_errors タイプの statsd プラグイン。
<code>collectd.statsd.host_nic_tx_errors</code>	type: integer collectd nic_tx_errors タイプの statsd プラグイン。
<code>collectd.statsd.host_storage</code>	type: integer collectd ストレージタイプの statsd プラグイン。
<code>collectd.statsd.host_swap</code>	type: integer collectd swap タイプの statsd プラグイン。
<code>collectd.statsd.host_vdsm</code>	type: integer collectd VDSM タイプの statsd プラグイン。
<code>collectd.statsd.host_vms</code>	type: integer collectd VMS タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_tx_dropped</code>	type: integer collectd nic_tx_dropped タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_rx_bytes</code>	type: integer collectd nic_rx_bytes タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_tx_bytes</code>	type: integer collectd nic_tx_bytes タイプの statsd プラグイン。

パラメーター	説明
collectd.statsd.vm_balloon_min	type: integer collectd balloon_min タイプの statsd プラグイン。
collectd.statsd.vm_balloon_max	type: integer collectd balloon_max タイプの statsd プラグイン。
collectd.statsd.vm_balloon_target	type: integer collectd balloon_target タイプの statsd プラグイン。
collectd.statsd.vm_balloon_cur	type: integer collectd balloon_cur タイプの statsd プラグイン。
collectd.statsd.vm_cpu_sys	type: integer collectd cpu_sys タイプの statsd プラグイン。
collectd.statsd.vm_cpu_usage	type: integer collectd cpu_usage タイプの statsd プラグイン。
collectd.statsd.vm_disk_read_ops	type: integer collectd disk_read_ops タイプの statsd プラグイン。
collectd.statsd.vm_disk_write_ops	type: integer collectd disk_write_ops タイプの statsd プラグイン。
collectd.statsd.vm_disk_flush_latency	type: integer collectd disk_flush_latency タイプの statsd プラグイン。
collectd.statsd.vm_disk_apparent_size	type: integer collectd disk_apparent_size タイプの statsd プラグイン。
collectd.statsd.vm_disk_write_bytes	type: integer collectd disk_write_bytes タイプの statsd プラグイン。
collectd.statsd.vm_disk_write_rate	type: integer collectd disk_write_rate タイプの statsd プラグイン。

パラメーター	説明
<code>collectd.statsd.vm_disk_true_size</code>	type: integer collectd disk_true_size タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_read_rate</code>	type: integer collectd disk_read_rate タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_write_latency</code>	type: integer collectd disk_write_latency タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_read_latency</code>	type: integer collectd disk_read_latency タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_read_bytes</code>	type: integer collectd disk_read_bytes タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_rx_dropped</code>	type: integer collectd nic_rx_dropped タイプの statsd プラグイン。
<code>collectd.statsd.vm_cpu_user</code>	type: integer collectd cpu_user タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_rx_errors</code>	type: integer collectd nic_rx_errors タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_tx_errors</code>	type: integer collectd nic_tx_errors タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_speed</code>	type: integer collectd nic_speed タイプの statsd プラグイン。

collectd.postgresql フィールド
collectd postgresql プラグインに対応します。

パラメーター	説明
<code>collectd.postgresql.pg_n_tup_g</code>	type: integer collectd pg_n_tup_g タイプの postgresql プラグイン。

パラメーター	説明
<code>collectd.postgresql.pg_n_tup_c</code>	type: integer collectd pg_n_tup_c タイプの postgresql プラグイン。
<code>collectd.postgresql.pg_n_umbakends</code>	type: integer collectd pg_numbackends タイプの postgresql プラグイン。
<code>collectd.postgresql.pg_xact</code>	type: integer collectd pg_xact タイプの postgresql プラグイン。
<code>collectd.postgresql.pg_db_size</code>	type: integer collectd pg_db_size タイプの postgresql プラグイン。
<code>collectd.postgresql.pg_blks</code>	type: integer collectd pg_blks タイプの postgresql プラグイン。

13.2. SYSTEMD のエクスポートされるフィールド

これらのフィールドは OpenShift Container Platform クラスターロギングによってエクスポートされる **systemd** フィールドであり、Elasticsearch および Kibana での検索に利用できます。

systemd ジャーナルに固有の共通フィールドが含まれます。アプリケーションは、独自のフィールドをジャーナルに書き込む可能性があります。それらは **systemd.u** namespace で利用できます。RESULT および UNIT はこれらの2つのフィールドです。

systemd.k フィールド

以下の表には、**systemd** カーネル固有のメタデータが含まれます。

パラメーター	説明
<code>systemd.k.KERNEL_DEVICE</code>	systemd.k.KERNEL_DEVICE は、カーネルのデバイス名です。
<code>systemd.k.KERNEL_SUBSYSTEM</code>	systemd.k.KERNEL_SUBSYSTEM は、カーネルのサブシステム名です。
<code>systemd.k.UDEV_DEVLINK</code>	systemd.k.UDEV_DEVLINK には、ノードを参照する追加のシンボリックリンク名が含まれます。
<code>systemd.k.UDEV_DEVNODE</code>	systemd.k.UDEV_DEVNODE は、デバイスのノードパスです。

パラメーター	説明
systemd.k.UDEV_SYSNAME	systemd.k.UDEV_SYSNAME は、カーネルのデバイス名です。

systemd.t フィールド

systemd.t Fields は信頼されたジャーナルフィールドであり、ジャーナルによって暗黙的に追加されるフィールドであり、クライアントノードで変更することはできません。

パラメーター	説明
systemd.t.AUDIT_LOGIN_UID	systemd.t.AUDIT_LOGINUID は、ジャーナルエントリプロセスのユーザー ID です。
systemd.t.BOOT_ID	systemd.t.BOOT_ID は、カーネルのブート ID です。
systemd.t.AUDIT_SESSION	systemd.t.AUDIT_SESSION は、ジャーナルエントリプロセスのセッションです。
systemd.t.CAP_EFFECTIVE	systemd.t.CAP_EFFECTIVE は、ジャーナルエントリプロセスの機能を表します。
systemd.t.CMDLINE	systemd.t.CMDLINE は、ジャーナルエントリプロセスのコマンドラインです。
systemd.t.COMM	systemd.t.COMM は、ジャーナルエントリプロセスの名前です。
systemd.t.EXE	systemd.t.EXE は、ジャーナルエントリプロセスの実行可能パスです。
systemd.t.GID	systemd.t.GID は、ジャーナルエントリプロセスのグループ ID です。
systemd.t.HOSTNAME	systemd.t.HOSTNAME は、ホストの名前です。
systemd.t.MACHINE_ID	systemd.t.MACHINE_ID は、ホストのマシン ID です。
systemd.t.PID	systemd.t.PID は、ジャーナルエントリプロセスのプロセス ID です。
systemd.t.SELINUX_CONTEXT	systemd.t.SELINUX_CONTEXT は、ジャーナルエントリプロセスのセキュリティコンテキストまたはラベルです。
systemd.t.SOURCE_REALTIME_TIMESTAMP	systemd.t.SOURCE_REALTIME_TIMESTAMP は、最も早くかつ最も信頼できるメッセージのタイムスタンプです。これは RFC 3339 NS 形式に変換されます。
systemd.t.SYSTEMD_CGROUP	systemd.t.SYSTEMD_CGROUP は、 systemd コントロールグループパスです。

パラメーター	説明
<code>systemd.t.SYSTEMD_OWNER_UID</code>	<code>systemd.t.SYSTEMD_OWNER_UID</code> は、セッションの所有者 ID です。
<code>systemd.t.SYSTEMD_SESSION</code>	<code>systemd.t.SYSTEMD_SESSION</code> は、(該当する場合) <code>systemd</code> セッション ID です。
<code>systemd.t.SYSTEMD_SLICE</code>	<code>systemd.t.SYSTEMD_SLICE</code> は、ジャーナルエントリプロセスのスライス (slice) ユニットです。
<code>systemd.t.SYSTEMD_UNIT</code>	<code>systemd.t.SYSTEMD_UNIT</code> は、セッションのユニット名です。
<code>systemd.t.SYSTEMD_USER_UNIT</code>	<code>systemd.t.SYSTEMD_USER_UNIT</code> は、(該当する場合) セッションのユーザーユニット名です。
<code>systemd.t.TRANSPORT</code>	<code>systemd.t.TRANSPORT</code> は、ジャーナルサービス別のエントリーのメソッドです。これには、 <code>audit</code> 、 <code>driver</code> 、 <code>syslog</code> 、 <code>journal</code> 、 <code>stdout</code> 、および <code>kernel</code> が含まれます。
<code>systemd.t.UID</code>	<code>systemd.t.UID</code> は、ジャーナルエントリプロセスのユーザー ID です。
<code>systemd.t.SYSLOG_FACILITY</code>	<code>systemd.t.SYSLOG_FACILITY</code> は、 <code>syslog</code> の 10 進数の文字列としてフォーマットされる機能を含むフィールドです。
<code>systemd.t.SYSLOG_IDENTIFIER</code>	<code>systemd.t.systemd.t.SYSLOG_IDENTIFIER</code> は、 <code>syslog</code> の識別子です。
<code>systemd.t.SYSLOG_PID</code>	<code>SYSLOG_PID</code> は、 <code>syslog</code> のクライアントプロセス ID です。

systemd.u フィールド

`systemd.u Fields` はクライアントから直接渡され、ジャーナルに保存されます。

パラメーター	説明
<code>systemd.u.CODE_FILE</code>	<code>systemd.u.CODE_FILE</code> は、ソースのファイル名が含まれるコードの場所です。
<code>systemd.u.CODE_FUNCTION</code>	<code>systemd.u.CODE_FUNCTION</code> は、ソースの関数が含まれるコードの場所です。
<code>systemd.u.CODE_LINE</code>	<code>systemd.u.CODE_LINE</code> は、ソースの行数が含まれるコードの場所です。

パラメーター	説明
systemd.u.ERRNO	systemd.u.ERRNO は (ある場合)、10 進数の文字列として数値でフォーマットされる低位のエラー番号です。
systemd.u.MESSAGE_ID	systemd.u.MESSAGE_ID は、メッセージタイプを認識するためのメッセージ識別子の ID です。
systemd.u.RESULT	非公式の使用のみの場合に限定されます。
systemd.u.UNIT	非公式の使用のみの場合に限定されます。

13.3. KUBERNETES のエクスポートされるフィールド

これらは OpenShift Container Platform クラスターロギングでエクスポートされる Kubernetes フィールドであり、Elasticsearch および Kibana での検索に利用できます。

Kubernetes 固有メタデータの namespace です。 **kubernetes.pod_name** は Pod の名前です。

kubernetes.labels フィールド

OpenShift オブジェクトに割り当てられるラベルは **kubernetes.labels** です。各ラベル名はラベルフィールドのサブフィールドです。それぞれのラベル名ではドットが取られます。つまり、名前のドットはアンダースコアに置き換えられます。

パラメーター	説明
kubernetes.pod_id	Pod の Kubernetes ID。
kubernetes.namespace_name	Kubernetes の namespace の名前。
kubernetes.namespace_id	Kubernetes の namespace の ID。
kubernetes.host	Kubernetes ノード名。
kubernetes.container_name	Kubernetes のコンテナの名前。
kubernetes.labels.deployment	Kubernetes オブジェクトに関連付けられるデプロイメント。
kubernetes.labels.deploymentconfig	Kubernetes オブジェクトに関連付けられる deploymentconfig。
kubernetes.labels.component	Kubernetes オブジェクトに関連付けられるコンポーネント。

パラメーター	説明
kubernetes.labels.provider	Kubernetes オブジェクトに関連付けられるプロバイダー。

kubernetes.annotations フィールド

OpenShift オブジェクトに関連付けられるアノテーションは **kubernetes.annotations** フィールドです。

13.4. コンテナのエクスポートされるフィールド

これらは OpenShift Container Platform クラスターロギングによってエクスポートされる Docker フィールドであり、Elasticsearch および Kibana での検索に利用できます。namespace は docker コンテナ固有のメタデータの namespace です。docker.container_id は Docker コンテナ ID です。

pipeline_metadata.collector フィールド

このセクションには、コレクターに固有のメタデータが含まれます。

パラメーター	説明
pipeline_metadata.collector.hostname	コレクターの FQDN。これはログの実際のエミッターの FQDN とは異なる場合があります。
pipeline_metadata.collector.name	コレクターの名前。
pipeline_metadata.collector.version	コレクターのバージョン。
pipeline_metadata.collector.ipaddr4	コレクターサーバの IP アドレス v4。配列である場合があります。
pipeline_metadata.collector.ipaddr6	コレクターサーバの IP アドレス v6。配列である場合があります。
pipeline_metadata.collector.inputname	ログメッセージがコレクターによって受信された方法。TCP/UDP または imjournal/imfile。
pipeline_metadata.collector.received_at	メッセージがコレクターによって受信された時間。
pipeline_metadata.collector.original_raw_message	コレクターで収集された、解析されていない元のログメッセージ、または限りなくソースに近いログメッセージ。

pipeline_metadata.normalizer フィールド

このセクションには、ノーマライザーに固有のメタデータが含まれます。

パラメーター	説明
<code>pipeline_metadata.normalizer.hostname</code>	ノーマライザーの FQDN。
<code>pipeline_metadata.normalizer.name</code>	ノーマライザーの名前。
<code>pipeline_metadata.normalizer.version</code>	ノーマライザーのバージョン。
<code>pipeline_metadata.normalizer.ipaddr4</code>	ノーマライザーサーバーの IP アドレス v4。配列である場合があります。
<code>pipeline_metadata.normalizer.ipaddr6</code>	ノーマライザーサーバーの IP アドレス v6。配列である場合があります。
<code>pipeline_metadata.normalizer.inputname</code>	ログメッセージがノーマライザーによって受信された方法。TCP/UDP かどうか。
<code>pipeline_metadata.normalizer.received_at</code>	メッセージがノーマライザーによって受信された時間。
<code>pipeline_metadata.normalizer.original_raw_message</code>	ノーマライザーで受信される、解析されていない元のログメッセージ。
<code>pipeline_metadata.trace</code>	このフィールドは、メッセージの追跡を記録します。各コレクターおよびノーマライザーは自らについての情報およびメッセージが処理された日時についての情報を追加します。

13.5. OVIRT のエクスポートされるフィールド

これらは OpenShift Container Platform クラスターロギングでエクスポートされる oVirt フィールドであり、Elasticsearch および Kibana での検索に利用できます。

oVirt メタデータの namespace。

パラメーター	説明
<code>ovirt.entity</code>	データソース、ホスト、VMS、およびエンジンのタイプ。
<code>ovirt.host_id</code>	oVirt ホストの UUID。

ovirt.engine フィールド

oVirt エンジン関連のメタデータの namespace。oVirt エンジンの FQDN は `ovirt.engine.fqdn` です。

13.6. AUSHAPE のエクスポートされるフィールド

これらは OpenShift Container Platform クラスターロギングでエクスポートされる Aushape フィールドであり、Elasticsearch および Kibana での検索に利用できます。

Aushape で変換される監査イベント。詳細は [Aushape](#) を参照してください。

パラメーター	説明
aushape.serial	監査イベントのシリアル番号。
aushape.node	監査イベントが発生したホストの名前。
aushape.error	イベントの変換中に aushape に発生したエラー。
aushape.trimmed	イベントオブジェクトに関連する JSONPath 表現の配列であり、イベントサイズの制限の結果として削除されたコンテンツを持つオブジェクトまたは配列を指定します。空の文字列は、イベントがコンテンツを削除したことを意味し、空の配列は、指定されていないオブジェクトおよび配列によってトリミングが生じたことを意味します。
aushape.text	元の監査イベントを表す配列ログレコード文字列。

aushape.data フィールド

Aushape に関連する解析された監査イベントデータ。

パラメーター	説明
aushape.data.avc	type: nested
aushape.data.execve	type: string
aushape.data.netfilter_cfg	type: nested
aushape.data.obj_pid	type: nested
aushape.data.path	type: nested

13.7. TLOG のエクスポートされるフィールド

これらは OpenShift Container Platform クラスターロギングでエクスポートされる Tlog フィールドであり、Elasticsearch および Kibana での検索に利用できます。

Tlog ターミナル I/O の記録メッセージ。詳細は [Tlog](#) を参照してください。

パラメーター	説明
tlog.ver	メッセージ形式のバージョン番号。

パラメーター	説明
tlog.user	記録されたユーザー名。
tlog.term	ターミナルタイプ名。
tlog.session	記録されたセッションの監査セッション ID。
tlog.id	セッション内のメッセージの ID。
tlog.pos	セッション内のメッセージの位置 (ミリ秒単位)。
tlog.timing	このメッセージのイベントの配分 (時間)。
tlog.in_txt	無効な文字が除去された入力テキスト。
tlog.in_bin	除去された無効な入力文字 (バイト)。
tlog.out_txt	無効な文字が除去された出力テキスト。
tlog.out_bin	除去された無効な出力文字 (バイト)。

第14章 クラスターロギングのアンインストール

クラスターロギングをお使いの OpenShift Container Platform クラスターから削除することができます。

14.1. OPENSIFT CONTAINER PLATFORM からのクラスターロギングのアンインストール

クラスターロギングをクラスターから削除できます。

前提条件

- クラスターロギングおよび Elasticsearch がインストールされている必要があります。

手順

クラスターロギングを削除するには、以下を実行します。

1. 以下のコマンドを使用して、デプロイメント時に生成されたすべてのものを削除します。

```
$ oc delete clusterlogging instance -n openshift-logging
```

2. 以下のコマンドを使用して、Operator インスタンスの削除後も保持される Persistent Volume Claim（永続ボリューム要求、PVC）を削除します。

```
$ oc delete pvc --all -n openshift-logging
```