



OpenShift Container Platform 4.1

ロギング

OpenShift Container Platform 4.1でのクラスターロギングの設定

OpenShift Container Platform 4.1 ロギング

OpenShift Container Platform 4.1 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について	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	11
第3章 クラスターロギングのデプロイ	13
3.1. クラスターロギングおよび ELASTICSEARCH OPERATOR のインストール	13
3.2. 追加リソース	20
第4章 KIBANA インターフェースの表示	21
4.1. KIBANA インターフェースの起動	21
第5章 イベントルーターの使用	22
5.1. イベントルーターのデプロイおよび設定	22
第6章 クラスターロギングデプロイメントの設定	26
6.1. クラスターロギングの設定について	26
6.2. クラスターロギングの管理状態の変更	33
6.3. クラスターロギングの設定	35
6.4. ログデータを保存し、整理するための ELASTICSEARCH の設定	37
6.5. KIBANA の設定	44
6.6. ELASTICSEARCH データのキューレーション	46
6.7. FLUENTD の設定	53
6.8. OPENSIFT CONTAINER PLATFORM ログの外部デバイスへの送信	63
6.9. SYSTEMD-JOURNALD および RSYSLOG の設定	69
第7章 ELASTICSEARCH ステータスの表示	73
7.1. ELASTICSEARCH ステータスの表示	73
7.2. ELASTICSEARCH コンポーネントのステータスの表示	75
第8章 ノードセクターを使用したクラスターロギングリソースの移動	79
8.1. ノードセクターを使用したクラスターロギングコンポーネントのノードの指定	79
第9章 追加リソース	81
第10章 ELASTICSEARCH の手動によるロールアウト	82
10.1. ELASTICSEARCH クラスターのローリング再起動の実行	82
第11章 KIBANA のトラブルシューティング	86
11.1. KUBERNETES ログインループのトラブルシューティング	86
11.2. 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	86
11.3. 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	87
第12章 エクスポートされるフィールド	88
12.1. デフォルトのエクスポートされるフィールド	88
12.2. RSYSLOG のエクスポートされるフィールド	103
12.3. SYSTEMD のエクスポートされるフィールド	104
12.4. KUBERNETES のエクスポートされるフィールド	106
12.5. コンテナのエクスポートされるフィールド	107
12.6. OVIRT のエクスポートされるフィールド	109
12.7. AUSHAPE のエクスポートされるフィールド	109

12.8. TLOG のエクスポートされるフィールド	110
第13章 クラスターログインのアンインストール	112
13.1. OPENSIFT CONTAINER PLATFORM からのクラスターログインのアンインストール	112

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

OpenShift Container Platform クラスターロギングは、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について

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 (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のコンポーネントについて

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、4 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。

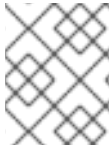
1.1.2. Elasticsearch について

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 (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。

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。

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

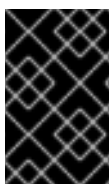
1.1.3. Fluentd について

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。

Fluentd 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。

Fluentd は **journald** をシステムログソースとして使用します。これらは、オペレーティングシステム、コンテナーランタイム、および OpenShift Container Platform からのログメッセージです。

コンテナーランタイムは、プロジェクト、Pod 名、およびコンテナー 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ません。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ーランタイムは、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. Kibana 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. Curator 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 はその設定に基づき、日次ベースで動作を実行します。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 * * *" ❶
```

❶ 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. イベントルーターについて

イベントルーターは、OpenShift Container Platform イベントをクラスターロギングに転送する 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イベントを収集し、それらを JSON 形式に変換します。この形式では、イベントの取得後、それらを **STDOUT** にプッシュします。Fluentd はイベントを **.operations** インデックスにインデックス化します。

1.1.7. クラスターロギングのカスタムリソース定義 (CRD、Custom Resource Definition) について

Cluster Logging Operator のカスタムリソース定義 (CRD) は、ログを収集し、保存し、視覚化するためのロギングスタックのすべてのコンポーネントが含まれる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。

この CRD を変更する必要は一切ありません。デプロイメントを変更するには、特定のカスタムリソース (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: 2
  resources:
    limits:
      memory: 2Gi
    requests:
```

```
    cpu: 200m
    memory: 2Gi
  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:
        requests:
          cpu: 1
          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
  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:
    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:
    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 format] で指定できます (<https://en.wikipedia.org/wiki/Cron>)。

```
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: 2
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 200m
          memory: 2Gi
      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。

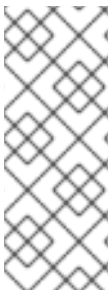
第3章 クラスターロギングのデプロイ

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。
- Web コンソールを使用してクラスターロギングのサブスクリプションをインストールします。

3.1. クラスターロギングおよび ELASTICSEARCH OPERATOR のインストール

OpenShift Container Platform コンソールを使用し、クラスターロギングおよび Elasticsearch Operator をデプロイしてクラスターロギング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。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。



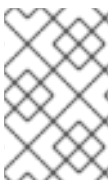
注記

OpenShift Container Platform クラスターロギングソリューションでは、Cluster Logging Operator と Elasticsearch Operator の両方をインストールする必要があります。OpenShift Container Platform では、これらの Operator を個別にインストールするユースケースはありません。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**。Cluster Logging Operator は、Web コンソールまたは CLI のどちらを使用してもインストールできます。

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。それぞれの Elasticsearch ノードには、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

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**。Cluster Logging Operator は、Web コンソールまたは CLI のどちらを使用してもインストールできます。

手順

1. Elasticsearch Operator および Cluster Logging Operator の namespace を作成します。



注記

また、Web コンソールの **Administration** → **Namespaces** ページを使用して namespace を作成することもできます。サンプル YAML に一覧表示されている **cluster-logging** および **cluster-monitoring** ラベルを作成する namespace に適用する必要があります。

- a. Elasticsearch Operator の namespace (**eo-namespace.yaml** など) を作成します。

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

- ❶ **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。
- ❷ 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. 以下のコマンドを実行して namespace を作成します。

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

例:

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

- c. Cluster Logging Operator の namespace (**clo-namespace.yaml** など) を作成します。

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

- ❶ ❷ ❸ これらのラベルを以下のように指定します。

- d. 以下のコマンドを実行して namespace を作成します。

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

例:

```
$ oc create -f clo-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 ❶
spec: {}
```

❶ **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 eo-og.yaml
```

- c. クラスター上で Elasticsearch Operator を有効にするために、CatalogSourceConfig オブジェクト YAML ファイル (**eo-csc.yaml** など) を作成します。

CatalogSourceConfig のサンプル

```
apiVersion: "operators.coreos.com/v1"
kind: "CatalogSourceConfig"
metadata:
  name: "elasticsearch"
  namespace: "openshift-marketplace"
spec:
  targetNamespace: "openshift-operators-redhat" ❶
  packages: "elasticsearch-operator"
  source: "redhat-operators"
```

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

Operator は、**targetNamespace** に指定される namespace で CatalogSourceConfig から CatalogSource を生成します。

- d. CatalogSourceConfig オブジェクトを作成します。

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

- e. 以下のコマンドを使用して、次の手順に必要な **channel** の値を取得します。

```
$ oc get packagemanifest elasticsearch-operator -n openshift-marketplace -o
jsonpath='{.status.channels[].name}'
```

```
preview
```

- f. 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:
  generateName: "elasticsearch-"
  namespace: "openshift-operators-redhat" ❶
spec:
  channel: "preview" ❷
  installPlanApproval: "Automatic"
  source: "elasticsearch"
  sourceNamespace: "openshift-operators-redhat" ❸
  name: "elasticsearch-operator"
```

- ❶❸ **namespace** および **sourceNameSpace** に **openshift-operators-redhat** namespace を指定する必要があります。

- ❷ 直前の手順の **.status.channels[].name** 値を指定します。

- g. Subscription オブジェクトを作成します。

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

- h. **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"
```

- i. 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

```

j. RBAC オブジェクトを作成します。

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

Elasticsearch Operator はクラスター内の各プロジェクトにインストールされます。

3. 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と、最善の結果が得られます。
 - a. OpenShift Container Platform Web コンソールで、**Catalog → 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。
4. Operator のインストールを確認します。
 - a. **Catalog → 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。
 - c. **Elasticsearch Operator** が **Status** が **InstallSucceeded** の状態で **openshift-operator-redhat** プロジェクトに一覧表示されていることを確認します。Elasticsearch Operator は他のすべてのプロジェクトにコピーされます。



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。

いずれかの 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行うには、以下を実行します。

- **Catalog → Operator Management** ページに切り替え、**Operator Subscriptions** および **Install Plans** タブで **Status** の下の失敗またはエラー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。
5. クラスターロギングのインスタンスを作成します。
 - a. **Administration** → **Custom Resource Definitions** ページに切り替えます。
 - b. **Custom Resource Definitions** ページで、**ClusterLogging** をクリックします。
 - c. **Custom Resource Definition Overview** ページで、**Actions** メニューから **View Instances** を選択します。
 - d. **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。
 - e. YAML 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: gp2 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 CR の名前。これは **instance** でなければなりません。

- 2 クラスターロギングの管理状態。ほとんどの場合、デフォルト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。
- 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 クラスターの各 Elasticsearch 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。
- 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、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 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
```

- 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。
6. 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

3.2. 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 KIBANA インターフェースの表示

クラスターロギングインストールは Kibana インターフェースをデプロイします。

4.1. KIBANA インターフェースの起動

Kibana インターフェースは、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ーベースのコンソールです。

手順

Kibana インターフェースを起動するには、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。

第5章 イベントルーターの使用

イベントルーターは OpenShift Container Platform と通信し、イベントの発生する Pod のログに OpenShift Container Platform イベントを出力します。

クラスターロギングがデプロイされている場合、Kibana で OpenShift Container Platform イベントを表示できます。

5.1. 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。

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、ClusterRole、および ClusterRoleBinding を作成します。

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。
- Elasticsearch でイベントルーターのイベントを処理し、保存するために **TRANSFORM_EVENTS=true** を設定します。
 - クラスターロギングを管理外の状態に設定する。
 - **TRANSFORM_EVENTS** 機能を有効にします。

```
$ oc set env ds/fluentd TRANSFORM_EVENTS=true
```

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: 3
  - apiGroups: [""]
    resources: ["events"]
```

```
  verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding 4
  apiVersion: v1
  metadata:
    name: event-reader-binding
  subjects:
- kind: ServiceAccount
  name: eventrouter
  namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment
  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:
        limits:
          memory: ${MEMORY}
        requests:
          cpu: ${CPU}
```

```

        memory: ${MEMORY}
        volumeMounts:
        - name: config-volume
          mountPath: /etc/eventrouter
        volumes:
        - name: config-volume
          configMap:
            name: eventrouter
parameters:
- name: IMAGE 5
  displayName: Image
  value: "registry.redhat.io/openshift4/ose-logging-eventrouter:latest"
- name: MEMORY 6
  displayName: Memory
  value: "128Mi"
- name: CPU 7
  displayName: CPU
  value: "100m"
- name: NAMESPACE 8
  displayName: Namespace
  value: "openshift-logging"

```

- 1 イベントルーターのサービスアカウントを作成します。
- 2 クラスターロールを作成し、クラスター内のイベントを監視します。
- 3 **events** リソースの **get**、**watch**、および **list** パーミッションを許可します。
- 4 ClusterRoleBinding を作成し、ClusterRole を ServiceAccount にバインドします。
- 5 イベントルーターのイメージバージョンを指定します。
- 6 イベントルーター Pod のメモリー制限を指定します。デフォルトは '128Mi' になります。
- 7 イベントルーターに割り当てる CPU の最小量を指定します。デフォルトは '100m' になります。
- 8 イベントルーターがデプロイされる namespace を指定します。デフォルトは **openshift-logging** になります。この値は、ServiceAccount および ClusterRoleBinding に指定された値と同じでなければなりません。プロジェクトは、イベントを見つけることのできる Kibana の場所を示唆します。
 - イベントルーター Pod が、**kube-*** および **openshift-*** などのデフォルトプロジェクトでデプロイされる場合、イベントは **.operation** インデックスの下に見つけることができます。
 - イベントルーター Pod が他のプロジェクトにデプロイされている場合、イベントはプロジェクト namespace を使用してインデックスの下に見つけることができます。

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 -
```

例:

```
$ oc process -f eventrouter.yaml | oc apply -f -  
  
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インストールされていることを確認します。

```
$ oc get pods --selector component=eventrouter -o name  
  
pod/logging-eventrouter-d649f97c8-qvv8r
```

```
$ oc logs logging-eventrouter-d649f97c8-qvv8r
```

```
{"verb":"ADDED","event":{"metadata":{"name":"elasticsearch-  
operator.v0.0.1.158f402e25397146","namespace":"openshift-  
operators","selfLink":"/api/v1/namespaces/openshift-operators/events/elasticsearch-  
operator.v0.0.1.158f402e25397146","uid":"37b7ff11-4f1a-11e9-a7ad-  
0271b2ca69f0","resourceVersion":"523264","creationTimestamp":"2019-03-  
25T16:22:43Z"},"involvedObject":{"kind":"ClusterServiceVersion","namespace":"openshift-  
operators","name":"elasticsearch-operator.v0.0.1","uid":"27b2ca6d-4f1a-11e9-8fba-  
0ea949ad61f6","apiVersion":"operators.coreos.com/v1alpha1","resourceVersion":"523096"},"re-  
ason":"InstallSucceeded","message":"waiting for install components to report  
healthy","source":{"component":"operator-lifecycle-manager"},"firstTimestamp":"2019-03-  
25T16:22:43Z","lastTimestamp":"2019-03-25T16:22:43Z","count":1,"type":"Normal"}}
```

第6章 クラスターロギングデプロイメントの設定

6.1. 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クラスターにインストールした後に、以下の設定を行います。



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。

6.1.1. 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。

6.1.1.1. 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:
          requests:
            cpu: 1
            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

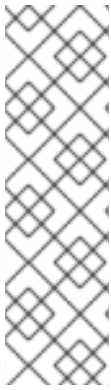
storageClass name および **size** パラメーターを使用し、Elasticsearch クラスターの永続ストレージのクラスおよびサイズを設定できます。Cluster Logging Operator は、これらのパラメーターに基づいて、Elasticsearch クラスターの各データノードについて **PersistentVolumeClaim** を作成します。

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    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:
    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 format] で指定できます (<https://en.wikipedia.org/wiki/Cron>)。

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

6.1.1.2. 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: 2
```



```

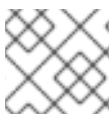
resources:
  limits:
    memory: 2Gi
  requests:
    cpu: 200m
    memory: 2Gi
storage: {}
redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    replicas: 1
curation:
  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

```

6.1.2. 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ること。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
      rsyslog:
        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、移動する必要が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。

例:

- 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

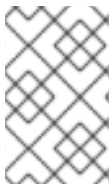
6.2. クラスターロギングの管理状態の変更

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。

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、CR に一致するようクラスターの更新を試行します。

OpenShift Container Platform ドキュメントでは、前提条件の段階でクラスターを 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

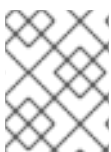
Elasticsearch Operator (EO) を管理外の状態に設定し、Cluster Logging Operator (CLO) を管理対象のままにする場合、CLO は EO に加えた変更を元に戻します。EO は CLO によって管理されているためです。

6.2.1. 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。

6.2.2. 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 クラスター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。

6.3. 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
      rsyslog:
        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。

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

**注記**

値は、環境によって異なる可能性があります。

6.4. 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。

Elasticsearch デプロイメントを以下を実行するように設定できます。

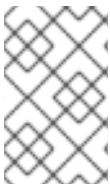
- Elasticsearch クラスターのストレージを設定します。
- シャードをクラスター内の複数のデータノードにレプリケートする方法 (完全なレプリケーションからレプリケーションなしを含む) を定義します。
- Elasticsearch データへの外部アクセスを設定します。

**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、ClusterLogging カスタムリソースに指定しない限り、メモリー要求および CPU 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。

6.4.1. 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こと。

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: ①
      limits:
        memory: "16Gi"
      requests:
        cpu: "1"
        memory: "16Gi"
```

- ① 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。

6.4.2. 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データノードに複製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こと。

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" ①
```

1 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/復元ストラテジーを実装している場合は、このモードを使用できます。

6.4.3. 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 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こと。

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。

6.4.4. 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こと。

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: {}
```

6.4.5. 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。

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

```
172.30.183.229
```

```
oc get service elasticsearch
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 -- 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こと。
- 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。以下は例になります。

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. 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": "ODdiNWlyYzAtMjg5Ni0TAAtNWE3MDY1MjMzNTc3",
        "_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"
}
}
]
}
}

```

6.4.6. 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付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。ノードにディスクを追加することを検討する必要があります。	alert

アラート	説明	重大度
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

6.5. 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。

6.5.1. 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: ❶
    limits:
      memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
  proxy: ❷
    resources:
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
```

- ❶ 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。
- ❷ 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。

6.5.2. 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 ❶
```

- ❶ Kibana ノードの数を指定します。

6.5.3. 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-l9mj7	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が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>
```

例:

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

6.6. 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 はその設定に基づき、日次ベースで動作を実行します。

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。

6.6.1. 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こと。

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せん。

6.6.2. 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

- クラスタログイン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6.1 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6.2 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。

変数名	説明
<code>pattern</code>	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-qe` 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。

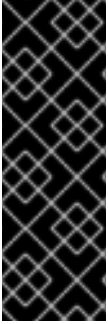
```
config.yaml: |
  .defaults:
    delete:
      days: 31

  .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。

6.6.3. 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こと。

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。

1. 次の Curator の反復をトリガーします。

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

2. 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。

6.6.4. 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使用します。

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

- a. Curator 設定ファイルおよび **action** ファイルを Curator 設定マップから削除し、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 \
  --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。

6.6.5. 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っ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。

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** `filters` 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。

6.7. FLUENTD の設定

OpenShift Container Platform は Fluentd を使用して、OpenShift Container Platform が Kubernetes Pod および Namespace メタデータで拡充した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、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。

6.7.1. Fluentd Pod の表示

oc get pods -o wide コマンドを使用して、Fluentd Pod 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 -o wide | grep fluentd
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
fluentd-5mr28	1/1	Running	0	4m56s	10.129.2.12	ip-10-0-164-233.ec2.internal
<none>						
fluentd-cnc4c	1/1	Running	0	4m56s	10.128.2.13	ip-10-0-155-142.ec2.internal
<none>						
fluentd-nlp8z	1/1	Running	0	4m56s	10.131.0.13	ip-10-0-138-77.ec2.internal
<none>						
fluentd-rknlk	1/1	Running	0	4m56s	10.128.0.33	ip-10-0-128-130.ec2.internal
<none>						
fluentd-rsm49	1/1	Running	0	4m56s	10.129.0.37	ip-10-0-163-191.ec2.internal
<none>						
fluentd-wjt8s	1/1	Running	0	4m56s	10.130.0.42	ip-10-0-156-251.ec2.internal
<none>						

6.7.2. Fluentd ログの表示

ログの表示方法は、**LOGGING_FILE_PATH** の設定によって異なります。

- **LOGGING_FILE_PATH** がファイルを参照する場合、デフォルトで、**logs** ユーティリティーを Pod が置かれているプロジェクトから使用し、Fluentd ログファイルの内容を出力します。

```
$ oc exec <any-fluentd-pod> -- logs 1
```

- 1** Fluentd Pod の名前を指定します。**logs** の前にスペースがあることに注意してください。

例:

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

現在の設定を表示するには、以下を実行します。

```
oc -n openshift-logging set env daemonset/fluentd --list | grep LOGGING_FILE_PATH
```

- **LOGGING_FILE_PATH=console** を使用している場合、Fluentd はログを stdout/stderr に書き込みます。**oc logs [-f] <pod_name>** コマンド (**-f** はオプション) を使用して、Pod が置かれているプロジェクトからログを取得できます。

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

- 1** Fluentd 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は最も古いログから順番に出力されます。

6.7.3. Fluentd 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
```

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...
spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: ①
            cpu: 250m
            memory: 1Gi
          requests:
            cpu: 250m
            memory: 1Gi
```

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

6.7.4. Fluentd ログの場所の設定

Fluentd は、**LOGGING_FILE_PATH** 環境変数に応じて、指定されたファイルか、またはデフォルトの場所の **/var/log/fluentd/fluentd.log** にログを書き込みます。

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。

手順

Fluentd ログの出力の場所を設定するには、以下を実行します。

1. **fluentd** daemonset で **LOGGING_FILE_PATH** パラメーターを編集します。特定のファイルまたは **console** を指定できます。

```
spec:
  template:
    spec:
      containers:
        env:
          - name: LOGGING_FILE_PATH
            value: console ❶
LOGGING_FILE_PATH= ❷
```

- ❶ ログの出力方法を指定します。

- **console** を使用して Fluentd のデフォルトの場所を使用します。 **oc logs [-f] <pod_name>** コマンドでログを取得します。
- **<path-to-log/fluentd.log>** を使用してログ出力を指定されたファイルに送信します。 **oc exec <pod_name> — logs** コマンドでログを取得します。これはデフォルトの設定です。または、CLI を使用します。

```
oc -n openshift-logging set env daemonset/fluentd LOGGING_FILE_PATH=console
```

6.7.5. Fluentd ログのスロットリング

とくに詳細なプロジェクトについては、管理者は処理される前の Fluentd によるログの読み取り速度を減速することができます。スロットリングによってログの読み取り速度が遅くなり、Kibana がレコードを表示するのにより長い時間がかかる可能性があります。



警告

スロットリングは設定されたプロジェクトのログ集計が遅れる一因になる可能性があります。Fluentd が追い付く前に Pod が削除された場合、ログエントリーが消失する可能性があります。



注記

Systemd ジャーナルをログソースとして使用している場合、スロットリングは機能しません。スロットリングの実装は、各プロジェクトの個々のログファイルの読み取りを調整できる機能によって決まります。ジャーナルからの読み取り時に、単一のログソースしか存在せず、ログファイルが存在しないと、ファイルベースのスロットリングは利用できません。Fluentd プロセスに読み込まれるログエントリーを制限する方法はありません。

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。

手順

1. Fluentd を特定プロジェクトを制限するように設定するには、デプロイメント後に Fluentd ConfigMap でスロットル設定を編集します。

```
$ oc edit configmap/fluentd
```

throttle-config.yaml キーの形式は、プロジェクト名と、各ノードでのログの読み取りに必要な速度が含まれる YAML ファイルです。デフォルトはノードごとに一度に 1000 行です。例:

```
throttle-config.yaml: |
- opensift-logging:
  read_lines_limit: 10
- .operations:
  read_lines_limit: 100
```

6.7.6. Fluentd のバッファークチャンクの制限について

Fluentd ロガーが多数のログを処理できない場合、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に切り換える必要があります。

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 **buffer_chunk_limit** は、デフォルト値が **8m** の環境変数 **BUFFER_SIZE_LIMIT** によって決定されます。出力ごとのファイルのバッファサイズは、デフォルト値が **256Mi** の環境変数 **FILE_BUFFER_LIMIT** によって決定されます。永続的なボリュームサイズは、**FILE_BUFFER_LIMIT** に出力を乗算した結果よりも大きくなければなりません。

Fluentd Pod では、永続ボリューム **/var/lib/fluentd** は PVC または **hostmount** などによって作成する必要があります。その領域はファイルバッファに使用されます。

buffer_type および **buffer_path** は、以下のように Fluentd 設定ファイルで設定されます。

```
$ egrep "buffer_type|buffer_path" *.conf
output-es-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-config`
output-es-ops-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-ops-config`
```

Fluentd **buffer_queue_limit** は変数 **BUFFER_QUEUE_LIMIT** の値です。この値はデフォルトで **32** になります。

環境変数 **BUFFER_QUEUE_LIMIT** は **(FILE_BUFFER_LIMIT / (number_of_outputs * BUFFER_SIZE_LIMIT))** として計算されます。

BUFFER_QUEUE_LIMIT 変数にデフォルトの値のセットが含まれる場合、以下のようになります。

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

- **BUFFER_SIZE_LIMIT = 8Mi**

buffer_queue_limit の値は **32** になります。**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。

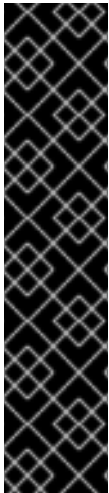
この数式では、**number_of_outputs** は、すべてのログが単一リソースに送信され、追加のリソースごとに 1 つずつ増分する場合に 1 になります。たとえば、**number_of_outputs** の値は以下ようになります。

- **1:** すべてのログが単一の Elasticsearch Pod に送信される場合
- **2:** アプリケーションログが Elasticsearch Pod に送信され、運用ログが別の Elasticsearch Pod に送信される場合
- **4:** アプリケーションログが Elasticsearch Pod に送信され、運用ログが別の Elasticsearch Pod に送信される場合で、それらがどちらも他の Fluentd インスタンスに転送される場合

6.7.7. Fluentd JSON 構文解析の設定

Fluentd を、メッセージが **JSON** 形式であるかどうかを判別するために各ログメッセージを検査し、そのメッセージを Elasticsearch に送信された JSON ペイロードドキュメントにマージできるように設定します。この機能はデフォルトでは無効にされています。

この機能は、**fluentd daemonset** で **MERGE_JSON_LOG** 環境変数を編集することによって有効にしたり、無効にしたりできます。



重要

この機能を有効することには、以下を含むリスクが伴います。

- Elasticsearch が整合性のないタイプのマッピングによりドキュメントを拒否することでログが失われる可能性があります。
- 拒否されるメッセージの繰り返し処理によってバッファーストレージ不足が生じる可能性があります。
- 同じ名前を持つフィールドのデータが上書きされます。

このトピックの機能は、経験のある Fluentd および Elasticsearch ユーザーによってのみ使用される必要があります。

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。

手順

以下のコマンドを使用してこの機能を有効にします。

```
oc set env ds/fluentd MERGE_JSON_LOG=true ①
```

- ① これを **false** に設定してこの機能を無効にするか、または **true** に設定してこの機能を有効にします。

MERGE_JSON_LOG および CDM_UNDEFINED_TO_STRING の設定

MERGE_JSON_LOG および **CDM_UNDEFINED_TO_STRING** 環境変数を **true** に設定する場合、Elasticsearch 400 エラーを受信する可能性があります。このエラーは、`MERGE_JSON_LOG=true` の場合に Fluentd がフィールドに **文字列** 以外のデータタイプを追加するために発生します。**CDM_UNDEFINED_TO_STRING=true** を設定する場合、Fluentd はそれらのフィールドに **文字列** の値を追加することを試行し、これにより Elasticsearch 400 エラーが生じます。このエラーはインデックスが翌日にロールオーバーされるとクリアされます。

Fluentd が次の日のログのインデックスをロールオーバーする場合、全く新しいインデックスが作成されます。フィールドの定義は更新され、400 エラーは発生しません。

スキーマ違反やデータ破損などの **ハード (hard)** エラーのあるレコードについては、再試行できません。Fluent はエラー処理のためにレコードを送信します。最後に表示される `<label>` のように Fluentd 設定に `<label @ERROR>` セクションを追加する場合、それらのレコードを随時処理することができます。

例:

```
data:
  fluent.conf:
  ....

  <label @ERROR>
    <match **>
      @type file
      path /var/log/fluent/dlq
      time_slice_format %Y%m%d
      time_slice_wait 10m
      time_format %Y%m%dT%H%M%S%z
      compress gzip
    </match>
  </label>
```

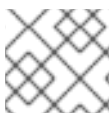
このセクションではエラーレコードを [Elasticsearch dead letter queue \(DLQ\) ファイル](#) に書き込みます。ファイル出力についての詳細は [fluentd のドキュメント](#) を参照してください。

次に、レコードを手動でクリーンアップし、ファイルを Elasticsearch `/_bulk index` API で使用し、`cURL` を使用してそれらのレコードを追加できるようにファイルを編集することができます。Elasticsearch Bulk API についての詳細は [Elasticsearch のドキュメント](#) を参照してください。

6.7.8. ログコレクターによってログを正規化する方法の設定

クラスターロギングは、データベーススキーマなどの特定のデータモデルを使用して、ログレコードとそのメタデータをロギングストアに格納します。このデータについてはいくつかの制限があります。

- 実際のログメッセージが含まれる **"message"** フィールドがなければなりません。
- RFC 3339 形式のログレコードのタイムスタンプを含む **"@timestamp"** フィールドがなければなりません (ミリ秒以上の単位の使用が望ましい)。
- **err**、**info**、**unknown** などのログレベルが含まれる **"level"** フィールドがなければなりません。



注記

データモデルの詳細については、「[Exported Fields](#)」を参照してください。

これらの要件により、異なるサブシステムから収集されるログデータで競合や不整合が生じる場合があります。

たとえば、**MERGE_JSON_LOG** 機能 (**MERGE_JSON_LOG=true**) を使用する場合、アプリケーションの出力のログを JSON で記録し、ログコレクターが Elasticsearch 内でデータを自動的に解析し、インデックス化すると非常に便利です。ただし、これにより、以下のような問題が発生します。

- フィールド名が空になることや、Elasticsearch で使用できない文字が含まれる可能性があります。
- 同じ namespace の異なるアプリケーションが、同じフィールド名を異なる値データタイプで出力する可能性があります。
- アプリケーションが過剰にフィールドを出力する可能性があります。
- フィールドはクラスターロギングのビルトインフィールドと競合する可能性があります。

以下の表で、Fluentd ログコレクターの daemonset、Fluentd または Rsyslog を編集し、環境変数を設定することで、クラスターロギングが異種ソースからフィールドを処理する方法を設定できます。

- 未定義のフィールド。異種システムのログデータに関する問題の1つとして、一部のフィールドが ViaQ データモデルで認識されない可能性があります。このようなフィールドは **undefined** と呼ばれます。ViaQ では、すべての最上位フィールドが定義され、記述される必要があります。

パラメーターを使用して OpenShift Container Platform が、**undefined** という最上位フィールド以下のすべての未定義のフィールドを移動し、**既知**の ViaQ の最上位フィールドとの競合を避けます。未定義フィールドを最上位フィールドに追加し、他のフィールドを **undefined** コンテナに移動できます。

また、未定義のフィールドで特殊文字を置き換えることができ、未定義フィールドを JSON 文字列表現に変換することもできます。JSON 文字列に変換しても値の構造を維持されるため、値を後で取得してマップまたは配列に戻すことができます。

- 番号やブール値などの簡単なスカラー値は、引用符で囲まれた文字列に変更されます。たとえば、**10** は "**10**" に、**3.1415** は "**3.1415**" に、**false** は "**false**" になります。
- map/dict の値と配列の値は JSON 文字列表現に変換されます: "**mapfield**":{"key":"value"}" は "**mapfield**":{"key":"value"}" に、"**arrayfield**":[1,2,"three"] は "**arrayfield**":["1,2,\"three\"]" に変換されます。
- 定義されたフィールド。ログの最上位に表示する定義されたフィールドを設定することもできます。

CDM_DEFAULT_KEEP_FIELDS パラメーターで定義されるデフォルトの最上位のフィールドは次のとおりです: **CEE, time, @timestamp, aushape, ci_job, collectd, docker, fedora-ci, file, foreman, geoip, hostname, ipaddr4, ipaddr6, kubernetes, level, message, namespace_name, namespace_uuid, offset, openstack, ovirt, pid, pipeline_metadata, rsyslog, service, systemd, tags, testcase, tlog, viaq_msg_id**

#{CDM_DEFAULT_KEEP_FIELDS} または **#{CDM_EXTRA_KEEP_FIELDS}** に含まれないフィールドはすべて、**CDM_USE_UNDEFINED** が **true** の場合、**#{CDM_UNDEFINED_NAME}** に移動されます。



注記

CDM_DEFAULT_KEEP_FIELDS パラメーターは、上級ユーザーのみが使用するか、Red Hat サポートによって使用することが指示された場合に使用されることが想定されます。

- 空のフィールド。異種ログから保持する空のフィールドを決定することができます。

表6.3 ログの正規化に使用する環境パラメーター

パラメーター	定義	例
CDM_EXTRA_KEEP_FIELDS	CDM_DEFAULT_KEEP_FIELDS に加えて、ログの最上位に保持する定義されたフィールドの追加セットを指定します。デフォルトは "" です。	CDM_EXTRA_KEEP_FIELDS="broker"
CDM_KEEP_EMPTY_FIELDS	CSV 形式で空のままでも保持するフィールドを指定します。空の定義されたフィールドで指定されないものはドロップされます。デフォルトは「message」で、空のメッセージを維持します。	CDM_KEEP_EMPTY_FIELDS="message"
CDM_USE_UNDEFINED	未定義のフィールドを undefined の最上位フィールドに移動するには、 true に設定します。デフォルトは false です。 true の場合、 CDM_DEFAULT_KEEP_FIELDS および CDM_EXTRA_KEEP_FIELDS の値は undefined に移動されません。	CDM_USE_UNDEFINED=true
CDM_UNDEFINED_NAME	CDM_USE_UNDEFINED を使用する場合は、未定義の最上位フィールドの名前を指定します。デフォルトは `undefined` です。 CDM_USE_UNDEFINED が true の場合にのみ有効にされます。	CDM_UNDEFINED_NAME="undef"
CDM_UNDEFINED_MAX_NUM_FIELDS	未定義フィールドの数がこの数を超える場合、未定義のフィールドはすべて JSON 文字列表現に変換され、 CDM_UNDEFINED_NAME フィールドに保存されます。レコードに未定義フィールド値を超える数が含まれる場合、これらのフィールドでの処理はこれ以上実行されません。代わりに、フィールドは最上位の CDM_UNDEFINED_NAME フィールドに保存された単一文字列の JSON 値に変換されます。デフォルトの -1 を維持することで、未定義フィールドの無制限の値が許可されますが、これは推奨されません。	CDM_UNDEFINED_MAX_NUM_FIELDS=4
	 <p>注記</p> <p>このパラメーターは CDM_USE_UNDEFINED が false の場合でも有効です。</p>	

パラメーター	定義	例
CDM_UNDEFINED_TO_STRING	未定義フィールドをすべて JSON 文字列表現に変換するには、 true に設定します。デフォルトは false です。	CDM_UNDEFINED_TO_STRING=true
CDM_UNDEFINED_DOT_REPLACE_CHARACTER	未定義フィールドでドット文字 '.' の代わりに使用する文字を指定します。 MERGE_JSON_LOG は true である必要があります。デフォルトは UNUSED です。 MERGE_JSON_LOG パラメーターを true に設定した場合は、以下の注を参照してください。	CDM_UNDEFINED_DOT_REPLACE_CHARACTER="_"

注記

ログコレクター daemonset の **MERGE_JSON_LOG** パラメーターおよび **CDM_UNDEFINED_TO_STRING** 環境変数を **true** に設定した場合、Elasticsearch 400 エラーを受信する可能性があります。このエラーは、`MERGE_JSON_LOG=true` の場合に、ログコレクターがフィールドに文字列以外のデータタイプを追加するために発生します。**CDM_UNDEFINED_TO_STRING=true**を設定する場合、ログコレクターはそれらのフィールドへの文字列の値の追加を試行し、これにより Elasticsearch 400 エラーが生じます。ログコレクターが翌日のログにインデックスをロールオーバーすると、エラーはクリアされます。

ログコレクターがインデックスをロールオーバーする場合、完全に新規のインデックスが作成されます。フィールドの定義は更新され、400 エラーは発生しません。

手順

CDM_* パラメーターを使用して、未定義および空のフィールドの処理を設定します。

- 必要に応じてフィールドの処理方法を設定します。
 - CDM_EXTRA_KEEP_FIELDS** を使用して移動するフィールドを指定します。
 - CSV 形式で **CDM_KEEP_EMPTY_FIELDS** パラメーターで保持する空のフィールドをすべて指定します。
- 必要に応じて未定義フィールドの処理方法を設定します。
 - CDM_USE_UNDEFINED** を **true** に設定し、未定義のフィールドを最上位の **undefined** フィールドに移動します。
 - CDM_UNDEFINED_NAME** パラメーターを使用して、未定義フィールドの名前を指定します。
 - CDM_UNDEFINED_MAX_NUM_FIELDS** をデフォルトの **-1** 以外の値に設定し、単一コードにおける未定義フィールド数の上限を設定します。
- CDM_UNDEFINED_DOT_REPLACE_CHARACTER** を指定して、未定義フィールド名のすべてのドット (.) 文字を別の文字に変更します。たとえば、**CDM_UNDEFINED_DOT_REPLACE_CHARACTER=@@@** の場合で、**foo.bar.baz** という名前のフィールドがある場合、そのフィールドは **foo@@@bar@@@baz** に変換されます。

4. **UNDEFINED_TO_STRING** を **true** に設定し、未定義フィールドを JSON 文字列表現に変換します。

注記

CDM_UNDEFINED_TO_STRING または **CDM_UNDEFINED_MAX_NUM_FIELDS** パラメーターを設定する場合、**CDM_UNDEFINED_NAME** を使用して未定義フィールドの名前を変更します。**CDM_UNDEFINED_TO_STRING** または **CDM_UNDEFINED_MAX_NUM_FIELDS** が未定義フィールドの値タイプを変更する可能性があるため、このフィールドは必要になります。**CDM_UNDEFINED_TO_STRING** または **CDM_UNDEFINED_MAX_NUM_FIELDS** が **true** に設定されていて、ログにさらに多くの未定義フィールドがある場合、値タイプは **string** になります。値タイプが変更される場合、Elasticsearch はレコードの受け入れを停止します (JSON から JSON 文字列への変更など)。

たとえば、**CDM_UNDEFINED_TO_STRING** が **false** であるか、または **CDM_UNDEFINED_MAX_NUM_FIELDS** がデフォルトの **-1** の場合、未定義フィールドの値タイプは **json** になります。**CDM_UNDEFINED_MAX_NUM_FIELDS** をデフォルト以外の値に変更し、ログにさらに多くの未定義フィールドがある場合、値タイプは **string** (json string) になります。値タイプが変更された場合、Elasticsearch はレコードの受け入れを停止します。

6.7.9. 環境変数の使用による Fluentd の設定

環境変数を使用して Fluentd 設定を変更することができます。

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。

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>
```

例:

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

6.8. OPENSIFT CONTAINER PLATFORM ログの外部デバイスへの送信

Elasticsearch ログを、外部でホストされた Elasticsearch インスタンスまたは外部 syslog サーバーなどの外部デバイスに送信することができます。また、ログを外部ログアグリゲーターに送信するように Fluentd を設定することもできます。

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。

6.8.1. Fluentd をログを外部 Elasticsearch インスタンスに送信するように設定する

Fluentd は、Elasticsearch デプロイメント設定の **ES_HOST**、**ES_PORT**、**OPS_HOST**、および **OPS_PORT** 環境変数の値にログを送信します。アプリケーションログは **ES_HOST** の宛先に、操作ログは **OPS_HOST** の宛先に送信されます。



注記

AWS Elasticsearch インスタンスへのログの直接送信はサポートされていません。Fluentd Secure Forward を使用して、**fluent-plugin-aws-elasticsearch-service** プラグインで設定した制御対象の Fluentd のインスタンスにログを送信してください。

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こと。
- クラスタロギングを管理外の状態に設定する。

手順

ログを特定の Elasticsearch インスタンスに送信するには、以下を実行します。

1. **openshift-logging** プロジェクトで **fluentd** daemonset を編集します。

```
$ oc edit ds/fluentd

spec:
  template:
    spec:
      containers:
        env:
          - name: ES_HOST
            value: elasticsearch
          - name: ES_PORT
            value: '9200'
          - name: ES_CLIENT_CERT
            value: /etc/fluent/keys/app-cert
          - name: ES_CLIENT_KEY
            value: /etc/fluent/keys/app-key
          - name: ES_CA
            value: /etc/fluent/keys/app-ca
          - name: OPS_HOST
            value: elasticsearch
          - name: OPS_PORT
            value: '9200'
          - name: OPS_CLIENT_CERT
            value: /etc/fluent/keys/infra-cert
          - name: OPS_CLIENT_KEY
            value: /etc/fluent/keys/infra-key
          - name: OPS_CA
            value: /etc/fluent/keys/infra-ca
```

2. 外部 Elasticsearch インスタンスにアプリケーションログと操作ログの両方を含めるには、**ES_HOST** と **OPS_HOST** を同じ宛先に設定し、**ES_PORT** と **OPS_PORT** が同一の値を持つことを確認します。
3. TLS の外部でホストされる Elasticsearch インスタンスを設定します。相互 TLS を使用する外部でホストされた Elasticsearch インスタンスのみが許可されます。



注記

指定された Kibana と Elasticsearch イメージを使用していない場合、同じマルチテナント機能は利用できず、データは特定プロジェクトへのユーザーアクセスによる制限を受けません。

6.8.2. Fluentd をログを外部 syslog サーバーに送信するように設定する

fluent-plugin-remote-syslog プラグインをホストで使用して、ログを外部 syslog サーバーに送信します。

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。

手順

1. **openshift-logging** プロジェクトの **fluentd** daemonset で環境変数を設定します。

```
spec:
  template:
    spec:
      containers:
        - name: fluentd
          image: 'registry.redhat.io/openshift4/ose-logging-fluentd:v4.1'
          env:
            - name: REMOTE_SYSLOG_HOST ❶
              value: host1
            - name: REMOTE_SYSLOG_HOST_BACKUP
              value: host2
            - name: REMOTE_SYSLOG_PORT_BACKUP
              value: 5555
```

- ❶ 必要なりモート syslog ホスト。各ホストで必須です。

これによって2つの宛先が作成されます。**host1** の syslog サーバーはデフォルトポート **514** でメッセージを受信し、**host2** は同じメッセージをポート **5555** で受信します。

2. または、**openshift-logging** プロジェクトに独自の **fluentd** daemonset を設定できます。

Fluentd 環境変数

パラメーター	説明
USE_REMOTE_SYSLOG	デフォルトは false です。 fluent-plugin-remote-syslog gem を使用できるようにするには、 true に設定します。
REMOTE_SYSLOG_HOST	(必須) リモート syslog サーバーのホスト名または IP アドレス。
REMOTE_SYSLOG_PORT	接続先のポート番号。デフォルトは 514 です。

パラメーター	説明
REMOTE_SYSLOG_SEVERITY	syslog の重大度を設定します。デフォルトは debug です。
REMOTE_SYSLOG_FACILITY	syslog ファシリティを設定します。デフォルトは local0 です。
REMOTE_SYSLOG_USE_RECORD	デフォルトは false です。レコードの重大度フィールドおよびファシリティフィールドを使用して syslog メッセージに設定するには、 true に設定します。
REMOTE_SYSLOG_REMOVE_TAG_PREFIX	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。
REMOTE_SYSLOG_TAG_KEY	これが指定されている場合、このフィールドをキーとして使用してレコードを検索し、syslog メッセージにタグを設定します。
REMOTE_SYSLOG_PAYLOAD_KEY	これが指定されている場合、このフィールドをキーとして使用してレコードを検索し、syslog メッセージにペイロードを設定します。
REMOTE_SYSLOG_TYPE	トランスポート層プロトコルタイプを設定します。デフォルトは syslog_buffered になり、これにより、TCP プロトコルが設定されます。UDP に切り替えるには、これを syslog に設定します。



警告

この実装は安全ではないため、接続にスヌーピングがないことを保証できる環境でのみ使用してください。

6.8.3. ログを外部ログアグリゲーターに送信するように Fluentd を設定する

out_forward プラグインを使用して、デフォルトの Elasticsearch 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するように Fluentd を設定することができます。ローカルにホストされている Fluentd による処理の後に、ログレコードをさらに処理することができます。

forward プラグインは Fluentd によってのみサポートされます。**out_forward** プラグインはクライアント側 (セNDER) を実装し、**in_forward** プラグインはサーバー側 (レシーバー) を実装します。

Out_forward を使用してログを送信するように OpenShift Container Platform を設定するには、受信側を参照する **openshift -logging namespace** に **secure-forward** という ConfigMap を作成します。レシーバーで、**in_forward** プラグインを OpenShift Container Platform からログを受信するように設定します。**in_forward** プラグインの使用方法についての詳細は、[Fluentd ドキュメント](#) を参照してください。

デフォルトの `secure-forward.conf` セクション

```

# <store>
# @type forward
# <security>
# self_hostname ${hostname} # ${hostname} is a placeholder.
# shared_key <shared_key_between_forwarder_and_forwardee>
# </security>
# transport tls
# tls_verify_hostname true      # Set false to ignore server cert hostname.

# tls_cert_path /path/for/certificate/ca_cert.pem
# <buffer>
# @type file
# path '/var/lib/fluentd/forward'
# 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 server.fqdn.example.com # or IP
# port 24284
# </server>
# <server>
# host 203.0.113.8 # ip address to connect
# name server.fqdn.example.com # The name of the server. Used for logging and certificate
# verification in TLS transport (when host is address).
# </server>
# </store>

```

手順

Fluentd ログのコピーを外部ログアグリゲーターに送信するには、以下を実行します。

1. Fluentd 設定マップの `secure-forward.conf` セクションを編集します。

```
$ oc edit configmap/fluentd -n openshift-logging
```

2. 外部 Fluentd サーバーの名前、ホスト、およびポートを入力します。

```

# <server>
# host server.fqdn.example.com # or IP
# port 24284
# </server>
# <server>
# host 203.0.113.8 # ip address to connect

```

```
# name server.fqdn.example.com # The name of the server. Used for logging and
certificate verification in TLS transport (when host is address).
# </server>
```

例:

```
<server>
  name externalserver1 ❶
  host 192.168.1.1 ❷
  port 24224 ❸
</server>
<server> ❹
  name externalserver1
  host 192.168.1.2
  port 24224
</server>
</store>
```

- ❶ オプションで、この外部のアグリゲーターの名前を入力します。
- ❷ 外部アグリゲーターのホスト名または IP を指定します。
- ❸ 外部アグリゲーターのポートを指定します。
- ❹ オプションで、追加の外部のアグリゲーターを追加します。

3. CA 証明書とプライベートキーへのパスを **secure-forward.conf** セクションに追加します。

```
# <security>
# self_hostname ${hostname} # ${hostname} is a placeholder. ❶
# shared_key <shared_key_between_forwarder_and_forwardee> ❷
# </security>

# tls_cert_path /path/for/certificate/ca_cert.pem ❸
```

- ❶ 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。
- ❷ 認証に共有キーを指定します。
- ❸ CA 証明書へのパスを指定します。

以下は例になります。

```
<security>
  self_hostname client.fqdn.local
  shared_key cluster_logging_key
</security>

tls_cert_path /etc/fluent/keys/ca.crt
```

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。

4. **secure-forward.conf** で使用される証明書を Fluentd Pod にマウントされる既存のシークレットに追加します。 **your_ca_cert** と **your_private_key** の値は **configmap/logging-fluentd** の **secure-forward.conf** で指定されている値と一致している必要があります。

```
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/your_ca_cert','value':'$(base64 -w0
/path/to/your_ca_cert.pem)}]"
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/your_private_key','value':'$(base64 -w0
/path/to/your_private_key.pem)}]"
```



注記

your_private_key は、汎用的な名前に置き換えます。これは、JSON パスへのリンクであり、使用しているホストシステムのパスではありません。

以下は例になります。

```
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/ca.crt','value':'$(base64 -w0 /etc/fluent/keys/ca.crt)}]"
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/ext-agg','value':'$(base64 -w0 /etc/fluent/keys/ext-agg.pem)}]"
```

5. 外部アグリゲーターで **secure-forward.conf** ファイルを設定し、Fluentd からのメッセージを安全に受け入れられるようにします。
外部アグリゲーターを設定する際は、Fluentd からのメッセージを安全に受信する必要があります。

詳細情報は、[inforward プラグインのセットアップ方法](#)について、および [out_forward プラグイン](#)を参照してください。

6.9. SYSTEMD-JOURNALD および RSYSLOG の設定

Fluentd および rsyslog 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。

6.9.1. 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=no 1
ForwardToConsole=yes 2
ForwardToSyslog=no 3
MaxRetentionSec=30s 4
RateLimitBurst=10000 5
RateLimitInterval=1s 6
Storage=volatile 7
SyncIntervalSec=1s 8
SystemMaxUse=8g 9
SystemKeepFree=20% 10
SystemMaxFileSize10M 11
```

- 1 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 3 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。
- 4 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。
- 5 6 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。
- 7 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。
- 8

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、

- 9 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。
- 10 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。
- 11 `/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。

例:

```
...
config:
  storage:
    files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,{jrnl_cnf}
          verification: {}
        filesystem: root
        mode: 0644 ①
        path: /etc/systemd/journald.conf ②
    systemd: {}
```

- 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7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。

7.1. 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こと。

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
```

例:

```
$ 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: ❶
clusterHealth: green ❷
conditions: ❸
.....
nodes: ❹
.....
pods: ❺
.....
shardAllocationEnabled: "All" ❻
```

- ❶ 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。
- ❷ Elasticsearch クラスターのステータス (**green**、**red**、**yellow**)。
- ❸ 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

このメッセージは、Elasticsearch ノードのステータスが変更されたときに、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である。
- 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。

7.1.1. 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況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

```

7.2. 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            chOwDnQISsqhfSPcot1Yiw 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:v4.1
    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:v4.1
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
              period=5s #success=1 #failure=3
...
Events:     <none>

```

第8章 ノードセクターを使用したクラスターロギングリソースの移動

ノードセクターを使用して Elasticsearch、Kibana、Curator Pod を異なるノードにデプロイすることができます。

8.1. ノードセクターを使用したクラスターロギングコンポーネントのノードの指定

各コンポーネントの仕様により、コンポーネントは特定のノードにターゲットを設定できます。

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こと。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. 必要なラベルをノードに追加します。

```
$ oc label <resource> <name> <key>=<value>
```

たとえば、ノードにラベルを付けるには、以下を実行します。

```
$ oc label nodes ip-10-0-142-25.ec2.internal type=elasticsearch
```

2. **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: "nodeselector"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeSelector: ❶
        logging: es
      nodeCount: 1
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 200m
        memory: 2Gi
    storage:
      size: "20G"
      storageClassName: "gp2"
      redundancyPolicy: "ZeroRedundancy"
  visualization:
```

```
type: "kibana"
kibana:
  nodeSelector: ❷
  logging: kibana
  replicas: 1
curation:
  type: "curator"
  curator:
    nodeSelector: ❸
    logging: curator
    schedule: "*/10 * * * *"
collection:
  logs:
    type: "fluentd"
    fluentd:
      nodeSelector: ❹
      logging: fluentd
```

- ❶ Elasticsearch のノードセレクター。
- ❷ Kibana のノードセレクター。
- ❸ Curator のノードセレクター。
- ❹ Fluentd のノードセレクター。

第9章 追加リソース

- ノードセクターについての詳細は、[「Placing pods on specific nodes using node selectors」](#)を参照してください。

第10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。

10.1. 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

- クラスターログインおよび Elasticsearch がインストールされていること。

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'
```

例:

```
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シングを防ぎます。

```
$ 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" } }'
```

例:

```
$ 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>
```

例:

```
$ 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>
```

例:

```
$ 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。

例:

```
$ 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" } }'
```

例:

```
$ 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11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。

11.1. 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
```

11.2. 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
```

11.3. 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12章 エクスポートされるフィールド

ロギング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、ロギング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-rsyslog**
- **exported-fields-systemd**
- **exported-fields-kubernetes**
- **exported-fields-pipeline_metadata**
- **exported-fields-ovirt**
- **exported-fields-aushape**
- **exported-fields-tlog**

12.1. デフォルトのエクスポートされるフィールド

これらは、ロギング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。

パラメーター	説明
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。
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指定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> <ul style="list-style-type: none"> * ロギングシステムが認識しない値を取得する際は unknown のみを使用します。これは最も高いレベル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、詳細度が高いと見なされます。 * error の使用は推奨されていません。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>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 および rsyslog programname プロパティはサービスフィールドにマップされます。

パラメーター	説明
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。
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視覚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視覚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。

パラメーター	説明
collectd.type	type: string collectd タイプ。
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。

パラメーター	説明
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_cputime.user	type: float TODO
collectd.processes.ps_cputime.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_count.processes	type: integer TODO
collectd.processes.ps_count.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_pagefaults.majflt	type: float TODO
collectd.processes.ps_pagefaults.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_disk_octets.read	type: float TODO
collectd.processes.ps_disk_octets.write	type: float TODO
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_merged.read	type: float TODO
collectd.disk.disk_merged.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
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
--------	----

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。

パラメーター	説明
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。
collectd.statsd.host_nic_tx	type: integer collectd nic_tx タイプの statsd プラグイン。
collectd.statsd.host_nic_rx_dropped	type: integer collectd nic_rx_dropped タイプの statsd プラグイン。
collectd.statsd.host_nic_tx_dropped	type: integer collectd nic_tx_dropped タイプの statsd プラグイン。
collectd.statsd.host_nic_rx_errors	type: integer collectd nic_rx_errors タイプの statsd プラグイン。
collectd.statsd.host_nic_tx_errors	type: integer collectd nic_tx_errors タイプの statsd プラグイン。
collectd.statsd.host_storage	type: integer collectd ストレージタイプの statsd プラグイン。
collectd.statsd.host_swap	type: integer collectd swap タイプの statsd プラグイン。
collectd.statsd.host_vdsm	type: integer collectd VDSM タイプの statsd プラグイン。

パラメーター	説明
<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 <code>nic_tx_dropped</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_rx_bytes</code>	type: integer collectd <code>nic_rx_bytes</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_nic_tx_bytes</code>	type: integer collectd <code>nic_tx_bytes</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_balloon_min</code>	type: integer collectd <code>balloon_min</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_balloon_max</code>	type: integer collectd <code>balloon_max</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_balloon_target</code>	type: integer collectd <code>balloon_target</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_balloon_cur</code>	type: integer collectd <code>balloon_cur</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_cpu_sys</code>	type: integer collectd <code>cpu_sys</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_cpu_usage</code>	type: integer collectd <code>cpu_usage</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_read_ops</code>	type: integer collectd <code>disk_read_ops</code> タイプの statsd プラグイン。
<code>collectd.statsd.vm_disk_write_ops</code>	type: integer collectd <code>disk_write_ops</code> タイプの statsd プラグイン。

パラメーター	説明
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。
collectd.statsd.vm_disk_true_size	type: integer collectd disk_true_size タイプの statsd プラグイン。
collectd.statsd.vm_disk_read_rate	type: integer collectd disk_read_rate タイプの statsd プラグイン。
collectd.statsd.vm_disk_write_latency	type: integer collectd disk_write_latency タイプの statsd プラグイン。
collectd.statsd.vm_disk_read_latency	type: integer collectd disk_read_latency タイプの statsd プラグイン。
collectd.statsd.vm_disk_read_bytes	type: integer collectd disk_read_bytes タイプの statsd プラグイン。
collectd.statsd.vm_nic_rx_dropped	type: integer collectd nic_rx_dropped タイプの statsd プラグイン。
collectd.statsd.vm_cpu_user	type: integer collectd cpu_user タイプの statsd プラグイン。
collectd.statsd.vm_nic_rx_errors	type: integer collectd nic_rx_errors タイプの statsd プラグイン。

パラメーター	説明
<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。
<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_umbackends</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。

12.2. RSYSLOG 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、ロギングシステムによってエクスポートされる **rsyslog** フィールドであり、Elasticsearch および Kibana での検索に利用できます。

以下のフィールドは RFC5424 ベースのメタデータです。

パラメーター	説明
<code>rsyslog.facility</code>	rsyslog についての詳細は、 syslog 仕様を参照してください。

パラメーター	説明
<code>rsyslog.protocol-version</code>	これは rsyslog プロトコルバージョンです。
<code>rsyslog.structured-data</code>	syslog の Structured-Data についての詳細は、 syslog 仕様を参照してください。
<code>rsyslog.msgid</code>	これは syslog msgid フィールドです。
<code>rsyslog.appname</code>	app-name が programname と同じ場合、最上位フィールド service のみに入力します。App -name が programname と同等でない場合、このフィールドは app-name を保持します。詳細は、syslog 仕様を参照してください。

12.3. 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。
<code>systemd.k.UDEV_SYSNAME</code>	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ー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。
systemd.t.SYSTEMD_OWNER_UID	systemd.t.SYSTEMD_OWNER_UID は、セッションの所有者 ID です。
systemd.t.SYSTEMD_SESSION	systemd.t.SYSTEMD_SESSION は、(該当する場合) systemd セッション ID です。
systemd.t.SYSTEMD_SLICE	systemd.t.SYSTEMD_SLICE 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。

パラメーター	説明
systemd.t.SYSTEMD_UNIT	systemd.t.SYSTEMD_UNIT は、セッションのユニット名です。
systemd.t.SYSTEMD_USER_UNIT	systemd.t.SYSTEMD_USER_UNIT は、(該当する場合) セッションのユーザーユニット名です。
systemd.t.TRANSPORT	systemd.t.TRANSPORT 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、 audit 、 driver 、 syslog 、 journal 、 stdout 、および kernel が含まれます。
systemd.t.UID	systemd.t.UID は、ジャーナルエントリープロセスのユーザー ID です。
systemd.t.SYSLOG_FACILITY	systemd.t.SYSLOG_FACILITY は、 syslog 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。
systemd.t.SYSLOG_IDENTIFIER	systemd.t.systemd.t.SYSLOG_IDENTIFIER は、 syslog の識別子です。
systemd.t.SYSLOG_PID	SYSLOG_PID は、 syslog 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
systemd.u.CODE_FILE	systemd.u.CODE_FILE は、ソースのファイル名が含まれるコードの場所です。
systemd.u.CODE_FUNCTION	systemd.u.CODE_FUNCTION は、ソースの関数が含まれるコードの場所です。
systemd.u.CODE_LINE	systemd.u.CODE_LINE は、ソースの行数が含まれるコードの場所です。
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。

12.4. 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。
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。

12.5. 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
<code>pipeline_metadata.collect_or.hostname</code>	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。
<code>pipeline_metadata.collect_or.name</code>	コレクターの名前。
<code>pipeline_metadata.collect_or.version</code>	コレクターのバージョン。
<code>pipeline_metadata.collect_or.ipaddr4</code>	コレクターサーバーの IP アドレス v4。配列である場合があります。
<code>pipeline_metadata.collect_or.ipaddr6</code>	コレクターサーバーの IP アドレス v6。配列である場合があります。
<code>pipeline_metadata.collect_or.inputname</code>	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 <code>imjournal/imfile</code> 。
<code>pipeline_metadata.collect_or.received_at</code>	メッセージがコレクターによって受信された時間。
<code>pipeline_metadata.collect_or.original_raw_message</code>	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。

パラメーター	説明
pipeline_metadata.normalizer.received_at	メッセージがノーマライザーによって受信された時間。
pipeline_metadata.normalizer.original_raw_message	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。
pipeline_metadata.trace	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。

12.6. 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
ovirt.entity	データソース、ホスト、VMS、およびエンジンのタイプ。
ovirt.host_id	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。

12.7. 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。

パラメーター	説明
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

12.8. 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。
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)。

パラメーター	説明
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13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せん。

13.1. 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こと。

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
```