OpenShift Container Platform 4.7

ロギング

OpenShift Logging のインストール、使用法、およびリリースノート

Last Updated: 2023-02-24

OpenShift Container Platform 4.7 ロギング

OpenShift Logging のインストール、使用法、およびリリースノート

法律上の通知

概要

本書では、OpenShift Logging 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。
OpenShift Logging 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 RED HAT OPENSHIFT LOGGING のリリースノート

1.1. 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

1.2. サポート対象バージョン

表1.1 Red Hat OpenShift Logging (RHOL) の OpenShift Container Platform バージョンのサポート

	4.7	4.8	4.9
RHOL 5.1	X	X	
RHOL 5.2	X	X	X
RHOL 5.3		X	X

1.2.1. OpenShift Logging 5.1.0

このリリースには、RHSA-2021:2112 OpenShift Logging のバグ修正リリース 5.1.0 が含まれます。

1.2.1.1. 新機能および機能拡張

OpenShift Logging 5.1 は、以下で実行されている OpenShift Container Platform 4.7 以降をサポートするようになりました。

- IBM Power Systems
- IBM Z および LinuxONE

今回のリリースでは、以下のコンポーネントおよび概念に関連する拡張機能が追加されました。

- クラスター管理者は、Kubernetes Pod ラベルを使用してアプリケーションからログデータを収集し、特定のログストアに送信します。**ClusterLogForwarder** カスタムリソース (CR) YAML ファイルに **inputs[].application.selector.matchLabels** 要素を設定してログデータを収集できます。namespace で収集したログデータをフィルターすることもできます。(LOG-883)
- 今回のリリースにより、以下の新しい **ElasticsearchNodeDiskWatermarkReached** 警告が OpenShift Elasticsearch Operator (EO) に追加されました。
  - Elasticsearch Node Disk Low Watermark Reached (Elasticsearch ノードのディスクで低い基準値に達する)
  - Elasticsearch Node Disk High Watermark Reached (Elasticsearch ノードのディスクで高い基準値に達する)

- Elasticsearch Node Disk Flood Watermark Reached (Elasticsearch ノードのディスクがいっぱいの基準値に達する)

アラートは、今後 6 時間で Elasticsearch ノードが **Disk Low Watermark**、**Disk High Watermark** または **Disk Flood Stage Watermark** のしきい値に到達すると推測した場合に、過去の警告が複数適用されます。この警告期間があることで、ノードがディスクの基準しきい値に到達するまでに対応する時間ができます。警告メッセージには、トラブルシューティング手順へのリンクも含まれており、この問題の軽減に役立ちます。EO は、過去数時間のディスク領域データをリニアモデルに適用し、これらの警告を生成します。(LOG-1100)

- JSON ログは、引用符で囲まれた文字列ではなく、JSON オブジェクトとして、Red Hat の管理対象 Elasticsearch クラスターまたはその他のサポート対象のサードパーティーのシステムのいずれかに、転送できるようになりました。さらに、Kibana 内の JSON ログメッセージから個別のフィールドをクエリーできるようになり、特定のログの検出性が向上します。(LOG-785, LOG-1148)

1.2.1.2. 非推奨および削除された機能

以前のリリースで利用可能であった一部の機能が非推奨になるか、または削除されました。

非推奨の機能は依然として OpenShift Container Logging に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

1.2.1.2.1. Elasticsearch Curator が削除される

今回の更新では、Elasticsearch Curator が削除され、サポート対象外になりました。Elasticsearch Curator は、OpenShift Container Platform 4.4 以前でインデックスをキュレートして管理するのに便利でした。Elasticsearch Curator を使用する代わりに、ログの保持時間を設定します。

1.2.1.2.2. 従来の Fluentd および従来の syslog メソッドを使用したログの転送が非推奨に

OpenShift Container Platform 4.6 から現在まで、レガシー Fluentd およびレガシー syslog メソッドを使用したログの転送は非推奨になり、将来のリリースで削除される予定です。代わりに、標準の非レガシーメソッドを使用してください。

1.2.1.3. バグ修正

- 今回の更新前は、**ClusterLogForwarder** CR では **input[].selector** 要素の作成後に表示されませんでした。今回の更新により、**ClusterLogForwarder** CR で **セレクター** を指定すると、表示されたままになります。このバグを修正するには LOG-883 で必要でした。これにより、Pod ラベルセレクターを使用してアプリケーションログデータを転送できるようになります。(LOG-1338)

- 今回の更新前は、クラスターサービスバージョン (CSV) の更新により、OpenShift Elasticsearch Operator コンテナーのリソースおよび制限が誤って導入されました。特定の条件下では、これにより Elasticsearch Operator Pod を終了するメモリー不足の状態が生じました。今回の更新では、CSV リソースおよび Operator コンテナーの制限を削除して、この問題は修正されています。Operator は問題なくスケジュールされるようになりました。(LOG-1254)

- 今回の更新前は、チェーンされた証明書を使用して Kafka にログを転送すると、以下のエラーメッセージで失敗していました。
state=error: certificate verify failed (ローカル発行者証明書を取得できない)

  中間 CA によって署名された証明書を使用して Kafka ブローカーにログを転送できませんでした。これは、Fluentd Kafka プラグインが対応するシークレットの **ca-bundle.crt** エントリーに

指定された単一の CA 証明書しか処理できないために生じました。今回の更新では、対応するシークレットの **ca-bundle.crt** エントリーで指定される複数の CA 証明書を処理できるように Fluentd Kafka プラグインを有効にすることで、この問題は修正されています。これで、中間 CA によって署名された証明書でログを Kafka ブローカーに転送できるようになりました。(LOG-1218, LOG-1216)

- 今回の更新前は、負荷のある場合に、Elasticsearch はクラスターに問題がない場合でも HTTP 500 エラーを出して一部の要求に応答しました。要求の再試行は正常に実行されました。今回の更新では、一時的な HTTP 500 エラーが生じる場合に、インデックス管理の cron ジョブを回復性が強化されるように更新し、この問題が修正されています。更新されたインデックス管理の cron ジョブは、要求を複数回再試行してから失敗します。(LOG-1215)

- 今回の更新前には、クラスターインストール設定で **.proxy** の値を指定せずに、インストールされたクラスターでグローバルプロキシーを設定している場合、Fluentd がログを Elasticsearch に転送できないというバグがありました。この問題を回避するには、プロキシーまたはクラスター設定で **no_proxy** を **.svc.cluster.local** に設定して、内部トラフィックがスキップされるようにします。今回の更新では、プロキシー設定の問題が修正されています。OpenShift Container Platform クラスターのインストール後にグローバルプロキシーを設定する場合には、Fluentd はログを Elasticsearch に転送するようになりました。(LOG-1187, BZ#1915448)

- 今回の更新以前は、ロギングコレクターは必要以上のソケット接続を作成しました。今回の更新により、ロギングコレクターは既存のソケット接続を再利用してログを送信します。(LOG-1186)

- 今回の更新前は、クラスター管理者が Elasticsearch クラスターからストレージの追加または削除を試行する場合に、OpenShift Elasticsearch Operator (EO) は **scheduledUpgrade: "True"**、**shardAllocationEnabled: primaries** と表示して Elasticsearch クラスターを誤ってアップグレードしてボリュームを変更しようとしていました。今回の更新により、EO は Elasticsearch クラスターのアップグレードを試行しなくなりました。
  EO ステータスには、以下の新規ステータス情報が表示され、無視された Elasticsearch ストレージへのサポート対象外の変更を加えようとしたタイミングを示します。

  - **StorageStructureChangeIgnored**。一時ストレージ構造を使用してから永続ストレージ構造を使用しようとするまでに変更を加えようとした場合。

  - **StorageClassNameChangeIgnored**。ストレージのクラス名を変更しようとした場合、

  - **StorageSizeChangeIgnored**。ストレージサイズを変更しようとした場合。

注記

ClusterLogging カスタムリソース (CR) を一時ストレージから永続ストレージに切り替えるように設定する場合に、EO は永続ボリューム要求 (PVC) を作成しますが、永続ボリューム (PV) は作成されません。**StorageStructureChangeIgnored** ステータスを削除するには、**ClusterLogging** CR への変更を元に戻し、永続ボリューム要求 (PVC) を削除する必要があります。

(LOG-1351)

- 今回の更新前は、全 Elasticsearch クラスターを再デプロイすると、データ以外のノードが 1 つ実行中となり、それ以外の全データノードがシャットダウンし、正常でない状態で停止しました。この問題は、新規証明書が原因で Elasticsearch Operator が Elasticsearch クラスターのデータ以外のノードがスケールダウンできなくなったため発生していました。今回の更新によ

り、Elasticsearch Operator はすべてのデータおよびデータ以外のノードがスケールダウンして再びバックアップし、新規証明書が読み込まれるようになりました。Elasticsearch Operator は、新規証明書の読み込み後に新規ノードに到達できます。(LOG-1536)

1.2.2. OpenShift Logging 5.0.9

このリリースには、RHBA-2021:3705 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.9) が含まれます。

1.2.2.1. バグ修正

このリリースには、次のバグ修正が含まれています。

- 今回の更新以前は、一部のログエントリーで認識されない UTF-8 バイトが含まれていたため、Elasticsearch はメッセージを拒否し、バッファーリングされたペイロード全体をブロックしていました。この更新により、問題が解決されます。拒否されたペイロードは、無効なログエントリーを削除し、残りのエントリーを再送信します。(LOG-1574)

- この更新の前は、**ClusterLogging** カスタムリソース (CR) を編集しても、**totalLimitSize** の値が Fluentd **total_limit_size** フィールドに適用されなかったため、バッファープラグインインスタンスのサイズが制限されていました。その結果、Fluentd はデフォルト値を適用しました。この更新により、CR は **totalLimitSize** の値を Fluentd **total_limit_size** フィールドに適用します。Fluentd は、**total_limit_size** フィールドの値またはデフォルト値のいずれか小さい方を使用します。(LOG-1736)

1.2.2.2. CVE

- CVE-2020-25648
- CVE-2021-22922
- CVE-2021-22923
- CVE-2021-22924
- CVE-2021-36222
- CVE-2021-37576
- CVE-2021-37750
- CVE-2021-38201

1.2.3. OpenShift Logging 5.0.8

このリリースには、RHBA-2021:3526 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.8) が含まれます。

1.2.3.1. バグ修正

本リリースには、以下のバグ修正も含まれます。

- リリースパイプラインスクリプトの問題により、**olm.skipRange** フィールドの値は **5.2.0** のまま変更されず、z ストリーム番号 **0** が増加しても更新されませんでした。現在のリリースでは、リリース番号が変更されたときにこのフィールドの値を更新するようにパイプラインスク

リプトが修正されています。(LOG-1741)

1.2.4. OpenShift Logging 5.0.7

このリリースには、RHBA-2021:2884 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.7) が含まれます。

1.2.4.1. バグ修正

本リリースには、以下のバグ修正も含まれます。

- LOG-1594 Vendored viaq/logerr の依存関係にライセンスファイルがない

1.2.4.2. CVE

- CVE-2016-10228
- CVE-2017-14502
- CVE-2018-25011
- CVE-2019-2708
- CVE-2019-3842
- CVE-2019-9169
- CVE-2019-13012
- CVE-2019-18276
- CVE-2019-18811
- CVE-2019-19523
- CVE-2019-19528
- CVE-2019-25013
- CVE-2020-0431
- CVE-2020-8231
- CVE-2020-8284
- CVE-2020-8285
- CVE-2020-8286
- CVE-2020-8927
- CVE-2020-9948
- CVE-2020-9951
- CVE-2020-9983

- CVE-2020-10543
- CVE-2020-10878
- CVE-2020-11608
- CVE-2020-12114
- CVE-2020-12362
- CVE-2020-12363
- CVE-2020-12364
- CVE-2020-12464
- CVE-2020-13434
- CVE-2020-13543
- CVE-2020-13584
- CVE-2020-13776
- CVE-2020-14314
- CVE-2020-14344
- CVE-2020-14345
- CVE-2020-14346
- CVE-2020-14347
- CVE-2020-14356
- CVE-2020-14360
- CVE-2020-14361
- CVE-2020-14362
- CVE-2020-14363
- CVE-2020-15358
- CVE-2020-15437
- CVE-2020-24394
- CVE-2020-24977
- CVE-2020-25212
- CVE-2020-25284
- CVE-2020-25285

- CVE-2020-25643
- CVE-2020-25704
- CVE-2020-25712
- CVE-2020-26116
- CVE-2020-26137
- CVE-2020-26541
- CVE-2020-27618
- CVE-2020-27619
- CVE-2020-27786
- CVE-2020-27835
- CVE-2020-28196
- CVE-2020-28974
- CVE-2020-29361
- CVE-2020-29362
- CVE-2020-29363
- CVE-2020-35508
- CVE-2020-36322
- CVE-2020-36328
- CVE-2020-36329
- CVE-2021-0342
- CVE-2021-0605
- CVE-2021-3177
- CVE-2021-3326
- CVE-2021-3501
- CVE-2021-3516
- CVE-2021-3517
- CVE-2021-3518
- CVE-2021-3520
- CVE-2021-3537

- CVE-2021-3541
- CVE-2021-3543
- CVE-2021-20271
- CVE-2021-23336
- CVE-2021-27219
- CVE-2021-33034

1.2.5. OpenShift Logging 5.0.6

このリリースには、RHBA-2021:2655 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.6) が含まれます。

1.2.5.1. バグ修正

本リリースには、以下のバグ修正も含まれます。

- LOG-1451 - [1927249] fieldmanager.go:186] [発生すべきでない] [name="POLICY_MAPPING"] キーの managedFields... の重複エントリーを更新できない (LOG-1451)
- LOG-1537: ES クラスターにデータ以外のノードが含まれる場合に完全なクラスター証明書の再デプロイが失敗する (LOG-1537)
- LOG-1430 - eventrouter で "Observed a panic: &runtime.TypeAssertionError" が発生する (LOG-1430)
- LOG-1461: ジョブログにエラーがある場合でも、インデックス管理ジョブのステータスが常に **Completed** になる。(LOG-1461)
- LOG-1459 - Operator に非接続アノテーションがない (LOG-1459)
- LOG-1572 - Bug 1981579: 全ログを収集する組み込みアプリケーションの動作を修正 (LOG-1572)

1.2.5.2. CVE

- CVE-2016-10228
- CVE-2017-14502
- CVE-2018-25011
- CVE-2019-2708
- CVE-2019-9169
- CVE-2019-25013
- CVE-2020-8231
- CVE-2020-8284
- CVE-2020-8285

- CVE-2020-8286
- CVE-2020-8927
- CVE-2020-10543
- CVE-2020-10878
- CVE-2020-13434
- CVE-2020-14344
- CVE-2020-14345
- CVE-2020-14346
- CVE-2020-14347
- CVE-2020-14360
- CVE-2020-14361
- CVE-2020-14362
- CVE-2020-14363
- CVE-2020-15358
- CVE-2020-25712
- CVE-2020-26116
- CVE-2020-26137
- CVE-2020-26541
- CVE-2020-27618
- CVE-2020-27619
- CVE-2020-28196
- CVE-2020-29361
- CVE-2020-29362
- CVE-2020-29363
- CVE-2020-36328
- CVE-2020-36329
- CVE-2021-3177
- CVE-2021-3326
- CVE-2021-3516

- CVE-2021-3517
- CVE-2021-3518
- CVE-2021-3520
- CVE-2021-3537
- CVE-2021-3541
- CVE-2021-20271
- CVE-2021-23336
- CVE-2021-27219
- CVE-2021-33034

1.2.6. OpenShift Logging 5.0.5

このリリースには、RHSA-2021:2374 - Security Advisory、Moderate: Openshift Logging Bug Fix Release (5.0.5) が含まれます。

1.2.6.1. セキュリティー修正

- gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がありません。(**CVE-2021-3121**)
- glib: 64 ビットから 32 ビットに暗黙的にキャストされることが原因で、64 ビットプラットフォームにおける g_bytes_new 関数で整数オーバーフローが発生します (**CVE-2021-27219**)。

以下の問題は、上記の CVE に関連しています。

- BZ#1921650 gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がない (**BZ#1921650**)
- LOG-1361 CVE-2021-3121 elasticsearch-operator-container: gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がない [openshift-logging-5] (**LOG-1361**)
- LOG-1362 CVE-2021-3121 elasticsearch-proxy-container: gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がない [openshift-logging-5] (**LOG-1362**)
- LOG-1363 CVE-2021-3121 logging-eventrouter-container: gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がない [openshift-logging-5] (**LOG-1363**)

1.2.7. OpenShift Logging 5.0.4

このリリースには、RHSA-2021:2136 - Security Advisory、Moderate: Openshift Logging security and bugs update (5.0.4) が含まれます。

1.2.7.1. セキュリティー修正

- gogo/protobuf: plugin/unmarshal/unmarshal.go には特定のインデックス検証がありません。(CVE-2021-3121)

以下の Jira の問題には、上記の CVE が含まれます。

- LOG-1364 CVE-2021-3121 cluster-logging-operator-container: gogo/protobuf: plugin/unmarshal/unmarshal.go lacks certain index validation [openshift-logging-5].(LOG-1364)

1.2.7.2. バグ修正

本リリースには、以下のバグ修正も含まれます。

- LOG-1328 ポートの 5.0.z に対する修正 (BZ-1945168)(LOG-1328)

1.2.8. OpenShift Logging 5.0.3

このリリースには、RHSA-2021:1515 - Security Advisory、Important OpenShift Logging Bug Fix Release (5.0.3) が含まれます。

1.2.8.1. セキュリティー修正

- jackson-databind: jackson-databind: slf4j-ext クラスでの任意のコード実行 (CVE-2018-14718)
- jackson-databind: blaze-ds-opt および blaze-ds-core クラスでの任意のコード実行 (CVE-2018-14719)
- jackson-databind: 一部の JDK クラスでの exfiltration/XXE (CVE-2018-14720)
- jackson-databind: axis2-jaxws クラスでの SSRF (server-side request forgery) CVE-2018-14721)
- jackson-databind: axis2-transport-jms クラスでの不適切なポリモーフィックなデシリアライズ (CVE-2018-19360)
- jackson-databind: openjpa クラスでの適切でないポリモーフィックなデシリアライズ (CVE-2018-19361)
- jackson-databind: jboss-common-core クラスでの適切でないポリモーフィックなデシリアライズ (CVE-2018-19362)
- jackson-databind: リモートコード実行につながるデフォルトのタイプミス (CVE-2019-14379)
- jackson-databind: com.pastdev.httpcomponents.configuration.JndiConfiguration のシリアル化ガジェット (CVE-2020-24750)
- jackson-databind: org.apache.commons.dbcp2.datasources.PerUserPoolDataSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-35490)
- jackson-databind: org.apache.commons.dbcp2.datasources.SharedPoolDataSource(CVE-2020-35491) に関連するシリアル化ガジェットと入力間の対話を誤操作 (CVE-2020-35491)
- jackson-databind: com.oracle.wls.shaded.org.apache.xalan.lib.sql.JNDIConnectionPool に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-35728)

- jackson-databind: oadd.org.apache.commons.dbcp.cpdsadapter.DriverAdapterCPDS に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36179)。
- jackson-databind: org.apache.commons.dbcp2.cpdsadapter.DriverAdapterCPDS に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36180)
- jackson-databind: org.apache.tomcat.dbcp.dbcp.dbcp.cpdsadapter.DriverAdapterCPDS に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36181)
- jackson-databind: org.apache.tomcat.dbcp.dbcp2.cpdsadapter.DriverAdapterCPDS に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36182)
- jackson-databind: org.docx4j.org.apache.xalan.lib.sql.JNDIConnectionPool に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36183)
- jackson-databind: org.apache.tomcat.dbcp.dbcp2.datasources.PerUserPoolDataSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36184)
- jackson-databind: org.apache.tomcat.dbcp.dbcp2.datasources.SharedPoolDataSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36185)
- jackson-databind: org.apache.tomcat.dbcp.dbcp.datasources.PerUserPoolDataSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36186)
- jackson-databind: org.apache.tomcat.dbcp.dbcp.datasources.SharedPoolDataSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36187)
- jackson-databind: com.newrelic.agent.deps.ch.qos.logback.core.db.JNDIConnectionSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36188)
- jackson-databind: com.newrelic.agent.deps.ch.qos.logback.core.db.DriverManagerConnectionSource に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2020-36189)
- jackson-databind: javax.swing に関連するシリアル化ガジェットと入力間の対話の誤操作 (CVE-2021-20190)
- golang: ReverseProxy を含む特定の net/http サーバーでのデータ競合が DoS 引き起こす可能性がある (CVE-2020-15586)
- golang: ReadUvarint および ReadVarint が 無効な入力から無制限のバイト数を読み取る可能性がある (CVE-2020-16845)
- OpenJDK: JAR 署名が完了されない場合にアルゴリズム (ライブラリー、8249906) が無効になる (CVE-2021-2163)

以下の Jira の問題には、上記の CVE が含まれます。

- LOG-1234 CVE-2020-15586 CVE-2020-16845 openshift-eventrouter: 各種の不具合 [openshift-4](LOG-1234)
- LOG-1243 CVE-2018-14718 CVE-2018-14719 CVE-2018-14720 CVE-2018-14721 CVE-2018-19360 CVE-2018-19361 CVE-2018-19362 CVE-2019-14379 CVE-2020-35490 CVE-2020-35491 CVE-2020-35728... logging-elasticsearch6-container: 各種の不具合 [openshift-logging-5.0](LOG-1243)

1.2.8.2. バグ修正

本リリースには、以下のバグ修正も含まれます。

- LOG-1224 リリース 5.0 - ClusterLogForwarder namespace 固有のログ転送は予想通りに機能しない。(LOG-1224)
- LOG-1232 5.0 - Bug 1859004 - イベントルーターがイベントログを収集しないことがあります。(LOG-1232)
- LOG-1299 Release 5.0 - チェーンされた証明書を使用して Kafka にログを転送すると、state=error: certificate verify failed (unable to get local issuer certificate) というエラーを出して失敗します。(LOG-1299)

1.2.9. OpenShift Logging 5.0.2

このリリースには、RHBA-2021:1167 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.2) が含まれます。

1.2.9.1. バグ修正

- クラスターインストール設定で **.proxy** を設定しておらず、その後にインストールされたクラスターでグローバルプロキシーを設定している場合、Fluentd がログを Elasticsearch に転送できないというバグがありました。この問題を回避するには、プロキシー/クラスター設定で **no_proxy** を **.svc.cluster.local** に設定して、内部トラフィックがスキップされるようにします。現行リリースでは、プロキシー設定の問題が修正されています。OpenShift クラスターのインストール後にグローバルプロキシーを設定する場合、Fluentd はログを Elasticsearch に転送するようになりました。(LOG-1187)
- 以前のバージョンでは、チェーンされた証明書を使用して Kafka にログを転送すると、state=error: certificate verify failed (ローカル発行者の証明書の取得できない) というエラーを出して失敗しました。 中間 CA によって署名された証明書を使用して Kafka ブローカーにログを転送できませんでした。これは、fluentd Kafka プラグインが対応するシークレットの ca-bundle.crt エントリーに指定された単一の CA 証明書しか処理できないために生じました。現行リリースでは、対応するシークレットの ca-bundle.crt エントリーで指定される複数の CA 証明書を処理できるように fluentd Kafka プラグインを有効にすることで、この問題は修正されています。これで、中間 CA によって署名された証明書でログを Kafka ブローカーに転送できるようになりました。(LOG-1216, LOG-1218)
- 以前のバージョンでは、クラスターサービスバージョン (CSV) の更新により、OpenShift Elasticsearch Operator コンテナーのリソースおよび制限が誤って導入されました。特定の条件下では、これにより Elasticsearch Operator Pod を終了するメモリー不足の状態が生じました。現行リリースでは、CSV リソースおよび Operator コンテナーの制限を削除することで、この問題は修正されています。Operator は問題なくスケジュールされるようになりました。(LOG-1254)

1.2.10. OpenShift Logging 5.0.1

このリリースには、RHBA-2021:0963 - Bug Fix Advisory、OpenShift Logging Bug Fix Release (5.0.1) が含まれます。

1.2.10.1. バグ修正

- 以前のバージョンでは、レガシーログ転送を有効にしている場合に、ログは管理対象ストレージに送信されませんでした。この問題は、生成されるログ転送設定でログ転送またはレガシーログ転送のいずれかの間の不適切に選択が行われるために発生しました。現在のリリースではこの問題は修正されています。**ClusterLogging** CR が **logstore** を定義する場合、ログは管理

対象ストレージに送信されます。さらに、レガシーログ転送を有効にすると、管理対象ストレージが有効化されているかどうかにかかわらず、ログはレガシーログ転送に送信されます。(LOG-1172)

- 以前のバージョンでは、負荷のある場合に、Elasticsearch はクラスターに問題がない場合でも HTTP 500 エラーを出して一部の要求に応答しました。要求の再試行は正常に実行されました。本リリースでは、一時的な HTTP 500 エラーが生じる場合に cron ジョブを回復性を強化するために更新し、この問題が修正されています。今回のリリースより、障害が発生する前に要求を複数回再試行するようになりました。(LOG-1215)

1.2.11. OpenShift Logging 5.0.0

このリリースには、RHBA-2021:0652 - Bug Fix Advisory、Errata Advisory for Openshift Logging 5.0.0 が含まれます。

1.2.11.1. 新機能および機能拡張

今回のリリースでは、以下の概念に関連する拡張機能が追加されました。

クラスターロギングが Red Hat OpenShift Logging に
今回のリリースにより、Cluster Logging は Red Hat OpenShift Logging 5.0 になりました。

インデックスあたり最大 5 つのプライマリーシャード
今回のリリースにより、OpenShift Elasticsearch Operator (EO) は、クラスターに定義されるデータノードの数に応じて 1 から 5 までのインデックスのプライマリーシャードの数を設定するようになりました。

以前のバージョンでは、EO はインデックスのシャード数をデータノードの数に設定していました。Elasticsearch のインデックスが数多くのレプリカで設定されている場合、それはインデックスごとにではなく、プライマリーシャードごとに多数のレプリカを作成しました。そのため、インデックスがシャード化されると、多数のレプリカシャードがクラスターに存在するため、クラスターでの複製および同期に多くのオーバーヘッドが発生しました。

OpenShift Elasticsearch Operator 名および成熟度レベルの更新
今回のリリースにより、OpenShift Elasticsearch Operator の表示名および Operator の成熟度レベルが更新されました。OpenShift Elasticsearch Operator の新規の表示名および明確化された特定の使用法についての更新が Operator Hub で行われています。

OpenShift Elasticsearch Operator の CSV の成功についてのレポート
今回のリリースにより、レポートメトリクスが追加され、OpenShift Elasticsearch Operator の **ClusterServiceVersion** (CSV) オブジェクトのインストールまたはアップグレードが正常に行われたことが示唆されるようになりました。以前のバージョンでは、CSV のインストールまたはアップグレードに失敗した場合に、アラートを判別したり、アラートを生成したりする方法はありませんでした。アラートは OpenShift Elasticsearch Operator の一部として提供されるようになりました。

Elasticsearch Pod 証明書のパーミッション関連の警告の縮小
以前のバージョンでは、Elasticsearch Pod の起動時に、証明書パーミッションに関する警告が生成され、クラスターのトラブルシューティングについて一部のユーザーに誤解を与えました。現在のリリースでは、これらのパーミッション関連の問題が修正され、これらの種類の通知が縮小されました。

アラートから説明およびトラブルシューティングへの新規リンク
今回のリリースにより、Elasticsearch クラスターが生成するアラートから、そのアラートの説明およびトラブルシューティング手順のページへのリンクが追加されました。

削除ジョブの新規の接続タイムアウト
現在のリリースでは、削除ジョブの接続タイムアウトが追加されました。これは、 Pod がインデック

スを削除するために Elasticsearch に対してクエリーを実行する際に Pod がハングすることを防ぐのに役立ちます。タイムアウトが経過する前に、基礎となる curl 呼び出しが接続できない場合、タイムアウトは呼び出しを終了します。

ロールオーバーインデックステンプレートへの更新を最小限に抑える
今回の拡張機能により、OpenShift Elasticsearch Operator は、フィールド値が異なる場合にのみロールオーバーインデックステンプレートを更新するようになりました。インデックステンプレートはインデックスよりも優先度が高くなります。テンプレートが更新されると、クラスターはそれらをインデックスシャードに分散し、パフォーマンスに影響が及びます。Elasticsearch クラスターの操作を最小限にするために、Operator は、プライマリーシャードまたはレプリカシャードの数が現在設定されている内容から変更される場合にのみテンプレートを更新します。

1.2.11.2. テクノロジープレビューの機能

現在、今回のリリースに含まれる機能にはテクノロジープレビューのものがあります。これらの実験的機能は、実稼働環境での使用を目的としていません。これらの機能に関しては、Red Hat カスタマーポータルの以下のサポート範囲を参照してください。

テクノロジープレビュー機能のサポート範囲

以下の表では、機能は以下のステータスでマークされています。

- **TP: テクノロジープレビュー**
- **GA: 一般公開機能**
- **-: 利用不可の機能**

表1.2 テクノロジープレビュートラッカー

機能	OCP 4.5	OCP 4.6	Logging 5.0
ログ転送	TP	GA	GA

1.2.11.3. 非推奨および削除された機能

以前のリリースで利用可能であった一部の機能が非推奨になるか、または削除されました。

非推奨の機能は依然として OpenShift Container Logging に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

1.2.11.3.1. ElasticsearchCurator は非推奨になりました

Elasticsearch Curator は非推奨になり、将来のリリースで削除される予定です。Elasticsearch Curator は、OpenShift Container Platform 4.4 以前でインデックスをキュレートして管理するのに便利でした。Elasticsearch Curator を使用する代わりに、ログの保持時間を設定します。

1.2.11.3.2. 従来の Fluentd および従来の syslog メソッドを使用したログの転送が非推奨に

OpenShift Container Platform 4.6 から現在まで、レガシー Fluentd およびレガシー syslog メソッドを使用したログの転送は非推奨になり、将来のリリースで削除される予定です。代わりに、標準の非レガシーメソッドを使用してください。

1.2.11.4. バグ修正

- 以前のバージョンでは、Elasticsearch はヘッダーがデフォルトの最大ヘッダーサイズの 8 KB を超える HTTP 要求を拒否していました。最大ヘッダーサイズは 128 KB となり、Elasticsearch は最大ヘッダーサイズを超える HTTP 要求を拒否しなくなりました。(BZ#1845293)

- 以前のバージョンでは、ソフトウェアのバグにより Elasticsearch カスタムリソース (CR) のノードのステータスが正しく更新されないために、ノードは **Pending** ステータスから回復しませんでした。現在のリリースでは、この問題は修正され、ノードはステータスが **Pending** の場合に回復できるようになりました。(BZ#1887357)

- 以前のバージョンでは、Cluster Logging Operator (CLO) は **clusterlogging** CR の Elasticsearch ノードの数を 3 つのノードにスケールダウンする場合、一意の ID を持つ以前に作成されたノードを省略しました。OpenShift Elasticsearch Operator は、一意の ID を持つノードを削除することを防ぐ予防策があったために更新を拒否しました。CLO がノード数をスケールダウンし、Elasticsearch CR を更新すると、CLO は一意の ID のあるノードを省略せず、それらのノードにカウント 0 のマークを付けます。その結果、ユーザーは **clusterlogging** CR を使用してクラスターを 3 つのノードにスケールダウンできます。(BZ#1879150)

注記

OpenShift Logging 5.0 以降では、Cluster Logging Operator は Red Hat OpenShift Logging Operator と呼ばれます。

- 以前のバージョンでは、**ClusterLogForwarder** でシークレットが誤って設定されていると、Fluentd コレクター Pod はクラッシュループに陥りました。現在のリリースではこの問題は修正されています。**ClusterLogForwarder** はシークレットを検証し、ステータスフィールドにエラーを報告するようになりました。結果として、Fluentd コレクター Pod がクラッシュしなくなりました。(BZ#1888943)

- 以前のバージョンでは、**clusterlogging** インスタンスの Kibana リソース設定を **resource{}** に更新した場合、生成される nil マップはパニックを発生させ、OpenShift Elasticsearch Operator のステータスを **CrashLoopBackOff** に変更しました。現在のリリースでは、マップの初期化によりこの問題は修正されています。(BZ#1889573)

- 以前のバージョンでは、ClusterLogForwarder に同じシークレットを使用する複数の出力がある場合に、fluentd コレクター Pod はクラッシュループに陥りました。現在のリリースではこの問題は修正されています。複数の出力がシークレットを共有できるようになりました。(BZ#1890072)

- 以前のバージョンでは、Kibana ルートを削除した場合、Cluster Logging Operator (CLO) はこれを復旧したり、再作成したりすることができませんでした。今回のリリースより、CLO はルートを監視し、ルートを削除すると、OpenShift Elasticsearch Operator はこれを調整するか、または再作成できるようになりました。(BZ#1890825)

- 以前のバージョンでは、Cluster Logging Operator (CLO) は、Red Hat が提供する Elastic Resource Definition (CRD) に依存する Elasticsearch リソースの調整を試行しました。不明な種類の一覧表示を試みると、CLO はその調整ループを終了しました。これは、CLO がそれらが定義されているかに関係なく、管理対象のリソースすべての調整を試行するために生じました。現在のリリースではこの問題は修正されています。CLO は、ユーザーが管理対象ストレージを定義する場合に OpenShift Elasticsearch Operator によって提供されるタイプのみを調整します。これにより、ユーザーは CLO をデプロイしてクラスターロギングのコレクターのみのデプロイメントを作成できます。(BZ#1891738)

- 以前のバージョンでは、RFC 3164 の LF GA syslog 実装により、リモート syslog に送信されるログはレガシーの動作と互換性がありませんでした。現在のリリースではこの問題は修正され

ています。AddLogSource はログのソースの詳細を message フィールドに追加します。リモート syslog に送信されるログは、レガシー動作と互換性を持つようになりました。(BZ#1891886)

- 以前のバージョンでは、Elasticsearch のロールオーバー Pod は **resource_already_exists_exception** エラーで失敗しました。Elasticsearch ロールオーバー API 内では、次のインデックスが作成されると ***-write** エイリアスがこれを参照するように更新されませんでした。その結果、次にロールオーバー API エンドポイントが特定のインデックスについてトリガーされると、リソースがすでに存在することを示すエラーを受信しました。現在のリリースではこの問題は修正されています。ロールオーバーが **indexmanagement** cronjob で生じる差異に、新規インデックスが作成されていると、エイリアスが新規インデックスをポイントしていることを検証するようになりました。この動作により、エラーが発生しないようになります。クラスターがすでにこのエラーを受信している場合、cronjob は問題を修正し、後続の実行が予想通りに機能するようにします。今回のリリースより、ロールオーバーを実行しても例外が生成されなくなりました。(BZ#1893992)

- 以前のバージョンでは、Fluent はロギングスタックが機能しているように見える場合でもログの送信を停止しました。エンドポイントがバックアップされていても、ログは長期間エンドポイントに送られませんでした。これは、最大バックオフ時間が長過ぎエンドポイントが停止した場合に生じました。現在のリリースでは、最大バックオフ時間を引下げ、ログをより早く送ることができるようにすることでこの問題が解決されています。(BZ#1894634)

- 以前のバージョンでは、OpenShift Elasticsearch ノードのストレージサイズを省略すると、Elasticsearch Operator コードでパニックが発生していました。このパニックは以下のようにログに出力されました。**Observed a panic: "invalid memory address or nil pointer dereference"** パニックは、ストレージサイズが必須フィールドにもかかわらずソフトウェアがこれをチェックしないために発生しました。現在のリリースでは、この問題は修正され、ストレージサイズが省略されてもパニックが発生しなくなりました。代わりに、ストレージはデフォルトで一時ストレージに設定され、ユーザーのログメッセージを生成します。(BZ#1899589)

- 以前のバージョンでは、**elasticsearch-rollover** および **elasticsearch-delete** Pod は、**Invalid JSON:** または **ValueError: No JSON object could be decoded** のエラー状態のままになりました。この例外は、無効な JSON 入力の例外ハンドラーがないために発生しました。現在のリリースでは、無効な JSON 入力のハンドラーを提供することで、この問題は修正されています。その結果、ハンドラーは例外トレースバックではなくエラーメッセージを出力し、**elasticsearch-rollover** および **elasticsearch-delete** ジョブはそれらのエラー状態を維持しなくなりました。(BZ#1899905)

- 以前のバージョンでは、Fluentd をスタンドアロンとしてデプロイする際に、**replicas** の値が **0** の場合でも Kibana Pod が作成されました。これは、Elasticsearch ノードがない場合でも Kibana はデフォルトで **1** Pod に設定されていたために生じました。現在のリリースでこの問題は修正されています。Kibana は、1 つ以上の Elasticsearch ノードがある場合にのみデフォルトで **1** に設定されるようになりました。(BZ#1901424)

- 以前のバージョンでは、シークレットを削除すると再作成されませんでした。証明書は Operator のローカルディスクに存在していても、それらは変更されていないために書き換えられませんでした。つまり、証明書は変更された場合にのみ作成されました。現在のリリースではこの問題は修正されています。証明書が変更された場合や見つからない場合は、シークレットが再作成されるようになりました。マスター証明書を削除すると、それらは置き換えられます。(BZ#1901869)

- 以前のバージョンでは、クラスターに同じ名前を持つ複数のカスタムリソースがある場合、API グループで完全修飾されていない場合にリソースはアルファベット順に選択されました。その結果、OpenShift Elastic Elasticsearch Operator と共に Red Hat の OpenShift Elasticsearch Operator をインストールしている場合、must-gather レポートを使用してデータを収集すると

失敗が生じました。現在のリリースでは、must-gather がクラスターのカスタムリソースについての情報を収集する際に完全な API グループを使用できるようにすることで、この問題は修正されています。(BZ#1897731)

- 証明書の生成に関連する問題に対応する以前のバグ修正により、エラーが生じました。証明書の読み取りを試みると、証明書が不足していると認識されるためにこれが再生成されました。これにより、OpenShift Elasticsearch Operator がトリガーされ、Elasticsearch クラスターでのローリングアップグレードが実行され、一致しない証明書が生成されることがありました。このバグは、Operator が証明書を作業ディレクトリーに誤って書き込むことによって生じました。現在のリリースではこの問題は修正されています。Operator は、常に同じ作業ディレクトリーに対して証明書の読み取りおよび書き込みを行い、証明書は必要に応じて証明書を再生成されるようになりました。(BZ#1905910)

- 以前のバージョンでは、Elasticsearch バージョンを取得するためにルートエンドポイントに対してクエリーを実行すると、403 応答が受信されました。403 応答は、以前のリリースでこのエンドポイントを使用していたサービスを中断しました。このエラーは、管理者以外のユーザーにルートエンドポイントのクエリーおよび Elasticsearch バージョンの取得に必要な **monitor** パーミッションがないために生じました。管理者以外のユーザーは、デプロイされた Elasticsearch のバージョンのルートエンドポイントをクエリーできるようになりました。(BZ#1906765)

- 以前のバージョンでは、一部のバルク挿入の状況で、Elasticsearch プロキシーは fluentd と Elasticsearch 間の接続をタイムアウトしました。その結果、fluentd はメッセージの配信に失敗し、**Server returned nothing (no headers, no data)** エラーを返しました。現在のリリースではこの問題は修正され、Elasticsearch プロキシーのデフォルトの HTTP の読み取りおよび書き込みタイムアウトを 5 秒から 1 分に引き上げられました。また、フィールドの HTTP タイムアウトを制御するための Elasticsearch プロキシーのコマンドラインオプションも提供されます。(BZ#1908707)

- 以前のバージョンでは、{ProductName}/Elasticsearch ダッシュボードは OpenShift Container Platform モニターリングダッシュボードから欠落していました。これは、ダッシュボード設定リソースが異なる namespace の所有者を参照し、OpenShift Container Platform がそのリソースのガベージコレクションを行うために生じました。所有者の参照は OpenShift Elasticsearch Operator の調整機能 (reconciler) の設定から削除され、ロギングダッシュボードがコンソールに表示されるようになりました。(BZ#1910259)

- 以前のバージョンでは、環境変数を使用して Kibana 設定ファイルの値を置き換えるコードは、コメント化された行を考慮しませんでした。これにより、ユーザーは server.maxPayloadBytes のデフォルト値を上書きすることができませんでした。現在のリリースでは、server.maxPayloadByteswithin のデフォルト値のコメントを解除でき、この問題は修正されています。ユーザーは、説明されている通りに環境変数を使用して値を上書きできるようになりました。(BZ#1918876)

- 以前のバージョンでは、移行に失敗したインデックスを削除する命令を非表示にしないように Kibana ログレベルが引き上げられました。これにより、Kibana ユーザーのメールアドレスおよび OAuth トークンが含まれる INFO レベルの GET 要求も表示されました。現在のリリースでは、これらのフィールドをマスクすることでこの問題が修正され、Kibana ログにそれが表示されなくなりました。(BZ#1925081)

1.2.11.5. 既知の問題

- **ruby-kafka-1.1.0** および **fluent-plugin-kafka-0.13.1** gem を持つ Fluentd Pod には Apache Kafka バージョン 0.10.1.0 との互換性はありません。
そのため、Kafka へのログ転送が **error_class=Kafka::DeliveryFailed error="Failed to send messages to flux-openshift-v4/1"** のメッセージを出して失敗します。

ruby-kafka-0.7 gem は Kafka 0.11 のネイティブサポートを優先して Kafka 0.10 のサポートを中断しました。**ruby-kafka-1.0.0** gem は Kafka 2.3 および 2.4 のサポートを追加しました。現行バージョンの OpenShift Logging は Kafka バージョン 2.4.1 をテストし、これをサポートしています。

この問題を回避するには、サポートされるバージョンの Apache Kafka にアップグレードします。

(BZ#1907370)

第2章 RED HAT OPENSHIFT LOGGING について

クラスター管理者は、OpenShift Logging をデプロイし、ノードシステムの監査ログ、アプリケーションコンテナーログ、およびインフラストラクチャーログなどの OpenShift Container Platform クラスターからのすべてのログを集計できます。OpenShift Logging はクラスター全体でこれらのログを集計し、それらをデフォルトのログストアに保存します。Kibana Web コンソールを使用して、ログデータを可視化 できます。

OpenShift Logging は以下のタイプのログを集計します。

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

注記

内部 OpenShift Container Platform Elasticsearch ログストアは監査ログのセキュアなストレージを提供しないため、デフォルトで監査ログは内部 Elasticsearch インスタンスに保存されません。監査ログを内部ログストアに送信する必要がある場合 (Kibana で監査ログを表示するなど)、監査ログのログストアへの転送 で説明されているようにログ転送 API を使用する必要があります。

2.1. OPENSHIFT LOGGING のデプロイについて

OpenShift Container Platform クラスター管理者は、OpenShift Container Platform Web コンソールまたは CLI コマンドを使用して OpenShift Logging をデプロイし、OpenShift Elasticsearch Operator および Red Hat OpenShift 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、 **ClusterLogging** カスタムリソース (Custom Resource、CR) を作成して OpenShift Logging Pod および OpenShift Logging 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 OpenShift Logging のデプロイ、アップグレード、および維持を行います。

ClusterLogging 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 OpenShift Logging 環境を定義します。Red Hat OpenShift Logging Operator は OpenShift Logging CR 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、表示アクセスのあるプロジェクトのログを表示できます。

詳細は、ログコレクターの設定 について参照してください。

2.1.1. JSON OpenShift コンテナープラットフォームロギング

JSON ロギングを使用して、JSON 文字列を構造化オブジェクトに解析するようにログ転送 API を設定できます。以下のタスクを実行します。

- JSON ログの解析
- Elasticsearch の JSON ログデータの設定

- JSON ログの Elasticsearch ログストアへの転送

詳細は、 JSON ログ を参照してください。

2.1.2. Kubernetes イベントの収集および保存

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

詳細は、 About collecting and storing Kubernetes events について参照してください。

2.1.3. OpenShift Container Platform ロギングの更新

OpenShift Container Platform を使用すると、OpenShift Container Platform のロギングを更新できます。OpenShift Container Platform Logging の更新時には、以下の Operator を更新する必要があります。

- Elasticsearch Operator
- Cluster Logging Operator

詳細は、 OpenShift Container Platform Logging の更新 を参照してください。

2.1.4. クラスターダッシュボードの表示

OpenShift Container Platform Logging ダッシュボードには、クラスターレベルで Elasticsearch インスタンスに関する詳細を示すチャートが含まれています。これらのチャートは、問題の診断と予測に役立ちます。

詳細は、About viewing the cluster dashboard を参照してください。

2.1.5. OpenShift Container Platform ロギングのトラブルシューティング

次のタスクを実行してログの問題をトラブルシューティングできます。

- ロギングステータスの表示
- ログストアのステータスの表示
- ロギングアラートの理解
- Red Hat サポート用のロギングデータの収集
- Critical Alerts のトラブルシューティング

2.1.6. OpenShift Container Platform ロギングのアンインストール

ClusterLogging カスタムリソース (CR) を削除して、ログ集計を停止できます。CR の削除後に残る他のクラスターロギングコンポーネントがあり、これらはオプションで削除できます。

詳細は、 About uninstalling OpenShift Container Platform Logging を参照してください。

2.1.7. フィールドのエクスポート

ロギングシステムはフィールドをエクスポートします。エクスポートされたフィールドはログレコードに存在し、Elasticsearch および Kibana から検索できます。

詳細は、About exporting fields を参照してください。

2.1.8. OpenShift Logging コンポーネントについて

OpenShift ロギングコンポーネントには、すべてのノードおよびコンテナーログを収集し、それらをログストアに書き込む OpenShift Container Platform クラスターの各ノードにデプロイされるコレクターが含まれます。一元化された Web UI を使用し、集計されたデータを使用して高度な可視化 (visualization) およびダッシュボードを作成できます。

クラスターロギングの主要コンポーネントは以下の通りです。

- collection: これは、クラスターからログを収集し、それらをフォーマットし、ログストアに転送するコンポーネントです。現在の実装は Fluentd です。
- log store: これはログが保存される場所です。デフォルトの実装は Elasticsearch です。デフォルトの Elasticsearch ログストアを使用するか、またはログを外部ログストアに転送することができます。デフォルトのログストアは、短期の保存について最適化され、テストされています。
- 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。

本書では、特筆されない限り、log store と Elasticsearch、visualization と Kibana、collection と Fluentd を区別せずに使用する場合があります。

2.1.9. ロギングコレクターについて

OpenShift Container Platform は Fluentd を使用してコンテナーおよびノードのログを収集します。

デフォルトでは、ログコレクターは以下のソースを使用します。

- すべてのシステムログ用の journald
- すべてのコンテナーログ用の **/var/log/containers/*.log**

監査ログを収集するようにログコレクターを設定すると、**/var/log/audit/audit.log** から取得されます。

ロギングコレクターは、Pod を各 OpenShift Container Platform ノードにデプロイするデーモンセットです。システムおよびインフラストラクチャーのログは、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 journald ログメッセージによって生成されます。アプリケーションログは CRI-O コンテナーエンジンによって生成されます。Fluentd はこれらのソースからログを収集し、OpenShift Container Platform で設定したように内部または外部に転送します。

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ない可能性があります。そのため、似たような名前の Pod やプロジェクトからログメッセージを区別したり、ログのソースを追跡することができない場合があります。この制限により、ログの収集および正規化は **ベストエフォート** ベースであると見なされます。

重要

利用可能なコンテナーランタイムは、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。

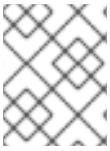
詳細は、ログコレクターの設定 について参照してください。

2.1.10. ログストアについて

デフォルトで、OpenShift Container Platform は Elasticsearch (ES) を使用してログデータを保存します。オプションで、ログ転送機能を使用して、Fluentd プロトコル、syslog プロトコル、または OpenShift Container Platform ログ転送 API を使用してログを外部ログストアに転送できます。

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

Elasticsearch は 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、それぞれのインデックスを **シャード** と呼ばれる複数の部分に分割します。これは、Elasticsearch クラスターの Elasticsearch ノードセット全体に分散されます。Elasticsearch を、**レプリカ** と呼ばれるシャードのコピーを作成するように設定できます。Elasticsearch はこれを Elasticsearch ノード全体に分散します。**ClusterLogging** カスタムリソース (CR) により、データの冗長性および耐障害性を確保するためにシャードを複製する方法を指定できます。また、**ClusterLogging** CR の保持ポリシーを使用して各種のログが保持される期間を指定することもできます。



注記

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

Red Hat OpenShift Logging Operator および OpenShift 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デプロイメントを使用してデプロイされるようにします。**ClusterLogging** カスタムリソース (CR) を使用して Elasticsearch ノードの数を適宜増やすことができます。ストレージの設定に関する考慮事項については、Elasticsearch ドキュメント を参照してください。

注記

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

Elasticsearch インデックスに適用されているロールベースアクセス制御 (RBAC) は、開発者のログの制御アクセスを可能にします。管理者はすべてのログに、開発者は各自のプロジェクトのログにのみアクセスできます。

詳細は、ログストアの設定 について参照してください。

2.1.11. ロギングの可視化について

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

Kibana は、ヒストグラム、線グラフ、円グラフその他の可視化機能を使用して Elasticsearch データをクエリーし、検出し、可視化するためのブラウザーベースのコンソールインターフェイスです。

詳細は、ログビジュアライザーの設定 について参照してください。

2.1.12. イベントのルーティングについて

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

イベントルーターは手動でデプロイする必要があります。

詳細は、 Kubernetes イベントの収集および保存 について参照してください。

2.1.13. ログ転送

デフォルトで、OpenShift Logging は **ClusterLogging** カスタムリソース (CR) に定義されるデフォルトの内部 Elasticsearch ログストアにログを送信します。ログを他のログアグリゲーターに転送する必要がある場合、ログ転送機能を使用してログをクラスター内外の特定のエンドポイントに送信することができます。

詳細は、ログのサードパーティーシステムへの転送 について参照してください。

第3章 OPENSHIFT LOGGING のインストール

OpenShift Logging は、OpenShift Elasticsearch および Red Hat OpenShift Logging Operator をデプロイしてインストールできます。OpenShift Elasticsearch Operator は、OpenShift Logging によって使用される Elasticsearch クラスターを作成し、管理します。Red Hat OpenShift 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。

OpenShift Logging 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。

- OpenShift Logging ストレージについての考慮事項 を確認します。
- OpenShift Container Platform Web コンソール、または CLI を使用した OpenShift Elasticsearch Operator および Red Hat OpenShift Logging Operator のインストール

3.1. WEB コンソールを使用した OPENSHIFT LOGGING のインストール

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

注記

デフォルトの Elasticsearch ログストアを使用しない場合、内部 Elasticsearch **logStore**、Kibana **visualization** コンポーネントを **ClusterLogging** カスタムリソース (CR) から削除することができます。これらのコンポーネントの削除はオプションですが、これによりリソースを節約できます。詳細は、デフォルトの Elasticsearch ログストアを使用しない場合の未使用のコンポーネントの削除 を参照してください。

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。

注記

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

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

手順

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

1. OpenShift Elasticsearch Operator をインストールします。
   a. OpenShift Container Platform Web コンソールで、**Operators → OperatorHub** をクリックします。

b. 利用可能な Operator の一覧から **OpenShift Elasticsearch Operator** を選択し、**Install** をクリックします。

c. **All namespaces on the cluster**が **Installation Mode** で選択されていることを確認します。

d. **openshift-operators-redhat** が **Installed Namespace** で選択されていることを確認します。
openshift-operators-redhat namespace を指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メ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。

e. **Enable operator recommended cluster monitoring on this namespace**を選択します。
このオプションは、namespace オブジェクトに **openshift.io/cluster-monitoring: "true"** ラベルを設定します。クラスターモニターリングが **openshift-operators-redhat** namespace を収集できるように、このオプションを選択する必要があります。

f. **Update Channel** として **stable-5.x** を選択します。

g. **Approval Strategy** を選択します。

- **Automatic** ストラテジーにより、Operator Lifecycle Manager (OLM) は新規バージョンが利用可能になると Operator を自動的に更新できます。
- **Manual** ストラテジーには、Operator の更新を承認するための適切な認証情報を持つユーザーが必要です。

h. **Install** をクリックします。

i. **Operators → Installed Operators** ページに切り替えて、OpenShift Elasticsearch Operator がインストールされていることを確認します。

j. **Status** が **Succeeded** の状態で、**OpenShift Elasticsearch Operator** が すべてのプロジェクトに一覧表示されていることを確認します。

2. Red Hat OpenShift Logging Operator をインストールします。

a. OpenShift Container Platform Web コンソールで、**Operators → OperatorHub** をクリックします。

b. 利用可能な Operator の一覧から **Red Hat OpenShift Logging** を選択し、**Install** をクリックします。

c. **A specific namespace on the cluster**が **Installation Mode** で選択されていることを確認します。

d. **Operator recommended namespace** が **Installed Namespace** で **openshift-logging** になっていることを確認します。

e. **Enable operator recommended cluster monitoring on this namespace**を選択します。
このオプションは、namespace オブジェクトに **openshift.io/cluster-monitoring: "true"** ラベルを設定します。クラスターモニターリングが **openshift-logging** namespace を収集できるように、このオプションを選択する必要があります。

f. **Update Channel** として **stable-5.x** を選択します。

g. **Approval Strategy** を選択します。

- **Automatic** ストラテジーにより、Operator Lifecycle Manager (OLM) は新規バージョンが利用可能になると Operator を自動的に更新できます。
- **Manual** ストラテジーには、Operator の更新を承認するための適切な認証情報を持つユーザーが必要です。

h. **Install** をクリックします。

i. **Operators → Installed Operators** ページに切り替えて、Red Hat OpenShift Logging Operator がインストールされていることを確認します。

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

- **Operators → Installed Operators** ページに切り替え、**Status** 列でエラーまたは失敗の有無を確認します。
- **Workloads → Pods** ページに切り替え、 **openshift-logging** 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。

3. OpenShift Logging インスタンスを作成します。

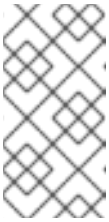
a. **Administration → Custom Resource Definitions**ページに切り替えます。

b. **Custom Resource Definitions**ページで、**ClusterLogging** をクリックします。

c. **Custom Resource Definition details**ページで、**Actions** メニューから **View Instances** を選択します。

d. **ClusterLoggings** ページで、 **Create ClusterLogging** 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 フィールドで、コードを以下に置き換えます。



注記

このデフォルトの OpenShift Logging 設定は各種の環境をサポートすることが予想されます。OpenShift Logging 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、OpenShift Logging 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
    retentionPolicy: 4
      application:
        maxAge: 1d
```

```
    infra:
      maxAge: 7d
    audit:
      maxAge: 7d
  elasticsearch:
    nodeCount: 3 5
    storage:
      storageClassName: "<storage_class_name>" 6
      size: 200G
    resources: 7
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
    proxy: 8
      resources:
        limits:
          memory: 256Mi
        requests:
          memory: 256Mi
    redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana" 9
  kibana:
    replicas: 1
collection:
  logs:
    type: "fluentd" 10
    fluentd: {}
```

1. 名前は **instance** である必要があります。
2. OpenShift Logging の管理状態。OpenShift Logging 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る場合があります。ただし、管理外のデプロイメントは OpenShift Logging が管理対象の状態に戻されるまで更新を受信しません。
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。
4. Elasticsearch が各ログソースを保持する期間を指定します。整数および時間の指定 (weeks(w)、hour(h/H)、minutes(m)、および seconds(s)) を入力します。たとえば、7 日の場合は **7d** となります。**maxAge** よりも古いログは削除されます。各ログソースの保持ポリシーを指定する必要があります。そうしない場合、Elasticsearch インデックスはそのソースに対して作成されません。
5. 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。
6. Elasticsearch ストレージの既存のストレージクラスの名前を入力します。最適なパフォーマンスを得るには、ブロックストレージを割り当てるストレージクラスを指定します。ストレージクラスを指定しない場合、OpenShift Logging は一時ストレージを使用します。
7. 必要に応じて CPU およびメモリー要求を指定します。これらの値を空のままにすると、OpenShift 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。デフォルト値

は、メモリー要求の場合は **16Gi** であり、CPU 要求の場合は **1** です。

8 必要に応じて Elasticsearch プロキシーの CPU およびメモリーの制限および要求を指定します。これらの値を空のままにすると、OpenShift 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。デフォルト値は、メモリー要求の場合は **256Mi**、CPU 要求の場合は **100m** です。

9 Kibana の設定に必要な設定。CR を使用して、冗長性を確保するために Kibana をスケーリングし、Kibana ノードの CPU およびメモリーを設定できます。詳細は、**ログビジュアライザーの設定** について参照してください。

10 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 コントロールプレーンノード (マスターノードとも呼ばれる) 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のノードとして作成されます。コントロールプレーンノードは、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
```

出力例

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

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

f. **Create** をクリックします。これにより、OpenShift Logging コンポーネント、**Elasticsearch** カスタムリソースおよびコンポーネント、および Kibana インターフェイスが作成されます。

4. 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 OpenShift Logging、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

関連情報

- OperatorHub からの Operator のインストール

3.2. インストール後のタスク

Kibana を使用する場合、Kibana のデータを確認し、び可視化するために、Kibana インデックスパターンおよびビジュアライゼーションを手動で作成する 必要があります。

クラスターネットワークプロバイダーがネットワークの分離を実施している場合、OpenShift Logging Operator が含まれるプロジェクト間のネットワークトラフィックを許可します。

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

OpenShift Container Platform CLI を使って OpenShift Elasticsearch および Red Hat OpenShift Logging Operator をインストールすることができます。

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。

注記

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

Elasticsearch はメモリー集約型アプリケーションです。デフォルトで、OpenShift Container Platform はメモリー要求および 16 GB の制限を持つ 3 つの Elasticsearch ノードをインストー

ルします。OpenShift Container Platform ノードの最初の 3 つのセットには、Elasticsearch をクラスター内で実行するのに十分なメモリーがない可能性があります。Elasticsearch に関連するメモリーの問題が発生した場合、既存ノードのメモリーを増やすのではなく、Elasticsearch ノードをクラスターにさらに追加します。

手順

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

1. OpenShift Elasticsearch Operator の namespace を作成します。

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

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

   (1) **openshift-operators-redhat** namespace を指定する必要があります。メトリクスとの競合が発生する可能性を防ぐには、Prometheus のクラスターモニターリングスタックを、**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メトリクスを公開する可能性があるため、これによって競合が生じる可能性があります。

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

   b. namespace を作成します。

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

   以下に例を示します。

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

2. Red Hat OpenShift Logging Operator の namespace を作成します。

   a. Red Hat OpenShift Logging Operator の namespace オブジェクト YAML ファイル (**olo-namespace.yaml** など) を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging
  annotations:
```

```
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true"
```

b. namespace を作成します。

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

以下に例を示します。

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

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

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

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

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

b. Operator グループオブジェクトを作成します。

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

以下に例を示します。

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

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

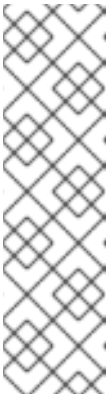
Subscription の例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat" (1)
spec:
  channel: "stable-5.1" (2)
  installPlanApproval: "Automatic"
  source: "redhat-operators" (3)
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

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

2 **5.0**、**stable** または **stable-5.<x>** をチャネルとして指定します。以下の注意点を参照してください。

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



注記

stable を指定すると、最新の安定したリリースの現行バージョンがインストールされます。**installPlanApproval: "Automatic"** で **stable** 使用すると、Operatar が自動的に最新の安定したメジャーおよびマイナーリリースにアップグレードします。

stable-5.<x> を指定すると、特定のメジャーリリースの現在のマイナーバージョンがインストールされます。**installPlanApproval: "Automatic"** で **stable-5.<x>** を使用すると、**x** で指定したメジャーリリース内で最新の安定マイナーリリースに Operator が自動的にアップグレードされます。

d. Subscription オブジェクトを作成します。

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

以下に例を示します。

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

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

e. Operator のインストールを確認します。

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

出力例

```
NAMESPACE                                          NAME                                         DISPLAY
VERSION            REPLACES   PHASE
default                                            elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0              Succeeded
kube-node-lease                                    elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0              Succeeded
kube-public                                        elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0              Succeeded
kube-system                                        elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0              Succeeded
openshift-apiserver-operator                       elasticsearch-operator.5.1.0-
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
openshift-apiserver                                elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0              Succeeded
openshift-authentication-operator                  elasticsearch-operator.5.1.0-
```

```
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
openshift-authentication                              elasticsearch-operator.5.1.0-
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
...
```

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

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

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

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

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

   b. OperatorGroup オブジェクトを作成します。

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

   以下に例を示します。

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

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

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging (1)
spec:
  channel: "stable" (2)
  name: cluster-logging
  source: redhat-operators (3)
  sourceNamespace: openshift-marketplace
```

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

   (2) **5.0**、**stable** または **stable-5.<x>** をチャネルとして指定します。

   (3) **redhat-operators** を指定します。OpenShift Container Platform クラスターが、非接続クラスターとも呼ばれる制限されたネットワークにインストールされている場合、

Operator Lifecycle Manager (OLM) の設定時に作成した CatalogSource オブジェクトの名前を指定します。

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

以下に例を示します。

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

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

d. Operator のインストールを確認します。
openshift-logging namespace には Red Hat OpenShift Logging Operator がなければなりません。バージョン番号が表示されるものと異なる場合があります。

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

出力例

```
NAMESPACE                                       NAME                                         DISPLAY
VERSION            REPLACES   PHASE
...
openshift-logging                               clusterlogging.5.1.0-202007012112.p0
OpenShift Logging        5.1.0-202007012112.p0            Succeeded
...
```

5. OpenShift Logging インスタンスを作成します。

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

注記

このデフォルトの OpenShift Logging 設定は各種の環境をサポートすることが予想されます。OpenShift Logging 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、OpenShift Logging 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
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
```

```
      maxAge: 7d
    audit:
      maxAge: 7d
  elasticsearch:
    nodeCount: 3 5
    storage:
      storageClassName: "<storage-class-name>" 6
      size: 200G
    resources: 7
      limits:
        memory: "16Gi"
      requests:
        memory: "16Gi"
    proxy: 8
      resources:
        limits:
          memory: 256Mi
        requests:
          memory: 256Mi
    redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana" 9
  kibana:
    replicas: 1
collection:
  logs:
    type: "fluentd" 10
    fluentd: {}
```

1. 名前は **instance** である必要があります。
2. OpenShift Logging の管理状態。OpenShift Logging 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る場合があります。ただし、管理外のデプロイメントは OpenShift Logging が管理対象の状態に戻されるまで更新を受信しません。デプロイメントを管理対象の状態に戻すと、加えた変更が元に戻される可能性があります。
3. Elasticsearch の設定に必要な設定。カスタムリソース (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。
4. Elasticsearch が各ログソースを保持する期間を指定します。整数および時間の指定 (weeks(w)、hour(h/H)、minutes(m)、および seconds(s)) を入力します。たとえば、7 日の場合は **7d** となります。**maxAge** よりも古いログは削除されます。各ログソースの保持ポリシーを指定する必要があります。そうしない場合、Elasticsearch インデックスはそのソースに対して作成されません。
5. 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。
6. Elasticsearch ストレージの既存のストレージクラスの名前を入力します。最適なパフォーマンスを得るには、ブロックストレージを割り当てるストレージクラスを指定します。ストレージクラスを指定しない場合、OpenShift Container Platform は一時ストレージのみの OpenShift Logging をデプロイします。
7. 必要に応じて CPU およびメモリー要求を指定します。これらの値を空のままにすると、OpenShift 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ます。デフォルト値は、

メモリー要求の場合は **16Gi** であり、CPU 要求の場合は **1** です。

8 必要に応じて Elasticsearch プロキシーの CPU およびメモリーの制限および要求を指定します。これらの値を空のままにすると、OpenShift 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。デフォルト値は、メモリー要求の場合は **256Mi**、CPU 要求の場合は **100m** です。

9 Kibana の設定に必要な設定。CR を使用して、冗長性を確保するために Kibana をスケーリングし、Kibana Pod の CPU およびメモリーを設定できます。詳細は、**ログビジュアライザーの設定** について参照してください。

10 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 コントロールプレーンノードの最大数は 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のノードとして作成されます。コントロールプレーンノードは、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
```

出力例

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

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

b. インスタンスを作成します。

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

以下は例になります。

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

これにより、OpenShift Logging コンポーネント、**Elasticsearch** カスタムリソースおよびコンポーネント、および Kibana インターフェイスが作成されます。

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

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

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE
cluster-logging-operator-66f77ffccb-ppzbg   1/1     Running   0          7m
elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp   2/2     Running   0          2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc   2/2     Running   0          2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2   2/2     Running   0          2m4s
fluentd-587vb                               1/1     Running   0          2m26s
fluentd-7mpb9                               1/1     Running   0          2m30s
fluentd-flm6j                               1/1     Running   0          2m33s
fluentd-gn4rn                               1/1     Running   0          2m26s
fluentd-nlgb6                               1/1     Running   0          2m30s
fluentd-snpkt                               1/1     Running   0          2m28s
kibana-d6d5668c5-rppqm                      2/2     Running   0          2m39s
```

3.4. インストール後のタスク

Kibana を使用する場合、Kibana のデータを確認し、び可視化するために、Kibana インデックスパターンおよびビジュアライゼーションを手動で作成する 必要があります。

クラスターネットワークプロバイダーがネットワークの分離を実施している場合、OpenShift Logging Operator が含まれるプロジェクト間のネットワークトラフィックを許可します。

3.4.1. Kibana インデックスパターンの定義

インデックスパターンは、可視化する必要のある Elasticsearch インデックスを定義します。Kibana でデータを確認し、可視化するには、インデックスパターンを作成する必要があります。

前提条件

- Kibana で **infra** および **audit** インデックスを表示するには、ユーザーには **cluster-admin** ロール、 **cluster-reader** ロール、または両方のロールが必要です。デフォルトの **kubeadmin** ユーザーには、これらのインデックスを表示するための適切なパーミッションがあります。
default、**kube-** および **openshift-** プロジェクトで Pod およびログを表示できる場合、これらのインデックスにアクセスできるはずです。以下のコマンドを使用して、現在のユーザーが適切なパーミッションを持っているかどうかを確認することができます。

```
$ oc auth can-i get pods/log -n <project>
```

出力例

```
yes
```

注記

監査ログは、デフォルトでは内部 OpenShift Container Platform Elasticsearch インスタンスに保存されません。Kibana で監査ログを表示するには、ログ転送 API を使用して監査ログの **default** 出力を使用するパイプラインを設定する必要があります。

- Elasticsearch ドキュメントは、インデックスパターンを作成する前にインデックス化する必要があります。これは自動的に実行されますが、新規または更新されたクラスターでは数分の時間がかかる可能性があります。

手順

Kibana でインデックスパターンを定義し、ビジュアライゼーションを作成するには、以下を実行します。

1. OpenShift Container Platform コンソールで、Application Launcher をクリックし、**Logging** を選択します。
2. **Management → Index Patterns → Create index pattern** をクリックして Kibana インデックスパターンを作成します。
   - 各ユーザーは、プロジェクトのログを確認するために、Kibana に初めてログインする際にインデックスパターンを手動で作成する必要があります。ユーザーは **app** という名前のインデックスパターンを作成し、**@timestamp** 時間フィールドを使用してコンテナーログを表示する必要があります。
   - 管理ユーザーはそれぞれ、最初に Kibana にログインする際に、**@timestamp** 時間フィールドを使用して **app**、 **infra** および **audit** インデックスについてインデックスパターンを作成する必要があります。
3. 新規インデックスパターンから Kibana のビジュアライゼーション (Visualization) を作成します。

3.4.2. ネットワークの分離が有効にされている場合のプロジェクト間のトラフィックの許可

クラスターネットワークプロバイダーはネットワークの分離を有効にする可能性があります。その場合、OpenShift Logging によってデプロイされる Operator が含まれるプロジェクト間のネットワークトラフィックを許可する必要があります。

ネットワークの分離は、異なるプロジェクトにある Pod およびサービス間のネットワークトラフィックをブロックします。OpenShift Logging は、**OpenShift Elasticsearch Operator** を **openshift-operators-redhat** プロジェクトにインストールし、**Red Hat OpenShift Logging Operator** を **openshift-logging** プロジェクトにインストールします。したがって、これら 2 つのプロジェクト間のトラフィックを許可する必要があります。

OpenShift Container Platform は、2 つのサポート対象のオプションをデフォルトの Container Network Interface (CNI) ネットワークプロバイダー、OpenShift SDN および OVN-Kubernetes 用に提供します。これら 2 つのプロバイダーはさまざまなネットワーク分離ポリシーを実装します。

OpenShift SDN には 3 つのモードがあります。

network policy (ネットワークポリシー)

これはデフォルトモードになります。ポリシーが定義されていない場合は、すべてのトラフィックを許可します。ただし、ユーザーがポリシーを定義する場合、通常はすべてのトラフィックを拒否

し、例外を追加して開始します。このプロセスでは、異なるプロジェクトで実行されているアプリケーションが破損する可能性があります。そのため、ポリシーを明示的に設定し、1 つのロギング関連のプロジェクトから他のプロジェクトへの egress のトラフィックを許可します。

multitenant (マルチテナント)

このモードは、ネットワークの分離を実行します。2 つのロギング関連のプロジェクトを結合して、それらのプロジェクト間のトラフィックを許可します。

subnet (サブネット)

このモードでは、すべてのトラフィックを許可します。ネットワーク分離は実行しません。アクションは不要です。

OVN-Kubernetes は常に **ネットワークポリシー** を使用します。そのため、OpenShift SDN の場合と同様に、ポリシーを明示的に設定し、1 つのロギング関連のプロジェクトから他のプロジェクトへの egress のトラフィックを許可する必要があります。

手順

- **multitenant** モードで OpenShift SDN を使用している場合は、2 つのプロジェクトに参加します。以下に例を示します。

```
$ oc adm pod-network join-projects --to=openshift-operators-redhat openshift-logging
```

- または、**network policy** の OpenShift SDN および OVN-Kubernetes の場合は、以下の操作を実行します。

  a. **openshift-operators-redhat** namespace にラベルを設定します。以下に例を示します。

```
$ oc label namespace openshift-operators-redhat project=openshift-operators-redhat
```

  b. **openshift-operators-redhat**、**openshift-monitoring**、および**openshift-ingress**プロジェクトから openshift-logging プロジェクトへの入力を許可する、**openshift-logging** namespace にネットワークポリシーオブジェクトを作成します。以下に例を示します。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-monitoring-ingress-operators-redhat
spec:
  ingress:
  - from:
    - podSelector: {}
  - from:
    - namespaceSelector:
        matchLabels:
          project: "openshift-operators-redhat"
  - from:
    - namespaceSelector:
        matchLabels:
          name: "openshift-monitoring"
  - from:
    - namespaceSelector:
        matchLabels:
          network.openshift.io/policy-group: ingress
```

```
  podSelector: {}
  policyTypes:
  - Ingress
```

関連情報

- ネットワークポリシーについて
- OpenShift SDN デフォルト CNI ネットワークプロバイダーについて
- OVN-Kubernetes デフォルト Container Network Interface (CNI) ネットワークプロバイダーについて

第4章 ロギングデプロイメントの設定

4.1. クラスターロギングカスタムリソースについて

OpenShift Logging を設定するには、**ClusterLogging** カスタムリソース (CR) をカスタマイズします。

4.1.1. ClusterLogging カスタムリソースについて

OpenShift Logging 環境を変更するには、**ClusterLogging** 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、OpenShift Logging の通常のカスタムリソースの例です。

ClusterLogging カスタムリソース (CRD) のサンプル

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging" 2
spec:
  managementState: "Managed" 3
  logStore:
    type: "elasticsearch" 4
    retentionPolicy:
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization: 5
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 736Mi
        requests:
          cpu: 100m
          memory: 736Mi
```

```
    replicas: 1
  collection: 6
    logs:
      type: "fluentd"
      fluentd:
        resources:
          limits:
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi
```

1 CR の名前は **instance** である必要があります。

2 CR は **openshift-logging** namespace にインストールされる必要があります。

3 Red Hat OpenShift Logging Operator の管理状態。**Unmanaged** に設定すると、Operator はサポート対象外となり、更新を取得しません。

4 保持ポリシー、ノード数、リソース要求および制限およびストレージクラスなどのログストアの設定。

5 リソース要求および制限、Pod レプリカ数などのビジュアライザーの設定。

6 リソース要求および制限を含むログコレクターの設定。

4.2. ロギングコレクターの設定

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

ログコレクターの CPU およびメモリー制限を設定し、ログコレクター Pod を特定のノードに移動 できます。ログコレクターに対するサポートされるすべての変更は、**ClusterLogging** カスタムリソース (CR) の **spec.collection.log.fluentd** スタンザを使用して実行できます。

4.2.1. サポートされる設定

OpenShift Logging の設定のサポートされる方法として、本書で説明されているオプションを使用してこれを設定することができます。サポートされていない他の設定は使用しないでください。設定のパラダイムが OpenShift Container Platform リリース間で変更される可能性があり、このような変更は、設定のすべての可能性が制御されている場合のみ適切に対応できます。本書で説明されている設定以外の設定を使用する場合、OpenShift Elasticsearch Operator および Red Hat OpenShift Logging Operator が差分を調整するため、変更内容は失われます。Operator はデフォルトで定義された状態にすべて戻します。

注記

OpenShift Container Platform ドキュメントで説明されていない設定を実行する **必要がある** 場合、Red Hat OpenShift Logging Operator または OpenShift Elasticsearch Operator を **Unmanaged** (管理外) に設定する **必要があります**。管理外の OpenShift Logging 環境は **サポート外** であり、OpenShift Logging を **Managed** に戻すまで変更を受信しません。

4.2.2. ロギングコレクター Pod の表示

Fluentd ロギングコレクター Pod およびそれらが実行されている対応するノードを表示できます。Fluentd ロギングコレクター Pod は **openshift-logging** プロジェクトでのみ実行されます。

手順

- **openshift-logging** プロジェクトで以下のコマンドを実行し、Fluentd ロギングコレクター Pod とそれらの詳細を表示します。

```
$ oc get pods --selector component=fluentd -o wide -n openshift-logging
```

出力例

```
NAME          READY  STATUS    RESTARTS   AGE     IP           NODE              NOMINATED
NODE   READINESS GATES
fluentd-8d69v  1/1    Running   0          134m    10.130.2.30   master1.example.com   <none>
<none>
fluentd-bd225  1/1    Running   0          134m    10.131.1.11   master2.example.com   <none>
<none>
fluentd-cvrzs  1/1    Running   0          134m    10.130.0.21   master3.example.com   <none>
<none>
fluentd-gpqg2  1/1    Running   0          134m    10.128.2.27   worker1.example.com   <none>
<none>
fluentd-l9j7j  1/1    Running   0          134m    10.129.2.31   worker2.example.com   <none>
<none>
```

4.2.3. ログコレクター CPU およびメモリー制限の設定

ログコレクターは、CPU とメモリー制限の両方への調整を許可します。

手順

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

```
$ oc -n openshift-logging edit ClusterLogging instance
```

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

...

spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: 1
            memory: 736Mi
```

```
      requests:
        cpu: 100m
        memory: 736Mi
```

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

4.2.4. ログフォワーダーの高度な設定

OpenShift Logging には、Fluentd ログフォワーダーのパフォーマンスチューニングに使用できる複数の Fluentd パラメーターが含まれます。これらのパラメーターを使用すると、以下の Fluentd の動作を変更できます。

- Fluentd チャンクおよびチャンクバッファーのサイズ
- Fluentd チャンクのフラッシュ動作
- Fluentd チャンクの転送の再試行動作

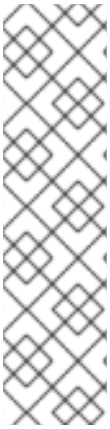
Fluentd は、**チャンク** という単一の Blob でログデータを収集します。Fluentd がチャンクを作成する際に、チャンクは **ステージ** にあると見なされます。ここでチャンクはデータで一杯になります。チャンクが一杯になると、Fluentd はチャンクを **キュー** に移動します。ここでチャンクはフラッシュされる前か、または送信先に書き込まれるまで保持されます。Fluentd は、ネットワークの問題や送信先での容量の問題などのさまざまな理由でチャンクをフラッシュできない場合があります。チャンクをフラッシュできない場合、Fluentd は設定通りにフラッシュを再試行します。

OpenShift Container Platform のデフォルトで、Fluentd は **指数関数的バックオフ** 方法を使用してフラッシュを再試行します。この場合、Fluentd はフラッシュを再試行するまで待機する時間を 2 倍にします。これは、送信先への接続要求を減らすのに役立ちます。指数関数的バックオフを無効にし、代わりに **定期的な** 再試行方法を使用できます。これは、指定の間隔でチャンクのフラッシュを再試行します。デフォルトで、Fluentd はチャンクのフラッシュを無限に再試行します。OpenShift Container Platform では、無限の再試行動作を変更することはできません。

これらのパラメーターは、待ち時間とスループット間のトレードオフを判断するのに役立ちます。

- Fluentd のスループットを最適化するには、これらのパラメーターを使用して、より大きなバッファーおよびキューを設定し、フラッシュを遅延し、再試行の間隔の長く設定することで、ネットワークパケット数を減らすことができます。より大きなバッファーにはノードのファイルシステムでより多くの領域が必要になることに注意してください。
- 待機時間が低い場合に最適化するには、パラメーターを使用してすぐにデータを送信し、バッチの蓄積を回避し、キューとバッファーが短くして、より頻繁にフラッシュおよび再試行を使用できます。

ClusterLogging カスタムリソース (CR) で以下のパラメーターを使用して、チャンクおよびフラッシュ動作を設定できます。次に、パラメーターは Fluentd で使用するために Fluentd 設定マップに自動的に追加されます。



注記

これらのパラメーターの特徴は以下の通りです。

- ほとんどのユーザーには関連性がありません。デフォルト設定で、全般的に良いパフォーマンスが得られるはずです。
- Fluentd 設定およびパフォーマンスに関する詳しい知識を持つ上級ユーザーのみが対象です。
- パフォーマンスチューニングのみを目的とします。ロギングの機能面に影響を与えることはありません。

表4.1 高度な Fluentd 設定パラメーター

パラメーター	説明	デフォルト
chunkLimitSize	各チャンクの最大サイズ。Fluentd はこのサイズに達するとデータのチャンクへの書き込みを停止します。次に、Fluentd はチャンクをキューに送信し、新規のチャンクを開きます。	**8m**
totalLimitSize	ステージおよびキューの合計サイズであるバッファーの最大サイズ。バッファーサイズがこの値を超えると、Fluentd はデータのチャンクへの追加を停止し、エラーを出して失敗します。チャンクにないデータはすべて失われます。	**8G**
flushInterval	チャンクのフラッシュの間隔。**s** (秒)、**m** (分)、 **h** (時間)、または **d** (日) を使用できます。	**1s**
flushMode	フラッシュを実行する方法: ● **lazy**: **timekey** パラメーターに基づいてチャンクをフラッシュします。**timekey** パラメーターを変更することはできません。 ● **interval**: **flushInterval** パラメーターに基づいてチャンクをフラッシュします。 ● **immediate**: データをチャンクに追加後すぐにチャンクをフラッシュします。	**interval**

パラメーター	説明	デフォルト
flushThreadCount	チャンクのフラッシュを実行するスレッドの数。スレッドの数を増やすと、フラッシュのスループットが改善し、ネットワークの待機時間が非表示になります。	**2**
overflowAction	キューが一杯になると、チャンク動作は以下のようになります。 ● **throw_exception**: ログに表示される例外を発生させます。 ● **block**: 詳細のバッファーの問題が解決されるまでデータのチャンクを停止します。 ● **drop_oldest_chunk**: 新たな受信チャンクを受け入れるために最も古いチャンクをドロップします。古いチャンクの値は新しいチャンクよりも小さくなります。	**block**
retryMaxInterval	**exponential_backoff** 再試行方法の最大時間 (秒単位)。	**300s**
retryType	フラッシュに失敗する場合の再試行方法: ● **exponential_backoff**: フラッシュの再試行の間隔を増やします。Fluentd は、**retry_max_interval** パラメーターに達するまで、次の試行までに待機する時間を 2 倍にします。 ● **periodic**: **retryWait** パラメーターに基づいてフラッシュを定期的に再試行します。	**exponential_backoff**
retryWait	次のチャンクのフラッシュまでの時間 (秒単位)。	**1s**

Fluentd チャンクのライフサイクルの詳細は、Fluentd ドキュメントの Buffer Plugins を参照してください。

手順

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

```
$ oc edit ClusterLogging instance
```

2. 以下のパラメーターを追加または変更します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  forwarder:
    fluentd:
      buffer:
        chunkLimitSize: 8m (1)
        flushInterval: 5s (2)
        flushMode: interval (3)
        flushThreadCount: 3 (4)
        overflowAction: throw_exception (5)
        retryMaxInterval: "300s" (6)
        retryType: periodic (7)
        retryWait: 1s (8)
        totalLimitSize: 32m (9)
...
```

(1) 各チャンクの最大サイズを指定してから、フラッシュ用にキューに入れます。

(2) チャンクのフラッシュの間隔を指定します。

(3) チャンクのフラッシュを実行する方法を指定します (**lazy**、**interval**、または **immediate**)。

(4) チャンクのフラッシュに使用するスレッドの数を指定します。

(5) キューが一杯になる場合のチャンクの動作を指定します (**throw_exception**、**block**、または **drop_oldest_chunk**)。

(6) **exponential_backoff** チャンクのフラッシュ方法について最大の間隔 (秒単位) を指定します。

(7) チャンクのフラッシュが失敗する場合の再試行タイプ (**exponential_backoff** または **periodic**) を指定します。

(8) 次のチャンクのフラッシュまでの時間 (秒単位) を指定します。

(9) チャンクバッファーの最大サイズを指定します。

3. Flunentd Pod が再デプロイされていることを確認します。

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

4. 新規の値が **fluentd** 設定マップにあることを確認します。

```
$ oc extract configmap/fluentd --confirm
```

fluentd.conf の例

```
<buffer>
 @type file
 path '/var/lib/fluentd/default'
 flush_mode interval
 flush_interval 5s
 flush_thread_count 3
 retry_type periodic
 retry_wait 1s
 retry_max_interval 300s
 retry_timeout 60m
 queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '32'}"
 total_limit_size 32m
 chunk_limit_size 8m
 overflow_action throw_exception
</buffer>
```

4.2.5. デフォルトの Elasticsearch ログストアを使用しない場合の未使用のコンポーネントの削除

管理者がログをサードパーティーのログストアに転送し、デフォルトの Elasticsearch ログストアを使用しない場合には、ロギングクラスターからいくつかの未使用のコンポーネントを削除できます。

つまり、デフォルトの Elasticsearch ログストアを使用しない場合、内部 Elasticsearch **logStore**、Kibana **visualization** コンポーネントを **ClusterLogging** カスタムリソース (CR) から削除することができます。これらのコンポーネントの削除はオプションですが、これによりリソースを節約できます。

前提条件

- ログフォワーダーがログデータをデフォルトの内部 Elasticsearch クラスターに送信しないことを確認します。ログ転送の設定に使用した **ClusterLogForwarder** CR YAML ファイルを検査します。これには **default** を指定する **outputRefs** 要素が **ない** ことを確認します。以下に例を示します。

```
outputRefs:
- default
```

警告

ClusterLogForwarder CR がログデータを内部 Elasticsearch クラスターに転送し、**ClusterLogging** CR から **logStore** コンポーネントを削除するとします。この場合、内部 Elasticsearch クラスターはログデータを保存するために表示されません。これがないと、データが失われる可能性があります。

手順

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

```
$ oc edit ClusterLogging instance
```

2. これらが存在する場合、**logStore**、**visualization** スタンザを **ClusterLogging** CR から削除します。

3. **ClusterLogging** CR の **collection** スタンザを保持します。結果は以下の例のようになります。

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

4. Flunentd Pod が再デプロイされていることを確認します。

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

関連情報

- ログのサードパーティーシステムへの転送

4.3. ログストアの設定

OpenShift Container Platform は Elasticsearch 6 (ES) を使用してログデータを保存し、整理します。

ログストアに加えることのできる変更には、以下が含まれます。

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

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** カスタムリソースで指定しない限り、メモリー要求および制限の両方に 16G 以上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 (各 Elasticsearch ノードに最大 64G) 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、これは実稼働環境には推奨されません。

4.3.1. 監査ログのログストアへの転送

内部 OpenShift Container Platform Elasticsearch ログストアは監査ログのセキュアなストレージを提供しないため、デフォルトで監査ログは内部 Elasticsearch インスタンスに保存されません。

監査ログを内部ログストアに送信する必要がある場合 (Kibana で監査ログを表示するなど)、ログ転送 API を使用する必要があります。

重要

内部 OpenShift Container Platform Elasticsearch ログストアは、監査ログのセキュアなストレージを提供しません。監査ログを転送するシステムが組織および政府の規制に準拠しており、適切にセキュリティーが保護されていることを確認することが推奨されています。OpenShift Logging はこれらの規制に準拠しません。

手順

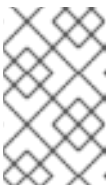
ログ転送 API を使用して監査ログを内部 Elasticsearch インスタンスに転送するには、以下を実行します。

1. **ClusterLogForwarder** CR YAML ファイルを作成するか、または既存の CR を編集します。

   - すべてのログタイプを内部 Elasticsearch インスタンスに送信するために CR を作成します。変更せずに以下の例を使用できます。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines: 1
  - name: all-to-default
    inputRefs:
    - infrastructure
    - application
    - audit
    outputRefs:
    - default
```

1 パイプラインは、指定された出力を使用して転送するログのタイプを定義します。デフォルトの出力は、ログを内部 Elasticsearch インスタンスに転送します。



注記

パイプラインの 3 つのすべてのタイプのログをパイプラインに指定する必要があります (アプリケーション、インフラストラクチャー、および監査)。ログの種類を指定しない場合、それらのログは保存されず、失われます。

   - 既存の **ClusterLogForwarder** CR がある場合、パイプラインを監査ログのデフォルト出力に追加します。デフォルトの出力を定義する必要はありません。以下に例を示します。

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

```
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
   - name: elasticsearch-insecure
     type: "elasticsearch"
     url: http://elasticsearch-insecure.messaging.svc.cluster.local
     insecure: true
   - name: elasticsearch-secure
     type: "elasticsearch"
     url: https://elasticsearch-secure.messaging.svc.cluster.local
     secret:
       name: es-audit
   - name: secureforward-offcluster
     type: "fluentdForward"
     url: https://secureforward.offcluster.com:24224
     secret:
       name: secureforward
  pipelines:
   - name: container-logs
     inputRefs:
     - application
     outputRefs:
     - secureforward-offcluster
   - name: infra-logs
     inputRefs:
     - infrastructure
     outputRefs:
     - elasticsearch-insecure
   - name: audit-logs
     inputRefs:
     - audit
     outputRefs:
     - elasticsearch-secure
     - default (1)
```

(1) このパイプラインは、外部インスタンスに加えて監査ログを内部 Elasticsearch インスタンスに送信します。

関連情報

- ログ転送 API の詳細は、Forwarding logs using the Log Forwarding API を参照してください。

4.3.2. ログ保持時間の設定

デフォルトの Elasticsearch ログストアがインフラストラクチャーログ、アプリケーションログ、監査ログなどの 3 つのログソースのインデックスを保持する期間を指定する **保持ポリシー** を設定できます。

保持ポリシーを設定するには、**ClusterLogging** カスタムリソース (CR) に各ログソースの **maxAge** パラメーターを設定します。CR はこれらの値を Elasticsearch ロールオーバースケジュールに適用し、Elasticsearch がロールオーバーインデックスを削除するタイミングを決定します。

Elasticsearch はインデックスをロールオーバーし、インデックスが以下の条件のいずれかに一致する場合に現在のインデックスを移動し、新規インデックスを作成します。

- インデックスは **Elasticsearch** CR の **rollover.maxAge** の値よりも古い値になります。
- インデックスサイズは、40 GB x プライマリーシャードの数よりも大きくなります。
- インデックスの doc 数は、40960 KB × プライマリーシャードの数よりも大きくなります。

Elasticsearch は、設定する保持ポリシーに基づいてロールオーバーインデックスを削除します。ログソースの保持ポリシーを作成しない場合、ログはデフォルトで 7 日後に削除されます。

前提条件

- OpenShift Logging および OpenShift Elasticsearch Operator がインストールされている。

手順

ログの保持時間を設定するには、以下を実行します。

1. **ClusterLogging** CR を編集して、**retentionPolicy** パラメーターを追加するか、または変更します。

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    retentionPolicy: 1
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
...
```

1 Elasticsearch が各ログソースを保持する時間を指定します。整数および時間の指定 (weeks(w)、hour(h/H)、minutes(m)、および seconds(s)) を入力します。たとえば、1 日の場合は **1d** になります。**maxAge** よりも古いログは削除されます。デフォルトで、ログは 7 日間保持されます。

2. **Elasticsearch** カスタムリソース (CR) で設定を確認できます。
たとえば、Red Hat OpenShift Logging Operator は以下の **Elasticsearch** CR を更新し、8 時間ごとにインフラストラクチャーログのアクティブなインデックスをロールオーバーし、ロールオーバーされたインデックスはロールオーバーの 7 日後に削除される設定を含む保持ポリシーを設定するとします。OpenShift Container Platform は 15 分ごとにチェックし、インデックスをロールオーバーする必要があるかどうかを判別します。

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

```
metadata:
  name: "elasticsearch"
spec:
...
  indexManagement:
    policies: 1
      - name: infra-policy
        phases:
          delete:
            minAge: 7d 2
          hot:
            actions:
              rollover:
                maxAge: 8h 3
        pollInterval: 15m 4
...
```

1 各ログソースについて、保持ポリシーは、そのソースのログを削除/ロールオーバーするタイミングを示します。

2 OpenShift Container Platform がロールオーバーされたインデックスを削除する場合。この設定は、**ClusterLogging** CR に設定する **maxAge** になります。

3 インデックスをロールオーバーする際に考慮する OpenShift Container Platform のインデックス期間。この値は、**ClusterLogging** CR に設定する **maxAge** に基づいて決定されます。

4 OpenShift Container Platform がインデックスをロールオーバーする必要があるかどうかをチェックする場合。この設定はデフォルトであるため、変更できません。

注記

Elasticsearch CR の変更はサポートされていません。保持ポリシーに対するすべての変更は **ClusterLogging** CR で行う必要があります。

OpenShift Elasticsearch Operator は cron ジョブをデプロイし、**pollInterval** を使用してスケジュールされる定義されたポリシーを使用して各マッピングのインデックスをロールオーバーします。

```
$ oc get cronjob
```

出力例

```
NAME                  SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          4s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          4s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          4s
```

4.3.3. ログストアの CPU およびメモリー要求の設定

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

注記

大規模なクラスターでは、Elasticsearch プロキシーコンテナーのデフォルトのメモリー制限が不十分である場合があり、これにより、プロキシーコンテナーが OOM による強制終了 (OOMKilled) が生じます。この問題が発生した場合には、Elasticsearch プロキシーのメモリー要求および制限を引き上げます。

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 **されていません**。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

- OpenShift Logging および Elasticsearch がインストールされている。

手順

1. **openshift-logging** プロジェクトで **ClusterLogging** 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: 1
        resources:
          limits: 2
            memory: "32Gi"
          requests: 3
            cpu: "1"
            memory: "16Gi"
        proxy: 4
          resources:
            limits:
              memory: 100Mi
            requests:
              memory: 100Mi
```

1. 必要に応じて CPU およびメモリー要求を指定します。これらの値を空のままにすると、OpenShift 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。デフォルト値は、メモリー要求の場合は **16Gi** であり、CPU 要求の場合は **1** です。

2. Pod が使用できるリソースの最大量。

3 Pod のスケジュールに必要最小限のリソース。

4 必要に応じて Elasticsearch プロキシーの CPU およびメモリーの制限および要求を指定します。これらの値を空のままにすると、OpenShift 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ます。デフォルト値は、メモリー要求の場合は **256Mi**、CPU 要求の場合は **100m** です。

Elasticsearch メモリーの量を調整するときは、**要求** と **制限** の両方に同じ値を使用する必要があります。

以下に例を示します。

```
resources:
  limits: 1
    memory: "32Gi"
  requests: 2
    cpu: "8"
    memory: "32Gi"
```

1 リソースの最大量。

2 必要最小限の量。

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

4.3.4. ログストアのレプリケーションポリシーの設定

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

1. **openshift-logging** プロジェクトで **ClusterLogging** 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。

注記

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

4.3.5. Elasticsearch Pod のスケールダウン

クラスター内の Elasticsearch Pod 数を減らすと、データ損失や Elasticsearch のパフォーマンスが低下する可能性があります。

スケールダウンする場合、一度に 1 つの Pod 分スケールダウンし、クラスターがシャードおよびレプリカのリバランスを実行できるようにする必要があります。Elasticsearch のヘルスステータスが **green** に戻された後に、別の Pod でスケールダウンできます。

注記

Elasticsearch クラスターが **ZeroRedundancy** に設定される場合、Elasticsearch Pod をスケールダウンしないでください。

4.3.6. ログストアの永続ストレージの設定

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

警告

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

1. **ClusterLogging** CR を編集してクラスターの各データノードが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。

注記

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

4.3.7. emptyDir ストレージのログストアの設定

ログストアで 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、ログストアが再起動するか、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

- OpenShift Logging および Elasticsearch がインストールされている。

手順

1. **ClusterLogging** CR を編集して emptyDir を指定します。

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

4.3.8. Elasticsearch クラスターのローリング再起動の実行

elasticsearch 設定マップまたは **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

- OpenShift Logging および Elasticsearch がインストールされている。

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. Elasticsearch Pod の名前を取得します。

```
$ oc get pods | grep elasticsearch-
```

3. Fluentd Pod をスケールダウンし、新規ログの Elasticsearch への送信を停止します。

```
$ oc -n openshift-logging patch daemonset/logging-fluentd -p '{"spec":{"template":{"spec":
{"nodeSelector":{"logging-infra-fluentd": "false"}}}}}'
```

4. OpenShift Container Platform es_util ツールを使用してシャードの同期フラッシュを実行して、シャットダウンの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 -- es_util --query="_flush/synced" -
XPOST
```

以下に例を示します。

```
$ oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6  -c elasticsearch -- es_util --
query="_flush/synced" -XPOST
```

出力例

```
{"_shards":{"total":4,"successful":4,"failed":0},".security":
{"total":2,"successful":2,"failed":0},".kibana_1":{"total":2,"successful":2,"failed":0}}
```

5. 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 -d '{ "persistent": { "cluster.routing.allocation.enable" :
"primaries" } }'
```

以下に例を示します。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" :
"primaries" } }'
```

出力例

```
{"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":
{"enable":"primaries"}}}},"transient":
```

6. コマンドが完了したら、ES クラスターのそれぞれのデプロイメントについて、以下を実行します。

   a. デフォルトで、OpenShift Container Platform Elasticsearch クラスターはノードのロールアウトをブロックします。以下のコマンドを使用してロールアウトを許可し、Pod が変更を取得できるようにします。

   ```
   $ oc rollout resume deployment/<deployment-name>
   ```

   以下に例を示します。

   ```
   $ oc rollout resume deployment/elasticsearch-cdm-0-1
   ```

   出力例

   ```
   deployment.extensions/elasticsearch-cdm-0-1 resumed
   ```

   新規 Pod がデプロイされます。Pod に準備状態のコンテナーがある場合、次のデプロイメントに進むことができます。

   ```
   $ oc get pods | grep elasticsearch-
   ```

   出力例

   ```
   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. デプロイメントが完了したら、ロールアウトを許可しないように Pod をリセットします。

```
$ oc rollout pause deployment/<deployment-name>
```

以下に例を示します。

```
$ oc rollout pause deployment/elasticsearch-cdm-0-1
```

出力例

```
deployment.extensions/elasticsearch-cdm-0-1 paused
```

c. Elasticsearch クラスターが **green** または **yellow** 状態にあることを確認します。

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query=_cluster/health?pretty=true
```

注記

直前のコマンドで使用した Elasticsearch Pod でロールアウトを実行した場合、Pod は存在しなくなっているため、ここで新規 Pod 名が必要になります。

以下に例を示します。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/health?pretty=true
```

```
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow", (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** または **yellow** であることを確認します。

7. 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。

8. 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 -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

以下に例を示します。

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

出力例

```
{
  "acknowledged" : true,
  "persistent" : { },
  "transient" : {
    "cluster" : {
      "routing" : {
        "allocation" : {
          "enable" : "all"
        }
      }
    }
  }
}
```

9. Fluentd Pod をスケールアップして、新規ログを Elasticsearch に送信します。

```
$ oc -n openshift-logging patch daemonset/logging-fluentd -p '{"spec":{"template":{"spec":
{"nodeSelector":{"logging-infra-fluentd": "true"}}}}}'
```

4.3.9. ログストアサービスのルートとしての公開

デフォルトでは、OpenShift Logging でデプロイされたログストア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、ログストアへの外部アクセスのために 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ログストア CA 証明書を作成して、ログストア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 要求でログストアサービスをホストする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、ログストアクラスター IP を使用してログストアサービスにアクセスできます。これは、以下のコマンドのいずれかを使用して取得できます。

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

出力例

```
172.30.183.229
```

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

出力例

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

以下のようなコマンドを使用して、クラスター IP アドレスを確認できます。

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

出力例

```
  % 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

- OpenShift Logging および Elasticsearch がインストールされている。
- ログにアクセスできるようになるには、プロジェクトへのアクセスが必要です。

手順

ログストア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. ログストアから 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
   ```

   出力例

   ```
   admin-ca
   ```

3. ログストア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: | 1
```

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

b. 以下のコマンドを実行して、前のステップで作成したルート YAML にログストア 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
```

出力例

```
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サービスアカウント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}"
```

以下のような出力が表示されます。

出力例

```
{
  "name" : "elasticsearch-cdm-i40ktba0-1",
  "cluster_name" : "elasticsearch",
```

```
  "cluster_uuid" : "0eY-tJzcR3KOdpgeMJo-MQ",
  "version" : {
  "number" : "6.8.1",
  "build_flavor" : "oss",
  "build_type" : "zip",
  "build_hash" : "Unknown",
  "build_date" : "Unknown",
  "build_snapshot" : true,
  "lucene_version" : "7.7.0",
  "minimum_wire_compatibility_version" : "5.6.0",
  "minimum_index_compatibility_version" : "5.0.0"
},
  "<tagline>" : "<for search>"
}
```

4.4. ログビジュアライザーの設定

OpenShift Container Platform は Kibana を使用して OpenShift Logging で収集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。

4.4.1. CPU およびメモリー制限の設定

OpenShift Logging コンポーネントは、CPU とメモリーの制限の両方への調整を許可します。

手順

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

```
$ oc -n openshift-logging edit ClusterLogging instance
```

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: 1
        limits:
          memory: 16Gi
        requests:
          cpu: 200m
          memory: 16Gi
      storage:
```

```
      storageClassName: "gp2"
      size: "200G"
    redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    resources: 2
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy:
      resources: 3
        limits:
          memory: 100Mi
        requests:
          cpu: 100m
          memory: 100Mi
    replicas: 2
collection:
  logs:
    type: "fluentd"
    fluentd:
      resources: 4
        limits:
          memory: 736Mi
        requests:
          cpu: 200m
          memory: 736Mi
```

1. 必要に応じてログの CPU およびメモリーの制限および要求を指定します。Elasticsearch の場合、要求値と制限値の両方を調整する必要があります。

2 3. 必要に応じて、ログビジュアライザーの CPU およびメモリーの制限および要求を指定します。

4. 必要に応じて、ログコレクターの CPU およびメモリーの制限および要求を指定します。

4.4.2. ログビジュアライザーノードの冗長性のスケーリング

冗長性を確保するために、ログビジュアライザーをホストする Pod をスケーリングできます。

手順

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

```
$ oc edit ClusterLogging instance
```

```
$ oc edit ClusterLogging instance

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

```
  name: "instance"

....

spec:
   visualization:
     type: "kibana"
     kibana:
       replicas: 1 (1)
```

(1) Kibana ノードの数を指定します。

4.5. OPENSHIFT LOGGING ストレージの設定

Elasticsearch はメモリー集約型アプリケーションです。デフォルトの OpenShift Logging インストールでは、メモリー要求およびメモリー制限の両方に対して 16G のメモリーをデプロイします。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、これは実稼働環境には推奨されません。

4.5.1. OpenShift Logging および OpenShift Container Platform のストレージについての考慮事項

永続ボリュームがそれぞれの Elasticsearch デプロイメント設定に必要です。OpenShift Container Platform では、これは永続ボリューム要求 (PVC) を使用して実行されます。

注記

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

OpenShift Elasticsearch Operator は Elasticsearch リソース名を使って PVC に名前を付けます。

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。

Elasticsearch では、大規模なマージ操作を実行するのに十分なメモリーが必要です。十分なメモリーがない場合、応答しなくなります。この問題を回避するには、必要なアプリケーションのログデータの量を評価し、空き容量の約 2 倍を割り当てます。

デフォルトで、ストレージ容量が 85% に達すると、Elasticsearch は新規データのノードへの割り当てを停止します。90% になると、Elasticsearch は可能な場合に既存のシャードをそのノードから他のノードに移動しようとします。ただし、空き容量のレベルが 85% 未満のノードがない場合、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。これらのデフォルト値は変更できます。アラートは同じデフォルト値を使用しますが、これらの値をアラートで変更することはできません。

4.5.2. 関連情報

- ログストアの永続ストレージの設定

4.6. OPENSHIFT LOGGING コンポーネントの CPU およびメモリー制限の設定

必要に応じて、それぞれの OpenShift Logging コンポーネントの CPU およびメモリー制限の両方を設定できます。

4.6.1. CPU およびメモリー制限の設定

OpenShift Logging コンポーネントは、CPU とメモリーの制限の両方への調整を許可します。

手順

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

```
$ oc -n openshift-logging edit ClusterLogging instance
```

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: 1
        limits:
          memory: 16Gi
        requests:
          cpu: 200m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: 2
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      proxy:
```

```
      resources: 3
        limits:
          memory: 100Mi
        requests:
          cpu: 100m
          memory: 100Mi
    replicas: 2
collection:
  logs:
    type: "fluentd"
    fluentd:
      resources: 4
        limits:
          memory: 736Mi
        requests:
          cpu: 200m
          memory: 736Mi
```

1 必要に応じてログの CPU およびメモリーの制限および要求を指定します。Elasticsearch の場合、要求値と制限値の両方を調整する必要があります。

2 3 必要に応じて、ログビジュアライザーの CPU およびメモリーの制限および要求を指定します。

4 必要に応じて、ログコレクターの CPU およびメモリーの制限および要求を指定します。

4.7. 容認を使用した OPENSHIFT LOGGING POD 配置の制御

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

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

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

容認を使用した OpenShift Logging CR のサンプル

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
```

```
    tolerations: 1
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
    resources:
      limits:
        memory: 16Gi
      requests:
        cpu: 200m
        memory: 16Gi
    storage: {}
    redundancyPolicy: "ZeroRedundancy"
visualization:
  type: "kibana"
  kibana:
    tolerations: 2
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi
    replicas: 1
collection:
  logs:
    type: "fluentd"
    fluentd:
      tolerations: 3
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 100m
          memory: 1Gi
```

1 この容認は Elasticsearch Pod に追加されます。

2 この容認は Kibana Pod に追加されます。

3 この容認はロギングコレクター Pod に追加されます。

4.7.1. 容認を使用したログストア Pod の配置の制御

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

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

デフォルトで、ログストア Pod には以下の容認があります。

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

   以下に例を示します。

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

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

2. **ClusterLogging** CR の **logstore** セクションを編集し、Elasticsearch Pod の容認を設定します。

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

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

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

4.7.2. 容認を使用したログビジュアライザー Pod の配置の制御

ログビジュアライザー Pod が実行されるノードを制御し、Pod の容認を使用して他のワークロードがそれらのノードを使用しないようにすることができます。

ClusterLogging カスタムリソース (CR) を使用して容認をログビジュアライザー Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value pair** です。他の Pod にはない特定の **key:value** ペアを使用することで、Kibana Pod のみをそのノード上で実行できます。

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

以下に例を示します。

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

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

2. **ClusterLogging** CR の **visualization** セクションを編集し、Kibana Pod の容認を設定します。

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

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

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

4.7.3. 容認を使用したログコレクター Pod 配置の制御

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

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

デフォルトで、ロギングコレクター Pod には以下の容認があります。

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

以下に例を示します。

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

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

2. **ClusterLogging** カスタムリソース (CR) の **collection** スタンザを編集して、ロギングコレクター Pod の容認を設定します。

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

(1) ノードに追加したキーを指定します。

2 **Exists** Operator を指定して、**key/value/effect** パラメーターが一致するようにします。

3 **NoExecute** effect を指定します。

4 オプションで、**tolerationSeconds** パラメーターを指定して、エビクトされる前に Pod がノードにバインドされる期間を設定します。

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

4.7.4. 関連情報

- ノードテイントを使用した Pod 配置の制御.

4.8. ノードセレクターを使用した OPENSHIFT LOGGING リソースの移動

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

4.8.1. OpenShift Logging リソースの移動

Elasticsearch および Kibana などの OpenShift Logging コンポーネントの 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。

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。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 **ClusterLogging** 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

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
```

```
      nodeSelector: 1
        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: 2
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

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

検証

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

以下に例を示します。

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

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

出力例

```
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
```

出力例

```
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.20.0
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.20.0
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.20.0
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.20.0
```

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
```

出力例

```
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、**ClusterLogging** 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: 1
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 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
```

出力例

```
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-nkxl7                       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
```

出力例

```
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
```

出力例

```
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-nkxl7                       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
```

4.9. SYSTEMD-JOURNALD および FLUENTD の設定

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

ジャーナルでエントリーが失われるのを防ぐことができるように **RateLimitIntervalSec=30s** 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。

4.9.1. OpenShift Logging 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、OpenShift Logging がログをドロップせずに過剰なリソースを使用しないようにすることができます。

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

手順

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

```
Compress=yes (1)
ForwardToConsole=no (2)
ForwardToSyslog=no
MaxRetentionSec=1month (3)
RateLimitBurst=10000 (4)
RateLimitIntervalSec=30s
Storage=persistent (5)
SyncIntervalSec=1s (6)
SystemMaxUse=8G (7)
SystemKeepFree=20% (8)
SystemMaxFileSize=10M (9)
```

(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) 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ーに転送します。
- **ForwardToSyslog**: syslog デーモンに転送します。
- **ForwardToWall**: メッセージを wall メッセージとしてすべてのログインしているユーザーに転送します。

(3) ジャーナルエントリーを保存する最大時間を指定します。数字を入力して秒数を指定します。または、year、month、week、day、h または m などの単位を含めます。無効にするには **0** を入力します。デフォルトは **1month** です。

(4) レート制限を設定します。**RateLimitIntervalSec** で定義される期間に、 **RateLimitBurst** で指定される以上のログが受信される場合、この期間内の追加のメッセージすべてはこの期間が終了するまでにドロップされます。デフォルト値である **RateLimitIntervalSec=30s** および **RateLimitBurst=10000** を設定することが推奨されます。

5 ログの保存方法を指定します。デフォルトは **persistent** です。

- **volatile**: ログを **/var/log/journal/** のメモリーに保存します。
- **persistent**: ログを **/var/log/journal/** のディスクに保存します。systemd は存在しない場合はディレクトリーを作成します。
- **auto**: ディレクトリーが存在する場合に、ログを **/var/log/journal/** に保存します。存在しない場合は、systemd はログを **/run/systemd/journal** に一時的に保存します。
- **none**: ログを保存しません。systemd はすべてのログをドロップします。

6 ERR、WARNING、NOTICE、INFO、および DEBUG ログについてジャーナルファイルをディスクに同期させるまでのタイムアウトを指定します。 systemd は、**CRIT**、**ALERT**、または **EMERG** ログの受信後すぐに同期を開始します。デフォルトは **1s** です。

7 ジャーナルが使用できる最大サイズを指定します。デフォルトは **8G** です。

8 systemd が残す必要のあるディスク領域のサイズを指定します。デフォルトは **20%** です。

9 **/var/log/journal** に永続的に保存される個別のジャーナルファイルの最大サイズを指定します。デフォルトは **10M** です。

注記

レート制限を削除する場合、システムロギングデーモンの CPU 使用率が高くなることがあります。 以前はスロットリング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. 次のコマンドを実行して、**journal.conf** ファイルを base64 に変換し、**jrnl_cnf** という名前の変数に保存します。

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

3. 前の手順で作成した **jrnl_cnf** 変数を含む **MachineConfig** オブジェクトを作成します。次のサンプルコマンドは、ワーカーの **MachineConfig** オブジェクトを作成します。

```
$ cat << EOF > ./40-worker-custom-journald.yaml 1
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 2
  name: 40-worker-custom-journald 3
spec:
  config:
    ignition:
      config: {}
```

```
    security:
      tls: {}
    timeouts: {}
    version: 3.2.0
  networkd: {}
  passwd: {}
  storage:
    files:
    - contents:
        source: data:text/plain;charset=utf-8;base64,${jrnl_cnf} 4
        verification: {}
      filesystem: root
      mode: 0644 5
      path: /etc/systemd/journald.conf.d/custom.conf
 osImageURL: ""
EOF
```

1. オプション: コントロールプレーン (マスターとも呼ばれます) ノードの場合、ファイル名を **40-master-custom-journald.yaml** として指定できます。
2. オプション: コントロールプレーン (マスターとも呼ばれます) ノードの場合、**master** としてのロールを指定します。
3. オプション: コントロールプレーン (マスターとも呼ばれます) ノードの場合、名前を **40-master-custom-journald** として指定できます。
4. オプション: **journald.conf** ファイルにパラメーターの静的コピーを含めるには、**${jrnl_cnf}** を **echo$jrnl_cnf** コマンドの出力に置き換えます。
5. **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。

4. 次のコマンドを実行して、マシン設定を作成します。

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

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

5. 次のコマンドを実行して、各ノードへの新しいレンダリング設定のロールアウトのステータスを監視します。

```
$ oc describe machineconfigpool/<node> 1
```

1. ノードを **master** または **worker** として指定します。

worker の出力例

```
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
```

4.10. メンテナンスとサポート

4.10.1. サポートされる設定

OpenShift Logging の設定のサポートされる方法として、本書で説明されているオプションを使用してこれを設定することができます。サポートされていない他の設定は使用しないでください。設定のパラダイムが OpenShift Container Platform リリース間で変更される可能性があり、このような変更は、設定のすべての可能性が制御されている場合のみ適切に対応できます。本書で説明されている設定以外の設定を使用する場合、OpenShift Elasticsearch Operator および Red Hat OpenShift Logging Operator が差分を調整するため、変更内容は失われます。Operator はデフォルトで定義された状態にすべて戻します。

注記

OpenShift Container Platform ドキュメントで説明されていない設定を実行する **必要がある** 場合、Red Hat OpenShift Logging Operator または OpenShift Elasticsearch Operator を **Unmanaged** (管理外) に設定する **必要があります**。管理外の OpenShift Logging 環境は **サポート外** であり、OpenShift Logging を **Managed** に戻すまで変更を受信しません。

4.10.2. サポートされない設定

以下のコンポーネントを変更するには、Red Hat OpenShift Logging Operator を Unmanaged (管理外) の状態に設定する必要があります。

- **Elasticsearch** CR
- Kibana デプロイメント
- **fluent.conf** ファイル
- Fluentd デーモンセット

以下のコンポーネントを変更するには、OpenShift Elasticsearch Operator を Unmanaged(管理外) の状態に設定する必要があります。

- Elasticsearch デプロイメントファイル。

明示的にサポート対象外とされているケースには、以下が含まれます。

- **デフォルトのログローテーションの設定**。デフォルトのログローテーション設定は変更できません。
- **収集したログの場所の設定**。ログコレクターの出力ファイルの場所を変更することはできません。デフォルトは **/var/log/fluentd/fluentd.log** です。

- **ログコレクションのスロットリング**。ログコレクターによってログが読み取られる速度を調整して減速することはできません。
- **環境変数を使用したロギングコレクターの設定**。環境変数を使用してログコレクターを変更することはできません。
- **ログコレクターによってログを正規化する方法の設定**。デフォルトのログの正規化を変更することはできません。

4.10.3. 管理外の Operator のサポートポリシー

Operator の **管理状態** は、Operator が設計通りにクラスター内の関連するコンポーネントのリソースをアクティブに管理しているかどうかを定めます。Operator が **unmanaged** 状態に設定されている場合、これは設定の変更に応答せず、更新を受信しません。

これは非実稼働クラスターやデバッグ時に便利ですが、管理外の状態の Operator はサポートされず、クラスター管理者は個々のコンポーネント設定およびアップグレードを完全に制御していることを前提としています。

Operator は以下の方法を使用して管理外の状態に設定できます。

- **個別の Operator 設定**
  個別の Operator には、それらの設定に **managementState** パラメーターがあります。これは Operator に応じてさまざまな方法でアクセスできます。たとえば、Red Hat OpenShift Logging Operator は管理するカスタムリソース (CR) を変更することによってこれを実行しますが、Cluster Samples Operator はクラスター全体の設定リソースを使用します。

  managementState パラメーターを **Unmanaged** に変更する場合、Operator はそのリソースをアクティブに管理しておらず、コンポーネントに関連するアクションを取らないことを意味します。Operator によっては、クラスターが破損し、手動リカバリーが必要になる可能性があるため、この管理状態に対応しない可能性があります。

  警告

  個別の Operator を **Unmanaged** 状態に変更すると、特定のコンポーネントおよび機能がサポート対象外になります。サポートを継続するには、報告された問題を **Managed** 状態で再現する必要があります。

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

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

警告

CVO のオーバーライドを設定すると、クラスター全体がサポートされない状態になります。サポートを継続するには、オーバーライドを削除した後に、報告された問題を再現する必要があります。

第5章 リソースのログの表示

OpenShift CLI (oc) および Web コンソールを使用して、ビルド、デプロイメント、および Pod などの各種リソースのログを表示できます。

注記

リソースログは、制限されたログ表示機能を提供するデフォルトの機能です。ログの取得および表示のエクスペリエンスを強化するには、OpenShift Logging をインストールすることが推奨されます。OpenShift Logging は、ノードシステムの監査ログ、アプリケーションコンテナーログ、およびインフラストラクチャーログなどの OpenShift Container Platform クラスターからのすべてのログを専用のログストアに集計します。次に、Kibana インターフェイス を使用してログデータをクエリーし、検出し、可視化できます。リソースログは OpenShift Logging ログストアにアクセスしません。

5.1. リソースログの表示

OpenShift CLI (oc) および Web コンソールで、各種リソースのログを表示できます。ログの末尾から読み取られるログ。

前提条件

- OpenShift CLI (oc) へのアクセス。

手順 (UI)

1. OpenShift Container Platform コンソールで **Workloads → Pods** に移動するか、または調査するリソースから Pod に移動します。

注記

ビルドなどの一部のリソースには、直接クエリーする Pod がありません。このような場合には、リソースについて **Details** ページで **Logs** リンクを特定できます。

2. ドロップダウンメニューからプロジェクトを選択します。
3. 調査する Pod の名前をクリックします。
4. **Logs** をクリックします。

手順 (CLI)

- 特定の Pod のログを表示します。

```
$ oc logs -f <pod_name> -c <container_name>
```

ここでは、以下のようになります。

-f
オプション: ログに書き込まれている内容に沿って出力することを指定します。

<pod_name>

Pod の名前を指定します。

<container_name>

オプション: コンテナーの名前を指定します。Pod に複数のコンテナーがある場合、コンテナー名を指定する必要があります。

以下に例を示します。

```
$ oc logs ruby-58cd97df55-mww7r
```

```
$ oc logs -f ruby-57f7f4855b-znl92 -c ruby
```

ログファイルの内容が出力されます。

- 特定のリソースのログを表示します。

```
$ oc logs <object_type>/<resource_name> 1
```

1 リソースタイプおよび名前を指定します。

以下に例を示します。

```
$ oc logs deployment/ruby
```

ログファイルの内容が出力されます。

第6章 KIBANA を使用したクラスターログの表示

OpenShift Logging には、収集したログデータを視覚化するための Web コンソールが含まれます。現時点で、OpenShift Container Platform では、可視化できるように Kibana コンソールをデプロイします。

ログビジュアライザーを使用して、データで以下を実行できます。

- **Discover** タブを使用してデータを検索し、参照します。
- **Visualize** タブ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** タブ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 インターフェイ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 ドキュメント を参照してください。

注記

監査ログは、デフォルトでは内部 OpenShift Container Platform Elasticsearch インスタンスに保存されません。Kibana で監査ログを表示するには、ログ転送 API を使用して、監査ログの **default** 出力を使用するパイプラインを設定する必要があります。

6.1. KIBANA インデックスパターンの定義

インデックスパターンは、可視化する必要のある Elasticsearch インデックスを定義します。Kibana でデータを確認し、可視化するには、インデックスパターンを作成する必要があります。

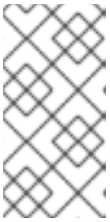
前提条件

- Kibana で **infra** および **audit** インデックスを表示するには、ユーザーには **cluster-admin** ロール、 **cluster-reader** ロール、または両方のロールが必要です。デフォルトの **kubeadmin** ユーザーには、これらのインデックスを表示するための適切なパーミッションがあります。**default**、**kube-** および **openshift-** プロジェクトで Pod およびログを表示できる場合、これらのインデックスにアクセスできるはずです。以下のコマンドを使用して、現在のユーザーが適切なパーミッションを持っているかどうかを確認することができます。

```
$ oc auth can-i get pods/log -n <project>
```

出力例

```
yes
```



注記

監査ログは、デフォルトでは内部 OpenShift Container Platform Elasticsearch インスタンスに保存されません。Kibana で監査ログを表示するには、ログ転送 API を使用して監査ログの **default** 出力を使用するパイプラインを設定する必要があります。

- Elasticsearch ドキュメントは、インデックスパターンを作成する前にインデックス化する必要があります。これは自動的に実行されますが、新規または更新されたクラスターでは数分の時間がかかる可能性があります。

手順

Kibana でインデックスパターンを定義し、ビジュアライゼーションを作成するには、以下を実行します。

1. OpenShift Container Platform コンソールで、Application Launcher  をクリックし、**Logging** を選択します。
2. **Management → Index Patterns → Create index pattern** をクリックして Kibana インデックスパターンを作成します。
   - 各ユーザーは、プロジェクトのログを確認するために、Kibana に初めてログインする際にインデックスパターンを手動で作成する必要があります。ユーザーは **app** という名前のインデックスパターンを作成し、**@timestamp** 時間フィールドを使用してコンテナーログを表示する必要があります。
   - 管理ユーザーはそれぞれ、最初に Kibana にログインする際に、**@timestamp** 時間フィールドを使用して **app**、 **infra** および **audit** インデックスについてインデックスパターンを作成する必要があります。
3. 新規インデックスパターンから Kibana のビジュアライゼーション (Visualization) を作成します。

6.2. KIBANA を使用したクラスターログの表示

Kibana Web コンソールでクラスターのログを表示します。Kibana でデータを表示し、可視化する方法については、本書では扱いません。詳細は、Kibana ドキュメント を参照してください。

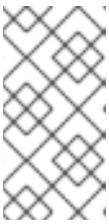
前提条件

- OpenShift Logging および Elasticsearch がインストールされている。
- Kibana インデックスパターンが存在する。
- Kibana で **infra** および **audit** インデックスを表示するには、ユーザーには **cluster-admin** ロール、 **cluster-reader** ロール、または両方のロールが必要です。デフォルトの **kubeadmin** ユーザーには、これらのインデックスを表示するための適切なパーミッションがあります。**default**、**kube-** および **openshift-** プロジェクトで Pod およびログを表示できる場合、これらのインデックスにアクセスできるはずです。以下のコマンドを使用して、現在のユーザーが適切なパーミッションを持っているかどうかを確認することができます。

```
$ oc auth can-i get pods/log -n <project>
```

出力例

```
yes
```



注記

監査ログは、デフォルトでは内部 OpenShift Container Platform Elasticsearch インスタンスに保存されません。Kibana で監査ログを表示するには、ログ転送 API を使用して監査ログの **default** 出力を使用するパイプラインを設定する必要があります。

手順

Kibana でログを表示するには、以下を実行します。

1. OpenShift Container Platform コンソールで、Application Launcher  をクリックし、**Logging** を選択します。

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 インターフェイスが起動します。

3. Kibana で **Discover** をクリックします。

4. 左上隅のドロップダウンメニューから作成したインデックスパターン (**app**、**audit**、または **infra**) を選択します。
ログデータは、タイムスタンプ付きのドキュメントとして表示されます。

5. タイムスタンプ付きのドキュメントの1つを展開します。

6. **JSON** タブをクリックし、ドキュメントのログエントリーを表示します。

例6.1 Kibana のインフラストラクチャーログエントリーのサンプル

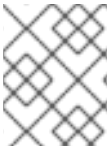

```
{
  "_index": "infra-000001",
  "_type": "_doc",
  "_id": "YmJmYTBlNDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_version": 1,
  "_score": null,
  "_source": {
    "docker": {
      "container_id": "f85fa55bbef7bb783f041066be1e7c267a6b88c4603dfce213e32c1"
    },
    "kubernetes": {
      "container_name": "registry-server",
      "namespace_name": "openshift-marketplace",
      "pod_name": "redhat-marketplace-n64gc",
      "container_image": "registry.redhat.io/redhat/redhat-marketplace-index:v4.7",
      "container_image_id": "registry.redhat.io/redhat/redhat-marketplace-
index@sha256:65fc0c45aabb95809e376feb065771ecda9e5e59cc8b3024c4545c168f",
      "pod_id": "8f594ea2-c866-4b5c-a1c8-a50756704b2a",
      "host": "ip-10-0-182-28.us-east-2.compute.internal",
      "master_url": "https://kubernetes.default.svc",
      "namespace_id": "3abab127-7669-4eb3-b9ef-44c04ad68d38",
      "namespace_labels": {
        "openshift_io/cluster-monitoring": "true"
      },
      "flat_labels": [
        "catalogsource_operators_coreos_com/update=redhat-marketplace"
      ]
    },
    "message": "time=\"2020-09-23T20:47:03Z\" level=info msg=\"serving registry\"
database=/database/index.db port=50051",
    "level": "unknown",
    "hostname": "ip-10-0-182-28.internal",
    "pipeline_metadata": {
      "collector": {
```

```
          "ipaddr4": "10.0.182.28",
          "inputname": "fluent-plugin-systemd",
          "name": "fluentd",
          "received_at": "2020-09-23T20:47:15.007583+00:00",
          "version": "1.7.4 1.6.0"
        }
      },
      "@timestamp": "2020-09-23T20:47:03.422465+00:00",
      "viaq_msg_id": "YmJmYTBlNDktMDMGQtMjE3NmFiOGUyOWM3",
      "openshift": {
        "labels": {
          "logging": "infra"
        }
      }
    },
    "fields": {
      "@timestamp": [
        "2020-09-23T20:47:03.422Z"
      ],
      "pipeline_metadata.collector.received_at": [
        "2020-09-23T20:47:15.007Z"
      ]
    },
    "sort": [
      1600894023422
    ]
  }
```

第7章 ログのサードパーティーシステムへの転送

デフォルトで、OpenShift Logging は **ClusterLogging** カスタムリソースに定義されるデフォルトの内部 Elasticsearch ログストアにコンテナーおよびインフラストラクチャーログを送信します。ただし、監査ログは内部ストアに送信しません。セキュアなストレージを提供しないためです。このデフォルト設定が要件を満たす場合には、クラスターログフォワーダーを設定する必要はありません。

ログを他のログアグリゲーターに送信するには、OpenShift Container Platform クラスターログフォワーダーを使用します。この API を使用すると、コンテナー、インフラストラクチャーおよび監査ログをクラスター内外の特定のエンドポイントに送信できます。さらに、異なるタイプのログをさまざまなシステムに送信できるため、さまざまなユーザーがそれぞれのタイプにアクセスできます。また、Transport Layer Security (TLS) のサポートを有効にして、組織の要件に合わせてログを安全に送信することもできます。



注記

監査ログを内部ログストアに送信するには、監査ログのログストアへの転送 で説明されているようにカスタムログフォワーダーを使用します。

ログを外部に転送する場合、Red Hat OpenShift Logging Operator は必要なプロトコルを使用してログを送信するように Fluentd 設定マップを作成するか、またはこれを変更します。外部ログアグリゲーターでプロトコルを設定する必要があります。

または、設定マップを作成して Fluentd **転送** プロトコル、または syslog プロトコル を使用し、ログを外部システムに送信できます。ただし、ログを転送するためのこれらの方法は OpenShift Container Platform では非推奨となり、今後のリリースでは取り除かれます。

重要

同じクラスターで設定マップのメソッドおよびクラスターログフォワーダーを使用することはできません。

7.1. ログのサードパーティーシステムへの転送

クラスターログを外部サードパーティーシステムに転送するには、**ClusterLogForwarder** カスタムリソース (CR) に指定される **出力** および **パイプライン** を使用して、OpenShift Container Platform クラスター内外の特定のエンドポイントにログを送信します。**入力** を使用して、特定のプロジェクトに関連付けられたアプリケーションログをエンドポイントに転送することもできます。

- **出力** は、定義するログデータの宛先、またはログの送信先です。出力は以下のいずれかのタイプになります。
  - **elasticsearch**.外部 Elasticsearch 6(すべてのリリース) インスタンス。**elasticsearch** 出力では、TLS 接続を使用できます。
  - **fluentdForward**。Fluentd をサポートする外部ログ集計ソリューション。このオプションは Fluentd **forward** プロトコルを使用します。**fluentForward** 出力は TCP または TLS 接続を使用でき、シークレットに **shared_key** フィールドを指定して共有キーの認証をサポートします。共有キーの認証は、TLS の有無に関係なく使用できます。
  - **syslog**。syslog RFC3164 または RFC5424 プロトコルをサポートする外部ログ集計ソリューション。**syslog** 出力は、UDP、TCP、または TLS 接続を使用できます。
  - **kafka**。Kafka ブローカー。**kafka** 出力は TCP または TLS 接続を使用できます。

- **default**。内部 OpenShift Container Platform Elasticsearch インスタンス。デフォルトの出力を設定する必要はありません。**default** 出力を設定する場合、**default** 出力は Red Hat OpenShift Logging Operator 用に予約されるため、エラーメッセージが送信されます。

出力 URL スキームに TLS(HTTPS、TLS、または UDPS) が必要な場合、TLS サーバー側の認証が有効になります。クライアント認証も有効にするには、出力で **openshift-logging** プロジェクトのシークレットに名前を指定する必要があります。シークレットには、**tls.crt**、**tls.key**、および **ca-bundle.crt** のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。

- **パイプライン** は、1つのログタイプから1つまたは複数の出力への単純なルーティング、または送信するログを定義します。ログタイプは以下のいずれかになります。

  - **application**。クラスターで実行される、インフラストラクチャーコンテナーアプリケーションを除くユーザーアプリケーションによって生成されるコンテナーログ。

  - **infrastructure**。**openshift***、**kube***、または **default** プロジェクトで実行される Pod のコンテナーログおよびノードファイルシステムから取得されるジャーナルログ。

  - **audit**auditd、ノード監査システム、および Kubernetes API サーバーおよび OpenShift API サーバーから生成される監査ログで生成されるログ。

  パイプラインで **key:value** ペアを使用すると、アウトバウンドログメッセージにラベルを追加できます。たとえば、他のデータセンターに転送されるメッセージにラベルを追加したり、タイプ別にログにラベルを付けたりすることができます。オブジェクトに追加されるラベルもログメッセージと共に転送されます。

- **入力** は、特定のプロジェクトに関連付けられるアプリケーションログをパイプラインに転送します。

パイプラインでは、**inputRef** パラメーターを使用して転送するログタイプと、**outputRef** パラメーターを使用してログを転送する場所を定義します。

次の点に注意してください。

- **ClusterLogForwarder** CR オブジェクトが存在する場合に、**default** 出力のあるパイプラインがない限り、ログはデフォルト Elasticsearch インスタンスに転送されません。

- デフォルトで、OpenShift Logging は **ClusterLogging** カスタムリソースに定義されるデフォルトの内部 Elasticsearch ログストアにコンテナーおよびインフラストラクチャーログを送信します。ただし、監査ログは内部ストアに送信しません。セキュアなストレージを提供しないためです。このデフォルト設定が要件を満たす場合には、ログ転送 API を設定する必要はありません。

- ログタイプのパイプラインを定義しない場合、未定義タイプのログはドロップされます。たとえば、**application** および **audit** タイプのパイプラインを指定するものの、**infrastructure** タイプのパイプラインを指定しない場合、**infrastructure** ログはドロップされます。

- **ClusterLogForwarder** カスタムリソース (CR) で出力の複数のタイプを使用し、ログを複数の異なるプロトコルをサポートするサーバーに送信できます。

- 内部 OpenShift Container Platform Elasticsearch インスタンスは、監査ログのセキュアなストレージを提供しません。監査ログを転送するシステムが組織および政府の規制に準拠しており、適切にセキュリティーが保護されていることを確認することが推奨されています。OpenShift Logging はこれらの規制に準拠しません。

- キーやシークレット、サービスアカウント、ポートのオープン、またはグローバルプロキシー設定など、外部の宛先で必要となる可能性のある追加設定を作成し、維持する必要があります。

以下の例では、監査ログをセキュアな外部 Elasticsearch インスタンスに転送し、インフラストラクチャーログをセキュアでない外部 Elasticsearch インスタンスに、アプリケーションログを Kafka ブローカーに転送し、アプリケーションログを **my-apps-logs** プロジェクトから内部 Elasticsearch インスタンスに転送します。

ログ転送の出力とパイプラインのサンプル

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: elasticsearch-secure (3)
     type: "elasticsearch"
     url: https://elasticsearch.secure.com:9200
     secret:
        name: elasticsearch
   - name: elasticsearch-insecure (4)
     type: "elasticsearch"
     url: http://elasticsearch.insecure.com:9200
   - name: kafka-app (5)
     type: "kafka"
     url: tls://kafka.secure.com:9093/app-topic
  inputs: (6)
   - name: my-app-logs
     application:
        namespaces:
        - my-project
  pipelines:
   - name: audit-logs (7)
     inputRefs:
      - audit
     outputRefs:
      - elasticsearch-secure
      - default
     parse: json (8)
     labels:
       secure: "true" (9)
       datacenter: "east"
   - name: infrastructure-logs (10)
     inputRefs:
      - infrastructure
     outputRefs:
      - elasticsearch-insecure
     labels:
       datacenter: "west"
   - name: my-app (11)
     inputRefs:
```

```
    - my-app-logs
  outputRefs:
    - default
- inputRefs: 12
    - application
  outputRefs:
    - kafka-app
  labels:
    datacenter: "south"
```

1 **ClusterLogForwarder** CR の名前は **instance** である必要があります。

2 **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。

3 シークレットとセキュアな URL を使用したセキュアな Elasticsearch 出力の設定。

- 出力を記述する名前。
- 出力のタイプ: **elasticsearch**。
- 接頭辞を含む、有効な絶対 URL としての Elasticsearch インスタンスのセキュアな URL およびポート。
- TLS 通信のエンドポイントで必要なシークレット。シークレットは **openshift-logging** プロジェクトに存在する必要があります。

4 非セキュアな Elasticsearch 出力の設定:

- 出力を記述する名前。
- 出力のタイプ: **elasticsearch**。
- 接頭辞を含む、有効な絶対 URL として Elasticsearch インスタンスのセキュアではない URL およびポート。

5 セキュアな URL を介したクライアント認証 TLS 通信を使用した Kafka 出力の設定

- 出力を記述する名前。
- 出力のタイプ: **kafka**。
- Kafka ブローカーの URL およびポートを、接頭辞を含む有効な絶対 URL として指定します。

6 **my-project** namespace からアプリケーションログをフィルターするための入力の設定。

7 監査ログをセキュアな外部 Elasticsearch インスタンスに送信するためのパイプラインの設定。

- オプション。パイプラインを説明する名前。
- **inputRefs** はログタイプです (例: **audit**)。
- **outputRefs** は使用する出力の名前です。この例では、**elasticsearch-secure** はセキュアな Elasticsearch インスタンスに転送され、**default** は内部 Elasticsearch インスタンスに転送されます。
- オプション: ログに追加する複数のラベル。

8 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。

9 オプション: 文字列。ログに追加する 1 つまたは複数のラベル。"true" などの引用値は、ブール値としてではなく、文字列値として認識されるようにします。

10 インフラストラクチャーログをセキュアでない外部 Elasticsearch インスタンスに送信するためのパイプラインの設定。

11 **my-project** プロジェクトから内部 Elasticsearch インスタンスにログを送信するためのパイプラインの設定。

- オプション。パイプラインを説明する名前。
- **inputRefs** は特定の入力 **my-app-logs** です。
- **outputRefs** は **default** です。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

12 パイプライン名がない場合にログを Kafka ブローカーに送信するためのパイプラインの設定。

- **inputRefs** はログタイプです (例: **application**)。
- **outputRefs** は使用する出力の名前です。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

外部ログアグリゲーターが利用できない場合の Fluentd のログの処理
外部ロギングアグリゲーターが利用できず、ログを受信できない場合、Fluentd は継続してログを収集し、それらをバッファーに保存します。ログアグリゲーターが利用可能になると、バッファーされたログを含む、ログの転送が再開されます。バッファーが完全に一杯になると、Fluentd はログの収集を停止します。OpenShift Container Platform はログをローテーションし、それらを削除します。バッファーサイズを調整したり、永続ボリューム要求 (PVC) を Fluentd デーモンセットまたは Pod に追加したりすることはできません。

7.2. サポート対象のログデータ出力タイプ

Red Hat OpenShift Logging バージョン 5.0 は、ログデータをターゲットログコレクターに送信するために以下の出力タイプおよびプロトコルを提供します。

Red Hat は、以下の表に記載されているそれぞれの組み合わせをテストします。ただし、これらのプロトコルを取り込むより広範囲のターゲットログコレクターにログデータを送信できるはずです。

出力のタイプ	プロトコル	テストで使用
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 7.10.1

出力のタイプ	プロトコル	テストで使用
syslog	RFC-3164、RFC-5424	rsyslog 8.37.0-9.el7
kafka	kafka 0.11	kafka 2.4.1

注記

以前のバージョンでは、syslog 出力は RFC-3164 でのみサポートされました。現在の syslog 出力では RFC-5424 のサポートを追加します。

7.3. 外部 ELASTICSEARCH インスタンスへのログの送信

オプションで、内部 OpenShift Container Platform Elasticsearch インスタンスに加えて、またはその代わりに外部 Elasticsearch インスタンスにログを転送できます。外部ログアグリゲーターを OpenShift Container Platform からログデータを受信するように設定する必要があります。

外部 Elasticsearch インスタンスへのログ転送を設定するには、そのインスタンスへの出力および出力を使用するパイプラインで **ClusterLogForwarder** カスタムリソース (CR) を作成します。外部 Elasticsearch 出力では、HTTP(セキュアでない) または HTTPS(セキュアな HTTP) 接続を使用できます。

外部 Elasticsearch インスタンスと内部 Elasticsearch インスタンスの両方にログを転送するには、出力および外部インスタンスへのパイプライン、および **default** 出力を使用してログを内部インスタンスに転送するパイプラインを作成します。**default** 出力を作成する必要はありません。**default** 出力を設定する場合、**default** 出力は Red Hat OpenShift Logging Operator 用に予約されるため、エラーメッセージが送信されます。

注記

ログを内部 OpenShift Container Platform Elasticsearch インスタンス ***のみ*** に転送する必要がある場合は、**ClusterLogForwarder** CR を作成する必要はありません。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

手順

1. 以下のように **ClusterLogForwarder** CR YAML ファイルを作成します。

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: elasticsearch-insecure 3
     type: "elasticsearch" 4
     url: http://elasticsearch.insecure.com:9200 5
```

```
  - name: elasticsearch-secure
    type: "elasticsearch"
    url: https://elasticsearch.secure.com:9200
    secret:
      name: es-secret 6
  pipelines:
  - name: application-logs 7
    inputRefs: 8
    - application
    - audit
    outputRefs:
    - elasticsearch-secure 9
    - default 10
    parse: json 11
    labels:
      myLabel: "myValue" 12
  - name: infrastructure-audit-logs 13
    inputRefs:
    - infrastructure
    outputRefs:
    - elasticsearch-insecure
    labels:
      logs: "audit-infra"
```

1. **ClusterLogForwarder** CR の名前は **instance** である必要があります。
2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. 出力の名前を指定します。
4. **elasticsearch** タイプを指定します。
5. 外部 Elasticsearch インスタンスの URL およびポートを有効な絶対 URL として指定します。**http** (セキュアでない) プロトコルまたは **https** (セキュアな HTTP) プロトコルを使用できます。CIDR アノテーションを使用するクラスター全体のプロキシーが有効になっている場合、出力は IP アドレスではなくサーバー名または FQDN である必要があります。
6. **https** 接頭辞を使用する場合、TLS 通信のエンドポイントに必要なシークレットの名前を指定する必要があります。シークレットは **openshift-logging** プロジェクトに存在し、**tls.crt**、**tls.key**、および **ca-bundle.crt** のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。
7. オプション: パイプラインの名前を指定します。
8. パイプラインを使用して、転送する必要のあるログタイプを指定します (**application** **infrastructure**、または **audit**)。
9. ログを転送するためにそのパイプラインで使用する出力を指定します。
10. オプション: ログを内部 Elasticsearch インスタンスに送信するために **default** 出力を指定します。
11. 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。

12 オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

13 オプション: サポートされるタイプの他の外部ログアグリゲーターにログを転送するように複数の出力を設定します。

- オプション。パイプラインを説明する名前。
- **inputRefs** は、そのパイプラインを使用して転送するログタイプです (**application**、**infrastructure**、または **audit**)。
- **outputRefs** は使用する出力の名前です。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

2. CR オブジェクトを作成します。

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

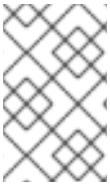
Red Hat OpenShift Logging Operator は Fluentd Pod を再デプロイします。Pod が再デプロイされない場合、強制的に再デプロイするために Fluentd Pod を削除できます。

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

7.4. FLUENTD 転送プロトコルを使用したログの転送

Fluentd **forward** プロトコルを使用して、デフォルトの Elasticsearch ログストアの代わり、またはこれに加えてプロトコルを受け入れるように設定された外部ログアグリゲーターにログのコピーを送信できます。外部ログアグリゲーターを OpenShift Container Platform からログを受信するように設定する必要があります。

forward プロトコルを使用してログ転送を設定するには、Fluentd サーバーに対する 1 つ以上の出力およびそれらの出力を使用するパイプラインと共に **ClusterLogForwarder** カスタムリース (CR) を作成します。Fluentd の出力は TCP(セキュアでない) または TLS(セキュアな TCP) 接続を使用できます。



注記

または、設定マップを使用して **転送** プロトコルを使用してログを転送することもできます。ただし、この方法は OpenShift Container Platform では非推奨となり、今後のリリースで取り除かれます。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

手順

1. 以下のように **ClusterLogForwarder** CR YAML ファイルを作成します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
```

```
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: fluentd-server-secure (3)
     type: fluentdForward (4)
     url: 'tls://fluentdserver.security.example.com:24224' (5)
     secret: (6)
       name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  pipelines:
   - name: forward-to-fluentd-secure (7)
     inputRefs: (8)
     - application
     - audit
     outputRefs:
     - fluentd-server-secure (9)
     - default (10)
     parse: json (11)
     labels:
       clusterId: "C1234" (12)
   - name: forward-to-fluentd-insecure (13)
     inputRefs:
     - infrastructure
     outputRefs:
     - fluentd-server-insecure
     labels:
       clusterId: "C1234"
```

1. **ClusterLogForwarder** CR の名前は **instance** である必要があります。
2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. 出力の名前を指定します。
4. **fluentdForward** タイプを指定します。
5. 外部 Fluentd インスタンスの URL およびポートを有効な絶対 URL として指定します。**tcp** (セキュアでない) プロトコルまたは **tls** (セキュアな TCP) プロトコルを使用できます。CIDR アノテーションを使用するクラスター全体のプロキシーが有効になっている場合、出力は IP アドレスではなくサーバー名または FQDN である必要があります。
6. **tls** 接頭辞を使用する場合、TLS 通信のエンドポイントに必要なシークレットの名前を指定する必要があります。シークレットは **openshift-logging** プロジェクトに存在し、**tls.crt**、**tls.key**、および **ca-bundle.crt** のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。
7. オプション。パイプラインの名前を指定します。
8. パイプラインを使用して、転送する必要のあるログタイプを指定します (**application**、**infrastructure**、または **audit**)。
9. ログを転送するためにそのパイプラインで使用する出力を指定します。

10 オプション。ログを内部 Elasticsearch インスタンスに転送するために **default** 出力を指定します。

11 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。

12 オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

13 オプション: サポートされるタイプの他の外部ログアグリゲーターにログを転送するように複数の出力を設定します。

- オプション。パイプラインを説明する名前。
- **inputRefs** は、そのパイプラインを使用して転送するログタイプです (**application**、**infrastructure**、または **audit**)。
- **outputRefs** は使用する出力の名前です。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

2. CR オブジェクトを作成します。

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

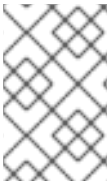
Red Hat OpenShift Logging Operator は Fluentd Pod を再デプロイします。Pod が再デプロイされない場合、強制的に再デプロイするために Fluentd Pod を削除できます。

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

7.5. SYSLOG プロトコルを使用したログの転送

syslog RFC3164 または RFC5424 プロトコルを使用して、デフォルトの Elasticsearch ログストアの代わり、またはこれに加えてプロトコルを受け入れるように設定された外部ログアグリゲーターにログのコピーを送信できます。syslog サーバーなど、外部ログアグリゲーターを OpenShift Container Platform からログを受信するように設定する必要があります。

syslog プロトコルを使用してログ転送を設定するには、syslog サーバーに対する 1 つ以上の出力およびそれらの出力を使用するパイプラインと共に **ClusterLogForwarder** カスタムリース (CR) を作成します。syslog 出力では、UDP、TCP、または TLS 接続を使用できます。



注記

または、設定マップを使用して **syslog** RFC3164 プロトコルを使用してログを転送することもできます。ただし、この方法は OpenShift Container Platform では非推奨となり、今後のリリースで取り除かれます。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

手順

1. 以下のように **ClusterLogForwarder** CR YAML ファイルを作成します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: rsyslog-east (3)
     type: syslog (4)
     syslog: (5)
       facility: local0
       rfc: RFC3164
       payloadKey: message
       severity: informational
     url: 'tls://rsyslogserver.east.example.com:514' (6)
     secret: (7)
        name: syslog-secret
   - name: rsyslog-west
     type: syslog
     syslog:
      appName: myapp
      facility: user
      msgID: mymsg
      procID: myproc
      rfc: RFC5424
      severity: debug
     url: 'udp://rsyslogserver.west.example.com:514'
  pipelines:
   - name: syslog-east (8)
     inputRefs: (9)
     - audit
     - application
     outputRefs: (10)
     - rsyslog-east
     - default (11)
     parse: json (12)
     labels:
       secure: "true" (13)
       syslog: "east"
   - name: syslog-west (14)
     inputRefs:
     - infrastructure
     outputRefs:
     - rsyslog-west
     - default
     labels:
       syslog: "west"
```

(1) **ClusterLogForwarder** CR の名前は **instance** である必要があります。

2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. 出力の名前を指定します。
4. **syslog** タイプを指定します。
5. オプション。以下に一覧表示されている syslog パラメーターを指定します。
6. 外部 syslog インスタンスの URL およびポートを指定します。**udp** (セキュアでない)、**tcp** (セキュアでない) プロトコル、または **tls** (セキュアな TCP) プロトコルを使用できます。CIDR アノテーションを使用するクラスター全体のプロキシーが有効になっている場合、出力は IP アドレスではなくサーバー名または FQDN である必要があります。
7. **tls** 接頭辞を使用する場合、TLS 通信のエンドポイントに必要なシークレットの名前を指定する必要があります。シークレットは **openshift-logging** プロジェクトに存在し、**tls.crt**、**tls.key**、および **ca-bundle.crt** のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。
8. オプション: パイプラインの名前を指定します。
9. パイプラインを使用して、転送する必要のあるログタイプを指定します (**application**、**infrastructure**、または **audit**)。
10. ログを転送するためにそのパイプラインで使用する出力を指定します。
11. オプション: ログを内部 Elasticsearch インスタンスに転送するために **default** 出力を指定します。
12. 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。
13. オプション: 文字列。ログに追加する 1 つまたは複数のラベル。"true" などの引用値は、ブール値としてではなく、文字列値として認識されるようにします。
14. オプション: サポートされるタイプの他の外部ログアグリゲーターにログを転送するように複数の出力を設定します。

    - オプション。パイプラインを説明する名前。
    - **inputRefs** は、そのパイプラインを使用して転送するログタイプです (**application**、**infrastructure**、または **audit**)。
    - **outputRefs** は使用する出力の名前です。
    - オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

2. CR オブジェクトを作成します。

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

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

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

-

7.5.1. syslog パラメーター

syslog 出力には、以下を設定できます。詳細は、syslog の RFC3164 または RFC5424 RFC を参照してください。

- facility: syslog ファシリティー。値には 10 進数の整数または大文字と小文字を区別しないキーワードを使用できます。
  - カーネルメッセージの場合、**0** または **kern**
  - ユーザーレベルのメッセージの場合、**1** または **user**。デフォルトです。
  - メールシステムの場合、**2** または **mail**
  - システムデーモンの場合、**3** または **daemon**
  - セキュリティー/認証メッセージの場合、**4** または **auth**
  - syslogd によって内部に生成されるメッセージの場合、**5** または **syslog**
  - ラインプリンターサブシステムの場合、**6** または **lpr**
  - ネットワーク news サブシステムの場合、**7** または **news**
  - UUCP サブシステムの場合、**8** または **uucp**
  - クロックデーモンの場合、**9** または **cron**
  - セキュリティー認証メッセージの場合、**10** または **authpriv**
  - FTP デーモンの場合、**11** または **ftp**
  - NTP サブシステムの場合、**12** または **ntp**
  - syslog 監査ログの場合、**13** または **security**
  - syslog アラートログの場合、**14** または **console**
  - スケジューリングデーモンの場合、**15** または **solaris-cron**
  - ローカルに使用される facility の場合、**16–23** または **local0** – **local7**
- オプション。**payloadKey**: syslog メッセージのペイロードとして使用するレコードフィールド。

  注記

  payloadKey パラメーターを設定すると、他のパラメーターが syslog に転送されなくなります。

- rfc: syslog を使用してログを送信するために使用される RFC。デフォルトは RFC5424 です。
- severity: 送信 syslog レコードに設定される syslog の重大度。値には 10 進数の整数または大文字と小文字を区別しないキーワードを使用できます。
  - システムが使用不可であることを示すメッセージの場合、**0** または **Emergency**

- 即時にアクションを実行する必要があることを示すメッセージの場合、**1** または **Alert**
- 重大な状態を示すメッセージの場合、**2** または **Critical**
- エラーの状態を示すメッセージの場合、**3** または **Error**
- 警告状態を示すメッセージの場合は、**4** または **Warning**
- 正常であるが重要な状態を示すメッセージの場合は、**5** または **Notice**
- 情報を提供するメッセージの場合は、**6** または **Informational**
- デバッグレベルのメッセージを示唆するメッセージの場合、**7** または **Debug**。デフォルトです。

- tag: タグは、syslog メッセージでタグとして使用するレコードフィールドを指定します。
- trimPrefix: 指定された接頭辞をタグから削除します。

7.5.2. 追加の RFC5424 syslog パラメーター

以下のパラメーターは RFC5424 に適用されます。

- appName: APP-NAME は、ログを送信したアプリケーションを識別するフリーテキストの文字列です。**RFC5424** に対して指定する必要があります。
- msgID: MSGID は、メッセージのタイプを識別するフリーテキスト文字列です。**RFC5424** に対して指定する必要があります。
- procID: PROCID はフリーテキスト文字列です。値が変更される場合は、syslog レポートが中断していることを示します。**RFC5424** に対して指定する必要があります。

7.6. ログの KAFKA ブローカーへの転送

デフォルトの Elasticsearch ログストアに加えて、またはこの代わりに外部の Kafka ブローカーにログを転送できます。

外部 Kafka インスタンスへのログ転送を設定するには、そのインスタンスへの出力および出力を使用するパイプラインで **ClusterLogForwarder** カスタムリソース (CR) を作成します。出力に特定の Kafka トピックを追加するか、デフォルトを使用できます。Kafka の出力は TCP(セキュアでない) または TLS(セキュアな TCP) 接続を使用できます。

手順

1. 以下のように **ClusterLogForwarder** CR YAML ファイルを作成します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: app-logs 3
     type: kafka 4
```

```
    url: tls://kafka.example.devlab.com:9093/app-topic 5
    secret:
      name: kafka-secret 6
  - name: infra-logs
    type: kafka
    url: tcp://kafka.devlab2.example.com:9093/infra-topic 7
  - name: audit-logs
    type: kafka
    url: tls://kafka.qelab.example.com:9093/audit-topic
    secret:
       name: kafka-secret-qe
pipelines:
  - name: app-topic 8
    inputRefs: 9
    - application
    outputRefs: 10
    - app-logs
    parse: json 11
    labels:
      logType: "application" 12
  - name: infra-topic 13
    inputRefs:
    - infrastructure
    outputRefs:
    - infra-logs
    labels:
      logType: "infra"
  - name: audit-topic
    inputRefs:
    - audit
    outputRefs:
    - audit-logs
    - default 14
    labels:
      logType: "audit"
```

1. **ClusterLogForwarder** CR の名前は **instance** である必要があります。
2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. 出力の名前を指定します。
4. **kafka** タイプを指定します。
5. Kafka ブローカーの URL およびポートを有効な絶対 URL として指定し、オプションで特定のトピックで指定します。**tcp** (セキュアでない) プロトコルまたは **tls** (セキュアな TCP) プロトコルを使用できます。CIDR アノテーションを使用するクラスター全体のプロキシーが有効になっている場合、出力は IP アドレスではなくサーバー名または FQDN である必要があります。
6. **tls** 接頭辞を使用する場合、TLS 通信のエンドポイントに必要なシークレットの名前を指定する必要があります。シークレットは **openshift-logging** プロジェクトに存在し、**tls.crt**、**tls.key**、および **ca-bundle.crt** のキーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。

7 オプション: 非セキュアな出力を送信するには、URL の前に **tcp** の接頭辞を使用します。また、この出力の **secret** キーとその **name** を省略します。

8 オプション: パイプラインの名前を指定します。

9 パイプラインを使用して、転送する必要のあるログタイプを指定します (**application**、**infrastructure**、または **audit**)。

10 ログを転送するためにそのパイプラインで使用する出力を指定します。

11 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。

12 オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

13 オプション: サポートされるタイプの他の外部ログアグリゲーターにログを転送するように複数の出力を設定します。

- オプション。パイプラインを説明する名前。
- **inputRefs** は、そのパイプラインを使用して転送するログタイプです (**application**、**infrastructure**、または **audit**)。
- **outputRefs** は使用する出力の名前です。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

14 オプション: ログを内部 Elasticsearch インスタンスに転送するために **default** を指定します。

2. オプション: 1 つの出力を複数の kafka ブローカーに転送するには、以下の例のように kafka ブローカーの配列を指定します。

```
...
spec:
  outputs:
  - name: app-logs
    type: kafka
    secret:
      name: kafka-secret-dev
    kafka: 1
      brokers: 2
        - tls://kafka-broker1.example.com:9093/
        - tls://kafka-broker2.example.com:9093/
      topic: app-topic 3
...
```

1 **brokers** および **topic** キーを持つ **kafka** キーを指定します。

2 **brokers** キーを指定して、1 つ以上のブローカーを指定します。

3 **topic** キーを使用して、ログを受信するターゲットトピックを指定します。

3. CR オブジェクトを作成します。

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

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

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

7.7. 特定のプロジェクトからのアプリケーションログの転送

クラスターログフォワーダーを使用して、外部ログアグリゲーターに、特定のプロジェクトからアプリケーションログのコピーを送信できます。これは、デフォルトの Elasticsearch ログストアの代わりに、またはこれに加えてデフォルトの Elasticsearch ログストアを使用して実行できます。また、外部ログアグリゲーターを OpenShift Container Platform からログデータを受信できるように設定する必要もあります。

アプリケーションログのプロジェクトからの転送を設定するには、プロジェクトから少なくとも 1 つの入力で **ClusterLogForwarder** カスタムリソース (CR) を作成し、他のログアグリゲーターのオプション出力、およびそれらの入出力を使用するパイプラインを作成します。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

手順

1. 以下のように **ClusterLogForwarder** CR YAML ファイルを作成します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: fluentd-server-secure 3
     type: fluentdForward 4
     url: 'tls://fluentdserver.security.example.com:24224' 5
     secret: 6
        name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: 7
   - name: my-app-logs
     application:
        namespaces:
        - my-project
  pipelines:
   - name: forward-to-fluentd-insecure 8
     inputRefs: 9
```

```
    - my-app-logs
    outputRefs: 10
    - fluentd-server-insecure
    parse: json 11
    labels:
      project: "my-project" 12
  - name: forward-to-fluentd-secure 13
    inputRefs:
    - application
    - audit
    - infrastructure
    outputRefs:
    - fluentd-server-secure
    - default
    labels:
      clusterId: "C1234"
```

1. **ClusterLogForwarder** CR の名前は **instance** である必要があります。
2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. 出力の名前を指定します。
4. 出力タイプ **elasticsearch**、**fluentdForward**、**syslog**、または **kafka** を指定します。
5. 外部ログアグリゲーターの URL およびポートを有効な絶対 URL として指定します。CIDR アノテーションを使用するクラスター全体のプロキシーが有効になっている場合、出力は IP アドレスではなくサーバー名または FQDN である必要があります。
6. **tls** 接頭辞を使用する場合、TLS 通信のエンドポイントに必要なシークレットの名前を指定する必要があります。シークレットは **openshift-logging** プロジェクトに存在し、**tls.crt**、 **tls.key**、 および **ca-bundle.crt** キーが含まれる必要があります。これらは、それぞれが表す証明書を参照します。
7. 指定されたプロジェクトからアプリケーションログをフィルターするための入力の設定。
8. 入力を使用してプロジェクトアプリケーションログを外部 Fluentd インスタンスに送信するためのパイプラインの設定。
9. **my-app-logs** 入力。
10. 使用する出力の名前。
11. 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。
12. オプション: 文字列。ログに追加する 1 つまたは複数のラベル。
13. ログを他のログアグリゲーターに送信するためのパイプラインの設定。
    - オプション: パイプラインの名前を指定します。
    - パイプラインを使用して、転送する必要のあるログタイプを指定します (**application** **infrastructure**、または **audit**)。

- ログを転送するためにそのパイプラインで使用する出力を指定します。
- オプション: ログを内部 Elasticsearch インスタンスに転送するために **default** 出力を指定します。
- オプション: 文字列。ログに追加する 1 つまたは複数のラベル。

2. CR オブジェクトを作成します。

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

7.8. 特定の POD からのアプリケーションログの転送

クラスター管理者は、Kubernetes Pod ラベルを使用して特定の Pod からログデータを収集し、これをログコレクターに転送できます。

アプリケーションがさまざまな namespace の他の Pod と共に実行される Pod で設定されるとします。これらの Pod にアプリケーションを識別するラベルがある場合、それらのログデータを収集し、特定のログコレクターに出力できます。

Pod ラベルを指定するには、1 つ以上の **matchLabels** のキー/値のペアを使用します。複数のキー/値のペアを指定する場合、Pod は選択されるそれらすべてに一致する必要があります。

手順

1. **ClusterLogForwarder** カスタムリソース (CR) YAML ファイルを作成します。
2. YAML ファイルで、以下の例が示すように **inputs[].name.application.selector.matchLabels** の下で単純な等価ベース (Equality-based) のセレクターを使用して Pod ラベルを指定します。

ClusterLogForwarder CR YAML ファイルのサンプル

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  pipelines:
    - inputRefs: [ myAppLogData ] 3
      outputRefs: [ default ] 4
      parse: json 5
  inputs: 6
    - name: myAppLogData
      application:
        selector:
          matchLabels: 7
            environment: production
            app: nginx
        namespaces: 8
        - app1
        - app2
```

```
outputs: 9
 - default
 ...
```

1. **ClusterLogForwarder** CR の名前は **instance** である必要があります。
2. **ClusterLogForwarder** CR の namespace は **openshift-logging** である必要があります。
3. **inputs[].name** から 1 つ以上のコンマ区切りの値を指定します。
4. **outputs[]** から 1 つ以上のコンマ区切りの値を指定します。
5. 必要に応じて、構造化された JSON ログエントリーを **structured** フィールドの JSON オブジェクトとして転送します。ログエントリーに有効な構造化された JSON が含まれる必要があります。そうでない場合は、OpenShift Logging は **構造化** フィールドを削除し、代わりにログエントリーをデフォルトのインデックス **app-00000x** に送信します。
6. Pod ラベルの一意のセットを持つ各アプリケーションの一意の **inputs[].name** を定義します。
7. 収集するログデータを持つ Pod ラベルのキー/値のペアを指定します。キーだけではなく、キーと値の両方を指定する必要があります。Pod を選択するには、Pod はすべてのキーと値のペアと一致する必要があります。
8. オプション: namespace を 1 つ以上指定します。
9. ログデータを転送する 1 つ以上の出力を指定します。ここで表示されるオプションの **default** 出力はログデータを内部 Elasticsearch インスタンスに送信します。

3. オプション: ログデータの収集を特定の namespace に制限するには、前述の例のように **inputs[].name.application.namespaces** を使用します。
4. オプション: 異なる Pod ラベルを持つ追加のアプリケーションから同じパイプラインにログデータを送信できます。
   a. Pod ラベルの一意の組み合わせごとに、表示されるものと同様の追加の **inputs[].name** セクションを作成します。
   b. このアプリケーションの Pod ラベルに一致するように、**selectors** を更新します。
   c. 新規の **inputs[].name** 値を **inputRefs** に追加します。以下に例を示します。

      ```
      - inputRefs: [ myAppLogData, myOtherAppLogData ]
      ```

5. CR オブジェクトを作成します。

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

関連情報

- Kubernetes の **matchLabels** についての詳細は、セットベースの要件をサポートするリソースについて参照してください。

7.9. レガシー FLUENTD メソッドを使用したログの転送

Fluentd **転送** プロトコルを使用して、設定ファイルおよび設定マップを作成して、ログを OpenShift Container Platform クラスター外の宛先に送信することができます。外部ログアグリゲーターを OpenShift Container Platform からログデータを受信するように設定する必要があります。

重要

ログ転送のこの方法は、OpenShift Container Platform では非推奨となり、今後のリリースでは取り除かれます。

Fluentd **転送** プロトコルを使用してログを送信するには、外部のログアグリゲーターを参照する **secure-forward.conf** という設定ファイルを作成します。次に、そのファイルを使用して OpenShift Container Platform がログの転送時に使用する **openshift-logging** プロジェクトの **secure-forward** という設定マップを作成します。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

Fluentd 設定ファイルのサンプル

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key "fluent-receiver"
  </security>
  transport tls
  tls_verify_hostname false
  tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
  <buffer>
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "1024"
    chunk_limit_size "1m"
    flush_interval "5s"
    flush_at_shutdown "false"
    flush_thread_count "2"
    retry_max_interval "300"
    retry_forever true
    overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'throw_exception'}"
  </buffer>
  <server>
    host fluent-receiver.example.com
    port 24224
  </server>
</store>
```

手順

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

1. **secure-forward** という名前の設定ファイルを作成し、**<store>** スタンザ内に以下のようなパラメーターを指定します。

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key <key> 1
  </security>
  transport tls 2
  tls_verify_hostname <value> 3
  tls_cert_path <path_to_file> 4
  <buffer> 5
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }"
    chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }"
    flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
    flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
    flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
    retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
    retry_forever true
  </buffer>
  <server>
    name 6
    host 7
    hostlabel 8
    port 9
  </server>
  <server> 10
    name
    host
  </server>
```

1. ノード間で共有キーを入力します。
2. **tls** を指定して TLS 検証を有効にします。
3. サーバー証明書のホスト名を確認するには **true** に設定します。サーバー証明書のホスト名を無視するには、**false** に設定します。
4. プライベート CA 証明書ファイルへのパスを **/etc/ocp-forward/ca_cert.pem** として指定します。
5. 必要に応じて Fluentd バッファーパラメーター を指定します。
6. オプションで、このサーバーの名前を入力します。
7. サーバーのホスト名または IP を指定します。
8. サーバーのホストラベルを指定します。
9. サーバーのポートを指定します。
10. オプションで、サーバーを追加します。2 つ以上のサーバーを指定する場合、**forward** はこれらのサーバーノードをラウンドロビン順で使用します。

相互 TLS (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その他の設定に関する情報として Fluentd ドキュメント を参照してください。

2. 設定ファイルから **openshift-logging** プロジェクトに **secure-forward** という名前の設定マップを作成します。

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

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

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

7.10. レガシー SYSLOG メソッドを使用したログの転送

syslog RFC3164 プロトコルを使用して、設定ファイルおよび設定マップを作成して、ログを OpenShift Container Platform クラスター外の宛先に送信することができます。syslog サーバーなど、外部ログアグリゲーターを OpenShift Container Platform からログを受信するように設定する必要があります。

重要

ログ転送のこの方法は、OpenShift Container Platform では非推奨となり、今後のリリースでは取り除かれます。

syslog プロトコルには、以下の 2 つのバージョンがあります。

- **out_syslog**: UDP で通信するバッファーなしの実装は、データをバッファーせずに結果を即時に書き込みます。
- **out_syslog_buffered**: バッファーの実装は TCP で通信し、データをいくつかのチャンクにバッファーリングします。

syslog プロトコルを使用してログを送信するには、ログを転送するために必要な情報を使って **syslog.conf** という設定ファイルを作成します。次に、そのファイルを使用して OpenShift Container Platform がログの転送時に使用する **openshift-logging** プロジェクトの **syslog** という設定マップを作成します。

前提条件

- 指定されたプロトコルまたは形式を使用してロギングデータを受信するように設定されたロギングサーバーが必要です。

syslog 設定ファイルのサンプル

```
<store>
@type syslog_buffered
remote_syslog rsyslogserver.example.com
port 514
hostname ${hostname}
remove_tag_prefix tag
facility local0
severity info
```

```
use_record true
payload_key message
rfc 3164
</store>
```

以下の **syslog** パラメーターを設定できます。詳細は、syslog の RFC3164 を参照してください。

- facility: syslog ファシリティー。値には 10 進数の整数または大文字と小文字を区別しないキーワードを使用できます。
  - カーネルメッセージの場合、**0** または **kern**
  - ユーザーレベルのメッセージの場合、**1** または **user**。デフォルトです。
  - メールシステムの場合、**2** または **mail**
  - システムデーモンの場合、**3** または **daemon**
  - セキュリティー/認証メッセージの場合、**4** または **auth**
  - syslogd によって内部に生成されるメッセージの場合、**5** または **syslog**
  - ラインプリンターサブシステムの場合、**6** または **lpr**
  - ネットワーク news サブシステムの場合、**7** または **news**
  - UUCP サブシステムの場合、**8** または **uucp**
  - クロックデーモンの場合、**9** または **cron**
  - セキュリティー認証メッセージの場合、**10** または **authpriv**
  - FTP デーモンの場合、**11** または **ftp**
  - NTP サブシステムの場合、**12** または **ntp**
  - syslog 監査ログの場合、**13** または **security**
  - syslog アラートログの場合、**14** または **console**
  - スケジューリングデーモンの場合、**15** または **solaris-cron**
  - ローカルに使用される facility の場合、**16–23** または **local0** – **local7**
- payloadKey: syslog メッセージのペイロードとして使用するレコードフィールド。
- rfc: syslog を使用してログを送信するために使用される RFC。
- severity: 送信 syslog レコードに設定される syslog の重大度。値には 10 進数の整数または大文字と小文字を区別しないキーワードを使用できます。
  - システムが使用不可であることを示すメッセージの場合、**0** または **Emergency**
  - 即時にアクションを実行する必要があることを示すメッセージの場合、**1** または **Alert**
  - 重大な状態を示すメッセージの場合、**2** または **Critical**
  - エラーの状態を示すメッセージの場合、**3** または **Error**

- 警告状態を示すメッセージの場合は、**4** または **Warning**
- 正常であるが重要な状態を示すメッセージの場合は、**5** または **Notice**
- 情報を提供するメッセージの場合は、**6** または **Informational**
- デバッグレベルのメッセージを示唆するメッセージの場合、**7** または **Debug**。デフォルトです。

- tag: syslog メッセージでタグとして使用するレコードフィールド。
- trimPrefix: タグから削除する接頭辞。

手順

OpenShift Container Platform をレガシーの設定方法を使用してログを転送できるように設定するには、以下を実行します。

1. **syslog.conf** という名前の設定ファイルを作成し、**<store>** スタンザ内に以下のようなパラメーターを指定します。

```
<store>
@type <type> 1
remote_syslog <syslog-server> 2
port 514 3
hostname ${hostname}
remove_tag_prefix <prefix> 4
facility <value>
severity <value>
use_record <value>
payload_key message
rfc 3164 5
</store>
```

1. 使用するプロトコル (**syslog** または **syslog_buffered** のいずれか) を指定します。
2. syslog サーバーの FQDN または IP アドレスを指定します。
3. syslog サーバーのポートを指定します。
4. オプション: 以下のように適切な syslog パラメーターを指定します。
   - syslog 接頭辞から指定された **tag** フィールドを削除するためのパラメーター。
   - 指定されたフィールドを syslog キーとして設定するパラメーター。
   - syslog ログファシリティーまたはソースを指定するパラメーター。
   - syslog ログの重大度を指定するパラメーター。
   - レコードの重大度およびファシリティーを使用するためのパラメーター (ある場合)。**true** の場合、**container_name**、**namespace_name**、および **pod_name** は、出力の内容に組み込まれます。
   - syslog メッセージのペイロードを設定するためのキーの指定に使用するパラメーター。デフォルトは **message** に設定されます。

5 レガシーの syslog メソッドでは、**rfc** 値に **3164** を指定する必要があります。

2. 設定ファイルから **openshift-logging** プロジェクトに **syslog** という名前の設定マップを作成します。

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

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

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

第8章 JSON ロギングの有効化

ログ転送 API を設定して、構造化されたオブジェクトに対して JSON 文字列を解析できます。

8.1. JSON ログの解析

JSON ログなどのログは、通常 **message** フィールド内の文字列として表されます。これにより、JSON ドキュメント内の特定のフィールドをクエリーすることが困難になります。OpenShift Logging のログ転送 API を使用すると、JSON ログを構造化オブジェクトに解析し、それらを OpenShift Logging が管理する Elasticsearch またはログ転送 API でサポートされる他のサードパーティーシステムに転送できます。

以下の構造化された JSON ログエントリーがあると想定して、これがどのように機能するか説明します。

構造化された JSON ログエントリーの例

```
{"level":"info","name":"fred","home":"bedrock"}
```

通常、**ClusterLogForwarder** カスタムリソース (CR) は、そのログエントリーを **message** フィールドに転送します。**message** フィールドには、以下の例のように JSON ログエントリーと同等の JSON 引用符で囲まれた文字列が含まれます。

message フィールドの例

```
{"message":"{\"level\":\"info\",\"name\":\"fred\",\"home\":\"bedrock\"",
 "more fields..."}
```

JSON ログの解析を有効にするには、以下の例のように、**parse: json** を **ClusterLogForwarder** CR のパイプラインに追加します。

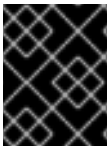
parse: json を示すスニペット例

```
pipelines:
- inputRefs: [ application ]
  outputRefs: myFluentd
  parse: json
```

parse: json を使用して JSON ログの解析を有効にすると、以下の例のように CR は **構造化** フィールドに JSON-structured ログエントリーをコピーします。元の **message** フィールドは変更されません。

構造化された JSON ログエントリーを含む 構造化された 出力例

```
{"structured": { "level": "info", "name": "fred", "home": "bedrock" },
 "more fields..."}
```



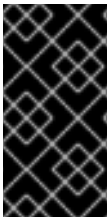
重要

ログエントリーに有効な構造化された JSON が含まれていない場合に、**構造化された** フィールドはなくなります。

特定のロギングプラットフォームの JSON ログの解析を有効にするには、ログのサードパーティーシステムへの転送 を参照してください。

8.2. ELASTICSEARCH の JSON ログデータの設定

JSON ログが複数のスキーマに従う場合は、それらを 1 つのインデックスに保存すると、タイプの競合やカーディナリティーの問題が発生する可能性があります。これを回避するには、1 つの出力定義に、各スキーマをグループ化するように **ClusterLogForwarder** カスタムリソース (CR) を設定する必要があります。これにより、各スキーマが別のインデックスに転送されます。



重要

JSON ログを OpenShift Logging によって管理されるデフォルトの Elasticsearch インスタンスに転送する場合に、設定に基づいて新規インデックスが生成されます。インデックスが多すぎることが原因のパフォーマンスの問題を回避するには、共通のスキーマに標準化して使用できるスキーマの数を保持することを検討してください。

構造化タイプ

ClusterLogForwarder CR で以下の構造タイプを使用し、Elasticsearch ログストアのインデックス名を作成できます。

- **structuredTypeKey** (string, optional) は、メッセージフィールドの名前です。このフィールドの値 (ある場合) はインデックス名の作成に使用されます。
  - **kubernetes.labels.<key>** は、インデックス名の作成に使用される Kubernetes pod ラベルの値です。
  - **openshift.labels.<key>** は、インデックス名の作成に使用される **ClusterLogForwarder** CR の **pipeline.label.<key>**; 要素です。
  - **kubernetes.container_name** はコンテナー名を使用してインデックス名を作成します。
- **structuredTypeName**:(文字列、オプション) **structuredTypeKey** が設定されておらず、そのキーが存在しない場合、OpenShift Logging は **structuredTypeName** の値を構造化型として使用します。**structuredTypeKey** and **structuredTypeName** の両方を使用する場合に、**structuredTypeName** は、構造化された **TypeKey** のキーが JSON ログデータにない場合にフォールバックインデックス名を指定します。

注記

structuredTypeKey の値を Log Record Fields トピックに記載されている任意のフィールドに設定できますが、構造タイプの前に来る一覧に最も便利なフィールドが表示されます。

structuredTypeKey: kubernetes.labels.<key> の例

以下と仮定します。

- クラスターが、apache および google という 2 つの異なる形式で JSON ログを生成するアプリケーション Pod を実行している。
- ユーザーはこれらのアプリケーション Pod に **logFormat=apache** と **logFormat=google** のラベルを付ける。
- 以下のスニペットを **ClusterLogForwarder** CR YAML ファイルで使用する。

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: kubernetes.labels.logFormat 1
    structuredTypeName: nologformat
pipelines:
- inputRefs: <application>
  outputRefs: default
  parse: json 2
```

1 Kubernetes **logFormat** ラベルで形成される key-value ペアの値を使用します。

2 JSON ログの解析を有効にします。

この場合、以下の構造化ログレコードが **app-apache-write** インデックスに送信されます。

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "apache", ...}}
}
```

また、以下の構造化ログレコードは **app-google-write** インデックスに送信されます。

```
{
  "structured":{"name":"wilma","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "google", ...}}
}
```

A structuredTypeKey: openshift.labels.<key> の例

以下のスニペットを **ClusterLogForwarder** CR YAML ファイルで使用すると仮定します。

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: openshift.labels.myLabel 1
    structuredTypeName: nologformat
pipelines:
 - name: application-logs
   inputRefs:
   - application
   - audit
   outputRefs:
   - elasticsearch-secure
   - default
   parse: json
   labels:
     myLabel: myValue 2
```

1 OpenShift **myLabel** ラベルによって形成されるキーと値のペアの値を使用します。

2 **myLabel** 要素は、文字列の値 **myValue** を構造化ログレコードに提供します。

この場合、以下の構造化ログレコードが **app-myValue-write** インデックスに送信されます。

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "openshift":{"labels":{"myLabel": "myValue", ...}}
}
```

その他の考慮事項

- 構造化されたレコードの Elasticsearch **インデックス** は、構造化型の前に app-を、後ろに-write を追加して設定されます。
- 非構造化レコードは、構造化されたインデックスに送信されません。これらは、通常アプリケーション、インフラストラクチャー、または監査インデックスでインデックス化されます。
- 空でない構造化タイプがない場合は、**unstructured** レコードを **structured** フィールドなしで転送します。

過剰なインデックスで Elasticsearch を読み込まないようにすることが重要です。各アプリケーションや namespace ごとに**ではなく**、個別のログ**形式** のみに特定の構造化タイプを使用します。たとえば、ほとんどの Apache アプリケーションは、**LogApache** などの同じ JSON ログ形式と構造化タイプを使用します。

8.3. JSON ログの ELASTICSEARCH ログストアへの転送

Elasticsearch ログストアの場合、JSON ログエントリーが**異なるスキーマに従う場合**、各 JSON スキーマを 1 つの出力定義にグループ化するように **ClusterLogForwarder** カスタムリソース (CR) を設定します。これにより、Elasticsearch はスキーマごとに個別のインデックスを使用します。

重要

異なるスキーマを同じインデックスに転送するとタイプの競合やカーディナリティーの問題を引き起こす可能性があるため、データを Elasticsearch ストアに転送する前にこの設定を実行する必要があります。

インデックスが多すぎることが原因のパフォーマンスの問題を回避するには、共通のスキーマに標準化して使用できるスキーマの数を保持することを検討してください。

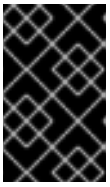
手順

1. 以下のスニペットを **ClusterLogForwarder** CR YAML ファイルに追加します。

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: <log record field>
    structuredTypeName: <name>
pipelines:
- inputRefs:
  - application
  outputRefs: default
  parse: json
```

2. オプション: 上には、Elasticsearch の JSON ログデータの設定 で説明されているように、**structuredTypeKey** を使用してログレコードフィールドのいずれかを指定します。それ以外の場合は、この行を削除します。

3. オプション: Elasticsearch の JSON ログデータの設定 で前述しているように **structuredTypeName** を使用して **<name>** を指定します。それ以外の場合は、この行を削除します。



重要

JSON ログを解析するには、**structuredTypeKey** または **structuredTypeName** か、**structuredTypeKey** と **structuredTypeName** の両方を設定する必要があります。

4. **inputRefs** の場合は、**application**、**infrastructure** または **audit** などのパイプラインを使用して、転送する必要のあるログタイプを指定します。

5. **parse: json** 要素をパイプラインに追加します。

6. CR オブジェクトを作成します。

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

Red Hat OpenShift Logging Operator は Fluentd Pod を再デプロイします。ただし、再デプロイが完了しない場合は、Fluentd Pod を削除して、強制的に再デプロイされるようにします。

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

関連情報

- ログのサードパーティーシステムへの転送

第9章 KUBERNETES イベントの収集および保存

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

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

重要

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

9.1. イベントルーターのデプロイおよび設定

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

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

前提条件

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

手順

1. イベントルーターのテンプレートを作成します。

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to OpenShift Logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount (1)
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole (2)
    apiVersion: rbac.authorization.k8s.io/v1
    metadata:
```

```
    name: event-reader
  rules:
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding (3)
  apiVersion: rbac.authorization.k8s.io/v1
  metadata:
    name: event-reader-binding
  subjects:
  - kind: ServiceAccount
    name: eventrouter
    namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap (4)
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment (5)
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
    labels:
      component: "eventrouter"
      logging-infra: "eventrouter"
      provider: "openshift"
  spec:
    selector:
      matchLabels:
        component: "eventrouter"
        logging-infra: "eventrouter"
        provider: "openshift"
    replicas: 1
    template:
      metadata:
        labels:
          component: "eventrouter"
          logging-infra: "eventrouter"
          provider: "openshift"
        name: eventrouter
      spec:
        serviceAccount: eventrouter
        containers:
          - name: kube-eventrouter
            image: ${IMAGE}
            imagePullPolicy: IfNotPresent
            resources:
```

```
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
            volumeMounts:
            - name: config-volume
              mountPath: /etc/eventrouter
          volumes:
            - name: config-volume
              configMap:
                name: eventrouter
parameters:
  - name: IMAGE 6
    displayName: Image
    value: "registry.redhat.io/openshift-logging/eventrouter-rhel8:v0.3"
  - name: CPU 7
    displayName: CPU
    value: "100m"
  - name: MEMORY 8
    displayName: Memory
    value: "128Mi"
  - name: NAMESPACE
    displayName: Namespace
    value: "openshift-logging" 9
```

1. イベントルーターの **openshift-logging** プロジェクトでサービスアカウントを作成します。
2. ClusterRole を作成し、クラスター内のイベントを監視します。
3. ClusterRoleBinding を作成し、ClusterRole をサービスアカウントにバインドします。
4. **openshift-logging** プロジェクトで設定マップを作成し、必要な **config.json** ファイルを生成します。
5. **openshift-logging** プロジェクトでデプロイメントを作成し、イベントルーター Pod を生成し、設定します。
6. **v0.3** などのタグで識別されるイメージを指定します。
7. イベントルーター Pod に割り当てるメモリーの最小量を指定します。デフォルトは **128Mi** に設定されます。
8. イベントルーター Pod に割り当てる CPU の最小量を指定します。デフォルトは **100m** に設定されます。
9. オブジェクトをインストールする **openshift-logging** プロジェクトを指定します。

2. 以下のコマンドを使用してテンプレートを処理し、これを適用します。

```
$ oc process -f <templatefile> | oc apply -n openshift-logging -f -
```

以下に例を示します。

```
$ oc process -f eventrouter.yaml | oc apply -n openshift-logging -f -
```

出力例

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

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

   a. 新規イベントルーター Pod を表示します。

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

   出力例

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

   b. イベントルーターによって収集されるイベントを表示します。

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

   以下に例を示します。

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

   出力例

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

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

第10章 OPENSHIFT LOGGING について

表10.1 Red Hat OpenShift Logging (RHOL) の OpenShift Container Platform バージョンのサポート

	4.7	4.8	4.9
RHOL 5.1	X	X	
RHOL 5.2	X	X	X
RHOL 5.3		X	X

OpenShift Container Platform 4.6 以前でクラスターロギングから OpenShift Logging 5.x にアップグレードするには、OpenShift Container Platform クラスターをバージョン 4.7 または 4.8 に更新します。次に、以下の Operator を更新します。

- Elasticsearch Operator 4.x から OpenShift Elasticsearch Operator 5.x へ
- Cluster Logging Operator 4.x から Red Hat OpenShift Logging Operator 5.x へ

以前のバージョンの OpenShift Logging から現行バージョンにアップグレードするには、OpenShift Elasticsearch Operator および Red Hat OpenShift Logging Operator を現行バージョンに更新します。

10.1. OPENSHIFT CONTAINER PLATFORM 4.6 以前でのクラスターロギングから OPENSHIFT LOGGING 5.X への更新。

OpenShift Container Platform 4.7 は以下の名前を変更します。

- **クラスターロギング** 機能は、**Red Hat OpenShift Logging** 5.x 製品になりました。
- **Cluster Logging** Operator は **Red Hat OpenShift Logging** Operator になりました。
- **Elasticsearch** Operator は **OpenShift Elasticsearch** Operator になりました。

OpenShift Container Platform 4.6 以前でクラスターロギングから OpenShift Logging 5.x にアップグレードするには、OpenShift Container Platform クラスターをバージョン 4.7 または 4.8 に更新します。次に、以下の Operator を更新します。

- Elasticsearch Operator 4.x から OpenShift Elasticsearch Operator 5.x へ
- Cluster Logging Operator 4.x から Red Hat OpenShift Logging Operator 5.x へ

重要

Red Hat OpenShift Logging Operator を更新する**前に** OpenShift Elasticsearch Operator を更新する必要があります。また、**両方の** Operator を同じバージョンに更新する必要があります。

Operator を間違った順序で更新すると、Kibana は更新されず、Kibana カスタムリソース (CR) は作成されません。この問題を回避するには、Red Hat OpenShift Logging Operator Pod を削除します。Red Hat OpenShift Logging Operator Pod が再デプロイされると、Kibana CR が作成され、Kibana が再度利用可能になります。

前提条件

- OpenShift Container Platform バージョンが 4.7 以降である。
- OpenShift Logging のステータスが正常である。
  - すべての Pod が **Ready** 状態にある。
  - Elasticsearch クラスターが正常である。
- Elasticsearch および Kibana データのバックアップが作成されている。

手順

1. OpenShift Elasticsearch Operator を更新します。
   a. Web コンソールで **Operators → Installed Operators** をクリックします。
   b. **openshift-operators-redhat** プロジェクトを選択します。
   c. **OpenShift Elasticsearch Operator** をクリックします。
   d. **Subscription → Channel** をクリックします。
   e. **Change Subscription Update Channel**ウィンドウで **5.0** または **stable-5.1** を選択し、**Save** をクリックします。
   f. 数秒待ってから **Operators → Installed Operators** をクリックします。
   OpenShift Elasticsearch Operator のバージョンが 5.x.x であることを確認します。

   Status フィールドで **Succeeded** を報告するのを待機します。
2. Cluster Logging Operator を更新します。
   a. Web コンソールで **Operators → Installed Operators** をクリックします。
   b. **openshift-logging** プロジェクトを選択します。
   c. **Cluster Logging Operator**をクリックします。
   d. **Subscription → Channel** をクリックします。
   e. **Change Subscription Update Channel**ウィンドウで **5.0** または **stable-5.1** を選択し、**Save** をクリックします。
   f. 数秒待ってから **Operators → Installed Operators** をクリックします。
   Red Hat OpenShift Logging Operator のバージョンが 5.0.x または 5.1.x であることを確認します。

   Status フィールドで **Succeeded** を報告するのを待機します。
3. ロギングコンポーネントを確認します。
   a. すべての Elasticsearch Pod が **Ready** ステータスであることを確認します。

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

   出力例

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

b. Elasticsearch クラスターが正常であることを確認します。

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

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

c. Elasticsearch cron ジョブが作成されていることを確認します。

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

```
$ oc get cronjob
```

```
NAME                     SCHEDULE       SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          56s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          56s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          56s
```

d. ログストアが 5.0 または 5.1 に更新され、インデックスが **green** であることを確認します。

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

出力に **app-00000x**、 **infra-00000x**、**audit-00000x**、**.security** インデックス が含まれることを確認します。

例10.1 緑色のステータスのインデックスを含む出力例

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                    uuid                   pri rep
docs.count docs.deleted store.size pri.store.size
green  open   infra-000008
bnBvUFEXTWi92z3zWAzieQ   3 1      222195            0       289          144
green  open   infra-000004
rtDSzoqsSl6saisSK7Au1Q   3 1      226717            0       297          148
green  open   infra-000012
RSf_kUwDSR2xEuKRZMPqZQ   3 1      227623            0       295          147
green  open   .kibana_7
1SJdCqlZTPWlIAaOUd78yg   1 1          4            0         0            0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1      248368            0       317          158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1      258799            0       337          168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1      223788            0       292          146
```

```
green open infra-000015
JRBbAbEmSMqK5X40df9HbQ 3 1 224371 0 291 145
green open .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA 3 1 9 0 0 0
green open infra-000007
llkkAVSzSOmosWTSAJM_hg 3 1 228584 0 296 148
green open infra-000005
d9BoGQdiQASsS3BBFm2iRA 3 1 227987 0 297 148
green open infra-000003 1-
goREK1QUKlQPAIVkWVaQ 3 1 226719 0 295 147
green open .security
zeT65uOuRTKZMjg_bbUc1g 1 1 5 0 0 0
green open .kibana-377444158_kubeadmin wvMhDwJkR-
mRZQO84K0gUQ 3 1 1 0 0 0
green open infra-000006 5H-
KBSXGQKiO7hdapDE23g 3 1 226676 0 295 147
green open infra-000001 eH53BQ-
bSxSWR5xYZB6lVg 3 1 341800 0 443 220
green open .kibana-6
RVp7TemSSemGJcsSUmuf3A 1 1 4 0 0 0
green open infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1 226100 0 293 146
green open app-000001
axSAFfONQDmKwatkjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLtStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002 Hz6WvINtTvKcQzw-
ewmbYg 3 1 228994 0 296 148
green open infra-000013 KR9mMFUpQl-
jraYtanyIGw 3 1 228166 0 298 148
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0
```

e. ログコレクターが 5.0 または 5.1 に更新されていることを確認します。

```
$ oc get ds fluentd -o json | grep fluentd-init
```

出力に **fluentd-init** コンテナーが含まれていることを確認します。

```
"containerName": "fluentd-init"
```

f. Kibana CRD を使用してログビジュアライザーが 5.0 または 5.1 に更新されていることを確認します。

```
$ oc get kibana kibana -o json
```

出力に **ready** ステータスの Kibana Pod が含まれることを確認します。

例10.2 準備状態にある Kibana Pod の出力例

```
[
{
"clusterCondition": {
```

```
"kibana-5fdd766ffd-nb2jj": [
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
},
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
}
]
},
"deployment": "kibana",
"pods": {
"failed": [],
"notReady": []
"ready": []
},
"replicaSets": [
"kibana-5fdd766ffd"
],
"replicas": 1
}
]
```


10.2. OPENSHIFT LOGGING の現行バージョンへの更新

OpenShift Logging を 5.x から現行バージョンに更新するには、OpenShift Elasticsearch Operator および Red Hat OpenShift Logging Operator のサブスクリプションを変更します。

重要

Red Hat OpenShift Logging Operator を更新する**前に** OpenShift Elasticsearch Operator を更新する必要があります。また、**両方の** Operator を同じバージョンに更新する必要があります。

Operator を間違った順序で更新すると、Kibana は更新されず、Kibana カスタムリソース (CR) は作成されません。この問題を回避するには、Red Hat OpenShift Logging Operator Pod を削除します。Red Hat OpenShift Logging Operator Pod が再デプロイされると、Kibana CR が作成され、Kibana が再度利用可能になります。

前提条件

- OpenShift Container Platform バージョンが 4.7 以降である。
- OpenShift Logging のステータスが正常である。
  - すべての Pod が **Ready** 状態にある。
  - Elasticsearch クラスターが正常である。

- Elasticsearch および Kibana データのバックアップが作成されている。

手順

1. OpenShift Elasticsearch Operator を更新します。
   a. Web コンソールで **Operators → Installed Operators** をクリックします。
   b. **openshift-operators-redhat** プロジェクトを選択します。
   c. **OpenShift Elasticsearch Operator** をクリックします。
   d. **Subscription → Channel** をクリックします。
   e. **Change Subscription Update Channel**ウィンドウで **stable-5.x** を選択し、**Save** をクリックします。
   f. 数秒待ってから **Operators → Installed Operators** をクリックします。
   OpenShift Elasticsearch Operator のバージョンが 5.x.x であることを確認します。

   Status フィールドで **Succeeded** を報告するのを待機します。
2. Red Hat OpenShift Logging Operator を更新します。
   a. Web コンソールで **Operators → Installed Operators** をクリックします。
   b. **openshift-logging** プロジェクトを選択します。
   c. **Red Hat OpenShift Logging Operator** をクリックします。
   d. **Subscription → Channel** をクリックします。
   e. **Change Subscription Update Channel**ウィンドウで **stable-5.x** を選択し、**Save** をクリックします。
   f. 数秒待ってから **Operators → Installed Operators** をクリックします。
   Red Hat OpenShift Logging Operator のバージョンが 5.x.x であることを確認します。

   Status フィールドで **Succeeded** を報告するのを待機します。
3. ロギングコンポーネントを確認します。
   a. すべての Elasticsearch Pod が **Ready** ステータスであることを確認します。

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

   出力例

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

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

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

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

c. Elasticsearch cron ジョブが作成されていることを確認します。

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

```
$ oc get cronjob
```

```
NAME                     SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          56s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          56s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          56s
```

d. ログストアが 5.x に更新され、インデックスが **green** であることを確認します。

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

出力に **app-00000x**、 **infra-00000x**、**audit-00000x**、**.security** インデックス が含まれることを確認します。

例10.3 緑色のステータスのインデックスを含む出力例

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                    uuid                   pri rep
docs.count docs.deleted store.size pri.store.size
green  open   infra-000008
bnBvUFEXTWi92z3zWAzieQ   3 1      222195          0       289          144
green  open   infra-000004
rtDSzoqsSl6saisSK7Au1Q   3 1      226717          0       297          148
green  open   infra-000012
RSf_kUwDSR2xEuKRZMPqZQ   3 1      227623          0       295          147
green  open   .kibana_7
1SJdCqlZTPWlIAaOUd78yg   1 1          4          0         0            0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1      248368          0       317          158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1      258799          0       337          168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1      223788          0       292          146
green  open   infra-000015
JRBbAbEmSMqK5X40df9HbQ   3 1      224371          0       291          145
green  open   .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA   3 1          9          0         0            0
green  open   infra-000007
llkkAVSzSOmosWTSAJM_hg   3 1      228584          0       296          148
green  open   infra-000005
d9BoGQdiQASsS3BBFm2iRA   3 1      227987          0       297          148
```

```
green open infra-000003 1-
goREK1QUKlQPAIVkWVaQ 3 1 226719 0 295 147
green open .security
zeT65uOuRTKZMjg_bbUc1g 1 1 5 0 0 0
green open .kibana-377444158_kubeadmin wvMhDwJkR-
mRZQO84K0gUQ 3 1 1 0 0 0
green open infra-000006 5H-
KBSXGQKiO7hdapDE23g 3 1 226676 0 295 147
green open infra-000001 eH53BQ-
bSxSWR5xYZB6lVg 3 1 341800 0 443 220
green open .kibana-6
RVp7TemSSemGJcsSUmuf3A 1 1 4 0 0 0
green open infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1 226100 0 293 146
green open app-000001
axSAFfONQDmKwatkjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLtStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002 Hz6WvINtTvKcQzw-
ewmbYg 3 1 228994 0 296 148
green open infra-000013 KR9mMFUpQl-
jraYtanyIGw 3 1 228166 0 298 148
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0
```

e. ログコレクターが 5.x に更新されていることを確認します。

```
$ oc get ds fluentd -o json | grep fluentd-init
```

出力に **fluentd-init** コンテナーが含まれていることを確認します。

```
"containerName": "fluentd-init"
```

f. Kibana CRD を使用してログビジュアライザーが 5.x に更新されていることを確認します。

```
$ oc get kibana kibana -o json
```

出力に **ready** ステータスの Kibana Pod が含まれることを確認します。

例10.4 準備状態にある Kibana Pod の出力例

```
[
{
"clusterCondition": {
"kibana-5fdd766ffd-nb2jj": [
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
},
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
```

```
"reason": "ContainerCreating",
"status": "True",
"type": ""
}
]
},
"deployment": "kibana",
"pods": {
"failed": [],
"notReady": []
"ready": []
},
"replicaSets": [
"kibana-5fdd766ffd"
],
"replicas": 1
}
]
```

第11章 クラスターダッシュボードの表示

OpenShift Container Platform Web コンソールの **Logging/Elasticsearch Nodes** および **Openshift Logging** ダッシュボードは、 Elasticsearch インスタンスや、問題の発生防止および診断に使用できる個別の Elasticsearch ノードについての詳細情報を表示します。

OpenShift Logging ダッシュボードには、クラスターリソース、ガベージコレクション、クラスターのシャード、Fluentd 統計など、クラスターレベルでの Elasticsearch インスタンスについての詳細を表示するチャートが含まれます。

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch インスタンスの詳細を表示するチャートが含まれます。これらのチャートの多くはノードレベルのものであり、これには、インデックス、シャード、リソースなどの詳細が含まれます。

注記

より詳細なデータについては、ダッシュボードの **Grafana UI** リンクをクリックして Grafana ダッシュボードを起動します。Grafana には OpenShift クラスターモニターリング が同梱されています。

11.1. ELASTISEARCH および OPENSHIFT LOGGING ダッシュボードへのアクセス

OpenShift Container Platform Web コンソールで **Logging/Elasticsearch Nodes** および **Openshift Logging** ダッシュボードを表示できます。

手順

ダッシュボードを起動するには、以下を実行します。

1. OpenShift Container Platform Web コンソールで、 **Monitoring** → **Dashboards** をクリックします。

2. **Dashboards** ページで、 **Dashboard** メニューから **Logging/Elasticsearch Nodes** または **Openshift Logging** を選択します。
   Logging/Elasticsearch Nodes ダッシュボードの場合、表示する必要のある Elasticsearch ノードを選択し、データの解像度を設定できます。

   適切なダッシュボードが表示され、データの複数のチャートが表示されます。

3. 必要に応じて、**Time Range** メニューおよび **Refresh Interval** メニューから、データを表示する別の時間の範囲またはデータのリフレッシュレートを選択します。

注記

より詳細なデータについては、**Grafana UI** リンクをクリックして Grafana ダッシュボードを起動します。

ダッシュボードチャートについての詳細は、OpenShift Logging ダッシュボードについて および Logging/Elastisearch Nodes ダッシュボード について参照してください。

11.2. OPENSHIFT LOGGING ダッシュボードについて

OpenShift Logging ダッシュボードには、クラスターレベルで Elasticsearch インスタンスの詳細を表示するチャートが含まれており、これを使用して問題を診断し、予測できます。

表11.1 OpenShift Logging チャート

メトリクス	説明
Elastic Cluster Status (Elastic Cluster のステータス)	Elasticsearch の現行ステータス: ● ONLINE: Elasticsearch インスタンスがオンラインであることを示します。 ● OFFLINE: Elasticsearch インスタンスがオフラインであることを示します。
Elastic Nodes (Elastic ノード)	Elasticsearch インスタンス内の Elasticsearch ノードの合計数。
Elastic Shards (Elastic シャード)	Elasticsearch インスタンス内の Elasticsearch シャードの合計数。
Elastic Documents (Elastic ドキュメント)	Elasticsearch インスタンス内の Elasticsearch ドキュメントの合計数。
Total Index Size on Disk (ディスク上の合計インデックスサイズ)	Elasticsearch インデックスに使用されるディスク容量の合計。
Elastic Pending Tasks (Elastic の保留中のタスク)	インデックスの作成、インデックスのマッピング、シャードの割り当て、シャードの失敗など、完了していない Elasticsearch 変更の合計数。
Elastic JVM GC time (Elastic JVM GC 時間)	JVM がクラスターでの Elasticsearch ガベージコレクション操作の実行に費した時間。
Elastic JVM GC Rate (Elastic JVM GC レート)	JVM が 1 秒ごとにガベージアクティビティーを実行する合計回数。
Elastic Query/Fetch Latency Sum (Elastic クエリー/フェッチのレイテンシーの合計)	● クエリーレイテンシー: 各 Elasticsearch 検索クエリーの実行に必要な平均時間。 ● フェッチレイテンシー: 各 Elasticsearch 検索クエリーがデータのフェッチに費す平均時間。 通常、フェッチレイテンシーの時間はクエリーレイテンシーよりも短くなります。フェッチレイテンシーが一貫して増加する場合、これはディスクの速度の低下、データの増加、または結果が多すぎる大規模な要求があることを示している可能性があります。

メトリクス	説明
Elastic Query Rate (Elastic クエリーレート)	各 Elasticsearch ノードについて 1 秒ごとに Elasticsearch インスタンスに対して実行される合計クエリー。
CPU	それぞれのコンポーネントについて表示される Elasticsearch、Fluentd、および Kibana によって使用される CPU の量。
Elastic JVM Heap Used (Elastic JVM ヒープの使用)	使用される JVM メモリーの量。正常なクラスターでは、JVM ガベージコレクションによってメモリーが解放されると、グラフは定期的な低下を示します。
Elasticsearch Disk Usage (Elasticsearch ディスクの使用)	各 Elasticsearch ノードについて Elasticsearch インスタンスによって使用されるディスク容量の合計。
File Descriptors In Use (使用中のファイル記述子)	Elasticsearch、Fluentd、および Kibana によって使用されるファイル記述子の合計数。
FluentD emit count (Fluentd の生成数)	Fluentd デフォルト出力についての 1 秒あたりの Fluentd メッセージの合計数およびデフォルト出力の再試行数。
FluentD Buffer Availability (Fluentd バッファーの可用性)	チャンクに使用できる Fluentd バッファーのパーセント。バッファーが一杯になると、Fluentd が受信するログ数を処理できないことを示す可能性があります。
Elastic rx bytes (Elastic rx バイト)	Elasticsearch が FluentD、Elasticsearch ノード、およびその他のソースから受信した合計バイト数。
Elastic Index Failure Rate (Elastic インデックス失敗率)	Elasticsearch インデックスで失敗した 1 秒あたりの合計回数。レートが高い場合は、インデックスに問題があることを示す可能性があります。
FluentD Output Error Rate (Fluentd 出力エラー率)	FluentD がログの出力に失敗する 1 秒あたりの合計回数。

11.3. LOGGING/ELASTICSEARCH ノードダッシュボードのチャート

Logging/Elasticsearch Nodes ダッシュボードには、追加の診断に使用できる Elasticsearch インスタンスの詳細を表示するチャートが含まれます。これらのチャートの多くはノードレベルのものです。

Elasticsearch ステータス

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch インスタンスのステータスに関する以下のチャートが含まれます。

表11.2 Elasticsearch ステータスフィールド

メトリクス	説明
Cluster status (クラスターステータス)	Elasticsearch の green、yellow、および red ステータスを使用する、選択された期間におけるクラスターの正常性ステータス。 ● 0: Elasticsearch インスタンスが green ステータスであることを示します。これは、すべてのシャードが割り当てられることを意味します。 ● 1: Elasticsearch インスタンスが yellow ステータスであることを示します。これは、1 つ以上のシャードのレプリカシャードが割り当てられないことを意味します。 ● 2: Elasticsearch インスタンスが red ステータスであることを示します。これは、1 つ以上のプライマリーシャードとそのレプリカが割り当てられないことを意味します。
Cluster nodes (クラスターノード)	クラスター内の Elasticsearch ノードの合計数。
Cluster data nodes (クラスターデータノード)	クラスター内の Elasticsearch データノードの数。
Cluster pending tasks (クラスターの保留中のタスク)	終了しておらず、クラスターキューで待機中のクラスター状態変更の数。たとえば、インデックスの作成、インデックスの削除、シャードの割り当てなどがあります。増加傾向は、クラスターが変更に対応できないことを示します。

Elasticsearch クラスターインデックスシャードのステータス

各 Elasticsearch インデックスは、永続化されたデータの基本単位である 1 つ以上のシャードの論理グループです。インデックスシャードには、プライマリーシャードとレプリカシャードの 2 つのタイプがあります。ドキュメントがインデックスにインデックス化されると、これはプライマリーシャードのいずれかに保存され、そのシャードのすべてのレプリカにコピーされます。プライマリーシャードの数はインデックスの作成時に指定され、この数はインデックスの有効期間に変更することはできません。レプリカシャードの数はいつでも変更できます。

インデックスシャードは、ライフサイクルフェーズまたはクラスターで発生するイベントに応じて複数の状態に切り替わります。シャードが検索およびインデックス要求を実行できる場合、シャードはアクティブになります。シャードがこれらの要求を実行できない場合、シャードは非アクティブになります。シャードが初期化、再割り当て、未割り当てなどの状態にある場合、シャードが非アクティブになる可能性があります。

インデックスシャードは、データの物理表現であるインデックスセグメントと呼ばれる数多くの小さな内部ブロックで設定されます。インデックスセグメントは、Lucene が新たにインデックス化されたデータをコミットしたときに作成される比較的小さく、イミュータブルな Lucene インデックスです。Lucene (Elasticsearch によって使用される検索ライブラリー) は、バックグラウンドでインデックスセグメントをより大きなセグメントにマージし、セグメントの合計数を低い状態に維持します。セグメントをマージするプロセスが新規セグメントが作成される速度よりも遅くなる場合、問題があることを示す可能性があります。

Lucene が検索操作などのデータ操作を実行する場合、Lucene は関連するインデックスのインデックスセグメントに対して操作を実行します。そのため、各セグメントには、メモリーにロードされ、マップされる特定のデータ構造が含まれます。インデックスマッピングは、セグメントデータ構造で使用されるメモリーに大きく影響を与える可能性があります。

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch インデックスシャードに関する以下のチャートが含まれます。

表11.3 Elasticsearch クラスターのシャードステータスのチャート

メトリクス	説明
Cluster active shards (クラスターのアクティブシャード)	クラスターにおけるアクティブなプライマリーシャードの数と、レプリカを含むシャードの合計数。シャードの数が大きくなると、クラスターのパフォーマンスが低下し始める可能性があります。
Cluster initializing shards (クラスターの初期化シャード)	クラスターのアクティブではないシャードの数。アクティブではないシャードは、初期化され、別のノードに再配置されるているシャードや、割り当てられていないシャードを指します。通常、クラスターには短期間アクティブではないシャードがあります。長期間にわたってアクティブではないシャードの数が増える場合、問題があることを示す可能性があります。
Cluster relocating shards (クラスターの再配置シャード)	Elasticsearch が新規ノードに再配置されているシャードの数。Elasticsearch は、ノードでのメモリー使用率が高い場合や新規ノードがクラスターに追加された後などの複数の理由によりノードを再配置します。
Cluster unassigned shards (クラスター未割り当てシャード)	未割り当てのシャードの数。Elasticsearch シャードは、新規インデックスの追加やノードの障害などの理由で割り当てられない可能性があります。

Elasticsearch ノードメトリクス

各 Elasticsearch ノードには、タスクの処理に使用できるリソースの量に制限があります。すべてのリソースが使用中で、Elasticsearch が新規タスクの実行を試行する場合、Elasticsearch は一部のリソースが利用可能になるまでタスクをキューに入れます。

Logging/Elasticsearch Nodes ダッシュボードには、選択されたノードのリソース使用状況に関する以下のチャートと Elasticsearch キューで待機中のタスクの数が含まれます。

表11.4 Elasticsearch ノードのメトリクスチャート

メトリクス	説明
ThreadPool tasks (ThreadPool タスク)	個別のキューの待機中のタスクの数 (タスクタイプ別に表示されます)。キュー内のタスクの長期間累積した状態は、ノードリソースの不足やその他の問題があることを示す可能性があります。

メトリクス	説明
CPU usage (CPU の使用率)	ホストコンテナーに割り当てられる CPU の合計の割合として、選択した Elasticsearch ノードによって使用される CPU の量。
Memory usage (メモリー使用量)	選択した Elasticsearch ノードによって使用されるメモリー量。
Disk usage (ディスク使用量)	選択された Elasticsearch ノードのインデックスデータおよびメタデータに使用されるディスク容量の合計。
Documents indexing rate (ドキュメントインデックス化レート)	ドキュメントが選択された Elasticsearch ノードでインデックス化されるレート。
Indexing latency (インデックス化レイテンシー)	選択された Elasticsearch ノードでドキュメントをインデックス化するのに必要となる時間。インデックス化レイテンシーは、JVM ヒープメモリーや全体の負荷などの多くの要素による影響を受ける可能性があります。レイテンシーが増加する場合、インスタンス内のリソース容量が不足していることを示します。
Search rate (検索レート)	選択された Elasticsearch ノードで実行される検索要求の数。
Search latency (検索レイテンシー)	選択された Elasticsearch ノードで検索要求を完了するのに必要となる時間。検索レイテンシーは、数多くの要因の影響を受ける可能性があります。レイテンシーが増加する場合、インスタンス内のリソース容量が不足していることを示します。
Documents count (with replicas)(ドキュメント数 (レプリカ使用))	選択された Elasticsearch ノードに保管される Elasticsearch ドキュメントの数。これには、ノードで割り当てられるプライマリーシャードとレプリカシャードの両方に保存されるドキュメントが含まれます。
Documents deleting rate (ドキュメントの削除レート)	選択された Elasticsearch ノードに割り当てられるいずれかのインデックスシャードから削除される Elasticsearch ドキュメントの数。
Documents merging rate (ドキュメントのマージレート)	選択された Elasticsearch ノードに割り当てられるインデックスシャードのいずれかでマージされる Elasticsearch ドキュメントの数。

Elasticsearch ノードフィールドデータ

Fielddata はインデックスの用語の一覧を保持する Elasticsearch データ構造であり、JVM ヒープに保持されます。fielddata のビルドはコストのかかる操作であるため、Elasticsearch は fielddata 構造をキャッシュします。Elasticsearch は、基礎となるインデックスセグメントが削除されたり、

マージされる場合や、すべての fielddata キャッシュに JVM HEAP メモリーが十分にない場合に、fielddata キャッシュをエビクトできます。

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch fielddata に関する以下のチャートが含まれます。

表11.5 Elasticsearch ノードフィールドデータチャート

メトリクス	説明
Fielddata memory size (Fielddata メモリーサイズ)	選択された Elasticsearch ノードの fielddata キャッシュに使用される JVM ヒープの量。
Fielddata evictions (Fielddata エビクション)	選択された Elasticsearch ノードから削除された fielddata 構造の数。

Elasticsearch ノードのクエリーキャッシュ

インデックスに保存されているデータが変更されない場合、検索クエリーの結果は Elasticsearch で再利用できるようにノードレベルのクエリーキャッシュにキャッシュされます。

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch ノードのクエリーキャッシュに関する以下のチャートが含まれます。

表11.6 Elasticsearch ノードのクエリーチャート

メトリクス	説明
Query cache size (クエリーキャッシュサイズ)	選択された Elasticsearch ノードに割り当てられるすべてのシャードのクエリーキャッシュに使用されるメモリーの合計量。
Query cache evictions (クエリーキャッシュエビクション)	選択された Elasticsearch ノードでのクエリーキャッシュのエビクション数。
Query cache hits (クエリーキャッシュヒット)	選択された Elasticsearch ノードでのクエリーキャッシュのヒット数。
Query cache misses (クエリーキャッシュミス)	選択された Elasticsearch ノードでのクエリーキャッシュのミス数。

Elasticsearch インデックスのスロットリング

ドキュメントのインデックスを作成する場合、Elasticsearch はデータの物理表現であるインデックスセグメントにドキュメントを保存します。同時に、Elasticsearch はリソースの使用を最適化する方法として、より小さなセグメントをより大きなセグメントに定期的にマージします。インデックス処理がセグメントをマージする機能よりも高速になる場合、マージプロセスが十分前もって終了せずに、検索やパフォーマンスに関連した問題が生じる可能性があります。この状況を防ぐために、Elasticsearch はインデックスをスロットリングします。通常、インデックスに割り当てられるスレッド数を 1 つのスレッドに減らすことで制限できます。

Logging/Elasticsearch Nodes ダッシュボードには、Elasticsearch インデックスのスロットリングについての以下のチャートが含まれます。

表11.7 インデックススロットリングチャート

メトリクス	説明
Indexing throttling (インデックスのスロットリング)	Elasticsearch が選択された Elasticsearch ノードでインデックス操作をスロットリングしている時間。
Merging throttling (マージのスロットリング)	Elasticsearch が選択された Elasticsearch ノードでセグメントのマージ操作をスロットリングしている時間。

ノード JVM ヒープの統計

Logging/Elasticsearch Nodes ダッシュボードには、JVM ヒープ操作に関する以下のチャートが含まれます。

表11.8 JVM ヒープ統計チャート

メトリクス	説明
Heap used (ヒープの使用)	選択された Elasticsearch ノードで使用される割り当て済みの JVM ヒープ領域の合計。
GC count (GC 数)	新旧のガベージコレクションによって、選択された Elasticsearch ノードで実行されてきたガベージコレクション操作の数。
GC time (GC 時間)	JVM が、新旧のガベージコレクションによって選択された Elasticsearch ノードでガベージコレクションを実行してきた時間。

第12章 ロギングのトラブルシューティング

12.1. OPENSHIFT LOGGING ステータスの表示

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

12.1.1. Red Hat OpenShift Logging Operator のステータス表示

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

2. OpenShift Logging のステータスを表示するには、以下を実行します。

   a. OpenShift Logging のステータスを取得します。

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

出力例

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

....

status: 1
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd 2
        nodes:
          fluentd-2rhqp: ip-10-0-169-13.ec2.internal
          fluentd-6fgjh: ip-10-0-165-244.ec2.internal
          fluentd-6l2ff: ip-10-0-128-218.ec2.internal
          fluentd-54nx5: ip-10-0-139-30.ec2.internal
          fluentd-flpnn: ip-10-0-147-228.ec2.internal
          fluentd-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
          - fluentd-2rhqp
          - fluentd-54nx5
          - fluentd-6fgjh
```

```
        - fluentd-6l2ff
        - fluentd-flpnn
        - fluentd-n2frh
    logstore: 3
      elasticsearchStatus:
      - ShardAllocationEnabled:  all
        cluster:
          activePrimaryShards:    5
          activeShards:           5
          initializingShards:     0
          numDataNodes:           1
          numNodes:               1
          pendingTasks:           0
          relocatingShards:       0
          status:                 green
          unassignedShards:       0
        clusterName:              elasticsearch
        nodeConditions:
          elasticsearch-cdm-mkkdys93-1:
        nodeCount:  1
        pods:
          client:
            failed:
            notReady:
            ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
          data:
            failed:
            notReady:
            ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
          master:
            failed:
            notReady:
            ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  visualization: 4
    kibanaStatus:
    - deployment: kibana
      pods:
        failed: []
        notReady: []
        ready:
        - kibana-7fb4fd4cc9-f2nls
      replicaSets:
      - kibana-7fb4fd4cc9
      replicas: 1
```

1. 出力の **status** スタンザに、クラスターステータスのフィールドが表示されます。
2. Fluentd Pod についての情報
3. Elasticsearch クラスターの健全性 (**green**、**yellow**、または **red**) などの Elasticsearch Pod についての情報
4. Kibana Pod についての情報

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

以下は、OpenShift Logging インスタンスの **Status.Nodes** セクションからの一部の状態メッセージの例です。

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

出力例

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

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

出力例

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

以下のようなステータスメッセージは、CR の Elasticsearch ノードセ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。

出力例

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

```
    Node Conditions:
      elasticsearch-cdm-mkkdys93-1:
        Last Transition Time:  2019-06-26T03:37:32Z
        Message:               0/5 nodes are available: 5 node(s) didn't match node selector.
        Reason:                Unschedulable
        Status:                True
        Type:                  Unschedulable
      elasticsearch-cdm-mkkdys93-2:
    Node Count:  2
    Pods:
      Client:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Data:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Master:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
```

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

出力例

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

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

出力例

```
Status:
  Collection:
    Logs:
      Fluentd Status:
        Daemon Set:  fluentd
        Nodes:
        Pods:
```

```
    Failed:
    Not Ready:
    Ready:
```

12.1.2. OpenShift Logging コンポーネントのステータスの表示

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

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

2. OpenShift Logging 環境のステータスを表示します。

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

出力例

```
Name:                   cluster-logging-operator

....

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

....

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

3. OpenShift Logging レプリカセットのステータスを表示します。

   a. レプリカセットの名前を取得します。

出力例

```
$ oc get replicaset
```

出力例

```
NAME                                  DESIRED   CURRENT   READY   AGE
```

```
cluster-logging-operator-574b8987df       1         1         1       159m
elasticsearch-cdm-uhr537yu-1-6869694fb    1         1         1       157m
elasticsearch-cdm-uhr537yu-2-857b6d676f   1         1         1       156m
elasticsearch-cdm-uhr537yu-3-5b6fdd8cfd   1         1         1       155m
kibana-5bd5544f87                         1         1         1       157m
```

b. レプリカセットのステータスを取得します。

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

出力例

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

....

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

....

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

12.2. ログストアのステータスの表示

OpenShift 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。

12.2.1. ログストアのステータスの表示

ログストアのステータスを表示することができます。

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

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

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

2. ステータスを表示するには、以下を実行します。

a. ログストアインスタンスの名前を取得します。

```
$ oc get Elasticsearch
```

出力例

```
NAME            AGE
elasticsearch   5h9m
```

b. ログストアのステータスを取得します。

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

以下に例を示します。

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

出力には、以下のような情報が含まれます。

出力例

```
status: 1
  cluster: 2
    activePrimaryShards: 30
    activeShards: 60
    initializingShards: 0
    numDataNodes: 3
    numNodes: 3
    pendingTasks: 0
    relocatingShards: 0
    status: green
    unassignedShards: 0
  clusterHealth: ""
  conditions: [] 3
  nodes: 4
  - deploymentName: elasticsearch-cdm-zjf34ved-1
    upgradeStatus: {}
  - deploymentName: elasticsearch-cdm-zjf34ved-2
    upgradeStatus: {}
  - deploymentName: elasticsearch-cdm-zjf34ved-3
    upgradeStatus: {}
  pods: 5
    client:
      failed: []
      notReady: []
      ready:
      - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    data:
      failed: []
      notReady: []
      ready:
      - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    master:
      failed: []
      notReady: []
      ready:
```

```
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
  shardAllocationEnabled: all
```

1. 出力の **status** スタンザに、クラスターステータスのフィールドが表示されます。
2. ログストアのステータス:
   - アクティブなプライマリーシャードの数
   - アクティブなシャードの数
   - 初期化されるシャードの数
   - ログストアデータノードの数。
   - ログストアノードの合計数。
   - 保留中のタスクの数。
   - ログストアのステータス: **green**、**red**、**yellow**。
   - 未割り当てのシャードの数。
3. ステータス状態 (ある場合)。ログストアのステータスは、Pod が配置されていない場合にスケジューラーからの理由を示します。以下の状況に関連したイベントが表示されます。
   - ログストアおよびプロキシーコンテナーの両方についてコンテナーが待機中。
   - ログストアおよびプロキシーコンテナーの両方についてコンテナーが終了している。
   - Pod がスケジュール対象外である。さらに多数の問題についての状態が表示されます。詳細は、**状態メッセージのサンプル** を参照してください。
4. **upgradeStatus** のあるクラスター内のログストアノード。
5. 'failed`、**notReady** または **ready** 状態の下に一覧表示された、クラスター内のログストアクライアント、データ、およびマスター Pod。

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

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

以下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: {}
```

以下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: {}
```

以下のステータスメッセージは、CR のログストア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
```

以下のステータスメッセージは、ログストア CR が存在しない 永続ボリューム要求 (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
```

以下のステータスメッセージは、ログストアクラスターには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、クラスター内のコントロールプレーンノード (別称マスターノード) 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
```

以下のステータスメッセージは、加えようとした変更が Elasticsearch ストレージでサポートされないことを示します。

以下に例を示します。

```
status:
  clusterHealth: green
  conditions:
    - lastTransitionTime: "2021-05-07T01:05:13Z"
      message: Changing the storage structure for a custom resource is not supported
      reason: StorageStructureChangeIgnored
      status: 'True'
      type: StorageStructureChangeIgnored
```

reason および **type** フィールドは、サポート対象外の変更のタイプを指定します。

StorageClassNameChangeIgnored
ストレージクラス名の変更がサポートされていません。

StorageSizeChangeIgnored
ストレージサイズの変更がサポートされていません。

StorageStructureChangeIgnored
一時ストレージと永続ストレージ構造間での変更がサポートされていません。

重要

ClusterLogging カスタムリソース (CR) を一時ストレージから永続ストレージに切り替えるように設定する場合に、OpenShift Elasticsearch Operator は永続ボリューム要求 (PVC) を作成しますが、永続ボリューム (PV) は作成されません。**StorageStructureChangeIgnored** ステータスを削除するには、**ClusterLogging** CR への変更を元に戻し、PVC を削除する必要があります。

12.2.2. ログストアコンポーネントのステータスの表示

数多くのログストア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
```

出力例

```
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-4vjor49p-2-6d4d7db474-q2w7z -- indices
```

出力例

```
Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -n openshift-
logging' to see all of the containers in this pod.

green  open   infra-000002                                S4QANnf1QP6NgCegfnrnbQ
3   1     119926            0       157            78
green  open   audit-000001                                8_EQx77iQCSTzFOXtxRqFw
3   1          0            0         0             0
green  open   .security                                 iDjscH7aSUGhIdq0LheLBQ   1
1          5            0         0             0
green  open   .kibana_-377444158_kubeadmin
yBywZ9GfSrKebz5gWBZbjw   3   1          1            0         0             0
green  open   infra-000001                                z6Dpe__ORgiopEpW6Yl44A
3   1     871000            0       874           436
green  open   app-000001                                  hIrazQCeSISewG3c2VIvsQ
3   1       2453            0         3             1
green  open   .kibana_1                                 JCitcBMSQxKOvIq6iQW6wg
1   1          0            0         0             0
green  open   .kibana_-1595131456_user1                        gIYFIEGRRe-
ka0W3okS-mQ   3   1          1            0         0             0
```

ログストア Pod

ログストアをホストする 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
```

出力例

```
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。

出力例

```
....
Status:             Running

....

Containers:
  elasticsearch:
    Container ID:   cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
    State:          Running
      Started:      Mon, 08 Jun 2020 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 Jun 2020 10:18:38 -0400
    Ready:          True
    Restart Count:  0

....

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

....

Events:         <none>
```

ログストレージ Pod デプロイメント設定

ログストアの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
```

出力例

```
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。

出力例

```
....
  Containers:
   elasticsearch:
    Image:      registry.redhat.io/openshift-logging/elasticsearch6-rhel8
    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>
```

ログストアのレプリカセット

ログストアのレプリカセット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。

出力例

```
....
  Containers:
   elasticsearch:
    Image:      registry.redhat.io/openshift-logging/elasticsearch6-
rhel8@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6908e7b1
c25
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

Events:         <none>
```

12.3. OPENSHIFT LOGGING アラートについて

ロギングコレクターのアラートはすべて、OpenShift Container Platform Web コンソールの Alerting UI に一覧表示されます。

12.3.1. ロギングコレクターアラートの表示

アラートは、OpenShift Container Platform Web コンソールの、Alerting UI の **Alerts** タブに表示されます。アラートは以下の状態のいずれかになります。

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

手順

OpenShift Logging およびその他の OpenShift Container Platform アラートを表示するには、以下を実行します。

1. OpenShift Container Platform コンソールで、**Monitoring → Alerting** をクリックします。
2. **Alerts** タブをクリックします。選択したフィルターに基づいてアラートが一覧表示されます。

関連情報

- Alerting UI の詳細は、Managing alerts を参照してください。

12.3.2. ロギングコレクターのアラートについて

以下のアラートはロギングコレクターによって生成されます。これらのアラートは、OpenShift Container Platform Web コンソールの Alerting UI の **Alerts** ページで表示できます。

表12.1 Fluentd Prometheus アラート

アラート	メッセージ	説明	重大度
FluentDHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	FluentD 出力エラーの数は、デフォルトでは直前の 15 分間で 10 分を超えます。	Warning
FluentdNodeDown	**Prometheus could not scrape fluentd <instance> for more than 10m.**	Fluentd は Prometheus が特定の Fluentd インスタンスを収集できなかったことを報告します。	Critical
FluentdQueueLengthIncreasing	**In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1.Current value is <value>.**	Fluentd はキューサイズが増加していることを報告しています。	Critical
FluentDVeryHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	FluentD 出力エラーの数は非常に高くなります。デフォルトでは、直前の 15 分間で 25 を超えます。	Critical

12.3.3. 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。

表12.2 アラートルール

アラート	説明	重要度
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
ElasticsearchDiskSpaceRunningLow	クラスターでは、次の 6 時間以内にディスク領域が不足することが予想されます。	Critical
ElasticsearchHighFileDescriptorUsage	クラスターでは、次の 1 時間以内にファイル記述子が不足することが予想されます。	Warning
ElasticsearchJVMHeapUseHigh	指定されたノードでの JVM ヒープの使用率が高くなっています。	アラート

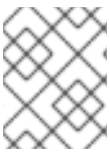
アラート	説明	重要度
ElasticsearchNodeDiskWatermarkReached	指定されたノードは、ディスクの空き容量が少ないために低基準値に達しています。シャードをこのノードに割り当てることはできません。ノードにディスク領域を追加することを検討する必要があります。	Info
ElasticsearchNodeDiskWatermarkReached	指定されたノードは、ディスクの空き容量が少ないために高基準値に達しています。一部のシャードは可能な場合に別のノードに再度割り当てられる可能性があります。ノードにディスク領域が追加されるか、またはこのノードに割り当てられる古いインデックスをドロップします。	Warning
ElasticsearchNodeDiskWatermarkReached	指定されたノードは、ディスクの空き容量が少ないために高基準値に達しています。このノードにシャードが割り当てられるすべてのインデックスは、読み取り専用ブロックになります。インデックスブロックは、ディスクの使用状況が高基準値を下回る場合に手動で解放される必要があります。	Critical
ElasticsearchJVMHeapUseHigh	指定されたノードの JVM ヒープの使用率が高すぎます。	アラート
ElasticsearchWriteRequestsRejectionJumps	Elasticsearch では、指定されたノードで書き込み拒否が増加しています。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
AggregatedLoggingSystemCPUHigh	指定されたノードのシステムで使用される CPU が高すぎます。	アラート
ElasticsearchProcessCPUHigh	指定されたノードで Elasticsearch によって使用される CPU が高すぎます。	アラート

12.4. RED HAT サポート用のロギングデータの収集

サポートケースを作成する際、ご使用のクラスターについてのデバッグ情報を Red Hat サポートに提供していただくと Red Hat のサポートに役立ちます。

must-gather ツール を使用すると、プロジェクトレベルのリソース、クラスターレベルのリソース、および各 OpenShift Logging コンポーネントについての診断情報を収集できます。

迅速なサポートを得るには、OpenShift Container Platform と OpenShift Logging の両方の診断情報を提供してください。



注記

hack/logging-dump.sh スクリプトは使用しないでください。このスクリプトはサポートされなくなり、データを収集しません。

12.4.1. must-gather ツールについて

oc adm must-gather CLI コマンドは、問題のデバッグに必要となる可能性のあるクラスターからの情報を収集します。

OpenShift Logging 環境の場合、**must-gather** は以下の情報を収集します。

- プロジェクトレベルの Pod、設定マップ、サービスアカウント、ロール、ロールバインディングおよびイベントを含むプロジェクトレベルのリソース
- クラスターレベルでのノード、ロール、およびロールバインディングを含むクラスターレベルのリソース
- ログコレクター、ログストア、およびログビジュアライザーなどの **openshift-logging** および **openshift-operators-redhat** namespace の OpenShift Logging リソース

oc adm must-gather を実行すると、新しい Pod がクラスターに作成されます。データは Pod で収集され、**must-gather.local** で始まる新規ディレクトリーに保存されます。このディレクトリーは、現行の作業ディレクトリーに作成されます。

12.4.2. 前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

12.4.3. OpenShift Logging データの収集

oc adm must-gather CLI コマンドを使用して、OpenShift Logging 環境についての情報を収集できます。

手順

must-gather で OpenShift Logging 情報を収集するには、以下を実行します。

1. **must-gather** 情報を保存する必要のあるディレクトリーに移動します。
2. OpenShift Logging イメージに対して **oc adm must-gather** コマンドを実行します。

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

   must-gather ツールは、現行ディレクトリー内の **must-gather.local** で始まる新規ディレクトリーを作成します。例: **must-gather.local.4157245944708210408**

3. 作成された **must-gather** ディレクトリーから圧縮ファイルを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -cvaf must-gather.tar.gz must-gather.local.4157245944708210408
```

4. 圧縮ファイルを Red Hat カスタマーポータル で作成したサポートケースに添付します。

12.5. CRITICAL ALERTS のトラブルシューティング

12.5.1. Elasticsearch クラスターの正常性が赤である

1 つ以上のプライマリーシャードとそのレプリカがノードに割り当てられません。

トラブルシューティング

1. Elasticsearch クラスターの正常性を確認し、クラスターの **ステータス** が赤であることを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health
```

2. クラスターにに参加したノードを一覧表示します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/nodes?v
```

3. Elasticsearch Pod を一覧表示し、この Pod を直前の手順のコマンド出力にあるノードと比較します。

```
oc -n openshift-logging get pods -l component=elasticsearch
```

4. 一部の Elasticsearch ノードがクラスターに参加していない場合は、以下の手順を実行します。

   a. Elasticsearch に選ばれたコントロールプレーンノードがあることを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/master?v
```

   b. 選ばれたコントロールプレーンノードの Pod ログで問題を確認します。

```
oc logs <elasticsearch_master_pod_name> -c elasticsearch -n openshift-logging
```

   c. 問題がないか、クラスターに参加していないノードのログを確認します。

```
oc logs <elasticsearch_node_name> -c elasticsearch -n openshift-logging
```

5. 全ノードがクラスターに参加している場合は、以下の手順を実行し、クラスターがリカバリープロセスにあるかどうかを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/recovery?active_only=true
```

   コマンドの出力がない場合には、リカバリープロセスが保留中のタスクによって遅延しているか、停止している可能性があります。

6. 保留中のタスクがあるかどうかを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health |grep
number_of_pending_tasks
```

7. 保留中のタスクがある場合は、そのステータスを監視します。
   そのステータスが変化し、クラスターがリカバリー中の場合には、そのまま待機します。リカバリー時間は、クラスターのサイズや他の要素により異なります。

   保留中のタスクのステータスが変更されない場合には、リカバリーが停止していることがわかります。

8. リカバリーが停止しているようであれば、**cluster.routing.allocation.enable** が **none** に設定されているかどうかを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty
```

9. **cluster.routing.allocation.enable** が **none** に設定されている場合、これを **all** に設定します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty -X PUT -d '{"persistent":
{"cluster.routing.allocation.enable":"all"}}'
```

10. どのインデックスが赤のままかを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

11. インデックスがまだ赤い場合は、以下の手順を実行して赤のインデックスをなくします。

    a. キャッシュをクリアします。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_cache/clear?pretty
```

    b. 最大割り当ての再試行回数を増やします。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.allocation.max_retries":10}'
```

    c. スクロールアイテムをすべて削除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_search/scroll/_all -X DELETE
```

    d. タイムアウトを増やします。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.unassigned.node_left.delayed_timeout":"10m"}'
```

12. 前述の手順で赤色のインデックスがなくならない場合には、インデックスを個別に削除します。

    a. 赤色のインデックスの名前を特定します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

    b. 赤色のインデックスを削除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_red_index_name> -X DELETE
```

13. 赤色のインデックスがなく、クラスターのステータスが赤の場合は、データノードで継続的に過剰な処理負荷がかかっていないかを確認します。

    a. Elasticsearch JVM ヒープの使用量が多いかどうかを確認します。

    ```
    oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --query=_nodes/stats?pretty
    ```

    コマンド出力で **node_name.jvm.mem.heap_used_percent** フィールドを確認し、JVM ヒープ使用量を判別します。

    b. 使用量が多い CPU がないかを確認します。

関連情報

- Elasticsearch で "Free up or increase disk space" と検索し、クラスターのステータスが赤または黄色の問題を修正します。

12.5.2. Elasticsearch クラスターの正常性が黄色である

1 つ以上のプライマリーシャードのレプリカシャードがノードに割り当てられません。

トラブルシューティング

1. **ClusterLogging** CR で **nodeCount** を調整してノード数を増やします。

関連情報

- クラスターロギングカスタムリソースについて
- ログストアの永続ストレージの設定
- Elasticsearch で "Free up or increase disk space" と検索し、クラスターのステータスが赤または黄色の問題を修正します。

12.5.3. Elasticsearch Node Disk Low Watermark Reached (Elasticsearch ノードのディスクで低い基準値に達する)

Elasticsearch で、低基準値に到達した ノードにシャードが割り当てられません。

トラブルシューティング

1. Elasticsearch のデプロイ先のノードを特定します。

    ```
    oc -n openshift-logging get po -o wide
    ```

2. **未割り当てのシャード** があるかどうかを確認します。

    ```
    oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --query=_cluster/health?pretty | grep unassigned_shards
    ```

3. 未割り当てのシャードがある場合は、各ノードのディスク領域を確認します。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

4. **nodes.node_name.fs** フィールドで、対象のノードの空きディスク領域を確認します。使用済みディスクの割合が 85% を超える場合は、ノードは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くなります。

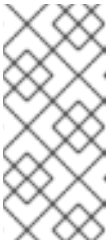
5. すべてのノードでディスク領域を増やしてみてください。

6. ディスク領域を増やせない場合は、新しいデータノードをクラスターに追加してみてください。

7. 新規データノードの追加に問題がある場合には、クラスターの冗長性ポリシー総数を減らします。

   a. 現在の **redundancyPolicy** を確認します。

   ```
   oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
   ```

   注記

   ClusterLogging CR を使用している場合は、以下を入力します。

   ```
   oc -n openshift-logging get cl -o
   jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
   ```

   b. クラスター **redundancyPolicy** が **SingleRedundancy** よりも大きい場合は、**SingleRedundancy** に設定し、この変更を保存します。

8. 前述の手順で問題が解決しない場合は、古いインデックスを削除します。

   a. Elasticsearch の全インデックスのステータスを確認します。

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
   ```

   b. 古いインデックスで削除できるものを特定します。

   c. インデックスを削除します。

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=<elasticsearch_index_name> -X DELETE
   ```

関連情報

- クラスターロギングカスタムリソース の **ClusterLogging カスタムリソース** (CR) のサンプルで redundancyPolicy を参照します。

12.5.4. Elasticsearch Node Disk High Watermark Reached (Elasticsearch ノードのディスクで高い基準値に達する)

Elasticsearch が 高基準値に達した ノードからシャードを移動しようとします。

トラブルシューティング

1. Elasticsearch のデプロイ先のノードを特定します。

```
oc -n openshift-logging get po -o wide
```

2. 各ノードのディスク容量を確認します。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

3. クラスターがリバランスされているかどうかを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/health?pretty | grep relocating_shards
```

コマンドの出力でシャードの再配置が表示される場合には、高い基準値を超過しています。高い基準値のデフォルト値は 90% です。

基準値のしきい値上限を超えておらず、ディスクの使用量が少ないノードに、シャードを移動します。

4. シャードを特定ノードに割り当てるには、領域の一部を解放します。

5. すべてのノードでディスク領域を増やしてみてください。

6. ディスク領域を増やせない場合は、新しいデータノードをクラスターに追加してみてください。

7. 新規データノードの追加に問題がある場合には、クラスターの冗長性ポリシー総数を減らします。

   a. 現在の **redundancyPolicy** を確認します。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```

注記

ClusterLogging CR を使用している場合は、以下を入力します。

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

   b. クラスター **redundancyPolicy** が **SingleRedundancy** よりも大きい場合は、**SingleRedundancy** に設定し、この変更を保存します。

8. 前述の手順で問題が解決しない場合は、古いインデックスを削除します。

   a. Elasticsearch の全インデックスのステータスを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

   b. 古いインデックスで削除できるものを特定します。

c. インデックスを削除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

関連情報

- クラスターロギングカスタムリソース の **ClusterLogging カスタムリソース** (CR) のサンプルで redundancyPolicy を参照します。

12.5.5. Elasticsearch Node Disk Flood Watermark Reached (Elasticsearch ノードのディスクがいっぱいの基準値に達する)

Elasticsearch は、両条件が含まれるすべてのインデックスに対して読み取り専用のインデックスブロックを強制的に適用します。

- 1 つ以上のシャードがノードに割り当てられます。
- 1 つ以上のディスクが いっぱいの段階 を超えています。

トラブルシューティング

1. Elasticsearch ノードのディスク領域を確認します。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

   nodes.node_name.fs フィールドで、対象のノードの空きディスク領域を確認します。

2. 使用済みディスクの割合が 95% を超える場合は、ノードがいっぱいの基準値が越えたことを意味します。この特定のノードに割り当てられたシャードへの書き込みは、ブロックされます。

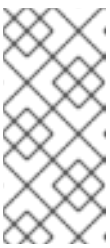
3. すべてのノードでディスク領域を増やしてみてください。

4. ディスク領域を増やせない場合は、新しいデータノードをクラスターに追加してみてください。

5. 新規データノードの追加に問題がある場合には、クラスターの冗長性ポリシー総数を減らします。

   a. 現在の **redundancyPolicy** を確認します。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



注記

ClusterLogging CR を使用している場合は、以下を入力します。

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

b. クラスター **redundancyPolicy** が **SingleRedundancy** よりも大きい場合は、**SingleRedundancy** に設定し、この変更を保存します。

6. 前述の手順で問題が解決しない場合は、古いインデックスを削除します。

a. Elasticsearch の全インデックスのステータスを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

b. 古いインデックスで削除できるものを特定します。

c. インデックスを削除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

7. ディスク使用領域が 90% 未満になるまで、このままディスク領域を解放して監視します。次に、この特定のノードへの書き込み禁止を解除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_all/_settings?pretty -X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

関連情報

- クラスターロギングカスタムリソース の **ClusterLogging カスタムリソース** (CR) のサンプルで redundancyPolicy を参照します。

12.5.6. Elasticsearch JVM ヒープの使用量が高い

Elasticsearch ノードで使用済みの JVM ヒープメモリーが 75% を超えます。

トラブルシューティング

ヒープサイズを増やす ことを検討してください。

12.5.7. 集計ロギングシステムの CPU が高い

ノード上のシステムの CPU 使用量が高くなります。

トラブルシューティング

クラスターノードの CPU を確認します。ノードへ割り当てる CPU リソースを増やすことを検討してください。

12.5.8. Elasticsearch プロセスの CPU が高い

ノードでの Elasticsearch プロセスの CPU 使用量が高くなります。

トラブルシューティング

クラスターノードの CPU を確認します。ノードへ割り当てる CPU リソースを増やすことを検討してください。

12.5.9. Elasticsearch ディスク領域が不足している

Elasticsearch クラスターは、現在のディスク使用量に基づいて次の 6 時間以内にディスク領域が不足することが予想します。

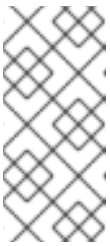
トラブルシューティング

1. Elasticsearch ノードのディスク領域を取得します。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

2. コマンド出力の **nodes.node_name.fs** フィールドで、対象ノードの空きディスク領域を確認します。
3. すべてのノードでディスク領域を増やしてみてください。
4. ディスク領域を増やせない場合は、新しいデータノードをクラスターに追加してみてください。
5. 新規データノードの追加に問題がある場合には、クラスターの冗長性ポリシー総数を減らします。
   a. 現在の **redundancyPolicy** を確認します。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



注記

ClusterLogging CR を使用している場合は、以下を入力します。

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

   b. クラスター **redundancyPolicy** が **SingleRedundancy** よりも大きい場合は、**SingleRedundancy** に設定し、この変更を保存します。

6. 前述の手順で問題が解決しない場合は、古いインデックスを削除します。
   a. Elasticsearch の全インデックスのステータスを確認します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

   b. 古いインデックスで削除できるものを特定します。
   c. インデックスを削除します。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

関連情報

- クラスターロギングカスタムリソース の **ClusterLogging カスタムリソース** (CR) のサンプルで redundancyPolicy を参照します。

- Elasticsearch アラートルール で ElasticsearchDiskSpaceRunningLow を参照します。
- Elasticsearch で "Free up or increase disk space" と検索し、クラスターのステータスが赤または黄色の問題を修正します。

12.5.10. Elasticsearch FileDescriptor の使用量が高い

現在の使用傾向に基づいて、ノードで予測されるファイル記述子の数は十分ではありません。

トラブルシューティング

必要に応じて、Elasticsearch ファイル記述子 のトピックで説明されているように、各ノードの **max_file_descriptors** の値を確認して設定します。

関連情報

- Elasticsearch ルール で ElasticsearchHighFileDescriptorUsage を参照します。
- OpenShift Logging ダッシュボード で File Descriptors In Use を参照します。

第13章 OPENSHIFT LOGGING のアンインストール

OpenShift Logging をお使いの OpenShift Container Platform クラスターから削除することができます。

13.1. OPENSHIFT CONTAINER PLATFORM からの OPENSHIFT LOGGING のアンインストール

ClusterLogging カスタムリソース (CR) を削除して、ログ集計を停止できます。CR の削除後に残る他の OpenShift Logging コンポーネントについては、オプションで削除できます。

ClusterLogging CR を削除しても、永続ボリューム要求 (PVC) は削除されません。残りの PVC、永続ボリューム (PV)、および関連データを保持するか、または削除するには、さらにアクションを実行する必要があります。

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。

手順

OpenShift Logging を削除するには、以下を実行します。

1. OpenShift Container Platform Web コンソールを使って **ClusterLogging** CR を削除できます。
   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 Details**ページで、 **Instances** をクリックします。
   d. インスタンスの横にある Options メニュー  をクリックし、 **Delete ClusterLogging** を選択します。
2. オプション: カスタムリソース定義 (CRD) を削除します。
   a. **Administration → Custom Resource Definitions**ページに切り替えます。
   b. **ClusterLogForwarder** の横にある Options メニュー  をクリックし、 **Delete Custom Resource Definition** を選択します。
   c. **ClusterLogging** の横にある Options メニュー  をクリックし、 **Delete Custom Resource Definition** を選択します。
   d. **Elasticsearch** の横にある Options メニュー  をクリックし、**Delete Custom Resource Definition** を選択します。
3. オプション: Red Hat OpenShift Logging Operator および OpenShift Elasticsearch Operator を削除します。

a. **Operators → Installed Operators** ページに切り替えます。

b. Red Hat OpenShift Logging Operator の横にある Options メニュー をクリックし、**Uninstall Operator** を選択します。

c. OpenShift Elasticsearch Operator の横にある Options メニュー をクリックし、**Uninstall Operator** を選択します。

4. オプション: Cluster Logging および Elasticsearch プロジェクトを削除します。

a. **Home → Projects** ページに切り替えます。

b. **openshift-logging** プロジェクトの横にある Options メニュー をクリックし、**Delete Project** を選択します。

c. ダイアログボックスで **openshift-logging** を入力して、**Delete** をクリックし、削除を確認します。

d. **openshift-operators-redhat** プロジェクトの横にある Options メニュー をクリックし、**Delete Project** を選択します。

重要

他のグローバル Operator がこの namespace にインストールされている場合、**openshift-operators-redhat** プロジェクトを削除しないでください。

e. ダイアログボックスで **openshift-operators-redhat** を入力し、**Delete** をクリックして削除を確認します。

5. 他の Pod で再利用するために PVC を保持するには、PVC の回収に必要なラベルまたは PVC 名を保持します。

6. オプション: PVC を保持する必要がない場合は、それらを削除できます。

警告

PVC の解放または削除により PV が削除され、データの損失が生じる可能性があります。

a. **Storage → Persistent Volume Claims**ページに切り替えます。

b. 各 PVC の横にある Options メニュー  をクリックし、**Delete Persistent Volume Claim** を選択します。

c. ストレージ領域を回復する必要がある場合は、PV を削除できます。

関連情報

- 永続ボリュームの手動回収

第14章 ログレコードのフィールド

以下のフィールドは、OpenShift Logging でエクスポートされるログレコードに表示される場合があります。ログレコードは通常 JSON オブジェクトとしてフォーマットされますが、同じデータモデルは他のエンコーディングに適用できます。

Elasticsearch および Kibana からこれらのフィールドを検索するには、検索時に点線の全フィールド名を使用します。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。

最上位フィールドはすべてのレコードに存在する可能性があります。

第15章 MESSAGE

元のログエントリーテキスト (UTF-8 エンコード)。このフィールドが存在しないか、空でない **構造化** フィールドが存在する可能性があります。詳細は、**structured** の説明を参照してください。

データのタイプ	text
値の例	**HAPPY**

第16章 STRUCTURED

構造化されたオブジェクトとしての元のログエントリー。このフィールドは、フォワーダーが構造化された JSON ログを解析するように設定されている場合に存在する可能性があります。元のログエントリーの構造化ログが有効である場合に、このフィールドには同等の JSON 構造が含まれます。それ以外の場合は、このフィールドは空または存在しないため、**message** フィールドに元のログメッセージが含まれます。**構造化された** フィールドには、ログメッセージに含まれるサブフィールドがあるので、ここでは制約が定義されていません。

データのタイプ	group
値の例	map[message:starting fluentd worker pid=21631 ppid=21618 worker=0 pid:21631 ppid:21618 worker:0]

第17章 @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。@接頭辞は、特定の用途で使用できるように予約されているフィールドを表します。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。

データのタイプ	日付
値の例	**2015-01-24 14:06:05.071000000 Z**

第18章 HOSTNAME

このログメッセージの発信元のホスト名。Kubernetes クラスターでは、これは **kubernetes.host** と同じです。

データのタイプ	キーワード

第19章 IPADDR4

ソースサーバーの IPv4 アドレス。配列を指定できます。

データのタイプ	ip

第20章 IPADDR6

ソースサーバーの IPv6 アドレス (ある場合)。配列を指定できます。

データのタイプ	ip

第21章 LEVEL

rsyslog (severitytext プロパティー)、python のロギングモジュールなどのさまざまなソースのロギングレベル。

以下の値は **syslog.h** から取得されます。値の前には 同等の数値 が追加されます。

- **0** = **emerg**、システムが使用できない。
- **1** = **alert**。アクションをすぐに実行する必要がある。
- **2** = **crit**、致命的な状況。
- **3** = **err**、エラーのある状況。
- **4** = **warn**、警告のある状況。
- **5** = **notice**、通常ではあるが、影響が大きい状況。
- **6** = **info**、情報提供。
- **7** = **debug**、デバッグレベルのメッセージ。

以下の 2 つの値は **syslog.h** の一部ではありませんが、広く使用されています。

- **8** = **trace**、トレースレベルメッセージ。これは、**debug** メッセージよりも詳細にわたります。
- **9** = **unknown**、ロギングシステムで認識できない値を取得した場合。

他のロギングシステムのログレベルまたは優先度を前述の一覧で最も近い一致にマップします。たとえば python logging では、**CRITICAL** と **crit**、**ERROR** と **err** が同じです。

データのタイプ	キーワード
値の例	**info**

第22章 PID

ロギングエンティティーのプロセス ID です (ある場合)。

データのタイプ	キーワード

第23章 サービス

ロギングエンティティー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 の **APP-NAME** および rsyslog の **programname** プロパティーはサービスフィールドにマップされます。

データのタイプ	キーワード

第24章 TAGS

オプション: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、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。

データのタイプ	text

第25章 FILE

コレクターがこのログエントリーを読み取るログファイルへのパス。通常、これはクラスターノードの **/var/log** ファイルシステム内のパスです。

データのタイプ	text

第26章 OFFSET

オフセット値。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)。

データのタイプ	Long

第27章 KUBERNETES

Kubernetes 固有メタデータの namespace です。

データのタイプ	group

27.1. KUBERNETES.POD_NAME

Pod の名前。

データのタイプ	キーワード

27.2. KUBERNETES.POD_ID

Pod の Kubernetes ID。

データのタイプ	キーワード

27.3. KUBERNETES.NAMESPACE_NAME

Kubernetes の namespace の名前。

データのタイプ	キーワード

27.4. KUBERNETES.NAMESPACE_ID

Kubernetes の namespace ID。

データのタイプ	キーワード

27.5. KUBERNETES.HOST

Kubernetes ノード名。

データのタイプ	キーワード

27.6. KUBERNETES.CONTAINER_NAME

Kubernetes のコンテナーの名前。

データのタイプ	キーワード

27.7. KUBERNETES.ANNOTATIONS

Kubernetes オブジェクトに関連付けられるアノテーション。

データのタイプ	group

27.8. KUBERNETES.LABELS

元の Kubernetes Pod にあるラベル

データのタイプ	group

27.9. KUBERNETES.EVENT

Kubernetes マスター API から取得した Kubernetes イベント。このイベントの説明は基本的に、Event v1 コア の **type Event** に準拠します。

データのタイプ	group

27.9.1. kubernetes.event.verb

イベントのタイプ: **ADDED**、**MODIFIED** または **DELETED**

データのタイプ	キーワード
値の例	**追加済み**

27.9.2. kubernetes.event.metadata

イベント作成の場所および時間に関する情報

データのタイプ	group

27.9.2.1. kubernetes.event.metadata.name

イベント作成をトリガーしたオブジェクトの名前

データのタイプ	キーワード
値の例	**java-mainclass-1.14d888a4cfc24890**

27.9.2.2. kubernetes.event.metadata.namespace

イベントが最初に発生した namespace の名前。これは、**eventrouter** アプリケーションのデプロイ先の namespace である **kubernetes.namespace_name** とは異なることに注意してください。

データのタイプ	キーワード
値の例	**default**

27.9.2.3. kubernetes.event.metadata.selfLink

イベントへのリンク

データのタイプ	キーワード
値の例	**/api/v1/namespaces/javaj/events/java-mainclass-1.14d888a4cfc24890**

27.9.2.4. kubernetes.event.metadata.uid

イベントの一意の ID

データのタイプ	キーワード
値の例	**d828ac69-7b58-11e7-9cf5-5254002f560c**

27.9.2.5. kubernetes.event.metadata.resourceVersion

イベントが発生したサーバーの内部バージョンを識別する文字列。クライアントはこの文字列を使用して、オブジェクトが変更されたタイミングを判断できます。

データのタイプ	integer
値の例	**311987**

27.9.3. kubernetes.event.involvedObject

イベントに関するオブジェクト。

データのタイプ	group

27.9.3.1. kubernetes.event.involvedObject.kind

オブジェクトのタイプ

データのタイプ	キーワード
値の例	**ReplicationController**

27.9.3.2. kubernetes.event.involvedObject.namespace

関係するオブジェクトの namespace 名。これは、**eventrouter** アプリケーションのデプロイ先の namespace である **kubernetes.namespace_name** とは異なる可能性があることに注意してください。

データのタイプ	キーワード
値の例	**default**

27.9.3.3. kubernetes.event.involvedObject.name

イベントをトリガーしたオブジェクトの名前

データのタイプ	キーワード
値の例	**java-mainclass-1**

27.9.3.4. kubernetes.event.involvedObject.uid

オブジェクトの一意の ID

データのタイプ	キーワード
値の例	**e6bff941-76a8-11e7-8193-5254002f560c**

27.9.3.5. kubernetes.event.involvedObject.apiVersion

kubernetes マスター API のバージョン

データのタイプ	キーワード
値の例	**v1**

27.9.3.6. kubernetes.event.involvedObject.resourceVersion

イベントをトリガーしたサーバーの内部バージョンの Pod を識別する文字列。クライアントはこの文字列を使用して、オブジェクトが変更されたタイミングを判断できます。

データのタイプ	キーワード
値の例	**308882**

27.9.4. kubernetes.event.reason

このイベントを生成する理由を示す、マシンが理解可能な短い文字列

データのタイプ	キーワード
値の例	**SuccessfulCreate**

27.9.5. kubernetes.event.source_component

このイベントを報告したコンポーネント

データのタイプ	キーワード
値の例	**replication-controller**

27.9.6. kubernetes.event.firstTimestamp

イベントが最初に記録された時間

データのタイプ	日付
値の例	**2017-08-07 10:11:57.000000000 Z**

27.9.7. kubernetes.event.count

このイベントが発生した回数

データのタイプ	integer
値の例	**1**

27.9.8. kubernetes.event.type

イベントのタイプ、**Normal** または **Warning**。今後、新しいタイプが追加される可能性があります。

データのタイプ	キーワード
値の例	**Normal**

第28章 OPENSHIFT

openshift-logging 固有のメタデータの namespace

データのタイプ	group

28.1. OPENSHIFT.LABELS

クラスターログフォワーダー設定によって追加されるラベル

データのタイプ	group