



OpenShift Dedicated 4

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

OpenShift Dedicated 4 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

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

このドキュメントでは、OpenShift Dedicated で OpenShift Logging を設定する手順を説明します。

目次

第1章 リリースノート	5
1.1. LOGGING 5.9	5
1.2. LOGGING 5.8	8
1.3. LOGGING 5.7	19
第2章 サポート	31
2.1. サポート対象の API カスタムリソース定義	31
2.2. サポートされない設定	32
2.3. 管理外の OPERATOR のサポートポリシー	32
2.4. RED HAT サポート用のロギングデータの収集	33
第3章 ロギングのトラブルシューティング	35
3.1. ロギングステータスの表示	35
3.2. ログ転送のトラブルシューティング	40
3.3. ログアラートのトラブルシューティング	42
3.4. ELASTICSEARCH ログストアのステータスの表示	52
第4章 ロギング	61
4.1. ロギングアーキテクチャー	61
4.2. ロギングのデプロイ	62
4.3. OPENSIFT DEDICATED に推奨される CLOUDWATCH	63
第5章 ロギングのインストール	65
5.1. WEB コンソールを使用して RED HAT OPENSIFT LOGGING OPERATOR をインストールする	65
5.2. WEB コンソールを使用して CLUSTERLOGGING オブジェクトを作成する	66
5.3. CLI を使用して RED HAT OPENSIFT LOGGING OPERATOR をインストールする	67
5.4. CLI を使用して CLUSTERLOGGING オブジェクトを作成する	69
5.5. インストール後のタスク	72
第6章 ロギングの更新	78
6.1. マイナーリリースの更新	78
6.2. メジャーリリースの更新	78
6.3. すべての NAMESPACE を監視するための RED HAT OPENSIFT LOGGING OPERATOR のアップグレード	78
6.4. RED HAT OPENSIFT LOGGING OPERATOR の更新	79
6.5. LOKI OPERATOR の更新	80
6.6. OPENSIFT ELASTICSEARCH OPERATOR の更新	80
第7章 ログの可視化	85
7.1. ログの可視化について	85
7.2. WEB コンソールによるログの可視化	87
7.3. クラスタダッシュボードの表示	89
7.4. KIBANA によるログの可視化	97
第8章 OPENSIFT DEDICATED クラスタのサービスログへのアクセス	104
8.1. OPENSIFT CLUSTER MANAGER を使用したサービスログの表示	104
8.2. クラスタ通知連絡先の追加	104
第9章 ロギングデプロイメントの設定	106
9.1. ロギングコンポーネントの CPU およびメモリー制限の設定	106
第10章 ログの収集および転送	108
10.1. ログの収集と転送	108
10.2. ログ出力のタイプ	114

10.3. JSON ログ転送の有効化	116
10.4. ログ転送の設定	122
10.5. ロギングコレクターの設定	167
10.6. KUBERNETES イベントの収集および保存	176
第11章 ログストレージ	181
11.1. ログストレージについて	181
11.2. ログストレージのインストール	182
11.3. LOKISTACK ログストアの設定	202
11.4. ELASTICSEARCH ログストアの設定	213
第12章 ロギングアラート	231
12.1. デフォルトのロギングアラート	231
12.2. カスタムロギングアラート	234
第13章 パフォーマンスと信頼性のチューニング	240
13.1. フロー制御メカニズム	240
13.2. コンテンツによるログのフィルタリング	243
13.3. メタデータによるログのフィルタリング	247
第14章 スケジューリングリソース	251
14.1. ノードセクターを使用したロギングリソースの移動	251
14.2. ティントと容認を使用したロギング POD の配置制御	260
第15章 ロギングのアンインストール	271
15.1. ロギングのアンインストール	271
15.2. ロギング PVC の削除	272
15.3. LOKI のアンインストール	272
15.4. ELASTICSEARCH のアンインストール	273
15.5. CLI の使用によるクラスターからの OPERATOR の削除	274
第16章 ログレコードのフィールド	276
MESSAGE	276
STRUCTURED	276
@TIMESTAMP	276
HOSTNAME	276
IPADDR4	277
IPADDR6	277
LEVEL	277
PID	278
サービス	278
第17章 TAGS	279
FILE	279
OFFSET	279
第18章 KUBERNETES	280
18.1. KUBERNETES.POD_NAME	280
18.2. KUBERNETES.POD_ID	280
18.3. KUBERNETES.NAMESPACE_NAME	280
18.4. KUBERNETES.NAMESPACE_ID	280
18.5. KUBERNETES.HOST	280
18.6. KUBERNETES.CONTAINER_NAME	280
18.7. KUBERNETES.ANNOTATIONS	281
18.8. KUBERNETES.LABELS	281

18.9. KUBERNETES.EVENT	281
第19章 OPENSIFT	286
19.1. OPENSIFT.LABELS	286
第20章 API リファレンス	287
20.1.5.6 LOGGING API リファレンス	287
第21章 用語集	327

第1章 リリースノート

1.1. LOGGING 5.9



注記

ロギングはインストール可能なコンポーネントとして提供され、コアの OpenShift Dedicated とは異なるリリースサイクルで提供されます。[Red Hat OpenShift Container Platform ライフサイクルポリシー](#) はリリースの互換性を概説しています。



注記

stable チャンネルは、Logging の最新リリースを対象とする更新のみを提供します。以前のリリースの更新を引き続き受信するには、サブスクリプションチャンネルを **stable-x.y** に変更する必要があります。**xy** は、インストールしたログのメジャーバージョンとマイナーバージョンを表します。たとえば、**stable-5.7** です。

1.1.1. Logging 5.9.0

このリリースには、[OpenShift Logging バグ修正リリース 5.9.0](#) が含まれています。

1.1.1.1. 削除通知

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。以前のロギングリリースの OpenShift Elasticsearch Operator のインスタンスは、ロギングリリースの EOL までサポートされます。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

1.1.1.2. 非推奨のお知らせ

- Logging 5.9 では、Fluentd および Kibana が非推奨となり、Logging 6.0 で削除される予定です。Logging 6.0 は、今後の OpenShift Dedicated リリースに同梱される見込みです。Red Hat は、現行リリースのライフサイクルにおいて、該当コンポーネントの「重大」以上の CVE に対するバグ修正とサポートを提供しますが、機能拡張は提供しません。Red Hat OpenShift Logging Operator が提供する Vector ベースのコレクターと、Loki Operator が提供する LokiStack は、ログの収集と保存に推奨される Operator です。Vector および Loki ログスタックは今後強化される予定であるため、すべてのユーザーに対しこのスタックの採用が推奨されます。
- Logging 5.9 では、Splunk 出力タイプの **Fields** オプションが実装されていません。現在、このオプションは非推奨となっています。これは今後のリリースで削除されます。

1.1.1.3. 機能拡張

1.1.1.3.1. ログの収集

- この機能拡張により、ワークロードのメタデータを使用して、その内容に基づいてログを **drop** または **prune** することで、ログ収集のプロセスを改善する機能が追加されます。さらに、ジャーナルログやコンテナログなどのインフラストラクチャーログ、および **kube api** ログや **ovn** ログなどの監査ログを収集して、個々のソースのみを収集することもできます。(LOG-2155)

- この機能拡張により、新しいタイプのリモートログレシーバーである `syslog` レシーバーが導入されます。これにより、ネットワーク経由でポートを公開するように設定し、外部システムが `rsyslog` などの互換性のあるツールを使用して `syslog` ログを送信することができます。(LOG-3527)
- この更新により、**ClusterLogForwarder** API は Azure Monitor ログへのログ転送をサポートするようになり、ユーザーのモニタリング機能が向上します。この機能により、ユーザーは最適なシステムパフォーマンスを維持し、Azure Monitor のログ分析プロセスを合理化できるため、問題解決が迅速化され、運用効率が向上します。(LOG-4605)
- この機能拡張により、コレクターを2つのレプリカを持つデプロイメントとしてデプロイすることで、コレクターリソースの使用率が向上します。これは、すべてのノードでデーモンセットを使用するのではなく、**ClusterLogForwarder** カスタムリソース (CR) で定義されている唯一の入力ソースがレシーバー入力である場合に発生します。さらに、この方法でデプロイされたコレクターは、ホストファイルシステムをマウントしません。この機能拡張を使用するには、**ClusterLogForwarder** CR に、**logging.openshift.io/dev-preview-enable-collector-as-deployment** アノテーションを付ける必要があります。(LOG-4779)
- この機能拡張により、サポートされているすべての出力にわたって、カスタムテナント設定の機能が導入され、ログレコードを論理的に整理できるようになります。ただし、管理対象ストレージのロギングに対するカスタムテナント設定は許可されません。(LOG-4843)
- この更新により、**default**、**openshift***、**kube*** などの1つ以上のインフラストラクチャー namespace を使用してアプリケーション入力を指定する **ClusterLogForwarder** CR には、**collect-infrastructure-logs** ロールを持つサービスアカウントが必要になりました。(LOG-4943)
- この機能拡張により、圧縮、再試行期間、最大ペイロードなどの出力設定を、受信者の特性に合わせて調整する機能が導入されました。さらに、この機能には、管理者がスループットとログの耐久性を選択できる配信モードが含まれています。たとえば、**AtLeastOnce** オプションは、収集されたログのディスクバッファリングを最小限に設定し、コレクターが再起動後にそれらのログを配信できるようにします。(LOG-5026)
- この機能拡張により、Elasticsearch、Fluentd、Kibana の廃止についてユーザーに警告する3つの新しい Prometheus アラートが追加されました。(LOG-5055)

1.1.1.3.2. ログのストレージ

- LokiStack のこの機能拡張により、新しい V13 オブジェクトストレージ形式を使用し、デフォルトで自動ストリームシャードイングを有効にすることで、OTEL のサポートが向上します。これにより、コレクターは今後の機能拡張や設定にも備えることができます。(LOG-4538)
- この機能拡張により、STS 対応の OpenShift Dedicated 4.14 以降のクラスターで、Azure および AWS ログストアにおける有効期間の短いトークンワークロードアイデンティティフェデレーションのサポートが導入されます。ローカルストレージには、LokiStack CR の **spec.storage.secret** の下に **CredentialMode: static** アノテーションを追加する必要があります。(LOG-4540)
- この更新により、Azure ストレージシークレットの検証が拡張され、特定のエラー状態に対する早期警告が可能になりました。(LOG-4571)
- この更新により、Loki は、GCP ワークロードアイデンティティフェデレーションメカニズムのアップストリームおよびダウンストリームサポートを追加するようになりました。これにより、対応するオブジェクトストレージサービスへの認証および承認されたアクセスが可能になります。(LOG-4754)

1.1.1.4. バグ修正

- この更新前は、ロギングの `must-gather` は FIPS 対応のクラスターでログを収集できませんでした。この更新により、`cluster-logging-rhel9-operator` で新しい `oc` クライアントが利用できるようになり、`must-gather` が FIPS クラスターで適切に動作するようになりました。(LOG-4403)
- この更新前は、LokiStack ルーラー Pod は、Pod 間通信に使用される HTTP URL で IPv6 Pod IP をフォーマットできませんでした。この問題により、Prometheus と互換性のある API を介したルールとアラートのクエリーが失敗していました。この更新により、LokiStack ルーラー Pod は IPv6 Pod IP を角かっこでカプセル化し、問題を解決しています。現在、Prometheus と互換性のある API を介したルールとアラートのクエリーは、IPv4 環境の場合と同じように機能するようになりました。(LOG-4709)
- この修正前は、ロギングの `must-gather` からの YAML コンテンツが 1 行でエクスポートされていたため、判読不能でした。この更新により、YAML の空白が保持され、ファイルが適切にフォーマットされるようになります。(LOG-4792)
- この更新前は、`ClusterLogForwarder` CR が有効になっていると、`ClusterLogging.Spec.Collection` が `nil` のときに Red Hat OpenShift Logging Operator で `nil` ポインター例外が発生する可能性があります。この更新により、この問題は Red Hat OpenShift Logging Operator で解決されました。(LOG-5006)
- この更新前は、特定のコーナーケースで、`ClusterLogForwarder` CR ステータスフィールドを置き換えると、`Status` 条件のタイムスタンプが変更されるため、`resourceVersion` が常に更新されていました。この状況により、無限の調整ループが発生しました。この更新により、すべてのステータス条件が同期されるため、条件が同じであればタイムスタンプは変更されません。(LOG-5007)
- この更新前は、コレクターによるメモリー消費量の増加に対処するために、`drop_newest` で内部バッファリング動作が行われ、大量のログ損失が発生していました。この更新により、動作はコレクターのデフォルトを使用するようになりました。(LOG-5123)
- この更新前は、`openshift-operators-redhat` namespace の Loki Operator の `ServiceMonitor` が、静的トークンと CA ファイルを認証に使用していました。そのため、`ServiceMonitor` 設定の User Workload Monitoring 仕様の Prometheus Operator でエラーが発生していました。この更新により、`openshift-operators-redhat` namespace の Loki Operator の `ServiceMonitor` が、`LocalReference` オブジェクトによってサービスアカウントトークンシークレットを参照するようになりました。このアプローチにより、Prometheus Operator の User Workload Monitoring 仕様が Loki Operator の `ServiceMonitor` を正常に処理できるようになり、Prometheus が Loki Operator メトリクスを収集できるようになりました。(LOG-5165)
- この更新前は、Loki Operator の `ServiceMonitor` の設定が多くの Kubernetes サービスと一致することがありました。その結果、Loki Operator のメトリクスが複数回収集されることがありました。この更新により、`ServiceMonitor` の設定が専用のメトリクスサービスのみと一致するようになりました。(LOG-5212)

1.1.1.5. 既知の問題

なし。

1.1.1.6. CVE

- [CVE-2023-5363](#)
- [CVE-2023-5981](#)

- [CVE-2023-46218](#)
- [CVE-2024-0553](#)
- [CVE-2023-0567](#)

1.2. LOGGING 5.8



注記

ロギングはインストール可能なコンポーネントとして提供され、コアの OpenShift Dedicated とは異なるリリースサイクルで提供されます。[Red Hat OpenShift Container Platform ライフサイクルポリシー](#) はリリースの互換性を概説しています。



注記

stable チャンネルは、Logging の最新リリースを対象とする更新のみを提供します。以前のリリースの更新を引き続き受信するには、サブスクリプションチャンネルを **stable-x.y** に変更する必要があります。**xy** は、インストールしたログのメジャーバージョンとマイナーバージョンを表します。たとえば、**stable-5.7** です。

1.2.1. Logging 5.8.5

このリリースには、[OpenShift Logging バグ修正リリース 5.8.5](#) が含まれています。

1.2.1.1. バグ修正

- この更新前は、Loki Operator の **ServiceMonitor** の設定が多くの Kubernetes サービスと一致することがありました。その結果、Loki Operator のメトリクスが複数回収集されることがありました。この更新により、**ServiceMonitor** の設定が専用のメトリクスサービスのみと一致するようになりました。(LOG-5250)
- この更新前は、Red Hat のビルドパイプラインが Loki ビルドの既存のビルド詳細を使用せず、リビジョン、ブランチ、バージョンなどの情報を省略していました。この更新により、Red Hat のビルドパイプラインがこれらの詳細を Loki ビルドに追加するようになり、問題が修正されました。(LOG-5201)
- この更新前は、**LokiStack** の準備ができていないかどうかを判断するために、Pod が実行中かどうかを Loki Operator がチェックしていました。この更新により、Pod の準備ができていないのかも Loki Operator がチェックするようになり、**LokiStack** の準備状況がそのコンポーネントの状態を反映するようになりました。(LOG-5171)
- この更新前は、ログメトリクスのクエリーを実行すると、ヒストグラムでエラーが発生していました。この更新により、ヒストグラムの切り替え機能とグラフが無効になり、非表示になりました。ヒストグラムはログメトリクスでは機能しないためです。(LOG-5044)
- この更新前は、Loki および Elasticsearch バンドルの **maxOpenShiftVersion** が間違っていました。その結果、**IncompatibilityOperatorsInstalled** アラートが発生していました。この更新により、バンドルの **maxOpenShiftVersion** プロパティーとして 4.16 が追加され、問題が修正されました。(LOG-5272)
- この更新前は、ビルドパイプラインにビルド日付のリンカーフラグが含まれていなかったため、Loki ビルドの **buildDate** と **goVersion** に空の文字列が表示されていました。この更新により、欠落していたリンカーフラグがビルドパイプラインに追加され、問題が修正されました。

た。(LOG-5274)

- この更新前は、LogQL 解析のバグにより、クエリーから一部の行フィルターが除外されていました。この更新により、元のクエリーが変更されることなく、すべての行フィルターが解析に含まれるようになりました。(LOG-5270)
- この更新前は、**openshift-operators-redhat** namespace の Loki Operator の **ServiceMonitor** が、静的トークンと CA ファイルを認証に使用していました。そのため、**ServiceMonitor** 設定の User Workload Monitoring 仕様の Prometheus Operator でエラーが発生していました。この更新により、**openshift-operators-redhat** namespace の Loki Operator の **ServiceMonitor** が、**LocalReference** オブジェクトによってサービスアカウントトークンシークレットを参照するようになりました。このアプローチにより、Prometheus Operator の User Workload Monitoring 仕様が Loki Operator の **ServiceMonitor** を正常に処理できるようになり、Prometheus が Loki Operator メトリクスを収集できるようになりました。(LOG-5240)

1.2.1.2. CVE

- [CVE-2023-5363](#)
- [CVE-2023-5981](#)
- [CVE-2023-6135](#)
- [CVE-2023-46218](#)
- [CVE-2023-48795](#)
- [CVE-2023-51385](#)
- [CVE-2024-0553](#)
- [CVE-2024-0567](#)
- [CVE-2024-24786](#)
- [CVE-2024-28849](#)

1.2.2. Logging 5.8.4

このリリースには、[OpenShift Logging バグ修正リリース 5.8.4](#) が含まれています。

1.2.2.1. バグ修正

- この更新まで、開発者コンソールのログでは現在の namespace が考慮されなかったため、クラスター全体のログアクセスを持たないユーザーのクエリーが拒否されていました。今回の更新により、サポートされているすべての OCP バージョンで、正しい namespace が確実に含まれるようになりました。(LOG-4905)
- この更新まで、Cluster Logging Operator は、デフォルトのログ出力が LokiStack の場合のみ、LokiStack デプロイメントをサポートする **ClusterRoles** をデプロイしていました。今回の更新により、このロールは読み取りおよび書き込みの 2 つのグループに分割されました。書き込みロールは、これまで使用されていたすべてのロールと同様に、デフォルトのログストレージ設定に基づきデプロイされます。読み取りロールは、ロギングコンソールプラグインがアクティブかどうかに基づきデプロイされます。(LOG-4987)
- この更新まで、1つのサービスで **ownerReferences** が変更されると、同じ出力レシーバー名を

定義する複数の **ClusterLogForwarder** はサービスを延々と調整していました。今回の更新により、各レシーバー入力には、**<CLF.Name>-<input.Name>** の規則に準じて名付けられた独自のサービスが追加されます。(LOG-5009)

- この更新まで、**ClusterLogForwarder** は、シークレットなしでログを cloudwatch に転送する際にエラーを報告しませんでした。今回の更新により、シークレットなしでログを cloudwatch に転送すると、**secret must be provided for cloudwatch output** のエラーメッセージが表示されるようになりました。(LOG-5021)
- この更新まで、**log_forwarder_input_info** には **application**、**infrastructure**、**audit** の入力メトリクスポイントが含まれていました。今回の更新により、**http** もメトリクスポイントとして追加されました。(LOG-5043)

1.2.2.2. CVE

- [CVE-2021-35937](#)
- [CVE-2021-35938](#)
- [CVE-2021-35939](#)
- [CVE-2022-3545](#)
- [CVE-2022-24963](#)
- [CVE-2022-36402](#)
- [CVE-2022-41858](#)
- [CVE-2023-2166](#)
- [CVE-2023-2176](#)
- [CVE-2023-3777](#)
- [CVE-2023-3812](#)
- [CVE-2023-4015](#)
- [CVE-2023-4622](#)
- [CVE-2023-4623](#)
- [CVE-2023-5178](#)
- [CVE-2023-5363](#)
- [CVE-2023-5388](#)
- [CVE-2023-5633](#)
- [CVE-2023-6679](#)
- [CVE-2023-7104](#)
- [CVE-2023-27043](#)

- [CVE-2023-38409](#)
- [CVE-2023-40283](#)
- [CVE-2023-42753](#)
- [CVE-2023-43804](#)
- [CVE-2023-45803](#)
- [CVE-2023-46813](#)
- [CVE-2024-20918](#)
- [CVE-2024-20919](#)
- [CVE-2024-20921](#)
- [CVE-2024-20926](#)
- [CVE-2024-20945](#)
- [CVE-2024-20952](#)

1.2.3. Logging 5.8.3

このリリースには、[Logging Bug Fix 5.8.3](#) と [Logging Security Fix 5.8.3](#) が含まれます。

1.2.3.1. バグ修正

- この更新前は、カスタム S3 認証局を読み取るように設定されている場合、ConfigMap の名前または内容が変更されても、Loki Operator は設定を自動的に更新しませんでした。今回の更新により、Loki Operator は ConfigMap への変更を監視し、生成された設定を自動的に更新します。(LOG-4969)
- この更新前は、設定された Loki 出力に有効な URL がない場合にコレクター Pod がクラッシュしていました。今回の更新により、出力は URL 検証の対象となり、問題が解決されました。(LOG-4822)
- この更新前は、Cluster Logging Operator は、サービスアカウントのベアラートークンを使用するためのシークレットを指定していない出力に対してコレクター設定フィールドを生成していました。今回の更新により、出力に認証が不要になり、問題が解決されました。(LOG-4962)
- この更新前は、シークレットが定義されていない場合、出力の `tls.insecureSkipVerify` フィールドの値が `true` に設定されませんでした。今回の更新により、この値を設定するためのシークレットは必要なくなりました。(LOG-4963)
- この更新より前の出力設定では、セキュアではない (HTTP) URL と TLS 認証の組み合わせが許可されていました。今回の更新により、TLS 認証用に設定された出力にはセキュアな (HTTPS) URL が必要になります。(LOG-4893)

1.2.3.2. CVE

- [CVE-2021-35937](#)

- [CVE-2021-35938](#)
- [CVE-2021-35939](#)
- [CVE-2023-7104](#)
- [CVE-2023-27043](#)
- [CVE-2023-48795](#)
- [CVE-2023-51385](#)
- [CVE-2024-0553](#)

1.2.4. Logging 5.8.2

このリリースには、[OpenShift Logging バグ修正リリース 5.8.2](#) が含まれています。

1.2.4.1. バグ修正

- この更新まで、LokiStack ルーラー Pod は、Pod 間通信に使用される HTTP URL の IPv6 Pod IP をフォーマットしなかったため、Prometheus 互換 API を介したルールとアラートのクエリーが失敗していました。この更新により、LokiStack ルーラー Pod は IPv6 Pod IP を角かっこでカプセル化して、問題を解決しました。(LOG-4890)
- この更新まで、開発者コンソールログは現在の namespace を考慮しなかったため、クラスター全体のログアクセスを持たないユーザーのクエリーが拒否されていました。今回の更新により、namespace の追加が修正され、問題が解決されました。(LOG-4947)
- 今回の更新まで、OpenShift Dedicated Web コンソールのロギングビュープラグインではカスタムのノード配置および容認ができませんでした。今回の更新により、カスタムノード配置と容認の定義が OpenShift Dedicated Web コンソールのロギングビュープラグインに追加されました。(LOG-4912)

1.2.4.2. CVE

- [CVE-2022-44638](#)
- [CVE-2023-1192](#)
- [CVE-2023-5345](#)
- [CVE-2023-20569](#)
- [CVE-2023-26159](#)
- [CVE-2023-39615](#)
- [CVE-2023-45871](#)

1.2.5. Logging 5.8.1

このリリースには、[OpenShift Logging のバグ修正リリース 5.8.1](#) および [OpenShift Logging のバグ修正リリース 5.8.1 Kibana](#) が含まれています。

1.2.5.1. 機能拡張

1.2.5.1.1. ログの収集

- この更新により、Vector をコレクターとして設定する際に、サービスアカウントに関連付けられたトークンの代わりにシークレットで指定されたトークンを使用するロジックを Red Hat OpenShift Logging Operator に追加できるようになりました。(LOG-4780)
- この更新により、**BoltDB Shipper** Loki ダッシュボードの名前が **Index** ダッシュボードに変更されました。(LOG-4828)

1.2.5.2. バグ修正

- この更新前は、JSON ログの解析を有効にすると、**ClusterLogForwarder** が、ロールオーバー条件が満たされていない場合でも、空のインデックスを作成していました。今回の更新により、**write-index** が空の場合、**ClusterLogForwarder** はロールオーバーをスキップするようになりました。(LOG-4452)
- この更新前は、Vector が **default** ログレベルを誤って設定していました。この更新により、ログレベル検出のための正規表現 (**regex**) の機能拡張により、正しいログレベルが設定されるようになりました。(LOG-4480)
- この更新前は、インデックスパターンの作成プロセス中に、各ログ出力の初期インデックスからデフォルトのエイリアスが欠落していました。その結果、Kibana ユーザーは OpenShift Elasticsearch Operator を使用してインデックスパターンを作成できませんでした。この更新により、不足しているエイリアスが OpenShift Elasticsearch Operator に追加され、問題が解決されます。Kibana ユーザーは、**{app,infra,audit}-000001** インデックスを含むインデックスパターンを作成できるようになりました。(LOG-4683)
- この更新前は、IPv6 クラスター上の Prometheus サーバーのバインドが原因で、Fluentd コレクター Pod が **CrashLoopBackOff** 状態になっていました。この更新により、コレクターが IPv6 クラスター上で適切に動作するようになりました。(LOG-4706)
- この更新前は、**ClusterLogForwarder** に変更があるたびに Red Hat OpenShift Logging Operator が何度も調整を受けていました。この更新により、Red Hat OpenShift Logging Operator が、調整をトリガーしたコレクターデーモンセットのステータス変更を無視するようになりました。(LOG-4741)
- この更新前は、IBM Power マシン上で Vector ログコレクター Pod が **CrashLoopBackOff** 状態のままになっていました。この更新により、Vector ログコレクター Pod が IBM Power アーキテクチャーマシン上で正常に起動するようになりました。(LOG-4768)
- この更新前は、従来のフォワーダーを使用して内部 LokiStack に転送すると、Fluentd コレクター Pod の使用により SSL 証明書エラーが発生していました。この更新により、ログコレクターサービスアカウントがデフォルトで認証に使用され、関連するトークンと **ca.crt** が使用されるようになりました。(LOG-4791)
- この更新前は、従来のフォワーダーを使用して内部 LokiStack に転送すると、Vector コレクター Pod の使用による SSL 証明書エラーが発生していました。この更新により、ログコレクターサービスアカウントがデフォルトで認証に使用され、関連するトークンと **ca.crt** も使用されるようになりました。(LOG-4852)
- この修正が行われる前は、プレースホルダーに対して1つ以上のホストが評価された後、IPv6 アドレスが正しく解析されませんでした。この更新により、IPv6 アドレスが正しく解析されるようになりました。(LOG-4811)

- この更新前は、HTTP レシーバー入力の監査権限を収集するために **ClusterRoleBinding** を作成する必要がありました。この更新により、**ClusterRoleBinding** を作成する必要がなくなりました。エンドポイントがすでにクラスター認証局に依存しているためです。(LOG-4815)
- この更新前は、Loki Operator がカスタム CA バンドルをルーラー Pod にマウントしませんでした。その結果、アラートルールまたは記録ルールを評価するプロセス中に、オブジェクトストレージへのアクセスが失敗していました。この更新により、Loki Operator がカスタム CA バンドルをすべてのルーラー Pod にマウントするようになりました。ルーラー Pod は、オブジェクトストレージからログをダウンロードして、アラートルールまたは記録ルールを評価できます。(LOG-4836)
- この更新前は、**ClusterLogForwarder** の **inputs.receiver** セクションを削除しても、HTTP 入力サービスとそれに関連するシークレットが削除されませんでした。この更新により、HTTP 入力リソースが、不要な場合に削除されるようになりました。(LOG-4612)
- この更新前は、**ClusterLogForwarder** のステータスに検証エラーが示されていても、出力とパイプラインのステータスにその問題が正確に反映されていませんでした。この更新により、出力、入力、またはフィルターが正しく設定されていない場合に、パイプラインステータスに検証失敗の理由が正しく表示されるようになりました。(LOG-4821)
- この更新前は、時間範囲や重大度などのコントロールを使用する **LogQL** クエリーを変更すると、ラベルマッチャー演算子が正規表現のように定義されて変更されました。この更新により、クエリーの更新時に正規表現演算子が変更されなくなりました。(LOG-4841)

1.2.5.3. CVE

- [CVE-2007-4559](#)
- [CVE-2021-3468](#)
- [CVE-2021-3502](#)
- [CVE-2021-3826](#)
- [CVE-2021-43618](#)
- [CVE-2022-3523](#)
- [CVE-2022-3565](#)
- [CVE-2022-3594](#)
- [CVE-2022-4285](#)
- [CVE-2022-38457](#)
- [CVE-2022-40133](#)
- [CVE-2022-40982](#)
- [CVE-2022-41862](#)
- [CVE-2022-42895](#)
- [CVE-2023-0597](#)
- [CVE-2023-1073](#)

- [CVE-2023-1074](#)
- [CVE-2023-1075](#)
- [CVE-2023-1076](#)
- [CVE-2023-1079](#)
- [CVE-2023-1206](#)
- [CVE-2023-1249](#)
- [CVE-2023-1252](#)
- [CVE-2023-1652](#)
- [CVE-2023-1855](#)
- [CVE-2023-1981](#)
- [CVE-2023-1989](#)
- [CVE-2023-2731](#)
- [CVE-2023-3138](#)
- [CVE-2023-3141](#)
- [CVE-2023-3161](#)
- [CVE-2023-3212](#)
- [CVE-2023-3268](#)
- [CVE-2023-3316](#)
- [CVE-2023-3358](#)
- [CVE-2023-3576](#)
- [CVE-2023-3609](#)
- [CVE-2023-3772](#)
- [CVE-2023-3773](#)
- [CVE-2023-4016](#)
- [CVE-2023-4128](#)
- [CVE-2023-4155](#)
- [CVE-2023-4194](#)
- [CVE-2023-4206](#)
- [CVE-2023-4207](#)

- [CVE-2023-4208](#)
- [CVE-2023-4273](#)
- [CVE-2023-4641](#)
- [CVE-2023-22745](#)
- [CVE-2023-26545](#)
- [CVE-2023-26965](#)
- [CVE-2023-26966](#)
- [CVE-2023-27522](#)
- [CVE-2023-29491](#)
- [CVE-2023-29499](#)
- [CVE-2023-30456](#)
- [CVE-2023-31486](#)
- [CVE-2023-32324](#)
- [CVE-2023-32573](#)
- [CVE-2023-32611](#)
- [CVE-2023-32665](#)
- [CVE-2023-33203](#)
- [CVE-2023-33285](#)
- [CVE-2023-33951](#)
- [CVE-2023-33952](#)
- [CVE-2023-34241](#)
- [CVE-2023-34410](#)
- [CVE-2023-35825](#)
- [CVE-2023-36054](#)
- [CVE-2023-37369](#)
- [CVE-2023-38197](#)
- [CVE-2023-38545](#)
- [CVE-2023-38546](#)
- [CVE-2023-39191](#)

- [CVE-2023-39975](#)
- [CVE-2023-44487](#)

1.2.6. Logging 5.8.0

このリリースには、[OpenShift Logging のバグ修正リリース 5.8.0](#) および [OpenShift Logging のバグ修正リリース 5.8.0 Kibana](#) が含まれています。

1.2.6.1. 非推奨のお知らせ

Logging 5.8 で、Elasticsearch、Fluentd、および Kibana が非推奨となりました。これらは今後の OpenShift Dedicated リリースに同梱される見込みの Logging 6.0 で削除される予定です。Red Hat は、現行リリースのライフサイクルにおいて、該当コンポーネントの「重大」以上の CVE に対するバグ修正とサポートを提供しますが、機能拡張は提供しません。Red Hat OpenShift Logging Operator が提供する Vector ベースのコレクターと、Loki Operator が提供する LokiStack は、ログの収集と保存に推奨される Operator です。Vector および Loki ログスタックは今後強化される予定であるため、すべてのユーザーに対しこのスタックの採用が推奨されます。

1.2.6.2. 機能拡張

1.2.6.2.1. ログの収集

- 今回の更新により、LogFileMetricExporter はデフォルトでコレクターを使用してデプロイされなくなりました。実行中のコンテナによって生成されたログからメトリクスを生成するには、**LogFileMetricExporter** カスタムリソース (CR) を手動で作成する必要があります。**LogFileMetricExporter** CR を作成しない場合、OpenShift Dedicated Web コンソールのダッシュボードで **Produced Logs** のメッセージ **No datapoints found** が表示される場合があります。(LOG-3819)
- この更新により、RBAC で保護された複数の分離された **ClusterLogForwarder** カスタムリソース (CR) インスタンスを任意の namespace にデプロイできるようになります。これにより、独立したグループは、設定を他のコレクターデプロイメントから分離したまま、任意のログを任意の宛先に転送できます。(LOG-1343)



重要

openshift-logging namespace 以外の追加の namespace でマルチクラスターログ転送をサポートするには、すべての namespace を監視するように Red Hat OpenShift Logging Operator を更新する必要があります。この機能は、新しい Red Hat OpenShift Logging Operator バージョン 5.8 インストールでデフォルトでサポートされています。

- この更新により、フロー制御またはレート制限メカニズムを使用して、過剰なログレコードを削除することで収集または転送できるログデータの量を制限できるようになります。入力制限により、低パフォーマンスのコンテナによる Logging の過負荷が防止され、出力制限により、指定されたデータストアへのログ送信レートに上限が設定されます。(LOG-884)
- この更新により、HTTP 接続を検索し、Webhook と呼ばれる HTTP サーバーとしてログを受信するように、ログコレクターを設定できるようになりました。(LOG-4562)
- この更新により、監査ポリシーを設定して、ログコレクターによって転送される Kubernetes および OpenShift API サーバーイベントを制御できるようになりました。(LOG-3982)

1.2.6.2.2. ログのストレージ

- この更新により、LokiStack 管理者は、namespace ごとにログへのアクセスを許可することで、誰がどのログにアクセスできるかをより詳細に制御できるようになりました。(LOG-3841)
- この更新では、Loki Operator によって LokiStack デプロイメントに **PodDisruptionBudget** 設定が導入されました。その結果、OpenShift Dedicated クラスターの再起動中も、取り込みとクエリーパスの可用性が維持され、通常の操作を行えるようになりました。(LOG-3839)
- この更新では、デフォルトのアフィニティーおよび非アフィニティーポリシーのセットを適用することで、既存の LokiStack インストールの信頼性がシームレスに向上しました。(LOG-3840)
- この更新により、ゾーンに障害が発生した場合の信頼性を高めるために、LokiStack で管理者としてゾーン対応のデータレプリケーションを管理できるようになりました。(LOG-3266)
- この更新により、いくつかのワークロードと小規模な取り込みボリューム (最大 100 GB/日) をホストする OpenShift Dedicated クラスターに、新しくサポートされた小規模な LokiStack サイズである 1x.extra-small が導入されました。(LOG-4329)
- この更新により、LokiStack 管理者は公式 Loki ダッシュボードにアクセスして、ストレージのパフォーマンスと各コンポーネントの正常性を検査できるようになりました。(LOG-4327)

1.2.6.2.3. ログコンソール

- この更新により、Elasticsearch がデフォルトのログストアである場合に、Logging Console プラグインを有効にできます。(LOG-3856)
- この更新により、OpenShift Dedicated アプリケーションの所有者は、OpenShift Dedicated バージョン 4.14 以降の OpenShift Dedicated Web コンソールの **Developer** パースペクティブで、アプリケーションログベースのアラートの通知を受信できるようになります。(LOG-3548)

1.2.6.3. 既知の問題

- 現在、Red Hat OpenShift Logging Operator のバージョン 5.8 にアップグレードした後、Splunk ログ転送が機能しない場合があります。この問題は、OpenSSL バージョン 1.1.1 からバージョン 3.0.7 に移行することによって発生します。新しい OpenSSL バージョンでは、デフォルトの動作が変更され、TLS 1.2 エンドポイントへの接続は、RFC 5746 エクステンションを公開しない場合は拒否されます。
回避策として、Splunk HEC (HTTP Event Collector) エンドポイントの前にある TLS 終端ロードバランサーで TLS 1.3 サポートを有効にします。Splunk はサードパーティーシステムであるため、これは Splunk 側から設定する必要があります。
- 現在、HTTP/2 プロトコルでの多重化ストリームの処理には、新しい多重化ストリームを繰り返しリクエストし、直後に **RST_STREAM** フレームを送信してキャンセルできるという欠陥があります。これにより、サーバーのセットアップで余分な作業が発生し、ストリームが切断され、サーバーのリソース消費を原因とするサービス拒否が発生します。現在、この問題に対する回避策はありません。(LOG-4609)
- 現在、FluentD をコレクターとして使用すると、コレクター Pod は OpenShift Dedicated IPv6 対応クラスター上で起動できません。Pod ログでは、**fluentd pod [error]: unexpected error error_class=SocketError error="getaddrinfo: Name or service not known** エラーが生成されます。現在、この問題に対する回避策はありません。(LOG-4706)

- 現在、ログアラートは IPv6 対応クラスターで利用できません。現在、この問題に対する回避策はありません。(LOG-4709)
- 現在、**must-gather** は FIPS 対応クラスター上のログを収集できません。これは、必要な OpenSSL ライブラリーが **cluster-logging-rhel9-operator** で使用できないためです。現在、この問題に対する回避策はありません。(LOG-4403)
- 現在、FIPS 対応クラスターに Logging バージョン 5.8 をデプロイする場合に、FluentD をコレクターとして使用する間は、コレクター Pod を起動できず、**CrashLoopBackOff** ステータスでスタックします。現在、この問題に対する回避策はありません。(LOG-3933)

1.2.6.4. CVE

- [CVE-2023-40217](#)

1.3. LOGGING 5.7



注記

ロギングはインストール可能なコンポーネントとして提供され、コアの OpenShift Dedicated とは異なるリリースサイクルで提供されます。[Red Hat OpenShift Container Platform ライフサイクルポリシー](#) はリリースの互換性を概説しています。



注記

stable チャンネルは、Logging の最新リリースを対象とする更新のみを提供します。以前のリリースの更新を引き続き受信するには、サブスクリプションチャンネルを **stable-x.y** に変更する必要があります。**xy** は、インストールしたログのメジャーバージョンとマイナーバージョンを表します。たとえば、**stable-5.7** です。

1.3.1. Logging 5.7.8

このリリースには、[OpenShift Logging バグ修正リリース 5.7.8](#) が含まれています。

1.3.1.1. バグ修正

- この更新前は、**ClusterLogForwarder** カスタムリソース (CR) の **outputRefs** パラメーターと **inputRefs** パラメーターに同じ名前が使用されている場合、競合が発生することがありました。その結果、コレクター Pod が **CrashLoopBackOff** ステータスになりました。この更新により、出力ラベルとパイプライン名を確実に区別するために、出力ラベルに接頭辞 **OUTPUT_** が含まれるようになりました。(LOG-4383)
- この更新前は、JSON ログパーサーの設定中に、クラスターログオペレーターの **structuredTypeKey** パラメーターまたは **structuredTypeName** パラメーターを設定しなかった場合、無効な設定に関するアラートが表示されませんでした。この更新により、Cluster Logging Operator が設定の問題について通知するようになりました。(LOG-4441)
- この更新前は、Splunk 出力に指定されたシークレットで **hecToken** キーが欠落しているか正しくない場合、Vector がトークンなしでログを Splunk に転送するため、検証が失敗していました。この更新により、**hecToken** キーが見つからないか正しくない場合、検証は失敗し、**A non-empty hecToken entry is required** というエラーメッセージが表示されるようになりました。(LOG-4580)

- この更新前は、ログの **Custom time range** から日付を選択すると、Web コンソールでエラーが発生していました。この更新により、Web コンソールで時間範囲モデルから日付を正常に選択できるようになりました。(LOG-4684)

1.3.1.2. CVE

- [CVE-2023-40217](#)
- [CVE-2023-44487](#)

1.3.2. Logging 5.7.7

このリリースには、[OpenShift Logging バグ修正リリース 5.7.7](#) が含まれています。

1.3.2.1. バグ修正

- この更新前は、FluentD は Vector とは異なる方法で EventRouter によって出力されたログを正規化していました。この更新により、Vector は一貫した形式でログレコードを生成します。(LOG-4178)
- この更新前は、Cluster Logging Operator によって作成されたメトリクスダッシュボードの **FluentD Buffer Availability** グラフに使用されるクエリーに、最小バッファ使用量が表示されるため、エラーがありました。今回の更新により、グラフの最大バッファ使用量が表示され、名前が **FluentD Buffer Age** に変更になりました。(LOG-4555)
- この更新が行われる前は、IPv6 のみまたはデュアルスタックの OpenShift Dedicated クラスターに LokiStack をデプロイすると、LokiStack メンバーリストの登録が失敗していました。その結果、ディストリビューター Pod はクラッシュループに陥りました。この更新により、管理者は **lokistack.spec.hashRing.memberlist.enableIPv6**: 値を **true** に設定することで IPv6 を有効にできるようになり、問題は解決されました。(LOG-4569)
- この更新前は、ログコレクターはデフォルトの設定をもとに、コンテナのログ行を読み取っていました。その結果、ログコレクターはローテーションされたファイルを効率的に読み取ることができませんでした。今回の更新により、読み取りバイト数が増加し、ログコレクターがローテーションされたファイルを効率的に処理できるようになりました。(LOG-4575)
- この更新前は、Event Router 内の未使用のメトリクスにより、過剰なメモリー使用量が原因でコンテナが失敗する原因となっていました。この更新により、未使用のメトリクスが削除され、イベントルーターのメモリー使用量が削減されました。(LOG-4686)

1.3.2.2. CVE

- [CVE-2023-0800](#)
- [CVE-2023-0801](#)
- [CVE-2023-0802](#)
- [CVE-2023-0803](#)
- [CVE-2023-0804](#)
- [CVE-2023-2002](#)
- [CVE-2023-3090](#)

- [CVE-2023-3390](#)
- [CVE-2023-3776](#)
- [CVE-2023-4004](#)
- [CVE-2023-4527](#)
- [CVE-2023-4806](#)
- [CVE-2023-4813](#)
- [CVE-2023-4863](#)
- [CVE-2023-4911](#)
- [CVE-2023-5129](#)
- [CVE-2023-20593](#)
- [CVE-2023-29491](#)
- [CVE-2023-30630](#)
- [CVE-2023-35001](#)
- [CVE-2023-35788](#)

1.3.3. Logging 5.7.6

このリリースには、[OpenShift Logging バグ修正リリース 5.7.6](#) が含まれています。

1.3.3.1. バグ修正

- この更新前は、コレクターはデフォルトの設定をもとに、コンテナのログ行を読み取っていました。その結果、コレクターはローテーションされたファイルを効率的に読み取ることができませんでした。今回の更新により、読み取りバイト数が増加し、コレクターがローテーションされたファイルを効率的に処理できるようになりました。(LOG-4501)
- この更新前は、ユーザーが事前定義されたフィルターを含む URL を貼り付けた場合、一部のフィルターは反映されませんでした。今回の更新により、UI には URL 内のすべてのフィルターを反映します。(LOG-4459)
- この更新前は、Fluentd から Vector に切り替えるときに、カスタムラベルを使用して Loki に転送するとエラーが発生していました。今回の更新により、Vector 設定は Fluentd と同じ方法でラベルをサニタイズし、コレクターが確実に起動してラベルを正しく処理できるようになりました。(LOG-4460)
- この更新前は、Observability Logs コンソールの検索フィールドではエスケープする必要がある特殊文字を使用できませんでした。今回の更新により、クエリーで特殊文字を適切にエスケープするようになりました。(LOG-4456)
- この更新前は、Splunk へのログの送信中に、**Timestamp was not found.** という警告メッセージが表示されました。今回の更新では、タイムスタンプの取得に使用されるログフィールドの名前が変更により上書きされ、警告なしに Splunk に送信されます。(LOG-4413)
- 今回の更新が行われる前は、Vector の CPU とメモリーの使用量が時間の経過とともに増加し

ていました。今回の更新により、Vector 設定に **expire_metrics_secs=60** 設定が含まれるようになり、メトリクスの有効期間を制限し、関連する CPU 使用率とメモリーフットプリントが制限されます。(LOG-4171)

- 今回の更新が行われる前は、LokiStack ゲートウェイは承認されたリクエストを非常に広範囲にキャッシュしていました。その結果、誤った認証結果が発生しました。今回の更新により、LokiStack ゲートウェイは詳細にキャッシュを行うようになり、この問題が解決されました。(LOG-4393)
- この更新前は、Fluentd ランタイムイメージには、実行時には不要なビルダーツールが含まれていました。今回の更新により、ビルダーツールが削除され、問題が解決されました。(LOG-4467)

1.3.3.2. CVE

- [CVE-2023-3899](#)
- [CVE-2023-4456](#)
- [CVE-2023-32360](#)
- [CVE-2023-34969](#)

1.3.4. Logging 5.7.4

このリリースには、[OpenShift Logging バグ修正リリース 5.7.4](#) が含まれています。

1.3.4.1. バグ修正

- この更新前は、ログを CloudWatch に転送するときに、**namespaceUUID** 値が **logGroupName** フィールドに追加されませんでした。この更新では、**namespaceUUID** 値が含まれるため、CloudWatch の **logGroupName** は **logGroupName: vectorcw.b443fb9e-bd4c-4b6a-b9d3-c0097f9ed286** として表示されます。(LOG-2701)
- この更新前は、HTTP 経由でログをクラスター外の宛先に転送する場合、プロキシ URL に正しい認証情報が指定されていたとしても、Vector コレクターはクラスター全体の HTTP プロキシに対して認証できませんでした。この更新により、Vector ログコレクターはクラスター全体の HTTP プロキシに対して認証できるようになりました。(LOG-3381)
- この更新前は、Fluentd コレクターが出力として Splunk を使用して設定されている場合、この設定がサポートされていないため、Operator は失敗していました。今回の更新により、設定検証でサポートされていない出力が拒否され、問題が解決されました。(LOG-4237)
- この更新前は、Vector コレクターが更新されると、AWS Cloudwatch ログと GCP Stackdriver の TLS 設定の **enabled = true** 値によって設定エラーが発生しました。この更新により、これらの出力の **enabled = true** 値が削除され、問題が解決されます。(LOG-4242)
- この更新前は、Vector コレクターに、ログに **thread 'vector-worker' panicked at 'all branches are disabled and there is no else branch', src/kubernetes/reflector.rs:26:9** エラーメッセージでパニックを発生させることができました。今回の更新により、このエラーは解決されました。(LOG-4275)
- この更新前は、Loki Operator の問題により、Operator がそのテナントの追加オプションで設定されている場合、アプリケーションテナントの **alert-manager** 設定が表示されなくなりました。今回の更新により、生成された Loki 設定にはカスタム設定と自動生成された設定の両方が含まれるようになりました。(LOG-4361)

- この更新前は、AWS Cloudwatch 転送で STS を使用した認証に複数のロールが使用されていた場合、最近の更新により認証情報が一意でなくなりました。この更新により、STS ロールと静的認証情報の複数の組み合わせを再び AWS Cloudwatch での認証に使用できるようになりました。(LOG-4368)
- この更新前は、Loki はアクティブなストリームのラベル値をフィルタリングしていましたが、重複を削除しなかったため、Grafana の Label Browser が使用できなくなりました。今回の更新により、Loki はアクティブなストリームの重複するラベル値をフィルターで除外し、問題を解決しました。(LOG-4389)
- **ClusterLogForwarder** カスタムリソース (CR) で **name** フィールドが指定されていないパイプラインが、OpenShift Logging 5.7 へのアップグレード後に機能しなくなりました。今回の更新により、このエラーは解決されました。(LOG-4120)

1.3.4.2. CVE

- [CVE-2022-25883](#)
- [CVE-2023-22796](#)

1.3.5. Logging 5.7.3

このリリースには、[OpenShift Logging バグ修正リリース 5.7.3](#) が含まれています。

1.3.5.1. バグ修正

- この更新前は、OpenShift Dedicated Web コンソール内でログを表示する際に、キャッシュされたファイルが原因でデータがリフレッシュされませんでした。今回の更新により、ブートストラップファイルはキャッシュされなくなり、問題は解決しました。(LOG-4100)
- この更新前は、設定の問題が特定しにくい方法で Loki Operator がエラーをリセットしていました。今回の更新により、設定エラーが解決されるまでエラーが持続するようになりました。(LOG-4156)
- この更新前は、**RulerConfig** カスタムリソース (CR) を変更すると LokiStack ルーラーが再起動しませんでした。今回の更新により、Loki Operator は **RulerConfig** CR の更新後にルーラー Pod を再起動するようになりました。(LOG-4161)
- この更新前は、入力一致ラベル値の **ClusterLogForwarder** 内に / 文字が含まれる場合は、vector コレクターが予期せず終了していました。今回の更新では、一致ラベルを引用符で囲み、コレクターがログを開始および収集できるようにすることで問題が解決されました。(LOG-4176)
- この更新前は、**LokiStack** CR がグローバル制限ではなくテナント制限を定義している場合、Loki Operator が予期せず終了していました。今回の更新では、Loki Operator がグローバル制限なしで **LokiStack** CR を処理できるようになり、問題が解決されました。(LOG-4198)
- この更新前は、提供された秘密鍵がパスフレーズで保護されていた場合、Fluentd はログを Elasticsearch クラスターに送信しませんでした。今回の更新では、Fluentd は Elasticsearch との接続を確立する際に、パスフレーズで保護される秘密鍵を適切に処理するようになりました。(LOG-4258)
- この更新前は、8,000 を超える namespace を持つクラスターの場合、namespace のリストが **http.max_header_size** 設定よりも大きくなるため Elasticsearch がクエリーを拒否していました。今回の更新では、ヘッダーサイズのデフォルト値が引き上げられ、問題が解決されました。(LOG-4277)

- この更新前は、**ClusterLogForwarder** CR 内に / の文字を含むラベル値が原因で、コレクターが予期せず終了していました。今回の更新では、スラッシュがアンダースコアに置き換えられ、問題が解決されました。(LOG-4095)
- この更新前は、unmanaged 状態に設定された Cluster Logging Operator が予期せず終了していました。今回の更新では、**ClusterLogForwarder** CR の調整を開始する前に **ClusterLogging** リソースが適切な Management 状態にあることを確認するチェックが実施されるようになり、問題が解決されました。(LOG-4177)
- この更新前は、OpenShift Dedicated Web コンソール内でログを表示する際に、ヒストグラムをドラッグして時間範囲を選択しても、Pod 詳細内の集約ログビューでは機能しませんでした。今回の更新では、このビューのヒストグラムをドラッグして時間範囲を選択できるようになりました。(LOG-4108)
- この更新前は、OpenShift Dedicated Web コンソール内でログを表示する際に、30 秒を超えるクエリーがタイムアウトになりました。今回の更新では、configmap/logging-view-plugin でタイムアウト値を設定できるようになりました。(LOG-3498)
- この更新前は、OpenShift Dedicated Web コンソール内でログを表示する際に、**more data available** オプションをクリックすると、初回クリック時にのみ、より多くのログエントリーがロードされました。今回の更新では、クリックごとにさらに多くのエントリーが読み込まれるようになりました。(OU-188)
- この更新前は、OpenShift Dedicated Web コンソール内でログを表示する際に、**streaming** オプションをクリックすると、実際のログは表示されず、**streaming logs** メッセージのみが表示されました。今回の更新により、メッセージとログストリームの両方が正しく表示されるようになりました。(OU-166)

1.3.5.2. CVE

- [CVE-2020-24736](#)
- [CVE-2022-48281](#)
- [CVE-2023-1667](#)
- [CVE-2023-2283](#)
- [CVE-2023-24329](#)
- [CVE-2023-26115](#)
- [CVE-2023-26136](#)
- [CVE-2023-26604](#)
- [CVE-2023-28466](#)

1.3.6. Logging 5.7.2

このリリースには、[OpenShift Logging Bug Fix Release 5.7.2](#) が含まれています。

1.3.6.1. バグ修正

- この更新前は、保留中のファイナライザーが存在するため、**openshift-logging** namespace を直接削除できませんでした。今回の更新により、ファイナライザーが使用されなくなり、namespace を直接削除できるようになりました。(LOG-3316)
- この更新より前は、**run.sh** スクリプトが OpenShift Dedicated ドキュメントに従って変更された場合、間違った **chunk_limit_size** 値が表示されていました。ただし、環境変数 **\$BUFFER_SIZE_LIMIT** を介して **chunk_limit_size** を設定すると、スクリプトは正しい値を表示します。今回の更新により、どちらのシナリオでも **run.sh** スクリプトで正しい **chunk_limit_size** 値が表示されるようになりました。(LOG-3330)
- 今回の更新より前は、OpenShift Dedicated Web コンソールのログインビュープラグインはカスタムのノード配置または容認を許可しませんでした。今回の更新により、ログインビュープラグインのノード配置および容認を定義する機能が追加されました。(LOG-3749)
- この更新前は、Fluentd HTTP プラグインを介して DataDog にログを送信しようとする、Cluster Logging Operator で Unsupported Media Type 例外が発生していました。今回の更新により、ユーザーは HTTP ヘッダー Content-Type を設定して、ログ転送用のコンテンツタイプをシームレスに割り当てることができるようになりました。指定された値は、プラグイン内の **content_type** パラメーターに自動的に割り当てられ、ログの送信が正常に行われます。(LOG-3784)
- この更新前は、**ClusterLogForwarder** カスタムリソース (CR) で **detectMultilineErrors** フィールドが **true** に設定されている場合に、PHP マルチラインエラーが別のログエントリーとして記録され、スタックトレースが複数のメッセージに分割されていました。今回の更新により、PHP のマルチラインエラー検出が有効になり、スタックトレース全体が単一のログメッセージに含まれるようになりました。(LOG-3878)
- この更新前は、名前にスペースが含まれる **ClusterLogForwarder** パイプラインが原因で、Vector コレクター Pod が継続的にクラッシュしていました。今回の更新により、パイプラインの名前に含まれるすべてのスペース、ダッシュ (-)、およびドット (.) がアンダースコア (_) に置き換えられました。(LOG-3945)
- この更新前は、**log_forwarder_output** メトリクスに **http** パラメーターが含まれていませんでした。今回の更新で、不足しているパラメーターがメトリクスに追加されました。(LOG-3997)
- この更新前は、コロンで終わる場合に Fluentd は一部のマルチライン JavaScript クライアント例外を特定しませんでした。今回の更新により、Fluentd バッファ名の前にアンダースコアが付けられ、問題は解決しました。(LOG-4019)
- この更新前は、ペイロード内のキーに一致する Kafka 出力トピックに書き込むようにログ転送を設定すると、エラーが原因でログが破棄されていました。今回の更新により、Fluentd のバッファ名の前にアンダースコアが付けられ、問題は解決しました。(LOG-4027)
- この更新前は、LokiStack ゲートウェイはユーザーのアクセス権を適用せずに namespace のラベル値を返していました。今回の更新により、LokiStack ゲートウェイはパーミッションをラベル値のリクエストに適用するようになり、問題は解決しました。(LOG-4049)
- この更新前は、**tls.insecureSkipVerify** オプションが **true** に設定されている場合に、Cluster Logging Operator API にはシークレットにより提供される証明書が必要でした。今回の更新により、そのような場合でも Cluster Logging Operator API はシークレットに証明書の提供を求めなくなりました。次の設定が Operator の CR に追加されました。

```
tls.verify_certificate = false
tls.verify_hostname = false
```

(LOG-3445)

- この更新前は、LokiStack ルート設定が原因で、クエリーの実行時間が 30 秒を超えるとタイムアウトが発生していました。今回の更新で、LokiStack global および per-tenant **queryTimeout** の設定によりルートタイムアウトの設定が影響を受け、問題は解決しました。(LOG-4052)
- この更新前は、修正として **collection.type** のデフォルトを削除したことで、Operator はリソース、ノードの選択、容認に関する非推奨の仕様を受け入れなくなりました。今回の更新により、Operator の動作が変更され、**collection** はなく **collection.logs** の仕様が必ず優先されるようになりました。これは、優先フィールドと非推奨フィールドの両方を使用できる以前の動作とは異なりますが、**collection.type** が指定されている場合に非推奨フィールドを無視します。(LOG-4185)
- この更新前は、ブローカー URL が出力で指定されていない場合、Vector ログコレクターはログを複数の Kafka ブローカーに転送するための TLS 設定を生成しませんでした。今回の更新により、複数のブローカーに対して TLS 設定が適切に生成されるようになりました。(LOG-4163)
- この更新前は、Kafka へのログ転送のパスフレーズを有効にするオプションは使用できませんでした。この制限により、機密情報が公開される可能性があるため、セキュリティリスクが発生していました。今回の更新により、ユーザーは Kafka へのログ転送用にパスフレーズを有効にするシームレスなオプションを使用できるようになりました。(LOG-3314)
- この更新前は、Vector ログコレクターは送信 TLS 接続の **tlsSecurityProfile** 設定を受け入れませんでした。この更新後、Vector は TLS 接続設定を適切に処理します。(LOG-4011)
- この更新前は、**log_forwarder_output_info** メトリクスに利用可能なすべての出力タイプが含まれていませんでした。今回の更新により、以前は含まれていなかった Splunk および Google Cloud Logging データが含まれるようになりました。(LOG-4098)
- この更新前は、**follow_inodes** が **true** に設定されている場合、Fluentd コレクターはファイルローテーション時にクラッシュする可能性があります。今回の更新により、**follow_inodes** 設定が原因でコレクターがクラッシュしなくなりました。(LOG-4151)
- この更新前は、ファイルの追跡方法が原因で、Fluentd コレクターが監視する必要があるファイルを誤って閉じる可能性があります。今回の更新で、追跡パラメーターが修正されました。(LOG-4149)
- この更新前は、Vector コレクターでログを転送し、**ClusterLogForwarder** インスタンスの名前を **audit**、**application**、**infrastructure** のいずれかにすると、コレクター Pod が **CrashLoopBackOff** 状態のままになり、次のエラーがコレクターログに記録されました。

```
ERROR vector::cli: Configuration error. error=redefinition of table transforms.audit for key transforms.audit
```

今回の更新の後には、パイプライン名が予約された入力名と競合しなくなり、パイプラインの名前を **audit**、**application** または **infrastructure** にできます。(LOG-4218)

- この更新前は、Vector コレクターを使用してログを syslog 宛先に転送し、**addLogSource** フラグを **true** に設定すると、転送されたメッセージに **namespace_name=**、**container_name=**、**pod_name=** の空のフィールドが追加されました。今回の更新により、これらのフィールドはジャーナルログに追加されなくなりました。(LOG-4219)
- この更新前は、**structuredTypeKey** が見つからず、**structuredTypeName** が指定されていない場合、ログメッセージは引き続き構造化オブジェクトに解析されていました。今回の更新により、ログの解析が想定どおりになりました。(LOG-4220)

1.3.6.2. CVE

- [CVE-2021-26341](#)
- [CVE-2021-33655](#)
- [CVE-2021-33656](#)
- [CVE-2022-1462](#)
- [CVE-2022-1679](#)
- [CVE-2022-1789](#)
- [CVE-2022-2196](#)
- [CVE-2022-2663](#)
- [CVE-2022-3028](#)
- [CVE-2022-3239](#)
- [CVE-2022-3522](#)
- [CVE-2022-3524](#)
- [CVE-2022-3564](#)
- [CVE-2022-3566](#)
- [CVE-2022-3567](#)
- [CVE-2022-3619](#)
- [CVE-2022-3623](#)
- [CVE-2022-3625](#)
- [CVE-2022-3627](#)
- [CVE-2022-3628](#)
- [CVE-2022-3707](#)
- [CVE-2022-3970](#)
- [CVE-2022-4129](#)
- [CVE-2022-20141](#)
- [CVE-2022-25147](#)
- [CVE-2022-25265](#)
- [CVE-2022-30594](#)
- [CVE-2022-36227](#)

- [CVE-2022-39188](#)
- [CVE-2022-39189](#)
- [CVE-2022-41218](#)
- [CVE-2022-41674](#)
- [CVE-2022-42703](#)
- [CVE-2022-42720](#)
- [CVE-2022-42721](#)
- [CVE-2022-42722](#)
- [CVE-2022-43750](#)
- [CVE-2022-47929](#)
- [CVE-2023-0394](#)
- [CVE-2023-0461](#)
- [CVE-2023-1195](#)
- [CVE-2023-1582](#)
- [CVE-2023-2491](#)
- [CVE-2023-22490](#)
- [CVE-2023-23454](#)
- [CVE-2023-23946](#)
- [CVE-2023-25652](#)
- [CVE-2023-25815](#)
- [CVE-2023-27535](#)
- [CVE-2023-29007](#)

1.3.7. Logging 5.7.1

このリリースには、[OpenShift Logging Bug Fix Release 5.7.1](#) が含まれています。

1.3.7.1. バグ修正

- この更新前は、Cluster Logging Operator Pod ログ内に多数のノイズの多いメッセージが存在するため、ログの可読性が低下し、重要なシステムイベントを識別することが困難になりました。この更新により、Cluster Logging Operator Pod ログ内のノイズの多いメッセージが大幅に削減されることで、この問題が解決されました。(LOG-3482)
- この更新前は、カスタムリソースで別の値が使用されている場合でも、API サーバーは **CollectorSpec.Type** フィールドの値を **Vector** にリセットしていました。この更新で

は、**CollectorSpec.Type** フィールドのデフォルトが削除され、以前の動作が復元されます。
([LOG-4086](#))

- この更新が行われる前は、OpenShift Dedicated Web コンソールでログヒストグラムをクリックしてドラッグしても時間範囲を選択できませんでした。今回の更新により、クリックとドラッグを使用して時間範囲を正常に選択できるようになりました。(LOG-4501)
- この更新が行われる前は、OpenShift Dedicated Web コンソールで **Show Resources** リンクをクリックしても何の効果もありませんでした。この更新では、ログエントリーごとにリソースの表示を切り替えるリソースの表示リンクの機能を修正することで、この問題が解決されました。(LOG-3218)

1.3.7.2. CVE

- [CVE-2023-21930](#)
- [CVE-2023-21937](#)
- [CVE-2023-21938](#)
- [CVE-2023-21939](#)
- [CVE-2023-21954](#)
- [CVE-2023-21967](#)
- [CVE-2023-21968](#)
- [CVE-2023-28617](#)

1.3.8. Logging 5.7.0

このリリースには、[OpenShift Logging Bug Fix Release 5.7.0](#) が含まれています。

1.3.8.1. 機能拡張

今回の更新により、ロギングを有効にして複数行の例外を検出し、それらを1つのログエントリーに再アセンブルできるようになりました。

ロギングを有効にして複数行の例外を検出し、それらを1つのログエントリーに再アセンブルできるようにする場合は、**ClusterLogForwarder** カスタムリソース (CR) に、値が **true** の **detectMultilineErrors** フィールドが含まれていることを確認します。

1.3.8.2. 既知の問題

なし。

1.3.8.3. バグ修正

- 今回の更新以前は、LokiStack の Gateway コンポーネントの **nodeSelector** 属性は、ノードのスケジューリングに影響を与えませんでした。今回の更新により、**nodeSelector** 属性が想定どおりに機能するようになりました。(LOG-3713)

1.3.8.4. CVE

- [CVE-2023-1999](#)
- [CVE-2023-28617](#)

第2章 サポート

このドキュメントで説明されている設定オプションのみがロギングでサポートされています。

他の設定オプションはサポートされていないため、使用しないでください。設定のパラダイムが OpenShift Dedicated 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。Operator は相違点を調整するように設計されているため、このドキュメントで説明されている以外の設定を使用すると、変更は上書きされます。



注記

OpenShift Dedicated 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は、Red Hat OpenShift Logging Operator を **Unmanaged** に設定する必要があります。管理外のロギングインスタンスはサポートされていないため、ステータスを **Managed** に戻すまで更新は受信されません。



注記

ロギングはインストール可能なコンポーネントとして提供され、コアの OpenShift Dedicated とは異なるリリースサイクルで提供されます。[Red Hat OpenShift Container Platform ライフサイクルポリシー](#) はリリースの互換性を概説しています。

Red Hat OpenShift Logging は、アプリケーション、インフラストラクチャー、および監査ログの独自のコレクターおよびノーマライザーです。これは、サポートされているさまざまなシステムにログを転送するために使用することを目的としています。

Logging は、以下ではありません。

- 大規模なログ収集システム
- セキュリティ情報およびイベント監視 (SIEM) に準拠
- 履歴または長期のログの保持または保管
- 保証されたログシンク
- 安全なストレージ - 監査ログはデフォルトでは保存されません

2.1. サポート対象の API カスタムリソース定義

LokiStack は開発中です。現在、一部の API はサポートされていません。

表2.1 Loki API のサポート状態

CustomResourceDefinition (CRD)	ApiVersion	サポートの状態
LokiStack	lokistack.loki.grafana.com/v1	5.5 でサポート
RulerConfig	rulerconfig.loki.grafana/v1	5.7 でサポート
AlertingRule	alertingrule.loki.grafana/v1	5.7 でサポート

CustomResourceDefinition (CRD)	ApiVersion	サポートの状態
RecordingRule	recordingrule.loki.grafana/v1	5.7 でサポート

2.2. 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** カスタムリソース (CR)
- Kibana デプロイメント
- **fluent.conf** ファイル
- Fluentd デモンセット

Elasticsearch デプロイメントファイルを変更するには、OpenShift Elasticsearch Operator を **非管理** 状態に設定する必要があります。

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。

2.3. 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と、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。コンポーネントの **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。

2.4. 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、および各ロギングコンポーネント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 Dedicated と 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。

2.4.1. 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。

ロギングの場合、**must-gather** は次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。

2.4.2. ロギングデータの収集

oc adm must-gather CLI コマンドを使用して、ロギングに関する情報を収集できます。

手順

must-gather でロギング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があるディレクトリーに移動します。
2. ログイメージに対して **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。

第3章 ロギングのトラブルシューティング

3.1. ロギングステータスの表示

Red Hat OpenShift Logging Operator およびその他のロギングコンポーネントのステータスを表示できます。

3.1.1. 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できます。

前提条件

- Red Hat OpenShift Logging Operator と OpenShift Elasticsearch Operator がインストールされている。

手順

- 次のコマンドを実行して、**openshift-logging** プロジェクトに変更します。

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

- 次のコマンドを実行して、**ClusterLogging** インスタンス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:
          collector-2rhqp: ip-10-0-169-13.ec2.internal
          collector-6fgjh: ip-10-0-165-244.ec2.internal
          collector-6l2ff: ip-10-0-128-218.ec2.internal
          collector-54nx5: ip-10-0-139-30.ec2.internal
          collector-flpnn: ip-10-0-147-228.ec2.internal
          collector-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
            - collector-2rhqp
            - collector-54nx5
            - collector-6fgjh
            - collector-6l2ff
            - collector-flpnn
            - collector-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 に関する情報
- ③ Elasticsearch クラスターの健全性 (**green**、**yellow**、または **red**) などの Elasticsearch Pod に関する情報
- ④ Kibana Pod に関する情報

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

以下は、**ClusterLogging** インスタンスの **Status.Nodes** セクションに含まれる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:
```

3.1.2. ロギングコンポーネントのステータスの表示

数多くのロギングコンポーネントのステータスを表示できます。

前提条件

- Red Hat OpenShift Logging Operator と OpenShift Elasticsearch Operator 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. ロギング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. ロギング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----
```

3.2. ログ転送のトラブルシューティング

3.2.1. Fluentd Pod の再デプロイ

ClusterLogForwarder カスタムリソース (CR) の作成時に、Red Hat OpenShift Logging Operator により Fluentd Pod が自動的に再デプロイされない場合は、Fluentd Pod を削除して、強制的に再デプロイできます。

前提条件

- **ClusterLogForwarder** カスタムリソース (CR) オブジェクトを作成している。

手順

- 次のコマンドを実行し、Fluentd Pod を削除して強制的に再デプロイします。

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

3.2.2. Loki レート制限エラーのトラブルシューティング

Log Forwarder API がレート制限を超える大きなメッセージブロックを Loki に転送すると、Loki により、レート制限 (**429**) エラーが生成されます。

これらのエラーは、通常の動作中に発生する可能性があります。たとえば、すでにいくつかのログがあ

このエラーは受信側にも表示されます。たとえば、LokiStack 取り込み Pod で以下を行います。

Loki 取り込みエラーメッセージの例

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream
```

手順

- **LokiStack** CR の **ingestionBurstSize** および **ingestionRate** フィールドを更新します。

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 ①
        ingestionRate: 8 ②
# ...
```

- ① **ingestionBurstSize** フィールドは、ディストリビューターレプリカごとに最大ローカルレート制限サンプルサイズを MB 単位で定義します。この値はハードリミットです。この値を、少なくとも1つのプッシュリクエストで想定される最大ログサイズに設定します。**ingestionBurstSize** 値より大きい単一リクエストは使用できません。
- ② **ingestionRate** フィールドは、1秒あたりに取り込まれるサンプルの最大量 (MB 単位) に対するソフト制限です。ログのレートが制限を超えているにもかかわらず、コレクターがログの送信を再試行すると、レート制限エラーが発生します。合計平均が制限よりも少ない場合に限り、システムは回復し、ユーザーの介入なしでエラーが解決されます。

3.3. ログアラートのトラブルシューティング

次の手順を使用して、クラスター上のログアラートのトラブルシューティングを行うことができます。

3.3.1. Elasticsearch クラスターの健全性ステータスが赤になっている

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。このアラートのトラブルシューティングを行うには、次の手順を使用します。

ヒント

このドキュメントの一部のコマンドは、**\$ES_POD_NAME** シェル変数を使用して Elasticsearch Pod を参照します。このドキュメントからコマンドを直接コピーして貼り付ける場合は、この変数を Elasticsearch クラスターに有効な値に設定する必要があります。

次のコマンドを実行すると、利用可能な Elasticsearch Pod をリスト表示できます。

```
$ oc -n openshift-logging get pods -l component=elasticsearch
```

リストされている Pod のいずれかを選択し、次のコマンドを実行して **\$ES_POD_NAME** 変数を設定します。

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

コマンドで **\$ES_POD_NAME** 変数を使用できるようになりました。

手順

1. 次のコマンドを実行して、Elasticsearch クラスターの健全性をチェックし、クラスターの **status** の色が赤であることを確認します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- health
```

2. 次のコマンドを実行して、クラスターに参加しているノードをリスト表示します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \  
-- es_util --query=_cat/nodes?v
```

3. 次のコマンドを実行して、Elasticsearch Pod をリストし、前のステップのコマンド出力の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. 次のコマンドを実行し、出力を確認して、Elasticsearch にマスターノードが選択されていることを確認します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \  
-- es_util --query=_cat/master?v
```

- b. 次のコマンドを実行して出力を確認し、選択されたマスターノードの Pod ログに問題がないか確認します。

```
$ oc logs <elasticsearch_master_pod_name> -c elasticsearch -n openshift-logging
```

- c. 次のコマンドを実行して出力を確認し、クラスターに参加していないノードのログに問題がないか確認します。

```
$ oc logs <elasticsearch_node_name> -c elasticsearch -n openshift-logging
```

5. すべてのノードがクラスターに参加している場合は、次のコマンドを実行して出力を観察し、クラスターが回復中かどうかを確認します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_cat/recovery?active_only=true
```

コマンドの出力がない場合は、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. 次のコマンドを実行し、出力を確認して、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 $ES_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に見える場合は、次のコマンドを実行して出力を確認し、**cluster.routing.allocation.enable** 値が **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 $ES_POD_NAME \
  -- es_util --query=_cluster/settings?pretty
```

9. **cluster.routing.allocation.enable** 値が **none** に設定されている場合は、次のコマンドを実行して **all** に設定します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_cluster/settings?pretty \
  -X PUT -d '{"persistent": {"cluster.routing.allocation.enable": "all"}}'
```

10. 次のコマンドを実行して出力を確認し、まだ赤いインデックスがあるかどうかを確認します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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. 次のコマンドを実行してキャッシュをクリアします。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=<elasticsearch_index_name>/_cache/clear?pretty
```

- b. 次のコマンドを実行して、割り当ての最大再試行回数を増やします。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=<elasticsearch_index_name>/_settings?pretty \
  -X PUT -d '{"index.allocation.max_retries":10}'
```

- c. 次のコマンドを実行して、すべてのスクロール項目を削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_search/scroll/_all -X DELETE
```

- d. 次のコマンドを実行してタイムアウトを増やします。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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る場合は、インデックスを個別に削除します。

- a. 次のコマンドを実行して、赤いインデックス名を特定します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
-- es_util --query=_cat/indices?v
```

- b. 次のコマンドを実行して、赤いインデックスを削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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. 次のコマンドを実行して、Elasticsearch JVM ヒープの使用率が高いかどうかを確認します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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。CPU 使用率の詳細は、OpenShift Dedicated ドキュメントの「モニタリングダッシュボードの確認」を参照してください。

関連情報

- [モニタリングダッシュボードの確認](#)
- [赤または黄色のクラスターステータスを修正する](#)

3.3.2. Elasticsearch クラスタ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。**ClusterLogging** カスタムリソース (CR) の **nodeCount** 値を調整して、ノード数を増やします。

関連情報

- [赤または黄色のクラスターステータスを修正する](#)

3.3.3. Elasticsearch ノードのディスクの最低水準点に達した

Elasticsearch は、最低水準点に達するノードにシャードを割り当てません。

ヒント

このドキュメントの一部のコマンドは、**\$ES_POD_NAME** シェル変数を使用して Elasticsearch Pod を参照します。このドキュメントからコマンドを直接コピーして貼り付ける場合は、この変数を Elasticsearch クラスターに有効な値に設定する必要があります。

次のコマンドを実行すると、利用可能な Elasticsearch Pod をリスト表示できます。

```
$ oc -n openshift-logging get pods -l component=elasticsearch
```

リストされている Pod のいずれかを選択し、次のコマンドを実行して **\$ES_POD_NAME** 変数を設定します。

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

コマンドで **\$ES_POD_NAME** 変数を使用できるようになりました。

手順

1. 次のコマンドを実行して、Elasticsearch がデプロイされているノードを特定します。

```
$ oc -n openshift-logging get po -o wide
```

2. 次のコマンドを実行して、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 $ES_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、次のコマンドを実行して、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
```

4. コマンド出力で、**Use** 列を確認して、そのノードで使用されているディスクの割合を確認します。

出力例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1    19G  522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme2n1    19G  522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1    19G  528M  19G   3% /elasticsearch/persistent
```

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. 現在の **redundancyPolicy** を確認するには、次のコマンドを実行します。

```
$ oc -n openshift-logging get es elasticsearch \
  -o jsonpath='{.spec.redundancyPolicy}'
```

クラスターで **ClusterLogging** リソースを使用している場合は、次のコマンドを実行します。

```
$ oc -n openshift-logging get cl \
  -o jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

クラスター **redundancyPolicy** 値が **SingleRedundancy** 値より大きい場合は、それを **SingleRedundancy** 値に設定し、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. 次のコマンドを実行して、Elasticsearch 上のすべての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 $ES_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. 次のコマンドを実行してインデックスを削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=<elasticsearch_index_name> -X DELETE
```

3.3.4. Elasticsearch ノードのディスク最高水準点に達した

Elasticsearch は、高基準値に達したノードから、基準値のしきい値制限を超えていないディスク使用量の低いノードにシャードを再配置しようとします。

シャードを特定のノードに割り当てるには、そのノード上のスペースを解放する必要があります。ディスク容量を増やすことができない場合は、新しいデータノードをクラスターに追加するか、クラスターの合計冗長性ポリシーを減らしてみてください。

ヒント

このドキュメントの一部のコマンドは、**\$ES_POD_NAME** シェル変数を使用して Elasticsearch Pod を参照します。このドキュメントからコマンドを直接コピーして貼り付ける場合は、この変数を Elasticsearch クラスターに有効な値に設定する必要があります。

次のコマンドを実行すると、利用可能な Elasticsearch Pod をリスト表示できます。

```
$ oc -n openshift-logging get pods -l component=elasticsearch
```

リストされている Pod のいずれかを選択し、次のコマンドを実行して **\$ES_POD_NAME** 変数を設定します。

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

コマンドで **\$ES_POD_NAME** 変数を使用できるようになりました。

手順

1. 次のコマンドを実行して、Elasticsearch がデプロイされている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 $ES_POD_NAME \
-- es_util --query=_cluster/health?pretty | grep relocating_shards
```

コマンド出力にシャードの再配置が示されている場合は、最高水準点を超過しています。最高水準点のデフォルト値は 90% です。

4. すべてのノードのディスク容量を増やします。ディスク容量を増やすことができない場合は、新しいデータノードをクラスタに追加するか、クラスタの合計冗長性ポリシーを減らしてみてください。

5. 現在の **redundancyPolicy** を確認するには、次のコマンドを実行します。

```
$ oc -n openshift-logging get es elasticsearch \
-o jsonpath='{.spec.redundancyPolicy}'
```

クラスタで **ClusterLogging** リソースを使用している場合は、次のコマンドを実行します。

```
$ oc -n openshift-logging get cl \
-o jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

クラスタ **redundancyPolicy** 値が **SingleRedundancy** 値より大きい場合は、それを **SingleRedundancy** 値に設定し、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. 次のコマンドを実行して、Elasticsearch 上のすべての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 $ES_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. 次のコマンドを実行してインデックスを削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
-- es_util --query=<elasticsearch_index_name> -X DELETE
```

3.3.5. Elasticsearch ノードのディスクがいっぱいの基準値に達した

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。

このアラートのトラブルシューティングを行うには、次の手順を使用します。

ヒント

このドキュメントの一部のコマンドは、**\$ES_POD_NAME** シェル変数を使用して Elasticsearch Pod を参照します。このドキュメントからコマンドを直接コピーして貼り付ける場合は、この変数を Elasticsearch クラスターに有効な値に設定する必要があります。

次のコマンドを実行すると、利用可能な Elasticsearch Pod をリスト表示できます。

```
$ oc -n openshift-logging get pods -l component=elasticsearch
```

リストされている Pod のいずれかを選択し、次のコマンドを実行して **\$ES_POD_NAME** 変数を設定します。

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

コマンドで **\$ES_POD_NAME** 変数を使用できるようになりました。

手順

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. コマンド出力で、**Avail** 列を確認して、そのノード上の空きディスク容量を確認します。

出力例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1    19G  522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme2n1    19G  522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1    19G  528M  19G   3% /elasticsearch/persistent
```

3. すべてのノードのディスク容量を増やします。ディスク容量を増やすことができない場合は、新しいデータノードをクラスターに追加するか、クラスターの合計冗長性ポリシーを減らしてみてください。
4. 現在の **redundancyPolicy** を確認するには、次のコマンドを実行します。

```
$ oc -n openshift-logging get es elasticsearch \
-o jsonpath='{.spec.redundancyPolicy}'
```

クラスターで **ClusterLogging** リソースを使用している場合は、次のコマンドを実行します。

```
$ oc -n openshift-logging get cl \
-o jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

クラスター **redundancyPolicy** 値が **SingleRedundancy** 値より大きい場合は、それを **SingleRedundancy** 値に設定し、この変更を保存します。

5. 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. 次のコマンドを実行して、Elasticsearch 上のすべての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 $ES_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. 次のコマンドを実行してインデックスを削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
-- es_util --query=<elasticsearch_index_name> -X DELETE
```

6. ディスク領域の解放と監視を続けます。使用されているディスク容量が 90% を下回ったら、次のコマンドを実行して、このノードへの書き込みのブロックを解除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
-- es_util --query=_all/_settings?pretty \
-X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

3.3.6. Elasticsearch JVM ヒープ使用量が多い

Elasticsearch ノードの Java 仮想マシン (JVM) ヒープメモリの使用量が 75% を超えています。 [ヒープサイズを増やす](#) ことを検討してください。

3.3.7. 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。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。

3.3.8. 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。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。

3.3.9. Elasticsearch ディスク領域が不足している

現在のディスク使用量に基づいて、Elasticsearch は今後 6 時間以内にディスク容量が不足すると予測されています。このアラートのトラブルシューティングを行うには、次の手順を使用します。

手順

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. コマンド出力で、**Avail** 列を確認して、そのノード上の空きディスク容量を確認します。

出力例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1    19G 522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme2n1    19G 522M  19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1    19G 528M  19G   3% /elasticsearch/persistent
```

3. すべてのノードのディスク容量を増やします。ディスク容量を増やすことができない場合は、新しいデータノードをクラスターに追加するか、クラスターの合計冗長性ポリシーを減らしてみてください。
4. 現在の **redundancyPolicy** を確認するには、次のコマンドを実行します。

```
$ oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```

クラスターで **ClusterLogging** リソースを使用している場合は、次のコマンドを実行します。

```
$ oc -n openshift-logging get cl \
-o jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

クラスター **redundancyPolicy** 値が **SingleRedundancy** 値より大きい場合は、それを **SingleRedundancy** 値に設定し、この変更を保存します。

5. 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. 次のコマンドを実行して、Elasticsearch 上のすべての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 $ES_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. 次のコマンドを実行してインデックスを削除します。

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
-- es_util --query=<elasticsearch_index_name> -X DELETE
```

関連情報

- 赤または黄色のクラスターステータスを修正する

3.3.10. 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。Elasticsearch [ファイル記述子](#) のドキュメントの説明に従って、各ノードの `max_file_descriptors` の値を確認します。

3.4. ELASTICSEARCH 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。

3.4.1. Elasticsearch ログストアのステータスの表示

Elasticsearch ログストアのステータスを表示できます。

前提条件

- Red Hat OpenShift Logging Operator と OpenShift Elasticsearch Operator がインストールされている。

手順

1. 次のコマンドを実行して、**openshift-logging** プロジェクトに変更します。

```
$ 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. 次のコマンドを実行して、Elasticsearch 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. 次のコマンドを実行して、Elasticsearch 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 Elasticsearch 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
- Elasticsearch ログストアのデータノードの数
- Elasticsearch ログストアのノードの合計数

- 保留中のタスクの数
 - Elasticsearch 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)。Elasticsearch ログストアのステータスは、Pod を配置できなかった場合にスケジューラーからの理由を示します。以下の状況に関連したイベントが表示されます。
- Elasticsearch ログストアおよびプロキシコンテナの両方をコンテナが待機している。
 - Elasticsearch ログストアとプロキシコンテナの両方でコンテナが終了した。
 - Pod がスケジュール対象外である。また、いくつかの問題に関する条件も示されています。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** Elasticsearch ログには、**upgradeStatus** のクラスター内のノードが保存されます。
- 5** クラスター内にある Elasticsearch ログストアのクライアント、データ、およびマスター Pod。 **failed**、**notReady**、または **ready** 状態の下にリスト表示されます。

3.4.1.1. 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: {}

```

次のステータスメッセージは、カスタムリソース (CR) の Elasticsearch ログストアのノードセレクターがクラスター内のどのノードと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

```

次のステータスメッセージは、Elasticsearch ログストア CR が存在しない Persistent Volume Claim (PVC) を使用していることを示します。

```

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

```

次のステータスメッセージは、Elasticsearch ログストアクラスターに冗長性ポリシーをサポートするのに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、クラスターにコントロールプレーンノードが多すぎることを示しています。

```

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: StorageStructureChangelgnored
      status: 'True'
      type: StorageStructureChangelgnored
```

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。

StorageClassNameChangelgnored

ストレージクラス名の変更がサポートされていません。

StorageSizeChangelgnored

ストレージサイズの変更がサポートされていません。

StorageStructureChangelgnored

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

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 **ClusterLogging** CR を設定しようとする、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) は作成しません。**StorageStructureChangelgnored** 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。

3.4.2. 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                                   iDjscH7aSUGhldq0LheLBQ 1
1 5 0          0    0
green open .kibana_-377444158_kubeadmin
yBywZ9GfSrKebz5gWBZbjw 3 1 1 0 0 0
green open infra-000001                                z6Dpe__ORgiopEpW6YI44A
3 1 871000     0    874     436
green open app-000001                                  hlrazQCeSISewG3c2VlvsQ
3 1 2453      0    3        1
green open .kibana_1                                   JCitcBMSQxKOvIq6iQW6wg
1 1 0          0    0
green open .kibana_-1595131456_user1
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>

```

3.4.3. Elasticsearch クラスターのステータス

[OpenShift Cluster Manager](#) の **Observe** セクションにあるダッシュボードには、Elasticsearch クラスターのステータスが表示されます。

OpenShift Elasticsearch クラスターのステータスを取得するには、[OpenShift Cluster Manager](#) の **Observe** セクションにあるダッシュボード `<cluster_url>/monitoring/dashboards/grafana-dashboard-cluster-logging` にアクセスします。

Elasticsearch ステータスフィールド

`eo_elasticsearch_cr_cluster_management_state`

Elasticsearch クラスターがマネージドか、マネージド外かを示します。以下に例を示します。

```
eo_elasticsearch_cr_cluster_management_state{state="managed"} 1
eo_elasticsearch_cr_cluster_management_state{state="unmanaged"} 0
```

`eo_elasticsearch_cr_restart_total`

Elasticsearch ノードが証明書の再起動、ローリング再起動、またはスケジュールされた再起動など、再起動した回数を示します。以下に例を示します。

```
eo_elasticsearch_cr_restart_total{reason="cert_restart"} 1
eo_elasticsearch_cr_restart_total{reason="rolling_restart"} 1
eo_elasticsearch_cr_restart_total{reason="scheduled_restart"} 3
```

`es_index_namespaces_total`

Elasticsearch インデックス namespace の総数を表示します。以下に例を示します。

```
Total number of Namespaces.
es_index_namespaces_total 5
```

`es_index_document_count`

各 namespace のレコード数を表示します。以下に例を示します。

```
es_index_document_count{namespace="namespace_1"} 25
es_index_document_count{namespace="namespace_2"} 10
es_index_document_count{namespace="namespace_3"} 5
```

Secret Elasticsearch フィールドが見つからないか、空というメッセージ

Elasticsearch に **admin-cert**、**admin-key**、**logging-es.crt**、または **logging-es.key** ファイルがない場合、ダッシュボードには次の例のようなステータスメッセージが表示されます。

```
message": "Secret \"elasticsearch\" fields are either missing or empty: [admin-cert, admin-key, logging-es.crt, logging-es.key]",
"reason": "Missing Required Secrets",
```

第4章 ロギング

クラスター管理者は、OpenShift Dedicated クラスターにロギングをデプロイし、それを使用してノードシステム監査ログ、アプリケーションコンテナログ、インフラストラクチャーログを収集および集約できます。クラスター上の Red Hat が管理するログストレージなど、選択したログ出力にログを転送できます。デプロイされたログストレージソリューションに応じて、OpenShift Dedicated Web コンソールまたは Kibana Web コンソールでログデータを可視化することもできます。



注記

Kibana Web コンソールは現在非推奨となっており、将来のログリリースで削除される予定です。

OpenShift Dedicated クラスター管理者は、Operator を使用してロギングをデプロイできます。詳細は、[ロギングのインストール](#) を参照してください。

Operator は、ロギングのデプロイ、アップグレード、および保守を担当します。Operator をインストールした後に、**ClusterLogging** カスタムリソース (CR) を作成して、ロギング pod およびロギングをサポートするために必要なその他のリソースをスケジュールできます。**ClusterLogForwarder** CR を作成して、収集するログと、その変換方法および転送先を指定することもできます。



注記

内部 OpenShift Dedicated 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。監査ログをデフォルトの内部 Elasticsearch 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) 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、ログ転送 API を使用する必要があります。

4.1. ロギングアーキテクチャー

ロギングの主なコンポーネントは次のとおりです。

Collector

コレクターは、Pod を各 OpenShift Dedicated ノードにデプロイするデーモンセットです。各ノードからログデータを収集し、データを変換して、設定された出力に転送します。Vector コレクターまたは従来の Fluentd コレクターを使用できます。



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

ログストア

ログストアは分析用のログデータを保存し、ログフォワーダーのデフォルトの出力です。デフォルトの LokiStack ログストア、従来の Elasticsearch ログストアを使用したり、追加の外部ログストアにログを転送したりできます。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

可視化

UI コンポーネントを使用して、ログデータの視覚的表現を表示できます。UI は、保存されたログを検索、クエリー、および表示するためのグラフィカルインターフェイスを提供します。OpenShift Dedicated Web コンソール UI は、OpenShift Dedicated コンソールプラグインを有効にすることで提供されます。



注記

Kibana Web コンソールは現在非推奨となっており、将来のログリリースで削除される予定です。

ロギングはコンテナログとノードログを収集します。これらは次のタイプに分類されます。

アプリケーションログ

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

インフラストラクチャーログ

インフラストラクチャー namespace (**openshift***、**kube***、または **default**) によって生成されたコンテナのログ、およびノードからの journald メッセージ。

監査ログ

`/var/log/audit/audit.log` ファイルに保存されるノード監査システムである auditd によって生成されたログ、**auditd**、**kube-apiserver**、**openshift-apiserver** サービス、および有効な場合は **ovn** プロジェクトからのログ。

関連情報

- [Web コンソールによるログの可視化](#)

4.2. ロギングのデプロイ

管理者は、OpenShift Dedicated Web コンソールまたは OpenShift CLI (**oc**) を使用してロギングをデプロイし、ロギング Operator をインストールできます。Operator は、ロギングのデプロイ、アップグレード、および保守を担当します。

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。

4.2.1. カスタムリソースのロギング

各 Operator によって実装されたカスタムリソース (CR) YAML ファイルを使用して、ロギングのデプロイメントを設定できます。

Red Hat OpenShift Logging Operator:

- **ClusterLogging (CL) - Operator** をインストールした後に、**ClusterLogging** カスタムリソース (CR) を作成して、ロギング pod およびロギングをサポートするために必要なその他のリソースをスケジュールできます。**ClusterLogging** CR はコレクターとフォワーダーをデプロイします。現在、これらはどちらも各ノードで実行されているデーモンセットによって実装されています。Red Hat OpenShift Logging Operator は **ClusterLogging** CR を監視し、それに応じてロギングのデプロイメントを調整します。
- **ClusterLogForwarder (CLF)**- ユーザー設定ごとにログを転送するためのコレクター設定を生成します。

Loki Operator:

- **LokiStack** - Loki クラスタをログストアとして制御し、OpenShift Dedicated 認証統合を使用して Web プロキシを制御して、マルチテナンシーを強制します。

OpenShift Elasticsearch Operator:



注記

これらの CR は、OpenShift Elasticsearch Operator によって生成および管理されます。Operator によって上書きされない限り、手動で変更はできません。

- **Elasticsearch** - Elasticsearch インスタンスをデフォルトのログストアとして設定し、デプロイします。
- **Kibana** - ログの検索、クエリー、表示を実行するために Kibana インスタンスを設定し、デプロイします。

4.3. OPENSIFT DEDICATED に推奨される CLOUDWATCH

Red Hat は、ロギングのニーズに合わせて AWS CloudWatch ソリューションを使用することを推奨します。

4.3.1. ロギングの要件

独自のログスタックをホストするには、大量のコンピュートリソースとストレージが必要です。これらは、クラウドサービスのクォータに依存している場合があります。コンピュートリソースの要件は 48 GB 以上からですが、ストレージの要件は 1600 GB 以上になる可能性があります。ロギングスタックはワーカーノードで実行されるため、使用可能なワークロードリソースが減少します。これらのことから、独自のログスタックをホストすると、クラスタの運用コストが増加します。

次のステップ

- 手順は、[ログの Amazon CloudWatch への転送](#) を参照してください。

4.3.2. JSON OpenShift Dedicated Logging について

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

4.3.3. Kubernetes イベントの収集および保存

OpenShift Dedicated イベントルーターは、Kubernetes イベントを監視し、それらを OpenShift Dedicated 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、[Kubernetes イベントの収集および保存](#) を参照してください。

4.3.4. OpenShift Dedicated Logging のトラブルシューティングについて

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

4.3.5. 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、[フィールドのエクスポート](#) を参照してください。

4.3.6. イベントのルーティングについて

イベントルーターは、ロギングによって収集できるように OpenShift Dedicated 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 Dedicated 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](#) を参照してください。

第5章 ロギングのインストール

Red Hat OpenShift Logging Operator をインストールしてロギングをデプロイできます。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。



注記

ロギングはインストール可能なコンポーネントとして提供され、コアの OpenShift Dedicated とは異なるリリースサイクルで提供されます。[Red Hat OpenShift Container Platform ライフサイクルポリシー](#) はリリースの互換性を概説しています。



重要

新規インストールの場合は、Vector と LokiStack を使用してください。Elasticsearch と Fluentd は非推奨となり、今後のリリースで削除される予定です。

5.1. WEB コンソールを使用して RED HAT OPENSIFT LOGGING OPERATOR をインストールする

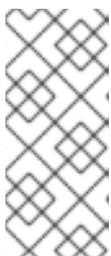
Red Hat OpenShift Logging Operator は、OpenShift Dedicated Web コンソールを使用してインストールできます。

前提条件

- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスできる。

手順

1. OpenShift Dedicated Web コンソールで、**Operators** → **OperatorHub** をクリックします。
2. **Filter by keyword** ボックスに **OpenShift Logging** と入力します。
3. 利用可能な Operator のリストから **Red Hat OpenShift Logging** を選択し、**Install** をクリックします。
4. **Update channel** として **stable-5.y** を選択します。



注記

stable チャンネルは、Logging の最新リリースを対象とする更新のみを提供します。以前のリリースの更新を引き続き受信するには、サブスクリプションチャンネルを **stable-x.y** に変更する必要があります。**xy** は、インストールしたログのメジャーバージョンとマイナーバージョンを表します。たとえば、**stable-5.7** です。

5. **Version** を選択します。
6. **Installation Mode** で **A specific namespace on the cluster** が選択されていることを確認します。

7. **Operator recommended namespace** が **Installed Namespace** で **openshift-logging** になっていることを確認します。
8. **Update approval** 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。
9. **Console plugin** で **Enable** または **Disable** を選択します。
10. **Install** をクリックします。

検証

1. **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。
2. **Status** 列に、緑色のチェックマークおよび **InstallSucceeded** と、**Up to date** というテキストが表示されていることを確認します。



重要

インストールが完了する前に、Operator に **Failed** ステータスが表示される場合があります。**InstallSucceeded** メッセージが表示されて Operator のインストールが完了した場合は、ページを更新します。

Operator がインストール済みとして表示されない場合は、次のトラブルシューティングオプションのいずれかを選択します。

- **Operators** → **Installed Operators** ページに移動し、**Status** 列でエラーまたは失敗の有無を確認します。
- **Workloads** → **Pods** ページに移動し、**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。

5.2. WEB コンソールを使用して CLUSTERLOGGING オブジェクトを作成する

Logging Operator をインストールした後、**ClusterLogging** カスタムリソースを作成して、クラスターのログストレージ、可視化、およびログコレクターを設定する必要があります。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- OpenShift Dedicated Web コンソールの **Administrator** パースペクティブにアクセスできる。

手順

1. **Custom Resource Definitions** ページに移動します。
2. **Custom Resource Definitions** ページで、**ClusterLogging** をクリックします。

3. **Custom Resource Definition details** ページで、**Actions** メニューから **View Instances** を選択します。
4. **ClusterLoggings** ページで、 **Create ClusterLogging** をクリックします。
5. **collection** セクションで、 **Collector Implementation** を選択します。



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

6. **logStore** セクションで、タイプを選択します。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

7. **Create** をクリックします。

5.3. CLI を使用して RED HAT OPENSIFT LOGGING OPERATOR をインストールする

OpenShift CLI (**oc**) を使用して、Red Hat OpenShift Logging Operator をインストールできます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **Namespace** オブジェクトを、YAML ファイルとして作成します。

Namespace オブジェクトの例

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

- 1 ロギングバージョン 5.7 以前の場合、**openshift-logging** を namespace の名前に指定する必要があります。ロギングバージョン 5.8 以降の場合、任意の名前を使用できます。
- 2 次のコマンドを実行して、**Namespace** オブジェクトを適用します。

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

- 3 **OperatorGroup** オブジェクトを、YAML ファイルとして作成します。

OperatorGroup オブジェクトのサンプル

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

- 1 2 ロギングバージョン 5.7 以前の場合、**openshift-logging** namespace を指定する必要があります。ロギングバージョン 5.8 以降の場合、任意の namespace を使用できます。

- 4 以下のコマンドを実行して **OperatorGroup** オブジェクトを適用します。

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

- 5 Red Hat OpenShift Logging Operator に namespace をサブスクライブするための **Subscription** オブジェクトを作成します。

Subscription オブジェクトの例

```
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 ロギングバージョン 5.7 以前の場合、**openshift-logging** namespace を指定する必要があります。ロギングバージョン 5.8 以降の場合、任意の namespace を使用できます。
- 2 チャンネルとして **stable** または **stable-x.y** を指定します。
- 3 **redhat-operators** を指定します。OpenShift Dedicated 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) の設定時に作成した **CatalogSource** オブジェクトの名前を指定します。

6. 次のコマンドを実行して、サブスクリプションを適用します。

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

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

検証

1. 以下のコマンドを実行します。

```
$ oc get csv -n <namespace>
```

2. 出力を観察し、Red Hat OpenShift Logging Operator が namespace に存在することを確認します。

出力例

```

NAMESPACE                                NAME                                DISPLAY
VERSION      REPLACES  PHASE
...
openshift-logging      clusterlogging.5.8.0-202007012112.p0
OpenShift Logging      5.8.0-202007012112.p0      Succeeded
...

```

5.4. CLI を使用して CLUSTERLOGGING オブジェクトを作成する

このデフォルトのログイング設定は、幅広い環境をサポートします。可能な変更については、コンポーネントのチューニングと設定に関するトピックを参照してください。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- ログストア用の OpenShift Elasticsearch Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **ClusterLogging** オブジェクトを YAML ファイルとして作成します。

ClusterLogging オブジェクトの例

```

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:
  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) に設定することが求められる場合があります。ただし、管理外のデプロイメントは OpenShift Logging がマネージド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。指定しないと、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。このリスト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と、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 をスケールリングし、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** を参照してください。
- 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 Dedicated 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専用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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
cluster-logging-operator	1/1	1	1	18h
elasticsearch-cd-x6kdekli-1	1/1	1	1	6m54s
elasticsearch-cdm-x6kdekli-1	1/1	1	1	18h
elasticsearch-cdm-x6kdekli-2	1/1	1	1	6m49s
elasticsearch-cdm-x6kdekli-3	1/1	1	1	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。

検証

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できます。

- 次のコマンドを実行して、Pod を一覧表示します。

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

次のリストのような、ロギングコンポーネントの Pod を観察します。

出力例

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

5.5. インストール後のタスク

Red Hat OpenShift Logging Operator をインストールした後、**ClusterLogging** カスタムリソース (CR) を作成および変更してデプロイメントを設定できます。

ヒント

Elasticsearch ログストアを使用していない場合は、内部 Elasticsearch **logStore** および Kibana **visualization** コンポーネントを **ClusterLogging** カスタムリソース (CR) から削除できます。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。[Elasticsearch ログストアを使用しない場合の未使用コンポーネントの削除](#) を参照してください。

5.5.1. ClusterLogging カスタムリソースについて

ロギング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。

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 ❶
  namespace: openshift-logging ❷
spec:
  managementState: Managed ❸
# ...
```

- ❶ CR の名前は **instance** である必要があります。
- ❷ CR は **openshift-logging** namespace にインストールされる必要があります。
- ❸ Red Hat OpenShift Logging Operator の管理状態。状態が **Unmanaged** に設定されている場合、Operator はサポート対象外となり、更新は受け取りません。

5.5.2. ログストレージの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログストレージのタイプを設定できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator と内部ログストア (LokiStack または Elasticsearch) がインストールされている。
- **ClusterLogging** CR が作成されている。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

手順

1. **ClusterLogging** CR の **logStore** 仕様を変更します。

ClusterLogging CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
logStore:
  type: <log_store_type> ①
  elasticsearch: ②
    nodeCount: <integer>
    resources: {}
    storage: {}
    redundancyPolicy: <redundancy_type> ③
  lokistack: ④
    name: {}
# ...
```

- ① ログストアのタイプを指定します。これは **lokistack** または **elasticsearch** のいずれかです。
- ② Elasticsearch ログストアの任意の設定オプション。
- ③ 冗長性のタイプを指定します。この値には、**ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy**、または **FullRedundancy** を指定できます。

4 LokiStack の任意の設定オプション。

LokiStack をログストアとして指定する ClusterLogging CR の例

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

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

5.5.3. ログコレクターの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログコレクターのタイプを設定できます。



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator がインストールされている。
- **ClusterLogging** CR が作成されている。

手順

1. **ClusterLogging** CR の **collection** 仕様を変更します。

ClusterLogging CR の例

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

```
# ...
collection:
  type: <log_collector_type> ❶
  resources: {}
  tolerations: {}
# ...
```

- ❶ ログングに使用するログコレクターのタイプ。これは、**vector** または **fluentd** にすることができます。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

5.5.4. ログビジュアライザーの設定

ClusterLogging カスタムリソース (CR) を変更することで、ログングで使用するログビジュアライザーのタイプを設定できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator がインストールされている。
- **ClusterLogging** CR が作成されている。



重要

可視化に OpenShift Dedicated Web コンソールを使用する場合は、ログングコンソールプラグインを有効にする必要があります。"Web コンソールによるログの可視化" に関するドキュメントを参照してください。

手順

1. **ClusterLogging** CR の **visualization** 仕様を変更します。

ClusterLogging CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
visualization:
  type: <visualizer_type> ❶
  kibana: ❷
  resources: {}
  nodeSelector: {}
  proxy: {}
```

```

replicas: {}
tolerations: {}
ocpConsole: ③
logsLimit: {}
timeout: {}
# ...

```

- ① ロギングに使用するビジュアライザーのタイプ。これは、**kibana** または **ocp-console** のいずれかです。Kibana コンソールは Elasticsearch ログストレージを使用するデプロイメントとのみ互換性があり、OpenShift Dedicated コンソールは LokiStack デプロイメントとのみ互換性があります。
- ② Kibana コンソールの任意の設定。
- ③ OpenShift Dedicated Web コンソールの任意の設定。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

5.5.5. ネットワークの分離が有効になっ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ラグインによって、ネットワークの分離が実施される場合があります。その場合は、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。ロギングは、**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 Dedicated は、ネットワークプラグインとして、OpenShift SDN と OVN-Kubernetes の2つのサポートされた選択肢を提供します。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。

subnet

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6章 ロギングの更新

ロギングの更新には、マイナーリリース更新 (5.yz) とメジャーリリース更新 (5.y) の2種類があります。

6.1. マイナーリリースの更新

Automatic 更新承認オプションを使用してロギング Operator をインストールした場合、Operator はマイナーバージョンの更新を自動的に受け取ります。手動での更新手順を完了する必要はありません。

Manual 更新承認オプションを使用してロギング Operators をインストールした場合は、マイナーバージョンの更新を手動で承認する必要があります。詳細は、[保留中の Operator 更新の手動承認](#) を参照してください。

6.2. メジャーリリースの更新

メジャーバージョンを更新するには、いくつかの手順を手動で完了する必要があります。

メジャーリリースバージョンの互換性とサポート情報については、[OpenShift Operator Life Cycles](#) を参照してください。

6.3. すべての NAMESPACE を監視するための RED HAT OPENSIFT LOGGING OPERATOR のアップグレード

Logging 5.7 以前のバージョンでは、Red Hat OpenShift Logging Operator は **openshift-logging** namespace のみを監視します。Red Hat OpenShift Logging Operator でクラスター上のすべての namespace を監視する場合は、Operator を再デプロイする必要があります。以下の手順を実行して、ロギングコンポーネントを削除せずに Operator を再デプロイします。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- 管理者権限がある。

手順

1. 次のコマンドを実行して、サブスクリプションを削除します。

```
$ oc -n openshift-logging delete subscription <subscription>
```

2. 以下のコマンドを実行して Operator グループを削除します。

```
$ oc -n openshift-logging delete operatorgroup <operator_group_name>
```

3. 次のコマンドを実行して、クラスターサービスバージョン (CSV) を削除します。

```
$ oc delete clusterserviceversion cluster-logging.<version>
```

4. 「ロギングのインストール」ドキュメントに従って、Red Hat OpenShift Logging Operator を再デプロイします。

検証

- **OperatorGroup** リソースの **targetNamespaces** フィールドが存在しないか、空の文字列に設定されていることを確認します。
これを行うには、次のコマンドを実行して出力を検査します。

```
$ oc get operatorgroup <operator_group_name> -o yaml
```

出力例

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-logging-f52cn
  namespace: openshift-logging
spec:
  upgradeStrategy: Default
status:
  namespaces:
  - ""
# ...
```

6.4. RED HAT OPENSIFT LOGGING OPERATOR の更新

Red Hat OpenShift Logging Operator を新しいメジャーリリースバージョンに更新するには、Operator サブスクリプションの更新チャンネルを変更する必要があります。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスでき、**Administrator** パースペクティブが表示されている。

手順

1. **Operators** → **Installed Operators** に移動します。
2. **openshift-logging** プロジェクトを選択します。
3. **Red Hat OpenShift Logging Operator** をクリックします。
4. **Subscription** をクリックします。**Subscription details** セクションで、**Update channel** リンクをクリックします。このリンクテキストは、現在の更新チャンネルによっては **stable** または **stable-5.y** である可能性があります。
5. **Change Subscription Update Channel** ウィンドウで、最新のメジャーバージョン更新チャンネル **stable-5.y** を選択し、**Save** をクリックします。**cluster-logging.v5.y.z** バージョンに注意してください。

検証

1. 数秒待ってから **Operators** → **Installed Operators** をクリックします。Red Hat OpenShift Logging Operator のバージョンが最新の **cluster-logging.v5.y.z** バージョンと一致することを確認します。
2. **Operators** → **Installed Operators** ページで、**Status** フィールドが **Succeeded** を報告するのを待機します。

6.5. LOKI OPERATOR の更新

Loki Operator を新しいメジャーリリースバージョンに更新するには、Operator サブスクリプションの更新チャンネルを変更する必要があります。

前提条件

- Loki Operator がインストールされている。
- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスでき、**Administrator** パースペクティブが表示されている。

手順

1. **Operators** → **Installed Operators** に移動します。
2. **openshift-operators-redhat** プロジェクトを選択します。
3. **Loki Operator** をクリックします。
4. **Subscription** をクリックします。**Subscription details** セクションで、**Update channel** リンクをクリックします。このリンクテキストは、現在の更新チャンネルによっては **stable** または **stable-5.y** である可能性があります。
5. **Change Subscription Update Channel** ウィンドウで、最新のメジャーバージョン更新チャンネル **stable-5.y** を選択し、**Save** をクリックします。**loki-operator.v5.y.z** バージョンに注意してください。

検証

1. 数秒待ってから **Operators** → **Installed Operators** をクリックします。Loki Operator のバージョンが最新の **loki-operator.v5.yz** バージョンと一致していることを確認します。
2. **Operators** → **Installed Operators** ページで、**Status** フィールドが **Succeeded** を報告するのを待機します。

6.6. OPENSIFT ELASTICSEARCH OPERATOR の更新

OpenShift Elasticsearch Operator を現在のバージョンに更新するには、サブスクリプションを変更する必要があります。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

前提条件

- Elasticsearch をデフォルトのログストアとして使用し、Kibana を UI として使用している場合は、Red Hat OpenShift Logging Operator を更新する前に OpenShift Elasticsearch Operator を更新します。



重要

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。この問題を解決するには、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。

- Logging のステータスが正常である。
 - すべての Pod のステータスは **ready** です。
 - Elasticsearch クラスタが正常である。
- [Elasticsearch および Kibana データのバックアップが作成されている](#)。
- 管理者権限がある。
- 検証手順のために OpenShift CLI (**oc**) がインストールされている。

手順

1. Red Hat Hybrid Cloud Console で、**Operators → Installed Operators** をクリックします。
2. **openshift-operators-redhat** プロジェクトを選択します。
3. **OpenShift Elasticsearch Operator** をクリックします。
4. **Subscription → Channel** をクリックします。
5. **Change Subscription Update Channel** ウィンドウで **stable-5.y** を選択し、**Save** をクリックします。**elasticsearch-operator.v5.y.z** バージョンに注意してください。
6. 数秒待ってから **Operators → Installed Operators** をクリックします。OpenShift Elasticsearch Operator のバージョンが最新の **elasticsearch-operator.v5.y.z** バージョンと一致していることを確認します。
7. **Operators → Installed Operators** ページで、**Status** フィールドが **Succeeded** を報告するのを待機します。

検証

1. 次のコマンドを入力し、出力を確認して、すべての Elasticsearch Pod が **Ready** ステータスになってい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
```

2. 以下のコマンドを入力して出力を確認し、Elasticsearch クラスターのステータスが **green** 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
```

出力例

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

3. 次のコマンドを入力し、出力を確認して、Elasticsearch cron ジョブが作成されたことを確認します。

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

```
$ oc get cronjob
```

出力例

```
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
```

4. 次のコマンドを入力し、出力を確認して、ログストアが正しいバージョンに更新され、インデックスが **緑色** になってい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6.1 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
3 1 226717 0 297 148
rtDSzoqsSl6saisSK7Au1Q
green open infra-000012
RSf_kUwDSR2xEuKRZMPqZQ 3 1 227623 0 295 147
green open .kibana_7
1 1 4 0 0 0
1SJdCqlZTPWIIAaOUd78yg
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_GVQVQZ6Yh9lg 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
goREK1QUKIQAIVkWWaQ 3 1 226719 0 295 147
1-
green open .security
zeT65uOuRTKZMjg_bbUc1g
1 1 5 0 0 0
green open .kibana-377444158_kubeadmin
mRZQO84K0gUQ 3 1 1 0 0 0
wvMhDwJkR-
green open infra-000006
KBSXGQKiO7hdapDE23g 3 1 226676 0 295 147
5H-
green open infra-000001
bSxSWR5xYZB6IVg 3 1 341800 0 443 220
eH53BQ-
green open .kibana-6
RVp7TemSSemGJcsSUumf3A 1 1 4 0 0 0
green open infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1 226100 0 293 146
green open app-000001
axSAFfONQDmKwatjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLtStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002
ewmbYg 3 1 228994 0 296 148
Hz6WvINtTvKcQzw-
green open infra-000013
jraYtanylGw 3 1 228166 0 298 148
KR9mMFUpQl-
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0

```

5. 次のコマンドを入力し、出力を確認して、ログビジュアライザーが正しいバージョン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6.2 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
  }
]
```

第7章 ログの可視化

7.1. ログの可視化について

デプロイされたログストレージソリューションに応じて、OpenShift Dedicated Web コンソールまたは Kibana Web コンソールでログデータを可視化できます。Kibana コンソールは ElasticSearch ログストアで使用でき、OpenShift Dedicated Web コンソールは ElasticSearch ログストアまたは LokiStack で使用できます。



注記

Kibana Web コンソールは現在非推奨となっており、将来のログリリースで削除される予定です。

7.1.1. ログビジュアライザーの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログビジュアライザーのタイプを設定できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator がインストールされている。
- **ClusterLogging** CR が作成されている。



重要

可視化に OpenShift Dedicated Web コンソールを使用する場合は、ロギングコンソールプラグインを有効にする必要があります。"Web コンソールによるログの可視化" に関するドキュメントを参照してください。

手順

1. **ClusterLogging** CR の **visualization** 仕様を変更します。

ClusterLogging CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  # ...
spec:
  # ...
  visualization:
    type: <visualizer_type> ❶
    kibana: ❷
      resources: {}
      nodeSelector: {}
      proxy: {}
```

```

replicas: {}
tolerations: {}
ocpConsole: ❸
logsLimit: {}
timeout: {}
# ...

```

- ❶ ロギングに使用するビジュアライザーのタイプ。これは、**kibana** または **ocp-console** のいずれかです。Kibana コンソールは Elasticsearch ログストレージを使用するデプロイメントとのみ互換性があり、OpenShift Dedicated コンソールは LokiStack デプロイメントとのみ互換性があります。
- ❷ Kibana コンソールの任意の設定。
- ❸ OpenShift Dedicated Web コンソールの任意の設定。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

7.1.2. リソースのログの表示

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。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。

ヒント

ログの取得と表示のエクスペリエンスを強化するには、ロギングをインストールします。ロギングは、ノードシステムの監査ログ、アプリケーションコンテナログ、およびインフラストラクチャーログなどの OpenShift Dedicated クラスタからのすべてのログを専用のログストアに集約します。その後、Kibana コンソールまたは OpenShift Dedicated Web コンソールを介してログデータをクエリー、検出、可視化できます。リソースログはロギングのログストアにアクセスしません。

7.1.2.1. リソースログの表示

OpenShift CLI (**oc**) および Web コンソールでさまざまな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**) へのアクセスがある。

手順 (UI)

1. OpenShift Dedicated 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。このような場合は、リソースの **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: コンテナの名前を指定します。Pod に複数のコンテナがある場合は、コンテナ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。

7.2. WEB コンソールによるログの可視化

ロギングコンソールプラグインを設定すると、OpenShift Dedicated Web コンソールを使用してログデータを可視化できます。

ロギングのインストール時にプラグインを設定する方法については、[Web コンソールを使用したロギングのインストール](#) を参照してください。

すでにロギングをインストールしており、プラグインを設定する場合は、次のいずれかの手順を使用します。

7.2.1. Red Hat OpenShift Logging Operator をインストールした後のロギングコンソールプラグインの有効化

ロギングコンソールプラグインは Red Hat OpenShift Logging Operator のインストール中に有効にできますが、プラグインを無効にして Red Hat OpenShift Logging Operator をインストールした場合も、プラグインを有効にすることができます。

前提条件

- 管理者権限がある。
- Red Hat OpenShift Logging Operator がインストールされており、**Console plugin** で **Disabled** が選択されている。
- OpenShift Dedicated Web コンソールにアクセスできる。

手順

1. OpenShift Dedicated Web コンソールの **Administrator** パースペクティブで、**Operators** → **Installed Operators** に移動します。
2. **Red Hat OpenShift Logging** をクリックします。Operator の **Details** ページが表示されます。
3. **Details** ページで、**Console plugin** オプションの **Disabled** をクリックします。
4. **Console plugin enablement** ダイアログで、**Enable** を選択します。
5. **Save** をクリックします。
6. **Console plugin** オプションに **Enabled** と表示されていることを確認します。
7. 変更が適用されると、Web コンソールにポップアップウィンドウが表示されます。ウィンドウに Web コンソールのリロードを求めるプロンプトが表示されます。ポップアップウィンドウが表示されたら、ブラウザを更新して変更を適用します。

7.2.2. Elasticsearch ログストアと LokiStack がインストールされている場合のロギングコンソールプラグインの設定

ロギングバージョン 5.8 以降では、Elasticsearch ログストアがデフォルトのログストアであるが、LokiStack もインストールされている場合は、次の手順を使用してロギングコンソールプラグインを有効にできます。

前提条件

- 管理者権限がある。
- Red Hat OpenShift Logging Operator、OpenShift Elasticsearch Operator、および Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- **ClusterLogging** カスタムリソース (CR) が作成されている。

手順

1. 次のコマンドを実行して、ロギングコンソールプラグインが有効になっていることを確認します。

```
$ oc get consoles.operator.openshift.io cluster -o yaml |grep logging-view-plugin \
|| oc patch consoles.operator.openshift.io cluster --type=merge \
--patch '{"spec": {"plugins": ["logging-view-plugin"]}]'
```

2. 次のコマンドを実行して **.metadata.annotations.logging.openshift.io/ocp-console-migration-target: lokistack-dev** アノテーションを **ClusterLogging** CR に追加します。

```
$ oc patch clusterlogging instance --type=merge --patch \
'{"metadata": {"annotations": {"logging.openshift.io/ocp-console-migration-target":
"lokistack-dev" }}}' \
-n openshift-logging
```

出力例

```
clusterlogging.logging.openshift.io/instance patched
```

検証

- 次のコマンドを実行し、出力を確認して、アノテーションが正常に追加されたことを確認します。

```
$ oc get clusterlogging instance \
-o=jsonpath='{.metadata.annotations.logging\.openshift\.io/ocp-console-migration-target}' \
-n openshift-logging
```

出力例

```
"lokistack-dev"
```

これで、ロギングコンソールプラグイン Pod がデプロイされました。ロギングデータを表示するには、OpenShift Dedicated Web コンソールに移動し、**Observe** → **Logs** ページを表示します。

7.3. クラスタダッシュボードの表示

[OpenShift Cluster Manager](#) の **Logging/Elasticsearch Nodes** および **Openshift Logging** ダッシュボードには、Elasticsearch インスタンスおよび個々の Elasticsearch ノードに関する詳細な情報が含まれており、問題の予防と診断に使用できます。

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。

7.3.1. Elastisearch および Openshift Logging ダッシュボードへのアクセス

[OpenShift Cluster Manager](#) 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 Dedicated Red Hat Hybrid Cloud Console で、**Observe** → **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は、表示する必要がある 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、データを表示するさまざまな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。

ダッシュボードチャートの詳細は、[OpenShift Logging ダッシュボードについて](#) および [Logging/Elasticsearch Nodes ダッシュボードについて](#) を参照してください。

7.3.2. 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7.1 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: <ul style="list-style-type: none"> ● 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。

メトリクス	説明
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)	<ul style="list-style-type: none"> ● 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。 <p>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。</p>
Elastic Query Rate (Elastic クエリーレート)	各 Elasticsearch ノードの1秒あたりに Elasticsearch インスタンスに対して実行されたクエリーの合計。
CPU	コンポーネントごとに表示される 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 ノードの 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 バッファの使用法	チャンクに使用されている Fluentd バッファの割合。バッファが一杯になると、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。

メトリクス	説明
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。

7.3.3. 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7.2 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)	<p>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。</p> <ul style="list-style-type: none"> ● 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。

メトリクス	説明
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は、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多数の小さな内部ブロックで構成されます。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は、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7.3 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。

メトリクス	説明
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は、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7.4 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。
CPU usage (CPU の使用率)	ホストコンテナに割り当てられる 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。
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。

メトリクス	説明
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は、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は、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リストを保持する 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 メモリーが十分でない場合に、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7.5 Elasticsearch ノードフィールドデータチャート

メトリクス	説明
-------	----

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7.6 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は、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 インデックスのロットリングに関する以下のチャートが含まれます。

表7.7 インデック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7.8 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。

7.4. KIBANA によるログの可視化

ElasticSearch ログストアを使用している場合は、Kibana コンソールを使用して収集されたログデータを可視化できます。

Kibana を使用すると、データに対して以下を実行できます。

- **Discover** タブを使用して、データを検索および参照します。
- **Visualize** タブを使用して、データをグラフ化およびマッピングします。
- **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は、このドキュメントでは扱いません。インターフェイスの使用に関する詳細は、[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 Dedicated 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。

7.4.1. 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に、これらのインデックスにアクセスできるはずですが、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できます。

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

出力例

```
yes
```



注記

監査ログは、デフォルトでは内部 OpenShift Dedicated 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 Dedicated 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** インデックスの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。

7.4.2. Kibana での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、可視化する方法は、このドキュメントでは扱いません。詳細は、[Kibana ドキュメント](#) を参照してください。

前提条件

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。
- 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に、これらのインデックスにアクセスできるはずですが、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できます。

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

出力例

```
yes
```



注記

監査ログは、デフォルトでは内部 OpenShift Dedicated 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 Dedicated コンソールで、Application Launcher  をクリックし、**Logging** を選択します。
2. OpenShift Dedicated 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つをデプロイメントします。
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7.1 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": "YmJmYTBINDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_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": "YmJmYTBINDktMDMGQtMjE3NmFiOGUyOWM3",
    "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
    ]
  }
}

```

7.4.3. Kibana の設定

Kibana コンソールを使用して、**ClusterLogging** カスタムリソース (CR) を変更することで設定できます。

7.4.3.1. CPU およびメモリー制限の設定

ロギングコンポーネントは、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 の場合は、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。

7.4.3.2. 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。

第8章 OPENSIFT DEDICATED クラスターのサービスログへのアクセス

Red Hat OpenShift Cluster Manager を使用して、OpenShift Dedicated クラスターのサービスログを表示できます。サービスログには、ロードバランサークォータの更新やスケジュールされたメンテナンスアップグレードなどの詳細なクラスターイベントが記録されます。ログには、ユーザー、グループ、および ID プロバイダーの追加または削除などのクラスターリソースの変更も表示されます。

さらに、OpenShift Dedicated クラスターの通知連絡先を追加できます。サブスクライブしたユーザーは、顧客の対応が必要なクラスターイベント、既知のクラスターインシデント、アップグレードのメンテナンス、およびその他のトピックに関するメールを受け取ります。

8.1. OPENSIFT CLUSTER MANAGER を使用したサービスログの表示

Red Hat OpenShift Cluster Manager を使用して、OpenShift Dedicated クラスターのサービスログを表示できます。

前提条件

- OpenShift Dedicated クラスターをインストールしている。

手順

1. [OpenShift Cluster Manager](#) に移動し、クラスターを選択します。
2. クラスターの **Overview** ページで、**Cluster history** セクションのサービスログを表示します。
3. オプション: ドロップダウンメニューから、**Description** または **Severity** でクラスターサービスのログをフィルタリングします。検索バーに特定の項目を入力して、さらにフィルタリングできます。
4. オプション: **Download history** をクリックして、クラスターのサービスログを JSON または CSV 形式でダウンロードします。

8.2. クラスター通知連絡先の追加

OpenShift Dedicated クラスターに関する通知の連絡先を追加できます。クラスター通知メールをトリガーするイベントが発生すると、サブスクライブしているユーザーに通知が送信されます。

手順

1. [OpenShift Cluster Manager](#) に移動し、クラスターを選択します。
2. **Notification contacts** 見出しの **Support** タブで、**Add notification contact** をクリックします。
3. 追加する連絡先の Red Hat ユーザー名またはメールアドレスを入力します。



注記

ユーザー名または電子メールアドレスは、クラスターがデプロイされている Red Hat 組織のユーザーアカウントに関連付けられている必要があります。

4. **Add contact** をクリックします。

検証

- 連絡先が正常に追加されると、確認メッセージが表示されます。ユーザーは、**Support** タブの **Notification contacts** 見出しの下に表示されます。

第9章 ロギングデプロイメントの設定

9.1. ロギングコンポーネントの CPU およびメモリー制限の設定

必要に応じて、それぞれのクラスターロギングコンポーネントの CPU およびメモリー制限の両方を設定できます。

9.1.1. CPU およびメモリー制限の設定

ロギングコンポーネントは、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 の場合は、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。

第10章 ログの収集および転送

10.1. ログの収集と転送

Red Hat OpenShift Logging Operator は、**ClusterLogForwarder** リソース仕様に基づいてコレクターをデプロイします。この Operator では、レガシーの Fluentd コレクターと Vector コレクターの2つのコレクターオプションがサポートされています。



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

10.1.1. ログの収集

ログコレクターは、コンテナとノードのログを収集するために各 OpenShift Dedicated ノードに Pod をデプロイするデーモンセットです。

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。

- 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、コンテナランタイム、および OpenShift Dedicated からの journald ログメッセージによって生成されません。
- すべてのコンテナログ用の `/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。

ログコレクターはこれらのソースからログを収集し、ロギングの設定に応じて内部または外部に転送します。

10.1.1.1. ログコレクターのタイプ

Vector は、ロギングの Fluentd の代替機能として提供されるログコレクターです。

ClusterLogging カスタムリソース(CR) コレクション 仕様を変更して、クラスターが使用するロギングコレクターのタイプを設定できます。

Vector をコレクターとして設定する ClusterLogging CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      type: vector
      vector: {}
# ...
```

10.1.1.2. ログ収集の制限

コンテナランタイムは、プロジェクト、Pod 名、およびコンテナ ID などのログメッセージのソースを特定するための最小限の情報を提供します。この情報だけでは、ログのソースを一意に特定することはできません。ログコレクターがログを処理する前に、指定された名前およびプロジェクトを持つ Pod が削除される場合は、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できない場合があります。この制限により、ログの収集および正規化は **ベストエフォート** ベースであると見なされます。



重要

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

10.1.1.3. タイプ別のログコレクター機能

表10.1 ログソース

機能	Fluentd	Vector
アプリコンテナのログ	✓	✓
アプリ固有のルーティング	✓	✓
namespace 別のアプリ固有のルーティング	✓	✓
インフラコンテナログ	✓	✓
インフラジャーナルログ	✓	✓
Kube API 監査ログ	✓	✓
OpenShift API 監査ログ	✓	✓
Open Virtual Network (OVN) 監査ログ	✓	✓

表10.2 認証および認可

機能	Fluentd	Vector
Elasticsearch 証明書	✓	✓
Elasticsearch ユーザー名/パスワード	✓	✓
Amazon Cloudwatch キー	✓	✓

機能	Fluentd	Vector
Amazon Cloudwatch STS	✓	✓
Kafka 証明書	✓	✓
Kafka のユーザー名/パスワード	✓	✓
Kafka SASL	✓	✓
Loki ベアラートークン	✓	✓

表10.3 正規化と変換

機能	Fluentd	Vector
Viaq データモデル - アプリ	✓	✓
Viaq データモデル - インフラ	✓	✓
Viaq データモデル - インフラ (ジャーナル)	✓	✓
Viaq データモデル - Linux 監査	✓	✓
Viaq データモデル - kube- apiserver 監査	✓	✓
Viaq データモデル - OpenShift API 監査	✓	✓
Viaq データモデル - OVN	✓	✓
ログレベルの正規化	✓	✓
JSON 解析	✓	✓
構造化インデックス	✓	✓
複数行エラー検出	✓	✓
マルチコンテナ/分割インデックス	✓	✓
ラベルのフラット化	✓	✓
CLF 静的ラベル	✓	✓

表10.4 チューニング

機能	Fluentd	Vector
Fluentd readlinelimit	✓	
Fluentd バッファ	✓	
-chunklimitsize	✓	
- totallimitsize	✓	
- overflowaction	✓	
-flushThreadCount	✓	
- flushmode	✓	
- flushinterval	✓	
- retrywait	✓	
- retrytype	✓	
- retrymaxinterval	✓	
- retrytimeout	✓	

表10.5 制約

機能	Fluentd	Vector
メトリクス	✓	✓
ダッシュボード	✓	✓
アラート	✓	✓

表10.6 その他

機能	Fluentd	Vector
グローバルプロキシーサポート	✓	✓
x86 サポート	✓	✓
ARM サポート	✓	✓

機能	Fluentd	Vector
IBM Power® サポート	✓	✓
IBM Z® サポート	✓	✓
IPv6 サポート	✓	✓
ログイベントのバッファリング	✓	
非接続クラスター	✓	✓

10.1.1.4. コレクターの出力

次のコレクターの出力がサポートされています。

表10.7 サポートされている出力

機能	Fluentd	Vector
Elasticsearch v6-v8	✓	✓
Fluent 転送	✓	
Syslog RFC3164	✓	✓ (Logging 5.7+)
Syslog RFC5424	✓	✓ (Logging 5.7+)
Kafka	✓	✓
Amazon Cloudwatch	✓	✓
Amazon Cloudwatch STS	✓	✓
Loki	✓	✓
HTTP	✓	✓ (Logging 5.7+)
Google Cloud Logging	✓	✓
Splunk		✓ (Logging 5.6+)

10.1.2. ログ転送

管理者は、収集するログ、その変換方法と転送先を指定する **ClusterLogForwarder** リソースを作成できます。

ClusterLogForwarder リソースは、コンテナ、インフラストラクチャー、監査ログをクラスター内外の特定のエンドポイントに転送するために使用できます。Transport Layer Security (TLS) がサポートされているため、ログを安全に送信するようにログフォワーダーを設定できます。

管理者は、どのサービスアカウントとユーザーがどの種類のログにアクセスして転送できるかを定義する RBAC アクセス許可を承認することもできます。

10.1.2.1. ログ転送の実装

レガシーの実装とマルチログフォワーダー機能という2つのログ転送実装を使用できます。



重要

マルチログフォワーダー機能と併用するには、Vector コレクターのみがサポートされます。Fluentd コレクターは、レガシーの実装でのみ使用できます。

10.1.2.1.1. レガシー実装

レガシー実装では、クラスターで1つのログフォワーダーのみを使用できます。このモードの **ClusterLogForwarder** リソースは、**instance** という名前を付け、**openshift-logging** namespace に作成する必要があります。**ClusterLogForwarder** リソースは、**openshift-logging** namespace に、**instance** という名前の、対応する **ClusterLogging** リソースも必要です。

10.1.2.1.2. マルチログフォワーダー機能

マルチログフォワーダー機能は、Logging 5.8 以降で利用でき、以下の機能が提供されます。

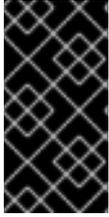
- 管理者は、どのユーザーにログ収集の定義を許可するか、どのユーザーにどのログの収集を許可するかを制御できます。
- 必要な権限が割り当てられたユーザーは、追加のログ収集設定を指定できます。
- 非推奨の Fluentd コレクターから Vector コレクターに移行する管理者は、既存のデプロイメントとは別に新しいログフォワーダーをデプロイできます。既存および新しいログフォワーダーは、ワークロードの移行中に同時に動作します。

マルチログフォワーダーの実装では、**ClusterLogForwarder** リソースに対応する **ClusterLogging** リソースを作成する必要はありません。任意の名前を使用して、任意の namespace で複数の **ClusterLogForwarder** リソースを作成できますが、次の例外があります。

- **instance** という名前の **ClusterLogForwarder** リソースは、Fluentd コレクターを使用するレガシーのワークフローに対応するログフォワーダー向けに予約されているので、**openshift-logging** namespace でこのリソースを作成できません。
- これはコレクター用に予約されているため、**openshift-logging** namespace に **collector** という名前の **ClusterLogForwarder** リソースを作成することはできません。

10.1.2.2. クラスターのマルチログフォワーダー機能の有効化

マルチログフォワーダー機能を使用するには、サービスアカウントとそのサービスアカウントのクラスターロールバインディングを作成する必要があります。その後、**ClusterLogForwarder** リソース内のサービスアカウントを参照して、アクセス許可を制御できます。



重要

openshift-logging namespace 以外の追加の namespace でマルチログ転送をサポートするには、[すべての namespace を監視するように Red Hat OpenShift Logging Operator を更新](#)する必要があります。この機能は、新しい Red Hat OpenShift Logging Operator バージョン 5.8 インストールでデフォルトでサポートされています。

10.1.2.2.1. ログ収集の RBAC 権限の認可

Logging 5.8 以降では、Red Hat OpenShift Operator は **collect-audit-logs**、**collect-application-logs**、および **collect-infrastructural-logs** クラスターロールを提供するため、コレクターは監査ログ、アプリケーションログ、インフラストラクチャーログをそれぞれ収集できます。

必要なクラスターロールをサービスアカウントにバインドすることで、ログコレクションの RBAC パーミッションを承認できます。

前提条件

- Red Hat OpenShift Logging Operator が **openshift-logging** namespace にインストールされている。
- 管理者権限がある。

手順

1. コレクターのサービスアカウントを作成します。認証にトークンを必要とするストレージにログを書き込む場合は、サービスアカウントにトークンを含める必要があります。
2. 適切なクラスターロールをサービスアカウントにバインドします。

バインドコマンドの例

```
$ oc adm policy add-cluster-role-to-user <cluster_role_name> system:serviceaccount:
<namespace_name>:<service_account_name>
```

関連情報

- [Using RBAC Authorization Kubernetes documentation](#)

10.2. ログ出力のタイプ

出力は、ログフォワーダーから送信されるログの宛先を定義します。**ClusterLogForwarder** カスタムリソース (CR) で出力タイプを複数定義し、複数の異なるプロトコルをサポートするサーバーにログを送信できます。

10.2.1. サポート対象のログ転送出力

出力は次のいずれかのタイプになります。

表10.8 サポート対象のログ出力タイプ

出力タイプ	プロトコル	テストで使用	ロギングバージョン	サポート対象のコレクタータイプ
Elasticsearch v6	HTTP 1.1	6.8.1, 6.8.23	5.6+	Fluentd、Vector
Elasticsearch v7	HTTP 1.1	7.12.2, 7.17.7, 7.10.1	5.6+	Fluentd、Vector
Elasticsearch v8	HTTP 1.1	8.4.3, 8.6.1	5.6+	Fluentd ^[1] 、Vector
Fluent Forward	Fluentd forward v1	Fluentd 1.14.6、 Logstash 7.10.1、 Fluentd 1.14.5	5.4+	Fluentd
Google Cloud Logging	REST over HTTPS	最新バージョン	5.7+	Vector
HTTP	HTTP 1.1	Fluentd 1.14.6、 Vector 0.21	5.7+	Fluentd、Vector
Kafka	Kafka 0.11	Kafka 2.4.1、 2.7.0、3.3.1	5.4+	Fluentd、Vector
Loki	REST over HTTP and HTTPS	2.3.0、2.5.0、 2.7、2.2.1	5.4+	Fluentd、Vector
Splunk	HEC	8.2.9, 9.0.0	5.7+	Vector
Syslog	RFC3164、 RFC5424	Rsyslog 8.37.0- 9.e17、rsyslog- 8.39.0	5.4+	Fluentd、Vector [2]
Amazon CloudWatch	REST over HTTPS	最新バージョン	5.4+	Fluentd、Vector

1. Fluentd は、ロギングバージョン 5.6.2 で Elasticsearch 8 をサポートしていません。
2. Vector は、ロギングバージョン 5.7 以降で Syslog をサポートします。

10.2.2. 出力タイプの説明

default

クラスター上の、Red Hat が管理するログストア。デフォルトの出力を設定する必要はありません。



注記

default 出力名は、クラスター上の Red Hat が管理するログストアを参照するために予約されているため、**default** 出力を設定するとエラーメッセージが表示されます。

loki

Loki: 水平方向にスケラブルで可用性の高いマルチテナントログ集計システム。

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。

elasticsearch

外部 Elasticsearch 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。

**重要**

fluentForward 出力は、Fluentd コレクターを使用している場合にのみサポートされます。Vector コレクターを使用している場合はサポートされません。Vector コレクターを使用している場合は、**http** 出力を使用してログを Fluentd に転送できます。

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。

cloudwatch

Amazon Web Services (AWS) がホストするモニタリングおよびログストレージサービスである Amazon CloudWatch。

10.3. JSON ログ転送の有効化

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。

10.3.1. JSON ログの解析

ClusterLogForwarder オブジェクトを使用すると、JSON ログを解析して構造化オブジェクトにし、サポートされている出力に転送できます。

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"}
```

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、CR は以下の例のように構造化された JSON ログエントリを **structured** フィールドにコピーします。

構造化された JSON ログエントリを含む 構造化された 出力例

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



重要

ログエントリに有効な構造化された JSON がない場合、**structured** フィールドは表示されません。

10.3.2. 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の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 で以下の構造化タイプを使用し、Elasticsearch ログストアのインデックス名を作成できます。

- **structuredTypeKey** 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** はコンテナ名を使用してインデックス名を作成します。
- **structuredTypeName: structuredTypeKey** フィールドが設定されていない場合、またはそのキーが存在しない場合、**structuredTypeName** 値が構造化タイプとして使用されます。**structuredTypeKey** フィールドと **structuredTypeName** フィールドの両方を一緒に使用すると、**structuredTypeKey** フィールドのキーが JSON ログデータにない場合に、**structuredTypeName** 値によってフォールバックインデックス名が提供されます。



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リスト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。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
outputDefaults:
  elasticsearch:
    structuredTypeKey: kubernetes.labels.logFormat ❶
    structuredTypeName: nologformat
  pipelines:
  - inputRefs:
    - application
    outputRefs:
    - default
  parse: json ❷
```

❶ Kubernetes **logFormat** ラベルで形成される key-value ペアの値を使用します。

❷ JSON ログの解析を有効にします。

この場合は、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 ❶
    structuredTypeName: nologformat
  pipelines:
    - name: application-logs
      inputRefs:
        - application
        - audit
      outputRefs:
        - elasticsearch-secure
        - default
      parse: json
      labels:
        myLabel: myValue ❷
```

❶ 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。

❷ **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。

この場合は、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

- 構造化レコードの Elasticsearch インデックス は、構造化タイプの前に "app-" を、後ろに "-write" を追加することによって形成されます。
- 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。

10.3.3. JSON ログの Elasticsearch ログストアへの転送

Elasticsearch ログストアの場合は、JSON ログエントリ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の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. **structuredTypeKey** フィールドを使用して、ログレコードフィールドの1つを指定します。
3. **structuredTypeName** フィールドを使用して名前を指定します。



重要

JSON ログを解析するには、**structuredTypeKey** と **structuredTypeName** フィールド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 <filename>.yaml
```

Red Hat OpenShift Logging Operator がコレクター 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、コレクター Pod を削除して強制的に再デプロイします。

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

10.3.4. 同じ Pod 内のコンテナから別のインデックスへの JSON ログの転送

構造化ログを、同じ Pod 内の異なるコンテナから別のインデックスに転送できます。この機能を使用するには、複数コンテナのサポートを使用してパイプラインを設定し、Pod にアノテーションを付ける必要があります。ログは接頭辞が **app-** のインデックスに書き込まれます。これに対応するために、エイリアスを使用して Elasticsearch を設定することを推奨します。



重要

ログの JSON 形式は、アプリケーションによって異なります。作成するインデックスが多すぎるとパフォーマンスに影響するため、この機能の使用は、互換性のない JSON 形式のログのインデックスの作成に限定してください。クエリーを使用して、さまざまな namespace または互換性のある JSON 形式のアプリケーションからログを分離します。

前提条件

- Red Hat OpenShift のロギング: 5.5

手順

1. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputDefaults:
    elasticsearch:
      structuredTypeKey: kubernetes.labels.logFormat 1
      structuredTypeName: nologformat
      enableStructuredContainerLogs: true 2
  pipelines:
  - inputRefs:
    - application
    name: application-logs
    outputRefs:
    - default
    parse: json

```

- 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** マルチコンテナ出力を有効にします。

2. **Pod** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```

apiVersion: v1
kind: Pod
metadata:
  annotations:
    containerType.logging.openshift.io/heavy: heavy 1
    containerType.logging.openshift.io/low: low
spec:
  containers:
  - name: heavy 2
    image: heavyimage
  - name: low
    image: lowimage

```

- 1** 形式: **containerType.logging.openshift.io/<container-name>: <index>**

- 2 アノテーション名はコンテナ名と同じでなければなりません。



警告

この設定により、クラスター上のシャードの数が大幅に増加する可能性があります。

関連情報

- [Kubernetes Annotations](#)

関連情報

- [ログ転送](#)

10.4. ログ転送の設定

ロギングデプロイメントでは、デフォルトでコンテナおよびインフラストラクチャーのログは **ClusterLogging** カスタムリソース (CR) に定義された内部ログストアに転送されます。

セキュアなストレージを提供しないため、監査ログはデフォルトで内部ログストアに転送されません。お客様の責任において、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、適切に保護されていることを確認してください。

このデフォルト設定が要件を満たす場合、**ClusterLogForwarder** CR を設定する必要はありません。**ClusterLogForwarder** CR が存在する場合、**default** 出力を含むパイプラインが定義されている場合を除き、ログは内部ログストアに転送されません。

10.4.1. ログのサードパーティーシステムへの転送

OpenShift Dedicated クラスターの内外の特定のエンドポイントにログを送信するには、**ClusterLogForwarder** カスタムリソース (CR) で **出力** と **パイプライン** の組み合わせを指定します。**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。認証は Kubernetes シークレットオブジェクトによって提供されます。

パイプライン

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アプリケーションを除くユーザーアプリケーションによって生成されるコンテナログ。
- **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** ノード監査システム、 **auditd**、 Kubernetes API サーバー、 OpenShift API サーバー、および OVN ネットワークで生成される監査ログ。

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。

input

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。

Secret

ユーザー認証情報などの機密データを含む **Key:Value** マップ。

以下の点に注意してください。

- 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と、**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。

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: <log_forwarder_name> ①
  namespace: <log_forwarder_namespace> ②
spec:
  serviceAccountName: <service_account_name> ③
  outputs:
  - name: elasticsearch-secure ④
    type: "elasticsearch"
    url: https://elasticsearch.secure.com:9200
    secret:
      name: elasticsearch
  - name: elasticsearch-insecure ⑤
    type: "elasticsearch"
    url: http://elasticsearch.insecure.com:9200
  - name: kafka-app ⑥
    type: "kafka"
    url: tls://kafka.secure.com:9093/app-topic
  inputs: ⑦
  - name: my-app-logs
    application:
      namespaces:
      - my-project
  pipelines:
  - name: audit-logs ⑧
```

```

inputRefs:
  - audit
outputRefs:
  - elasticsearch-secure
  - default
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 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 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に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。
- 5 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。

- 6 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。
- 7 **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。
- 8 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。
 - 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。
- 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。
 - 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、バッファされたログを含む、ログの転送が再開されます。バッファが完全に一杯になると、Fluentd はログの収集を停止します。OpenShift Dedicated はログをローテーションし、それらを削除します。バッファサイズを調整したり、永続ボリューム要求 (PVC) を Fluentd デモンセットまたは Pod に追加したりすることはできません。

サポート対象の認証キー

ここでは、一般的なキータイプを示します。出力タイプは追加の特殊キーをサポートするものもあります。出力固有の設定フィールドにまとめられています。すべての秘密鍵はオプションです。関連するキーを設定して、必要なセキュリティー機能を有効にします。キーやシークレット、サービスアカウント、ポートのオープン、またはグローバルプロキシ設定など、外部の宛先で必要となる可能性のある追加設定を作成し、維持する必要があります。OpenShift Logging は、認証の組み合わせ間の不一致を検証しません。

トランスポートレイヤーセキュリティー (Transport Layer Security, TLS)

シークレットなしで TLSURL (<http://...> または <https://...>) を使用すると、基本的な TLS サーバー側の認証が有効になります。シークレットを含め、次のオプションフィールドを設定すると、追加の TLS 機能が有効になります。

- **passphrase**:(文字列) エンコードされた TLS 秘密鍵をデコードするためのパスフレーズ。 **tls.key** が必要です。
- **ca-bundle.crt**:(文字列) サーバー認証用のカスタマー CA のファイル名。

ユーザー名およびパスワード

- **username**:(文字列) 認証ユーザー名。 **パスワード** が必要です。
- **password**:(文字列) 認証パスワード。 **ユーザー名** が必要です。

Simple Authentication Security Layer (SASL)

- **sasl.enable**(boolean) SASL を明示的に有効または無効にします。ない場合は、SASL は、他の **sasl**. キーが設定されている場合に自動的に有効になります。
- **sasl.mechanisms**:(配列) 許可された SASL メカニズム名のリスト。欠落しているか空の場合は、システムのデフォルトが使用されます。
- **sasl.allow-insecure**:(ブール値) クリアテキストのパスワードを送信するメカニズムを許可します。デフォルトは false です。

10.4.1.1. シークレットの作成

次のコマンドを使用して、証明書とキーファイルを含むディレクトリーにシークレットを作成できます。

```
$ oc create secret generic -n <namespace> <secret_name> \
  --from-file=ca-bundle.crt=<your_bundle_file> \
  --from-literal=username=<your_username> \
  --from-literal=password=<your_password>
```



注記

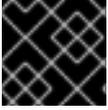
最適な結果を得るには、generic または opaque シークレットを使用することを推奨します。

10.4.2. ログフォワーダーの作成

ログフォワーダーを作成するには、サービスアカウントが収集できるログ入力の種類を指定する **ClusterLogForwarder** CR を作成する必要があります。ログを転送できる出力を指定することもできます。マルチログフォワーダー機能を使用している場合は、**ClusterLogForwarder** CR でサービスアカウ

ントも参照する必要があります。

クラスターでマルチログフォワーダー機能を使用している場合は、任意の名前を使用して、任意の namespace に **ClusterLogForwarder** カスタムリソース (CR) を作成できます。レガシー実装を使用している場合は、**ClusterLogForwarder** CR の名前を **instance** にし、**openshift-logging** namespace に作成する必要があります。



重要

ClusterLogForwarder CR を作成する namespace の管理者権限が必要です。

ClusterLogForwarder リソースの例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ❶
  namespace: <log_forwarder_namespace> ❷
spec:
  serviceAccountName: <service_account_name> ❸
  pipelines:
    - inputRefs:
      - <log_type> ❹
      outputRefs:
      - <output_name> ❺
  outputs:
    - name: <output_name> ❻
      type: <output_type> ❼
      url: <log_output_url> ❽
# ...

```

- ❶ レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- ❷ レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- ❸ サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- ❹ 収集されるログのタイプ。このフィールドの値は、監査ログの場合は **audit**、アプリケーションログの場合は **application**、インフラストラクチャーログの場合は **infrastructure**、またはアプリケーションに定義された指定の入力になります。
- ❺ ❷ ログの転送先の出力のタイプ。このフィールドの値は、**default**、**loki**、**kafka**、**elasticsearch**、**fluentdForward**、**syslog**、または **Cloudwatch** です。



注記

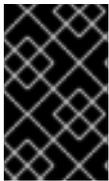
default の出力タイプは、複数のログフォワーダーの実装ではサポートされていません。

- 6 ログの転送先の出力の名前。
- 8 ログの転送先の出力の URL。

10.4.3. ログペイロードと配信の調整

Logging 5.9 以降のバージョンでは、**ClusterLogForwarder** カスタムリソース (CR) の **tuning** 仕様により、ログのスループットまたは耐久性のいずれかを優先するようにデプロイメントを設定する手段が提供されます。

たとえば、コレクターの再起動時にログが失われる可能性を減らす必要がある場合や、規制要件をサポートするために収集されたログメッセージがコレクターの再起動後も保持されるようにする必要があります。ログの耐久性を優先するようにデプロイメントを調整できます。受信できるバッチのサイズに厳しい制限がある出力を使用する場合は、ログスループットを優先するようにデプロイメントを調整することを推奨します。



重要

この機能を使用するには、ロギングのデプロイメントが Vector コレクターを使用するように設定されている必要があります。Fluentd コレクターを使用する場合、**ClusterLogForwarder** CR の **tuning** 仕様はサポートされません。

次の例は、**ClusterLogForwarder** CR オプションで、こちらを変更してログフォワーダーの出力を調整できます。

ClusterLogForwarder CR チューニングオプションの例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  tuning:
    delivery: AtLeastOnce 1
    compression: none 2
    maxWrite: <integer> 3
    minRetryDuration: 1s 4
    maxRetryDuration: 1s 5
# ...
```

- 1 ログ転送の配信モードを指定します。
 - **AtLeastOnce** 配信とは、ログフォワーダーがクラッシュしたり再起動したりした場合に、クラッシュ前に読み取られたが宛先に送信されなかったログが、再送信されることを意味します。クラッシュ後に一部のログが重複している可能性があります。
 - **AtMostOnce** 配信とは、クラッシュ中に失われたログを、ログフォワーダーが復元しようとしてことを意味します。このモードではスループットが向上しますが、ログの損失が大きくなる可能性があります。
- 2 **compression** 設定を指定すると、データはネットワーク経由で送信される前に圧縮されます。すべての出力タイプが圧縮をサポートしているわけではないことに注意してください。指定された圧縮タイプが出力でサポートされていない場合は、エラーが発生します。この設定に使用可能な値

は、**none** (圧縮なし)、**gzip**、**snappy**、**zlib**、または **zstd** です。Kafka の出力を使用している場合は、**lz4** 圧縮も使用できます。詳細は、チューニング出力でサポートされる圧縮タイプの表を参照してください。

- 3 出力への単一の送信操作の最大ペイロードの制限を指定します。
- 4 配信が失敗した後に次に再試行するまでの最小待機時間を指定します。この値は文字列であり、ミリ秒 (**ms**)、秒 (**s**)、または分 (**m**) を指定できます。
- 5 配信が失敗した後に次に再試行するまでの最大待機時間を指定します。この値は文字列であり、ミリ秒 (**ms**)、秒 (**s**)、または分 (**m**) を指定できます。

表10.9 チューニング出力でサポートされる圧縮タイプ

圧縮アルゴリズム	Splunk	Amazon Cloudwatch	Elastic search 8	LokiStack	Apache Kafka	HTTP	Syslog	Google Cloud	Microsoft Azure Monitoring
gzip	X	X	X	X		X			
snappy		X		X	X	X			
zlib		X	X			X			
zstd		X			X	X			
lz4					X				

10.4.4. 複数行の例外検出の有効化

コンテナログの複数行のエラー検出を有効にします。



警告

この機能を有効にすると、パフォーマンスに影響が出る可能性があり、追加のコンピューティングリソースや代替のロギングソリューションが必要になる場合があります。

ログパーサーは頻繁に、同じ例外の個別の行を別々の例外として誤って識別します。その結果、余分なログエントリが発生し、トレースされた情報が不完全または不正確な状態で表示されます。

Java 例外の例

```
java.lang.NullPointerException: Cannot invoke "String.toString()" because "<param1>" is null
```

```
at testjava.Main.handle(Main.java:47)
at testjava.Main.printMe(Main.java:19)
at testjava.Main.main(Main.java:10)
```

- ロギングを有効にして複数行の例外を検出し、それらを1つのログエントリに再アセンブルできるようにする場合は、**ClusterLogForwarder** カスタムリソース (CR) に、値が **true** の **detectMultilineErrors** フィールドが含まれていることを確認します。

ClusterLogForwarder CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines:
    - name: my-app-logs
      inputRefs:
        - application
      outputRefs:
        - default
  detectMultilineErrors: true
```

10.4.4.1. 詳細

ログメッセージが例外スタックトレースを形成する連続したシーケンスとして表示される場合、それらは単一の統合ログレコードに結合されます。最初のログメッセージの内容は、シーケンス内のすべてのメッセージフィールドの連結コンテンツに置き換えられます。

表10.10 各コレクターでサポートされている言語:

Language	Fluentd	Vector
Java	✓	✓
JS	✓	✓
Ruby	✓	✓
Python	✓	✓
golang	✓	✓
PHP	✓	✓
Dart	✓	✓

10.4.4.2. トラブルシューティング

有効にすると、コレクター設定には **detect_exceptions** タイプの新しいセクションが含まれます。

vector 設定セクションの例

```
[transforms.detect_exceptions_app-logs]
  type = "detect_exceptions"
  inputs = ["application"]
  languages = ["All"]
  group_by = ["kubernetes.namespace_name","kubernetes.pod_name","kubernetes.container_name"]
  expire_after_ms = 2000
  multiline_flush_interval_ms = 1000
```

fluentd 設定セクションの例

```
<label @MULTILINE_APP_LOGS>
  <match kubernetes.**>
    @type detect_exceptions
    remove_tag_prefix 'kubernetes'
    message message
    force_line_breaks true
    multiline_flush_interval .2
  </match>
</label>
```

10.4.5. ログの Google Cloud Platform (GCP) への転送

内部のデフォルトの OpenShift Dedicated ログストアに加えて、またはその代わりに、ログを [Google Cloud Logging](#) に転送できます。



注記

この機能を Fluentd で使用することはサポートされていません。

前提条件

- Red Hat OpenShift Logging Operator 5.5.1 以降

手順

1. [Google サービスアカウントキー](#) を使用してシークレットを作成します。

```
$ oc -n openshift-logging create secret generic gcp-secret --from-file google-application-credentials.json=<your_service_account_key_file.json>
```

2. 以下のテンプレートを使用して、**ClusterLogForwarder** カスタムリソース YAML を作成します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
```

```

- name: gcp-1
  type: googleCloudLogging
  secret:
    name: gcp-secret
  googleCloudLogging:
    projectId : "openshift-gce-devel" ④
    logId : "app-gcp" ⑤
  pipelines:
    - name: test-app
      inputRefs: ⑥
      - application
    outputRefs:
      - gcp-1

```

- ① レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- ② レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- ③ サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- ④ ログを保存する [GCP リソース階層](#) の場所に応じて、**projectId**、**folderId**、**organizationId**、または **billingAccountId** フィールドとそれに対応する値を設定します。
- ⑤ **Log Entry** の **logName** フィールドに追加する値を設定します。
- ⑥ パイプラインを使用して転送するログタイプ (**application**、**infrastructure**、または **audit**) を指定します。

関連情報

- [Google Cloud Billing に関するドキュメント](#)
- [Google Cloud Logging クエリー言語のドキュメント](#)

10.4.6. ログの Splunk への転送

内部のデフォルトの OpenShift Dedicated ログストアに加えて、またはその代わりに、[Splunk HTTP Event Collector \(HEC\)](#) にログを転送できます。



注記

この機能を Fluentd で使用することはサポートされていません。

前提条件

- Red Hat OpenShift Logging Operator 5.6 以降
- コレクターとして **vector** が指定された **ClusterLogging** インスタンス

- Base64 でエンコードされた Splunk HEC トークン

手順

1. Base64 でエンコードされた Splunk HEC トークンを使用してシークレットを作成します。

```
$ oc -n openshift-logging create secret generic vector-splunk-secret --from-literal hecToken=
<HEC_Token>
```

2. 以下のテンプレートを使用して、**ClusterLogForwarder** カスタムリソース (CR) を作成または編集します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ①
  namespace: <log_forwarder_namespace> ②
spec:
  serviceAccountName: <service_account_name> ③
  outputs:
    - name: splunk-receiver ④
      secret:
        name: vector-splunk-secret ⑤
        type: splunk ⑥
        url: <http://your.splunk.hec.url:8088> ⑦
  pipelines: ⑧
    - inputRefs:
        - application
        - infrastructure
      name: ⑨
      outputRefs:
        - splunk-receiver ⑩
```

- ① レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- ② レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- ③ サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- ④ 出力の名前を指定します。
- ⑤ HEC トークンが含まれるシークレットの名前を指定します。
- ⑥ 出力タイプを **splunk** として指定します。
- ⑦ Splunk HEC の URL (ポートを含む) を指定します。
- ⑧ パイプラインを使用して転送するログタイプ (**application**、**infrastructure**、または **audit**) を指定します。

- 9 オプション: パイプラインの名前を指定します。
- 10 このパイプラインでログを転送する時に使用する出力の名前を指定します。

10.4.7. HTTP 経由でのログ転送

HTTP 経由でのログ転送は、Fluentd と Vector ログコレクターの両方でサポートされています。有効にするには、**ClusterLogForwarder** カスタムリソース (CR) の出力タイプを **http** に指定します。

手順

- 以下のテンプレートを使用して、**ClusterLogForwarder** CR を作成または編集します。

ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
  - name: httpout-app
    type: http
    url: 4
    http:
      headers: 5
      h1: v1
      h2: v2
      method: POST
    secret:
      name: 6
    tls:
      insecureSkipVerify: 7
  pipelines:
  - name:
    inputRefs:
    - application
    outputRefs:
    - 8

```

- 1 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 ログの宛先アドレス。

- 5 ログレコードと送信する追加のヘッダー。
- 6 宛先認証情報のシークレット名。
- 7 値は **true** または **false** です。
- 8 この値は、出力名と同じである必要があります。

10.4.8. Azure Monitor ログへの転送

Logging 5.9 以降では、デフォルトのログストアに加えて、またはデフォルトのログストアの代わりに、[Azure Monitor Logs](#) にログを転送できます。この機能は、[Vector Azure Monitor Logs sink](#) によって提供されます。

前提条件

- **ClusterLogging** カスタムリソース (CR) インスタンスを管理および作成する方法を熟知している。
- **ClusterLogForwarder** CR インスタンスを管理および作成する方法を熟知している。
- **ClusterLogForwarder** CR 仕様を理解している。
- Azure サービスに関する基本的な知識がある。
- Azure Portal または Azure CLI アクセス用に設定された Azure アカウントがある。
- Azure Monitor Logs のプライマリーセキュリティキーまたはセカンダリーセキュリティキーを取得している。
- 転送するログの種類を決定している。

HTTP データコレクター API 経由で Azure Monitor Logs へのログ転送を有効にするには、以下を実行します。

共有キーを使用してシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
  namespace: openshift-logging
type: Opaque
data:
  shared_key: <your_shared_key> 1
```

- 1 要求を行う [Log Analytics ワークスペース](#) のプライマリーキーまたはセカンダリーキーを含める必要があります。

[共有キー](#) を取得するには、Azure CLI で次のコマンドを使用します。

```
Get-AzOperationalInsightsWorkspaceSharedKey -ResourceGroupName "<resource_name>" -Name
"<workspace_name>"
```

選択したログに一致するテンプレートを使用して、**ClusterLogForwarder** CR を作成または編集します。

すべてのログの転送

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: azure-monitor
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id ①
      logType: my_log_type ②
    secret:
      name: my-secret
  pipelines:
  - name: app-pipeline
    inputRefs:
    - application
    outputRefs:
    - azure-monitor
```

- ① Log Analytics ワークスペースの一意的識別子。必須フィールド。
- ② 送信されるデータの [Azure レコードタイプ](#)。文字、数字、アンダースコア (_) のみを含めることができ、100 文字を超えることはできません。

アプリケーションログおよびインフラストラクチャーログの転送

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: azure-monitor-app
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id
      logType: application_log ①
    secret:
      name: my-secret
  - name: azure-monitor-infra
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id
      logType: infra_log #
    secret:
      name: my-secret
```

```

pipelines:
  - name: app-pipeline
    inputRefs:
      - application
    outputRefs:
      - azure-monitor-app
  - name: infra-pipeline
    inputRefs:
      - infrastructure
    outputRefs:
      - azure-monitor-infra

```

- ① 送信されるデータの **Azure レコードタイプ**。文字、数字、アンダースコア (_) のみを含めることができ、100 文字を超えることはできません。

高度な設定オプション

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
    - name: azure-monitor
      type: azureMonitor
      azureMonitor:
        customerId: my-customer-id
        logType: my_log_type
        azureResourceId: "/subscriptions/111111111" ①
        host: "ods.opinsights.azure.com" ②
      secret:
        name: my-secret
  pipelines:
    - name: app-pipeline
      inputRefs:
        - application
      outputRefs:
        - azure-monitor

```

- ① データの関連付けが必要な Azure リソースのリソース ID。オプションフィールド。
- ② 専用 Azure リージョンの代替ホスト。オプションフィールド。デフォルト値は **ods.opinsights.azure.com** です。Azure Government のデフォルト値は **ods.opinsights.azure.us** です。

10.4.9. 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

内部ログストアの使用に加えて、またはその代わりに、アプリケーションログのコピーを特定のプロジェクトから外部ログアグリゲータに転送できます。また、外部ログアグリゲーターを OpenShift Dedicated 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する必要があります。

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

- ClusterLogForwarder** CR を定義する YAML ファイルを作成または編集します。

ClusterLogForwarder CR の例

```

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 ⑧
  pipelines:
    - name: forward-to-fluentd-insecure ⑨
      inputRefs: ⑩
        - my-app-logs
      outputRefs: ⑪
        - fluentd-server-insecure
      labels:
        project: "my-project" ⑫
    - name: forward-to-fluentd-secure ⑬
      inputRefs:
        - application ⑭
        - 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。
- 4 出力タイプ: **elasticsearch**、**fluentdForward**、**syslog**、または **kafka**。
- 5 有効な絶対 URL としての外部ログアグリゲーターの URL とポート。CIDR アノテーションを使用するクラスター全体のプロキシが有効になっている場合、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は、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 namespace が指定されていない場合、ログはすべての namespace から収集されます。
- 9 パイプライン設定は、名前付き入力から名前付き出力にログを送信します。この例では、**forward-to-fluentd-insecure** という名前のパイプラインは、**my-app-logs** という名前の入力から **fluentd-server-insecure** という名前の出力にログを転送します。
- 10 入力のリスト。
- 11 使用する出力の名前。
- 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ログタイプ (**application**、**infrastructure** または **audit**) を指定します。
 - このパイプラインでログを転送する時に使用する出力の名前を指定します。
 - オプション: デフォルトのログストアにログを転送するための **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。
- 14 この設定を使用すると、すべての namespace からのアプリケーションログが収集されることに注意してください。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

10.4.10. 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は、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** CR オブジェクトを定義する YAML ファイルを作成または編集します。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: <log_forwarder_name> ❶
  namespace: <log_forwarder_namespace> ❷
spec:
  pipelines:
    - inputRefs: [ myAppLogData ] ❸
      outputRefs: [ default ] ❹
  inputs: ❺
    - name: myAppLogData
      application:
        selector:
          matchLabels: ❻
            environment: production
            app: nginx
          namespaces: ❼
            - app1
            - app2
  outputs: ❽
    - <output_name>
    ...

```

- ❶ レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- ❷ レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- ❸ **inputs[].name** から1つ以上のコンマ区切りの値を指定します。
- ❹ **outputs[]** から1つ以上のコンマ区切りの値を指定します。
- ❺ 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。
- ❻ 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。

- 7 オプション: namespace を1つ以上指定します。
 - 8 ログデータを転送する1つ以上の出力を指定します。
2. 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。
 3. 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 ラベルの一意的組み合わせごとに、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 ]
```

4. 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** の詳細は、[セットベースの要件をサポートするリソース](#) を参照してください。

10.4.11. API 監査フィルターの概要

OpenShift API サーバーは、API 呼び出しごとに、リクエスト、レスポンス、リクエスターの ID の詳細を示す監査イベントを生成するため、大量のデータが生成されます。API 監査フィルターはルールを使用して、重要でないイベントを除外してイベントサイズを減少できるようにし、監査証跡をより管理しやすくします。ルールは順番にチェックされ、最初の一致で停止をチェックします。イベントに含まれるデータ量は、**level** フィールドの値によって決定されます。

- **None**: イベントはドロップされます。
- **Metadata**: 監査メタデータが含まれ、リクエストおよびレスポンスの本文は削除されます。
- **Request**: 監査メタデータとリクエスト本文が含まれ、レスポンス本文は削除されます。
- **RequestResponse**: メタデータ、リクエスト本文、レスポンス本文のすべてのデータが含まれます。レスポンス本文が非常に大きくなる可能性があります。たとえば、**oc get pods -A** はクラスター内のすべての Pod の YAML 記述を含むレスポンス本文を生成します。

ロギング 5.8 以降では、**ClusterLogForwarder** カスタムリソース (CR) は標準の [Kubernetes 監査ポリシー](#) と同じ形式を使用しますが、次の追加機能を提供します。

ワイルドカード

ユーザー、グループ、namespace、およびリソースの名前には、先頭または末尾に * アスタリスク文字を付けることができます。たとえば、namespace **openshift-*** は **openshift-apiserver** または **openshift-authentication** と一致します。リソース ***/status** は、**Pod/status** または **Deployment/status** と一致します。

デフォルトのルール

ポリシーのルールに一致しないイベントは、以下のようにフィルターされます。

- **get**、**list**、**watch** などの読み取り専用システムイベントは破棄されます。
- サービスアカウントと同じ namespace 内で発生するサービスアカウント書き込みイベントはドロップされます。
- 他のすべてのイベントは、設定されたレート制限に従って転送されます。

これらのデフォルトを無効にするには、**level** フィールドのみが含まれるルールでルールリストを終了するか、空のルールを追加します。

応答コードが省略される

省略する整数ステータスコードのリスト。**OmitResponseCodes** フィールドを使用して、イベントが作成されない HTTP ステータスコードのリストを使用して、応答の HTTP ステータスコードに基づいてイベントを削除できます。デフォルト値は **[404, 409, 422, 429]** です。値が空のリスト [] の場合、ステータスコードは省略されません。

ClusterLogForwarder CR 監査ポリシーは、OpenShift Dedicated 監査ポリシーに加えて動作します。**ClusterLogForwarder** CR 監査フィルターは、ログコレクターが転送する内容を変更し、動詞、ユーザー、グループ、namespace、またはリソースでフィルタリングする機能を提供します。複数のフィルターを作成して、同じ監査ストリームの異なるサマリーを異なる場所に送信できます。たとえば、詳細なストリームをローカルクラスターログストアに送信し、詳細度の低いストリームをリモートサイトに送信できます。



注記

提供されている例は、監査ポリシーで可能なルールの範囲を示すことを目的としており、推奨される設定ではありません。

監査ポリシーの例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines:
    - name: my-pipeline
      inputRefs: audit 1
      filterRefs: my-policy 2
      outputRefs: default
  filters:
    - name: my-policy
      type: kubeAPIAudit
      kubeAPIAudit:
        # Don't generate audit events for all requests in RequestReceived stage.
        omitStages:
          - "RequestReceived"
  rules:
    # Log pod changes at RequestResponse level
```

```
- level: RequestResponse
resources:
- group: ""
  resources: ["pods"]

# Log "pods/log", "pods/status" at Metadata level
- level: Metadata
resources:
- group: ""
  resources: ["pods/log", "pods/status"]

# Don't log requests to a configmap called "controller-leader"
- level: None
resources:
- group: ""
  resources: ["configmaps"]
  resourceNames: ["controller-leader"]

# Don't log watch requests by the "system:kube-proxy" on endpoints or services
- level: None
users: ["system:kube-proxy"]
verbs: ["watch"]
resources:
- group: "" # core API group
  resources: ["endpoints", "services"]

# Don't log authenticated requests to certain non-resource URL paths.
- level: None
userGroups: ["system:authenticated"]
nonResourceURLs:
- "/api*" # Wildcard matching.
- "/version"

# Log the request body of configmap changes in kube-system.
- level: Request
resources:
- group: "" # core API group
  resources: ["configmaps"]
# This rule only applies to resources in the "kube-system" namespace.
# The empty string "" can be used to select non-namespaced resources.
namespaces: ["kube-system"]

# Log configmap and secret changes in all other namespaces at the Metadata level.
- level: Metadata
resources:
- group: "" # core API group
  resources: ["secrets", "configmaps"]

# Log all other resources in core and extensions at the Request level.
- level: Request
resources:
- group: "" # core API group
- group: "extensions" # Version of group should NOT be included.

# A catch-all rule to log all other requests at the Metadata level.
- level: Metadata
```

- 1 収集されるログのタイプ。このフィールドの値は、監査ログの場合は **audit**、アプリケーションログの場合は **application**、インフラストラクチャーログの場合は **infrastructure**、またはアプリケーションに定義された指定の入力になります。
- 2 監査ポリシーの名前。

関連情報

- [Egress ファイアウォールとネットワークポリシーのロギング](#)

10.4.12. 外部 Loki ロギングシステムへのログ転送

デフォルトのログストアに加えて、またはその代わりに、外部の Loki ロギングシステムにログを転送できます。

Loki へのログ転送を設定するには、Loki の出力と、出力を使用するパイプラインで **ClusterLogForwarder** カスタムリソース (CR) を作成する必要があります。Loki への出力は 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。

前提条件

- CR の **url** フィールドで指定する URL で Loki ロギングシステムが実行されている必要がある。

手順

1. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
  - name: loki-insecure 4
    type: "loki" 5
    url: http://loki.insecure.com:3100 6
    loki:
      tenantKey: kubernetes.namespace_name
      labelKeys:
      - kubernetes.labels.foo
  - name: loki-secure 7
    type: "loki"
    url: https://loki.secure.com:3100
    secret:
      name: loki-secret 8
    loki:
      tenantKey: kubernetes.namespace_name 9
      labelKeys:
      - kubernetes.labels.foo 10
  pipelines:
  - name: application-logs 11

```

inputRefs: **12**

- application
- audit

outputRefs: **13**

- loki-secure

- 1 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 出力の名前を指定します。
- 5 タイプを **loki** として指定します。
- 6 Loki システムの 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。HTTP(S) 通信用の Loki のデフォルトポートは 3100 です。
- 7 セキュアな接続では、**シークレット** を指定して、認証する **https** または **http** URL を指定できます。
- 8 **https** 接頭辞の場合は、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します。シークレットには、それが表す証明書を指す **ca-bundle.crt** 鍵が含まれている必要があります。それ以外の場合、**http** および **https** 接頭辞の場合は、ユーザー名とパスワードを含むシークレットを指定できます。レガシー実装では、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。詳細は、「例: ユーザー名とパスワードを含むシークレットの設定」を参照してください。
- 9 オプション: メタデータキーフィールドを指定して、Loki の **TenantID** フィールドの値を生成します。たとえば、**tenantKey: kubernetes.namespace_name** を設定すると、Kubernetes namespace の名前を Loki のテナント ID の値として使用します。他にどのログレコードフィールドを指定できるかを確認するには、以下の Additional resources セクションの Log Record Fields リンクを参照してください。
- 10 オプション: デフォルトの Loki ラベルを置き換えるメタデータフィールドキーのリストを指定します。loki ラベル名は、正規表現 **[a-zA-Z_][a-zA-Z0-9_]*** と一致する必要があります。ラベル名を形成するため、メタデータキーの無効な文字は **_** に置き換えられます。たとえば、**kubernetes.labels.foo** メタデータキーは、Loki ラベル **kubernetes_labels_foo** になります。**labelKeys** を設定しないと、デフォルト値は **[log_type, kubernetes.namespace_name, kubernetes.pod_name, kubernetes_host]** です。Loki で指定可能なラベルのサイズと数に制限があるため、ラベルのセットを小さくします。[Configuring Loki, limits_config](#) を参照してください。クエリーフィルターを使用して、ログレコードフィールドに基づいてクエリーを実行できます。
- 11 オプション: パイプラインの名前を指定します。
- 12 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。

- 13 このパイプラインでログを転送する時に使用する出力の名前を指定します。



注記

Loki ではログストリームを正しくタイムスタンプで順序付ける必要があるため、**labelKeys** には指定しなくても **kubernetes_host** ラベルセットが常に含まれます。このラベルセットが含まれることで、各ストリームが1つのホストから発信されるので、ホストのクロック間の誤差が原因でタイムスタンプの順番が乱れないようになります。

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

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

関連情報

- [Loki サーバーの設定](#)

10.4.13. 外部 Elasticsearch インスタンスへのログの送信

内部ログストアに加えて、またはその代わりに外部の 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 Dedicated 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) を作成する必要があります。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。



注記

ログを内部 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. **ClusterLogForwarder** CR を定義する YAML ファイルを作成または編集します。

ClusterLogForwarder CR の例

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

```

name: <log_forwarder_name> ❶
namespace: <log_forwarder_namespace> ❷
spec:
  serviceAccountName: <service_account_name> ❸
  outputs:
    - name: elasticsearch-example ❹
      type: elasticsearch ❺
      elasticsearch:
        version: 8 ❻
      url: http://elasticsearch.example.com:9200 ❼
      secret:
        name: es-secret ❽
  pipelines:
    - name: application-logs ❾
      inputRefs: ❿
      - application
      - audit
      outputRefs:
        - elasticsearch-example ⓫
        - default ⓬
      labels:
        myLabel: "myValue" ⓭
# ...

```

- ❶ レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- ❷ レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- ❸ サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- ❹ 出力の名前を指定します。
- ❺ **elasticsearch** タイプを指定します。
- ❻ Elasticsearch バージョンを指定します。これは 6、7、または 8 のいずれかになります。
- ❼ 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。
- ❽ **https** 接頭辞の場合は、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します。シークレットには、それが表す証明書を指す **ca-bundle.crt** 鍵が含まれている必要があります。それ以外の場合、**http** および **https** 接頭辞の場合は、ユーザー名とパスワードを含むシークレットを指定できます。レガシー実装では、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。詳細は、「例: ユーザー名とパスワードを含むシークレットの設定」を参照してください。
- ❾ オプション: パイプラインの名前を指定します。

- 10 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。
- 11 このパイプラインでログを転送する時に使用する出力の名前を指定します。
- 12 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。
- 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。

2. ClusterLogForwarder CR を適用します。

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

例: ユーザー名とパスワードを含むシークレットの設定

ユーザー名とパスワードを含むシークレットを使用して、外部 Elasticsearch インスタンスへのセキュアな接続を認証できます。

たとえば、サードパーティーが Elasticsearch インスタンスを操作するため、相互 TLS (mTLS) キーを使用できない場合に、HTTP または HTTPS を使用してユーザー名とパスワードを含むシークレットを設定できます。

- 以下の例のような **Secret** YAML ファイルを作成します。 **username** および **password** フィールドに base64 でエンコードされた値を使用します。シークレットタイプはデフォルトで `opaque` です。

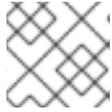
```
apiVersion: v1
kind: Secret
metadata:
  name: openshift-test-secret
data:
  username: <username>
  password: <password>
# ...
```

- シークレットを作成します。

```
$ oc create secret -n openshift-logging openshift-test-secret.yaml
```

- ClusterLogForwarder** CR にシークレットの名前を指定します。

```
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: elasticsearch
    type: "elasticsearch"
    url: https://elasticsearch.secure.com:9200
    secret:
      name: openshift-test-secret
# ...
```



注記

url フィールドの値では、接頭辞は **http** または **https** になります。

4. CR オブジェクトを適用します。

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

10.4.14. 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に、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 Dedicated 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** カスタムリソース (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。

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. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```
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
      labels:
        clusterId: "C1234" ⑪
```

```
- name: forward-to-fluentd-insecure 12
  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** を接頭辞として使用している場合は、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する必要があります。それが表す証明書を指す **ca-bundle.crt** 鍵が含まれている必要があります。
- 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ログタイプ (**application**、**infrastructure** または **audit**) を指定します。
- 9** このパイプラインでログを転送する時に使用する出力の名前を指定します。
- 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** 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** 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。
 - 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
```

10.4.14.1. Logstash が fluentd からデータを取り込むためのナノ秒精度の有効化

Logstash が fluentd からログデータを取り込むには、Logstash 設定ファイルでナノ秒精度を有効にする必要があります。

手順

- Logstash 設定ファイルで、`nanosecond_precision` を `true` に設定します。

Logstash 設定ファイルの例

```
input { tcp { codec => fluent { nanosecond_precision => true } port => 24114 } }
filter { }
output { stdout { codec => rubydebug } }
```

10.4.15. 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に、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 Dedicated 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** カスタムリソース (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。

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. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ①
  namespace: <log_forwarder_namespace> ②
spec:
  serviceAccountName: <service_account_name> ③
  outputs:
  - name: rsyslog-east ④
    type: syslog ⑤
    syslog: ⑥
      facility: local0
      rfc: RFC3164
      payloadKey: message
      severity: informational
    url: 'tls://rsyslogserver.east.example.com:514' ⑦
    secret: ⑧
      name: syslog-secret
  - name: rsyslog-west
    type: syslog
```

```

syslog:
  appName: myapp
  facility: user
  msgID: mymsg
  proclD: myproc
  rfc: RFC5424
  severity: debug
  url: 'tcp://rsyslogserver.west.example.com:514'
pipelines:
  - name: syslog-east 9
    inputRefs: 10
    - audit
    - application
    outputRefs: 11
    - rsyslog-east
    - default 12
    labels:
      secure: "true" 13
      syslog: "east"
  - name: syslog-west 14
    inputRefs:
    - infrastructure
    outputRefs:
    - rsyslog-west
    - default
    labels:
      syslog: "west"

```

- 1 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 出力の名前を指定します。
- 5 **syslog** タイプを指定します。
- 6 オプション: 以下にリスト表示されている syslog パラメーターを指定します。
- 7 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。
- 8 **tls** 接頭辞を使用する場合は、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。シークレットには、それが表す証明書を指す **ca-bundle.crt** 鍵が含まれている必要があります。レガシー実装では、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。
- 9 オプション: パイプラインの名前を指定します。

- 10 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。
- 11 このパイプラインでログを転送する時に使用する出力の名前を指定します。
- 12 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。
- 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。
 - 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 <filename>.yaml
```

10.4.15.1. メッセージ出力へのログソース情報の追加

AddLogSource フィールドを **ClusterLogForwarder** カスタムリソース (CR) に追加することで、**namespace_name**、**pod_name**、および **container_name** 要素をレコードの **メッセージ** フィールドに追加できます。

```
spec:
  outputs:
  - name: syslogout
    syslog:
      addLogSource: true
      facility: user
      payloadKey: message
      rfc: RFC3164
      severity: debug
      tag: mytag
      type: syslog
      url: tls://syslog-receiver.openshift-logging.svc:24224
  pipelines:
  - inputRefs:
    - application
    name: test-app
    outputRefs:
    - syslogout
```



注記

この設定は、RFC3164 と RFC5424 の両方と互換性があります。

AddLogSource を使用しない場合の syslog メッセージ出力の例

```
<15>1 2020-11-15T17:06:14+00:00 fluentd-9hkb4 mytag - - - {"msgcontent"=>"Message Contents",
"timestamp"=>"2020-11-15 17:06:09", "tag_key"=>"rec_tag", "index"=>56}
```

AddLogSource を使用した syslog メッセージ出力の例

```
<15>1 2020-11-16T10:49:37+00:00 crc-j55b9-master-0 mytag - - - namespace_name=clo-test-
6327,pod_name=log-generator-ff9746c49-qxm7l,container_name=log-generator,message=
{"msgcontent":"My life is my message", "timestamp":"2020-11-16 10:49:36", "tag_key":"rec_tag",
"index":76}
```

10.4.15.2. 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](#)。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は、**0** または **kern**
 - ユーザーレベルのメッセージの場合は、**1** または **user**。デフォルトです。
 - メールシステムの場合は、**2** または **mail**
 - システムデーモンの場合は、**3** または **daemon**
 - セキュリティ/認証メッセージの場合は、**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は、**5** または **syslog**
 - ラインプリンターサブシステムの場合は、**6** または **lpr**
 - ネットワーク news サブシステムの場合は、**7** または **news**
 - UUCP サブシステムの場合は、**8** または **uucp**
 - クロックデーモンの場合は、**9** または **cron**
 - セキュリティ認証メッセージの場合は、**10** または **authpriv**
 - FTP デーモンの場合は、**11** または **ftp**
 - NTP サブシステムの場合は、**12** または **ntp**
 - syslog 監査ログの場合は、**13** または **security**
 - syslog アラートログの場合は、**14** または **console**
 - スケジューリングデーモンの場合は、**15** または **solaris-cron**

- ローカルに使用される facility の場合は、**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は、**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は、**1** または **Alert**
 - 重大な状態を示すメッセージの場合は、**2** または **Critical**
 - エラーの状態を示すメッセージの場合は、**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は、**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。

10.4.15.3. 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。

10.4.16. ログの Kafka ブローカーへの転送

デフォルトのログストアに加えて、またはその代わりに、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を含む **ClusterLogForwarder** カスタムリソース (CR) と、その出力を使用するパイプラインを作成する必要があります。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. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ①
  namespace: <log_forwarder_namespace> ②
spec:
  serviceAccountName: <service_account_name> ③
  outputs:
    - name: app-logs ④
      type: kafka ⑤
      url: tls://kafka.example.devlab.com:9093/app-topic ⑥
      secret:
        name: kafka-secret ⑦
    - name: infra-logs
      type: kafka
      url: tcp://kafka.devlab2.example.com:9093/infra-topic ⑧
    - name: audit-logs
      type: kafka
      url: tls://kafka.qelab.example.com:9093/audit-topic
      secret:
        name: kafka-secret-qe
  pipelines:
    - name: app-topic ⑨
      inputRefs: ⑩
      - application
      outputRefs: ⑪
      - app-logs
      labels:
        logType: "application" ⑫
    - name: infra-topic ⑬
      inputRefs:
        - infrastructure
      outputRefs:
        - infra-logs
      labels:
        logType: "infra"
    - name: audit-topic
      inputRefs:
        - audit
      outputRefs:
        - audit-logs
      labels:
        logType: "audit"

```

① レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。

② レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチ

- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ
 - 4 出力の名前を指定します。
 - 5 **kafka** タイプを指定します。
 - 6 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** を接頭辞として使用している場合は、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。シークレットには、それが表す証明書を指す **ca-bundle.crt** 鍵が含まれている必要があります。レガシー実装では、シークレットは **openshift-logging** プロジェクトに存在する必要があります。
 - 8 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。
 - 9 オプション: パイプラインの名前を指定します。
 - 10 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。
 - 11 このパイプラインでログを転送する時に使用する出力の名前を指定します。
 - 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。
 - 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. オプション: 単一の出力を複数の Kafka ブローカーに転送するには、次の例に示すように 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 **トピック** 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. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

10.4.17. ログの Amazon CloudWatch への転送

Amazon Web Services (AWS) がホストするモニタリングおよびログストレージサービスである Amazon CloudWatch にログを転送できます。デフォルトのログストアに加えて、またはログストアの代わりに、CloudWatch にログを転送できます。

CloudWatch へのログ転送を設定するには、CloudWatch 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) を作成する必要があります。

手順

1. **aws_access_key_id** および **aws_secret_access_key** フィールドを使用する **Secret** YAML ファイルを作成し、base64 でエンコードされた AWS 認証情報を指定します。以下に例を示します。

```
apiVersion: v1
kind: Secret
metadata:
  name: cw-secret
  namespace: openshift-logging
data:
  aws_access_key_id: QUtJQUIPU0ZPRE5ON0VYQU1QTEUK
  aws_secret_access_key:
d0phbHJYVXRuRkVNSS9LN01ERU5HL2JQeFJmaUNZRVhBTVBMRUtFWQo=
```

2. シークレットを作成します。以下に例を示します。

```
$ oc apply -f cw-secret.yaml
```

3. **ClusterLogForwarder** CR オブジェクトを定義する YAML ファイルを作成または編集します。このファイルに、シークレットの名前を指定します。以下に例を示します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ①
  namespace: <log_forwarder_namespace> ②
spec:
  serviceAccountName: <service_account_name> ③
```

```

outputs:
- name: cw 4
  type: cloudwatch 5
  cloudwatch:
    groupBy: logType 6
    groupPrefix: <group prefix> 7
    region: us-east-2 8
  secret:
    name: cw-secret 9
pipelines:
- name: infra-logs 10
  inputRefs: 11
  - infrastructure
  - audit
  - application
  outputRefs:
  - cw 12

```

- 1 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **opensehift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 サービスアカウントの名前。サービスアカウントは、ログフォワーダーが **opensehift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 出力の名前を指定します。
- 5 **cloudwatch** タイプを指定します。
- 6 オプション: ログをグループ化する方法を指定します。
 - **logType** は、ログタイプごとにロググループを作成します。
 - **namespaceName** は、アプリケーションの namespace ごとにロググループを作成します。また、インフラストラクチャーおよび監査ログ用の個別のロググループも作成します。
 - **namespaceUUID** は、アプリケーション namespace UUID ごとに新しいロググループを作成します。また、インフラストラクチャーおよび監査ログ用の個別のロググループも作成します。
- 7 オプション: ロググループの名前に含まれるデフォルトの **infrastructureName** 接頭辞を置き換える文字列を指定します。
- 8 AWS リージョンを指定します。
- 9 AWS 認証情報が含まれるシークレットの名前を指定します。
- 10 オプション: パイプラインの名前を指定します。
- 11 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。

12 このパイプラインでログを転送する時に使用する出力の名前を指定します。

4. CR オブジェクトを作成します。

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

例: Amazon CloudWatch での ClusterLogForwarder の使用

ここでは、**ClusterLogForwarder** カスタムリソース (CR) のサンプルと、Amazon CloudWatch に出力するログデータが表示されます。

mycluster という名前の OpenShift Dedicated クラスタを実行しているとします。以下のコマンドは、クラスタの **infrastructureName** を返します。これは、後で **aws** コマンドの作成に使用します。

```
$ oc get Infrastructure/cluster -ojson | jq .status.infrastructureName  
"mycluster-7977k"
```

この例のログデータを生成するには、**app** という名前の namespace で **busybox** pod を実行します。**busybox** pod は、3 秒ごとに stdout にメッセージを書き込みます。

```
$ oc run busybox --image=busybox -- sh -c 'while true; do echo "My life is my message"; sleep 3; done'  
$ oc logs -f busybox  
My life is my message  
My life is my message  
My life is my message  
...
```

busybox pod が実行される **app** namespace の UUID を検索できます。

```
$ oc get ns/app -ojson | jq .metadata.uid  
"794e1e1a-b9f5-4958-a190-e76a9b53d7bf"
```

ClusterLogForwarder カスタムリソース (CR) で、**インフラストラクチャー、監査、およびアプリケーションログ** タイプを **all-logs** パイプラインへの入力として設定します。また、このパイプラインを **cw** 出力に接続し、**us-east-2** リージョンの CloudWatch インスタンスに転送します。

```
apiVersion: "logging.openshift.io/v1"  
kind: ClusterLogForwarder  
metadata:  
  name: instance  
  namespace: openshift-logging  
spec:  
  outputs:  
  - name: cw  
    type: cloudwatch  
    cloudwatch:  
      groupBy: logType  
      region: us-east-2  
    secret:  
      name: cw-secret  
  pipelines:  
  - name: all-logs  
    inputRefs:
```

```
- infrastructure
- audit
- application
outputRefs:
- cw
```

CloudWatch の各リージョンには、3つのレベルのオブジェクトが含まれます。

- ロググループ
 - ログストリーム
 - ログイベント

ClusterLogForwarding CR の **groupBy: logType** の場合に、**inputRefs** にある3つのログタイプで Amazon Cloudwatch に3つのロググループを生成します。

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.application"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

各ロググループにはログストリームが含まれます。

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.application | jq
.logStreams[].logStreamName
"kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log"
```

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.audit | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.k8s-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.linux-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.openshift-audit.log"
...
```

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.infrastructure | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-69f9fd9b58-
zqzw5_openshift-oauth-apiserver_oauth-apiserver-
453c5c4ee026fe20a6139ba6b1cdd1bed25989c905bf5ac5ca211b7cbb5c3d7b.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-
ce51532df7d4e4d5f21c4f4be05f6575b93196336be0027067fd7d93d70f66a4.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-check-endpoints-
82a9096b5931b5c3b1d6dc4b66113252da4a6472c9fff48623baee761911a9ef.log"
...
```

各ログストリームにはログイベントが含まれます。**busybox** Pod からログイベントを表示するには、**application** ロググループからログストリームを指定します。

```
$ aws logs get-log-events --log-group-name mycluster-7977k.application --log-stream-name
kubernetes.var.log.containers.busybox_app_busybox-
```

```

da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log
{
  "events": [
    {
      "timestamp": 1629422704178,
      "message": "{\"docker\":
{\"container_id\":\"da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76\"},\"kub
ernetes\":
{\"container_name\":\"busybox\",\"namespace_name\":\"app\",\"pod_name\":\"busybox\",\"container_ima
ge\":\"docker.io/library/busybox:latest\",\"container_image_id\":\"docker.io/library/busybox@sha256:0f35
4ec1728d9ff32edcd7d1b8bbdfc798277ad36120dc3dc683be44524c8b60\",\"pod_id\":\"870be234-
90a3-4258-b73f-4f4d6e2777c7\",\"host\":\"ip-10-0-216-3.us-east-2.compute.internal\",\"labels\":
{\"run\":\"busybox\"},\"master_url\":\"https://kubernetes.default.svc\",\"namespace_id\":\"794e1e1a-
b9f5-4958-a190-e76a9b53d7bf\",\"namespace_labels\":
{\"kubernetes_io/metadata_name\":\"app\"}},\"message\":\"My life is my
message\",\"level\":\"unknown\",\"hostname\":\"ip-10-0-216-3.us-east-
2.compute.internal\",\"pipeline_metadata\":{\"collector\":
{\"ipaddr4\":\"10.0.216.3\",\"inputname\":\"fluent-plugin-
systemd\",\"name\":\"fluentd\",\"received_at\":\"2021-08-
20T01:25:08.085760+00:00\",\"version\":\"1.7.4 1.6.0\"}},\"@timestamp\":\"2021-08-
20T01:25:04.178986+00:00\",\"viaq_index_name\":\"app-
write\",\"viaq_msg_id\":\"NWRjZmUyMWQzZjgzNC00MjI4LTk3MjMtNTk3NmY3ZjU4NDk1\",\"log_type\":
\"application\",\"time\":\"2021-08-20T01:25:04+00:00\"},
      "ingestionTime": 1629422744016
    },
    ...
  ]
}

```

例: ロググループ名の接頭辞のカスタマイズ

ロググループ名では、デフォルトの **infrastructureName** 接頭辞 **mycluster-7977k** は **demo-group-prefix** のように任意の文字列に置き換えることができます。この変更を加えるには、**ClusterLogForwarding** CR の **groupPrefix** フィールドを更新します。

```

cloudwatch:
  groupBy: logType
  groupPrefix: demo-group-prefix
  region: us-east-2

```

groupPrefix の値は、デフォルトの **infrastructureName** 接頭辞を置き換えます。

```

$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"demo-group-prefix.application"
"demo-group-prefix.audit"
"demo-group-prefix.infrastructure"

```

例: アプリケーションの namespace 名をもとにロググループの命名

クラスター内のアプリケーション namespace ごとに、名前がアプリケーション namespace 名をもとにする CloudWatch にロググループを作成できます。

アプリケーションの namespace オブジェクトを削除して、同じ名前の新しいオブジェクトを作成する場合は、CloudWatch は以前と同じロググループを使用し続けます。

相互に名前が同じアプリケーション namespace オブジェクトを引き継ぐ予定の場合は、この例で説明されている方法を使用します。それ以外で、生成されるログメッセージを相互に区別する必要がある場合は、代わりに Naming log groups for application namespace UUIDs のセクションを参照してください

い。

アプリケーション namespace 名を基にした名前を指定してアプリケーションロググループを作成するには、**ClusterLogForwarder** CR で **groupBy** フィールドの値を **namespaceName** に設定します。

```
cloudwatch:
  groupBy: namespaceName
  region: us-east-2
```

groupBy を **namespaceName** に設定すると、アプリケーションロググループのみが影響を受けます。これは、**audit** および **infrastructure** のロググループには影響しません。

Amazon Cloudwatch では、namespace 名が各ロググループ名の最後に表示されます。アプリケーション namespace (app) が1つであるため、以下の出力は **mycluster-7977k.application** ではなく、新しい **mycluster-7977k.app** ロググループを示しています。

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.app"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

この例のクラスターに複数のアプリケーション namespace が含まれる場合は、出力には namespace ごとに複数のロググループが表示されます。

groupBy フィールドは、アプリケーションロググループだけに影響します。これは、**audit** および **infrastructure** のロググループには影響しません。

例: アプリケーション namespace UUID をもとにロググループの命名

クラスター内のアプリケーション namespace ごとに、名前がアプリケーション namespace の UUID をもとにする CloudWatch にロググループを作成できます。

アプリケーションの namespace オブジェクトを削除して新規のロググループを作成する場合は、CloudWatch で新しいロググループを作成します。

相互に名前が異なるアプリケーション namespace オブジェクトを引き継ぐ予定の場合は、この例で説明されている方法を使用します。それ以外の場合は、前述の例: Naming log groups for application namespace name のセクションを参照してください。

アプリケーション namespace UUID をもとにログエントリーに名前を付けるには、**ClusterLogForwarder** CR で **groupBy** フィールドの値を **namespaceUUID** に設定します。

```
cloudwatch:
  groupBy: namespaceUUID
  region: us-east-2
```

Amazon Cloudwatch では、namespace UUID が各ロググループ名の最後に表示されます。アプリケーション namespace (app) が1つであるため、以下の出力は **mycluster-7977k.application** ではなく、新しい **mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf** ロググループを示しています。

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf" // uid of the "app" namespace
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

groupBy フィールドは、アプリケーションロググループだけに影響します。これは、**audit** および **infrastructure** のロググループには影響しません。

10.4.18. 既存の AWS ロールを使用した AWS CloudWatch のシークレット作成

AWS の既存のロールがある場合は、**oc create secret --from-literal** コマンドを使用して、STS で AWS のシークレットを作成できます。

手順

- CLI で次のように入力して、AWS のシークレットを生成します。

```
$ oc create secret generic cw-sts-secret -n openshift-logging --from-literal=role_arn=arn:aws:iam::123456789012:role/my-role_with-permissions
```

シークレットの例

```
apiVersion: v1
kind: Secret
metadata:
  namespace: openshift-logging
  name: my-secret-name
stringData:
  role_arn: arn:aws:iam::123456789012:role/my-role_with-permissions
```

10.4.19. STS 対応クラスターから Amazon CloudWatch へのログ転送

AWS Security Token Service (STS) が有効になっているクラスターの場合に、AWS サービスアカウントを手動で作成するか、[Cloud Credential Operator \(CCO\)](#) ユーティリティー **ccoctl** を使用してクレデンシャルのリクエストを作成できます。

前提条件

- Red Hat OpenShift のロギング: 5.5 以降

手順

1. 以下のテンプレートを使用して、**CredentialsRequest** カスタムリソース YAML を作成します。

CloudWatch クレデンシャルリクエストのテンプレート

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <your_role_name>-credrequest
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - action:
```

```

- logs:PutLogEvents
- logs:CreateLogGroup
- logs:PutRetentionPolicy
- logs:CreateLogStream
- logs:DescribeLogGroups
- logs:DescribeLogStreams
effect: Allow
resource: arn:aws:logs:*:*:*
secretRef:
  name: <your_role_name>
  namespace: openshift-logging
serviceAccountNames:
  - logcollector

```

2. **ccoctl** コマンドを使用して、**CredentialsRequest** CR を使用して AWS のロールを作成します。**CredentialsRequest** オブジェクトでは、この **ccoctl** コマンドを使用すると、特定の OIDC アイデンティティプロバイダーに紐付けされたトラストポリシーと、CloudWatch リソースでの操作実行パーミッションを付与するパーミッションポリシーを指定して IAM ロールを作成します。このコマンドは、`<path_to_ccoctl_output_dir>/manifests/openshift-logging-<your_role_name>-credentials.yaml` に YAML 設定ファイルも作成します。このシークレットファイルには、AWS IAM ID プロバイダーでの認証中に使用される **role_arn** キー/値が含まれています。

```

$ ccoctl aws create-iam-roles \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \
--identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com ❶

```

- ❶ `<name>` は、クラウドリソースのタグ付けに使用される名前であり、STS クラスターのインストール中に使用される名前と一致する必要があります。

3. 作成したシークレットを適用します。

```

$ oc apply -f output/manifests/openshift-logging-<your_role_name>-credentials.yaml

```

4. **ClusterLogForwarder** カスタムリソースを作成または編集します。

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> ❶
  namespace: <log_forwarder_namespace> ❷
spec:
  serviceAccountName: clf-collector ❸
  outputs:
    - name: cw ❹
      type: cloudwatch ❺
      cloudwatch:
        groupBy: logType ❻
        groupPrefix: <group prefix> ❼

```

```

region: us-east-2 8
secret:
  name: <your_secret_name> 9
pipelines:
  - name: to-cloudwatch 10
    inputRefs: 11
      - infrastructure
      - audit
      - application
    outputRefs:
      - cw 12

```

- 1 レガシー実装では、CR 名は **instance** である必要があります。マルチログフォワーダー実装では、任意の名前を使用できます。
- 2 レガシー実装では、CR namespace は **openshift-logging** である必要があります。マルチログフォワーダー実装では、任意の namespace を使用できます。
- 3 **clf-collector** サービスアカウントを指定します。サービスアカウントは、ログフォワーダーが **openshift-logging** namespace にデプロイされていない場合、マルチログフォワーダーの実装でのみ必要です。
- 4 出力の名前を指定します。
- 5 **cloudwatch** タイプを指定します。
- 6 オプション: ログをグループ化する方法を指定します。
 - **logType** は、ログタイプごとにロググループを作成します。
 - **namespaceName** は、アプリケーションの namespace ごとにロググループを作成します。インフラストラクチャーおよび監査ログは影響を受けず、**logType** によってグループ化されたままになります。
 - **namespaceUUID** は、アプリケーション namespace UUID ごとに新しいロググループを作成します。また、インフラストラクチャーおよび監査ログ用の個別のロググループも作成します。
- 7 オプション: ロググループの名前に含まれるデフォルトの **infrastructureName** 接頭辞を置き換える文字列を指定します。
- 8 AWS リージョンを指定します。
- 9 AWS 認証情報が含まれるシークレットの名前を指定します。
- 10 オプション: パイプラインの名前を指定します。
- 11 パイプラインを使用して転送するログタイプ (**application**、**infrastructure** または **audit**) を指定します。
- 12 このパイプラインでログを転送する時に使用する出力の名前を指定します。

関連情報

- [AWS STS API リファレンス](#)

10.5. ログコレクターの設定

Red Hat OpenShift のロギングは、クラスターからオペレーションとアプリケーションログを収集し、Kubernetes Pod と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** スタンザを使用して実行できます。

10.5.1. ログコレクターの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログコレクターのタイプを設定できます。



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator がインストールされている。
- **ClusterLogging** CR が作成されている。

手順

1. **ClusterLogging** CR の **collection** 仕様を変更します。

ClusterLogging CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  # ...
spec:
  # ...
  collection:
    type: <log_collector_type> ❶
    resources: {}
    tolerations: {}
  # ...
```

- ❶ ロギングに使用するログコレクターのタイプ。これは、**vector** または **fluentd** にすることができます。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

10.5.2. LogFileMetricExporter リソースの作成

ロギングバージョン 5.8 以降のバージョンでは、LogFileMetricExporter はデフォルトでコレクターを使用してデプロイされなくなりました。実行中のコンテナによって生成されたログからメトリクスを生成するには、**LogFileMetricExporter** カスタムリソース (CR) を手動で作成する必要があります。

LogFileMetricExporter CR を作成しない場合、OpenShift Dedicated Web コンソールのダッシュボードで **Produced Logs** のメッセージ **No datapoints found** が表示される場合があります。

前提条件

- 管理者権限がある。
- Red Hat OpenShift Logging Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **LogFileMetricExporter** CR を YAML ファイルとして作成します。

LogFileMetricExporter CR の例

```
apiVersion: logging.openshift.io/v1alpha1
kind: LogFileMetricExporter
metadata:
  name: instance
  namespace: openshift-logging
spec:
  nodeSelector: {} ❶
  resources: ❷
    limits:
      cpu: 500m
      memory: 256Mi
    requests:
      cpu: 200m
      memory: 128Mi
  tolerations: [] ❸
# ...
```

- ❶ オプション: **nodeSelector** スタンザは、Pod がスケジュールされるノードを定義します。
- ❷ **resources** スタンザは、**LogFileMetricExporter** CR のリソース要件を定義します。
- ❸ オプション: **tolerations** スタンザは、Pod が受け入れる許容範囲を定義します。

2. 次のコマンドを実行して、**LogFileMetricExporter** CR を適用します。

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

検証

logfilesmetricexporter Pod は、各ノードで **collector** Pod と同時に実行されます。

- 次のコマンドを実行して出力を確認し、**LogFilesmetricExporter** CR を作成した namespace で **logfilesmetricexporter** Pod が実行されていることを確認します。

```
$ oc get pods -l app.kubernetes.io/component=logfilesmetricexporter -n openshift-logging
```

出力例

```
NAME                                READY STATUS RESTARTS AGE
logfilesmetricexporter-9qbjj 1/1   Running 0      2m46s
logfilesmetricexporter-cbc4v 1/1   Running 0      2m46s
```

10.5.3. 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

- **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:
    type: 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。

10.5.4. 入力レシーバーの設定

Red Hat OpenShift Logging Operator は、クライアントがコレクターに書き込めるように、設定された各入力レシーバー用のサービスをデプロイします。このサービスは、入力レシーバーに指定されたポートを公開します。サービス名は、以下に基づいて生成されます。

- マルチログフォワーダー **ClusterLogForwarder** CR のデプロイメントの場合、サービス名は **<ClusterLogForwarder_CR_name>-<input_name>** という形式になります。たとえば、**example-http-receiver** などです。

- 従来の **ClusterLogForwarder** CR のデプロイメント (**instance** という名前が付けられ、**openshift-logging** namespace に配置されているデプロイメント) の場合、サービス名は **collector-<input_name>** という形式になります。たとえば、**collector-http-receiver** です。

10.5.4.1. 監査ログを HTTP サーバーとして受信するようにコレクターを設定する

ClusterLogForwarder カスタムリソース (CR) で **http** をレシーバー入力として指定すると、HTTP 接続をリッスンして監査ログを HTTP サーバーとして受信するようにログコレクターを設定できます。これにより、OpenShift Dedicated クラスターの内部と外部の両方から収集された監査ログに共通のログストアを使用できるようになります。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator がインストールされている。
- **ClusterLogForwarder** CR が作成されている。

手順

1. **ClusterLogForwarder** CR を変更して、**http** レシーバー入力の設定を追加します。

マルチログフォワーダーデプロイメントを使用している場合の ClusterLogForwarder CR の例

```
apiVersion: logging.openshift.io/v1beta1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccountName: <service_account_name>
  inputs:
    - name: http-receiver ❶
      receiver:
        type: http ❷
        http:
          format: kubeAPIAudit ❸
          port: 8443 ❹
  pipelines: ❺
    - name: http-pipeline
      inputRefs:
        - http-receiver
# ...
```

- ❶ 入力レシーバーの名前を指定します。
- ❷ 入力レシーバー型を **http** に指定します。
- ❸ 現在、**HTTP** 入力レシーバーでは **kube-apiserver** Webhook 形式のみがサポートされています。

- 4 オプション: 入力レシーバーがリッスンするポートを指定します。これは、**1024** から **65535** までの値とします。指定されていない場合、デフォルト値は **8443** です。
- 5 入力レシーバーのパイプラインを設定します。

従来のデプロイメントを使用している場合の ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  inputs:
    - name: http-receiver 1
      receiver:
        type: http 2
        http:
          format: kubeAPIAudit 3
          port: 8443 4
  pipelines: 5
    - inputRefs:
      - http-receiver
      name: http-pipeline
# ...

```

- 1 入力レシーバーの名前を指定します。
- 2 入力レシーバー型を **http** に指定します。
- 3 現在、**HTTP** 入力レシーバーでは **kube-apiserver** Webhook 形式のみがサポートされています。
- 4 オプション: 入力レシーバーがリッスンするポートを指定します。これは、**1024** から **65535** までの値とします。指定されていない場合、デフォルト値は **8443** です。
- 5 入力レシーバーのパイプラインを設定します。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR に変更を適用します。

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

関連情報

- [API 監査フィルターの概要](#)

10.5.5. Fluentd ログフォワーダーの高度な設定



注記

Fluentd は非推奨となっており、今後のリリースで削除される予定です。Red Hat は、現在のリリースのライフサイクル中にこの機能のバグ修正とサポートを提供しますが、この機能は拡張されなくなりました。Fluentd の代わりに、Vector を使用できます。

ロギングには、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。

- チャンクおよびチャンクのバッファサイズ
- チャンクのフラッシュ動作
- チャンク転送の再試行動作

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、送信先へ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 Dedicated のデフォルトで、Fluentd は **指数バックオフ** 方式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。

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10.11 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の最大サイズ。バッファ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。	ノードディスクの約 15% がすべての出力に分散されます。
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: <ul style="list-style-type: none"> ● 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
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

パラメーター	説明	デフォルト
overflowAction	<p>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。</p> <ul style="list-style-type: none"> ● 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	<p>フラッシュに失敗する場合の再試行方法:</p> <ul style="list-style-type: none"> ● 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
retryTimeOut	レコードが破棄される前に再試行を試みる最大時間間隔。	60m
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:
  collection:
    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 チャンクバッファの最大サイズを指定します。

3. Fluentd Pod が再デプロイされていることを確認します。

```
$ oc get pods -l component=collector -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/collector-config --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 "#{ENV['TOTAL_LIMIT_SIZE_PER_BUFFER'] || '8589934592'}"
  chunk_limit_size 8m
  overflow_action throw_exception
  disable_chunk_backup true
</buffer>
```

10.6. KUBERNETES イベントの収集および保存

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

イベントルーターはすべてのプロジェクトからイベントを収集し、それらを **STDOUT** に書き込みます。次に、コレクターはそれらのイベントを **ClusterLogForwarder** カスタムリソース (CR) で定義されたストアに転送します。



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。

10.6.1. 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。



注記

Event Router イメージは Red Hat OpenShift Logging Operator の一部ではないため、個別にダウンロードする必要があります。

次の **Template** 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。このテンプレートを変更せずに使用することも、テンプレートを編集して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もできます。

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。
- Red Hat OpenShift Logging Operator 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。

```

apiVersion: template.openshift.io/v1
kind: Template
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to OpenShift Logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
- kind: ServiceAccount ❶
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
- kind: ClusterRole ❷
  apiVersion: rbac.authorization.k8s.io/v1
  metadata:
    name: event-reader
  rules:
- apiGroups: [""]
  resources: ["events"]
  verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding ❸
  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 ❹
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment ❺
  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
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop: ["ALL"]
      securityContext:
        runAsNonRoot: true
        seccompProfile:
          type: RuntimeDefault
      volumes:
        - name: config-volume
          configMap:
            name: eventrouter
  parameters:
    - name: IMAGE 6
      displayName: Image
      value: "registry.redhat.io/openshift-logging/eventrouter-rhel9:v0.4"
    - 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 **v0.4** 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 CPU の最小量を指定します。デフォルトは **100m** に設定されます。
- 8 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。
- 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.rbac.authorization.k8s.io/event-reader created
clusterrolebinding.rbac.authorization.k8s.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11章 ログストレージ

11.1. ログストレージについて

クラスター上の内部 Loki または Elasticsearch ログストアを使用してログを保存したり、**ClusterLogForwarder** カスタムリソース (CR) を使用してログを外部ストアに転送したりできます。

11.1.1. ログストレージの種類

Loki は、水平方向にスケラブルで可用性の高いマルチテナントログ集約システムであり、ロギングのログストアとして Elasticsearch の代替として提供されています。

Elasticsearch は、取り込み中に受信ログレコードを完全にインデックス化します。Loki は、取り込み中にいくつかの固定ラベルのみをインデックスに登録し、ログが保存されるまで、より複雑な解析が延期されるので Loki がより迅速にログを収集できるようになります。

11.1.1.1. Elasticsearch ログストアについて

ロギングの Elasticsearch インスタンスは、約 7 日間の短期間の保存用に最適化され、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のみアクセスできます。

11.1.2. ログストアのクエリー

LogQL ログクエリ言語を使用して Loki にクエリー を実行できます。

11.1.3. 関連情報

- [Loki コンポーネントのドキュメント](#)
- [Loki オブジェクトストレージのドキュメント](#)

11.2. ログストレージのインストール

OpenShift CLI (**oc**) または OpenShift Dedicated Web コンソールを使用して、OpenShift Dedicated クラスタにログストアをデプロイできます。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

11.2.1. Loki ログストアのデプロイ

Loki Operator を使用して、OpenShift Dedicated クラスタに内部 Loki ログストアをデプロイできます。Loki Operator をインストールした後、シークレットを作成することで Loki オブジェクトストレージを設定し、**LokiStack** カスタムリソース (CR) を作成する必要があります。

11.2.1.1. Loki デプロイメントのサイズ

Loki のサイズは **1x.<size>** の形式に従います。この場合の **1x** はインスタンスの数を、**<size>** は性能を指定します。



重要

デプロイメントサイズの **1x** の数は変更できません。

表11.1 Loki のサイズ

	1x.demo	1x.extra-small	1x.small	1x.medium
データ転送	デモ使用のみ	100 GB/日	500 GB/日	2 TB/日
1秒あたりのクエリー数 (QPS)	デモ使用のみ	200 ミリ秒で 1-25 QPS	200 ミリ秒で 25 - 50 QPS	200 ミリ秒で 25 - 75 QPS
レプリケーション係数	なし	2	2	2
合計 CPU 要求	なし	仮想 CPU 14 個	仮想 CPU 34 個	仮想 CPU 54 個

	1x.demo	1x.extra-small	1x.small	1x.medium
ルーラーを使用する場合の合計 CPU リクエスト	なし	仮想 CPU 16 個	仮想 CPU 42 個	仮想 CPU 70 個
合計メモリー要求	なし	31 Gi	67 Gi	139 Gi
ルーラーを使用する場合の合計メモリーリクエスト	なし	35Gi	83Gi	171Gi
合計ディスク要求	40Gi	430 Gi	430 Gi	590Gi
ルーラーを使用する場合の合計ディスクリクエスト	80Gi	750Gi	750Gi	910Gi

11.2.1.2. OpenShift Dedicated Web コンソールを使用した Loki Operator のインストール

OpenShift Dedicated クラスタにロギングをインストールして設定するには、追加の Operator をインストールする必要があります。これは、Web コンソールの Operator Hub から実行できます。

OpenShift Dedicated Operator は、カスタムリソース (CR) を使用してアプリケーションとそのコンポーネントを管理します。高レベルの構成と設定は、CR 内でユーザーが指定します。Operator は、Operator のロジック内に組み込まれたベストプラクティスに基づいて、高レベルのディレクティブを低レベルのアクションに変換します。カスタムリソース定義 (CRD) は CR を定義し、Operator のユーザーが使用できるすべての設定をリストします。Operator をインストールすると CRD が作成され、CR の生成に使用されます。

前提条件

- サポートされているオブジェクトストア (AWS S3、Google Cloud Storage、Azure、Swift、Minio、OpenShift Data Foundation) にアクセスできる。
- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスできる。

手順

1. OpenShift Dedicated Web コンソールの **Administrator** パースペクティブで、**Operators** → **OperatorHub** に移動します。
2. **Filter by keyword** フィールドに Loki Operator と入力します。使用可能な Operator のリストで **Loki Operator** をクリックし、**Install** をクリックします。



重要

Community Loki Operator は Red Hat ではサポートされていません。

3. **Update channel** として **stable** または **stable-x.y** を選択します。



注記

stable チャンネルは、Logging の最新リリースを対象とする更新のみを提供します。以前のリリースの更新を引き続き受信するには、サブスクリプションチャンネルを **stable-x.y** に変更する必要があります。**xy** は、インストールしたログのメジャーバージョンとマイナーバージョンを表します。たとえば、**stable-5.7** です。

Loki Operator はグローバルオペレーターグループ namespace である **openshift-operators-redhat** にデプロイする必要があるため、**Installation mode** と **Installed Namespace** がすでに選択されています。この namespace がいない場合は、自動的に作成されます。

4. **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。
5. **Update approva** で **Automatic** を選択し、**Install** をクリックします。
サブスクリプションの承認ストラテジーが **Automatic** に設定されている場合、アップグレードプロセスは、選択したチャンネルで新規 Operator バージョンが利用可能になるとすぐに開始します。承認ストラテジーが **Manual** に設定されている場合は、保留中のアップグレードを手動で承認する必要があります。

検証

1. **Operators** → **Installed Operators** に移動します。
2. **openshift-logging** プロジェクトが選択されていることを確認します。
3. **Status** 列に、緑色のチェックマークおよび **InstallSucceeded** と、**Up to date** というテキストが表示されていることを確認します。



注記

インストールが完了する前に、Operator に **Failed** ステータスが表示される場合があります。**InstallSucceeded** メッセージが表示されて Operator のインストールが完了した場合は、ページを更新します。

11.2.1.3. Web コンソールを使用して Loki オブジェクトストレージのシークレットを作成する

Loki オブジェクトストレージを設定するには、シークレットを作成する必要があります。OpenShift Dedicated Web コンソールを使用してシークレットを作成できます。

前提条件

- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスできる。
- Loki Operator がインストールされている。

手順

1. OpenShift Dedicated Web コンソールの **Administrator** パースペクティブで、**Workloads** → **Secrets** に移動します。

2. **Create** ドロップダウンリストから、**From YAML** を選択します。
3. **access_key_id** フィールドと **access_key_secret** フィールドを使用して認証情報を指定し、**bucketnames**、**endpoint**、および **region** フィールドを使用してオブジェクトの保存場所を定義するシークレットを作成します。次の例では、AWS が使用されています。

Secret オブジェクトの例

```

apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1

```

関連情報

- [Loki オブジェクトストレージ](#)

11.2.1.4. ワークロードアイデンティティフェデレーション

ワークロードアイデンティティフェデレーションを使用すると、有効期間の短いトークンを使用してクラウドベースのログストアに対して認証できます。

前提条件

- OpenShift Dedicated 4.14 以降
- Logging 5.9 以降

手順

- OpenShift Dedicated Web コンソールを使用して Loki Operator をインストールすると、STS クラスターが自動的に検出されます。プロンプトが表示され、ルールを作成するように求められます。また、Loki Operator が **CredentialsRequest** オブジェクトを作成するのに必要なデータを提供するように求められます。このオブジェクトにより、シークレットが設定されます。
- OpenShift CLI (**oc**) を使用して Loki Operator をインストールする場合は、次の例に示すように、ストレージプロバイダーに適したテンプレートを使用してサブスクリプションオブジェクトを手動で作成する必要があります。この認証ストラテジーは、指定のストレージプロバイダーでのみサポートされます。

Azure のサンプルサブスクリプション

```

apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat

```

```
spec:
  channel: "stable-5.9"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: CLIENTID
        value: <your_client_id>
      - name: TENANTID
        value: <your_tenant_id>
      - name: SUBSCRIPTIONID
        value: <your_subscription_id>
      - name: REGION
        value: <your_region>
```

AWS のサンプルサブスクリプション

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-5.9"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: ROLEARN
        value: <role_ARN>
```

11.2.1.5. Web コンソールを使用して LokiStack カスタムリソースを作成する

OpenShift Dedicated Web コンソールを使用して、**LokiStack** カスタムリソース (CR) を作成できます。

前提条件

- 管理者権限がある。
- OpenShift Dedicated Web コンソールにアクセスできる。
- Loki Operator がインストールされている。

手順

1. **Operators** → **Installed Operators** ページに移動します。 **All Instances** タブをクリックします。
2. **Create new** ドロップダウンリストから、**LokiStack** を選択します。
3. **YAML view** を選択し、次のテンプレートを使用して **LokiStack** CR を作成します。

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki ❶
  namespace: openshift-logging
spec:
  size: 1x.small ❷
  storage:
    schemas:
      - effectiveDate: '2023-10-15'
        version: v13
    secret:
      name: logging-loki-s3 ❸
      type: s3 ❹
      credentialMode: ❺
  storageClassName: <storage_class_name> ❻
  tenants:
    mode: openshift-logging

```

- ❶ **logging-loki** という名前を使用します。
- ❷ デプロイメントサイズを指定します。ロギング 5.8 以降のバージョンでは、Loki の実稼働インスタンスでサポートされているサイズオプションは **1x.extra-small**、**1x.small**、または **1x.medium** です。
- ❸ ログストレージに使用するシークレットを指定します。
- ❹ 対応するストレージタイプを指定します。
- ❺ 任意のフィールド、Logging 5.9 以降。サポートされているユーザー設定値は、次のとおりです。**static** は、シークレットに保存された認証情報を使用する、サポートされているすべてのオブジェクトストレージタイプで使用できるデフォルトの認証モードです。**token** は、認証情報ソースから取得される有効期間が短いトークンです。このモードでは、オブジェクトストレージに必要な認証情報が静的設定に格納されません。代わりに、実行時にサービスを使用して認証情報が生成されるため、有効期間が短い認証情報の使用と、よりきめ細かい制御が可能になります。この認証モードは、すべてのオブジェクトストレージタイプでサポートされているわけではありません。Loki がマネージド STS モードで実行されていて、STS/WIF クラスターで CCO を使用している場合、**token-cco** がデフォルト値です。
- ❻ 一時ストレージ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。クラスターで使用可能なストレージクラスは、**oc get storageclasses** コマンドを使用してリスト表示できます。

11.2.1.6. CLI を使用して Loki Operator をインストールする

OpenShift Dedicated クラスターにロギングをインストールして設定するには、追加の Operator をインストールする必要があります。これは、OpenShift Dedicated CLI から実行できます。

OpenShift Dedicated Operator は、カスタムリソース (CR) を使用してアプリケーションとそのコンポーネントを管理します。高レベルの構成と設定は、CR 内でユーザーが指定します。Operator は、Operator のロジック内に組み込まれたベストプラクティスに基づいて、高レベルのディレクティブを

低レベルのアクションに変換します。カスタムリソース定義 (CRD) は CR を定義し、Operator のユーザーが使用できるすべての設定をリストします。Operator をインストールすると CRD が作成され、CR の生成に使用されます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- サポートされているオブジェクトストアにアクセスできる。例: AWS S3、Google Cloud Storage、Azure、Swift、Minio、OpenShift Data Foundation。

手順

1. **Subscription** オブジェクトを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat ❶
spec:
  channel: stable ❷
  name: loki-operator
  source: redhat-operators ❸
  sourceNamespace: openshift-marketplace
```

- ❶ **openshift-operators-redhat** namespace を指定する必要があります。
- ❷ チャンネルとして **stable** または **stable-5.<x>** を指定します。
- ❸ **redhat-operators** を指定します。OpenShift Dedicated 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) の設定時に作成した **CatalogSource** オブジェクトの名前を指定します。

2. **Subscription** オブジェクトを適用します。

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

11.2.1.7. CLI を使用して Loki オブジェクトストレージのシークレットを作成する

Loki オブジェクトストレージを設定するには、シークレットを作成する必要があります。これは、OpenShift CLI (**oc**) を使用して実行できます。

前提条件

- 管理者権限がある。
- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

- 次のコマンドを使用して、証明書とキーファイルが含まれるディレクトリーにシークレットを作成できます。

```
$ oc create secret generic -n openshift-logging <your_secret_name> \
--from-file=tls.key=<your_key_file>
--from-file=tls.crt=<your_cert_file>
--from-file=ca-bundle.crt=<your_bundle_file>
--from-literal=username=<your_username>
--from-literal=password=<your_password>
```



注記

最良の結果を得るには、generic または opaque シークレットを使用してください。

検証

- 次のコマンドを実行して、シークレットが作成されたことを確認します。

```
$ oc get secrets
```

関連情報

- [Loki オブジェクトストレージ](#)

11.2.1.8. CLI を使用して LokiStack カスタムリソースを作成する

OpenShift CLI (**oc**) を使用して、**LokiStack** カスタムリソース (CR) を作成できます。

前提条件

- 管理者権限がある。
- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **LokiStack** CR を作成します。

LokiStack CR の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki 1
  namespace: openshift-logging
spec:
  size: 1x.small 2
  storage:
    schemas:
```

```

- effectiveDate: '2023-10-15'
  version: v13
secret:
  name: logging-loki-s3 ③
  type: s3 ④
  credentialMode: ⑤
storageClassName: <storage_class_name> ⑥
tenants:
  mode: openshift-logging

```

- ① **logging-loki** という名前を使用します。
- ② デプロイメントサイズを指定します。ロギング 5.8 以降のバージョンでは、Loki の実稼働インスタンスでサポートされているサイズオプションは **1x.extra-small**、**1x.small**、または **1x.medium** です。
- ③ ログストレージに使用するシークレットを指定します。
- ④ 対応するストレージタイプを指定します。
- ⑤ 任意のフィールド、Logging 5.9 以降。サポートされているユーザー設定値は、次のとおりです。**static** は、シークレットに保存された認証情報を使用する、サポートされているすべてのオブジェクトストレージタイプで使用できるデフォルトの認証モードです。**token** は、認証情報ソースから取得される有効期間が短いトークンです。このモードでは、オブジェクトストレージに必要な認証情報が静的設定に格納されません。代わりに、実行時にサービスを使用して認証情報が生成されるため、有効期間が短い認証情報の使用と、よりきめ細かい制御が可能になります。この認証モードは、すべてのオブジェクトストレージタイプでサポートされているわけではありません。Loki がマネージド STS モードで実行されていて、STS/WIF クラスタで CCO を使用している場合、**token-cco** がデフォルト値です。
- ⑥ 一時ストレージ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。クラスタで使用可能なストレージクラスは、**oc get storageclasses** コマンドを使用してリスト表示できます。

2. 次のコマンドを実行して、**LokiStack** CR を適用します。

検証

- 次のコマンドを実行して出力を観察し、**openshift-logging** プロジェクト内の Pod をリスト表示してインストールを確認します。

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

次のリストのように、ロギングコンポーネント用の Pod が複数表示されていることを確認します。

出力例

```

NAME                                READY STATUS RESTARTS AGE
cluster-logging-operator-78fddc697-mnl82  1/1   Running 0    14m
collector-6cglq                          2/2   Running 0    45s
collector-8r664                          2/2   Running 0    45s

```

```

collector-8z7px          2/2  Running 0    45s
collector-pdxl9         2/2  Running 0    45s
collector-tc9dx         2/2  Running 0    45s
collector-xkd76         2/2  Running 0    45s
logging-loki-compactor-0      1/1  Running 0    8m2s
logging-loki-distributor-b85b7d9fd-25j9g  1/1  Running 0    8m2s
logging-loki-distributor-b85b7d9fd-xwjs6  1/1  Running 0    8m2s
logging-loki-gateway-7bb86fd855-hjhl4    2/2  Running 0    8m2s
logging-loki-gateway-7bb86fd855-qjtlb    2/2  Running 0    8m2s
logging-loki-index-gateway-0      1/1  Running 0    8m2s
logging-loki-index-gateway-1      1/1  Running 0    7m29s
logging-loki-ingester-0          1/1  Running 0    8m2s
logging-loki-ingester-1          1/1  Running 0    6m46s
logging-loki-querier-f5cf9cb87-9fdjd    1/1  Running 0    8m2s
logging-loki-querier-f5cf9cb87-fp9v5    1/1  Running 0    8m2s
logging-loki-query-frontend-58c579fcb7-lfvbc  1/1  Running 0    8m2s
logging-loki-query-frontend-58c579fcb7-tjf9k  1/1  Running 0    8m2s
logging-view-plugin-79448d8df6-ckgmx    1/1  Running 0    46s

```

11.2.2. Loki オブジェクトストレージ

Loki Operator は、[AWS S3](#) だけでなく、[Minio](#) や [OpenShift Data Foundation](#) などの他の S3 互換オブジェクトストアもサポートしています。[Azure](#)、[GCS](#)、および [Swift](#) もサポートされています。

Loki ストレージの推奨命名法は、**logging-loki-<your_storage_provider>** です。

次の表は、各ストレージプロバイダーの **LokiStack** カスタムリソース (CR) 内の **type** 値を示しています。詳細は、ストレージプロバイダーに関するセクションを参照してください。

表11.2 シークレットタイプのクイックリファレンス

ストレージプロバイダー	シークレットの type 値
AWS	s3
Azure	azure
Google Cloud	gcs
Minio	s3
OpenShift Data Foundation	s3
Swift	swift

11.2.2.1. AWS ストレージ

前提条件

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

- AWS 上に [バケット](#) を作成している。
- [AWS IAM ポリシーと IAM ユーザー](#) を作成している。

手順

- 次のコマンドを実行して、**logging-loki-aws** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create secret generic logging-loki-aws \
  --from-literal=bucketnames="<bucket\_name>" \
  --from-literal=endpoint="<aws\_bucket\_endpoint>" \
  --from-literal=access_key_id="<aws\_access\_key\_id>" \
  --from-literal=access_key_secret="<aws\_access\_key\_secret>" \
  --from-literal=region="<aws\_region\_of\_your\_bucket>"
```

11.2.2.1.1. STS 対応クラスターの AWS ストレージ

クラスターで STS が有効になっている場合、Cloud Credential Operator (CCO) によって AWS トークンを使用した短期認証がサポートされます。

次のコマンドを実行すると、Loki オブジェクトストレージシークレットを手動で作成できます。

```
$ oc -n openshift-logging create secret generic "logging-loki-aws" \
  --from-literal=bucketnames="<s3\_bucket\_name>" \
  --from-literal=region="<bucket\_region>" \
  --from-literal=audience="<oidc\_audience>" 1
```

- 1** 任意のアノテーション。デフォルト値は **openshift** です。

11.2.2.2. Azure ストレージ

前提条件

- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Azure 上に [バケット](#) を作成している。

手順

- 次のコマンドを実行して、**logging-loki-azure** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create secret generic logging-loki-azure \
  --from-literal=container="<azure\_container\_name>" \
  --from-literal=environment="<azure\_environment>" 1 \
  --from-literal=account_name="<azure\_account\_name>" \
  --from-literal=account_key="<azure\_account\_key>"
```

- 1 サポートされている環境値は、**AzureGlobal**、**AzureChinaCloud**、**AzureGermanCloud**、**AzureUSGovernment** です。

11.2.2.2.1. STS 対応クラスターの Azure ストレージ

クラスターで STS が有効になっている場合、Cloud Credential Operator (CCO) によって Azure AD Workload Identity を使用した短期認証がサポートされます。

次のコマンドを実行すると、Loki オブジェクトストレージシークレットを手動で作成できます。

```
$ oc -n openshift-logging create secret generic logging-loki-azure \
--from-literal=environment="<azure\_environment>" \
--from-literal=account_name="<storage\_account\_name>" \
--from-literal=container="<container\_name>"
```

11.2.2.3. Google Cloud Platform ストレージ

前提条件

- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Google Cloud Platform (GCP) 上に [プロジェクト](#) を作成している。
- 同じプロジェクト内に [バケット](#) を作成している。
- 同じプロジェクト内に GCP 認証用の [サービスアカウント](#) を作成している。

手順

1. GCP から受け取ったサービスアカウントの認証情報を **key.json** という名前のファイルにコピーします。
2. 次のコマンドを実行して、**logging-loki-gcs** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create secret generic logging-loki-gcs \
--from-literal=bucketname="<bucket\_name>" \
--from-file=key.json="<path/to/key.json>"
```

11.2.2.4. Minio ストレージ

前提条件

- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- [Minio](#) がクラスターにデプロイされている。
- Minio 上に [バケット](#) を作成している。

手順

- 次のコマンドを実行して、**logging-loki-minio** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create secret generic logging-loki-minio \
  --from-literal=bucketnames="<bucket_name>" \
  --from-literal=endpoint="<minio_bucket_endpoint>" \
  --from-literal=access_key_id="<minio_access_key_id>" \
  --from-literal=access_key_secret="<minio_access_key_secret>"
```

11.2.2.5. OpenShift Data Foundation ストレージ

前提条件

- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- [OpenShift Data Foundation](#) をデプロイしている。
- OpenShift Data Foundation クラスターを [オブジェクトストレージ用](#) に設定している。

手順

- openshift-logging** namespace に **ObjectBucketClaim** カスタムリソースを作成します。

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: loki-bucket-odf
  namespace: openshift-logging
spec:
  generateBucketName: loki-bucket-odf
  storageClassName: openshift-storage.noobaa.io
```

- 次のコマンドを実行して、関連付けられた **ConfigMap** オブジェクトからバケットのプロパティを取得します。

```
BUCKET_HOST=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_HOST}')
BUCKET_NAME=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_NAME}')
BUCKET_PORT=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_PORT}')
```

- 次のコマンドを実行して、関連付けられたシークレットからバケットアクセスキーを取得します。

```
ACCESS_KEY_ID=$(oc get -n openshift-logging secret loki-bucket-odf -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 -d)
SECRET_ACCESS_KEY=$(oc get -n openshift-logging secret loki-bucket-odf -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 -d)
```

- 次のコマンドを実行して、**logging-loki-odf** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create -n openshift-logging secret generic logging-loki-odf \
--from-literal=access_key_id="<access_key_id>" \
--from-literal=access_key_secret="<secret_access_key>" \
--from-literal=bucketnames="<bucket_name>" \
--from-literal=endpoint="https://<bucket_host>:<bucket_port>"
```

11.2.2.6. Swift ストレージ:

前提条件

- Loki Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Swift 上で **バケット** を作成している。

手順

- 次のコマンドを実行して、**logging-loki-swift** という名前のオブジェクトストレージシークレットを作成します。

```
$ oc create secret generic logging-loki-swift \
--from-literal=auth_url="<swift_auth_url>" \
--from-literal=username="<swift_usernameclaim>" \
--from-literal=user_domain_name="<swift_user_domain_name>" \
--from-literal=user_domain_id="<swift_user_domain_id>" \
--from-literal=user_id="<swift_user_id>" \
--from-literal=password="<swift_password>" \
--from-literal=domain_id="<swift_domain_id>" \
--from-literal=domain_name="<swift_domain_name>" \
--from-literal=container_name="<swift_container_name>"
```

- 必要に応じて、次のコマンドを実行して、プロジェクト固有のデータ、リージョン、またはその両方を指定できます。

```
$ oc create secret generic logging-loki-swift \
--from-literal=auth_url="<swift_auth_url>" \
--from-literal=username="<swift_usernameclaim>" \
--from-literal=user_domain_name="<swift_user_domain_name>" \
--from-literal=user_domain_id="<swift_user_domain_id>" \
--from-literal=user_id="<swift_user_id>" \
--from-literal=password="<swift_password>" \
--from-literal=domain_id="<swift_domain_id>" \
--from-literal=domain_name="<swift_domain_name>" \
--from-literal=container_name="<swift_container_name>" \
--from-literal=project_id="<swift_project_id>" \
--from-literal=project_name="<swift_project_name>" \
--from-literal=project_domain_id="<swift_project_domain_id>" \
--from-literal=project_domain_name="<swift_project_domain_name>" \
--from-literal=region="<swift_region>"
```

11.2.3. Elasticsearch ログストアのデプロイ

OpenShift Elasticsearch Operator を使用して、内部 Elasticsearch ログストアを OpenShift Dedicated クラスタにデプロイできます。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

11.2.3.1. Elasticsearch のストレージに関する考慮事項

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 Dedicated 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 リソース名を使用して PVC に名前を付けます。

Fluentd は **systemd ジャーナル** および `/var/log/containers/*.log` 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が必要です。十分なメモリがない場合は、応答なくなります。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。

11.2.3.2. Web コンソールを使用した OpenShift Elasticsearch Operator のインストール

OpenShift Elasticsearch Operator は、OpenShift Logging によって使用される Elasticsearch クラスタを作成し、管理します。

前提条件

- 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 16GB 以上のメモリーが必要です。
初期設定の OpenShift Dedicated ノードのセットは、Elasticsearch クラスタ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 64GB) を使用して実行できるようにノードを OpenShift Dedicated クラスタに追加する必要があります。

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。

- 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を注意してください。



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。

手順

1. OpenShift Dedicated Web コンソールで、**Operators** → **OperatorHub** をクリックします。
2. 利用可能な Operator のリストから **OpenShift Elasticsearch Operator**、**Install** の順にクリックします。
3. **All namespaces on the cluster** が **Installation Mode** で選択されていることを確認します。
4. **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 Dedicated メトリックと同じ名前でもトリック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。
5. **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。クラスタモニタリングが **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。
6. **Update Channel** として **stable-5.x** を選択します。
7. **Update 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。
8. **Install** をクリックします。

検証

1. **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。

2. **Status** が **Succeeded** の状態で、**OpenShift Elasticsearch Operator** がすべてのプロジェクトにリスト表示されていることを確認します。

11.2.3.3. CLI を使用して OpenShift Elasticsearch Operator をインストールする

OpenShift CLI (**oc**) を使用して、OpenShift Elasticsearch Operator をインストールできます。

前提条件

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



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 Dedicated は 16 GB のメモリー要求および制限を持つ 3 つの Elasticsearch ノードをインストールします。OpenShift Dedicated 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は、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。

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **Namespace** オブジェクトを、YAML ファイルとして作成します。

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

- 1** **openshift-operators-redhat** namespace を指定する必要があります。メトリクスとの競合が発生する可能性を防ぐには、Prometheus のクラスターモニタリングスタックを、**openshift-operators** namespace からではなく、**openshift-operators-redhat** namespace からメトリクスを収集するように設定します。**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 Dedicated メトリックと同じ名前でもトリック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。

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

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

3. **OperatorGroup** オブジェクトを、YAML ファイルとして作成します。

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

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

4. 以下のコマンドを実行して **OperatorGroup** オブジェクトを適用します。

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

5. OpenShift Elasticsearch Operator に namespace をサブスクライブするための **Subscription** オブジェクトを作成します。

Subscription の例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: elasticsearch-operator
  namespace: openshift-operators-redhat ❶
spec:
  channel: stable-x.y ❷
  installPlanApproval: Automatic ❸
  source: redhat-operators ❹
  sourceNamespace: openshift-marketplace
  name: elasticsearch-operator
```

- ❶ **openshift-operators-redhat** namespace を指定する必要があります。
- ❷ チャンネルとして **stable** または **stable-5.<x>** を指定します。以下の注意点を参照してください。
- ❸ **Automatic** 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** 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。
- ❹ **redhat-operators** を指定します。OpenShift Dedicated 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) の設定時に作成した **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。**stable** を **installPlanApproval: "Automatic"** とともに使用すると、Operator が最新の安定したメジャーリリースとマイナーリリースに自動的にアップグレードされます。

stable-x.y 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。**stable-x.y** を **installPlanApproval: "Automatic"** と併せて使用すると、Operator がメジャーリリース内の最新の stable マイナーリリースに自動的にアップグレードされます。

6. 次のコマンドを実行して、サブスクリプションを適用します。

```
$ oc apply -f <filename>.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。

検証

1. 以下のコマンドを実行します。

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

2. 出力を観察し、OpenShift Elasticsearch Operator の Pod が各 namespace に存在することを確認します。

出力例

NAMESPACE	VERSION	REPLACES	NAME	PHASE	DISPLAY
default			elasticsearch-operator.v5.8.1		OpenShift Elasticsearch
Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
kube-node-lease			elasticsearch-operator.v5.8.1		OpenShift
Elasticsearch Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
kube-public			elasticsearch-operator.v5.8.1		OpenShift Elasticsearch
Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
kube-system			elasticsearch-operator.v5.8.1		OpenShift Elasticsearch
Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
non-destructive-test			elasticsearch-operator.v5.8.1		OpenShift
Elasticsearch Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
openshift-apiserver-operator			elasticsearch-operator.v5.8.1		OpenShift
Elasticsearch Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
openshift-apiserver			elasticsearch-operator.v5.8.1		OpenShift
Elasticsearch Operator	5.8.1	elasticsearch-operator.v5.8.0		Succeeded	
...					

11.2.4. ログストレージの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログストレージのタイプを設定できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator と内部ログストア (LokiStack または Elasticsearch) がインストールされている。
- **ClusterLogging** CR が作成されている。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

手順

1. **ClusterLogging** CR の **logStore** 仕様を変更します。

ClusterLogging CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
logStore:
  type: <log_store_type> 1
  elasticsearch: 2
    nodeCount: <integer>
    resources: {}
    storage: {}
    redundancyPolicy: <redundancy_type> 3
  lokistack: 4
    name: {}
# ...

```

- 1 ログストアのタイプを指定します。これは **lokistack** または **elasticsearch** のいずれかです。
- 2 Elasticsearch ログストアの任意の設定オプション。
- 3 冗長性のタイプを指定します。この値には、**ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy**、または **FullRedundancy** を指定できます。
- 4 LokiStack の任意の設定オプション。

LokiStack をログストアとして指定する ClusterLogging CR の例

■

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
  lokistack:
    name: logging-loki
# ...

```

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

11.3. LOKISTACK ログストアの設定

ロギングのドキュメントでは、**LokiStack** はロギングでサポートされている Loki と Web プロキシと OpenShift Dedicated 認証統合の組み合わせを指します。LokiStack のプロキシは、OpenShift Dedicated 認証を使用してマルチテナンシーを適用します。Loki では、ログストアを個別のコンポーネントまたは外部ストアと呼んでいます。

11.3.1. cluster-admin ユーザーロールの新規グループの作成



重要

cluster-admin ユーザーとして複数の namespace のアプリケーションログをクエリーすると、クラスター内のすべての namespace の文字数の合計が 5120 を超え、**Parse error: input size too long (XXXX > 5120)** エラーが発生します。LokiStack のログへのアクセスをより適切に制御するには、**cluster-admin** ユーザーを **cluster-admin** グループのメンバーにします。**cluster-admin** グループが存在しない場合は、作成して必要なユーザーを追加します。

次の手順を使用して、**cluster-admin** 権限のあるユーザー用に、新しいグループを作成します。

手順

1. 以下のコマンドを入力して新規グループを作成します。

```
$ oc adm groups new cluster-admin
```

2. 以下のコマンドを実行して、必要なユーザーを **cluster-admin** グループに追加します。

```
$ oc adm groups add-users cluster-admin <username>
```

3. 以下のコマンドを実行して **cluster-admin** ユーザーロールをグループに追加します。

```
$ oc adm policy add-cluster-role-to-group cluster-admin cluster-admin
```

11.3.2. クラスターの再起動中の LokiStack 動作

ロギングバージョン 5.8 以降のバージョンでは、OpenShift Dedicated クラスターが再起動されると、LokiStack の取り込みとクエリーパスは、ノードで使用可能な CPU リソースとメモリーリソース内で動作し続けます。つまり、OpenShift Dedicated クラスターの更新中に LokiStack でダウンタイムは発生しません。この動作は、**PodDisruptionBudget** リソースを使用して実現されます。Loki Operator は、Loki に **PodDisruptionBudget** リソースをプロビジョニングするため、特定の条件下で通常の動作を保証するためにコンポーネントごとに必要最小限、使用可能な Pod 数が決定されます。

関連情報

- [Pod disruption budgets Kubernetes documentation](#)

11.3.3. ノードの障害を許容するための Loki の設定

ロギング 5.8 以降のバージョンでは、Loki Operator は、同じコンポーネントの Pod がクラスター内の異なる使用可能なノードにスケジュールされるように要求する Pod 非アフィニティールールを設定をサポートします。

アフィニティとは、スケジュールするノードを制御する Pod の特性です。非アフィニティとは、Pod がスケジュールされることを拒否する Pod の特性です。

OpenShift Dedicated では、**Pod のアフィニティ** と **Pod の非アフィニティ** によって、他の Pod のキー/値ラベルに基づいて、Pod のスケジュールに適したノードを制限することができます。

Operator は、すべての Loki コンポーネント

(**Compactor**、**Distribution**、**Gateway**、**IndexGateway**、**ingester**、**querier**、**queryFrontend**、および **Ruler** コンポーネントを含む) に対してデフォルトの優先 **podAntiAffinity** ルールを設定します。

requiredDuringSchedulingIgnoredDuringExecution フィールドに必要な設定を指定して、Loki コンポーネントの希望の **podAntiAffinity** 設定を上書きできます。

インジェスターコンポーネントのユーザー設定の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  # ...
  template:
    ingester:
      podAntiAffinity:
        # ...
        requiredDuringSchedulingIgnoredDuringExecution: ❶
        - labelSelector:
            matchLabels: ❷
              app.kubernetes.io/component: ingester
              topologyKey: kubernetes.io/hostname
        # ...
```

❶ 必要なルールを定義するスタンザです。

❷ ルールを適用するために一致している必要のあるキー/値のペア (ラベル) です。

関連情報

- [PodAntiAffinity v1 core Kubernetes documentation](#)
- [Assigning Pods to Nodes Kubernetes documentation](#)
- [アフィニティールールと非アフィニティールールの使用による他の Pod との相対での Pod の配置](#)

11.3.4. ゾーン対応のデータレプリケーション

ロギング 5.8 以降のバージョンでは、Loki Operator は Pod トポロジー分散制約を通じてゾーン対応のデータレプリケーションのサポートを提供します。この機能を有効にすると、信頼性が向上し、1つのゾーンで障害が発生した場合のログ損失に対する保護が強化されます。デプロイメントサイズを **1x.extra.small**、**1x.small**、または **1x.medium** に設定すると、**replication.factor** フィールドは自動的に 2 に設定されます。

適切なレプリケーションを実現するには、少なくともレプリケーション係数で指定されているのと同じ数のアベイラビリティゾーンが必要です。レプリケーション係数より多くのアベイラビリティゾーンを設定することは可能ですが、ゾーンが少ないと書き込みエラーが発生する可能性があります。最適な運用を実現するには、各ゾーンで同じ数のインスタンスをホストする必要があります。

ゾーンレプリケーションが有効になっている LokiStack CR の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  replicationFactor: 2 ①
  replication:
    factor: 2 ②
    zones:
      - maxSkew: 1 ③
        topologyKey: topology.kubernetes.io/zone ④
```

- ① 非推奨のフィールド。入力された値は **replication.factor** によって上書きされます。
- ② この値は、セットアップ時にデプロイメントサイズが選択されると自動的に設定されます。
- ③ 任意の 2 つのトポロジードメイン間の Pod 数の最大差。デフォルトは 1 で、0 の値を指定することはできません。
- ④ ノードラベルに対応するトポロジーキーの形式でゾーンを定義します。

11.3.4.1. 障害が発生したゾーンからの Loki Pod の回復

OpenShift Dedicated では、特定のアベイラビリティゾーンのリソースにアクセスできなくなると、ゾーン障害が発生します。アベイラビリティゾーンは、冗長性とフォールトトレランスを強化することを目的とした、クラウドプロバイダーのデータセンター内の分離されたエリアです。OpenShift Dedicated クラスタがこの問題を処理するように設定されていない場合、ゾーン障害によりサービスまたはデータの損失が発生する可能性があります。

Loki Pod は [StatefulSet](#) の一部であり、**StorageClass** オブジェクトによってプロビジョニングされた永続ボリューム要求 (PVC) が付属しています。各 Loki Pod とその PVC は同じゾーンに存在します。クラスターでゾーン障害が発生すると、StatefulSet コントローラーが、障害が発生したゾーン内の影響を受けた Pod の回復を自動的に試みます。



警告

次の手順では、障害が発生したゾーン内の PVC とそこに含まれるすべてのデータを削除します。完全なデータ損失を回避するには、**LokiStack** CR のレプリケーション係数フィールドを常に1より大きい値に設定して、Loki が確実にレプリケートされるようにする必要があります。

前提条件

- ロギングバージョン 5.8 以降。
- **LokiStack** CR のレプリケーション係数が1より大きいことを確認している。
- コントロールプレーンによってゾーン障害が検出され、障害が発生したゾーン内のノードがクラウドプロバイダー統合によってマークされている。

StatefulSet コントローラーは、障害が発生したゾーン内の Pod を自動的に再スケジュールしようとしません。関連する PVC も障害が発生したゾーンにあるため、別のゾーンへの自動再スケジュールは機能しません。新しいゾーンでステートフル Loki Pod とそのプロビジョニングされた PVC を正常に再作成できるようにするには、障害が発生したゾーンの PVC を手動で削除する必要があります。

手順

1. 次のコマンドを実行して、**Pending** 中ステータスの Pod をリスト表示します。

```
oc get pods --field-selector status.phase==Pending -n openshift-logging
```

oc get pods の出力例

```
NAME                READY STATUS RESTARTS AGE
logging-loki-index-gateway-1 0/1 Pending 0      17m
logging-loki-ingester-1    0/1 Pending 0      16m
logging-loki-ruler-1      0/1 Pending 0      16m
```

- 1 これらの Pod は、障害が発生したゾーンに対応する PVC があるため、**Pending** ステータスになっています。

2. 次のコマンドを実行して、**Pending** ステータスの PVC をリストします。

```
oc get pvc -o=json -n openshift-logging | jq '.items[] | select(.status.phase == "Pending") | .metadata.name' -r
```

oc get pvc の出力例

```
storage-logging-loki-index-gateway-1
storage-logging-loki-ingester-1
wal-logging-loki-ingester-1
storage-logging-loki-ruler-1
wal-logging-loki-ruler-1
```

3. 次のコマンドを実行して Pod の PVC を削除します。

```
oc delete pvc __<pvc_name>__ -n openshift-logging
```

4. 次のコマンドを実行して Pod を削除します。

```
oc delete pod __<pod_name>__ -n openshift-logging
```

これらのオブジェクトが正常に削除されると、使用可能なゾーンでオブジェクトが自動的に再スケジューリングされます。

11.3.4.1.1. terminating 状態の PVC のトラブルシューティング

PVC メタデータファイナライザーが **kubernetes.io/pv-protection** に設定されている場合、PVC が削除されずに terminating 状態でハングする可能性があります。ファイナライザーを削除すると、PVC が正常に削除されるようになります。

1. 以下のコマンドを実行して各 PVC のファイナライザーを削除し、削除を再実行します。

```
oc patch pvc __<pvc_name>__ -p '{"metadata":{"finalizers":null}}' -n openshift-logging
```

関連情報

- [トポロジー分散制約に関する Kubernetes ドキュメント](#)
- [Kubernetes ストレージのドキュメント](#)

11.3.5. Loki ログへのアクセスの詳細設定

ロギング 5.8 以降では、Red Hat OpenShift Logging Operator はデフォルトですべてのユーザーにログへのアクセスを許可しません。Operator のアップグレード後に以前の設定を適用していない限り、管理者はユーザーのアクセスを設定する必要があります。設定とニーズに応じて、以下を使用してログへのアクセスを細かく設定できます。

- クラスター全体のポリシー
- スコープ指定が namespace のポリシー
- カスタム管理者グループの作成

管理者は、デプロイメントに適したロールバインディングとクラスターのロールバインディングを作成する必要があります。Red Hat OpenShift Logging Operator には、次のクラスターロールがあります。

- **cluster-logging-application-view** は、アプリケーションログの読み取り権限を付与します。
- **cluster-logging-infrastructure-view** は、インフラストラクチャーログの読み取り権限を付与します。

- **cluster-logging-audit-view** は、監査ログの読み取り権限を付与します。

以前のバージョンからアップグレードした場合、追加のクラスターロール **logging-application-logs-reader** と関連するクラスターロールバインディング **logging-all-authenticated-application-logs-reader** により下位互換性が提供され、認証されたユーザーに namespace の読み取り権限が許可されます。



注記

namespace ごとのアクセス権を持つユーザーは、アプリケーションログをクエリーする際に namespace を提供する必要があります。

11.3.5.1. クラスター全体のアクセス

クラスターロールバインディングリソースはクラスターロールを参照し、クラスター全体に権限を設定します。

ClusterRoleBinding の例

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: logging-all-application-logs-reader
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-logging-application-view ❶
subjects: ❷
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

- ❶ **cluster-logging-infrastructor-view** および **cluster-logging-audit-view** は、追加の **ClusterRoles** です。
- ❷ このオブジェクトが適用されるユーザーまたはグループを指定します。

11.3.5.2. namespace アクセス

RoleBinding リソースを **ClusterRole** オブジェクトと使用して、ユーザーまたはグループがログにアクセスできる namespace を定義できます。

RoleBinding の例

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: allow-read-logs
  namespace: log-test-0 ❶
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-logging-application-view
```

```
subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: testuser-0
```

- 1 この **RoleBinding** が適用される namespace を指定します。

11.3.5.3. カスタム管理者グループのアクセス権

多数のユーザーが広範な権限を必要とする大規模デプロイメントの場合は、**adminGroup** フィールドを使用してカスタムグループを作成できます。**LokiStack** CR の **adminGroups** フィールドで指定されたグループのメンバーであるユーザーは、管理者とみなされます。**cluster-logging-application-view** ロールも割り当てられている管理者ユーザーは、すべての namespace のすべてのアプリケーションログにアクセスできます。

LokiStack CR の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  tenants:
    mode: openshift-logging 1
    openshift:
      adminGroups: 2
      - cluster-admin
      - custom-admin-group 3
```

- 1 カスタム管理者グループは、このモードでのみ使用できます。
- 2 このフィールドに空のリスト値 [] を入力すると、管理者グループが無効になります。
- 3 デフォルトのグループ (**system:cluster-admins**、**cluster-admin**、**dedicated-admin**) をオーバーライドします。

11.3.6. Loki でストリームベースの保持の有効化

関連情報

Logging バージョン 5.6 以降では、ログストリームに基づいて保持ポリシーを設定できます。これらのルールは、グローバル、テナントごと、またはその両方で設定できます。両方で設定すると、グローバルルールの前にテナントルールが適用されます。

- 1 ストリームベースの保持を有効にするには、**LokiStack** カスタムリソース (CR) を作成します。

グローバルなストリームベースの保持の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
```

```

namespace: openshift-logging
spec:
  limits:
    global: ①
    retention: ②
      days: 20
      streams:
        - days: 4
          priority: 1
          selector: '{kubernetes_namespace_name=~"test.+"}' ③
        - days: 1
          priority: 1
          selector: '{log_type="infrastructure"}'
  managementState: Managed
  replicationFactor: 1
  size: 1x.small
  storage:
    schemas:
      - effectiveDate: "2020-10-11"
        version: v11
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: standard
  tenants:
    mode: openshift-logging

```

- ① すべてのログストリームの保持ポリシーを設定します。注記: このフィールドは、オブジェクトストレージに保存されたログの保持期間には影響しません。
- ② このブロックが CR に追加されると、クラスターで保持が有効になります。
- ③ ログストリームの定義に使用される [LogQL クエリー](#) が含まれています。

テナントごとのストリームベースの保持の例

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      retention:
        days: 20
    tenants: ①
      application:
        retention:
          days: 1
        streams:
          - days: 4
            selector: '{kubernetes_namespace_name=~"test.+"}' ②
      infrastructure:

```

```

retention:
  days: 5
  streams:
    - days: 1
      selector: '{kubernetes_namespace_name=~"openshift-cluster.+"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
    - effectiveDate: "2020-10-11"
      version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: standard
tenants:
  mode: openshift-logging

```

- 1 テナントごとの保持ポリシーを設定します。有効なテナントタイプは、**application**、**audit**、および **infrastructure** です。
- 2 ログストリームの定義に使用される [LogQL クエリー](#) が含まれています。

2. LokiStack CR を適用します。

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



注記

これは、保存されたログの保持を管理するためのものではありません。保存されたログのグローバルな保持期間 (最大 30 日間までサポート) は、オブジェクトストレージで設定します。

11.3.7. Loki レート制限エラーのトラブルシューティング

Log Forwarder API がレート制限を超える大きなメッセージブロックを Loki に転送すると、Loki により、レート制限 (**429**) エラーが生成されます。

これらのエラーは、通常の動作中に発生する可能性があります。たとえば、すでにいくつかのログがあるクラスターにロギングを追加する場合、ロギングが既存のログエントリをすべて取り込もうとするとレート制限エラーが発生する可能性があります。この場合、新しいログの追加速度が合計レート制限よりも低い場合、履歴データは最終的に取り込まれ、ユーザーの介入を必要とせずにレート制限エラーが解決されます。

レート制限エラーが引き続き発生する場合は、**LokiStack** カスタムリソース (CR) を変更することで問題を解決できます。



重要

LokiStack CR は、Grafana がホストする Loki では利用できません。このトピックは、Grafana がホストする Loki サーバーには適用されません。

条件

- Log Forwarder API は、ログを Loki に転送するように設定されている。
- システムは、次のような 2MB を超えるメッセージのブロックを Loki に送信する。以下に例を示します。

```
"values":[[{"1630410392689800468","{"kind":"Event","apiVersion":"\n
.....
.....
.....
.....
\nreceived_at":"2021-08-31T11:46:32.800278+00:00","version":"1.7.4
1.6.0"}}, {"@timestamp":"2021-08-
31T11:46:32.799692+00:00","viaq_index_name":"audit-
write","viaq_msg_id":"MzFjYjkZjltNjY0MC0YWU4LWlwMTEtNGNmM2E5ZmViMGU4","lo
g_type":"audit"}]]}]}
```

- `oc logs -n openshift-logging -l component=collector` と入力すると、クラスター内のコレクターログに、次のいずれかのエラーメッセージを含む行が表示されます。

```
429 Too Many Requests Ingestion rate limit exceeded
```

Vector エラーメッセージの例

```
2023-08-25T16:08:49.301780Z WARN sink{component_kind="sink"
component_id=default_loki_infra component_type=loki component_name=default_loki_infra}:
vector::sinks::util::retries: Retrying after error. error=Server responded with an error: 429 Too
Many Requests internal_log_rate_limit=true
```

Fluentd エラーメッセージの例

```
2023-08-30 14:52:15 +0000 [warn]: [default_loki_infra] failed to flush the buffer. retry_times=2
next_retry_time=2023-08-30 14:52:19 +0000
chunk="604251225bf5378ed1567231a1c03b8b"
error_class=Fluent::Plugin::LokiOutput::LogPostError error="429 Too Many Requests
Ingestion rate limit exceeded for user infrastructure (limit: 4194304 bytes/sec) while
attempting to ingest '4082' lines totaling '7820025' bytes, reduce log volume or contact your
Loki administrator to see if the limit can be increased\n"
```

このエラーは受信側にも表示されます。たとえば、LokiStack 取り込み Pod で以下を行います。

Loki 取り込みエラーメッセージの例

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream
```

手順

- LokiStack CR の `ingestionBurstSize` および `ingestionRate` フィールドを更新します。

-

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 ❶
        ingestionRate: 8 ❷
# ...

```

- ❶ **ingestionBurstSize** フィールドは、ディストリビューターレプリカごとに最大ローカルレート制限サンプルサイズを MB 単位で定義します。この値はハードリミットです。この値を、少なくとも1つのプッシュリクエストで想定される最大ログサイズに設定します。**ingestionBurstSize** 値より大きい単一リクエストは使用できません。
- ❷ **ingestionRate** フィールドは、1秒あたりに取り込まれるサンプルの最大量 (MB 単位) に対するソフト制限です。ログのレートが制限を超えているにもかかわらず、コレクターがログの送信を再試行すると、レート制限エラーが発生します。合計平均が制限よりも少ない場合に限り、システムは回復し、ユーザーの介入なしでエラーが解決されます。

11.3.8. メンバーリストの作成の失敗を許容する Loki の設定

OpenShift クラスターでは、管理者は通常、非プライベート IP ネットワーク範囲を使用します。その結果、LokiStack メンバーリストはデフォルトでプライベート IP ネットワークのみを使用するため、LokiStack メンバーリストの設定は失敗します。

管理者は、メンバーリスト設定の Pod ネットワークを選択できます。**hashRing** 仕様で **podIP** を使用するように LokiStack CR を変更できます。LokiStack CR を設定するには、以下のコマンドを使用します。

```
$ oc patch LokiStack logging-loki -n openshift-logging --type=merge -p '{"spec": {"hashRing": {"memberlist":{"instanceAddrType":"podIP","type": "memberlist"}}}}'
```

podIPを含める LokiStack の例

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  hashRing:
    type: memberlist
    memberlist:
      instanceAddrType: podIP
# ...

```

11.3.9. 関連情報

- [Loki コンポーネントのドキュメント](#)
- [Loki クエリー言語 \(LogQL\) ドキュメント](#)
- [Grafana ダッシュボードのドキュメント](#)
- [Loki オブジェクトストレージのドキュメント](#)
- [Loki Operator `IngestionLimitSpec` のドキュメント](#)
- [Loki Storage Schema のドキュメント](#)

11.4. ELASTICSEARCH ログストアの設定

Elasticsearch 6 を使用して、ログデータを保存および整理できます。

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

11.4.1. ログストレージの設定

ClusterLogging カスタムリソース (CR) を変更することで、ロギングで使用するログストレージのタイプを設定できます。

前提条件

- 管理者権限がある。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift Logging Operator と内部ログストア (LokiStack または Elasticsearch) がインストールされている。
- **ClusterLogging** CR が作成されている。



注記

Logging 5.9 リリースに、OpenShift Elasticsearch Operator の更新バージョンは含まれていません。ロギング 5.8 でリリースされた OpenShift Elasticsearch Operator を現在使用している場合、Logging 5.8 の EOL まで引き続き Logging で機能します。OpenShift Elasticsearch Operator を使用してデフォルトのログストレージを管理する代わりに、Loki Operator を使用できます。Logging のライフサイクルの日付について、詳細は [Platform Agnostic Operator](#) を参照してください。

手順

1. **ClusterLogging** CR の **logStore** 仕様を変更します。

ClusterLogging CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  logStore:
    type: <log_store_type> ❶
    elasticsearch: ❷
      nodeCount: <integer>
      resources: {}
      storage: {}
      redundancyPolicy: <redundancy_type> ❸
    lokistack: ❹
      name: {}
# ...

```

- ❶ ログストアのタイプを指定します。これは **lokistack** または **elasticsearch** のいずれかです。
- ❷ Elasticsearch ログストアの任意の設定オプション。
- ❸ 冗長性のタイプを指定します。この値には、**ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy**、または **FullRedundancy** を指定できます。
- ❹ LokiStack の任意の設定オプション。

LokiStack をログストアとして指定する ClusterLogging CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
# ...

```

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

11.4.2. 監査ログのログストアへの転送

ロギングデプロイメントでは、デフォルトでコンテナおよびインフラストラクチャーのログは **ClusterLogging** カスタムリソース (CR) に定義された内部ログストアに転送されます。

セキュアなストレージを提供しないため、監査ログはデフォルトで内部ログストアに転送されません。お客様の責任において、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、適切に保護されていることを確認してください。

このデフォルト設定が要件を満たす場合、**ClusterLogForwarder** CR を設定する必要はありません。**ClusterLogForwarder** CR が存在する場合、**default** 出力を含むパイプラインが定義されている場合を除き、ログは内部ログストアに転送されません。

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 オブジェクトを定義する YAML ファイルを作成または編集します。

- 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 がある場合は、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 ❶

```

- ❶ 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

- [ログの収集と転送](#)

11.4.3. 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 × 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

- Red Hat OpenShift Logging Operator と 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、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 Dedicated 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 Dedicated がロールオーバーされたインデックスを削除するタイミング。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 Dedicated 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 Dedicated 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タイミング。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

11.4.4. 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 プロキシコンテナのデフォルトのメモリー制限が不十分な場合があります。これにより、プロキシコンテナが 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は、Elasticsearch プロキシ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**。実稼働環境で使用する場合は、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

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトで **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が、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、Elasticsearch 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)。

11.4.5. 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

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。

手順

1. **openshift-logging** プロジェクトで **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は、ログは利用不可となるか、失われる可能性があります。安全性よりもパフォーマンスを重視する場合や、独自のディスク/PVC バックアップ/復元戦略を実装している場合は、このモードを使用できます。



注記

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

11.4.6. 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は、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** に設定される場合は、Elasticsearch Pod をスケールダウンしないでください。

11.4.7. 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

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。

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。

11.4.8. 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 を使用する場合は、ログストアが再起動するか、再デプロイされる場合にデータが失われます。

前提条件

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。

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: {}

```

11.4.9. 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せん。

前提条件

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。

手順

クラスターのローリング再起動を実行するには、以下を実行します。

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

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

2. Elasticsearch Pod の名前を取得します。

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

3. コレクター Pod をスケールダウンして、Elasticsearch への新しいログの送信を停止します。

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

4. OpenShift Dedicated **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}}
```

5. OpenShift Dedicated **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 Dedicated 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がある場合は、次のデプロイメントに進むことができます。

```
$ oc get pods -l component=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。

- Elasticsearch 設定マップを変更した場合は、それぞれの Elasticsearch Pod に対してこれらの手順を繰り返します。
- クラスターのすべてのデプロイメントがロールアウトされたら、シャードのバランシングを再度有効にします。

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -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. 新しいログが Elasticsearch に送信されるように、コレクター Pod をスケールアップします。

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

11.4.10. 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、ロギング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 でルートを有効にできます。

re-encrypt ルート、Dedicated 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

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

手順

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: | ❶
```

- ❶ 次の手順でログストア CA 証明書を追加するか、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>"
}
```

11.4.11. 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は、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は、**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:  
    type: "fluentd"  
    fluentd: {}
```

4. コレクター Pod が再デプロイされたことを確認します。

```
$ oc get pods -l component=collector -n openshift-logging
```

第12章 ロギングアラート

12.1. デフォルトのロギングアラート

ロギングアラートは、Red Hat OpenShift Logging Operator のインストール中にインストールされます。アラートは、ログ収集およびログストレージバックエンドによってエクスポートされたメトリクスに依存します。これらのメトリクスは、Red Hat OpenShift Logging Operator のインストール時に、**Enable operator recommended cluster monitoring on this namespace** オプションを選択した場合に有効になります。ロギング Operator のインストールの詳細は、[Web コンソールを使用したロギングのインストール](#) を参照してください。

ローカルの Alertmanager インスタンスを無効にしていない限り、デフォルトのロギングアラートは、**openshift-monitoring** namespace の OpenShift Container Platform モニタリングスタック Alertmanager に送信されます。

12.1.1. Administrator および Developer パースペクティブでのアラート UI へのアクセス

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

- **Administrator** パースペクティブで、**Observe** → **Alerting** に移動します。このパースペクティブのアラート UI には主要なページが 3 つあり、それが **Alerts** ページ、**Silences** ページ、**Alerting rules** ページです。
- **Developer** パースペクティブで、**Observe** → **<project_name>** → **Alerts** に移動します。このパースペクティブのアラートでは、サイレンスおよびアラートルールはすべて **Alerts** ページで管理されます。**Alerts** ページに表示される結果は、選択されたプロジェクトに固有のものでず。



注記

Developer パースペクティブでは、**Project: <project_name>** リストでアクセス権のあるコア OpenShift Dedicated プロジェクトおよびユーザー定義プロジェクトから選択できます。ただし、クラスター管理者としてログインしていない場合、コア OpenShift Dedicated プロジェクトに関連するアラート、サイレンス、およびアラートルールは表示されません。

12.1.2. Vector コレクターのアラート

Logging 5.7 以降のバージョンでは、Vector コレクターによって次のアラートが生成されます。これらのアラートは、OpenShift Dedicated Web コンソールで表示できます。

表12.1 Vector コレクターのアラート

アラート	メッセージ	説明	重大度
CollectorHighErrorRate	<value> of records have resulted in an error by vector <instance>.	ベクター出力エラーの数は、デフォルトでは直前の 15 分間で 10 分を超えます。	Warning

アラート	メッセージ	説明	重大度
CollectorNodeDown	Prometheus could not scrape vector <instance> for more than 10m.	Vector は、Prometheus が特定の Vector インスタンスをスクレイピングできなかったと報告しています。	Critical
CollectorVeryHighError Rate	<value> of records have resulted in an error by vector <instance>.	Vector コンポーネントエラーの数は非常に多く、デフォルトでは過去 15 分間に 25 件を超えています。	Critical
FluentdQueueLengthIncreasing	In the last 1h, 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。	Warning

12.1.3. Fluentd コレクターのアラート

次のアラートは、従来の Fluentd ロ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 Dedicated Web コンソールで表示できます。

表12.2 Fluentd コレクターのアラート

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 1h, 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。	Warning
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

12.1.4. Elasticsearch アラートルール

これらのアラートルールは、OpenShift Dedicated Web コンソールで表示できます。

表12.3 アラートルール

アラート	説明	重大度
ElasticsearchClusterNotHealthy	クラスターのヘルスステータスは少なくとも 2m の間 RED になります。クラスターは書き込みを受け入れず、シャードが見つからない可能性があるか、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
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、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

アラート	説明	重大度
ElasticsearchProcessCPU High	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

12.1.5. 関連情報

- [コアプラットフォームのアラートルールの変更](#)

12.2. カスタムロギングアラート

Logging 5.7 以降のバージョンでは、ユーザーは、カスタマイズされたアラートと記録されたメトリクスを生成するように LokiStack デプロイメントを設定できます。カスタマイズされた [アラートおよび記録ルール](#) を使用する場合は、LokiStack ルーラーコンポーネントを有効にする必要があります。

LokiStack のログベースのアラートと記録されたメトリクスは、[LogQL](#) 式をルーラーコンポーネントに提供することによってトリガーされます。Loki Operator は、選択した LokiStack サイズ (**1x.extra-small**、**1x.small**、または **1x.medium**) に最適化されたルーラーを管理します。

これらの式を提供するには、Prometheus 互換の [アラートルール](#) を含む **AlertingRule** カスタムリソース (CR)、または Prometheus 互換の [記録ルール](#) を含む **RecordingRule** CR を作成する必要があります。

管理者は、**application**、**audit**、または **infrastructure** テナントのログベースのアラートまたは記録されたメトリクスを設定できます。管理者権限のないユーザーは、アクセス権のあるアプリケーションの **application** テナントに対してログベースのアラートまたは記録されたメトリクスを設定できます。

アプリケーション、監査、およびインフラストラクチャーのアラートは、ローカルの Alertmanager インスタンスを無効にしていない限り、デフォルトで **openshift-monitoring** namespace の OpenShift Dedicated モニタリングスタック Alertmanager に送信されます。**openshift-user-workload-monitoring** namespace でユーザー定義プロジェクトの監視に使用される Alertmanager が有効になっている場合、アプリケーションアラートはデフォルトでこの namespace の Alertmanager に送信されます。

12.2.1. ルーラーの設定

LokiStack ルーラーコンポーネントが有効になっている場合、ユーザーはログアラートや記録されたメトリクスをトリガーする [LogQL](#) 式のグループを定義できます。

管理者は、**LokiStack** カスタムリソース (CR) を変更することでルーラーを有効にできます。

前提条件

- Red Hat OpenShift Logging Operator と Loki Operator がインストールされている。
- **LokiStack** CR が作成されている。
- 管理者権限がある。

手順

- **LokiStack** CR に次の仕様設定が含まれていることを確認して、ルーラーを有効にします。

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: <name>
  namespace: <namespace>
spec:
  # ...
  rules:
    enabled: true ❶
    selector:
      matchLabels:
        openshift.io/<label_name>: "true" ❷
    namespaceSelector:
      matchLabels:
        openshift.io/<label_name>: "true" ❸

```

- ❶ クラスター内で Loki のアラートおよび記録ルールを有効にします。
- ❷ ログアラートとメトリクスの使用を有効にする namespace に追加できるカスタムラベルを追加します。
- ❸ ログアラートとメトリクスの使用を有効にする namespace に追加できるカスタムラベルを追加します。

12.2.2. LokiStack ルールの RBAC 権限の認可

管理者は、クラスターロールをユーザー名にバインドすることで、ユーザーが独自のアラートおよび記録ルールを作成および管理できるようにすることができます。クラスターロールは、ユーザーに必要なロールベースのアクセス制御 (RBAC) 権限を含む **ClusterRole** オブジェクトとして定義されます。

Logging 5.8 以降では、アラートおよび記録ルール用の次のクラスターロールを LokiStack で使用できません。

ルール名	説明
alertingrules.loki.grafana.com-v1-admin	このロールを持つユーザーは、アラートルールを管理する管理レベルのアクセス権を持ちます。このクラスターロールは、 loki.grafana.com/v1 API グループ内の AlertingRule リソースを作成、読み取り、更新、削除、リスト表示、および監視する権限を付与します。
alertingrules.loki.grafana.com-v1-crdview	このロールを持つユーザーは、 loki.grafana.com/v1 API グループ内の AlertingRule リソースに関連するカスタムリソース定義 (CRD) の定義を表示できますが、これらのリソースを変更または管理する権限を持ちません。
alertingrules.loki.grafana.com-v1-edit	このロールを持つユーザーは、 AlertingRule リソースを作成、更新、および削除する権限を持ちます。

ルール名	説明
alertingrules.loki.grafana.com-v1-view	このルールを持つユーザーは、 loki.grafana.com/v1 API グループ内の AlertingRule リソースを読み取ることができます。既存のアラートルールの設定、ラベル、およびアノテーションを検査できますが、それらを変更することはできません。
recordingrules.loki.grafana.com-v1-admin	このルールを持つユーザーは、記録ルールを管理する管理レベルのアクセス権を持ちます。このクラスタールールは、 loki.grafana.com/v1 API グループ内の RecordingRule リソースを作成、読み取り、更新、削除、リスト表示、および監視する権限を付与します。
recordingrules.loki.grafana.com-v1-crdview	このルールを持つユーザーは、 loki.grafana.com/v1 API グループ内の RecordingRule リソースに関連するカスタムリソース定義 (CRD) の定義を表示できますが、これらのリソースを変更または管理する権限を持ちません。
recordingrules.loki.grafana.com-v1-edit	このルールを持つユーザーは、 RecordingRule リソースを作成、更新、および削除する権限を持ちます。
recordingrules.loki.grafana.com-v1-view	このルールを持つユーザーは、 loki.grafana.com/v1 API グループ内の RecordingRule リソースを読み取ることができます。既存のアラートルールの設定、ラベル、およびアノテーションを検査できますが、それらを変更することはできません。

12.2.2.1. 例

ユーザーにクラスタールールを適用するには、既存のクラスタールールを特定のユーザー名にバインドする必要があります。

クラスタールールは、使用するロールバインディングの種類に応じて、クラスタースコープまたは namespace スコープにすることができます。 **RoleBinding** オブジェクトを使用する場合は、 **oc adm policy add-role-to-user** コマンドを使用する場合と同様に、クラスタールールが指定した namespace にのみ適用されます。 **ClusterRoleBinding** オブジェクトを使用する場合は、 **oc adm policy add-cluster-role-to-user** コマンドを使用する場合と同様に、クラスタールールがクラスター内のすべての namespace に適用されます。

次のコマンド例では、指定したユーザーに、クラスター内の特定の namespace のアラートルールに対する作成、読み取り、更新、および削除 (CRUD) 権限を付与します。

特定の namespace のアラートルールに対する CRUD 権限を付与するクラスターロールバインディングコマンドの例

```
$ oc adm policy add-role-to-user alertingrules.loki.grafana.com-v1-admin -n <namespace>
<username>
```

次のコマンドは、指定したユーザーに、すべての namespace のアラートルールに対する管理者権限を付与します。

管理者権限を付与するクラスターロールバインディングコマンドの例

```
$ oc adm policy add-cluster-role-to-user alertingrules.loki.grafana.com-v1-admin <username>
```

12.2.3. Loki を使用したログベースのアラートルールの作成

AlertingRule CR には、単一の **LokiStack** インスタンスのアラートルールグループを宣言するために使用する、仕様および Webhook 検証定義のセットが含まれます。Webhook 検証定義は、ルール検証条件もサポートします。

- **AlertingRule** CR に無効な **interval** 期間が含まれる場合、無効なアラートルールです。
- **AlertingRule** CR に無効な **for** 期間が含まれる場合、無効なアラートルールです。
- **AlertingRule** CR に無効な LogQL **expr** が含まれる場合、無効なアラートルールです。
- **AlertingRule** CR に同じ名前のグループが2つ含まれる場合、無効なアラートルールです。
- 上記のいずれにも当てはまらない場合、アラートルールは有効であるとみなされます。

テナントタイプ	AlertingRule CR の有効な namespace
application	
audit	openshift-logging
infrastructure	openshift-/*、kube-/*、default

前提条件

- Red Hat OpenShift Logging Operator 5.7 以降
- OpenShift Dedicated 4.13 以降

手順

1. **AlertingRule** カスタムリソース (CR) を作成します。

インフラストラクチャー AlertingRule CR の例

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
```

```

name: loki-operator-alerts
namespace: openshift-operators-redhat ❶
labels: ❷
  openshift.io/<label_name>: "true"
spec:
  tenantID: "infrastructure" ❸
  groups:
    - name: LokiOperatorHighReconciliationError
  rules:
    - alert: HighPercentageError
      expr: | ❹
        sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
kubernetes_pod_name=~"loki-operator-controller-manager.*"} |= "error" [1m])) by (job)
        /
        sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
kubernetes_pod_name=~"loki-operator-controller-manager.*"}[1m])) by (job)
        > 0.01
      for: 10s
      labels:
        severity: critical ❺
      annotations:
        summary: High Loki Operator Reconciliation Errors ❻
        description: High Loki Operator Reconciliation Errors ❼

```

- ❶ この **AlertingRule** CR が作成される namespace には、LokiStack **spec.rules.namespaceSelector** 定義に一致するラベルが必要です。
- ❷ **labels** ブロックは、LokiStack の **spec.rules.selector** 定義と一致する必要があります。
- ❸ **infrastructure** テナントの **AlertingRule** CR は、**openshift-***、**kube-***、または **default namespaces** でのみサポートされます。
- ❹ **kubernetes_namespace_name:** の値は、**metadata.namespace** の値と一致する必要があります。
- ❺ この必須フィールドの値は、**critical**、**warning**、または **info** である必要があります。
- ❻ このフィールドは必須です。
- ❼ このフィールドは必須です。

アプリケーション AlertingRule CR の例

```

apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: app-user-workload
  namespace: app-ns ❶
  labels: ❷
    openshift.io/<label_name>: "true"
spec:
  tenantID: "application"
  groups:
    - name: AppUserWorkloadHighError

```

```

rules:
  - alert:
      expr: | 3
        sum(rate({kubernetes_namespace_name="app-ns",
kubernetes_pod_name=~"podName.*"} |= "error" [1m])) by (job)
      for: 10s
      labels:
        severity: critical 4
      annotations:
        summary: 5
        description: 6

```

- 1** この **AlertingRule** CR が作成される namespace には、LokiStack **spec.rules.namespaceSelector** 定義に一致するラベルが必要です。
- 2** **labels** ブロックは、LokiStack の **spec.rules.selector** 定義と一致する必要があります。
- 3** **kubernetes_namespace_name:** の値は、**metadata.namespace** の値と一致する必要があります。
- 4** この必須フィールドの値は、**critical**、**warning**、または **info** である必要があります。
- 5** この必須フィールドの値は、ルールの概要です。
- 6** この必須フィールドの値は、ルールの詳細な説明です。

2. **AlertingRule** CR を適用します。

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

12.2.4. 関連情報

- [OpenShift Dedicated モニタリングについて](#)

第13章 パフォーマンスと信頼性のチューニング

13.1. フロー制御メカニズム

ログの生成速度が収集できる速度よりも速い場合、出力に送信されるログの量の予測や制御が困難になることがあります。出力に送信されるログの量を予測または制御できないと、ログが失われる可能性があります。システムの停止が発生し、ユーザーの制御なしにログバッファが蓄積されると、接続が復元されるときに回復時間と遅延が長くなることもあります。

管理者は、ログのフロー制御メカニズムを設定することで、ログの速度を制限できます。

13.1.1. フロー制御メカニズムの利点

- ログのコストと量をより正確に事前予測できます。
- ノイズの多いコンテナが無制限に生成するログトラフィックにより、他のコンテナのログが埋もれることがなくまります。
- 価値の低いログを無視することで、ロギングインフラストラクチャーの負荷が軽減されます。
- レート制限を引き上げることで、値の高いログを値の低いログよりも優先することができます。

13.1.2. レート制限の設定

レート制限はコレクターごとに設定されます。つまり、ログ収集の最大レートはコレクターインスタンスの数にレート制限を掛けたものになります。

ログは各ノードのファイルシステムから収集されるため、各クラスターノードにコレクターがデプロイされます。たとえば、3 ノードクラスターでは、コレクターあたりの最大レート制限が1秒あたり10レコードの場合、ログ収集の最大レートは1秒あたり30レコードになります。

出力に書き込まれるレコードの正確なバイトサイズは、変換、エンコーディングの違い、その他の要因によって異なる可能性があるため、レート制限はバイト数ではなくレコード数で設定されます。

ClusterLogForwarder カスタムリソース (CR) でレート制限を設定するには、次の2つの方法があります。

出力レート制限

出力のネットワークやストレージ容量などに合わせて、選択した出力への送信ログの速度を制限します。出力レート制限では、出力ごとの集約レートを制御します。

入力レート制限

選択したコンテナのコンテナごとのログ収集レートを制限します。

13.1.3. ログフォワーダーの出力レート制限の設定

ClusterLogForwarder カスタムリソース (CR) を設定することで、送信ログのレートを指定の出力に制限できます。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。

- 管理者権限がある。

手順

1. 特定の出力の **ClusterLogForwarder** CR に **maxRecordsPerSecond** 制限値を追加します。次の例は、**kafka-example** という名前の Kafka ブローカー出力のコレクターごとの出力レート制限を設定する方法を示しています。

ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  # ...
spec:
  # ...
  outputs:
    - name: kafka-example ❶
      type: kafka ❷
      limit:
        maxRecordsPerSecond: 1000000 ❸
  # ...

```

- ❶ 出力名。
- ❷ 出力のタイプ。
- ❸ ログの出力レート制限。この値は、1秒あたりに Kafka ブローカーに送信できるログの **最大量** を設定します。この値はデフォルトでは設定されていません。デフォルトの動作はベストエフォートであり、ログフォワーダーが処理が追いつかない場合、レコードが削除されます。この値が **0** の場合、ログは転送されません。

2. **ClusterLogForwarder** CR を適用します。

コマンドの例

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

関連情報

- [ログ出力のタイプ](#)

13.1.4. ログフォワーダーの入力レート制限の設定

ClusterLogForwarder カスタムリソース (CR) を設定することで、収集される受信ログの速度を制限できます。コンテナごとまたは namespace ごとに入力制限を設定できます。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。

手順

1. 特定の入力の **ClusterLogForwarder** CR に **maxRecordsPerSecond** 制限値を追加します。さまざまなシナリオで入力レート制限を設定する方法を以下に例示します。

特定のラベルを持つコンテナに対してコンテナごとの制限を設定する ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
inputs:
- name: <input_name> ❶
  application:
  selector:
    matchLabels: { example: label } ❷
  containerLimit:
    maxRecordsPerSecond: 0 ❸
# ...

```

- ❶ 入力の名前。
- ❷ ラベルのリスト。これらのラベルが Pod に適用されているラベルと一致する場合、**maxRecordsPerSecond** フィールドに指定したコンテナごとの制限がそれらのコンテナに適用されます。
- ❸ レート制限を設定します。**maxRecordsPerSecond** フィールドを **0** に設定すると、コンテナでログが収集されません。**maxRecordsPerSecond** フィールドを他の値に設定すると、コンテナで1秒あたりの最大数のレコードが収集されます。

選択した namespace 内のコンテナごとに制限を設定する ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
inputs:
- name: <input_name> ❶
  application:
    namespaces: [ example-ns-1, example-ns-2 ] ❷
  containerLimit:
    maxRecordsPerSecond: 10 ❸
- name: <input_name>
  application:
    namespaces: [ test ]

```

```

containerLimit:
  maxRecordsPerSecond: 1000
# ...

```

- 1 入力の名前。
- 2 namespace のリスト。 **maxRecordsPerSecond** フィールドで指定したコンテナごとの制限が、リストした namespace 内のすべてのコンテナに適用されます。
- 3 レート制限を設定します。 **maxRecordsPerSecond** フィールドを **10** に設定すると、リストした namespace 内の各コンテナで1秒あたり最大10レコードが収集されます。

2. ClusterLogForwarder CR を適用します。

コマンドの例

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

13.2. コンテンツによるログのフィルタリング

クラスターからすべてのログを収集すると、大量のデータが生成され、転送や保存にコストがかかる可能性があります。

保存する必要のない優先度の低いデータをフィルタリングすることで、ログデータの容量を削減できます。Logging ではコンテンツフィルターが提供されており、ログデータの量を減らすことができます。



注記

コンテンツフィルターは **input** セレクターとは異なります。 **input** セレクターは、ソースメタデータに基づいてログストリーム全体を選択または無視します。コンテンツフィルターはログストリームを編集して、レコードの内容に基づいてレコードを削除および変更します。

ログデータの量は、次のいずれかの方法を使用して削減できます。

- [不要なログレコードを削除するコンテンツフィルターの設定](#)
- [ログレコードを削除するコンテンツフィルターの設定](#)

13.2.1. 不要なログレコードを削除するコンテンツフィルターの設定

drop フィルターが設定されている場合、ログコレクターは転送する前にフィルターに従ってログストリームを評価します。コレクターは、指定された設定に一致する不要なログレコードを削除します。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- **ClusterLogForwarder** カスタムリソース (CR) を作成した。

手順

1. フィルターの設定を **ClusterLogForwarder** CR の **filters** 仕様に追加します。
以下の例は、正規表現に基づいてログレコードを削除するように **ClusterLogForwarder** CR を設定する方法を示しています。

ClusterLogForwarder CR の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
    - name: <filter_name>
      type: drop ①
      drop: ②
        test: ③
          - field: .kubernetes.labels."foo-bar/baz" ④
            matches: .+ ⑤
          - field: .kubernetes.pod_name
            notMatches: "my-pod" ⑥
  pipelines:
    - name: <pipeline_name> ⑦
      filterRefs: ["<filter_name>"]
# ...

```

- ① フィルターの種類を指定します。**drop** フィルターは、フィルター設定に一致するログレコードをドロップします。
- ② **drop** フィルターを適用するための設定オプションを指定します。
- ③ ログレコードが削除されるかどうかを評価するために使用されるテストの設定を指定します。
 - テストに指定されたすべての条件が **true** の場合、テストは合格し、ログレコードは削除されます。
 - **drop** フィルター設定に複数のテストが指定されている場合、いずれかのテストに合格すると、レコードは削除されます。
 - 条件の評価中にエラーが発生した場合 (たとえば、評価対象のログレコードにフィールドがない場合)、その条件は **false** と評価されます。
- ④ ドットで区切られたフィールドパス (ログレコード内のフィールドへのパス) を指定します。パスには、英数字とアンダースコア (**a-zA-Z0-9_**) を含めることができます (例: **.kubernetes.namespace_name**)。セグメントにこの範囲外の文字が含まれている場合、セグメントを引用符で囲む必要があります (例: **.kubernetes.labels."foo.bar-bar/baz"**)。1 つの **test** 設定に複数のフィールドパスを含めることができますが、テストに合格して **drop** フィルターを適用するには、すべてのフィールドパスが **true** と評価される必要があります。
- ⑤ 正規表現を指定します。ログレコードがこの正規表現と一致する場合は、破棄されます。単一の **field** パスに対して **matches** または **notMatches** 条件のいずれかを設定できますが、両方を設定することはできません。
- ⑥

正規表現を指定します。ログレコードがこの正規表現に一致しない場合、破棄されます。単一の **field** パスに対して **matches** または **notMatches** 条件のいずれかを設定できます

7 drop フィルターが適用されるパイプラインを指定します。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

追加例

次の例は、優先度の高いログレコードのみを保持するように **drop** フィルターを設定する方法を示しています。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: important
    type: drop
    drop:
      test:
      - field: .message
        notMatches: "(?!critical|error)"
      - field: .level
        matches: "info|warning"
# ...
```

単一の **test** 設定に複数のフィールドパスを追加する以外に、**OR** チェックとして扱われる追加のテストも追加できます。次の例では、いずれかの **test** 設定が true と評価されるとレコードが削除されます。ただし、2 番目の **test** 設定では、true と評価されるためには、両方のフィールド仕様が true である必要があります。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: important
    type: drop
    drop:
      test:
      - field: .kubernetes.namespace_name
        matches: "^open"
      test:
      - field: .log_type
        matches: "application"
      - field: .kubernetes.pod_name
        notMatches: "my-pod"
# ...
```

13.2.2. ログレコードを削除するコンテンツフィルターの設定

prune フィルターが設定されると、ログコレクターは転送前にフィルターをもとにログレベルを評価します。コレクターは、Pod アノテーションなどの値の低いフィールドを削除してログレコードを整理します。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- **ClusterLogForwarder** カスタムリソース (CR) を作成した。

手順

1. フィルターの設定を **ClusterLogForwarder** CR の **prune** 仕様に追加します。次の例は、フィールドパスに基づいてログレコードを削除するように **ClusterLogForwarder** CR を設定する方法を示しています。



重要

両方が指定されている場合、最初に **notIn** 配列に基づいてレコードが整理され、**in** 配列よりも優先されます。**notIn** 配列を使用してレコードが整理された後、**in** 配列を使用してレコードが整理されます。

ClusterLogForwarder CR の例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  # ...
spec:
  filters:
    - name: <filter_name>
      type: prune ①
      prune: ②
        in: [.kubernetes.annotations, .kubernetes.namespace_id] ③
        notIn: [.kubernetes,.log_type,.message,."@timestamp"] ④
  pipelines:
    - name: <pipeline_name> ⑤
      filterRefs: ["<filter_name>"]
  # ...
```

- ① フィルターのタイプを指定します。**prune** フィルターでは、設定されたフィールドでログレコードをプルーニングします。
- ② **prune** フィルターを適用するための設定オプションを指定します。**in** フィールドと **notIn** フィールドは、ログレコード内のフィールドへのパスであるドット区切りのフィールドパスの配列として指定されます。これらのパスには、英数字とアンダースコア (**a-zA-Z0-9_**) を含めることができます (例: **.kubernetes.namespace_name**)。セグメントにこの範囲外の文字が含まれている場合、セグメントを引用符で囲む必要があります (例: **.kubernetes.labels."foo.bar-bar/baz"**)。

- 3 オプション: この配列で指定されたフィールドはすべてログレコードから削除されます。
- 4 オプション: この配列で指定されていないフィールドはログレコードから削除されます。
- 5 **prune** フィルターを適用するパイプラインを指定します。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

13.2.3. 関連情報

- [ログのサードパーティーシステムへの転送](#)

13.3. メタデータによるログのフィルタリング

input セレクターを使用して、**ClusterLogForwarder** CR でログをフィルタリングし、メタデータに基づいてログストリーム全体を選択または無視できます。管理者または開発者は、ログ収集を含めるか除外して、コレクターのメモリーと CPU 負荷を軽減できます。



重要

この機能は、ロギングのデプロイメントで Vector コレクターが設定されている場合にのみ使用できます。



注記

input 仕様フィルタリングはコンテンツフィルタリングとは異なります。**input** セレクターは、ソースメタデータに基づいてログストリーム全体を選択または無視します。コンテンツフィルターはログストリームを編集し、レコードの内容に基づいてレコードを削除および変更します。

13.3.1. namespace またはコンテナ名を含めるか除外して入力時にアプリケーションログをフィルタリングする手順

input セレクターを使用して、namespace とコンテナ名に基づいてアプリケーションログを含めたり除外したりできます。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- **ClusterLogForwarder** カスタムリソース (CR) を作成した。

手順

1. **ClusterLogForwarder** CR に namespace とコンテナ名を含めるか除外するかの設定を追加します。
以下の例は、namespace およびコンテナ名を含めるか、除外するように **ClusterLogForwarder** CR を設定する方法を示しています。

ClusterLogForwarder CR の例

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs
      application:
        includes:
          - namespace: "my-project" ❶
            container: "my-container" ❷
        excludes:
          - container: "other-container*" ❸
            namespace: "other-namespace" ❹
# ...

```

- ❶ ログがこれらの namespace からのみ収集されることを指定します。
- ❷ ログがこれらのコンテナからのみ収集されることを指定します。
- ❸ ログを収集するときに無視する namespace のパターンを指定します。
- ❹ ログを収集するときに無視するコンテナのセットを指定します。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

excludes オプションは、**include** オプションよりも優先されます。

13.3.2. ラベル式または一致するラベルキーと値を含む入力 **ny** でのアプリケーションログのフィルタリング

input セレクターを使用して、ラベル式または一致するラベルキーとその値に基づいてアプリケーションログを含めることができます。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- **ClusterLogForwarder** カスタムリソース (CR) を作成した。

手順

1. **ClusterLogForwarder** CR の **input** 仕様にフィルターの設定を追加します。以下の例は、ラベル式または一致したラベルキー/値に基づいてログを組み込むように **ClusterLogForwarder** CR を設定する方法を示しています。

ClusterLogForwarder CR の例

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs
      application:
        selector:
          matchExpressions:
            - key: env ①
              operator: In ②
              values: ["prod", "qa"] ③
            - key: zone
              operator: NotIn
              values: ["east", "west"]
          matchLabels: ④
            app: one
            name: app1
# ...

```

- ① 照合するラベルキーを指定します。
- ② Operator を指定します。有効な値には、**In**、**NotIn**、**Exists**、**DoesNotExist** などがあります。
- ③ 文字列値の配列を指定します。**Operator** 値が **Exists** または **DoesNotExist** のいずれかの場合、値の配列は空である必要があります。
- ④ 正確なキーまたは値のマッピングを指定します。

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

13.3.3. ソースによる監査およびインフラストラクチャーログ入力のフィルタリング

input セレクターを使用して、ログを収集する **audit** および **infrastructure** ソースのリストを定義できます。

前提条件

- Red Hat OpenShift Logging Operator がインストールされている。
- 管理者権限がある。
- **ClusterLogForwarder** カスタムリソース (CR) を作成した。

手順

1. **ClusterLogForwarder** CR に **audit** および **infrastructure** ソースを定義する設定を追加します。
次の例は、**ClusterLogForwarder** CR を設定して **audit** および **infrastructure** ソースを定義する方法を示しています。

ClusterLogForwarder CR の例

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs1
      infrastructure:
        sources: ❶
        - node
    - name: mylogs2
      audit:
        sources: ❷
        - kubeAPI
        - openshiftAPI
        - ovn
# ...
```

- ❶ 収集するインフラストラクチャーソースのリストを指定します。有効なソースには以下が含まれます。
 - **node**: ノードからのジャーナルログ
 - **container**: namespace にデプロイされたワークロードからのログ
- ❷ 収集する audit ソースのリストを指定します。有効なソースには以下が含まれます。
 - **Kubeapi: Kubernetes API** サーバーからのログ
 - **openshiftAPI**: OpenShift API サーバーからのログ
 - **Auditd**: ノードの Auditd サービスからのログ
 - **ovn**: オープン仮想ネットワークサービスからのログ

2. 次のコマンドを実行して、**ClusterLogForwarder** CR を適用します。

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

第14章 スケジューリングリソース

14.1. ノードセクターを使用したロギングリソースの移動

ノードセクターは、ノードのカスタムラベルと Pod で指定されるセクターを使用して定義されるキー/値のペアのマップを指定します。

Pod がノードで実行する要件を満たすには、Pod にはノードのラベルと同じキー/値のペアがなければなりません。

14.1.1. ノードセクターについて

Pod でノードセクターを使用し、ノードでラベルを使用して、Pod がスケジュールされる場所を制御できます。ノードセクターを使用すると、OpenShift Dedicated は一致するラベルが含まれるノード上に Pod をスケジュールします。

ノードセクターを使用して特定の Pod を特定のノードに配置し、クラスタースコープのノードセクターを使用して特定ノードの新規 Pod をクラスター内の任意の場所に配置し、プロジェクトノードを使用して新規 Pod を特定ノードのプロジェクトに配置できます。

たとえば、クラスター管理者は、作成するすべての Pod にノードセクターを追加して、アプリケーション開発者が地理的に最も近い場所にあるノードにのみ Pod をデプロイできるインフラストラクチャーを作成できます。この例では、クラスターは2つのリージョンに分散する5つのデータセンターで構成されます。米国では、ノードに **us-east**、**us-central**、または **us-west** のラベルを付けます。アジア太平洋リージョン (APAC) では、ノードに **apac-east** または **apac-west** のラベルを付けます。開発者は、Pod がこれらのノードにスケジュールされるように、作成する Pod にノードセクターを追加できます。

Pod オブジェクトにノードセクターが含まれる場合でも、一致するラベルを持つノードがない場合、Pod はスケジュールされません。

重要

同じ Pod 設定でノードセクターとノードのアフィニティーを使用している場合は、以下のルールが Pod のノードへの配置を制御します。

- **nodeSelector** と **nodeAffinity** の両方を設定する場合、Pod が候補ノードでスケジュールされるにはどちらの条件も満たしている必要があります。
- **nodeAffinity** タイプに関連付けられた複数の **nodeSelectorTerms** を指定する場合、**nodeSelectorTerms** のいずれかが満たされている場合に Pod をノードにスケジュールすることができます。
- **nodeSelectorTerms** に関連付けられた複数の **matchExpressions** を指定する場合、すべての **matchExpressions** が満たされている場合にのみ Pod をノードにスケジュールすることができます。

特定の Pod およびノードのノードセクター

ノードセクターおよびラベルを使用して、特定の Pod がスケジュールされるノードを制御できます。

ノードセクターおよびラベルを使用するには、まずノードにラベルを付けて Pod がスケジュール解除されないようにしてから、ノードセクターを Pod に追加します。



注記

ノードセレクターを既存のスケジュールされている Pod に直接追加することはできません。デプロイメント設定などの Pod を制御するオブジェクトにラベルを付ける必要があります。

たとえば、以下の **Node** オブジェクトには **region: east** ラベルがあります。

ラベルを含む Node オブジェクトのサンプル

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-131-14.ec2.internal
  selfLink: /api/v1/nodes/ip-10-0-131-14.ec2.internal
  uid: 7bc2580a-8b8e-11e9-8e01-021ab4174c74
  resourceVersion: '478704'
  creationTimestamp: '2019-06-10T14:46:08Z'
  labels:
    kubernetes.io/os: linux
    topology.kubernetes.io/zone: us-east-1a
    node.openshift.io/os_version: '4.5'
    node-role.kubernetes.io/worker: ""
    topology.kubernetes.io/region: us-east-1
    node.openshift.io/os_id: rhcos
    node.kubernetes.io/instance-type: m4.large
    kubernetes.io/hostname: ip-10-0-131-14
    kubernetes.io/arch: amd64
    region: east ①
    type: user-node
#...
```

① Pod ノードセレクターに一致するラベル。

Pod には **type: user-node,region: east** ノードセレクターがあります。

ノードセレクターが含まれる Pod オブジェクトのサンプル

```
apiVersion: v1
kind: Pod
metadata:
  name: s1
#...
spec:
  nodeSelector: ①
    region: east
    type: user-node
#...
```

① ノードトラベルに一致するノードセレクター。ノードには、各ノードセレクターのラベルが必要です。

サンプル Pod 仕様を使用して Pod を作成する場合、これはサンプルノードでスケジュールできません。

クラスタースコープのデフォルトノードセレクター

デフォルトのクラスタースコープのノードセレクターを使用する場合、クラスターで Pod を作成すると、OpenShift Dedicated はデフォルトのノードセレクターを Pod に追加し、一致するラベルのあるノードで Pod をスケジュールします。

たとえば、以下の **Scheduler** オブジェクトにはデフォルトのクラスタースコープの **region=east** および **type=user-node** ノードセレクターがあります。

スケジューラー Operator カスタムリソースの例

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
#...
spec:
  defaultNodeSelector: type=user-node,region=east
#...
```

クラスター内のノードには **type=user-node,region=east** ラベルがあります。

Node オブジェクトの例

```
apiVersion: v1
kind: Node
metadata:
  name: ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
#...
labels:
  region: east
  type: user-node
#...
```

ノードセレクターを持つ Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  name: s1
#...
spec:
  nodeSelector:
    region: east
#...
```

サンプルクラスターでサンプル Pod 仕様を使用して Pod を作成する場合、Pod はクラスタースコープのノードセレクターで作成され、ラベルが付けられたノードにスケジュールされます。

ラベルが付けられたノード上の Pod を含む Pod リストの例

```
NAME      READY   STATUS    RESTARTS   AGE   IP           NODE
```

NOMINATED NODE READINESS GATES

```
pod-s1 1/1 Running 0 20s 10.131.2.6 ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
<none> <none>
```

**注記**

Pod を作成するプロジェクトにプロジェクトノードセレクターがある場合、そのセレクターはクラスタースコープのセレクターよりも優先されます。Pod にプロジェクトノードセレクターがない場合、Pod は作成されたり、スケジュールされたりしません。

プロジェクトノードセレクター

プロジェクトノードセレクターを使用する場合、このプロジェクトで Pod を作成すると、OpenShift Dedicated はノードセレクターを Pod に追加し、一致するラベルを持つノードで Pod をスケジュールします。クラスタースコープのデフォルトノードセレクターがない場合、プロジェクトノードセレクターが優先されます。

たとえば、以下のプロジェクトには **region=east** ノードセレクターがあります。

Namespace オブジェクトの例

```
apiVersion: v1
kind: Namespace
metadata:
  name: east-region
  annotations:
    openshift.io/node-selector: "region=east"
#...
```

以下のノードには **type=user-node,region=east** ラベルがあります。

Node オブジェクトの例

```
apiVersion: v1
kind: Node
metadata:
  name: ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
#...
labels:
  region: east
  type: user-node
#...
```

Pod をこのサンプルプロジェクトでサンプル Pod 仕様を使用して作成する場合、Pod はプロジェクトノードセレクターで作成され、ラベルが付けられたノードにスケジュールされます。

Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  namespace: east-region
#...
```

```
spec:
  nodeSelector:
    region: east
    type: user-node
#...
```

ラベルが付けられたノード上の Pod を含む Pod リストの例

```
NAME      READY  STATUS   RESTARTS  AGE  IP           NODE
NOMINATED NODE  READINESS GATES
pod-s1    1/1    Running  0          20s  10.131.2.6  ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
<none>    <none>
```

Pod に異なるノードセクターが含まれる場合、プロジェクトの Pod は作成またはスケジュールされません。たとえば、以下の Pod をサンプルプロジェクトにデプロイする場合、これは作成されません。

無効なノードセクターを持つ Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  name: west-region
#...
spec:
  nodeSelector:
    region: west
#...
```

14.1.2. Loki Pod の配置

Pod の容認またはノードセクターを使用して、Loki Pod が実行するノードを制御し、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できます。

LokiStack 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、ノード仕様を使用してテイントをノードに適用できます。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペアです。他の Pod にはない特定の **key:value** ペアを使用すると、ログストア Pod ののみがそのノードで実行できるようになります。

ノードセクターを使用する LokiStack の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
template:
  compactor: 1
  nodeSelector:
    node-role.kubernetes.io/infra: "" 2
  distributor:
```

```
nodeSelector:
  node-role.kubernetes.io/infra: ""
gateway:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
indexGateway:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
ingester:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
querier:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
queryFrontend:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
ruler:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
# ...
```

- 1 ノードセクターに適用されるコンポーネント Pod タイプを指定します。
- 2 定義されたラベルが含まれるノードに移動する Pod を指定します。

前述の設定例では、すべての Loki Pod が **node-role.kubernetes.io/infra: ""** ラベルを含むノードに移動されます。

ノードセクターと容認を使用する LokiStack CR の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
template:
  compactor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  distributor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
```

```
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
key: node-role.kubernetes.io/infra
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
indexGateway:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
key: node-role.kubernetes.io/infra
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
ingester:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
key: node-role.kubernetes.io/infra
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
querier:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
key: node-role.kubernetes.io/infra
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
queryFrontend:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
key: node-role.kubernetes.io/infra
value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved
ruler:
nodeSelector:
  node-role.kubernetes.io/infra: ""
```

```

tolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/infra
  value: reserved
- effect: NoExecute
  key: node-role.kubernetes.io/infra
  value: reserved
gateway:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/infra
  value: reserved
- effect: NoExecute
  key: node-role.kubernetes.io/infra
  value: reserved
# ...

```

LokiStack (CR) の **nodeSelector** フィールドと **tolerations** フィールドを設定するには、**oc explain** コマンドを使用して、特定のリソースの説明とフィールドを表示します。

```
$ oc explain lokistack.spec.template
```

出力例

```

KIND:   LokiStack
VERSION: loki.grafana.com/v1

RESOURCE: template <Object>

DESCRIPTION:
  Template defines the resource/limits/tolerations/nodeselectors per
  component

FIELDS:
  compactor <Object>
    Compactor defines the compaction component spec.

  distributor <Object>
    Distributor defines the distributor component spec.
...

```

詳細情報用に、特定のフィールドを追加できます。

```
$ oc explain lokistack.spec.template.compactor
```

出力例

```

KIND:   LokiStack
VERSION: loki.grafana.com/v1

RESOURCE: compactor <Object>

```

DESCRIPTION:

Compactor defines the compaction component spec.

FIELDS:

nodeSelector <map[string]string>

NodeSelector defines the labels required by a node to schedule the component onto it.

...

14.1.3. リソースの設定とロギングコレクターのスケジュール設定

管理者は、サポートされている **ClusterLogForwarder** CR と同じ namespace 内に、同じ名前の **ClusterLogging** カスタムリソース (CR) を作成することで、コレクターのリソースまたはスケジュールを変更できます。

デプロイメントで複数のログフォワーダーを使用する場合に **ClusterLogging** CR に適用できるスタanzas は、**managementState** と **collection** です。他のスタanzas はすべて無視されます。

前提条件

- 管理者権限がある。
- Red Hat OpenShift Logging Operator バージョン 5.8 以降がインストールされている。
- **ClusterLogForwarder** CR が作成されている。

手順

1. 既存の **ClusterLogForwarder** CR をサポートする **ClusterLogging** CR を作成します。

ClusterLogging CR YAML の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: <name> ①
  namespace: <namespace> ②
spec:
  managementState: "Managed"
  collection:
    type: "vector"
  tolerations:
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 1Gi
  nodeSelector:
    collector: needed
# ...

```

■

- ① この名前は、**ClusterLogForwarder** CR と同じ名前である必要があります。
- ② namespace は、**ClusterLogForwarder** CR と同じnamespace である必要があります。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

14.1.4. ロギングコレクター Pod の表示

ロギングコレクター Pod と、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。

手順

- プロジェクトで次のコマンドを実行して、ロギングコレクター Pod とその詳細を表示します。

```
$ oc get pods --selector component=collector -o wide -n <project_name>
```

出力例

```

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

```

14.1.5. 関連情報

- [ノードセレクターの使用による特定ノードへの Pod の配置](#)

14.2. テイントと容認を使用したロギング POD の配置制御

テイントおよび容認 (Toleration) により、ノードはノード上でスケジュールする必要のある (またはスケジュールすべきでない) Pod を制御できます。

14.2.1. テイントおよび容認 (Toleration) について

テイントにより、ノードは Pod に一致する **容認** がない場合に Pod のスケジュールを拒否することができます。

テイントは **Node** 仕様 (**NodeSpec**) でノードに適用され、容認は **Pod** 仕様 (**PodSpec**) で Pod に適用されます。テイントをノードに適用する場合、スケジューラーは Pod がテイントを容認しない限り、Pod をそのノードに配置することができません。

ノード仕様のテイントの例

```

apiVersion: v1
kind: Node
metadata:
  name: my-node
#...
spec:
  taints:
  - effect: NoExecute
    key: key1
    value: value1
#...

```

Pod 仕様での容認の例

```

apiVersion: v1
kind: Pod
metadata:
  name: my-pod
#...
spec:
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoExecute"
    tolerationSeconds: 3600
#...

```

テイントおよび容認は、key、value、および effect で構成されます。

表14.1 テイントおよび容認コンポーネント

パラメーター	説明
key	key には、253 文字までの文字列を使用できます。キー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。
value	value には、63 文字までの文字列を使用できます。値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。

パラメーター	説明						
effect	<p>effect は以下のいずれかにすることができます。</p> <table border="1"> <tr> <td>NoSchedule ^[1]</td> <td> <ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされません。 • ノードの既存 Pod はそのままになります。 </td> </tr> <tr> <td>PreferNoSchedule</td> <td> <ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされる可能性があります、スケジューラーはスケジュールしないようにします。 • ノードの既存 Pod はそのままになります。 </td> </tr> <tr> <td>NoExecute</td> <td> <ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールできません。 • 一致する容認を持たないノードの既存 Pod は削除されます。 </td> </tr> </table>	NoSchedule ^[1]	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされません。 • ノードの既存 Pod はそのままになります。 	PreferNoSchedule	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされる可能性があります、スケジューラーはスケジュールしないようにします。 • ノードの既存 Pod はそのままになります。 	NoExecute	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールできません。 • 一致する容認を持たないノードの既存 Pod は削除されます。
NoSchedule ^[1]	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされません。 • ノードの既存 Pod はそのままになります。 						
PreferNoSchedule	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールされる可能性があります、スケジューラーはスケジュールしないようにします。 • ノードの既存 Pod はそのままになります。 						
NoExecute	<ul style="list-style-type: none"> • テイントに一致しない新規 Pod はノードにスケジュールできません。 • 一致する容認を持たないノードの既存 Pod は削除されます。 						
operator	<table border="1"> <tr> <td>Equal</td> <td>key/value/effect パラメーターは一致する必要があります。これはデフォルトになります。</td> </tr> <tr> <td>Exists</td> <td>key/effect パラメーターは一致する必要があります。いずれかに一致する value パラメーターを空のままにする必要があります。</td> </tr> </table>	Equal	key/value/effect パラメーターは一致する必要があります。これはデフォルトになります。	Exists	key/effect パラメーターは一致する必要があります。いずれかに一致する value パラメーターを空のままにする必要があります。		
Equal	key/value/effect パラメーターは一致する必要があります。これはデフォルトになります。						
Exists	key/effect パラメーターは一致する必要があります。いずれかに一致する value パラメーターを空のままにする必要があります。						

1. **NoSchedule** テイントをコントロールプレーンノードに追加する場合、ノードには、デフォルトで追加される **node-role.kubernetes.io/master::NoSchedule** テイントが必要です。以下に例を示します。

```

apiVersion: v1
kind: Node
metadata:
  annotations:
    machine.openshift.io/machine: openshift-machine-api/ci-ln-62s7gtb-f76d1-v8jxv-master-0
    machineconfiguration.openshift.io/currentConfig: rendered-master-cdc1ab7da414629332cc4c3926e6e59c
    name: my-node
#...

```

```
spec:
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
  #...
```

容認はテイントと一致します。

- **operator** パラメーターが **Equal** に設定されている場合:
 - **key** パラメーターは同じになります。
 - **value** パラメーターは同じになります。
 - **effect** パラメーターは同じになります。
- **operator** パラメーターが **Exists** に設定されている場合:
 - **key** パラメーターは同じになります。
 - **effect** パラメーターは同じになります。

次のテイントは、OpenShift Dedicated に組み込まれています。

- **node.kubernetes.io/not-ready**: ノードは準備状態にありません。これはノード条件 **Ready=False** に対応します。
- **node.kubernetes.io/unreachable**: ノードはノードコントローラーから到達不能です。これはノード条件 **Ready=Unknown** に対応します。
- **node.kubernetes.io/memory-pressure**: ノードにはメモリー不足の問題が発生しています。これはノード条件 **MemoryPressure=True** に対応します。
- **node.kubernetes.io/disk-pressure**: ノードにはディスク不足の問題が発生しています。これはノード条件 **DiskPressure=True** に対応します。
- **node.kubernetes.io/network-unavailable**: ノードのネットワークは使用できません。
- **node.kubernetes.io/unschedulable**: ノードはスケジュールが行えません。
- **node.cloudprovider.kubernetes.io/uninitialized**: ノードコントローラーが外部のクラウドプロバイダーを使用して起動すると、このテイントはノード上に設定され、使用不可能とマークされます。cloud-controller-manager のコントローラーがこのノードを初期化した後に、kubelet がこのテイントを削除します。
- **node.kubernetes.io/pid-pressure**: ノードが pid 不足の状態です。これはノード条件 **PIDPressure=True** に対応します。



重要

OpenShift Dedicated では、デフォルトの pid.available **evictionHard** は設定されません。

14.2.2. Loki Pod の配置

Pod の容認またはノードセクターを使用して、Loki Pod が実行するノードを制御し、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できます。

LokiStack 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、ノード仕様を使用してテイントをノードに適用できます。ノードのテイントは、テイントを容認しないすべての Pod を拒否するようノードに指示する **key:value** ペアです。他の Pod にはない特定の **key:value** ペアを使用すると、ログストア Pod のみがそのノードで実行できるようになります。

ノードセクターを使用する LokiStack の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    compactor: ❶
      nodeSelector:
        node-role.kubernetes.io/infra: "" ❷
    distributor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    gateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    indexGateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ingester:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    querier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    queryFrontend:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ruler:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
# ...
```

❶ ノードセクターに適用されるコンポーネント Pod タイプを指定します。

❷ 定義されたラベルが含まれるノードに移動する Pod を指定します。

前述の設定例では、すべての Loki Pod が **node-role.kubernetes.io/infra: ""** ラベルを含むノードに移動されます。

ノードセクターと容認を使用する LokiStack CR の例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
```

```
namespace: openshift-logging
spec:
# ...
template:
  compactor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  distributor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  indexGateway:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  ingester:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  querier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
```

```

    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  queryFrontend:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  ruler:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
  gateway:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
# ...

```

LokiStack (CR) の **nodeSelector** フィールドと **tolerations** フィールドを設定するには、**oc explain** コマンドを使用して、特定のリソースの説明とフィールドを表示します。

```
$ oc explain lokistack.spec.template
```

出力例

```

KIND:   LokiStack
VERSION: loki.grafana.com/v1

RESOURCE: template <Object>

DESCRIPTION:
  Template defines the resource/limits/tolerations/nodeselectors per
  component

```

```

FIELDS:
  compactor <Object>
    Compactor defines the compaction component spec.

  distributor <Object>
    Distributor defines the distributor component spec.

  ...

```

詳細情報用に、特定のフィールドを追加できます。

```
$ oc explain lokistack.spec.template.compactor
```

出力例

```

KIND:   LokiStack
VERSION: loki.grafana.com/v1

RESOURCE: compactor <Object>

DESCRIPTION:
  Compactor defines the compaction component spec.

FIELDS:
  nodeSelector <map[string]string>
    NodeSelector defines the labels required by a node to schedule the
    component onto it.

  ...

```

14.2.3. 容認を使用したログコレクター Pod 配置の制御

デフォルトで、ログコレクター Pod には以下の **tolerations** 設定があります。

```

apiVersion: v1
kind: Pod
metadata:
  name: collector-example
  namespace: openshift-logging
spec:
# ...
  collection:
    type: vector
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
        operator: Exists
      - effect: NoSchedule
        key: node.kubernetes.io/disk-pressure
        operator: Exists
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists

```

```

- effect: NoSchedule
  key: node.kubernetes.io/memory-pressure
  operator: Exists
- effect: NoSchedule
  key: node.kubernetes.io/pid-pressure
  operator: Exists
- effect: NoSchedule
  key: node.kubernetes.io/unschedulable
  operator: Exists
# ...

```

前提条件

- Red Hat OpenShift Logging Operator および OpenShift CLI (**oc**) がインストールされている。

手順

1. 次のコマンドを実行して、ロギングコレクター 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>
```

コマンドの例

```
$ 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。

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
collection:
  type: vector
  tolerations:
    - key: collector 1
      operator: Exists 2
      effect: NoExecute 3
      tolerationSeconds: 6000 4
  resources:
    limits:
      memory: 2Gi
    requests:
      cpu: 100m
      memory: 1Gi
# ...

```

- 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。

14.2.4. リソースの設定とロギングコレクターのスケジュール設定

管理者は、サポートされている **ClusterLogForwarder** CR と同じ namespace 内に、同じ名前の **ClusterLogging** カスタムリソース (CR) を作成することで、コレクターのリソースまたはスケジュールを変更できます。

デプロイメントで複数のログフォワーダーを使用する場合に **ClusterLogging** CR に適用できるスタanzas は、**managementState** と **collection** です。他のスタanzas はすべて無視されます。

前提条件

- 管理者権限がある。
- Red Hat OpenShift Logging Operator バージョン 5.8 以降がインストールされている。
- **ClusterLogForwarder** CR が作成されている。

手順

1. 既存の **ClusterLogForwarder** CR をサポートする **ClusterLogging** CR を作成します。

ClusterLogging CR YAML の例

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: <name> 1
  namespace: <namespace> 2
spec:
  managementState: "Managed"
  collection:
    type: "vector"
    tolerations:
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 100m

```

```
memory: 1Gi
nodeSelector:
  collector: needed
# ...
```

- 1 この名前は、**ClusterLogForwarder** CR と同じ名前である必要があります。
- 2 namespace は、**ClusterLogForwarder** CR と同じnamespace である必要があります。

2. 次のコマンドを実行して、**ClusterLogging** CR を適用します。

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

14.2.5. ロギングコレクター Pod の表示

ロギングコレクター Pod と、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。

手順

- プロジェクトで次のコマンドを実行して、ロギングコレクター Pod とその詳細を表示します。

```
$ oc get pods --selector component=collector -o wide -n <project_name>
```

出力例

```
NAME          READY STATUS  RESTARTS  AGE  IP           NODE
NOMINATED NODE READINESS GATES
collector-8d69v 1/1   Running  0         134m  10.130.2.30  master1.example.com
<none>         <none>
collector-bd225 1/1   Running  0         134m  10.131.1.11  master2.example.com
<none>         <none>
collector-cvrzs 1/1   Running  0         134m  10.130.0.21  master3.example.com <none>
<none>
collector-gpqq2 1/1   Running  0         134m  10.128.2.27  worker1.example.com
<none>         <none>
collector-l9j7j 1/1   Running  0         134m  10.129.2.31  worker2.example.com <none>
<none>
```

14.2.6. 関連情報

- [ノードテイントを使用した Pod 配置の制御](#)

第15章 ロギングのアンインストール

インストールされている Operator と関連するカスタムリソース (CR) を削除して、OpenShift Dedicated クラスタからロギングを削除できます。

15.1. ロギングのアンインストール

Red Hat OpenShift Logging Operator と **ClusterLogging** カスタムリソース (CR) を削除することで、ログの集約を停止できます。

前提条件

- 管理者権限がある。
- OpenShift Dedicated Web コンソールの **Administrator** パースペクティブにアクセスできる。

手順

1. **Administration** → **Custom Resource Definitions** ページに移動し、**ClusterLogging** をクリックします。
2. **Custom Resource Definition Details** ページで、**Instances** をクリックします。
3. インスタンスの横にあるオプションメニュー  をクリックし、**Delete ClusterLogging** をクリックします。
4. **Administration** → **Custom Resource Definitions** ページに移動します。
5. **ClusterLogging** の横にあるオプションメニュー  をクリックし、**Delete Custom Resource Definition** を選択します。



警告

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削除するには、さらに操作を実行する必要があります。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。

6. **ClusterLogForwarder** CR を作成した場合は、**ClusterLogForwarder** の横にあるオプションメニュー  をクリックし、**Delete Custom Resource Definition** をクリックします。
7. **Operators** → **Installed Operators** ページに移動します。

8. Red Hat OpenShift Logging Operator の横にあるオプションメニュー  をクリックし、**Uninstall Operator** をクリックします。
9. オプション: **openshift-logging** プロジェクトを削除します。



警告

openshift-logging プロジェクトを削除すると、Persistent Volume Claim (PVC) を含む、その namespace 内にあるものがすべて削除されます。ロギングデータを保存する場合は、**openshift-logging** プロジェクトを削除しないでください。

- a. **Home** → **Projects** ページに移動します。
- b. **openshift-logging** プロジェクトの横にあるオプションメニュー  をクリックし、**Delete Project** をクリックします。
- c. ダイアログボックスに **openshift-logging** と入力して削除を確認し、**Delete** をクリックします。

15.2. ロギング PVC の削除

他の Pod で再利用できるように永続ボリューム要求 (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。PVC を保持する必要がない場合は、削除できます。ストレージ領域を回復する必要がある場合は、永続ボリューム (PV) を削除することもできます。

前提条件

- 管理者権限がある。
- OpenShift Dedicated Web コンソールの **Administrator** パースペクティブにアクセスできる。

手順

1. **Storage** → **Persistent Volume Claims** ページに移動します。
2. 各 PVC の横にあるオプションメニュー  をクリックし、**Delete Persistent Volume Claim** を選択します。

15.3. LOKI のアンインストール

前提条件

- 管理者権限がある。

- OpenShift Dedicated Web コンソールの **Administrator** パースペクティブにアクセスできる。
- Red Hat OpenShift Logging Operator と関連リソースをまだ削除していない場合は、**ClusterLogging** カスタムリソースから LokiStack への参照の削除が完了している。

手順

1. **Administration** → **Custom Resource Definitions** ページに移動し、**LokiStack** をクリックします。
2. **Custom Resource Definition Details** ページで、**Instances** をクリックします。
3. インスタンスの横にあるオプションメニュー  をクリックし、**Delete LokiStack** をクリックします。
4. **Administration** → **Custom Resource Definitions** ページに移動します。
5. **LokiStack** の横にあるオプションメニュー  をクリックし、**Delete Custom Resource Definition** を選択します。
6. オブジェクトストレージシークレットを削除します。
7. **Operators** → **Installed Operators** ページに移動します。
8. **Loki Operator** の横にあるオプションメニュー  をクリックし、**Uninstall Operator** をクリックします。
9. オプション: **openshift-operators-redhat** プロジェクトを削除します。



重要

他のグローバル Operator が **openshift-operators-redhat** namespace にインストールされている場合は、**openshift-operators-redhat** プロジェクトを削除しないでください。

- a. **Home** → **Projects** ページに移動します。
- b. **openshift-operators-redhat** プロジェクトの横にあるオプションメニュー  をクリックし、**Delete Project** をクリックします。
- c. ダイアログボックスに **openshift-operators-redhat** と入力して削除を確認し、**Delete** をクリックします。

15.4. ELASTICSEARCH のアンインストール

前提条件

- 管理者権限がある。

- OpenShift Dedicated Web コンソールの **Administrator** パースペクティブにアクセスできる。
- Red Hat OpenShift Logging Operator と関連リソースをまだ削除していない場合は、**ClusterLogging** カスタムリソースから Elasticsearch への参照の削除が完了している。

手順

1. **Administration** → **Custom Resource Definitions** ページに移動し、**Elasticsearch** をクリックします。
2. **Custom Resource Definition Details** ページで、**Instances** をクリックします。
3. インスタンスの横にあるオプションメニュー  をクリックし、**Delete Elasticsearch** をクリックします。
4. **Administration** → **Custom Resource Definitions** ページに移動します。
5. **Elasticsearch** の横にあるオプションメニュー  をクリックし、**Delete Custom Resource Definition** を選択します。
6. オブジェクトストレージシークレットを削除します。
7. **Operators** → **Installed Operators** ページに移動します。
8. OpenShift Elasticsearch Operator の横にあるオプションメニュー  をクリックし、**Uninstall Operator** をクリックします。
9. オプション: **openshift-operators-redhat** プロジェクトを削除します。



重要

他のグローバル Operator が **openshift-operators-redhat** namespace にインストールされている場合は、**openshift-operators-redhat** プロジェクトを削除しないでください。

- a. **Home** → **Projects** ページに移動します。
- b. **openshift-operators-redhat** プロジェクトの横にあるオプションメニュー  をクリックし、**Delete Project** をクリックします。
- c. ダイアログボックスに **openshift-operators-redhat** と入力して削除を確認し、**Delete** をクリックします。

15.5. CLI の使用によるクラスターからの OPERATOR の削除

クラスター管理者は CLI を使用して、選択した namespace からインストールされた Operator を削除できます。

前提条件

- **dedicated-admin** パーミッションを持つアカウントを使用して OpenShift Dedicated クラスターにアクセスできる。
- OpenShift CLI (**oc**) がワークステーションにインストールされている。

手順

1. サブスクリプションした Operator の最新バージョン (**serverless-operator** など) が、**currentCSV** フィールドで識別されていることを確認します。

```
$ oc get subscription.operators.coreos.com serverless-operator -n openshift-serverless -o yaml | grep currentCSV
```

出力例

```
currentCSV: serverless-operator.v1.28.0
```

2. サブスクリプション (**serverless-operator** など) を削除します。

```
$ oc delete subscription.operators.coreos.com serverless-operator -n openshift-serverless
```

出力例

```
subscription.operators.coreos.com "serverless-operator" deleted
```

3. 直前の手順で **currentCSV** 値を使用し、ターゲット namespace の Operator の CSV を削除します。

```
$ oc delete clusterserviceversion serverless-operator.v1.28.0 -n openshift-serverless
```

出力例

```
clusterserviceversion.operators.coreos.com "serverless-operator.v1.28.0" deleted
```

関連情報

- [永続ボリュームの手動回収](#)

第16章 ログレコードのフィールド

ロギングによってエクスポートされたログレコードには、以下のフィールドが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。

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

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]

@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

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
---------	-------

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
---------	----

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
---------	----

LEVEL

rsyslog (**severitytext** プロパティ)、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リスト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

PID

ロギングエンティティのプロセス ID です (ある場合)。

データのタイプ	キーワード
---------	-------

サービス

ロギングエンティティに関連付けられたサービスの名前です (ある場合)。たとえば、syslog の **APP-NAME** および rsyslog の **programname** プロパティ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17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 定義のタグのリストです。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
---------	------

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
---------	------

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18章 KUBERNETES

Kubernetes 固有メタデータの namespace です。

データのタイプ	group
---------	-------

18.1. KUBERNETES.POD_NAME

Pod の名前。

データのタイプ	キーワード
---------	-------

18.2. KUBERNETES.POD_ID

Pod の Kubernetes ID。

データのタイプ	キーワード
---------	-------

18.3. KUBERNETES.NAMESPACE_NAME

Kubernetes の namespace の名前。

データのタイプ	キーワード
---------	-------

18.4. KUBERNETES.NAMESPACE_ID

Kubernetes の namespace ID。

データのタイプ	キーワード
---------	-------

18.5. KUBERNETES.HOST

Kubernetes ノード名。

データのタイプ	キーワード
---------	-------

18.6. KUBERNETES.CONTAINER_NAME

Kubernetes のコンテナの名前。

データのタイプ	キーワード
---------	-------

18.7. 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
---------	-------

18.8. KUBERNETES.LABELS

元の Kubernetes Pod にあるラベル

データのタイプ	group
---------	-------

18.9. 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 core](#) の **type Event** に準拠します。

データのタイプ	group
---------	-------

18.9.1. 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

18.9.2. 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
---------	-------

18.9.2.1. 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

18.9.2.2. 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

18.9.2.3. 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

18.9.2.4. kubernetes.event.metadata.uid

イベントの一意的 ID

データのタイプ	キーワード
値の例	d828ac69-7b58-11e7-9cf5-5254002f560c

18.9.2.5. 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

18.9.3. 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
---------	-------

18.9.3.1. 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

18.9.3.2. 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

18.9.3.3. 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

18.9.3.4. kubernetes.event.involvedObject.uid

オブジェクトの一意的 ID

データのタイプ	キーワード
値の例	e6bff941-76a8-11e7-8193-5254002f560c

18.9.3.5. 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

18.9.3.6. 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

18.9.4. 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

18.9.5. 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

18.9.6. 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

18.9.7. 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

18.9.8. 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19章 OPENSIFT

openshift-logging 固有のメタデータの namespace

データのタイプ	group
---------	-------

19.1. OPENSIFT.LABELS

クラスターログフォワード設定によって追加されるラベル

データのタイプ	group
---------	-------

第20章 API リファレンス

20.1. 5.6 LOGGING API リファレンス

20.1.1. Logging 5.6 API リファレンス

20.1.1.1. ClusterLogForwarder

ClusterLogForwarder は、転送ログを設定するための API です。

名前付き入力のセットから名前付き出力のセットに転送する **pipelines** のリストを指定して、転送を設定します。

一般的なログカテゴリーには組み込みの入力名があり、カスタム入力を定義して、追加のフィルタリングを行うことができます。

デフォルトの OpenShift ログストアには組み込みの出力名がありますが、URL やその他の接続情報を使用して、独自の出力を定義し、クラスターの内部または外部の他のストアまたはプロセッサにログを転送できます。

詳細については、API フィールドに関するドキュメントを参照してください。

プロパティ	型	説明
spec	object	ClusterLogForwarder の期待される動作の仕様
status	object	ClusterLogForwarder のステータス

20.1.1.1.1. .spec

20.1.1.1.1.1. 説明

ClusterLogForwarderSpec は、ログをリモートターゲットに転送する方法を定義します。

20.1.1.1.1.1.1. 型

- object

プロパティ	型	説明
inputs	array	(オプション) 入力、転送されるログメッセージの名前付きフィルターです。

プロパティ	型	説明
outputDefaults	object	(オプション) DEPRECATED OutputDefaults は、デフォルトストアのフォワーダー設定を明示的に指定します。
outputs	array	(オプション) 出力は、ログメッセージの名前付きの宛先です。
pipelines	array	pipelines は、一連の入力によって選択されたメッセージを一連の出力に転送します。

20.1.1.1.2. .spec.inputs[]

20.1.1.1.2.1. 説明

InputSpec は、ログメッセージのセレクターを定義します。

20.1.1.1.2.1.1. 型

- array

プロパティ	型	説明
application	object	(オプション) アプリケーション (存在する場合は、 application ログの名前付きセットを有効にします。
name	string	pipeline の入力を参照するために使用される名前。

20.1.1.1.3. .spec.inputs[].application

20.1.1.1.3.1. 説明

アプリケーションログセレクター。ログを選択するには、セレクターのすべての条件が満たされる (論理 AND) 必要があります。

20.1.1.1.3.1.1. 型

- object

プロパティ	型	説明
-------	---	----

プロパティ	型	説明
namespace	array	(オプション) アプリケーションログを収集する namespace。
selector	object	(オプション) ラベルが一致する Pod からのログのセレクター。

20.1.1.1.4. .spec.inputs[].application.namespaces[]

20.1.1.1.4.1. 説明

20.1.1.1.4.1.1. 型

- array

20.1.1.1.5. .spec.inputs[].application.selector

20.1.1.1.5.1. 説明

ラベルセレクターとは、一連のリソースに対するラベルクエリー機能です。

20.1.1.1.5.1.1. 型

- object

プロパティ	型	説明
matchLabels	object	(オプション) matchLabels は {key,value} ペアのマップです。matchLabels の単一の {key,value}

20.1.1.1.6. .spec.inputs[].application.selector.matchLabels

20.1.1.1.6.1. 説明

20.1.1.1.6.1.1. 型

- object

20.1.1.1.7. .spec.outputDefaults

20.1.1.1.7.1. 説明

20.1.1.1.7.1.1. 型

- object

プロパティ	型	説明
elasticsearch	object	(オプション) Elasticsearch OutputSpec のデフォルト値

20.1.1.1.8. .spec.outputDefaults.elasticsearch

20.1.1.1.8.1. 説明

ElasticsearchStructuredSpec は、elasticsearch インデックスを決定するための構造化ログの変更に關連する仕様です。

20.1.1.1.8.1.1. 型

- object

プロパティ	型	説明
enableStructuredContainerLogs	bool	(オプション) EnableStructuredContainerLogs は、複数コンテナの構造化ログを許可します。
structuredTypeKey	string	(オプション) StructuredTypeKey は、elasticsearch インデックスの名前として使用されるメタデータキーを指定します。
structuredTypeName	string	(オプション) StructuredTypeName は、elasticsearch スキーマの名前を指定します。

20.1.1.1.9. .spec.outputs[]

20.1.1.1.9.1. 説明

出力は、ログメッセージの宛先を定義します。

20.1.1.1.9.1.1. 型

- array

プロパティ	型	説明
syslog	object	(オプション)
fluentdForward	object	(オプション)

プロパティ	型	説明
elasticsearch	object	(オプション)
kafka	object	(オプション)
cloudwatch	object	(オプション)
loki	object	(オプション)
googleCloudLogging	object	(オプション)
splunk	object	(オプション)
name	string	pipeline からの出力を参照するために使用される名前。
secret	object	(オプション) 認証のシークレット。
tls	object	TLS には、TLS クライアント接続のオプションを制御するための設定が含まれています。
type	string	出力プラグインのタイプ。
url	string	(オプション) ログレコードの送信先 URL。

20.1.1.1.10. .spec.outputs[].secret

20.1.1.1.10.1. 説明

OutputSecretSpec は、名前のみを含み、namespace を含まないシークレット参照です。

20.1.1.1.10.1.1. 型

- object

プロパティ	型	説明
name	string	ログフォワダーシークレット用に設定された namespace 内のシークレットの名前。

20.1.1.1.11. .spec.outputs[].tls

20.1.1.11.1. 説明

OutputTLSSpec には、出力タイプに依存しない TLS 接続のオプションが含まれています。

20.1.1.11.1.1. 型

- object

プロパティ	型	説明
insecureSkipVerify	bool	InsecureSkipVerify が true の場合、TLS クライアントは証明書のエラーを無視するように設定されます。

20.1.1.12. .spec.pipelines[]

20.1.1.12.1. 説明

PipelinesSpec は、一連の入力を一連の出力にリンクします。

20.1.1.12.1.1. 型

- array

プロパティ	型	説明
detectMultilineErrors	bool	(オプション) DetectMultilineErrors は、コンテナログの複数行エラー検出を有効にします。
inputRefs	array	InputRefs は、このパイプラインへの入力の名前 (input.name) をリストします。
labels	object	(オプション) このパイプラインを通過するログレコードに適用されるラベル。
name	string	(オプション) 名前は省略可能ですが、指定する場合は、 pipelines リスト内で一意である必要があります。
outputRefs	array	OutputRefs は、このパイプラインからの出力の名前 (output.name) を一覧表示します。

プロパティ	型	説明
parse	string	(オプション) 解析により、ログエントリーを構造化ログに解析できます。

20.1.1.1.13. .spec.pipelines[].inputRefs[]

20.1.1.1.13.1. 説明

20.1.1.1.13.1.1. 型

- array

20.1.1.1.14. .spec.pipelines[].labels

20.1.1.1.14.1. 説明

20.1.1.1.14.1.1. 型

- object

20.1.1.1.15. .spec.pipelines[].outputRefs[]

20.1.1.1.15.1. 説明

20.1.1.1.15.1.1. 型

- array

20.1.1.1.16. .status

20.1.1.1.16.1. 説明

ClusterLogForwarderStatus は、ClusterLogForwarder の監視状態を定義します。

20.1.1.1.16.1.1. 型

- object

プロパティ	型	説明
conditions	object	ログフォワーダーの条件。
inputs	Conditions	入力は、入力名を入力の条件にマッピングします。

プロパティ	型	説明
outputs	Conditions	出力は、出力名を出力の条件にマッピングします。
pipelines	Conditions	pipelines は、パイプライン名をパイプラインの条件にマッピングします。

20.1.1.17. .status.conditions

20.1.1.17.1. 説明

20.1.1.17.1.1. 型

- object

20.1.1.18. .status.inputs

20.1.1.18.1. 説明

20.1.1.18.1.1. 型

- Conditions

20.1.1.19. .status.outputs

20.1.1.19.1. 説明

20.1.1.19.1.1. 型

- Conditions

20.1.1.20. .status.pipelines

20.1.1.20.1. 説明

20.1.1.20.1.1. 型

- Conditions== ClusterLogging A Red Hat OpenShift Logging インスタンス。ClusterLogging は、clusterloggings API のスキーマです。

プロパティ	型	説明
spec	object	ClusterLogging の期待される動作の仕様

プロパティ	型	説明
status	object	Status は、ClusterLogging の監視状態を定義します。

20.1.1.1.21. .spec

20.1.1.1.21.1. 説明

ClusterLoggingSpec は ClusterLogging の期待される状態を定義します。

20.1.1.1.21.1.1. 型

- object

プロパティ	型	説明
コレクション	object	クラスターの Collection コンポーネントの仕様
キュレーション	object	(非推奨)(オプション) 非推奨。クラスターの Curation コンポーネントの仕様
フォワード	object	(非推奨)(オプション) 非推奨。クラスターの Forwarder コンポーネントの仕様
logStore	object	(オプション) クラスターの Log Storage コンポーネントの仕様
managementState	string	(オプション) リソースが Operator によって管理されているか管理されていないかを示す指標
可視化	object	(オプション) クラスターの Visualization コンポーネントの仕様

20.1.1.1.22. .spec.collection

20.1.1.1.22.1. 説明

これは、ログおよびイベントコレクションに関連する情報を含む構造体です。

20.1.1.1.22.1.1. 型

- object

プロパティ	型	説明
resources	object	(オプション) コレクターのリソース要件
nodeSelector	object	(オプション) Pod がスケジュールされるノードを定義します。
tolerations	array	(オプション) Pod が受け入れる Tolerations を定義します。
fluentd	object	(オプション) Fluentd は、fluentd タイプのフォワーダーの設定を表します。
logs	object	(非推奨)(オプション) 非推奨。クラスタのログ収集の仕様
type	string	(オプション) 設定するログ収集のタイプ

20.1.1.1.23. .spec.collection.fluentd

20.1.1.1.23.1. 説明

FluentdForwarderSpec は、fluentd タイプのフォワーダーの設定を表します。

20.1.1.1.23.1.1. 型

- object

プロパティ	型	説明
buffer	object	
inFile	object	

20.1.1.1.24. .spec.collection.fluentd.buffer

20.1.1.1.24.1. 説明

FluentdBufferSpec は、すべての fluentd 出力のバッファ設定をチューニングするための fluentd バッファパラメータのサブセットを表します。パラメータのサブセットをサポートして、バッファとキューのサイズ設定、フラッシュ操作、フラッシュの再試行を設定します。

一般的なパラメータについては、<https://docs.fluentd.org/configuration/buffer-section#buffering-parameters> を参照してください。

フラッシュパラメーターについては、<https://docs.fluentd.org/configuration/buffer-section#flushing-parameters> を参照してください。

再試行パラメーターについては、<https://docs.fluentd.org/configuration/buffer-section#retries-parameters> を参照してください。

20.1.1.1.24.1.1. 型

- object

プロパティ	型	説明
chunkLimitSize	string	(オプション) ChunkLimitSize は、各チャンクの最大サイズを表します。イベントは以下ようになります。
flushInterval	string	(オプション) FlushInterval は、2つの連続するフラッシュの間の待機時間を表します。
flushMode	string	(オプション) FlushMode は、チャンクを書き込むフラッシュスレッドのモードを表します。モード
flushThreadCount	int	(オプション) FlushThreadCount は、fluentd バッファによって使用されるスレッドの数を表します。
overflowAction	string	(オプション) OverflowAction は、fluentd バッファプラグインが実行するアクションを表します。
retryMaxInterval	string	(オプション) RetryMaxInterval は、指数バックオフの最大時間間隔を表します。
retryTimeout	string	(オプション) RetryTimeout は、あきらめる前に再試行を試みる最大時間間隔を表します。
retryType	string	(オプション) RetryType は、再試行するフラッシュ操作のタイプを表します。フラッシュ操作は以下を実行できます。
retryWait	string	(オプション) RetryWait は、2回連続して再試行してフラッシュするまでの時間を表します。

プロパティ	型	説明
totalLimitSize	string	(オプション) TotalLimitSize は、fluentd ごとに許可されるノード領域のしきい値を表します。

20.1.1.1.25. .spec.collection.fluentd.inFile

20.1.1.1.25.1. 説明

FluentdInFileSpec は、すべての fluentd in-tail 入力の設定をチューニングするための fluentd in-tail プラグインパラメーターのサブセットを表します。

一般的なパラメーターについては、<https://docs.fluentd.org/input/tail#parameters> を参照してください。

20.1.1.1.25.1.1. 型

- object

プロパティ	型	説明
readLinesLimit	int	(オプション) ReadLinesLimit は、各 I/O 操作で読み取る行数を表します。

20.1.1.1.26. .spec.collection.logs

20.1.1.1.26.1. 説明

20.1.1.1.26.1.1. 型

- object

プロパティ	型	説明
fluentd	object	Fluentd Log Collection コンポーネントの仕様
type	string	設定するログ収集のタイプ

20.1.1.1.27. .spec.collection.logs.fluentd

20.1.1.1.27.1. 説明

CollectorSpec は、コレクターのスケジュールとリソースを定義するための仕様です。

20.1.1.1.27.1.1. 型

- object

プロパティ	型	説明
nodeSelector	object	(オプション) Pod がスケジュールされるノードを定義します。
resources	object	(オプション) コレクターのリソース要件
tolerations	array	(オプション) Pod が受け入れる Tolerations を定義します。

20.1.1.1.28. .spec.collection.logs.fluentd.nodeSelector

20.1.1.1.28.1. 説明

20.1.1.1.28.1.1. 型

- object

20.1.1.1.29. .spec.collection.logs.fluentd.resources

20.1.1.1.29.1. 説明

20.1.1.1.29.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.30. .spec.collection.logs.fluentd.resources.limits

20.1.1.1.30.1. 説明

20.1.1.1.30.1.1. 型

- object

20.1.1.1.31. `.spec.collection.logs.fluentd.resources.requests`

20.1.1.1.31.1. 説明

20.1.1.1.31.1.1. 型

- object

20.1.1.1.32. `.spec.collection.logs.fluentd.tolerations[]`

20.1.1.1.32.1. 説明

20.1.1.1.32.1.1. 型

- array

プロパティ	型	説明
effect	string	(オプション) Effect は、一致する Taint 効果を示します。空の場合は、すべてのテイント効果に一致します。
鍵 (key)	string	(オプション) Key は、Toleration が適用される Taint キーです。空の場合は、すべてのテイントキーに一致します。
operator	string	(オプション) Operator は、キーと値の関係を表します。
tolerationSeconds	int	(オプション) TolerationSeconds は、Toleration の期間を表します。
値	string	(オプション) Value は、Toleration が一致する Taint 値です。

20.1.1.1.33. `.spec.collection.logs.fluentd.tolerations[].tolerationSeconds`

20.1.1.1.33.1. 説明

20.1.1.1.33.1.1. 型

- int

20.1.1.1.34. `.spec.curation`

20.1.1.1.34.1. 説明

これは、ログのキュレーション (Curator) に関連する情報を含む構造体です。

20.1.1.1.34.1.1. 型

- object

プロパティ	型	説明
curator	object	設定するキュレーションの仕様
type	string	設定するキュレーションの種類

20.1.1.1.35. .spec.curation.curator

20.1.1.1.35.1. 説明

20.1.1.1.35.1.1. 型

- object

プロパティ	型	説明
nodeSelector	object	Pod がスケジュールされているノードを定義します。
resources	object	(オプション) Curator のリソース要件
schedule	string	Curator ジョブが実行される cron スケジュール。デフォルトは「30 3***」です。
tolerations	array	

20.1.1.1.36. .spec.curation.curator.nodeSelector

20.1.1.1.36.1. 説明

20.1.1.1.36.1.1. 型

- object

20.1.1.1.37. .spec.curation.curator.resources

20.1.1.1.37.1. 説明

20.1.1.1.37.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.38. .spec.curation.curator.resources.limits

20.1.1.1.38.1. 説明

20.1.1.1.38.1.1. 型

- object

20.1.1.1.39. .spec.curation.curator.resources.requests

20.1.1.1.39.1. 説明

20.1.1.1.39.1.1. 型

- object

20.1.1.1.40. .spec.curation.curator.tolerations[]

20.1.1.1.40.1. 説明

20.1.1.1.40.1.1. 型

- array

プロパティ	型	説明
effect	string	(オプション) Effect は、一致する Taint 効果を示します。空の場合は、すべてのテイント効果に一致します。

プロパティ	型	説明
鍵 (key)	string	(オプション) Key は、Toleration が適用される Taint キーです。空の場合は、すべてのテイントキーに一致します。
operator	string	(オプション) Operator は、キーと値の関係を表します。
tolerationSeconds	int	(オプション) TolerationSeconds は、Toleration の期間を表します。
値	string	(オプション) Value は、Toleration が一致する Taint 値です。

20.1.1.1.41. .spec.curation.curator.tolerations[].tolerationSeconds

20.1.1.1.41.1. 説明

20.1.1.1.41.1.1. 型

- int

20.1.1.1.42. .spec.forwarder

20.1.1.1.42.1. 説明

ForwarderSpec には、特定のフォワーダー実装のグローバルチューニングパラメーターが含まれています。このフィールドは、一般的な使用には必要ありません。基礎となるフォワーダーテクノロジーに精通しているユーザーがパフォーマンスをチューニングできるようにします。現在サポートされているもの: **fluentd**。

20.1.1.1.42.1.1. 型

- object

プロパティ	型	説明
fluentd	object	

20.1.1.1.43. .spec.forwarder.fluentd

20.1.1.1.43.1. 説明

FluentdForwarderSpec は、fluentd タイプのフォワーダーの設定を表します。

20.1.1.1.43.1.1. 型

- object

プロパティ	型	説明
buffer	object	
inFile	object	

20.1.1.1.44. .spec.forwarder.fluentd.buffer

20.1.1.1.44.1. 説明

FluentdBufferSpec は、すべての fluentd 出力のバッファ設定をチューニングするための fluentd バッファパラメーターのサブセットを表します。パラメーターのサブセットをサポートして、バッファとキューのサイズ設定、フラッシュ操作、フラッシュの再試行を設定します。

一般的なパラメーターについては、<https://docs.fluentd.org/configuration/buffer-section#buffering-parameters> を参照してください。

フラッシュパラメーターについては、<https://docs.fluentd.org/configuration/buffer-section#flushing-parameters> を参照してください。

再試行パラメーターについては、<https://docs.fluentd.org/configuration/buffer-section#retries-parameters> を参照してください。

20.1.1.1.44.1.1. 型

- object

プロパティ	型	説明
chunkLimitSize	string	(オプション) ChunkLimitSize は、各チャンクの最大サイズを表します。イベントは以下ようになります。
flushInterval	string	(オプション) FlushInterval は、2つの連続するフラッシュの間の待機時間を表します。
flushMode	string	(オプション) FlushMode は、チャンクを書き込むフラッシュスレッドのモードを表します。モード
flushThreadCount	int	(オプション) FlushThreadCount は、fluentd バッファによって使用されるスレッドの数を表します。

プロパティ	型	説明
overflowAction	string	(オプション) OverflowAction は、fluentd バッファプラグインが実行するアクションを表します。
retryMaxInterval	string	(オプション) RetryMaxInterval は、指数バックオフの最大時間間隔を表します。
retryTimeout	string	(オプション) RetryTimeout は、あきらめる前に再試行を試みる最大時間間隔を表します。
retryType	string	(オプション) RetryType は、再試行するフラッシュ操作のタイプを表します。フラッシュ操作は以下を実行できます。
retryWait	string	(オプション) RetryWait は、2回連続して再試行してフラッシュするまでの時間を表します。
totalLimitSize	string	(オプション) TotalLimitSize は、fluentd ごとに許可されるノード領域のしきい値を表します。

20.1.1.1.45. .spec.forwarder.fluentd.inFile

20.1.1.1.45.1. 説明

FluentdInFileSpec は、すべての fluentd in-tail 入力の設定をチューニングするための fluentd in-tail プラグインパラメーターのサブセットを表します。

一般的なパラメーターについては、<https://docs.fluentd.org/input/tail#parameters> を参照してください。

20.1.1.1.45.1.1. 型

- object

プロパティ	型	説明
readLinesLimit	int	(オプション) ReadLinesLimit は、各 I/O 操作で読み取る行数を表します。

20.1.1.1.46. .spec.logStore

20.1.1.1.46.1. 説明

LogStoreSpec には、ログの保存方法に関する情報が含まれています。

20.1.1.1.46.1.1. 型

- object

プロパティ	型	説明
elasticsearch	object	Elasticsearch Log Store コンポーネントの仕様
lokistack	object	LokiStack には、Type が LogStoreTypeLokiStack に設定されている場合、ログストレージに使用する LokiStack に関する情報が含まれています。
retentionPolicy	object	(オプション) 保持ポリシーは、インデックスが削除されるまでの最大期間を定義します。
type	string	設定するログストレージのタイプ。現在、Operator は、ElasticSearch を使用して、いずれかをサポートしています。

20.1.1.1.47. .spec.logStore.elasticsearch

20.1.1.1.47.1. 説明

20.1.1.1.47.1.1. 型

- object

プロパティ	型	説明
nodeCount	int	Elasticsearch 用にデプロイするノードの数
nodeSelector	object	Pod がスケジュールされているノードを定義します。
proxy	object	Elasticsearch Proxy コンポーネントの仕様
redundancyPolicy	string	(オプション)

プロパティ	型	説明
resources	object	(オプション) Elasticsearch のリソース要件
storage	object	(オプション) Elasticsearch データノードのストレージ仕様
tolerations	array	

20.1.1.1.48. .spec.logStore.elasticsearch.nodeSelector

20.1.1.1.48.1. 説明

20.1.1.1.48.1.1. 型

- object

20.1.1.1.49. .spec.logStore.elasticsearch.proxy

20.1.1.1.49.1. 説明

20.1.1.1.49.1.1. 型

- object

プロパティ	型	説明
resources	object	

20.1.1.1.50. .spec.logStore.elasticsearch.proxy.resources

20.1.1.1.50.1. 説明

20.1.1.1.50.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。

プロパティ	型	説明
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.51. .spec.logStore.elasticsearch.proxy.resources.limits

20.1.1.1.51.1. 説明

20.1.1.1.51.1.1. 型

- object

20.1.1.1.52. .spec.logStore.elasticsearch.proxy.resources.requests

20.1.1.1.52.1. 説明

20.1.1.1.52.1.1. 型

- object

20.1.1.1.53. .spec.logStore.elasticsearch.resources

20.1.1.1.53.1. 説明

20.1.1.1.53.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.54. .spec.logStore.elasticsearch.resources.limits

20.1.1.1.54.1. 説明

20.1.1.1.54.1.1. 型

- object

20.1.1.1.55. `.spec.logStore.elasticsearch.resources.requests`

20.1.1.1.55.1. 説明

20.1.1.1.55.1.1. 型

- object

20.1.1.1.56. `.spec.logStore.elasticsearch.storage`

20.1.1.1.56.1. 説明

20.1.1.1.56.1.1. 型

- object

プロパティ	型	説明
size	object	ノードがプロビジョニングする最大ストレージ容量。
storageClassName	string	(オプション) ノードの PVC の作成に使用するストレージクラスの名前。

20.1.1.1.57. `.spec.logStore.elasticsearch.storage.size`

20.1.1.1.57.1. 説明

20.1.1.1.57.1.1. 型

- object

プロパティ	型	説明
形式	string	形式を自由に変更します。 Canonicalize のコメントを参照してください。
d	object	d.Dec != nil の場合、d は inf.Dec 形式の数量です。
i	int	d.Dec == nil の場合、i は int64 でスケールされた形式の数量です。
s	string	s は、再計算を避けるために生成されたこの量の値です。

20.1.1.1.58. `.spec.logStore.elasticsearch.storage.size.d`

20.1.1.1.58.1. 説明

20.1.1.1.58.1.1. 型

- object

プロパティ	型	説明
Dec	object	

20.1.1.1.59. `.spec.logStore.elasticsearch.storage.size.d.Dec`

20.1.1.1.59.1. 説明

20.1.1.1.59.1.1. 型

- object

プロパティ	型	説明
scale	int	
unscaled	object	

20.1.1.1.60. `.spec.logStore.elasticsearch.storage.size.d.Dec.unscaled`

20.1.1.1.60.1. 説明

20.1.1.1.60.1.1. 型

- object

プロパティ	型	説明
abs	Word	sign
neg	bool	

20.1.1.1.61. `.spec.logStore.elasticsearch.storage.size.d.Dec.unscaled.abs`

20.1.1.1.61.1. 説明

20.1.1.1.61.1.1. 型

- Word

20.1.1.1.62. .spec.logStore.elasticsearch.storage.size.i

20.1.1.1.62.1. 説明

20.1.1.1.62.1.1. 型

- int

プロパティ	型	説明
scale	int	
値	int	

20.1.1.1.63. .spec.logStore.elasticsearch.tolerations[]

20.1.1.1.63.1. 説明

20.1.1.1.63.1.1. 型

- array

プロパティ	型	説明
effect	string	(オプション) Effect は、一致する Taint 効果を示します。空の場合は、すべてのテイント効果に一致します。
鍵 (key)	string	(オプション) Key は、Toleration が適用される Taint キーです。空の場合は、すべてのテイントキーに一致します。
operator	string	(オプション) Operator は、キーと値の関係を表します。
tolerationSeconds	int	(オプション) TolerationSeconds は、Toleration の期間を表します。
値	string	(オプション) Value は、Toleration が一致する Taint 値です。

20.1.1.1.64. .spec.logStore.elasticsearch.tolerations[].tolerationSeconds

20.1.1.1.64.1. 説明

20.1.1.1.64.1.1. 型

- int

20.1.1.1.65. .spec.logStore.lokistack

20.1.1.1.65.1. 説明

LokiStackStoreSpec は、LokiStack をログストレージとして使用するよう、cluster-logging を設定するために使用されます。これは、同じ namespace 内の既存の LokiStack を指しています。

20.1.1.1.65.1.1. 型

- object

プロパティ	型	説明
name	string	LokiStack リソースの名前。

20.1.1.1.66. .spec.logStore.retentionPolicy

20.1.1.1.66.1. 説明

20.1.1.1.66.1.1. 型

- object

プロパティ	型	説明
application	object	
audit	object	
infra	object	

20.1.1.1.67. .spec.logStore.retentionPolicy.application

20.1.1.1.67.1. 説明

20.1.1.1.67.1.1. 型

- object

プロパティ	型	説明
diskThresholdPercent	int	(オプション) ES ディスク使用率のしきい値に達した場合、古いインデックスを削除する必要があります (例: 75)。
maxAge	string	(オプション)
namespaceSpec	array	(オプション) 指定された最小期間よりも古いドキュメントを削除する namespace ごとの仕様
pruneNamespacesInterval	string	(オプション) 新しい prune-namespaces ジョブを実行する頻度

20.1.1.1.68. .spec.logStore.retentionPolicy.application.namespaceSpec[]

20.1.1.1.68.1. 説明

20.1.1.1.68.1.1. 型

- array

プロパティ	型	説明
minAge	string	(オプション) この MinAge よりも古い namespace に一致するレコードを削除します (例: 1d)。
namespace	string	MinAge より古いログを削除するターゲット namespace (デフォルトは 7d)

20.1.1.1.69. .spec.logStore.retentionPolicy.audit

20.1.1.1.69.1. 説明

20.1.1.1.69.1.1. 型

- object

プロパティ	型	説明
-------	---	----

プロパティ	型	説明
diskThresholdPercent	int	(オプション) ES ディスク使用率のしきい値に達した場合、古いインデックスを削除する必要があります (例: 75)。
maxAge	string	(オプション)
namespaceSpec	array	(オプション) 指定された最小期間よりも古いドキュメントを削除する namespace ごとの仕様
pruneNamespacesInterval	string	(オプション) 新しい prune-namespaces ジョブを実行する頻度

20.1.1.1.70. .spec.logStore.retentionPolicy.audit.namespaceSpec[]

20.1.1.1.70.1. 説明

20.1.1.1.70.1.1. 型

- array

プロパティ	型	説明
minAge	string	(オプション) この MinAge よりも古い namespace に一致するレコードを削除します (例: 1d)。
namespace	string	MinAge より古いログを削除するターゲット namespace (デフォルトは 7d)

20.1.1.1.71. .spec.logStore.retentionPolicy.infra

20.1.1.1.71.1. 説明

20.1.1.1.71.1.1. 型

- object

プロパティ	型	説明
-------	---	----

プロパティ	型	説明
diskThresholdPercent	int	(オプション) ES ディスク使用率のしきい値に達した場合、古いインデックスを削除する必要があります (例: 75)。
maxAge	string	(オプション)
namespaceSpec	array	(オプション) 指定された最小期間よりも古いドキュメントを削除する namespace ごとの仕様
pruneNamespacesInterval	string	(オプション) 新しい prune-namespaces ジョブを実行する頻度

20.1.1.1.72. .spec.logStore.retentionPolicy.infra.namespaceSpec[]

20.1.1.1.72.1. 説明

20.1.1.1.72.1.1. 型

- array

プロパティ	型	説明
minAge	string	(オプション) この MinAge よりも古い namespace に一致するレコードを削除します (例: 1d)。
namespace	string	MinAge より古いログを削除するターゲット namespace (デフォルトは 7d)

20.1.1.1.73. .spec.visualization

20.1.1.1.73.1. 説明

これは、ログの視覚化 (Kibana) に関連する情報を含む構造体です。

20.1.1.1.73.1.1. 型

- object

プロパティ	型	説明
-------	---	----

プロパティ	型	説明
kibana	object	Kibana Visualization コンポーネントの仕様
type	string	設定する可視化のタイプ

20.1.1.1.74. .spec.visualization.kibana

20.1.1.1.74.1. 説明

20.1.1.1.74.1.1. 型

- object

プロパティ	型	説明
nodeSelector	object	Pod がスケジュールされているノードを定義します。
proxy	object	Kibana Proxy コンポーネントの仕様
replicas	int	Kibana デプロイメント用にデプロイするインスタンスの数
resources	object	(オプション) Kibana のリソース要件
tolerations	array	

20.1.1.1.75. .spec.visualization.kibana.nodeSelector

20.1.1.1.75.1. 説明

20.1.1.1.75.1.1. 型

- object

20.1.1.1.76. .spec.visualization.kibana.proxy

20.1.1.1.76.1. 説明

20.1.1.1.76.1.1. 型

- object

プロパティ	型	説明
resources	object	

20.1.1.1.77. .spec.visualization.kibana.proxy.resources

20.1.1.1.77.1. 説明

20.1.1.1.77.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.78. .spec.visualization.kibana.proxy.resources.limits

20.1.1.1.78.1. 説明

20.1.1.1.78.1.1. 型

- object

20.1.1.1.79. .spec.visualization.kibana.proxy.resources.requests

20.1.1.1.79.1. 説明

20.1.1.1.79.1.1. 型

- object

20.1.1.1.80. .spec.visualization.kibana.replicas

20.1.1.1.80.1. 説明

20.1.1.1.80.1.1. 型

- int

20.1.1.1.81. .spec.visualization.kibana.resources

20.1.1.1.81.1. 説明

20.1.1.1.81.1.1. 型

- object

プロパティ	型	説明
limits	object	(オプション) Limits は、許可されるコンピューティングリソースの最大量を示します。
requests	object	(オプション) Requests は、必要なコンピューティングリソースの最小量を示します。

20.1.1.1.82. .spec.visualization.kibana.resources.limits

20.1.1.1.82.1. 説明

20.1.1.1.82.1.1. 型

- object

20.1.1.1.83. .spec.visualization.kibana.resources.requests

20.1.1.1.83.1. 説明

20.1.1.1.83.1.1. 型

- object

20.1.1.1.84. .spec.visualization.kibana.tolerations[]

20.1.1.1.84.1. 説明

20.1.1.1.84.1.1. 型

- array

プロパティ	型	説明
effect	string	(オプション) Effect は、一致する Taint 効果を示します。空の場合は、すべてのテイント効果に一致します。

プロパティ	型	説明
鍵 (key)	string	(オプション) Key は、Toleration が適用される Taint キーです。空の場合は、すべてのテイントキーに一致します。
operator	string	(オプション) Operator は、キーと値の関係を表します。
tolerationSeconds	int	(オプション) TolerationSeconds は、Toleration の期間を表します。
値	string	(オプション) Value は、Toleration が一致する Taint 値です。

20.1.1.1.85. .spec.visualization.kibana.tolerations[].tolerationSeconds

20.1.1.1.85.1. 説明

20.1.1.1.85.1.1. 型

- int

20.1.1.1.86. .status

20.1.1.1.86.1. 説明

ClusterLoggingStatus は、ClusterLogging の監視状態を定義します。

20.1.1.1.86.1.1. 型

- object

プロパティ	型	説明
コレクション	object	(オプション)
conditions	object	(オプション)
キュレーション	object	(オプション)
logStore	object	(オプション)
可視化	object	(オプション)

20.1.1.1.87. .status.collection

20.1.1.1.87.1. 説明

20.1.1.1.87.1.1. 型

- object

プロパティ	型	説明
logs	object	(オプション)

20.1.1.1.88. .status.collection.logs

20.1.1.1.88.1. 説明

20.1.1.1.88.1.1. 型

- object

プロパティ	型	説明
fluentdStatus	object	(オプション)

20.1.1.1.89. .status.collection.logs.fluentdStatus

20.1.1.1.89.1. 説明

20.1.1.1.89.1.1. 型

- object

プロパティ	型	説明
clusterCondition	object	(オプション)
daemonSet	string	(オプション)
nodes	object	(オプション)
Pods	string	(オプション)

20.1.1.1.90. .status.collection.logs.fluentdStatus.clusterCondition

20.1.1.1.90.1. 説明

operator-sdk generate crds は、map-of-slice を許可していません。名前付きタイプを使用する必要があります。

20.1.1.1.90.1.1. 型

- object

20.1.1.1.91. .status.collection.logs.fluentdStatus.nodes

20.1.1.1.91.1. 説明

20.1.1.1.91.1.1. 型

- object

20.1.1.1.92. .status.conditions

20.1.1.1.92.1. 説明

20.1.1.1.92.1.1. 型

- object

20.1.1.1.93. .status.curation

20.1.1.1.93.1. 説明

20.1.1.1.93.1.1. 型

- object

プロパティ	型	説明
curatorStatus	array	(オプション)

20.1.1.1.94. .status.curation.curatorStatus[]

20.1.1.1.94.1. 説明

20.1.1.1.94.1.1. 型

- array

プロパティ	型	説明
clusterCondition	object	(オプション)
cronJobs	string	(オプション)

プロパティ	型	説明
スケジュール	string	(オプション)
suspended	bool	(オプション)

20.1.1.1.95. .status.curation.curatorStatus[].clusterCondition

20.1.1.1.95.1. 説明

operator-sdk generate crds は、map-of-slice を許可していません。名前付きタイプを使用する必要があります。

20.1.1.1.95.1.1. 型

- object

20.1.1.1.96. .status.logStore

20.1.1.1.96.1. 説明

20.1.1.1.96.1.1. 型

- object

プロパティ	型	説明
elasticsearchStatus	array	(オプション)

20.1.1.1.97. .status.logStore.elasticsearchStatus[]

20.1.1.1.97.1. 説明

20.1.1.1.97.1.1. 型

- array

プロパティ	型	説明
cluster	object	(オプション)
clusterConditions	object	(オプション)
clusterHealth	string	(オプション)
clusterName	string	(オプション)

プロパティ	型	説明
デプロイメント	array	(オプション)
nodeConditions	object	(オプション)
nodeCount	int	(オプション)
Pods	object	(オプション)
replicaSets	array	(オプション)
shardAllocationEnabled	string	(オプション)
statefulSets	array	(オプション)

20.1.1.1.98. .status.logStore.elasticsearchStatus[].cluster

20.1.1.1.98.1. 説明

20.1.1.1.98.1.1. 型

- object

プロパティ	型	説明
activePrimaryShards	int	Elasticsearch クラスターのアクティブなプライマリーシャードの数
activeShards	int	Elasticsearch クラスターのアクティブなシャードの数
initializingShards	int	Elasticsearch クラスターの初期化中のシャードの数
numDataNodes	int	Elasticsearch クラスターのデータノードの数
numNodes	int	Elasticsearch クラスターのノードの数
pendingTasks	int	
relocatingShards	int	Elasticsearch クラスターの再配置シャードの数

プロパティ	型	説明
status	string	Elasticsearch クラスターの現在のステータス
unassignedShards	int	Elasticsearch クラスターの未割り当てシャードの数

20.1.1.1.99. `.status.logStore.elasticsearchStatus[].clusterConditions`

20.1.1.1.99.1. 説明

20.1.1.1.99.1.1. 型

- object

20.1.1.1.100. `.status.logStore.elasticsearchStatus[].deployments[]`

20.1.1.1.100.1. 説明

20.1.1.1.100.1.1. 型

- array

20.1.1.1.101. `.status.logStore.elasticsearchStatus[].nodeConditions`

20.1.1.1.101.1. 説明

20.1.1.1.101.1.1. 型

- object

20.1.1.1.102. `.status.logStore.elasticsearchStatus[].pods`

20.1.1.1.102.1. 説明

20.1.1.1.102.1.1. 型

- object

20.1.1.1.103. `.status.logStore.elasticsearchStatus[].replicaSets[]`

20.1.1.1.103.1. 説明

20.1.1.1.103.1.1. 型

- array

20.1.1.1.104. `.status.logStore.elasticsearchStatus[].statefulSets[]`

20.1.1.1.104.1. 説明

20.1.1.1.104.1.1. 型

- array

20.1.1.1.105. `.status.visualization`

20.1.1.1.105.1. 説明

20.1.1.1.105.1.1. 型

- object

プロパティ	型	説明
kibanaStatus	array	(オプション)

20.1.1.1.106. `.status.visualization.kibanaStatus[]`

20.1.1.1.106.1. 説明

20.1.1.1.106.1.1. 型

- array

プロパティ	型	説明
clusterCondition	object	(オプション)
deployment	string	(オプション)
Pods	string	(オプション) 可視化コンポーネントの各 Kibana Pod のステータス
replicaSets	array	(オプション)
replicas	int	(オプション)

20.1.1.1.107. `.status.visualization.kibanaStatus[].clusterCondition`

20.1.1.1.107.1. 説明

20.1.1.1.107.1.1. 型

- object

20.1.1.1.108. .status.visualization.kibanaStatus[].replicaSets[]

20.1.1.1.108.1. 説明

20.1.1.1.108.1.1. 型

- array

第21章 用語集

この用語集では、ロギングのドキュメントで使用される一般的な用語を定義します。

アノテーション

アノテーションを使用して、メタデータをオブジェクトに添付できます。

Red Hat OpenShift Logging Operator

Red Hat OpenShift Logging Operator は、アプリケーション、インフラストラクチャー、監査ログの収集と転送を制御する一連の API を提供します。

カスタムリソース (CR)

CR は Kubernetes API のエクステンションです。ロギングとログ転送を設定するために、**ClusterLogging** および **ClusterLogForwarder** カスタムリソースをカスタマイズできます。

イベントルーター

イベントルーターは、OpenShift Dedicated のイベントを監視する Pod です。ロギングを使用してログを収集します。

Fluentd

Fluentd は、各 OpenShift Dedicated ノードに常駐するログコレクターです。アプリケーション、インフラストラクチャー、および監査ログを収集し、それらをさまざまな出力に転送します。

ガベージコレクション

ガベージコレクションは、終了したコンテナや実行中の Pod によって参照されていないイメージなどのクラスターリソースをクリーンアップするプロセスです。

Elasticsearch

Elasticsearch は、分散検索および分析エンジンです。OpenShift Dedicated は、ロギングのデフォルトのログストアとして Elasticsearch を使用します。

OpenShift Elasticsearch Operator

OpenShift Elasticsearch Operator は、OpenShift Dedicated で Elasticsearch クラスターを実行するために使用されます。OpenShift Elasticsearch Operator は、Elasticsearch クラスター操作のセルフサービスを提供し、ロギングによって使用されます。

インデックス作成

インデックス作成は、データをすばやく見つけてアクセスするために使用されるデータ構造手法です。インデックスを作成すると、クエリーの処理時に必要なディスクアクセスの量が最小限に抑えられるため、パフォーマンスが最適化されます。

JSON ロギング

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、それらをロギングが管理する 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。

Kibana

Kibana は、ヒストグラム、折れ線グラフ、円グラフを使用して Elasticsearch データを照会、検出、視覚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。

Kubernetes API サーバー

Kubernetes API サーバーは、API オブジェクトのデータを検証して設定します。

Labels

ラベルは、Pod などのオブジェクトのサブセットを整理および選択するために使用できるキーと値のペアです。

ロギング

ロギングを使用すると、クラスター全体のアプリケーション、インフラストラクチャー、監査ログ

を集約できます。また、ログをデフォルトのログストアに保存したり、サードパーティーのシステムに転送したり、デフォルトのログストアに保存されているログを照会して視覚化したりすることもできます。

ロギングコレクター

ロギングコレクターは、クラスターからログを収集してフォーマットし、ログストアまたはサードパーティーシステムに転送します。

ログストア

ログストアは、集約されたログを格納するために使用されます。内部ログストアを使用することも、外部ログストアにログを転送することもできます。

ログビジュアライザー

ログビジュアライザーは、ログ、グラフ、チャート、その他のメトリクスなどの情報を表示するために使用できるユーザーインターフェイス (UI) コンポーネントです。

ノード

ノードは、OpenShift Dedicated クラスター内のワーカーマシンです。ノードは、仮想マシン (VM) または物理マシンのいずれかです。

Operator

Operator は、OpenShift Dedicated クラスターで Kubernetes アプリケーションをパッケージ化、デプロイ、および管理するための推奨される方法です。Operator は、人間による操作に関する知識を取り入れて、簡単にパッケージ化してお客様と共有できるソフトウェアにエンコードします。

Pod

Pod は、Kubernetes における最小の論理単位です。Pod は1つ以上のコンテナで構成され、ワーカーノードで実行されます。

ロールベースアクセス制御 (RBAC)

RBAC は、クラスターユーザーとワークロードが、ロールを実行するために必要なリソースにのみアクセスできるようにするための重要なセキュリティコントロールです。

シャード

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、各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。

テイント

テイントは、Pod が適切なノードに確実にスケジュールされるようにします。ノードに1つ以上のテイントを適用できます。

容認

Pod に容認を適用できます。Tolerations を使用すると、スケジューラーは、テイントが一致する Pod をスケジュールできます。

Web コンソール

OpenShift Dedicated を管理するためのユーザーインターフェイス (UI)。OpenShift Dedicated の Web コンソールは、<https://console.redhat.com/openshift> にあります。