OpenShift Container Platform 4.17

로깅

OpenShift Container Platform에서 로깅 구성 및 사용

Last Updated: 2024-12-16

OpenShift Container Platform 4.17 로깅

OpenShift Container Platform에서 로깅 구성 및 사용

법적 공지

초록

로깅을 사용하여 로그 데이터를 수집, 시각화, 전달 및 저장하여 문제를 해결하고 성능 병목 현상을 식별하고 OpenShift Container Platform에서 보안 위협을 감지합니다.

차례

1장. 로깅 6.0

1.1. 릴리스 노트

1.1.1. 로깅 6.0.2

이 릴리스에는 RHBA-2024:10051 이 포함되어 있습니다.

1.1.1.1. 버그 수정

- 이번 업데이트 이전에는 Loki가 일부 구성을 올바르게 로드하지 않아 Alibaba Cloud 또는 IBM Cloud 오브젝트 스토리지를 사용할 때 문제가 발생했습니다. 이번 업데이트에서는 Loki의 구성 로드 코드가 수정되어 문제를 해결합니다. (LOG-5325)

- 이번 업데이트 이전에는 수집기에서 구성된 임계값을 초과하는 감사 로그 메시지를 삭제합니다. 이렇게 하면 최대 행 크기와 읽기 주기 동안 읽은 바이트 수에 대한 감사 구성 임계값이 수정됩니다. (LOG-5998)

- 이번 업데이트 이전에는 Cluster Logging Operator가 이전 릴리스에서와 같이 ClusterLogForwarder 인스턴스와 연결된 리소스를 감시하고 조정하지 않았습니다. 이번 업데이트에서는 Operator가 수정되어 해당 Operator가 소유하고 생성하는 모든 리소스를 감시하고 조정합니다. (LOG-6264)

- 이번 업데이트 이전에는 Google Cloud Logging으로 전송되는 알 수 없는 심각도 수준의 로그 이벤트가 벡터 수집기에서 경고를 트리거하여 심각도를 'DEFAULT'로 설정합니다. 이번 업데이트를 통해 이제 로그 심각도 수준이 Google Cloud Logging 사양과 일치하도록 표준화되고 감사 로그에 'INFO' 심각도가 할당됩니다. (LOG-6296)

- 이번 업데이트 이전에는 인프라 네임스페이스가 애플리케이션 입력에 포함된 경우 **log_type** 이 **application** 로 설정되었습니다. 이번 업데이트를 통해 애플리케이션 입력에 포함된 인프라 네임스페이스의 **log_type** 이 **인프라로** 설정됩니다. (LOG-6354)

- 이번 업데이트 이전에는 ClusterLogForwarder에서 컨테이너 이외의 로그 메시지에 **namespace_name**,**container_name**, **pod_name** 이 추가된 ClusterLogForwarder의 **syslog.enrichment** 필드 값을 지정합니다. 이번 업데이트를 통해 **syslog.enrichment** 가 설정된 경우 컨테이너 로그에 **namespace_name**,**container_name** 및 **pod_name** 만 메시지에 포함됩니다. (LOG-6402)

1.1.1.2. CVE

- CVE-2024-6119

- CVE-2024-6232

1.1.2. 로깅 6.0.1

이 릴리스에는 OpenShift Logging 버그 수정 릴리스 6.0.1 이 포함되어 있습니다.

1.1.2.1. 버그 수정

- 이번 업데이트를 통해 수집기의 기본 메모리 제한이 1024 Mi에서 2024 Mi로 증가했습니다. 그러나 사용자는 항상 클러스터 사양 및 요구 사항에 따라 리소스 제한을 조정해야 합니다. (LOG-6180)

- 이번 업데이트 이전에는 Loki Operator가 모든 **AlertingRule** 리소스에 기본 **네임스페이스** 레이블을 추가하지 않아 User-Workload-Monitoring Alertmanager가 이러한 경고의 라우팅을 건너뛰었습니다. 이번 업데이트에서는 규칙 네임스페이스를 모든 경고 및 레코딩 규칙에 레이블로 추가하고 문제를 해결하고 Alertmanager에서 적절한 경고 라우팅을 복원합니다. (LOG-6151)

- 이번 업데이트 이전에는 LokiStack 룰러 구성 요소 뷰가 올바르게 초기화되지 않아 ruler 구성 요소가 비활성화된 경우 잘못된 필드 오류가 발생했습니다. 이번 업데이트에서는 구성 요소 뷰가 빈 값으로 초기화되어 문제를 해결합니다. (LOG-6129)

- 이번 업데이트 이전에는 prune 필터에서 **log_source** 를 설정하여 일관되지 않은 로그 데이터로 이어질 수 있었습니다. 이번 업데이트를 통해 구성이 적용되기 전에 유효성을 검사하고 prune 필터에 **log_source** 를 포함하는 구성이 거부됩니다. (LOG-6202)

1.1.2.2. CVE

- CVE-2024-24791
- CVE-2024-34155
- CVE-2024-34156
- CVE-2024-34158
- CVE-2024-6104
- CVE-2024-6119
- CVE-2024-45490
- CVE-2024-45491
- CVE-2024-45492

1.1.3. 로깅 6.0.0

이 릴리스에는 Red Hat OpenShift 버그 수정 릴리스 6.0.0에 대한 로깅 이 포함되어 있습니다.

참고

로깅은 코어 OpenShift Container Platform과 별도의 릴리스 사이클을 사용하여 설치 가능한 구성 요소로 제공됩니다. Red Hat OpenShift Container Platform 라이프 사이클 정책은 릴리스 호환성에 대해 간단히 설명합니다.

표 1.1. 업스트림 구성 요소 버전

로깅 버전	구성 요소 버전					
Operator	**eventrouter**	**logfilemetricexporter**	**Loki**	**lokistack-gateway**	**opa-openshift**	**vector**
6.0	0.4	1.1	3.1.0	0.1	0.1	0.37.1

1.1.4. 제거 알림

- 이번 릴리스에서는 로깅에서 **ClusterLogging.logging.openshift.io** 및 **ClusterLogForwarder.logging.openshift.io** 사용자 정의 리소스를 더 이상 지원하지 않습니다. 교체 기능에 대한 자세한 내용은 제품 설명서를 참조하십시오. (LOG-5803)

- 이번 릴리스에서는 로깅에서 로그 스토리지(예: Elasticsearch), 시각화(예: Kibana) 또는 Fluentd 기반 로그 수집기를 더 이상 관리하거나 배포하지 않습니다. (LOG-5368)

참고

elasticsearch-operator에서 관리하는 Elasticsearch 및 Kibana를 계속 사용하려면 관리자가 ClusterLogging 리소스를 삭제하기 전에 해당 오브젝트의 ownerRefs를 수정해야 합니다.

1.1.5. 새로운 기능 및 개선 사항

- 이 기능을 사용하면 구성 요소의 책임을 스토리지, 시각화 및 컬렉션과 같은 관련 Operator로 전환하여 Red Hat OpenShift에 대한 로깅을 위한 새로운 아키텍처가 도입되었습니다. 로그 수집 및 전달을 위해 **ClusterLogForwarder.observability.openshift.io** API가 도입되었습니다. Red Hat 관리 Elastic 스택(Elasticsearch 및 Kibana)과 함께 **ClusterLogging.logging.openshift.io** 및 **ClusterLogForwarder.logging.openshift.io** API에 대한 지원이 제거되었습니다. 사용자는 로그 스토리지를 위해 Red Hat **LokiStack** 으로 마이그레이션하는 것이 좋습니다. 기존 관리형 Elasticsearch 배포는 제한된 기간 동안 사용할 수 있습니다. 로그 수집에 대한 자동화된 마이그레이션이 제공되지 않으므로 관리자가 이전 사용자 정의 리소스를 교체하기 위해 새 ClusterLogForwarder.observability.openshift.io 사양을 생성해야 합니다. 자세한 내용은 공식 제품 설명서를 참조하십시오. (LOG-3493)

- 이번 릴리스에서는 로깅 보기 플러그인 배포 책임이 Red Hat OpenShift Logging Operator에서 COO(Cluster Observability Operator)로 전환됩니다. 시각화가 필요한 새 로그 스토리지 설치의 경우 Cluster Observability Operator 및 관련 UIPlugin 리소스를 배포해야 합니다. 자세한 내용은 Cluster Observability Operator 개요 제품 설명서를 참조하십시오. (LOG-5461)

- 이번 개선된 기능을 통해 벡터 수집기 배포 메모리 및 CPU 사용량에 대한 기본 요청 및 제한이 벡터 문서 권장 사항에 따라 설정됩니다. (LOG-4745)

- 이번 개선된 기능에서는 업스트림 버전 v0.37.1에 맞게 Vector를 업데이트합니다. (LOG-5296)

- 이번 개선된 기능으로 로그 수집기가 노드의 파일 시스템에 버퍼가 로그될 때 트리거되고 사용 가능한 공간의 15% 이상을 사용하여 잠재적인 백압 문제를 나타내는 경고가 추가되었습니다. (LOG-5381)

- 이번 개선된 기능을 통해 공통 Kubernetes 레이블을 사용하도록 모든 구성 요소의 선택기가 업데이트되었습니다. (LOG-5906)

- 이번 개선된 기능을 통해 시크릿 대신 ConfigMap으로 배포하도록 수집기 구성이 변경되어 ClusterLogForwarder가 Unmanaged로 설정된 경우 사용자가 구성을 보고 편집할 수 있습니다. (LOG-5599)

- 이번 개선된 기능에는 추적, debug, info, warn, error, off 등의 옵션과 함께 ClusterLogForwarder의 주석을 사용하여 Vector 수집기 로그 수준을 구성하는 기능이 추가되었습니다. (LOG-5372)

- 이번 개선된 기능에는 Amazon CloudMonitor 출력이 여러 AWS 역할을 사용하는 구성을 거부하기 위한 검증이 추가되어 잘못된 로그 라우팅이 발생하지 않습니다. (LOG-5640)

- 이번 개선된 기능을 통해 지표 대시보드에서 Log Cryostats Collected 및 Log Cryostats Sent 그래프가 제거됩니다. (LOG-5964)

- 이번 개선된 기능에서는 ClusterLogForwarder.observability.openshift.io 리소스 및 Red Hat managed LokiStack의 Vector 배포를 포함하여 로깅 6.0 구성 요소 검사에 대한 정보만 캡처하도록 must-gather 기능을 업데이트합니다. (LOG-5949)

- 이번 개선된 기능을 통해 특정 오류 조건에 대한 조기 경고를 제공하여 Azure 스토리지 시크릿 유효성 검사를 개선할 수 있습니다. (LOG-4571)

1.1.6. 기술 프리뷰 기능

- 이번 릴리스에서는 OpenTelemetry를 사용하여 로그 전달을 위한 기술 프리뷰 기능이 도입되었습니다. 새로운 출력 유형인 OTLP를 사용하면 OpenTelemetry 데이터 모델 및 리소스 의미 규칙을 사용하여 JSON 인코딩 로그 레코드를 보낼 수 있습니다. (LOG-4225)

1.1.7. 버그 수정

- 이번 업데이트 이전에는 **CollectorHighErrorRate** 및 **CollectorVeryHighErrorRate** 경고가 계속 표시되었습니다. 이번 업데이트를 통해 logging 6.0 릴리스에서 두 경고가 모두 제거되지만 향후 릴리스에서 반환될 수 있습니다. (LOG-3432)

1.1.8. CVE

- CVE-2024-34397

1.2. 로깅 6.0

ClusterLogForwarder CR(사용자 정의 리소스)은 로그 수집 및 전달의 중앙 구성 지점입니다.

1.2.1. 입력 및 출력

입력은 전달할 로그 소스를 지정합니다. 로깅은 클러스터의 다른 부분에서 로그를 선택하는 **애플리케이션**,**인프라** 및 **감사** 와 같은 기본 입력 유형을 제공합니다. 네임스페이스 또는 Pod 레이블을 기반으로 사용자 정의 입력을 정의하여 로그 선택을 미세 조정할 수도 있습니다.

출력은 로그가 전송되는 대상을 정의합니다. 각 출력 유형에는 고유한 구성 옵션 세트가 있어 동작 및 인증 설정을 사용자 지정할 수 있습니다.

1.2.2. 수신자 입력 유형

수신자 입력 유형을 사용하면 로깅 시스템에서 외부 소스의 로그를 허용할 수 있습니다. 로그를 수신하기 위해 **http** 및 **syslog** 의 두 형식을 지원합니다.

ReceiverSpec 은 수신자 입력에 대한 구성을 정의합니다.

1.2.3. 파이프라인 및 필터

파이프라인은 입력에서 출력으로 로그 흐름을 결정합니다. 파이프라인은 하나 이상의 입력 참조, 출력 참조 및 선택적 필터 참조로 구성됩니다. 필터를 사용하여 파이프라인 내에서 로그 메시지를 변환하거나 삭제할 수 있습니다. 필터 순서는 순차적으로 적용되므로 중요하며 이전 필터로 인해 로그 메시지가 이후 단계에 도달하지 못할 수 있습니다.

1.2.4. Operator 동작

Cluster Logging Operator는 **managementState** 필드를 기반으로 수집기의 배포 및 구성을 관리합니다.

- **Managed** (기본값)로 설정하면 Operator에서 사양에 정의된 구성과 일치하도록 로깅 리소스를 적극적으로 관리합니다.
- **Unmanaged** 로 설정하면 Operator에서 작업을 수행하지 않으므로 로깅 구성 요소를 수동으로 관리할 수 있습니다.

1.2.5. 검증

로깅에는 광범위한 검증 규칙과 기본값이 포함되어 있어 원활하고 오류가 없는 구성 환경을 보장합니다. **ClusterLogForwarder** 리소스는 필수 필드, 필드 간 종속성 및 입력 값 형식에 대한 검증 검사를 적용합니다. 특정 필드에 기본값이 제공되어 일반적인 시나리오에서 명시적 구성의 필요성이 줄어듭니다.

1.2.5.1. 빠른 시작

사전 요구 사항

- 클러스터 관리자 권한

프로세스

1. OperatorHub에서 **OpenShift Logging** 및 **Loki** Operator를 설치합니다.
2. **openshift-logging** 네임스페이스에서 **LokiStack** CR(사용자 정의 리소스)을 생성합니다.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  managementState: Managed
  size: 1x.extra-small
  storage:
    schemas:
    - effectiveDate: '2022-06-01'
      version: v13
    secret:
      name: logging-loki-s3
      type: s3
    storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

3. 수집기의 서비스 계정을 생성합니다.

```
$ oc create sa collector -n openshift-logging
```

4. 수집기에 대한 **ClusterRole** 을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: logging-collector-logs-writer
rules:
```

```
- apiGroups:
  - loki.grafana.com
  resourceNames:
  - logs
  resources:
  - application
  - audit
  - infrastructure
  verbs:
  - create
```

5. **ClusterRole** 을 서비스 계정에 바인딩합니다.

```
$ oc adm policy add-cluster-role-to-user logging-collector-logs-writer -z collector
```

6. Cluster Observability Operator를 설치합니다.

7. **UIPlugin** 을 생성하여 모니터링 탭의 로그 섹션을 활성화합니다.

```
apiVersion: observability.openshift.io/v1alpha1
kind: UIPlugin
metadata:
  name: logging
spec:
  type: Logging
  logging:
    lokiStack:
      name: logging-loki
```

8. 수집기 서비스 계정에 역할을 추가합니다.

```
$ oc project openshift-logging
$ oc adm policy add-cluster-role-to-user collect-application-logs -z collector
$ oc adm policy add-cluster-role-to-user collect-audit-logs -z collector
$ oc adm policy add-cluster-role-to-user collect-infrastructure-logs -z collector
```

9. **ClusterLogForwarder** CR을 생성하여 로그 전달을 구성합니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  namespace: openshift-logging
spec:
  serviceAccount:
    name: collector
  outputs:
  - name: default-lokistack
    type: lokiStack
    lokiStack:
      target:
        name: logging-loki
        namespace: openshift-logging
      authentication:
        token:
```

```
      from: serviceAccount
  tls:
    ca:
      key: service-ca.crt
      configMapName: openshift-service-ca.crt
pipelines:
- name: default-logstore
  inputRefs:
  - application
  - infrastructure
  outputRefs:
  - default-lokistack
```

10. OpenShift 웹 콘솔의 Observe 탭의 Log 섹션에 로그가 표시되는지 확인합니다.

1.3. LOGGING 6.0으로 업그레이드

로깅 v6.0은 이전 릴리스에서 중요한 업그레이드로, 클러스터 로깅의 여러 장기 목표를 달성합니다.

- 로깅 구성 요소(예: 수집기, 스토리지, 시각화)를 관리하기 위한 별도의 운영자 소개.
- Elastic 제품(예: Elasticsearch, Kibana)을 기반으로 관리형 로그 스토리지 및 시각화에 대한 지원 제거
- Fluentd 로그 수집기 구현의 사용 중단
- **ClusterLogging.logging.openshift.io** 및 **ClusterLogForwarder.logging.openshift.io** 리소스에 대한 지원 제거

참고

cluster-logging-operator 는 자동 업그레이드 프로세스를 제공하지 않습니다.

로그 수집, 전달 및 스토리지에 대한 다양한 구성이 제공되어 **cluster-logging-operator** 에서 자동 업그레이드가 제공되지 않습니다. 이 문서는 관리자가 기존 **ClusterLogging.logging.openshift.io** 및 **ClusterLogForwarder.logging.openshift.io** 사양을 새 API로 변환하는 데 도움이 됩니다. 일반적인 사용 사례에 대한 마이그레이션된 **ClusterLogForwarder.observability.openshift.io** 리소스의 예가 포함되어 있습니다.

1.3.1. oc explain 명령 사용

oc explain 명령은 OpenShift CLI **oc** 의 사용자 정의 리소스(CR) 내의 필드에 대한 자세한 설명을 제공하는 필수 툴입니다. 이 명령은 OpenShift 클러스터에서 리소스를 구성하거나 문제를 해결하는 관리자 및 개발자에게 매우 중요합니다.

1.3.1.1. 리소스 설명

oc explain 은 특정 오브젝트와 관련된 모든 필드에 대한 심층적인 설명을 제공합니다. 여기에는 Pod 및 서비스와 같은 표준 리소스뿐만 아니라 상태 저장 세트 및 Operator에서 정의한 사용자 정의 리소스와 같은 더 복잡한 엔티티가 포함됩니다.

ClusterLogForwarder 사용자 정의 리소스의 **outputs** 필드에 대한 문서를 보려면 다음을 사용합니다.

```
$ oc explain clusterlogforwarders.observability.openshift.io.spec.outputs
```

참고

clusterlogforwarder 대신 단축 **형식을 사용할** 수 있습니다.

그러면 유형, 기본값 및 관련 하위 필드를 포함하여 이러한 필드에 대한 자세한 정보가 표시됩니다.

1.3.1.2. 계층 구조 구조

명령은 리소스 필드의 구조를 계층 구조로 표시하여 다양한 구성 옵션 간의 관계를 명확히 합니다.

예를 들어 **LokiStack** 사용자 정의 리소스의 **스토리지** 구성을 드릴다운하는 방법은 다음과 같습니다.

```
$ oc explain lokistacks.loki.grafana.com
$ oc explain lokistacks.loki.grafana.com.spec
$ oc explain lokistacks.loki.grafana.com.spec.storage
$ oc explain lokistacks.loki.grafana.com.spec.storage.schemas
```

각 명령은 리소스 사양의 깊은 수준을 표시하여 구조를 지웁니다.

1.3.1.3. 유형 정보

oc explain 은 각 필드의 유형(예: 문자열, 정수 또는 부울)을 나타내며 리소스 정의가 올바른 데이터 유형을 사용하는지 확인할 수 있습니다.

예를 들면 다음과 같습니다.

```
$ oc explain lokistacks.loki.grafana.com.spec.size
```

이 경우 정수 값을 사용하여 **크기를** 정의해야 합니다.

1.3.1.4. 기본값

해당하는 경우 명령은 필드의 기본값을 표시하여 명시적으로 지정되지 않은 경우 사용할 값에 대한 통찰력을 제공합니다.

lokistacks.loki.grafana.com 을 예제로 다시 사용합니다.

```
$ oc explain lokistacks.spec.template.distributor.replicas
```

출력 예

```
GROUP:      loki.grafana.com
KIND:       LokiStack
VERSION:    v1

FIELD: replicas <integer>

DESCRIPTION:
    Replicas defines the number of replica pods of the component.
```

1.3.2. 로그 스토리지

이 릴리스에서 사용할 수 있는 유일한 관리형 로그 스토리지 솔루션은 **loki-operator** 에서 관리하는 Lokistack입니다. 이전에 관리형 Elasticsearch 오퍼링의 대안으로 사용 가능한 이 솔루션은 배포 프로세스에서 변경되지 않은 상태로 유지됩니다.

중요

elasticsearch-operator 에서 제공하는 기존 Red Hat 관리 Elasticsearch 또는 Kibana 배포를 계속 사용하려면 동일한 네임스페이스에서 ClusterLogging이라는 **ClusterLogging** 리소스를 제거하기 전에 **elasticsearch** elasticsearch 라는 **Elasticsearch** 리소스와 **openshift-logging** 네임스페이스의 **kibana 라는 Kibana** 리소스를 제거합니다.

1. 임시로 **ClusterLogging** 을 **Unmanaged**상태로 설정

```
$ oc -n openshift-logging patch clusterlogging/instance -p '{"spec":{"managementState":
"Unmanaged"}}' --type=merge
```

2. **Elasticsearch** 리소스에서 **ClusterLogging ownerReferences** 제거
   다음 명령은 **ClusterLogging** 이 더 이상 **Elasticsearch** 리소스를 소유하지 않도록 합니다. **ClusterLogging** 리소스의 **logStore** 필드에 대한 업데이트는 더 이상 **Elasticsearch** 리소스에 영향을 미치지 않습니다.

```
$ oc -n openshift-logging patch elasticsearch/elasticsearch -p '{"metadata":
{"ownerReferences": []}}' --type=merge
```

3. **Kibana** 리소스에서 **ClusterLogging ownerReferences** 제거
   다음 명령은 **ClusterLogging** 이 더 이상 **Kibana** 리소스를 소유하지 않도록 합니다. **ClusterLogging** 리소스의 **시각화** 필드에 대한 업데이트는 더 이상 **Kibana** 리소스에 영향을 미치지 않습니다.

```
$ oc -n openshift-logging patch kibana/kibana -p '{"metadata":{"ownerReferences": []}}' --
type=merge
```

4. **ClusterLogging** 을 state **Managed**로 설정

```
$ oc -n openshift-logging patch clusterlogging/instance -p '{"spec":{"managementState": "Managed"}}'
--type=merge
```

1.3.3. 로그 시각화

로그 시각화를 위한 OpenShift 콘솔 UI 플러그인이 **cluster-logging-operator** 에서 **cluster-observability-operator** 로 이동되었습니다.

1.3.4. 로그 수집 및 전달

이제 **observability.openshift.io** API 그룹의 일부인 새 API 아래에 로그 수집 및 전달 구성이 지정됩니다. 다음 섹션에서는 이전 API 리소스의 차이점을 설명합니다.

참고

벡터는 지원되는 유일한 수집기 구현입니다.

1.3.5. 관리, 리소스 할당 및 워크로드 스케줄링

관리 상태(예: Managed, Unmanaged)에 대한 구성(예: Managed, Unmanaged), 리소스 요청 및 제한, 허용 오차 및 노드 선택이 이제 새 **ClusterLogForwarder** API의 일부입니다.

이전 설정

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
spec:
  managementState: "Managed"
  collection:
    resources:
      limits: {}
      requests: {}
    nodeSelector: {}
    tolerations: {}
```

현재 설정

```
apiVersion: "observability.openshift.io/v1"
kind: ClusterLogForwarder
spec:
  managementState: Managed
  collector:
    resources:
      limits: {}
      requests: {}
    nodeSelector: {}
    tolerations: {}
```

1.3.6. 입력 사양

입력 사양은 **ClusterLogForwarder** 사양의 선택적 부분입니다. 관리자는 계속해서 **애플리케이션**,**인프라** 및 **감사** 의 사전 정의된 값을 사용하여 이러한 소스를 수집할 수 있습니다.

1.3.6.1. 애플리케이션 입력

네임스페이스 및 컨테이너 포함 및 제외가 단일 필드에 통합되었습니다.

5.9 네임스페이스 및 컨테이너가 포함된 애플리케이션 입력 및 제외

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
spec:
  inputs:
   - name: application-logs
     type: application
     application:
       namespaces:
       - foo
       - bar
       includes:
       - namespace: my-important
```

```
        container: main
      excludes:
      - container: too-verbose
```

6.0 애플리케이션 입력 네임스페이스 및 컨테이너 포함 및 제외

```
apiVersion: "observability.openshift.io/v1"
kind: ClusterLogForwarder
spec:
  inputs:
   - name: application-logs
     type: application
     application:
       includes:
       - namespace: foo
       - namespace: bar
       - namespace: my-important
         container: main
       excludes:
       - container: too-verbose
```

참고

애플리케이션,**인프라**, **감사** 는 예약된 단어로, 입력을 정의할 때 이름으로 사용할 수 없습니다.

1.3.6.2. 입력 수신자

입력 수신자에 대한 변경 사항은 다음과 같습니다.

- 수신기 수준에서 유형의 명시적 구성입니다.
- 포트 설정이 수신자 수준으로 이동했습니다.

5.9 입력 수신자

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
spec:
  inputs:
  - name: an-http
    receiver:
      http:
        port: 8443
        format: kubeAPIAudit
  - name: a-syslog
    receiver:
      type: syslog
      syslog:
        port: 9442
```

6.0 입력 수신자

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

```
kind: ClusterLogForwarder
spec:
  inputs:
  - name: an-http
    type: receiver
    receiver:
      type: http
      port: 8443
      http:
        format: kubeAPIAudit
  - name: a-syslog
    type: receiver
    receiver:
      type: syslog
      port: 9442
```

1.3.7. 출력 사양

출력 사양에 대한 높은 수준의 변경 사항은 다음과 같습니다.

- URL 설정은 각 출력 유형 사양으로 이동했습니다.
- 튜닝 매개변수는 각 출력 유형 사양으로 이동했습니다.
- TLS 구성을 인증과 분리합니다.
- TLS 및 인증을 위한 키 및 시크릿/구성 맵에 대한 명시적 구성입니다.

1.3.8. 보안 및 TLS 구성

보안 및 TLS 구성은 각 출력에 대한 인증 및 TLS 구성으로 구분됩니다. 관리자가 인식된 키로 시크릿을 정의하는 대신 사양에 명시적으로 정의해야 합니다. TLS 및 권한 부여 구성을 업그레이드하려면 관리자가 기존 보안을 계속 사용하려면 이전에 인식된 키를 이해해야 합니다. 다음 섹션의 예제에서는 기존 Red Hat 관리 로그 스토리지 솔루션으로 전달하도록 **ClusterLogForwarder** 시크릿을 구성하는 방법에 대한 세부 정보를 제공합니다.

1.3.9. Red Hat Managed Elasticsearch

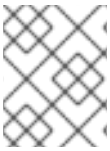
v5.9 Red Hat Managed Elasticsearch로 전달

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

v6.0, Red Hat Managed Elasticsearch로 전달

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

```
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: default-elasticsearch
    type: elasticsearch
    elasticsearch:
      url: https://elasticsearch:9200
      version: 6
      index: <log_type>-write-{+yyyy.MM.dd}
    tls:
      ca:
        key: ca-bundle.crt
        secretName: collector
      certificate:
        key: tls.crt
        secretName: collector
      key:
        key: tls.key
        secretName: collector
  pipelines:
  - outputRefs:
    - default-elasticsearch
  - inputRefs:
    - application
    - infrastructure
```



참고

이 예에서는 애플리케이션 로그가 **app-write** 대신 **application-write** alias/index에 기록됩니다.

1.3.10. Red Hat Managed LokiStack

v5.9 Red Hat Managed LokiStack으로 전달

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

v6.0, Red Hat Managed LokiStack으로 전달

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
```

```
spec:
  outputs:
  - name: default-lokistack
    type: lokiStack
    lokiStack:
      target:
        name: lokistack-dev
        namespace: openshift-logging
      authentication:
        token:
          from: serviceAccount
    tls:
      ca:
        key: service-ca.crt
        configMapName: openshift-service-ca.crt
  pipelines:
  - outputRefs:
    - default-lokistack
  - inputRefs:
    - application
    - infrastructure
```

1.3.11. 필터 및 파이프라인 구성

이제 파이프라인 구성이 필요한 변환을 필터로 별도로 구성하여 출력 대상에 대한 입력 소스의 라우팅만 정의합니다. 이전 릴리스의 파이프라인의 모든 속성이 이 릴리스의 필터로 변환되었습니다. 개별 필터는 **필터** 사양에 정의되어 파이프라인에서 참조합니다.

5.9 필터

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
spec:
  pipelines:
   - name: application-logs
     parse: json
     labels:
       foo: bar
     detectMultilineErrors: true
```

6.0 필터 설정

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
spec:
  filters:
  - name: detectexception
    type: detectMultilineException
  - name: parse-json
    type: parse
  - name: labels
    type: openshiftLabels
    openshiftLabels:
      foo: bar
  pipelines:
```

```
  - name: application-logs
    filterRefs:
    - detectexception
    - labels
    - parse-json
```

1.3.12. 검증 및 상태

대부분의 검증은 리소스가 생성 또는 업데이트되어 즉각적인 피드백을 제공할 때 적용됩니다. 이는 이전 릴리스의 전환으로, 유효성 검사가 생성 후 수행되었으며 리소스 상태를 검사해야 합니다. 일부 검증은 생성 또는 업데이트 시 유효성을 검사할 수 없는 경우 생성 후 계속 수행됩니다.

ClusterLogForwarder.observability.openshift.io 의 인스턴스는 Operator가 로그 수집기를 배포하기 전에 다음과 같은 조건을 충족해야 합니다: Authorized, Valid, Ready. 이러한 조건의 예는 다음과 같습니다.

6.0 상태 조건

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
status:
  conditions:
  - lastTransitionTime: "2024-09-13T03:28:44Z"
    message: 'permitted to collect log types: [application]'
    reason: ClusterRolesExist
    status: "True"
    type: observability.openshift.io/Authorized
  - lastTransitionTime: "2024-09-13T12:16:45Z"
    message: ""
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/Valid
  - lastTransitionTime: "2024-09-13T12:16:45Z"
    message: ""
    reason: ReconciliationComplete
    status: "True"
    type: Ready
  filterConditions:
  - lastTransitionTime: "2024-09-13T13:02:59Z"
    message: filter "detectexception" is valid
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/ValidFilter-detectexception
  - lastTransitionTime: "2024-09-13T13:02:59Z"
    message: filter "parse-json" is valid
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/ValidFilter-parse-json
  inputConditions:
  - lastTransitionTime: "2024-09-13T12:23:03Z"
    message: input "application1" is valid
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/ValidInput-application1
  outputConditions:
```

```
  - lastTransitionTime: "2024-09-13T13:02:59Z"
    message: output "default-lokistack-application1" is valid
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/ValidOutput-default-lokistack-application1
  pipelineConditions:
  - lastTransitionTime: "2024-09-13T03:28:44Z"
    message: pipeline "default-before" is valid
    reason: ValidationSuccess
    status: "True"
    type: observability.openshift.io/ValidPipeline-default-before
```

참고

충족되고 적용 가능한 조건은 "True"의 "상태" 값이 있습니다. "True" 이외의 상태가 있는 조건은 이유 및 문제를 설명하는 메시지를 제공합니다.

1.4. 로그 전달 구성

ClusterLogForwarder (CLF)를 사용하면 사용자가 다양한 대상으로 로그 전달을 구성할 수 있습니다. 다른 소스의 로그 메시지를 선택하고, 변환하거나 필터링할 수 있는 파이프라인을 통해 보내고, 하나 이상의 출력으로 전달할 수 있는 유연한 방법을 제공합니다.

ClusterLogForwarder의 주요 기능

- 입력을 사용하여 로그 메시지 선택
- 출력을 사용하여 외부 대상으로 로그를 전달
- 필터를 사용하여 로그 메시지를 필터링, 변환 및 삭제
- 입력, 필터 및 출력을 연결하는 로그 전달 파이프라인을 정의합니다.

1.4.1. 로그 컬렉션 설정

이 클러스터 로깅 릴리스에서는 관리자가 **ClusterLogForwarder** 와 연결된 서비스 계정에 로그 수집 권한을 명시적으로 부여해야 합니다. **ClusterLogging** 및 **ClusterLogForwarder.logging.openshift.io** 리소스로 구성된 레거시 로깅 시나리오의 이전 릴리스에서는 이 작업이 필요하지 않았습니다.

Red Hat OpenShift Logging Operator는 **collect-audit-logs**,**collect-application-logs**, **collect-infrastructure-logs** 클러스터 역할을 제공하여 수집기가 감사 로그, 애플리케이션 로그 및 인프라 로그를 각각 수집할 수 있습니다.

필요한 클러스터 역할을 서비스 계정에 바인딩하여 로그 컬렉션을 설정합니다.

1.4.1.1. 레거시 서비스 계정

기존 서비스 계정 **logcollector** 를 사용하려면 다음 **ClusterRoleBinding** 을 생성합니다.

```
$ oc adm policy add-cluster-role-to-user collect-application-logs system:serviceaccount:openshift-logging:logcollector
$ oc adm policy add-cluster-role-to-user collect-infrastructure-logs system:serviceaccount:openshift-logging:logcollector
```

또한 감사 로그를 수집하는 경우 다음 **ClusterRoleBinding** 을 생성합니다.

```
$ oc adm policy add-cluster-role-to-user collect-audit-logs system:serviceaccount:openshift-logging:logcollector
```

1.4.1.2. 서비스 계정 생성

사전 요구 사항

- Red Hat OpenShift Logging Operator는 **openshift-logging** 네임스페이스에 설치됩니다.
- 관리자 권한이 있습니다.

프로세스

1. 수집기의 서비스 계정을 생성합니다. 인증을 위해 토큰이 필요한 스토리지에 로그를 작성하려면 서비스 계정에 토큰을 포함해야 합니다.
2. 적절한 클러스터 역할을 서비스 계정에 바인딩합니다.

   바인딩 명령 예

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

1.4.1.2.1. 서비스 계정의 클러스터 역할 바인딩

role_binding.yaml 파일은 ClusterLogging Operator의 ClusterRole을 특정 ServiceAccount에 바인딩하여 클러스터 전체에서 Kubernetes 리소스를 관리할 수 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: manager-rolebinding
roleRef:                                          1
  apiGroup: rbac.authorization.k8s.io             2
  kind: ClusterRole                               3
  name: cluster-logging-operator                  4
subjects:                                         5
  - kind: ServiceAccount                          6
    name: cluster-logging-operator                7
    namespace: openshift-logging                  8
```

1. roleRef: 바인딩이 적용되는 ClusterRole을 참조합니다.
2. apiGroup: RBAC API 그룹을 나타내며 ClusterRole이 Kubernetes RBAC 시스템의 일부임을 지정합니다.
3. kind: 참조된 역할이 클러스터 전체에서 적용되는 ClusterRole임을 지정합니다.
4. name: ServiceAccount에 바인딩되는 ClusterRole의 이름입니다. 여기서 cluster-logging-operator.

5 제목: ClusterRole에서 권한을 부여하는 엔터티(사용자 또는 서비스 계정)를 정의합니다.

6 kind: subject가 ServiceAccount임을 지정합니다.

7 name: 권한이 부여된 ServiceAccount의 이름입니다.

8 namespace: ServiceAccount가 있는 네임스페이스를 나타냅니다.

1.4.1.2.2. 애플리케이션 로그 작성

write-application-logs-clusterrole.yaml 파일은 Loki 로깅 애플리케이션에 애플리케이션 로그를 작성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-application-logs
rules:                                          1
  - apiGroups:                                  2
      - loki.grafana.com                        3
    resources:                                  4
      - application                             5
    resourceNames:                              6
      - logs                                    7
    verbs:                                      8
      - create                                  9
Annotations
<1> rules: Specifies the permissions granted by this ClusterRole.
<2> apiGroups: Refers to the API group loki.grafana.com, which relates to the Loki logging system.
<3> loki.grafana.com: The API group for managing Loki-related resources.
<4> resources: The resource type that the ClusterRole grants permission to interact with.
<5> application: Refers to the application resources within the Loki logging system.
<6> resourceNames: Specifies the names of resources that this role can manage.
<7> logs: Refers to the log resources that can be created.
<8> verbs: The actions allowed on the resources.
<9> create: Grants permission to create new logs in the Loki system.
```

1.4.1.2.3. 감사 로그 작성

write-audit-logs-clusterrole.yaml 파일은 Loki 로깅 시스템에서 감사 로그를 생성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-audit-logs
rules:                                          1
  - apiGroups:                                  2
      - loki.grafana.com                        3
    resources:                                  4
      - audit                                   5
    resourceNames:                              6
```

```
    - logs                          7
  verbs:                            8
    - create                        9
```

1 1 rules: 이 ClusterRole에서 부여하는 권한을 정의합니다.

2 2 apiGroups: loki.grafana.com API 그룹을 지정합니다.

3 3 Loki.grafana.com: Loki 로깅 리소스를 담당하는 API 그룹입니다.

4 4 resources: 이 역할이 관리하는 리소스 유형을 나타냅니다(이 경우 audit).

5 5 audit: 역할이 Loki 내에서 감사 로그를 관리하도록 지정합니다.

6 6 resourceNames: 역할이 액세스할 수 있는 특정 리소스를 정의합니다.

7 7 logs: 이 역할에서 관리할 수 있는 로그를 참조합니다.

8 8 동사: 리소스에서 허용되는 작업입니다.

9 9 create: 새 감사 로그를 생성할 수 있는 권한을 부여합니다.

1.4.1.2.4. 인프라 로그 작성

write-infrastructure-logs-clusterrole.yaml 파일은 Loki 로깅 시스템에서 인프라 로그를 생성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

샘플 **YAML**

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-infrastructure-logs
rules:                              1
  - apiGroups:                      2
      - loki.grafana.com            3
    resources:                      4
      - infrastructure              5
    resourceNames:                  6
      - logs                        7
    verbs:                          8
      - create                      9
```

1 rules: 이 ClusterRole에서 부여하는 권한을 지정합니다.

2 apiGroups: Loki 관련 리소스의 API 그룹을 지정합니다.

3 Loki.grafana.com: Loki 로깅 시스템을 관리하는 API 그룹입니다.

4 resources: 이 역할이 상호 작용할 수 있는 리소스 유형을 정의합니다.

5 인프라: 이 역할이 관리하는 인프라 관련 리소스를 나타냅니다.

6 resourceNames: 이 역할이 관리할 수 있는 리소스의 이름을 지정합니다.

7 로그: 인프라와 관련된 로그 리소스를 참조합니다.

8 동사: 이 역할에서 허용하는 작업입니다.

9 생성: Loki 시스템에서 인프라 로그를 생성할 수 있는 권한을 부여합니다.

1.4.1.2.5. ClusterLogForwarder 편집기 역할

clusterlogforwarder-editor-role.yaml 파일은 사용자가 OpenShift에서 ClusterLogForwarder를 관리할 수 있는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: clusterlogforwarder-editor-role
rules:                                  1
  - apiGroups:                          2
      - observability.openshift.io      3
    resources:                          4
      - clusterlogforwarders            5
    verbs:                              6
      - create                          7
      - delete                          8
      - get                             9
      - list                            10
      - patch                           11
      - update                          12
      - watch                           13
```

1 rules: 이 ClusterRole에서 부여하는 권한을 지정합니다.

2 apiGroups: OpenShift 관련 API 그룹을 참조합니다.

3 obervability.openshift.io: 로깅과 같은 관찰 기능 리소스를 관리하는 API 그룹입니다.

4 resources: 이 역할이 관리할 수 있는 리소스를 지정합니다.

5 clusterlogforwarders: OpenShift의 로그 전달 리소스를 참조합니다.

6 verbs: ClusterLogForwarders에 허용되는 작업을 지정합니다.

7 생성: 새 ClusterLogForwarder를 생성할 수 있는 권한을 부여합니다.

8 delete: 기존 ClusterLogForwarder를 삭제할 수 있는 권한을 부여합니다.

9 get: 특정 ClusterLogForwarder에 대한 정보를 검색할 수 있는 권한을 부여합니다.

10 list: 모든 ClusterLogForwarder를 나열할 수 있습니다.

11 patch: ClusterLogForwarder를 부분적으로 수정할 수 있는 권한을 부여합니다.

12 update: 기존 ClusterLogForwarder를 업데이트할 수 있는 권한을 부여합니다.

13 watch: ClusterLogForwarder의 변경 사항을 모니터링할 수 있는 권한을 부여합니다.

1.4.2. 수집기에서 로그 수준 수정

수집기에서 로그 수준을 수정하려면 **observability.openshift.io/log-level** 주석을 **추적** 하도록 설정하고,**디버그**,**info**,**warn**,**error**, **off** 를 설정할 수 있습니다.

로그 수준 주석의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  annotations:
    observability.openshift.io/log-level: debug
# ...
```

1.4.3. Operator 관리

ClusterLogForwarder 리소스에는 Operator가 리소스를 적극적으로 관리하는지 또는 Unmanaged 상태로 둘지 여부를 제어하는 **managementState** 필드가 있습니다.

관리됨

(기본값) Operator는 CLF 사양의 원하는 상태와 일치하도록 로깅 리소스를 구동합니다.

Unmanaged

Operator는 로깅 구성 요소와 관련된 작업을 수행하지 않습니다.

이를 통해 관리자는 **managementState** 를 **Unmanaged** 로 설정하여 로그 전달을 일시적으로 일시 중지할 수 있습니다.

1.4.4. ClusterLogForwarder의 구조

CLF에는 다음과 같은 주요 구성 요소가 포함된 **spec** 섹션이 있습니다.

입력

전달할 로그 메시지를 선택합니다. 기본 입력 유형 **애플리케이션**,**인프라** 및 **감사** 는 클러스터의 다른 부분에서 로그를 전달합니다. 사용자 지정 입력을 정의할 수도 있습니다.

출력

로그를 전달할 대상을 정의합니다. 각 출력에는 고유한 이름과 유형별 구성이 있습니다.

파이프라인

입력에서 필터를 통해 출력으로 가져오는 경로 로그를 정의합니다. 파이프라인에는 고유한 이름이 있으며 입력, 출력 및 필터 이름 목록으로 구성됩니다.

필터

파이프라인에서 로그 메시지를 변환하거나 삭제합니다. 사용자는 특정 로그 필드와 일치하는 필터를 정의하고 메시지를 삭제하거나 수정할 수 있습니다. 필터는 파이프라인에 지정된 순서로 적용됩니다.

1.4.4.1. 입력

입력은 **spec.inputs** 의 배열에 구성됩니다. 세 가지 기본 제공 입력 유형이 있습니다.

애플리케이션

기본, openshift 또는 **kube-** 또는 **openshift -** 접두사가 있는 네임스페이스를 제외하고 모든 애플리케이션 컨테이너에서 로그를 선택합니다.

인프라

기본 및 **openshift** 네임스페이스 및 노드 로그에서 실행되는 인프라 구성 요소에서 로그를 선택합니다.

audit

auditd에서 OpenShift API 서버 감사 로그, Kubernetes API 서버 감사 로그, ovn 감사 로그 및 노드 감사 로그에서 로그를 선택합니다.

사용자는 특정 네임스페이스에서 로그를 선택하거나 Pod 레이블을 사용하는 유형 **애플리케이션** 의 사용자 정의 입력을 정의할 수 있습니다.

1.4.4.2. 출력

출력은 **spec.outputs** 아래의 배열에 구성됩니다. 각 출력에는 고유한 이름과 유형이 있어야 합니다. 지원되는 유형은 다음과 같습니다.

azureMonitor

Azure Monitor로 로그를 전달합니다.

CloudMonitor

AWS CloudMonitor로 로그를 전달합니다.

elasticsearch

로그를 외부 Elasticsearch 인스턴스로 전달합니다.

googleCloudLogging

Google Cloud Logging으로 로그를 전달합니다.

HTTP

일반 HTTP 끝점으로 로그를 전달합니다.

kafka

Kafka 브로커로 로그를 전달합니다.

Loki

로그를 Loki 로깅 백엔드로 전달합니다.

lokistack

OpenShift Container Platform 인증 통합을 사용하여 Loki 및 웹 프록시의 로깅 지원 조합으로 로그를 전달합니다. LokiStack의 프록시는 OpenShift Container Platform 인증을 사용하여 멀티 테넌시 적용

otlp

OpenTelemetry 프로토콜을 사용하여 로그를 전달합니다.

splunk

Splunk로 로그를 전달합니다.

syslog

로그를 외부 syslog 서버로 전달합니다.

각 출력 유형에는 고유한 구성 필드가 있습니다.

1.4.4.3. 파이프라인

파이프라인은 **spec.pipelines** 의 배열에 구성됩니다. 각 파이프라인에는 고유한 이름이 있어야 하며 다음으로 구성됩니다.

inputRefs

로그가 이 파이프라인으로 전달되어야 하는 입력의 이름입니다.

outputRefs

로그를 전송할 출력의 이름입니다.

filterRefs

(선택 사항) 적용할 필터 이름입니다.

filterRefs 순서가 순차적으로 적용되므로 중요합니다. 이전 필터는 이후 필터에서 처리되지 않는 메시지를 삭제할 수 있습니다.

1.4.4.4. 필터

필터는 **spec.filters** 아래의 배열에 구성됩니다. 구조화된 필드의 값을 기반으로 들어오는 로그 메시지를 일치시키고 수정하거나 삭제할 수 있습니다.

관리자는 다음 유형의 필터를 구성할 수 있습니다.

1.4.4.5. 여러 줄 예외 탐지 활성화

컨테이너 로그의 여러 줄 오류 감지를 활성화합니다.

주의

이 기능을 활성화하면 성능에 영향을 미칠 수 있으며 추가 컴퓨팅 리소스 또는 대체 로깅 솔루션이 필요할 수 있습니다.

로그 구문 분석기가 별도의 예외와 동일한 예외의 별도의 행을 잘못 식별하는 경우가 많습니다. 이로 인해 추가 로그 항목과 추적된 정보에 대한 불완전하거나 부정확한 보기가 발생합니다.

Java 예외 예

```
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)
```

- 로깅을 사용하여 여러 줄 예외를 감지하고 이를 단일 로그 항목으로 재조정하려면 **ClusterLogForwarder** 사용자 정의 리소스(CR)에 **.spec.filters** 아래의 **detectMultilineErrors** 필드가 포함되어 있는지 확인합니다.

Example ClusterLogForwarder CR

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

```
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name>
  namespace: <log_forwarder_namespace>
spec:
  serviceAccount:
    name: <service_account_name>
  filters:
  - name: <name>
    type: detectMultilineException
  pipelines:
    - inputRefs:
        - <input-name>
      name: <pipeline-name>
      filterRefs:
        - <filter-name>
      outputRefs:
        - <output-name>
```

1.4.4.5.1. 세부 정보

로그 메시지가 예외 스택 추적을 형성하는 연속 시퀀스로 표시되면 단일 통합 로그 레코드로 결합됩니다. 첫 번째 로그 메시지의 콘텐츠는 시퀀스의 모든 메시지 필드의 연결된 콘텐츠로 교체됩니다.

수집기는 다음 언어를 지원합니다.

- Java
- JS
- Ruby
- Python
- Golang
- PHP
- Dart

1.4.4.6. 원하지 않는 로그 레코드를 삭제하도록 콘텐츠 필터 구성

드롭 필터가 구성되면 로그 수집기는 전달 전에 필터에 따라 로그 스트림을 평가합니다. 수집기는 지정된 구성과 일치하는 원하지 않는 로그 레코드를 삭제합니다.

프로세스

1. **ClusterLogForwarder** CR의 **필터 사양에 필터** 구성을 추가합니다.
   다음 예제에서는 정규식을 기반으로 로그 레코드를 삭제하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

   ```
   apiVersion: observability.openshift.io/v1
   kind: ClusterLogForwarder
   ```

```
metadata:
# ...
spec:
 serviceAccount:
   name: <service_account_name>
 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. 필터 유형을 지정합니다. **drop** 필터는 필터 구성과 일치하는 로그 레코드를 삭제합니다.
2. **드롭** 필터를 적용하기 위한 구성 옵션을 지정합니다.
3. 로그 레코드 삭제 여부를 평가하는 데 사용되는 테스트에 대한 구성을 지정합니다.
   - 테스트에 지정된 모든 조건이 true이면 테스트가 통과되고 로그 레코드가 삭제됩니다.
   - **drop** 필터 구성에 대해 여러 테스트가 지정되면 테스트가 통과하면 레코드가 삭제됩니다.
   - 예를 들어 평가 중인 로그 레코드에서 필드가 누락되면 해당 조건이 false로 평가됩니다.
4. 로그 레코드의 필드 경로인 점으로 구분된 필드 경로를 지정합니다. 경로에는 alpha-numeric 문자 및 밑줄(**a-zA-Z0-9_**)을 포함할 수 있습니다(예: **.kubernetes.namespace_name**). 세그먼트에 이 범위를 벗어나는 문자가 포함된 경우 세그먼트는 따옴표로 묶어야 합니다(예: **.kubernetes.labels."foo.bar-bar/baz"**. 단일 **테스트 구성에 여러 필드 경로를 포함할 수 있지만 테스트가** 통과하고 **drop** 필터를 적용하려면 모두 true로 평가되어야 합니다.
5. 정규식을 지정합니다. 로그 레코드가 이 정규식과 일치하는 경우 해당 레코드가 삭제됩니다. 단일 **필드** 경로에 대해 **matches** 또는 **notMatches** 조건을 설정할 수 있지만 둘 다 설정할 수는 없습니다.
6. 정규식을 지정합니다. 로그 레코드가 이 정규식과 일치하지 않으면 삭제됩니다. 단일 **필드** 경로에 대해 **matches** 또는 **notMatches** 조건을 설정할 수 있지만 둘 다 설정할 수는 없습니다.
7. **drop** 필터가 적용되는 파이프라인을 지정합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

추가 예

다음 추가 예제에서는 우선 순위가 높은 로그 레코드만 유지하도록 **drop** 필터를 구성하는 방법을 보여줍니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  filters:
  - name: important
    type: drop
    drop:
    - test:
      - field: .message
        notMatches: "(?i)critical|error"
      - field: .level
        matches: "info|warning"
# ...
```

단일 **테스트** 구성에 여러 필드 경로를 포함하는 것 외에도 *또는*검사로 처리되는 추가 테스트를 포함할 수도 있습니다. 다음 예에서는 **테스트** 구성이 true로 평가되면 레코드가 삭제됩니다. 그러나 두 번째 **테스트** 구성의 경우 두 필드 사양을 모두 true로 평가하려면 모두 true여야 합니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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"
# ...
```

1.4.4.7. API 감사 필터 개요

OpenShift API 서버는 각 API 호출에 대한 감사 이벤트를 생성하여 요청자의 요청, 응답 및 ID를 자세히 설명하여 대량의 데이터를 생성합니다. API 감사 필터는 규칙을 사용하여 필수가 아닌 이벤트를 제외하고 이벤트 크기 감소를 활성화하여 보다 관리가 용이한 감사 추적을 지원합니다. 규칙은 순서대로 확인되며 첫 번째 일치 시에 확인 중지됩니다. 이벤트에 포함된 데이터 양은 **수준** 필드의 값에 따라 결정됩니다.

- **없음**: 이벤트가 삭제되었습니다.
- **metadata**: 감사 메타데이터가 포함되어 요청 및 응답 본문이 제거됩니다.
- **요청**: 감사 메타데이터와 요청 본문이 포함되어 응답 본문이 제거됩니다.

- **RequestResponse**: 메타데이터, 요청 본문, 응답 본문 등 모든 데이터가 포함됩니다. 응답 본문은 매우 커질 수 있습니다. 예를 들어 **oc get pods -A** 는 클러스터의 모든 포드에 대한 YAML 설명이 포함된 응답 본문을 생성합니다.

ClusterLogForwarder CR(사용자 정의 리소스)은 다음과 같은 추가 기능을 제공하는 동안 표준 Kubernetes 감사 정책과 동일한 형식을 사용합니다.

와일드카드

사용자, 그룹, 네임스페이스 및 리소스의 이름에는 선행 또는 후행 * 별표 문자가 있을 수 있습니다. 예를 들어 **openshift-*** 네임스페이스는 **openshift-apiserver** 또는 **openshift-authentication** 과 일치합니다. 리소스 ***/status** 는 **Pod/status** 또는 **Deployment/status** 와 일치합니다.

기본 규칙

정책의 규칙과 일치하지 않는 이벤트는 다음과 같이 필터링됩니다.

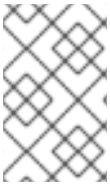
- **get**,**list** 및 **watch** 와 같은 읽기 전용 시스템 이벤트가 삭제됩니다.
- 서비스 계정은 서비스 계정과 동일한 네임스페이스 내에서 발생하는 이벤트를 기록합니다.
- 기타 모든 이벤트는 구성된 속도 제한에 따라 전달됩니다.

이러한 기본값을 비활성화하려면 **수준** 필드만 있는 규칙으로 규칙 목록을 종료하거나 빈 규칙을 추가합니다.

응답 코드 생략

생략할 정수 상태 코드 목록입니다. 이벤트가 생성되지 않은 HTTP 상태 코드를 나열하는 **OmitResponseCodes** 필드를 사용하여 응답에서 HTTP 상태 코드를 기반으로 이벤트를 삭제할 수 있습니다. 기본값은 **[404, 409, 422, 429]** 입니다. 값이 빈 목록 **[]** 인 경우 상태 코드는 생략되지 않습니다.

ClusterLogForwarder CR 감사 정책은 OpenShift Container Platform 감사 정책 외에도 작동합니다. **ClusterLogForwarder** CR 감사 필터는 로그 수집기가 전달하는 내용을 변경하고 동사, 사용자, 그룹, 네임스페이스 또는 리소스별로 필터링하는 기능을 제공합니다. 여러 필터를 생성하여 동일한 감사 스트림의 다른 요약을 다른 위치로 보낼 수 있습니다. 예를 들어 자세한 스트림을 로컬 클러스터 로그 저장소로 전송하고 더 자세한 스트림을 원격 사이트로 보낼 수 있습니다.



참고

감사 로그를 수집하려면 클러스터 역할 **collect-audit-logs** 가 있어야 합니다. 제공된 다음 예제는 감사 정책에서 가능한 규칙 범위를 설명하기 위한 것이며 권장되는 구성은 아닙니다.

감사 정책의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name>
  namespace: <log_forwarder_namespace>
spec:
  serviceAccount:
    name: <service_account_name>
  pipelines:
    - name: my-pipeline
      inputRefs: audit 1
```

```
    filterRefs: my-policy (2)
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 수집되는 로그 유형입니다. 이 필드의 값은 **감사** 로그, 애플리케이션 로그의 **애플리케이션**, 인프라 로그용 **인프라** 또는 애플리케이션에 대해 정의된 이름이 지정된 입력에 대한 감사일 수 있습니다.

2 감사 정책의 이름입니다.

1.4.4.8. 레이블 표현식 또는 일치하는 라벨 키와 값을 포함하여 입력 시 애플리케이션 로그 필터링

입력 선택기를 사용하여 라벨 표현식 또는 일치하는 라벨 키와 해당 값을 기반으로 애플리케이션 로그를 포함할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR의 **입력 사양에** 필터 구성을 추가합니다.
   다음 예제에서는 라벨 표현식 또는 일치하는 라벨 키/값에 따라 로그를 포함하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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
      type: application
# ...
```

1 일치시킬 레이블 키를 지정합니다.

2 Operator를 지정합니다. 유효한 값에는 **in**,**NotIn**,**Exists** 및 **DoesNotExist** 가 있습니다.

3 문자열 값의 배열을 지정합니다. **operator** 값이 **Exists** 또는 **DoesNotExist** 이면 값 배열이 비어 있어야 합니다.

4 정확한 키 또는 값 매핑을 지정합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

1.4.4.9. 로그 레코드를 정리하도록 콘텐츠 필터 구성

정리 필터가 구성되면 로그 수집기는 전달 전에 필터에 따라 로그 스트림을 평가합니다. 수집기는 Pod 주석과 같은 낮은 값 필드를 제거하여 로그 레코드를 정리합니다.

프로세스

1. **ClusterLogForwarder** CR의 **prune** 사양에 필터 구성을 추가합니다.
   다음 예제에서는 필드 경로를 기반으로 로그 레코드를 정리하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

중요

둘 다 지정된 경우 먼저 **notIn** 배열에 따라 레코드가 정리되며, 이 경우 **in** 배열보다 우선합니다. **notIn** 배열을 사용하여 레코드를 정리하면 **in** 배열을 사용하여 정리됩니다.

ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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>"]
# ...
```

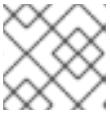
1 필터 유형을 지정합니다. **prune** 필터는 구성된 필드를 통해 로그 레코드를 정리합니다.

2 **prune** 필터를 적용하기 위한 구성 옵션을 지정합니다. **in** 및 **notIn** 필드는 로그 레코드의 필드 경로의 경로인 점으로 구분된 필드 경로의 배열로 지정됩니다. 이러한 경로에는 alpha-numeric 문자 및 밑줄(**a-zA-Z0-9_**)을 포함할 수 있습니다(예: **.kubernetes.namespace_name**). 세그먼트에 이 범위를 벗어나는 문자가 포함된 경우 세그먼트는 따옴표로 묶어야 합니다(예: **.kubernetes.labels."foo.bar-bar/baz"**.

3 선택 사항: 이 배열에 지정된 모든 필드는 로그 레코드에서 제거됩니다.

4 선택 사항: 이 배열에 지정되지 않은 모든 필드는 로그 레코드에서 제거됩니다.

5 **prune** 필터가 적용되는 파이프라인을 지정합니다.



참고

필터는 **log_type**,.**log_source**, **.message** 필드를 제외합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

1.4.5. 소스로 감사 및 인프라 로그 입력 필터링

입력 선택기를 사용하여 로그를 수집할 **감사** 및 **인프라** 소스 목록을 정의할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR에서 **감사** 및 **인프라** 소스를 정의하는 구성을 추가합니다.
   다음 예제에서는 **감사** 및 **인프라** 소스를 정의하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  inputs:
    - name: mylogs1
      type: infrastructure
      infrastructure:
        sources: 1
          - node
    - name: mylogs2
      type: audit
      audit:
        sources: 2
          - kubeAPI
          - openshiftAPI
          - ovn
# ...
```

1 수집할 인프라 소스 목록을 지정합니다. 유효한 소스는 다음과 같습니다.

- **Node**: 노드에서 저널 로그
- **컨테이너**: 네임스페이스에 배포된 워크로드의 로그

2 수집할 감사 소스 목록을 지정합니다. 유효한 소스는 다음과 같습니다.

- **kubeAPI**: Kubernetes API 서버의 로그
- **openshiftAPI**: OpenShift API 서버의 로그
- **auditd**: 노드 auditd 서비스의 로그
- **OVN**: 오픈 가상 네트워크 서비스의 로그

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

1.4.6. 네임스페이스 또는 컨테이너 이름을 포함하거나 제외하여 입력 시 애플리케이션 로그 필터링

입력 선택기를 사용하여 네임스페이스 및 컨테이너 이름을 기반으로 애플리케이션 로그를 포함하거나 제외할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR에 네임스페이스 및 컨테이너 이름을 포함하거나 제외하는 구성을 추가합니다.
   다음 예제에서는 네임스페이스 및 컨테이너 이름을 포함하거나 제외하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  inputs:
    - name: mylogs
      application:
        includes:
          - namespace: "my-project" 1
            container: "my-container" 2
        excludes:
          - container: "other-container*" 3
            namespace: "other-namespace" 4
      type: application
# ...
```

1. 이러한 네임스페이스에서만 로그를 수집하도록 지정합니다.
2. 이러한 컨테이너에서만 로그를 수집하도록 지정합니다.
3. 로그를 수집할 때 무시할 네임스페이스 패턴을 지정합니다.
4. 로그를 수집할 때 무시할 컨테이너 세트를 지정합니다.

참고

excludes 필드는 **includes** 필드보다 우선합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

1.5. LOKISTACK을 사용하여 로그 저장

애플리케이션, 감사 및 인프라 관련 로그를 저장하도록 **LokiStack** CR을 구성할 수 있습니다.

Loki는 수평으로 확장 가능한 고가용성 다중 테넌트 로그 집계 시스템으로 OpenShift Observability UI로 시각화할 수 있는 Red Hat OpenShift의 로깅을 위한 GA 로그 저장소로 제공됩니다. OpenShift Logging에서 제공하는 Loki 구성은 사용자가 수집된 로그를 사용하여 빠른 문제 해결을 수행할 수 있도록 설계된 단기 로그 저장소입니다. 이를 위해 Loki의 Red Hat OpenShift 구성에 대한 로깅은 단기 스토리지가 있으며 최근 쿼리에 최적화되어 있습니다.

중요

장기간의 기간 동안의 스토리지 또는 쿼리의 경우 사용자는 클러스터 외부에 있는 저장소를 로그하려고 합니다. Loki 크기 조정은 최대 30일 동안 단기 스토리지에서만 테스트 및 지원됩니다.

1.5.1. 사전 요구 사항

- CLI 또는 웹 콘솔을 사용하여 Loki Operator를 설치했습니다.
- **ClusterLogForwarder** 를 생성하는 동일한 네임스페이스에 **serviceAccount** 가 있습니다.
- **serviceAccount** 에는 **collect-audit-logs**,**collect-application-logs**, **collect-infrastructure-logs** 클러스터 역할이 할당됩니다.

1.5.2. 코어 설정 및 구성

Loki를 배포하기 위한 역할 기반 액세스 제어, 기본 모니터링 및 Pod 배치입니다.

1.5.3. LokiStack 규칙 RBAC 권한 승인

관리자는 클러스터 역할을 사용자 이름에 바인딩하여 사용자가 자체 경고 및 레코딩 규칙을 생성하고 관리할 수 있습니다. 클러스터 역할은 사용자에게 필요한 RBAC(역할 기반 액세스 제어) 권한이 포함된 **ClusterRole** 오브젝트로 정의됩니다.

경고 및 레코딩 규칙에 대한 다음 클러스터 역할은 LokiStack에 사용할 수 있습니다.

규칙 이름	설명
alertingrules.loki.grafana.com-v1-admin	이 역할의 사용자에게는 경고 규칙을 관리할 수 있는 관리 수준 액세스 권한이 있습니다. 이 클러스터 역할은 **loki.grafana.com/v1** API 그룹 내에서 **AlertingRule** 리소스를 생성, 읽기, 업데이트, 삭제, 나열 및 조사할 수 있는 권한을 부여합니다.
alertingrules.loki.grafana.com-v1-crdview	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내의 **AlertingRule** 리소스와 관련된 CRD(Custom Resource Definitions)의 정의를 볼 수 있지만 이러한 리소스를 수정하거나 관리할 수 있는 권한은 없습니다.
alertingrules.loki.grafana.com-v1-edit	이 역할의 사용자는 **AlertingRule** 리소스를 생성, 업데이트 및 삭제할 수 있는 권한이 있습니다.
alertingrules.loki.grafana.com-v1-view	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내에서 **AlertingRule** 리소스를 읽을 수 있습니다. 기존 경고 규칙에 대한 구성, 라벨 및 주석을 검사할 수 있지만 수정할 수는 없습니다.
recordingrules.loki.grafana.com-v1-admin	이 역할의 사용자는 레코딩 규칙을 관리할 수 있는 관리 수준의 액세스 권한이 있습니다. 이 클러스터 역할은 **loki.grafana.com/v1** API 그룹 내의 **RecordingRule** 리소스를 생성, 읽기, 업데이트, 삭제, 나열 및 감시할 수 있는 권한을 부여합니다.
recordingrules.loki.grafana.com-v1-crdview	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내의 **RecordingRule** 리소스와 관련된 CRD(Custom Resource Definitions)의 정의를 볼 수 있지만 이러한 리소스를 수정하거나 관리할 수 있는 권한은 없습니다.
recordingrules.loki.grafana.com-v1-edit	이 역할의 사용자는 **RecordingRule** 리소스를 생성, 업데이트 및 삭제할 수 있는 권한이 있습니다.
recordingrules.loki.grafana.com-v1-view	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내에서 **RecordingRule** 리소스를 읽을 수 있습니다. 기존 경고 규칙에 대한 구성, 라벨 및 주석을 검사할 수 있지만 수정할 수는 없습니다.

1.5.3.1. 예

사용자의 클러스터 역할을 적용하려면 기존 클러스터 역할을 특정 사용자 이름에 바인딩해야 합니다.

클러스터 역할은 사용하는 역할 바인딩 유형에 따라 클러스터 또는 네임스페이스 범위일 수 있습니다. **RoleBinding** 오브젝트가 사용되는 경우 **oc adm policy add-role-to-user** 명령을 사용하는 경우 클러스터 역할은 지정된 네임스페이스에만 적용됩니다. **oc adm policy add-cluster-role-to-user** 명령을 사용할 때 **ClusterRoleBinding** 오브젝트가 사용되는 경우 클러스터 역할은 클러스터의 모든 네임스페이스에 적용됩니다.

다음 예제 명령은 클러스터의 특정 네임스페이스의 경고 규칙에 대해 지정된 사용자 생성, 읽기, 업데이트 및 삭제(CRUD) 권한을 제공합니다.

특정 네임스페이스에서 경고 규칙 **CRUD** 권한에 대한 클러스터 역할 바인딩 명령의 예

```
$ oc adm policy add-role-to-user alertingrules.loki.grafana.com-v1-admin -n <namespace>
<username>
```

다음 명령은 모든 네임스페이스의 경고 규칙에 대해 지정된 사용자 관리자 권한을 부여합니다.

관리자 권한에 대한 클러스터 역할 바인딩 명령의 예

```
$ oc adm policy add-cluster-role-to-user alertingrules.loki.grafana.com-v1-admin <username>
```

1.5.4. Loki를 사용하여 로그 기반 경고 규칙 생성

AlertingRule CR에는 단일 **LokiStack** 인스턴스에 대한 경고 규칙 그룹을 선언하는 일련의 사양 및 Webhook 검증 정의가 포함되어 있습니다. 또한 웹 후크 검증 정의에서는 규칙 검증 조건을 지원합니다.

- **AlertingRule** CR에 잘못된 **간격** 기간이 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR **에 기간 동안** 유효하지 않은 항목이 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR에 잘못된 LogQL **expr** 가 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR에 동일한 이름의 두 개의 그룹이 포함된 경우 잘못된 경고 규칙입니다.
- 위의 어느 것도 적용되지 않으면 경고 규칙이 유효한 것으로 간주됩니다.

표 1.2. AlertingRule 정의

테넌트 유형	AlertingRule CR을 위한 유효한 네임스페이스
애플리케이션	**<your_application_namespace>**
audit	**openshift-logging**
인프라	**openshift-/*, kube-/*, default**

프로세스

1. **AlertingRule** 사용자 정의 리소스(CR)를 생성합니다.

   인프라 **AlertingRule CR**의 예

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: loki-operator-alerts
  namespace: openshift-operators-redhat (1)
  labels: (2)
    openshift.io/<label_name>: "true"
spec:
  tenantID: "infrastructure" (3)
  groups:
```

```
    - name: LokiOperatorHighReconciliationError
      rules:
        - alert: HighPercentageError
          expr: | (4)
            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 (5)
          annotations:
            summary: High Loki Operator Reconciliation Errors (6)
            description: High Loki Operator Reconciliation Errors (7)
```

(1) 이 **AlertingRule** CR이 생성되는 네임스페이스에는 LokiStack **spec.rules.namespaceSelector** 정의와 일치하는 레이블이 있어야 합니다.

(2) **labels** 블록은 LokiStack **spec.rules.selector** 정의와 일치해야 합니다.

(3) **인프라** 테넌트에 대한 **AlertingRule** CR은 **openshift-***, **kube-*** 또는 **default** 네임스페이스에서만 지원됩니다.

(4) **kubernetes_namespace_name:** 의 값은 **metadata.namespace** 값과 일치해야 합니다.

(5) 이 필수 필드의 값은 **중요,경고** 또는 **정보** 여야 합니다.

(6) 이 필드는 필수입니다.

(7) 이 필드는 필수입니다.

애플리케이션 **AlertingRule CR**의 예

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: app-user-workload
  namespace: app-ns (1)
  labels: (2)
    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 이 **AlertingRule** CR이 생성되는 네임스페이스에는 LokiStack **spec.rules.namespaceSelector** 정의와 일치하는 레이블이 있어야 합니다.

2 **labels** 블록은 LokiStack **spec.rules.selector** 정의와 일치해야 합니다.

3 **kubernetes_namespace_name:** 의 값은 **metadata.namespace** 값과 일치해야 합니다.

4 이 필수 필드의 값은 **중요**,**경고** 또는 **정보** 여야 합니다.

5 이 필수 필드의 값은 규칙에 대한 요약입니다.

6 이 필수 필드의 값은 규칙에 대한 자세한 설명입니다.

2. **AlertingRule** CR을 적용합니다.

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

1.5.5. 멤버 목록 생성 실패를 허용하도록 Loki 구성

OpenShift Container Platform 클러스터에서 관리자는 일반적으로 개인 IP 네트워크 범위를 사용합니다. 결과적으로 LokiStack 멤버 목록 구성은 기본적으로 개인 IP 네트워크만 사용하므로 실패합니다.

관리자는 memberlist 구성에 대한 Pod 네트워크를 선택할 수 있습니다. **hashRing** 사양에서 **podIP** 주소를 사용하도록 **LokiStack** CR(사용자 정의 리소스)을 수정할 수 있습니다. **LokiStack** CR을 구성하려면 다음 명령을 사용합니다.

```
$ oc patch LokiStack logging-loki -n openshift-logging  --type=merge -p '{"spec": {"hashRing":
{"memberlist":{"instanceAddrType":"podIP"},"type":"memberlist"}}}'
```

podIP를 포함하는 **LokiStack**의 예

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  hashRing:
    type: memberlist
    memberlist:
      instanceAddrType: podIP
# ...
```

1.5.6. Loki를 사용하여 스트림 기반 보존 활성화

로그 스트림을 기반으로 보존 정책을 구성할 수 있습니다. 이러한 규칙에 대한 규칙은 전역적으로, 테넌트별로 또는 둘 다 설정할 수 있습니다. 둘 다 구성하는 경우 테넌트 규칙이 글로벌 규칙 앞에 적용됩니다.

중요

s3 버킷 또는 LokiStack 사용자 정의 리소스(CR)에 정의된 보존 기간이 없으면 로그가 정리되지 않고 s3 스토리지를 채울 수 있습니다.

참고

스키마 v13이 권장됩니다.

프로세스

1. **LokiStack** CR을 생성합니다.

- 다음 예와 같이 스트림 기반 보존을 전역적으로 활성화합니다.

  AWS에 대한 글로벌 스트림 기반 보존의 예

```
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: v13
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

(1) 모든 로그 스트림에 대한 보존 정책을 설정합니다. **참고: 이 필드는 오브젝트 스토리지에 저장된 로그의 보존 기간에는 영향을 미치지 않습니다.**

(2) 이 블록이 CR에 추가되면 클러스터에서 보존이 활성화됩니다.

(3) 로그 스트림.spec: 제한을 정의하는 데 사용되는 LogQL 쿼리 를 포함합니다.

- 다음 예와 같이 테넌트별로 스트림 기반 보존을 활성화합니다.

  AWS에 대한 테넌트별 스트림 기반 보존 예

```
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: v13
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

1 테넌트별 보존 정책을 설정합니다. 유효한 테넌트 유형은 **애플리케이션**,**감사** 및 **인프라** 입니다.

2 로그 스트림을 정의하는 데 사용되는 LogQL 쿼리 를 포함합니다.

2. **LokiStack** CR을 적용합니다.

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

1.5.7. Loki Pod 배치

Pod의 허용 오차 또는 노드 선택기를 사용하여 Loki Pod가 실행되는 노드를 제어하고 다른 워크로드가 해당 노드를 사용하지 못하도록 할 수 있습니다.

LokiStack CR(사용자 정의 리소스)을 사용하여 로그 저장소 Pod에 허용 오차를 적용하고 노드 사양이 있는 노드에 taint를 적용할 수 있습니다. 노드의 테인트는 테인트를 허용하지 않는 모든 Pod를 거절하도록 노드에 지시하는 **키:값** 쌍입니다. 다른 Pod에 없는 특정 **키:값** 쌍을 사용하면 해당 노드에서 로그 저장소 Pod만 실행할 수 있습니다.

노드 선택기가 있는 **LokiStack**의 예

```
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 노드 선택기에 적용되는 구성 요소 Pod 유형을 지정합니다.

2 정의된 라벨이 포함된 노드로 이동되는 Pod를 지정합니다.

노드 선택기 및 허용 오차가 있는 **LokiStack CR**의 예

```
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)의 **nodeSelector** 및 **허용 오차** 필드를 구성하려면 **oc explain** 명령을 사용하여 특정 리소스에 대한 설명 및 필드를 볼 수 있습니다.

```
$ oc explain lokistack.spec.template
```

출력 예

```
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.
...
```

자세한 내용은 특정 필드를 추가할 수 있습니다.

```
$ oc explain lokistack.spec.template.compactor
```

출력 예

```
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.
...
```

1.5.7.1. 향상된 신뢰성 및 성능

프로덕션 환경에서 Loki의 안정성 및 효율성을 보장하는 구성입니다.

1.5.7.2. 수명이 짧은 토큰을 사용하여 클라우드 기반 로그 저장소에 대한 인증 활성화

워크로드 ID 페더레이션을 통해 단기 토큰을 사용하여 클라우드 기반 로그 저장소에 인증할 수 있습니다.

프로세스

- 다음 옵션 중 하나를 사용하여 인증을 활성화합니다.
  - OpenShift Container Platform 웹 콘솔을 사용하여 Loki Operator를 설치하면 수명이 짧은 토큰을 사용하는 클러스터가 자동으로 감지됩니다. 역할을 생성하고 Loki Operator에서 보안을 채우는 **CredentialsRequest** 오브젝트를 생성하는 데 필요한 데이터를 제공하라는 메시지가 표시됩니다.
  - OpenShift CLI(**oc**)를 사용하여 Loki Operator를 설치하는 경우 다음 예와 같이 스토리지 공급자에 적절한 템플릿을 사용하여 **Subscription** 오브젝트를 수동으로 생성해야 합니다. 이 인증 전략은 표시된 스토리지 공급자에 대해서만 지원됩니다.

    Azure 샘플 서브스크립션의 예

    ```
    apiVersion: operators.coreos.com/v1alpha1
    kind: Subscription
    metadata:
      name: loki-operator
      namespace: openshift-operators-redhat
    spec:
    ```

```
  channel: "stable-6.0"
  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 샘플 서브스크립션 예

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-6.0"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: ROLEARN
      value: <role_ARN>
```

1.5.7.3. 노드 장애를 허용하도록 Loki 구성

Loki Operator는 Pod 유사성 방지 규칙을 설정하여 동일한 구성 요소의 Pod가 클러스터의 다른 사용 가능한 노드에 예약되도록 요청할 수 있습니다.

유사성은 예약할 노드를 제어하는 Pod의 속성입니다. 유사성 방지는 Pod가 노드에서 예약되지 않도록 하는 Pod의 속성입니다.

OpenShift Container Platform에서 *Pod 유사성* 및 *Pod 유사성 방지*를 사용하면 다른 Pod의 키 값 라벨에 따라 Pod를 예약할 수 있는 노드를 제한할 수 있습니다.

Operator는 **compactor**,**distributor**,**gateway**,**indexGateway**,**ingester**,**querier**,**queryFrontend** 및 **ruler** 구성 요소를 포함하는 모든 Loki 구성 요소에 대해 기본 기본 **podAntiAffinity** 규칙을 설정합니다.

requiredDuringSchedulingIgnoredDuringExecution 필드에서 필요한 설정을 구성하여 Loki 구성 요소의 기본 **podAntiAffinity** 설정을 덮어쓸 수 있습니다.

ingester 구성 요소에 대한 사용자 설정 예

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
```

```
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    ingester:
      podAntiAffinity:
      # ...
        requiredDuringSchedulingIgnoredDuringExecution: 1
        - labelSelector:
            matchLabels: 2
              app.kubernetes.io/component: ingester
          topologyKey: kubernetes.io/hostname
# ...
```

1 필수 규칙을 정의하는 스탠자입니다.

2 규칙을 적용하려면 일치해야 하는 키-값 쌍(레이블)입니다.

1.5.7.4. 클러스터를 다시 시작하는 동안 LokiStack 동작

OpenShift Container Platform 클러스터가 다시 시작되면 LokiStack 수집 및 쿼리 경로가 노드에 사용 가능한 사용 가능한 CPU 및 메모리 리소스 내에서 계속 작동합니다. 즉, OpenShift Container Platform 클러스터 업데이트 중에 LokiStack에 대한 다운 타임이 없습니다. 이 동작은 **PodDisruptionBudget** 리소스를 사용하여 수행됩니다. Loki Operator는 특정 조건에서 정상적인 작업을 보장하기 위해 구성 요소별로 사용 가능한 최소 Pod 수를 결정하는 Loki의 **PodDisruptionBudget** 리소스를 프로비저닝합니다.

1.5.7.5. 고급 배포 및 확장성

고가용성, 확장성 및 오류 처리를 위한 특수 구성입니다.

1.5.7.6. 영역 인식 데이터 복제

Loki Operator는 Pod 토폴로지 분배 제약 조건을 통해 영역 인식 데이터 복제를 지원합니다. 이 기능을 사용하면 단일 영역 장애가 발생할 경우 로그 손실에 대한 안정성과 보호 장치가 향상됩니다. 배포 크기를 **1x.extra-** windows , **1x.** ngressController 또는 **1x.medium** 으로 구성하는 경우 **replication.factor** 필드가 자동으로 2로 설정됩니다.

적절한 복제를 위해서는 복제 요인에서 지정한 만큼 이상의 가용성 영역이 있어야 합니다. 복제 요인보다 가용성 영역을 더 많이 보유할 수는 있지만 영역이 줄어들면 오류를 작성할 수 있습니다. 각 영역은 최적의 작업을 위해 동일한 수의 인스턴스를 호스팅해야 합니다.

영역 복제가 활성화된 **LokiStack CR**의 예

```
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. 더 이상 사용되지 않는 필드로 입력된 값은 **replication.factor** 로 덮어씁니다.
2. 이 값은 배포 크기가 설정 시 선택되면 자동으로 설정됩니다.
3. 두 토폴로지 도메인 간 최대 Pod 수 차이입니다. 기본값은 1이며 값 0을 지정할 수 없습니다.
4. 노드 레이블에 해당하는 토폴로지 키 형태로 영역을 정의합니다.

1.5.7.7. 실패한 영역에서 Loki Pod 복구

OpenShift Container Platform에서 특정 가용성 영역 리소스에 액세스할 수 없게 되면 영역 오류가 발생합니다. 가용성 영역은 클라우드 공급자의 데이터 센터 내의 격리된 영역이며 중복성과 내결함성을 강화하기 위한 것입니다. OpenShift Container Platform 클러스터가 이를 처리하도록 구성되지 않은 경우 영역 실패로 인해 서비스 또는 데이터가 손실될 수 있습니다.

Loki 포드는 StatefulSet 의 일부이며 **StorageClass** 오브젝트에서 프로비저닝한 PVC(영구 볼륨 클레임)와 함께 제공됩니다. 각 Loki Pod 및 해당 PVC는 동일한 영역에 있습니다. 클러스터에서 영역 오류가 발생하면 StatefulSet 컨트롤러에서 실패한 영역에서 영향을 받는 Pod를 자동으로 복구하려고 합니다.

> 주의
>
> 다음 절차에서는 실패한 영역의 PVC와 그 안에 포함된 모든 데이터를 삭제합니다. 완전한 데이터 손실을 방지하려면 **LokiStack** CR의 복제 요소 필드를 항상 1보다 큰 값으로 설정하여 Loki가 복제되도록 해야 합니다.

사전 요구 사항

- **LokiStack** CR에 1보다 큰 복제 인수가 있는지 확인합니다.
- 컨트롤 플레인에서 감지한 영역 장애와 실패한 영역의 노드는 클라우드 공급자 통합으로 표시됩니다.

StatefulSet 컨트롤러는 실패한 영역에서 Pod 일정 변경을 자동으로 시도합니다. 연결된 PVC도 실패한 영역에 있기 때문에 다른 영역으로 자동 일정 조정이 작동하지 않습니다. 새 영역에서 상태 저장 Loki Pod와 프로비저닝된 PVC를 다시 생성할 수 있도록 실패한 영역에서 PVC를 수동으로 삭제해야 합니다.

프로세스

1. 다음 명령을 실행하여 **Pending** 상태의 Pod를 나열합니다.

```
$ oc get pods --field-selector status.phase==Pending -n openshift-logging
```

oc get pods 출력 예

```
NAME                          READY   STATUS    RESTARTS   AGE (1)
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) 해당 PVC가 실패한 영역에 있기 때문에 이러한 Pod는 **Pending** 상태입니다.

2. 다음 명령을 실행하여 **Pending** 상태의 PVC를 나열합니다.

```
$ oc get pvc -o=json -n openshift-logging | jq '.items[] | select(.status.phase == "Pending") | .metadata.name' -r
```

oc get pvc 출력 예

```
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. 다음 명령을 실행하여 Pod의 PVC를 삭제합니다.

```
$ oc delete pvc <pvc_name>  -n openshift-logging
```

4. 다음 명령을 실행하여 Pod를 삭제합니다.

```
$ oc delete pod <pod_name>  -n openshift-logging
```

이러한 오브젝트가 성공적으로 삭제되면 사용 가능한 영역에서 자동으로 다시 예약해야 합니다.

1.5.7.7.1. 종료 상태의 PVC 문제 해결

PVC 메타데이터 종료자가 **kubernetes.io/pv-protection** 으로 설정된 경우 PVC는 삭제 없이 종료 상태로 중단될 수 있습니다. 종료자를 제거하면 PVC가 성공적으로 삭제될 수 있습니다.

- 아래 명령을 실행하여 각 PVC의 종료자를 제거한 다음 삭제를 다시 시도합니다.

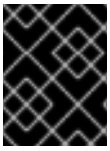
```
$ oc patch pvc <pvc_name> -p '{"metadata":{"finalizers":null}}' -n openshift-logging
```

1.5.7.8. Loki 속도 제한 오류 문제 해결

로그 전달자 API에서 속도 제한을 초과하는 대규모 메시지 블록을 Loki로 전달하면 Loki는 속도 제한(**429**) 오류를 생성합니다.

이러한 오류는 정상적인 작동 중에 발생할 수 있습니다. 예를 들어 이미 일부 로그가 있는 클러스터에 로깅을 추가할 때 로깅이 기존 로그 항목을 모두 수집하는 동안 속도 제한 오류가 발생할 수 있습니다. 이 경우 새 로그 추가 속도가 총 속도 제한보다 작으면 기록 데이터가 결국 수집되고 사용자 개입 없이도 속도 제한 오류가 해결됩니다.

속도 제한 오류가 계속 발생하는 경우 **LokiStack** CR(사용자 정의 리소스)을 수정하여 문제를 해결할 수 있습니다.



중요

LokiStack CR은 Grafana 호스팅 Loki에서 사용할 수 없습니다. 이는 Grafana 호스팅 Loki 서버에는 적용되지 않습니다.

조건

- Log Forwarder API는 로그를 Loki로 전달하도록 구성되어 있습니다.
- 시스템에서 2MB보다 큰 메시지 블록을 Loki로 보냅니다. 예를 들면 다음과 같습니다.

```
"values":[["1630410392689800468","{\"kind\":\"Event\",\"apiVersion\":\
.......
......
......
......
\"received_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\":\"MzFjYjJkZjItNjY0MC00YWU4LWIwMTEtNGNmM2E5ZmViMGU4\",\"lo
g_type\":\"audit\"}"]]}]}
```

- **oc logs -n openshift-logging -l component=collector** 를 입력하면 클러스터의 수집기 로그에 다음 오류 메시지 중 하나가 포함된 행이 표시됩니다.

```
429 Too Many Requests Ingestion rate limit exceeded
```

Vector 오류 메시지의 예

```
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 오류 메시지의 예

```
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"
```

이 오류는 수신 끝점에도 표시됩니다. 예를 들어 LokiStack ingester Pod에서 다음을 수행합니다.

Loki ingester 오류 메시지의 예

```
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
```

프로세스

- **LokiStack** CR에서 **ingestionBurstSize** 및 **ingestionRate** 필드를 업데이트합니다.

```
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** 필드는 배포자 복제본당 최대 로컬 속도 제한 샘플 크기를 MB로 정의합니다. 이 값은 하드 제한입니다. 이 값을 단일 푸시 요청에 예상되는 최대 로그 크기로 설정합니다. **ingestionBurstSize** 값보다 큰 단일 요청은 허용되지 않습니다.

(2) **ingestionRate** 필드는 초당 수집된 샘플의 최대 양(MB)에 대한 소프트 제한입니다. 로그 비율이 제한을 초과하는 경우 속도 제한 오류가 발생하지만 수집기는 로그를 다시 시도합니다. 총 평균이 제한보다 작으면 사용자 개입 없이 시스템을 복구하고 오류가 해결됩니다.

1.6. 로깅 시각화

로깅을 위한 시각화는 Operator 설치가 필요한 Cluster Observability Operator 의 로깅 UI 플러그인 을 배포하여 제공됩니다.

중요

Red Hat은 현재 기술 프리뷰 (TP)에 있는 Cluster Observability Operator (COO)의 GA(General Availability) 릴리스까지 OpenShift Container Platform 4.14 이상에서 Logging UI 플러그인에 대해 Logging 6.0 이상을 사용하는 고객에게 지원을 제공합니다. COO에는 여전히 TP 기능이지만 로깅 UI 플러그인은 GA에 사용할 준비가 된 몇 가지 독립적인 기능이 포함되어 있기 때문에 이러한 지원 예외는 일시적입니다.

2장. 로깅 6.1

2.1. 로깅 6.1

2.1.1. 로깅 6.1.0 릴리스 노트

이 릴리스에는 Red Hat OpenShift 버그 수정 릴리스 6.1.0용 로깅 이 포함되어 있습니다.

2.1.1.1. 새로운 기능 및 기능 개선

2.1.1.1.1. 로그 컬렉션

- 이번 개선된 기능에는 수집된 컨테이너 로그에서 전송된 속성에 소스 **iostream** 이 추가되었습니다. 값은 수집기에서 수신한 방법에 따라 **stdout** 또는 **stderr** 로 설정됩니다. (LOG-5292)
- 이번 업데이트를 통해 수집기의 기본 메모리 제한이 1024 Mi에서 2048 Mi로 증가합니다. 사용자는 클러스터의 특정 요구 및 사양에 따라 리소스 제한을 조정해야 합니다. (LOG-6072)
- 이번 업데이트를 통해 이제 **ClusterLogForwarder** CR의 syslog 출력 전달 모드를 **AtLeastOnce** 또는 **At CryostatOnce로 설정할 수 있습니다.** (LOG-6355)

2.1.1.1.2. 로그 스토리지

- 이번 업데이트를 통해 새로운 **1x.pico** LokiStack 크기는 워크로드 수와 낮은 로그 볼륨(최대 50GB/일)이 있는 클러스터를 지원합니다. (LOG-5939)

2.1.1.2. 기술 프리뷰

중요

OTLP(OpenTelemetry Protocol) 출력 로그 전달자는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

- 이번 업데이트를 통해 **OTel** (OpenTelemetry) 데이터 모델을 사용하여 Red Hat Managed LokiStack 인스턴스로 OpenTelemetry 로그를 전달할 수 있습니다. 이 기능을 활성화하려면 **observability.openshift.io/tech-preview-otlp-output: "enabled"** 주석을 **ClusterLogForwarder** 구성에 추가합니다. 추가 구성 정보는 OTLP 전달을 참조하십시오.
- 이번 업데이트를 통해 **lokiStack** 출력 사양에 **dataModel** 필드가 추가되었습니다. OpenTelemetry 데이터 형식을 사용하여 로그 전달을 구성하려면 **dataModel** 을 **Otel** 으로 설정합니다. 기본값은 **Viaq** 로 설정됩니다. 데이터 매핑에 대한 자세한 내용은 OTLP 사양 을 참조하십시오.

2.1.1.3. 버그 수정

없음.

2.1.1.4. CVE

- CVE-2024-6119
- CVE-2024-6232

2.2. 로깅 6.1

context: logging-6x-6.1

ClusterLogForwarder CR(사용자 정의 리소스)은 로그 수집 및 전달의 중앙 구성 지점입니다.

2.2.1. 입력 및 출력

입력은 전달할 로그 소스를 지정합니다. 로깅은 클러스터의 다른 부분에서 로그를 선택하는 **애플리케이션**,**수신자**,**인프라** 및 **감사** 와 같은 기본 입력 유형을 제공합니다. 네임스페이스 또는 Pod 레이블을 기반으로 사용자 정의 입력을 정의하여 로그 선택을 미세 조정할 수도 있습니다.

출력은 로그가 전송되는 대상을 정의합니다. 각 출력 유형에는 고유한 구성 옵션 세트가 있어 동작 및 인증 설정을 사용자 지정할 수 있습니다.

2.2.2. 수신자 입력 유형

수신자 입력 유형을 사용하면 로깅 시스템에서 외부 소스의 로그를 허용할 수 있습니다. 로그를 수신하기 위해 **http** 및 **syslog** 의 두 형식을 지원합니다.

ReceiverSpec 은 수신자 입력에 대한 구성을 정의합니다.

2.2.3. 파이프라인 및 필터

파이프라인은 입력에서 출력으로 로그 흐름을 결정합니다. 파이프라인은 하나 이상의 입력 참조, 출력 참조 및 선택적 필터 참조로 구성됩니다. 필터를 사용하여 파이프라인 내에서 로그 메시지를 변환하거나 삭제할 수 있습니다. 필터 순서는 순차적으로 적용되므로 중요하며 이전 필터로 인해 로그 메시지가 이후 단계에 도달하지 못할 수 있습니다.

2.2.4. Operator 동작

Cluster Logging Operator는 **ClusterLogForwarder** 리소스의 **managementState** 필드를 기반으로 수집기의 배포 및 구성을 관리합니다.

- **Managed** (기본값)로 설정하면 Operator에서 사양에 정의된 구성과 일치하도록 로깅 리소스를 적극적으로 관리합니다.
- **Unmanaged** 로 설정하면 Operator에서 작업을 수행하지 않으므로 로깅 구성 요소를 수동으로 관리할 수 있습니다.

2.2.5. 검증

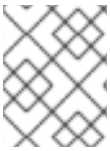
로깅에는 광범위한 검증 규칙과 기본값이 포함되어 있어 원활하고 오류가 없는 구성 환경을 보장합니다. **ClusterLogForwarder** 리소스는 필수 필드, 필드 간 종속성 및 입력 값 형식에 대한 검증 검사를 적용합니다. 특정 필드에 기본값이 제공되어 일반적인 시나리오에서 명시적 구성의 필요성이 줄어듭니다.

2.2.6. 퀵 스타트

OpenShift Logging은 다음 두 가지 데이터 모델을 지원합니다.

- viaq (일반 가용성)
- OpenTelemetry (기술 프리뷰)

ClusterLogForwarder 에서 **lokiStack.dataModel** 필드를 구성하여 요구 사항에 따라 이러한 데이터 모델 중 하나를 선택할 수 있습니다. viaq는 LokiStack으로 로그를 전달할 때 기본 데이터 모델입니다.



참고

향후 OpenShift Logging 릴리스에서는 기본 데이터 모델이 ViaQ에서 OpenTelemetry로 변경됩니다.

2.2.6.1. ViaQ로 퀵 스타트

기본 ViaQ 데이터 모델을 사용하려면 다음 단계를 따르십시오.

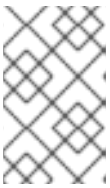
사전 요구 사항

- 클러스터 관리자 권한

프로세스

1. OperatorHub에서 Red Hat OpenShift Logging Operator, Loki Operator 및 Cluster Observability Operator(COO)를 설치합니다.
2. **openshift-logging** 네임스페이스에서 **LokiStack** CR(사용자 정의 리소스)을 생성합니다.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  managementState: Managed
  size: 1x.extra-small
  storage:
    schemas:
    - effectiveDate: '2024-10-01'
      version: v13
    secret:
      name: logging-loki-s3
      type: s3
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```



참고

logging-loki-s3 시크릿이 사전에 생성되었는지 확인합니다. 이 보안의 내용은 사용 중인 오브젝트 스토리지에 따라 다릅니다. 자세한 내용은 시크릿 및 TLS 구성을 참조하십시오.

3. 수집기의 서비스 계정을 생성합니다.

```
$ oc create sa collector -n openshift-logging
```

4. 수집기의 서비스 계정에서 **LokiStack** CR에 데이터를 쓸 수 있도록 허용합니다.

```
$ oc adm policy add-cluster-role-to-user logging-collector-logs-writer -z collector
```

참고

ClusterRole 리소스는 Cluster Logging Operator 설치 중에 자동으로 생성되며 수동으로 생성할 필요가 없습니다.

5. 수집기의 서비스 계정에서 로그를 수집할 수 있도록 허용합니다.

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

```
$ oc adm policy add-cluster-role-to-user collect-application-logs -z collector
```

```
$ oc adm policy add-cluster-role-to-user collect-audit-logs -z collector
```

```
$ oc adm policy add-cluster-role-to-user collect-infrastructure-logs -z collector
```

참고

이 예제에서는 수집기를 세 가지 역할(애플리케이션, 인프라 및 감사) 모두에 바인딩하지만 기본적으로 애플리케이션 및 인프라 로그만 수집됩니다. 감사 로그를 수집하려면 이를 포함하도록 **ClusterLogForwarder** 구성을 업데이트합니다. 환경에 필요한 특정 로그 유형에 따라 역할을 할당합니다.

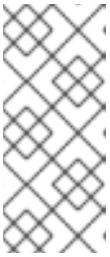
6. **UIPlugin** CR을 생성하여 **모니터링** 탭의 **Log** 섹션을 활성화합니다.

```
apiVersion: observability.openshift.io/v1alpha1
kind: UIPlugin
metadata:
  name: logging
spec:
  type: Logging
  logging:
    lokiStack:
      name: logging-loki
```

7. **ClusterLogForwarder** CR을 생성하여 로그 전달을 구성합니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  namespace: openshift-logging
spec:
  serviceAccount:
```

```
  name: collector
outputs:
- name: default-lokistack
  type: lokiStack
  lokiStack:
    authentication:
      token:
        from: serviceAccount
    target:
      name: logging-loki
      namespace: openshift-logging
  tls:
    ca:
      key: service-ca.crt
      configMapName: openshift-service-ca.crt
pipelines:
- name: default-logstore
  inputRefs:
  - application
  - infrastructure
  outputRefs:
  - default-lokistack
```



참고

dataModel 필드는 선택 사항이며 설정되지 않은(**dataModel: ""**) 기본적으로 유지됩니다. 이를 통해 Cluster Logging Operator(CLO)가 데이터 모델을 자동으로 선택할 수 있습니다. 현재 CLO는 필드가 설정되지 않은 경우 기본적으로 ViaQ 모델로 설정되지만 향후 릴리스에서는 변경될 예정입니다. **dataModel: ViaQ** 를 지정하면 기본 변경 사항이 있는 경우 구성이 계속 호환됩니다.

검증

- OpenShift 웹 콘솔의 **Observe** 탭의 **Log** 섹션에 로그가 표시되는지 확인합니다.

2.2.6.2. OpenTelemetry로 퀵 스타트

중요

OTLP(OpenTelemetry Protocol) 출력 로그 전달자는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

OTLP ingestion을 구성하고 OpenTelemetry 데이터 모델을 활성화하려면 다음 단계를 따르십시오.

사전 요구 사항

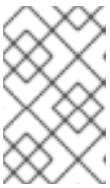
- 클러스터 관리자 권한

프로세스

1. OperatorHub에서 Red Hat OpenShift Logging Operator, Loki Operator 및 Cluster Observability Operator(COO)를 설치합니다.

2. **openshift-logging** 네임스페이스에서 **LokiStack** CR(사용자 정의 리소스)을 생성합니다.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  managementState: Managed
  size: 1x.extra-small
  storage:
    schemas:
    - effectiveDate: '2024-10-01'
      version: v13
    secret:
      name: logging-loki-s3
      type: s3
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```



참고

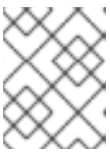
logging-loki-s3 시크릿이 사전에 생성되었는지 확인합니다. 이 보안의 내용은 사용 중인 오브젝트 스토리지에 따라 다릅니다. 자세한 내용은 "시크릿 및 TLS 구성"을 참조하십시오.

3. 수집기의 서비스 계정을 생성합니다.

```
$ oc create sa collector -n openshift-logging
```

4. 수집기의 서비스 계정에서 **LokiStack** CR에 데이터를 쓸 수 있도록 허용합니다.

```
$ oc adm policy add-cluster-role-to-user logging-collector-logs-writer -z collector
```



참고

ClusterRole 리소스는 Cluster Logging Operator 설치 중에 자동으로 생성되며 수동으로 생성할 필요가 없습니다.

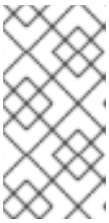
5. 수집기의 서비스 계정에서 로그를 수집할 수 있도록 허용합니다.

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

```
$ oc adm policy add-cluster-role-to-user collect-application-logs -z collector
```

```
$ oc adm policy add-cluster-role-to-user collect-audit-logs -z collector
```

```
$ oc adm policy add-cluster-role-to-user collect-infrastructure-logs -z collector
```



참고

이 예제에서는 수집기를 세 가지 역할(애플리케이션, 인프라 및 감사) 모두에 바인딩합니다. 기본적으로 애플리케이션 및 인프라 로그만 수집됩니다. 감사 로그를 수집하려면 이를 포함하도록 **ClusterLogForwarder** 구성을 업데이트합니다. 환경에 필요한 특정 로그 유형에 따라 역할을 할당합니다.

6. **UIPlugin** CR을 생성하여 **모니터링** 탭의 **Log** 섹션을 활성화합니다.

```
apiVersion: observability.openshift.io/v1alpha1
kind: UIPlugin
metadata:
  name: logging
spec:
  type: Logging
  logging:
    lokiStack:
      name: logging-loki
```

7. **ClusterLogForwarder** CR을 생성하여 로그 전달을 구성합니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  namespace: openshift-logging
  annotations:
    observability.openshift.io/tech-preview-otlp-output: "enabled" 1
spec:
  serviceAccount:
    name: collector
  outputs:
  - name: loki-otlp
    type: lokiStack 2
    lokiStack:
      target:
        name: logging-loki
        namespace: openshift-logging
      dataModel: Otel 3
      authentication:
        token:
          from: serviceAccount
    tls:
      ca:
        key: service-ca.crt
        configMapName: openshift-service-ca.crt
  pipelines:
  - name: my-pipeline
    inputRefs:
    - application
```

```
    - infrastructure
  outputRefs:
    - loki-otlp
```

1. 주석을 사용하여 기술 프리뷰 기능인 **Otel** 데이터 모델을 활성화합니다.
2. 출력 유형을 **lokiStack** 으로 정의합니다.
3. OpenTelemetry 데이터 모델을 지정합니다.

참고

dataModel 이 **Otel** 인 경우 **lokiStack.labelKeys** 를 사용할 수 없습니다. **dataModel** 이 **Otel** 일 때 유사한 기능을 얻으려면 "OTLP 데이터 수집용 LokiStack 구성"을 참조하십시오.

검증

- OpenShift 웹 콘솔에서 **Observe → OpenShift Logging → LokiStack → Writes** 로 이동하여 Cryostat **- structured Metadata** 가 올바르게 작동하는지 확인합니다.

2.3. 로그 전달 구성

ClusterLogForwarder (CLF)를 사용하면 사용자가 다양한 대상으로 로그 전달을 구성할 수 있습니다. 다른 소스의 로그 메시지를 선택하고, 변환하거나 필터링할 수 있는 파이프라인을 통해 보내고, 하나 이상의 출력으로 전달할 수 있는 유연한 방법을 제공합니다.

ClusterLogForwarder의 주요 기능

- 입력을 사용하여 로그 메시지 선택
- 출력을 사용하여 외부 대상으로 로그를 전달
- 필터를 사용하여 로그 메시지를 필터링, 변환 및 삭제
- 입력, 필터 및 출력을 연결하는 로그 전달 파이프라인을 정의합니다.

2.3.1. 로그 컬렉션 설정

이 클러스터 로깅 릴리스에서는 관리자가 **ClusterLogForwarder** 와 연결된 서비스 계정에 로그 수집 권한을 명시적으로 부여해야 합니다. **ClusterLogging** 및 **ClusterLogForwarder.logging.openshift.io** 리소스로 구성된 레거시 로깅 시나리오의 이전 릴리스에서는 이 작업이 필요하지 않았습니다.

Red Hat OpenShift Logging Operator는 **collect-audit-logs**,**collect-application-logs**, **collect-infrastructure-logs** 클러스터 역할을 제공하여 수집기가 감사 로그, 애플리케이션 로그 및 인프라 로그를 각각 수집할 수 있습니다.

필요한 클러스터 역할을 서비스 계정에 바인딩하여 로그 컬렉션을 설정합니다.

2.3.1.1. 레거시 서비스 계정

기존 서비스 계정 **logcollector** 를 사용하려면 다음 **ClusterRoleBinding** 을 생성합니다.

```
$ oc adm policy add-cluster-role-to-user collect-application-logs system:serviceaccount:openshift-logging:logcollector
$ oc adm policy add-cluster-role-to-user collect-infrastructure-logs system:serviceaccount:openshift-logging:logcollector
```

또한 감사 로그를 수집하는 경우 다음 **ClusterRoleBinding** 을 생성합니다.

```
$ oc adm policy add-cluster-role-to-user collect-audit-logs system:serviceaccount:openshift-logging:logcollector
```

2.3.1.2. 서비스 계정 생성

사전 요구 사항

- Red Hat OpenShift Logging Operator는 **openshift-logging** 네임스페이스에 설치됩니다.
- 관리자 권한이 있습니다.

프로세스

1. 수집기의 서비스 계정을 생성합니다. 인증을 위해 토큰이 필요한 스토리지에 로그를 작성하려면 서비스 계정에 토큰을 포함해야 합니다.
2. 적절한 클러스터 역할을 서비스 계정에 바인딩합니다.

   바인딩 명령 예

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

2.3.1.2.1. 서비스 계정의 클러스터 역할 바인딩

role_binding.yaml 파일은 ClusterLogging Operator의 ClusterRole을 특정 ServiceAccount에 바인딩하여 클러스터 전체에서 Kubernetes 리소스를 관리할 수 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: manager-rolebinding
roleRef:                                  1
  apiGroup: rbac.authorization.k8s.io     2
  kind: ClusterRole                       3
  name: cluster-logging-operator          4
subjects:                                 5
  - kind: ServiceAccount                  6
    name: cluster-logging-operator        7
    namespace: openshift-logging          8
```

1. roleRef: 바인딩이 적용되는 ClusterRole을 참조합니다.
2. apiGroup: RBAC API 그룹을 나타내며 ClusterRole이 Kubernetes RBAC 시스템의 일부임을 지정합니다.

3 kind: 참조된 역할이 클러스터 전체에서 적용되는 ClusterRole임을 지정합니다.

4 name: ServiceAccount에 바인딩되는 ClusterRole의 이름입니다. 여기서 cluster-logging-operator.

5 제목: ClusterRole에서 권한을 부여하는 엔터티(사용자 또는 서비스 계정)를 정의합니다.

6 kind: subject가 ServiceAccount임을 지정합니다.

7 name: 권한이 부여된 ServiceAccount의 이름입니다.

8 namespace: ServiceAccount가 있는 네임스페이스를 나타냅니다.

2.3.1.2.2. 애플리케이션 로그 작성

write-application-logs-clusterrole.yaml 파일은 Loki 로깅 애플리케이션에 애플리케이션 로그를 작성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-application-logs
rules:                                          1
  - apiGroups:                                  2
      - loki.grafana.com                        3
    resources:                                  4
      - application                             5
    resourceNames:                              6
      - logs                                    7
    verbs:                                      8
      - create                                  9
Annotations
<1> rules: Specifies the permissions granted by this ClusterRole.
<2> apiGroups: Refers to the API group loki.grafana.com, which relates to the Loki logging system.
<3> loki.grafana.com: The API group for managing Loki-related resources.
<4> resources: The resource type that the ClusterRole grants permission to interact with.
<5> application: Refers to the application resources within the Loki logging system.
<6> resourceNames: Specifies the names of resources that this role can manage.
<7> logs: Refers to the log resources that can be created.
<8> verbs: The actions allowed on the resources.
<9> create: Grants permission to create new logs in the Loki system.
```

2.3.1.2.3. 감사 로그 작성

write-audit-logs-clusterrole.yaml 파일은 Loki 로깅 시스템에서 감사 로그를 생성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-audit-logs
rules:                                          1
  - apiGroups:                                  2
      - loki.grafana.com                        3
```

```
  resources:                                  4
    - audit                                   5
  resourceNames:                                6
    - logs                                    7
  verbs:                                      8
    - create                                  9
```

1 1 rules: 이 ClusterRole에서 부여하는 권한을 정의합니다.

2 2 apiGroups: loki.grafana.com API 그룹을 지정합니다.

3 3 Loki.grafana.com: Loki 로깅 리소스를 담당하는 API 그룹입니다.

4 4 resources: 이 역할이 관리하는 리소스 유형을 나타냅니다(이 경우 audit).

5 5 audit: 역할이 Loki 내에서 감사 로그를 관리하도록 지정합니다.

6 6 resourceNames: 역할이 액세스할 수 있는 특정 리소스를 정의합니다.

7 7 logs: 이 역할에서 관리할 수 있는 로그를 참조합니다.

8 8 동사: 리소스에서 허용되는 작업입니다.

9 9 create: 새 감사 로그를 생성할 수 있는 권한을 부여합니다.

2.3.1.2.4. 인프라 로그 작성

write-infrastructure-logs-clusterrole.yaml 파일은 Loki 로깅 시스템에서 인프라 로그를 생성할 수 있는 권한을 부여하는 ClusterRole을 정의합니다.

샘플 **YAML**

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-logging-write-infrastructure-logs
rules:                                        1
  - apiGroups:                                2
      - loki.grafana.com                      3
    resources:                                4
      - infrastructure                        5
    resourceNames:                              6
      - logs                                  7
    verbs:                                    8
      - create                                9
```

1 rules: 이 ClusterRole에서 부여하는 권한을 지정합니다.

2 apiGroups: Loki 관련 리소스의 API 그룹을 지정합니다.

3 Loki.grafana.com: Loki 로깅 시스템을 관리하는 API 그룹입니다.

4 resources: 이 역할이 상호 작용할 수 있는 리소스 유형을 정의합니다.

5 인프라: 이 역할이 관리하는 인프라 관련 리소스를 나타냅니다.

6 resourceNames: 이 역할이 관리할 수 있는 리소스의 이름을 지정합니다.

7 로그: 인프라와 관련된 로그 리소스를 참조합니다.

8 동사: 이 역할에서 허용하는 작업입니다.

9 생성: Loki 시스템에서 인프라 로그를 생성할 수 있는 권한을 부여합니다.

2.3.1.2.5. ClusterLogForwarder 편집기 역할

clusterlogforwarder-editor-role.yaml 파일은 사용자가 OpenShift에서 ClusterLogForwarder를 관리할 수 있는 ClusterRole을 정의합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: clusterlogforwarder-editor-role
rules:                                          1
  - apiGroups:                                  2
      - observability.openshift.io              3
    resources:                                  4
      - clusterlogforwarders                    5
    verbs:                                      6
      - create                                  7
      - delete                                  8
      - get                                     9
      - list                                    10
      - patch                                   11
      - update                                  12
      - watch                                   13
```

1 rules: 이 ClusterRole에서 부여하는 권한을 지정합니다.

2 apiGroups: OpenShift 관련 API 그룹을 참조합니다.

3 obervability.openshift.io: 로깅과 같은 관찰 기능 리소스를 관리하는 API 그룹입니다.

4 resources: 이 역할이 관리할 수 있는 리소스를 지정합니다.

5 clusterlogforwarders: OpenShift의 로그 전달 리소스를 참조합니다.

6 verbs: ClusterLogForwarders에 허용되는 작업을 지정합니다.

7 생성: 새 ClusterLogForwarder를 생성할 수 있는 권한을 부여합니다.

8 delete: 기존 ClusterLogForwarder를 삭제할 수 있는 권한을 부여합니다.

9 get: 특정 ClusterLogForwarder에 대한 정보를 검색할 수 있는 권한을 부여합니다.

10 list: 모든 ClusterLogForwarder를 나열할 수 있습니다.

11 patch: ClusterLogForwarder를 부분적으로 수정할 수 있는 권한을 부여합니다.

12 update: 기존 ClusterLogForwarder를 업데이트할 수 있는 권한을 부여합니다.

13 watch: ClusterLogForwarder의 변경 사항을 모니터링할 수 있는 권한을 부여합니다.

2.3.2. 수집기에서 로그 수준 수정

수집기에서 로그 수준을 수정하려면 **observability.openshift.io/log-level** 주석을 **추적** 하도록 설정하고,**디버그**,**info**,**warn**,**error**, **off** 를 설정할 수 있습니다.

로그 수준 주석의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  annotations:
    observability.openshift.io/log-level: debug
# ...
```

2.3.3. Operator 관리

ClusterLogForwarder 리소스에는 Operator가 리소스를 적극적으로 관리하는지 또는 Unmanaged 상태로 둘지 여부를 제어하는 **managementState** 필드가 있습니다.

관리됨

(기본값) Operator는 CLF 사양의 원하는 상태와 일치하도록 로깅 리소스를 구동합니다.

Unmanaged

Operator는 로깅 구성 요소와 관련된 작업을 수행하지 않습니다.

이를 통해 관리자는 **managementState** 를 **Unmanaged** 로 설정하여 로그 전달을 일시적으로 일시 중지할 수 있습니다.

2.3.4. ClusterLogForwarder의 구조

CLF에는 다음과 같은 주요 구성 요소가 포함된 **spec** 섹션이 있습니다.

입력

전달할 로그 메시지를 선택합니다. 기본 입력 유형 **애플리케이션**,**인프라** 및 **감사** 는 클러스터의 다른 부분에서 로그를 전달합니다. 사용자 지정 입력을 정의할 수도 있습니다.

출력

로그를 전달할 대상을 정의합니다. 각 출력에는 고유한 이름과 유형별 구성이 있습니다.

파이프라인

입력에서 필터를 통해 출력으로 가져오는 경로 로그를 정의합니다. 파이프라인에는 고유한 이름이 있으며 입력, 출력 및 필터 이름 목록으로 구성됩니다.

필터

파이프라인에서 로그 메시지를 변환하거나 삭제합니다. 사용자는 특정 로그 필드와 일치하는 필터를 정의하고 메시지를 삭제하거나 수정할 수 있습니다. 필터는 파이프라인에 지정된 순서로 적용됩니다.

2.3.4.1. 입력

입력은 **spec.inputs** 의 배열에 구성됩니다. 세 가지 기본 제공 입력 유형이 있습니다.

애플리케이션

기본, openshift 또는 **kube-** 또는 **openshift -** 접두사가 있는 네임스페이스를 제외하고 모든 애플리케이션 컨테이너에서 로그를 선택합니다.

인프라

기본 및 **openshift** 네임스페이스 및 노드 로그에서 실행되는 인프라 구성 요소에서 로그를 선택합니다.

audit

auditd에서 OpenShift API 서버 감사 로그, Kubernetes API 서버 감사 로그, ovn 감사 로그 및 노드 감사 로그에서 로그를 선택합니다.

사용자는 특정 네임스페이스에서 로그를 선택하거나 Pod 레이블을 사용하는 유형 **애플리케이션** 의 사용자 정의 입력을 정의할 수 있습니다.

2.3.4.2. 출력

출력은 **spec.outputs** 아래의 배열에 구성됩니다. 각 출력에는 고유한 이름과 유형이 있어야 합니다. 지원되는 유형은 다음과 같습니다.

azureMonitor

Azure Monitor로 로그를 전달합니다.

CloudMonitor

AWS CloudMonitor로 로그를 전달합니다.

elasticsearch

로그를 외부 Elasticsearch 인스턴스로 전달합니다.

googleCloudLogging

Google Cloud Logging으로 로그를 전달합니다.

HTTP

일반 HTTP 끝점으로 로그를 전달합니다.

kafka

Kafka 브로커로 로그를 전달합니다.

Loki

로그를 Loki 로깅 백엔드로 전달합니다.

lokistack

OpenShift Container Platform 인증 통합을 사용하여 Loki 및 웹 프록시의 로깅 지원 조합으로 로그를 전달합니다. LokiStack의 프록시는 OpenShift Container Platform 인증을 사용하여 멀티 테넌시 적용

otlp

OpenTelemetry 프로토콜을 사용하여 로그를 전달합니다.

splunk

Splunk로 로그를 전달합니다.

syslog

로그를 외부 syslog 서버로 전달합니다.

각 출력 유형에는 고유한 구성 필드가 있습니다.

2.3.5. OTLP 출력 구성

클러스터 관리자는 OTLP(OpenTelemetry Protocol) 출력을 사용하여 로그를 수집하고 OTLP 수신자에 전달할 수 있습니다. OTLP 출력은 OpenTelemetry Observability 프레임워크에서 정의한 사양을 사용하여 JSON 인코딩으로 HTTP를 통해 데이터를 보냅니다.

중요

OTLP(OpenTelemetry Protocol) 출력 로그 전달자는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

프로세스

- 다음 주석을 추가하여 OTLP를 사용하여 전달을 활성화하도록 **ClusterLogForwarder** CR(사용자 정의 리소스)을 생성하거나 편집합니다.

  ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  annotations:
    observability.openshift.io/tech-preview-otlp-output: "enabled" 1
  name: clf-otlp
spec:
  serviceAccount:
    name: <service_account_name>
  outputs:
  - name: otlp
    type: otlp
    otlp:
      tuning:
        compression: gzip
        deliveryMode: AtLeastOnce
        maxRetryDuration: 20
        maxWrite: 10M
        minRetryDuration: 5
      url: <otlp_url> 2
  pipelines:
  - inputRefs:
    - application
    - infrastructure
    - audit
    name: otlp-logs
    outputRefs:
    - otlp
```

1 이 주석을 사용하여 기술 프리뷰 기능인OTLP(OpenTelemetry Protocol) 출력을 활성화합니다.

2 이 URL은 절대적이어야 하며 로그가 전송되는 OTLP 끝점의 자리 표시자입니다.

참고

OTLP 출력에서는 다른 출력 유형에서 사용하는 ViaQ 데이터 모델과 다른 OpenTelemetry 데이터 모델을 사용합니다. OpenTelemetry Observability 프레임워크에서 정의한 OpenTelemetry Semantic ations를 사용하여 OTLP를 준수합니다.

2.3.5.1. 파이프라인

파이프라인은 **spec.pipelines** 의 배열에 구성됩니다. 각 파이프라인에는 고유한 이름이 있어야 하며 다음으로 구성됩니다.

inputRefs

로그가 이 파이프라인으로 전달되어야 하는 입력의 이름입니다.

outputRefs

로그를 전송할 출력의 이름입니다.

filterRefs

(선택 사항) 적용할 필터 이름입니다.

filterRefs 순서가 순차적으로 적용되므로 중요합니다. 이전 필터는 이후 필터에서 처리되지 않는 메시지를 삭제할 수 있습니다.

2.3.5.2. 필터

필터는 **spec.filters** 아래의 배열에 구성됩니다. 구조화된 필드의 값을 기반으로 들어오는 로그 메시지를 일치시키고 수정하거나 삭제할 수 있습니다.

관리자는 다음 유형의 필터를 구성할 수 있습니다.

2.3.5.3. 여러 줄 예외 탐지 활성화

컨테이너 로그의 여러 줄 오류 감지를 활성화합니다.

주의

이 기능을 활성화하면 성능에 영향을 미칠 수 있으며 추가 컴퓨팅 리소스 또는 대체 로깅 솔루션이 필요할 수 있습니다.

로그 구문 분석기가 별도의 예외와 동일한 예외의 별도의 행을 잘못 식별하는 경우가 많습니다. 이로 인해 추가 로그 항목과 추적된 정보에 대한 불완전하거나 부정확한 보기가 발생합니다.

Java 예외 예

```
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)
```

- 로깅을 사용하여 여러 줄 예외를 감지하고 이를 단일 로그 항목으로 재조정하려면 **ClusterLogForwarder** 사용자 정의 리소스(CR)에 **.spec.filters** 아래의 **detectMultilineErrors** 필드가 포함되어 있는지 확인합니다.

Example ClusterLogForwarder CR

```
apiVersion: "observability.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name>
  namespace: <log_forwarder_namespace>
spec:
  serviceAccount:
    name: <service_account_name>
  filters:
  - name: <name>
    type: detectMultilineException
  pipelines:
    - inputRefs:
        - <input-name>
      name: <pipeline-name>
      filterRefs:
        - <filter-name>
      outputRefs:
        - <output-name>
```

2.3.5.3.1. 세부 정보

로그 메시지가 예외 스택 추적을 형성하는 연속 시퀀스로 표시되면 단일 통합 로그 레코드로 결합됩니다. 첫 번째 로그 메시지의 콘텐츠는 시퀀스의 모든 메시지 필드의 연결된 콘텐츠로 교체됩니다.

수집기는 다음 언어를 지원합니다.

- Java
- JS
- Ruby
- Python
- Golang
- PHP
- Dart

2.3.5.4. 원하지 않는 로그 레코드를 삭제하도록 콘텐츠 필터 구성

드롭 필터가 구성되면 로그 수집기는 전달 전에 필터에 따라 로그 스트림을 평가합니다. 수집기는 지정된 구성과 일치하는 원하지 않는 로그 레코드를 삭제합니다.

프로세스

1. **ClusterLogForwarder** CR의 **필터 사양에 필터** 구성을 추가합니다.
   다음 예제에서는 정규식을 기반으로 로그 레코드를 삭제하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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) 필터 유형을 지정합니다. **drop** 필터는 필터 구성과 일치하는 로그 레코드를 삭제합니다.

(2) **드롭** 필터를 적용하기 위한 구성 옵션을 지정합니다.

(3) 로그 레코드 삭제 여부를 평가하는 데 사용되는 테스트에 대한 구성을 지정합니다.

- 테스트에 지정된 모든 조건이 true이면 테스트가 통과되고 로그 레코드가 삭제됩니다.
- **drop** 필터 구성에 대해 여러 테스트가 지정되면 테스트가 통과하면 레코드가 삭제됩니다.
- 예를 들어 평가 중인 로그 레코드에서 필드가 누락되면 해당 조건이 false로 평가됩니다.

(4) 로그 레코드의 필드 경로인 점으로 구분된 필드 경로를 지정합니다. 경로에는 alpha-numeric 문자 및 밑줄(**a-zA-Z0-9_**)을 포함할 수 있습니다(예: **.kubernetes.namespace_name**). 세그먼트에 이 범위를 벗어나는 문자가 포함된 경우 세그먼트는 따옴표로 묶어야 합니다(예: **.kubernetes.labels."foo.bar-bar/baz"**. 단일 **테스트 구성에 여러 필드 경로를 포함할 수 있지만 테스트가** 통과하고 **drop** 필터를 적용하려면 모두 true로 평가되어야 합니다.

(5) 정규식을 지정합니다. 로그 레코드가 이 정규식과 일치하는 경우 해당 레코드가 삭제됩니다. 단일 **필드** 경로에 대해 **matches** 또는 **notMatches** 조건을 설정할 수 있지만 둘 다 설정할 수는 없습니다.

(6) 정규식을 지정합니다. 로그 레코드가 이 정규식과 일치하지 않으면 삭제됩니다. 단일 **필드** 경로에 대해 **matches** 또는 **notMatches** 조건을 설정할 수 있지만 둘 다 설정할 수는 없습니다.

(7) **drop** 필터가 적용되는 파이프라인을 지정합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

추가 예

다음 추가 예제에서는 우선 순위가 높은 로그 레코드만 유지하도록 **drop** 필터를 구성하는 방법을 보여줍니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  filters:
  - name: important
    type: drop
    drop:
    - test:
      - field: .message
        notMatches: "(?i)critical|error"
      - field: .level
        matches: "info|warning"
# ...
```

단일 **테스트** 구성에 여러 필드 경로를 포함하는 것 외에도 *또는* 검사로 처리되는 추가 테스트를 포함할 수도 있습니다. 다음 예에서는 **테스트** 구성이 true로 평가되면 레코드가 삭제됩니다. 그러나 두 번째 **테스트** 구성의 경우 두 필드 사양을 모두 true로 평가하려면 모두 true여야 합니다.

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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"
# ...
```

2.3.5.5. API 감사 필터 개요

OpenShift API 서버는 각 API 호출에 대한 감사 이벤트를 생성하여 요청자의 요청, 응답 및 ID를 자세히 설

명하여 대량의 데이터를 생성합니다. API 감사 필터는 규칙을 사용하여 필수가 아닌 이벤트를 제외하고 이벤트 크기 감소를 활성화하여 보다 관리가 용이한 감사 추적을 지원합니다. 규칙은 순서대로 확인되며 첫 번째 일치 시에 확인 중지됩니다. 이벤트에 포함된 데이터 양은 **수준** 필드의 값에 따라 결정됩니다.

- **없음**: 이벤트가 삭제되었습니다.
- **metadata**: 감사 메타데이터가 포함되어 요청 및 응답 본문이 제거됩니다.
- **요청**: 감사 메타데이터와 요청 본문이 포함되어 응답 본문이 제거됩니다.
- **RequestResponse**: 메타데이터, 요청 본문, 응답 본문 등 모든 데이터가 포함됩니다. 응답 본문은 매우 커질 수 있습니다. 예를 들어 **oc get pods -A** 는 클러스터의 모든 포드에 대한 YAML 설명이 포함된 응답 본문을 생성합니다.

ClusterLogForwarder CR(사용자 정의 리소스)은 다음과 같은 추가 기능을 제공하는 동안 표준 Kubernetes 감사 정책과 동일한 형식을 사용합니다.

와일드카드

사용자, 그룹, 네임스페이스 및 리소스의 이름에는 선행 또는 후행 * 별표 문자가 있을 수 있습니다. 예를 들어 **openshift-*** 네임스페이스는 **openshift-apiserver** 또는 **openshift-authentication** 과 일치합니다. 리소스 ***/status** 는 **Pod/status** 또는 **Deployment/status** 와 일치합니다.

기본 규칙

정책의 규칙과 일치하지 않는 이벤트는 다음과 같이 필터링됩니다.

- **get**,**list** 및 **watch** 와 같은 읽기 전용 시스템 이벤트가 삭제됩니다.
- 서비스 계정은 서비스 계정과 동일한 네임스페이스 내에서 발생하는 이벤트를 기록합니다.
- 기타 모든 이벤트는 구성된 속도 제한에 따라 전달됩니다.

이러한 기본값을 비활성화하려면 **수준** 필드만 있는 규칙으로 규칙 목록을 종료하거나 빈 규칙을 추가합니다.

응답 코드 생략

생략할 정수 상태 코드 목록입니다. 이벤트가 생성되지 않은 HTTP 상태 코드를 나열하는 **OmitResponseCodes** 필드를 사용하여 응답에서 HTTP 상태 코드를 기반으로 이벤트를 삭제할 수 있습니다. 기본값은 **[404, 409, 422, 429]** 입니다. 값이 빈 목록 **[]** 인 경우 상태 코드는 생략되지 않습니다.

ClusterLogForwarder CR 감사 정책은 OpenShift Container Platform 감사 정책 외에도 작동합니다. **ClusterLogForwarder** CR 감사 필터는 로그 수집기가 전달하는 내용을 변경하고 동사, 사용자, 그룹, 네임스페이스 또는 리소스별로 필터링하는 기능을 제공합니다. 여러 필터를 생성하여 동일한 감사 스트림의 다른 요약을 다른 위치로 보낼 수 있습니다. 예를 들어 자세한 스트림을 로컬 클러스터 로그 저장소로 전송하고 더 자세한 스트림을 원격 사이트로 보낼 수 있습니다.

참고

감사 로그를 수집하려면 클러스터 역할 **collect-audit-logs** 가 있어야 합니다. 제공된 다음 예제는 감사 정책에서 가능한 규칙 범위를 설명하기 위한 것이며 권장되는 구성은 아닙니다.

감사 정책의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
```

```
metadata:
  name: <log_forwarder_name>
  namespace: <log_forwarder_namespace>
spec:
  serviceAccount:
    name: <service_account_name>
  pipelines:
    - name: my-pipeline
      inputRefs: audit (1)
      filterRefs: my-policy (2)
  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 수집되는 로그 유형입니다. 이 필드의 값은 **감사** 로그, 애플리케이션 로그의 **애플리케이션**, 인프라 로그용 **인프라** 또는 애플리케이션에 대해 정의된 이름이 지정된 입력에 대한 감사일 수 있습니다.

2 감사 정책의 이름입니다.

2.3.5.6. 레이블 표현식 또는 일치하는 라벨 키와 값을 포함하여 입력 시 애플리케이션 로그 필터링

입력 선택기를 사용하여 라벨 표현식 또는 일치하는 라벨 키와 해당 값을 기반으로 애플리케이션 로그를 포함할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR의 **입력 사양에** 필터 구성을 추가합니다.
   다음 예제에서는 라벨 표현식 또는 일치하는 라벨 키/값에 따라 로그를 포함하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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
      type: application
# ...
```

1. 일치시킬 레이블 키를 지정합니다.
2. Operator를 지정합니다. 유효한 값에는 **in**,**NotIn**,**Exists** 및 **DoesNotExist** 가 있습니다.
3. 문자열 값의 배열을 지정합니다. **operator** 값이 **Exists** 또는 **DoesNotExist** 이면 값 배열이 비어 있어야 합니다.
4. 정확한 키 또는 값 매핑을 지정합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

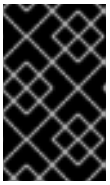
```
$ oc apply -f <filename>.yaml
```

2.3.5.7. 로그 레코드를 정리하도록 콘텐츠 필터 구성

정리 필터가 구성되면 로그 수집기는 전달 전에 필터에 따라 로그 스트림을 평가합니다. 수집기는 Pod 주석과 같은 낮은 값 필드를 제거하여 로그 레코드를 정리합니다.

프로세스

1. **ClusterLogForwarder** CR의 **prune** 사양에 필터 구성을 추가합니다.
   다음 예제에서는 필드 경로를 기반으로 로그 레코드를 정리하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.



중요

둘 다 지정된 경우 먼저 **notIn** 배열에 따라 레코드가 정리되며, 이 경우 **in** 배열보다 우선합니다. **notIn** 배열을 사용하여 레코드를 정리하면 **in** 배열을 사용하여 정리됩니다.

ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  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. 필터 유형을 지정합니다. **prune** 필터는 구성된 필드를 통해 로그 레코드를 정리합니다.
2. **prune** 필터를 적용하기 위한 구성 옵션을 지정합니다. **in** 및 **notIn** 필드는 로그 레코드의 필드 경로의 경로인 점으로 구분된 필드 경로의 배열로 지정됩니다. 이러한 경로에는 alpha-numeric 문자 및 밑줄(**a-zA-Z0-9_**)을 포함할 수 있습니다(예: **.kubernetes.namespace_name**). 세그먼트에 이 범위를 벗어나는 문자가 포함된 경우 세그먼트는 따옴표로 묶어야 합니다(예: **.kubernetes.labels."foo.bar-bar/baz"**.
3. 선택 사항: 이 배열에 지정된 모든 필드는 로그 레코드에서 제거됩니다.
4. 선택 사항: 이 배열에 지정되지 않은 모든 필드는 로그 레코드에서 제거됩니다.
5. **prune** 필터가 적용되는 파이프라인을 지정합니다.

참고

필터는 **log_type**,**.log_source**, **.message** 필드를 제외합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

2.3.6. 소스로 감사 및 인프라 로그 입력 필터링

입력 선택기를 사용하여 로그를 수집할 **감사** 및 **인프라** 소스 목록을 정의할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR에서 **감사** 및 **인프라** 소스를 정의하는 구성을 추가합니다.
   다음 예제에서는 **감사** 및 **인프라** 소스를 정의하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  inputs:
    - name: mylogs1
      type: infrastructure
      infrastructure:
        sources: 1
```

```
      - node
  - name: mylogs2
    type: audit
    audit:
      sources: 2
        - kubeAPI
        - openshiftAPI
        - ovn
# ...
```

1 수집할 인프라 소스 목록을 지정합니다. 유효한 소스는 다음과 같습니다.

- **Node**: 노드에서 저널 로그
- **컨테이너**: 네임스페이스에 배포된 워크로드의 로그

2 수집할 감사 소스 목록을 지정합니다. 유효한 소스는 다음과 같습니다.

- **kubeAPI**: Kubernetes API 서버의 로그
- **openshiftAPI**: OpenShift API 서버의 로그
- **auditd**: 노드 auditd 서비스의 로그
- **OVN**: 오픈 가상 네트워크 서비스의 로그

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

2.3.7. 네임스페이스 또는 컨테이너 이름을 포함하거나 제외하여 입력 시 애플리케이션 로그 필터링

입력 선택기를 사용하여 네임스페이스 및 컨테이너 이름을 기반으로 애플리케이션 로그를 포함하거나 제외할 수 있습니다.

프로세스

1. **ClusterLogForwarder** CR에 네임스페이스 및 컨테이너 이름을 포함하거나 제외하는 구성을 추가합니다.
   다음 예제에서는 네임스페이스 및 컨테이너 이름을 포함하거나 제외하도록 **ClusterLogForwarder** CR을 구성하는 방법을 보여줍니다.

   ClusterLogForwarder CR의 예

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
# ...
spec:
  serviceAccount:
    name: <service_account_name>
  inputs:
    - name: mylogs
      application:
```

```
      includes:
        - namespace: "my-project" 1
          container: "my-container" 2
      excludes:
        - container: "other-container*" 3
          namespace: "other-namespace" 4
    type: application
# ...
```

1 이러한 네임스페이스에서만 로그를 수집하도록 지정합니다.

2 이러한 컨테이너에서만 로그를 수집하도록 지정합니다.

3 로그를 수집할 때 무시할 네임스페이스 패턴을 지정합니다.

4 로그를 수집할 때 무시할 컨테이너 세트를 지정합니다.

참고

excludes 필드는 **includes** 필드보다 우선합니다.

2. 다음 명령을 실행하여 **ClusterLogForwarder** CR을 적용합니다.

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

2.4. LOKISTACK을 사용하여 로그 저장

애플리케이션, 감사 및 인프라 관련 로그를 저장하도록 **LokiStack** CR을 구성할 수 있습니다.

Loki는 수평으로 확장 가능한 고가용성 다중 테넌트 로그 집계 시스템으로 OpenShift Observability UI로 시각화할 수 있는 Red Hat OpenShift의 로깅을 위한 GA 로그 저장소로 제공됩니다. OpenShift Logging에서 제공하는 Loki 구성은 사용자가 수집된 로그를 사용하여 빠른 문제 해결을 수행할 수 있도록 설계된 단기 로그 저장소입니다. 이를 위해 Loki의 Red Hat OpenShift 구성에 대한 로깅은 단기 스토리지가 있으며 최근 쿼리에 최적화되어 있습니다.

중요

장기간의 기간 동안의 스토리지 또는 쿼리의 경우 사용자는 클러스터 외부에 있는 저장소를 로그하려고 합니다. Loki 크기 조정은 최대 30일 동안 단기 스토리지에서만 테스트 및 지원됩니다.

2.4.1. Loki 배포 크기 조정

Loki의 크기 조정은 **1x.<size> 형식을 따릅니다.** 여기서 **1x** 값은 인스턴스 수이고 < **size** >는 성능 기능을 지정합니다.

1x.pico 구성은 최소한의 리소스 및 제한 요구 사항으로 단일 Loki 배포를 정의하여 모든 Loki 구성 요소에 대한 HA(고가용성) 지원을 제공합니다. 이 구성은 단일 복제 요소 또는 자동 컴파일이 필요하지 않은 배포에 적합합니다.

디스크 요청은 크기 구성에서 유사합니다. 고객이 다양한 크기를 테스트하여 배포 요구 사항에 가장 적합한지 확인할 수 있습니다.

중요

배포 크기에 대해 숫자 **1x** 를 변경할 수 없습니다.

표 2.1. Loki 크기 조정

	1x.demo	1x.pico [6.1+ only]	1x.extra-small	1x.small	1x.medium
데이터 전송	데모만 사용	50GB/일	100GB/일	500GB/일	2TB/일
초당 쿼리 (QPS)	데모만 사용	200ms에서 1-25 QPS	200ms에서 1-25 QPS	25-50 QPS (200ms)	25-75 QPS (200ms)
복제 요인	없음	2	2	2	2
총 CPU 요청	없음	7개의 vCPU	14개의 vCPU	34 vCPU	54 vCPU
ruler을 사용하는 경우 총 CPU 요청	없음	8개의 vCPU	16개의 vCPU	42 vCPU	70개의 vCPU
총 메모리 요청	없음	17Gl	31Gl	67Gl	139Gi
룰러를 사용하는 경우 총 메모리 요청	없음	18Gl	35Gl	83Gi	171Gi
총 디스크 요청	40Gi	590Gi	430Gi	430Gi	590Gi
ruler을 사용하는 경우 총 디스크 요청	80Gl	910Gi	750Gi	750Gi	910Gi

2.4.2. 사전 요구 사항

- CLI 또는 웹 콘솔을 사용하여 Loki Operator를 설치했습니다.
- **ClusterLogForwarder** 를 생성하는 동일한 네임스페이스에 **serviceAccount** 가 있습니다.
- **serviceAccount** 에는 **collect-audit-logs**,**collect-application-logs**, **collect-infrastructure-logs** 클러스터 역할이 할당됩니다.

2.4.3. 코어 설정 및 구성

Loki를 배포하기 위한 역할 기반 액세스 제어, 기본 모니터링 및 Pod 배치입니다.

2.4.4. LokiStack 규칙 RBAC 권한 승인

관리자는 클러스터 역할을 사용자 이름에 바인딩하여 사용자가 자체 경고 및 레코딩 규칙을 생성하고 관리할 수 있습니다. 클러스터 역할은 사용자에게 필요한 RBAC(역할 기반 액세스 제어) 권한이 포함된 **ClusterRole** 오브젝트로 정의됩니다.

경고 및 레코딩 규칙에 대한 다음 클러스터 역할은 LokiStack에 사용할 수 있습니다.

규칙 이름	설명
alertingrules.loki.grafana.com-v1-admin	이 역할의 사용자에게는 경고 규칙을 관리할 수 있는 관리 수준 액세스 권한이 있습니다. 이 클러스터 역할은 **loki.grafana.com/v1** API 그룹 내에서 **AlertingRule** 리소스를 생성, 읽기, 업데이트, 삭제, 나열 및 조사할 수 있는 권한을 부여합니다.
alertingrules.loki.grafana.com-v1-crdview	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내의 **AlertingRule** 리소스와 관련된 CRD(Custom Resource Definitions)의 정의를 볼 수 있지만 이러한 리소스를 수정하거나 관리할 수 있는 권한은 없습니다.
alertingrules.loki.grafana.com-v1-edit	이 역할의 사용자는 **AlertingRule** 리소스를 생성, 업데이트 및 삭제할 수 있는 권한이 있습니다.
alertingrules.loki.grafana.com-v1-view	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내에서 **AlertingRule** 리소스를 읽을 수 있습니다. 기존 경고 규칙에 대한 구성, 라벨 및 주석을 검사할 수 있지만 수정할 수는 없습니다.
recordingrules.loki.grafana.com-v1-admin	이 역할의 사용자는 레코딩 규칙을 관리할 수 있는 관리 수준의 액세스 권한이 있습니다. 이 클러스터 역할은 **loki.grafana.com/v1** API 그룹 내의 **RecordingRule** 리소스를 생성, 읽기, 업데이트, 삭제, 나열 및 감시할 수 있는 권한을 부여합니다.
recordingrules.loki.grafana.com-v1-crdview	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내의 **RecordingRule** 리소스와 관련된 CRD(Custom Resource Definitions)의 정의를 볼 수 있지만 이러한 리소스를 수정하거나 관리할 수 있는 권한은 없습니다.
recordingrules.loki.grafana.com-v1-edit	이 역할의 사용자는 **RecordingRule** 리소스를 생성, 업데이트 및 삭제할 수 있는 권한이 있습니다.
recordingrules.loki.grafana.com-v1-view	이 역할의 사용자는 **loki.grafana.com/v1** API 그룹 내에서 **RecordingRule** 리소스를 읽을 수 있습니다. 기존 경고 규칙에 대한 구성, 라벨 및 주석을 검사할 수 있지만 수정할 수는 없습니다.

2.4.4.1. 예

사용자의 클러스터 역할을 적용하려면 기존 클러스터 역할을 특정 사용자 이름에 바인딩해야 합니다.

클러스터 역할은 사용하는 역할 바인딩 유형에 따라 클러스터 또는 네임스페이스 범위일 수 있습니다. **RoleBinding** 오브젝트가 사용되는 경우 **oc adm policy add-role-to-user** 명령을 사용하는 경우 클러스

터 역할은 지정된 네임스페이스에만 적용됩니다. **oc adm policy add-cluster-role-to-user** 명령을 사용할 때 **ClusterRoleBinding** 오브젝트가 사용되는 경우 클러스터 역할은 클러스터의 모든 네임스페이스에 적용됩니다.

다음 예제 명령은 클러스터의 특정 네임스페이스의 경고 규칙에 대해 지정된 사용자 생성, 읽기, 업데이트 및 삭제(CRUD) 권한을 제공합니다.

특정 네임스페이스에서 경고 규칙 **CRUD** 권한에 대한 클러스터 역할 바인딩 명령의 예

```
$ oc adm policy add-role-to-user alertingrules.loki.grafana.com-v1-admin -n <namespace>
<username>
```

다음 명령은 모든 네임스페이스의 경고 규칙에 대해 지정된 사용자 관리자 권한을 부여합니다.

관리자 권한에 대한 클러스터 역할 바인딩 명령의 예

```
$ oc adm policy add-cluster-role-to-user alertingrules.loki.grafana.com-v1-admin <username>
```

2.4.5. Loki를 사용하여 로그 기반 경고 규칙 생성

AlertingRule CR에는 단일 **LokiStack** 인스턴스에 대한 경고 규칙 그룹을 선언하는 일련의 사양 및 Webhook 검증 정의가 포함되어 있습니다. 또한 웹 후크 검증 정의에서는 규칙 검증 조건을 지원합니다.

- **AlertingRule** CR에 잘못된 **간격** 기간이 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR **에 기간 동안** 유효하지 않은 항목이 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR에 잘못된 LogQL **expr** 가 포함된 경우 잘못된 경고 규칙입니다.
- **AlertingRule** CR에 동일한 이름의 두 개의 그룹이 포함된 경우 잘못된 경고 규칙입니다.
- 위의 어느 것도 적용되지 않으면 경고 규칙이 유효한 것으로 간주됩니다.

표 2.2. AlertingRule 정의

테넌트 유형	AlertingRule CR을 위한 유효한 네임스페이스
애플리케이션	**<your_application_namespace>**
audit	**openshift-logging**
인프라	**openshift-/***, **kube-/***, **default**

프로세스

1. **AlertingRule** 사용자 정의 리소스(CR)를 생성합니다.

   인프라 **AlertingRule CR**의 예

   ```
   apiVersion: loki.grafana.com/v1
   kind: AlertingRule
   metadata:
   ```

```
  name: loki-operator-alerts
  namespace: openshift-operators-redhat (1)
  labels: (2)
    openshift.io/<label_name>: "true"
spec:
  tenantID: "infrastructure" (3)
  groups:
    - name: LokiOperatorHighReconciliationError
      rules:
        - alert: HighPercentageError
          expr: | (4)
            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 (5)
          annotations:
            summary: High Loki Operator Reconciliation Errors (6)
            description: High Loki Operator Reconciliation Errors (7)
```

(1) 이 **AlertingRule** CR이 생성되는 네임스페이스에는 LokiStack **spec.rules.namespaceSelector** 정의와 일치하는 레이블이 있어야 합니다.

(2) **labels** 블록은 LokiStack **spec.rules.selector** 정의와 일치해야 합니다.

(3) **인프라** 테넌트에 대한 **AlertingRule** CR은 **openshift-***, **kube-*** 또는 **default** 네임스페이스에서만 지원됩니다.

(4) **kubernetes_namespace_name:** 의 값은 **metadata.namespace** 값과 일치해야 합니다.

(5) 이 필수 필드의 값은 **중요**,**경고** 또는 **정보** 여야 합니다.

(6) 이 필드는 필수입니다.

(7) 이 필드는 필수입니다.

애플리케이션 **AlertingRule CR**의 예

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: app-user-workload
  namespace: app-ns (1)
  labels: (2)
    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. 이 **AlertingRule** CR이 생성되는 네임스페이스에는 LokiStack **spec.rules.namespaceSelector** 정의와 일치하는 레이블이 있어야 합니다.
2. **labels** 블록은 LokiStack **spec.rules.selector** 정의와 일치해야 합니다.
3. **kubernetes_namespace_name:** 의 값은 **metadata.namespace** 값과 일치해야 합니다.
4. 이 필수 필드의 값은 **중요**,**경고** 또는 **정보** 여야 합니다.
5. 이 필수 필드의 값은 규칙에 대한 요약입니다.
6. 이 필수 필드의 값은 규칙에 대한 자세한 설명입니다.

2. **AlertingRule** CR을 적용합니다.

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

2.4.6. 멤버 목록 생성 실패를 허용하도록 Loki 구성

OpenShift Container Platform 클러스터에서 관리자는 일반적으로 개인 IP 네트워크 범위를 사용합니다. 결과적으로 LokiStack 멤버 목록 구성은 기본적으로 개인 IP 네트워크만 사용하므로 실패합니다.

관리자는 memberlist 구성에 대한 Pod 네트워크를 선택할 수 있습니다. **hashRing** 사양에서 **podIP** 주소를 사용하도록 **LokiStack** CR(사용자 정의 리소스)을 수정할 수 있습니다. **LokiStack** CR을 구성하려면 다음 명령을 사용합니다.

```
$ oc patch LokiStack logging-loki -n openshift-logging  --type=merge -p '{"spec": {"hashRing":
{"memberlist":{"instanceAddrType":"podIP"},"type":"memberlist"}}}'
```

podIP를 포함하는 **LokiStack**의 예

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  hashRing:
    type: memberlist
```

```
    memberlist:
      instanceAddrType: podIP
# ...
```

2.4.7. Loki를 사용하여 스트림 기반 보존 활성화

로그 스트림을 기반으로 보존 정책을 구성할 수 있습니다. 이러한 규칙에 대한 규칙은 전역적으로, 테넌트별로 또는 둘 다 설정할 수 있습니다. 둘 다 구성하는 경우 테넌트 규칙이 글로벌 규칙 앞에 적용됩니다.

중요

s3 버킷 또는 LokiStack 사용자 정의 리소스(CR)에 정의된 보존 기간이 없으면 로그가 정리되지 않고 s3 스토리지를 채울 수 있습니다.

참고

스키마 v13이 권장됩니다.

프로세스

1. **LokiStack** CR을 생성합니다.
   - 다음 예와 같이 스트림 기반 보존을 전역적으로 활성화합니다.

     AWS에 대한 글로벌 스트림 기반 보존의 예

```
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: v13
    secret:
      name: logging-loki-s3
      type: aws
```

```
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

1. 모든 로그 스트림에 대한 보존 정책을 설정합니다. **참고: 이 필드는 오브젝트 스토리지에 저장된 로그의 보존 기간에는 영향을 미치지 않습니다.**
2. 이 블록이 CR에 추가되면 클러스터에서 보존이 활성화됩니다.
3. 로그 스트림.spec: 제한을 정의하는 데 사용되는 LogQL 쿼리 를 포함합니다.

- 다음 예와 같이 테넌트별로 스트림 기반 보존을 활성화합니다.

  AWS에 대한 테넌트별 스트림 기반 보존 예

```
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: v13
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

1. 테넌트별 보존 정책을 설정합니다. 유효한 테넌트 유형은 **애플리케이션**,**감사** 및 **인프라** 입니다.

2 로그 스트림을 정의하는 데 사용되는 LogQL 쿼리 를 포함합니다.

2. **LokiStack** CR을 적용합니다.

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

2.4.8. Loki Pod 배치

Pod의 허용 오차 또는 노드 선택기를 사용하여 Loki Pod가 실행되는 노드를 제어하고 다른 워크로드가 해당 노드를 사용하지 못하도록 할 수 있습니다.

LokiStack CR(사용자 정의 리소스)을 사용하여 로그 저장소 Pod에 허용 오차를 적용하고 노드 사양이 있는 노드에 taint를 적용할 수 있습니다. 노드의 테인트는 테인트를 허용하지 않는 모든 Pod를 거절하도록 노드에 지시하는 **키:값** 쌍입니다. 다른 Pod에 없는 특정 **키:값** 쌍을 사용하면 해당 노드에서 로그 저장소 Pod만 실행할 수 있습니다.

노드 선택기가 있는 **LokiStack**의 예

```
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 노드 선택기에 적용되는 구성 요소 Pod 유형을 지정합니다.

2 정의된 라벨이 포함된 노드로 이동되는 Pod를 지정합니다.

노드 선택기 및 허용 오차가 있는 **LokiStack CR**의 예

```
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)의 **nodeSelector** 및 **허용 오차** 필드를 구성하려면 **oc explain** 명령을 사용하여 특정 리소스에 대한 설명 및 필드를 볼 수 있습니다.

```
$ oc explain lokistack.spec.template
```

출력 예

```
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.
...
```

자세한 내용은 특정 필드를 추가할 수 있습니다.

```
$ oc explain lokistack.spec.template.compactor
```

출력 예

```
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.
...
```

2.4.8.1. 향상된 신뢰성 및 성능

프로덕션 환경에서 Loki의 안정성 및 효율성을 보장하는 구성입니다.

2.4.8.2. 수명이 짧은 토큰을 사용하여 클라우드 기반 로그 저장소에 대한 인증 활성화

워크로드 ID 페더레이션을 통해 단기 토큰을 사용하여 클라우드 기반 로그 저장소에 인증할 수 있습니다.

프로세스

- 다음 옵션 중 하나를 사용하여 인증을 활성화합니다.
  - OpenShift Container Platform 웹 콘솔을 사용하여 Loki Operator를 설치하면 수명이 짧은 토큰을 사용하는 클러스터가 자동으로 감지됩니다. 역할을 생성하고 Loki Operator에서 보안을 채우는 **CredentialsRequest** 오브젝트를 생성하는 데 필요한 데이터를 제공하라는 메시지가 표시됩니다.

- OpenShift CLI(**oc**)를 사용하여 Loki Operator를 설치하는 경우 다음 예와 같이 스토리지 공급자에 적절한 템플릿을 사용하여 **Subscription** 오브젝트를 수동으로 생성해야 합니다. 이 인증 전략은 표시된 스토리지 공급자에 대해서만 지원됩니다.

  Azure 샘플 서브스크립션의 예

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-6.0"
  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 샘플 서브스크립션 예

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-6.0"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: ROLEARN
      value: <role_ARN>
```

2.4.8.3. 노드 장애를 허용하도록 Loki 구성

Loki Operator는 Pod 유사성 방지 규칙을 설정하여 동일한 구성 요소의 Pod가 클러스터의 다른 사용 가능한 노드에 예약되도록 요청할 수 있습니다.

유사성은 예약할 노드를 제어하는 Pod의 속성입니다. 유사성 방지는 Pod가 노드에서 예약되지 않도록 하는 Pod의 속성입니다.

OpenShift Container Platform에서 *Pod 유사성* 및 *Pod 유사성 방지*를 사용하면 다른 Pod의 키 값 라벨에 따라 Pod를 예약할 수 있는 노드를 제한할 수 있습니다.

Operator는 **compactor**,**distributor**,**gateway**,**indexGateway**,**ingester**,**querier**,**queryFrontend** 및 **ruler** 구성 요소를 포함하는 모든 Loki 구성 요소에 대해 기본 기본 **podAntiAffinity** 규칙을 설정합니다.

requiredDuringSchedulingIgnoredDuringExecution 필드에서 필요한 설정을 구성하여 Loki 구성 요소의 기본 **podAntiAffinity** 설정을 덮어쓸 수 있습니다.

ingester 구성 요소에 대한 사용자 설정 예

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    ingester:
      podAntiAffinity:
      # ...
        requiredDuringSchedulingIgnoredDuringExecution: 1
        - labelSelector:
            matchLabels: 2
              app.kubernetes.io/component: ingester
          topologyKey: kubernetes.io/hostname
# ...
```

1. 필수 규칙을 정의하는 스탠자입니다.
2. 규칙을 적용하려면 일치해야 하는 키-값 쌍(레이블)입니다.

2.4.8.4. 클러스터를 다시 시작하는 동안 LokiStack 동작

OpenShift Container Platform 클러스터가 다시 시작되면 LokiStack 수집 및 쿼리 경로가 노드에 사용 가능한 사용 가능한 CPU 및 메모리 리소스 내에서 계속 작동합니다. 즉, OpenShift Container Platform 클러스터 업데이트 중에 LokiStack에 대한 다운 타임이 없습니다. 이 동작은 **PodDisruptionBudget** 리소스를 사용하여 수행됩니다. Loki Operator는 특정 조건에서 정상적인 작업을 보장하기 위해 구성 요소별로 사용 가능한 최소 Pod 수를 결정하는 Loki의 **PodDisruptionBudget** 리소스를 프로비저닝합니다.

2.4.8.5. 고급 배포 및 확장성

고가용성, 확장성 및 오류 처리를 위한 특수 구성입니다.

2.4.8.6. 영역 인식 데이터 복제

Loki Operator는 Pod 토폴로지 분배 제약 조건을 통해 영역 인식 데이터 복제를 지원합니다. 이 기능을 사용하면 단일 영역 장애가 발생할 경우 로그 손실에 대한 안정성과 보호 장치가 향상됩니다. 배포 크기를 **1x.extra-** windows , **1x.** ngressController 또는 **1x.medium** 으로 구성하는 경우 **replication.factor** 필드가 자동으로 2로 설정됩니다.

적절한 복제를 위해서는 복제 요인에서 지정한 만큼 이상의 가용성 영역이 있어야 합니다. 복제 요인보다 가용성 영역을 더 많이 보유할 수는 있지만 영역이 줄어들면 오류를 작성할 수 있습니다. 각 영역은 최적의 작업을 위해 동일한 수의 인스턴스를 호스팅해야 합니다.

영역 복제가 활성화된 **LokiStack CR**의 예

```
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. 더 이상 사용되지 않는 필드로 입력된 값은 **replication.factor** 로 덮어씁니다.
2. 이 값은 배포 크기가 설정 시 선택되면 자동으로 설정됩니다.
3. 두 토폴로지 도메인 간 최대 Pod 수 차이입니다. 기본값은 1이며 값 0을 지정할 수 없습니다.
4. 노드 레이블에 해당하는 토폴로지 키 형태로 영역을 정의합니다.

2.4.8.7. 실패한 영역에서 Loki Pod 복구

OpenShift Container Platform에서 특정 가용성 영역 리소스에 액세스할 수 없게 되면 영역 오류가 발생합니다. 가용성 영역은 클라우드 공급자의 데이터 센터 내의 격리된 영역이며 중복성과 내결함성을 강화하기 위한 것입니다. OpenShift Container Platform 클러스터가 이를 처리하도록 구성되지 않은 경우 영역 실패로 인해 서비스 또는 데이터가 손실될 수 있습니다.

Loki 포드는 StatefulSet 의 일부이며 **StorageClass** 오브젝트에서 프로비저닝한 PVC(영구 볼륨 클레임)와 함께 제공됩니다. 각 Loki Pod 및 해당 PVC는 동일한 영역에 있습니다. 클러스터에서 영역 오류가 발생하면 StatefulSet 컨트롤러에서 실패한 영역에서 영향을 받는 Pod를 자동으로 복구하려고 합니다.

주의

다음 절차에서는 실패한 영역의 PVC와 그 안에 포함된 모든 데이터를 삭제합니다. 완전한 데이터 손실을 방지하려면 **LokiStack** CR의 복제 요소 필드를 항상 1보다 큰 값으로 설정하여 Loki가 복제되도록 해야 합니다.

사전 요구 사항

- **LokiStack** CR에 1보다 큰 복제 인수가 있는지 확인합니다.

- 컨트롤 플레인에서 감지한 영역 장애와 실패한 영역의 노드는 클라우드 공급자 통합으로 표시됩니다.

StatefulSet 컨트롤러는 실패한 영역에서 Pod 일정 변경을 자동으로 시도합니다. 연결된 PVC도 실패한 영역에 있기 때문에 다른 영역으로 자동 일정 조정이 작동하지 않습니다. 새 영역에서 상태 저장 Loki Pod와 프로비저닝된 PVC를 다시 생성할 수 있도록 실패한 영역에서 PVC를 수동으로 삭제해야 합니다.

프로세스

1. 다음 명령을 실행하여 **Pending** 상태의 Pod를 나열합니다.

```
$ oc get pods --field-selector status.phase==Pending -n openshift-logging
```

oc get pods 출력 예

```
NAME                           READY   STATUS    RESTARTS   AGE (1)
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) 해당 PVC가 실패한 영역에 있기 때문에 이러한 Pod는 **Pending** 상태입니다.

2. 다음 명령을 실행하여 **Pending** 상태의 PVC를 나열합니다.

```
$ oc get pvc -o=json -n openshift-logging | jq '.items[] | select(.status.phase == "Pending") | .metadata.name' -r
```

oc get pvc 출력 예

```
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. 다음 명령을 실행하여 Pod의 PVC를 삭제합니다.

```
$ oc delete pvc <pvc_name>  -n openshift-logging
```

4. 다음 명령을 실행하여 Pod를 삭제합니다.

```
$ oc delete pod <pod_name>  -n openshift-logging
```

이러한 오브젝트가 성공적으로 삭제되면 사용 가능한 영역에서 자동으로 다시 예약해야 합니다.

2.4.8.7.1. 종료 상태의 PVC 문제 해결

PVC 메타데이터 종료자가 **kubernetes.io/pv-protection** 으로 설정된 경우 PVC는 삭제 없이 종료 상태로 중단될 수 있습니다. 종료자를 제거하면 PVC가 성공적으로 삭제될 수 있습니다.

- 아래 명령을 실행하여 각 PVC의 종료자를 제거한 다음 삭제를 다시 시도합니다.

```
$ oc patch pvc <pvc_name> -p '{"metadata":{"finalizers":null}}' -n openshift-logging
```

2.4.8.8. Loki 속도 제한 오류 문제 해결

로그 전달자 API에서 속도 제한을 초과하는 대규모 메시지 블록을 Loki로 전달하면 Loki는 속도 제한(**429**) 오류를 생성합니다.

이러한 오류는 정상적인 작동 중에 발생할 수 있습니다. 예를 들어 이미 일부 로그가 있는 클러스터에 로깅을 추가할 때 로깅이 기존 로그 항목을 모두 수집하는 동안 속도 제한 오류가 발생할 수 있습니다. 이 경우 새 로그 추가 속도가 총 속도 제한보다 작으면 기록 데이터가 결국 수집되고 사용자 개입 없이도 속도 제한 오류가 해결됩니다.

속도 제한 오류가 계속 발생하는 경우 **LokiStack** CR(사용자 정의 리소스)을 수정하여 문제를 해결할 수 있습니다.

중요

LokiStack CR은 Grafana 호스팅 Loki에서 사용할 수 없습니다. 이는 Grafana 호스팅 Loki 서버에는 적용되지 않습니다.

조건

- Log Forwarder API는 로그를 Loki로 전달하도록 구성되어 있습니다.
- 시스템에서 2MB보다 큰 메시지 블록을 Loki로 보냅니다. 예를 들면 다음과 같습니다.

```
"values":[["1630410392689800468","{\"kind\":\"Event\",\"apiVersion\":\
.......
......
......
......
\"received_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\":\"MzFjYjJkZjItNjY0MC00YWU4LWIwMTEtNGNmM2E5ZmViMGU4\",\"log_type\":\"audit\"}"]]}]}
```

- **oc logs -n openshift-logging -l component=collector** 를 입력하면 클러스터의 수집기 로그에 다음 오류 메시지 중 하나가 포함된 행이 표시됩니다.

```
429 Too Many Requests Ingestion rate limit exceeded
```

Vector 오류 메시지의 예

```
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 오류 메시지의 예

```
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"
```

이 오류는 수신 끝점에도 표시됩니다. 예를 들어 LokiStack ingester Pod에서 다음을 수행합니다.

Loki ingester 오류 메시지의 예

```
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
```

프로세스

- **LokiStack** CR에서 **ingestionBurstSize** 및 **ingestionRate** 필드를 업데이트합니다.

```
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** 필드는 배포자 복제본당 최대 로컬 속도 제한 샘플 크기를 MB로 정의합니다. 이 값은 하드 제한입니다. 이 값을 단일 푸시 요청에 예상되는 최대 로그 크기로 설정합니다. **ingestionBurstSize** 값보다 큰 단일 요청은 허용되지 않습니다.

(2) **ingestionRate** 필드는 초당 수집된 샘플의 최대 양(MB)에 대한 소프트 제한입니다. 로그 비율이 제한을 초과하는 경우 속도 제한 오류가 발생하지만 수집기는 로그를 다시 시도합니다. 총 평균이 제한보다 작으면 사용자 개입 없이 시스템을 복구하고 오류가 해결됩니다.

2.5. LOKI의 OTLP 데이터 수집

로깅 6.1은 OTLP(OpenTelemetry Protocol)를 사용하여 API 엔드포인트를 활성화합니다. OTLP는 Loki를 위해 특별히 설계되지 않은 표준화된 형식이므로 OpenTelemetry의 데이터 형식을 Loki의 데이터 모델에 매핑하려면 Loki의 측에 대한 추가 구성이 필요합니다. OTLP에는 *스트림 레이블* 또는 *구조화된 메타데이터* 와 같은 개념이 없습니다. 대신 OTLP는 세 가지 범주로 그룹화된 **속성으로** 로그 항목에 대한 메타데이터를 제공합니다.

- 리소스
- 범위
- log

이렇게 하면 필요에 따라 여러 항목에 대해 동시에 또는 개별적으로 메타데이터를 설정할 수 있습니다.

2.5.1. OTLP 데이터 수집용 LokiStack 구성

중요

OTLP(OpenTelemetry Protocol) 출력 로그 전달자는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

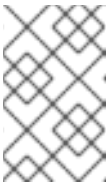
OTLP 수집에 대한 **LokiStack** CR(사용자 정의 리소스)을 구성하려면 다음 단계를 따르십시오.

사전 요구 사항

- Loki 설정이 스키마 버전 13에 도입된 구조화된 메타데이터를 지원하는지 확인하여 OTLP 로그 수집을 활성화합니다.

프로세스

1. 스키마 버전을 설정합니다.
   - 새 **LokiStack** CR을 생성할 때 스토리지 스키마 구성에서 **version: v13** 을 설정합니다.



참고

기존 구성의 경우 **version: v13** 과 **effectiveDate** 를 사용하여 새 스키마 항목을 추가합니다. 스키마 버전 업데이트에 대한 자세한 내용은 업그레이드 스키마 (Grafana 문서)를 참조하십시오.

2. 다음과 같이 스토리지 스키마를 구성합니다.

스토리지 스키마 구성 예

```
# ...
spec:
  storage:
    schemas:
    - version: v13
      effectiveDate: 2024-10-25
```

effectiveDate 가 전달되면 v13 스키마가 적용되어 **LokiStack** 에서 구조화된 메타데이터를 저장할 수 있습니다.

2.5.2. 속성 매핑

Loki Operator가 **openshift-logging** 모드로 설정되면 기본 속성 매핑 세트가 자동으로 적용됩니다. 이러한 매핑은 특정 OTLP 속성을 Loki의 스트림 레이블 및 구조화된 메타데이터와 조정합니다.

일반적인 설정의 경우 이러한 기본 매핑만으로도 충분합니다. 그러나 다음과 같은 경우 속성 매핑을 사용자 지정해야 할 수 있습니다.

- 사용자 지정 수집기 사용: 설정에 추가 속성을 생성하는 사용자 지정 수집기가 포함된 경우 이러한 특성이 Loki에 유지되도록 매핑을 사용자 정의하는 것이 좋습니다.
- 특성 세부 정보 수준 조정: 기본 특성 세트가 필요 이상으로 자세히 설명하는 경우 필수 속성으로만 줄일 수 있습니다. 이로 인해 과도한 데이터 스토리지를 방지하고 로깅 프로세스를 간소화할 수 있습니다.

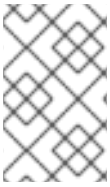
중요

스트림 레이블 또는 구조화된 메타데이터에 매핑되지 않은 속성은 Loki에 저장되지 않습니다.

2.5.2.1. OpenShift의 사용자 정의 속성 매핑

openshift-logging 모드에서 Loki Operator를 사용하는 경우 속성 매핑은 OpenShift 기본값을 따르지만 사용자 정의 매핑을 구성하여 조정할 수 있습니다. 사용자 지정 매핑을 사용하면 추가 구성이 특정 요구 사항을 충족할 수 있습니다.

openshift-logging 모드에서 사용자 정의 속성 매핑은 모든 테넌트 또는 필요에 따라 개별 테넌트에 대해 전역적으로 구성할 수 있습니다. 사용자 정의 매핑이 정의되면 OpenShift 기본값에 추가됩니다. 기본 권장 라벨이 필요하지 않은 경우 테넌트 구성에서 비활성화할 수 있습니다.



참고

Loki Operator와 Loki 자체의 주요 차이점은 상속 처리입니다. Loki는 **default_resource_attributes_as_index_labels** 만 기본적으로 테넌트에 복사하는 반면 Loki Operator는 **openshift-logging** 모드의 각 테넌트에 전체 글로벌 구성을 적용합니다.

LokiStack 내에서 속성 매핑 구성은 **limits** 설정을 통해 관리됩니다.

```
# ...
spec:
  limits:
    global:
      otlp: {} (1)
    tenants:
      application:
        otlp: {} (2)
```

(1) 글로벌 OTLP 특성 구성.

(2) **openshift-logging** 모드 내에서 **애플리케이션** 테넌트에 대한 OTLP 속성 구성입니다.

참고

글로벌 및 테넌트별 OTLP 구성 모두 속성을 스트림 레이블 또는 구조화된 메타데이터에 매핑할 수 있습니다. 로그 항목을 Loki 스토리지에 저장하려면 하나 이상의 스트림 레이블이 필요하므로 이 구성이 해당 요구 사항을 충족하는지 확인합니다.

스트림 레이블은 **LokiStack** 리소스 구조가 반영하는 리소스 수준 속성에서만 파생됩니다.

```
spec:
  limits:
    global:
      otlp:
        streamLabels:
          resourceAttributes:
          - name: "k8s.namespace.name"
          - name: "k8s.pod.name"
          - name: "k8s.container.name"
```

반대로 구조화된 메타데이터는 리소스, 범위 또는 로그 수준 특성에서 생성할 수 있습니다.

```
# ...
spec:
  limits:
    global:
      otlp:
        streamLabels:
          # ...
        structuredMetadata:
          resourceAttributes:
          - name: "process.command_line"
          - name: "k8s\\.pod\\.labels\\..+"
            regex: true
          scopeAttributes:
          - name: "service.name"
          logAttributes:
          - name: "http.route"
```

작은 정보

Loki에서 유사한 속성을 매핑할 때 속성 이름에 **regex: true** 를 설정하여 정규식을 사용합니다.

중요

데이터 볼륨이 증가할 수 있으므로 스트림 레이블에는 정규식을 사용하지 마십시오.

2.5.2.2. OpenShift 기본값 사용자 정의

openshift-logging 모드에서는 특정 속성이 필요하며 OpenShift 함수에서 역할로 인해 구성에서 제거할 수 없습니다. 성능에 영향을 주는 **경우** 레이블이 지정된 기타 속성이 비활성화될 수 있습니다.

사용자 정의 속성 없이 **openshift-logging** 모드를 사용하는 경우 OpenShift 툴과 즉시 호환성을 얻을 수 있습니다. 스트림 레이블 또는 구조화된 메타데이터로 추가 속성이 필요한 경우 사용자 지정 구성을 사용합니다. 사용자 지정 구성은 기본 구성과 병합할 수 있습니다.

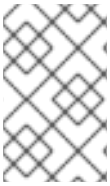
2.5.2.3. 권장 속성 제거

openshift-logging 모드에서 기본 속성을 줄이려면 권장 속성을 비활성화합니다.

```
# ...
spec:
  tenants:
    mode: openshift-logging
```

```
openshift:
  otlp:
    disableRecommendedAttributes: true (1)
```

(1) 기본 속성을 **필수** 속성으로 제한하는 권장 속성을 제거하려면 **disableRecommendedAttributes: true** 를 설정합니다.



참고

이 옵션은 기본 속성으로 인해 성능 또는 스토리지 문제가 발생하는 경우 유용합니다. 이 설정은 기본 스트림 레이블을 제거하므로 쿼리 성능에 부정적인 영향을 미칠 수 있습니다. 쿼리에 필수 속성을 유지하려면 이 옵션을 사용자 지정 특성 구성과 페어링해야 합니다.

2.5.3. 추가 리소스

- Loki 라벨
- 구조화된 메타데이터
- OpenTelemetry attribute

2.6. OPENTELEMETRY 데이터 모델

이 문서에서는 로깅 6.1을 사용한 Red Hat OpenShift Logging의 OpenTelemetry 지원에 대한 프로토콜 및 의미 규칙에 대해 간단히 설명합니다.

중요

OTLP(OpenTelemetry Protocol) 출력 로그 전달자는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

2.6.1. 전달 및 수집 프로토콜

Red Hat OpenShift Logging은 OTLP 사양 을 사용하여 OpenTelemetry 끝점으로 로그를 수집하고 전달합니다. OTLP 인코딩, 전송 및 Telemetry 데이터를 제공합니다. 로그 스트림을 수집하는 OTLP endpont을 제공하는 Loki 스토리지를 배포할 수도 있습니다. 이 문서에서는 다양한 OpenShift 클러스터 소스에서 수집된 로그에 대한 의미 규칙을 정의합니다.

2.6.2. 의미 체계 규칙

이 솔루션의 로그 수집기는 다음 로그 스트림을 수집합니다.

- 컨테이너 로그
- 클러스터 노드 저널 로그
- 클러스터 노드 auditd 로그

- Kubernetes 및 OpenShift API 서버 로그
- OpenShift Virtual Network(OVN) 로그

OpenTelemetry 의미 체계 특성에 의해 정의된 의미 규칙에 따라 이러한 스트림을 전달할 수 있습니다. OpenTelemetry의 의미 체계 규칙은 속성을 통해 식별되는 원격 분석을 생성하는 엔티티의 불변 표현으로 리소스를 정의합니다. 예를 들어 컨테이너에서 실행되는 프로세스에는 **container_name**,**cluster_id**,**pod_name**,**namespace**, **deployment** 또는 **app_name** 과 같은 특성이 포함됩니다. 이러한 속성은 리소스 오브젝트 아래에 그룹화되므로 반복을 줄이고 로그 전송을 원격 분석 데이터로 최적화할 수 있습니다.

리소스 속성 외에도 로그에는 각 로그 항목과 관련된 조정 라이브러리 및 로그 속성과 관련된 범위 속성이 포함될 수 있습니다. 이러한 속성은 각 로그 항목에 대해 자세히 설명하고 스토리지의 로그를 쿼리할 때 필터링 기능을 향상시킵니다.

다음 섹션에서는 일반적으로 전달되는 속성을 정의합니다.

2.6.2.1. 로그 항목 구조

모든 로그 스트림에는 다음 로그 데이터 필드가 포함됩니다.

적용 가능한 소스 열은 각 필드가 적용되는 로그 소스를 나타냅니다.

- **all**: 이 필드는 모든 로그에 있습니다.
- **컨테이너**: 이 필드는 애플리케이션 및 인프라 둘 다에 Kubernetes 컨테이너 로그에 있습니다.
- **audit**: 이 필드는 Kubernetes, OpenShift API 및 OVN 로그에 있습니다.
- **auditd**: 이 필드는 노드 auditd 로그에 있습니다.
- **저널**: 이 필드는 노드 저널 로그에 있습니다.

이름	적용 가능한 소스	댓글
body	all	
observedTimeUnixNano	all	
timeUnixNano	all	
severityText	컨테이너, 저널	
속성	all	(선택 사항) 스트림 특정 특성을 전달할 때 우선순위

2.6.2.2. 속성

로그 항목에는 다음 표에 설명된 대로 소스 기반의 리소스, 범위 및 로그 속성 세트가 포함됩니다.

Location 열은 특성 유형을 지정합니다.

- **resource**: resource 특성을 나타냅니다.

- **scope**: scope 특성을 나타냅니다.
- **log**: 로그 특성을 나타냅니다.

스토리지 열은 기본 **openshift-logging** 모드를 사용하여 특성이 LokiStack에 저장되는지 여부를 나타내며 특성이 저장된 위치를 지정합니다.

- **스트림 레이블**:
  - 특정 레이블을 기반으로 효율적으로 필터링 및 쿼리를 수행할 수 있습니다.
  - Loki Operator가 구성에 이 속성을 적용하는 경우 **필요에 따라** 레이블을 지정할 수 있습니다.
- **구조화된 메타데이터**:
  - 키-값 쌍의 자세한 필터링 및 스토리지를 허용합니다.
  - 사용자가 JSON 구문 분석 없이도 직접 라벨을 사용하여 간소화된 쿼리에 사용할 수 있습니다.

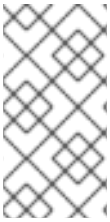
OTLP를 사용하면 사용자가 JSON 구문 분석을 사용하는 대신 레이블로 직접 쿼리를 필터링하여 쿼리의 속도와 효율성을 개선할 수 있습니다.

이름	위치	적용 가능한 소스	스토리지 (LokiStack)	댓글
log_source	resource	all	필수 스트림 레이블	(DEPRECATED) 호환성 속성에 **openshift.log.source**와 동일한 정보가 포함되어 있습니다.
log_type	resource	all	필수 스트림 레이블	(DEPRECATED) 호환성 속성에 **openshift.log.type**과 동일한 정보가 포함되어 있습니다.
kubernetes.container_name	resource	container	스트림 레이블	(DEPRECATED) 호환성 속성에 **k8s.container.name**과 동일한 정보가 포함되어 있습니다.
kubernetes.host	resource	all	스트림 레이블	(DEPRECATED) 호환성 속성에 **k8s.node.name**과 동일한 정보가 포함되어 있습니다.

이름	위치	적용 가능한 소스	스토리지 (LokiStack)	댓글
kubernetes.namespace_name	resource	container	필수 스트림 레이블	(DEPRECATED) 호환성 속성에 **k8s.namespace.name**과 동일한 정보가 포함되어 있습니다.
kubernetes.pod_name	resource	container	스트림 레이블	(DEPRECATED) 호환성 속성에 **k8s.pod.name**과 동일한 정보가 포함되어 있습니다.
openshift.cluster_id	resource	all		(DEPRECATED) 호환성 속성에 **openshift.cluster.uid**와 동일한 정보가 포함되어 있습니다.
level	log	컨테이너, 저널		(DEPRECATED) 호환성 속성에는 **심각도 Cryostat와 동일한 정보가 포함되어 있습니다.**
openshift.cluster.uid	resource	all	필수 스트림 레이블	
openshift.log.source	resource	all	필수 스트림 레이블	
openshift.log.type	resource	all	필수 스트림 레이블	
openshift.labels.*	resource	all	구조화된 메타데이터	
k8s.node.name	resource	all	스트림 레이블	
k8s.namespace.name	resource	container	필수 스트림 레이블	
k8s.container.name	resource	container	스트림 레이블	

이름	위치	적용 가능한 소스	스토리지 (LokiStack)	댓글
k8s.pod.labels.*	resource	container	구조화된 메타데이터	
k8s.pod.name	resource	container	스트림 레이블	
k8s.pod.uid	resource	container	구조화된 메타데이터	
k8s.cronjob.name	resource	container	스트림 레이블	Pod 작성자를 기반으로 조건부 전달
k8s.daemonset.name	resource	container	스트림 레이블	Pod 작성자를 기반으로 조건부 전달
k8s.deployment.name	resource	container	스트림 레이블	Pod 작성자를 기반으로 조건부 전달
k8s.job.name	resource	container	스트림 레이블	Pod 작성자를 기반으로 조건부 전달
k8s.replicaset.name	resource	container	구조화된 메타데이터	Pod 작성자를 기반으로 조건부 전달
k8s.statefulset.name	resource	container	스트림 레이블	Pod 작성자를 기반으로 조건부 전달
log.iostream	log	container	구조화된 메타데이터	
k8s.audit.event.level	log	audit	구조화된 메타데이터	
k8s.audit.event.stage	log	audit	구조화된 메타데이터	
k8s.audit.event.user_agent	log	audit	구조화된 메타데이터	
k8s.audit.event.request.uri	log	audit	구조화된 메타데이터	
k8s.audit.event.response.code	log	audit	구조화된 메타데이터	
k8s.audit.event.annotation.*	log	audit	구조화된 메타데이터	

이름	위치	적용 가능한 소스	스토리지 (LokiStack)	댓글
k8s.audit.event.object_ref.resource	log	audit	구조화된 메타데이터	
k8s.audit.event.object_ref.name	log	audit	구조화된 메타데이터	
k8s.audit.event.object_ref.namespace	log	audit	구조화된 메타데이터	
k8s.audit.event.object_ref.api_group	log	audit	구조화된 메타데이터	
k8s.audit.event.object_ref.api_version	log	audit	구조화된 메타데이터	
k8s.user.username	log	audit	구조화된 메타데이터	
k8s.user.groups	log	audit	구조화된 메타데이터	
process.executable.name	resource	journal	구조화된 메타데이터	
process.executable.path	resource	journal	구조화된 메타데이터	
process.command_line	resource	journal	구조화된 메타데이터	
process.pid	resource	journal	구조화된 메타데이터	
service.name	resource	journal	스트림 레이블	
systemd.t.*	log	journal	구조화된 메타데이터	
systemd.u.*	log	journal	구조화된 메타데이터	



참고

호환성 특성으로 표시된 속성은 ViaQ 데이터 모델과의 이전 버전과 최소 호환성을 지원합니다. 이러한 속성은 더 이상 사용되지 않으며 지속적인 UI 기능을 보장하기 위해 호환성 계층으로 작동합니다. 이러한 속성은 Logging UI가 향후 릴리스에서 OpenTelemetry 카운터를 완전히 지원할 때까지 계속 지원됩니다.

Loki는 스토리지로 유지할 때 속성 이름을 변경합니다. 이름은 소문자로 표시되고 집합의 모든 문자(,/,-)는 밑줄(_)으로 대체됩니다. 예를 들어 **k8s.namespace.name** 은 **k8s_namespace_name** 이 됩니다.

2.6.3. 추가 리소스

- 의미 체계
- 로그 데이터 모델
- 일반 로그 속성

2.7. 로깅 시각화

로깅을 위한 시각화는 Operator 설치가 필요한 Cluster Observability Operator 의 로깅 UI 플러그인 을 배포하여 제공됩니다.

중요

Red Hat은 현재 기술 프리뷰 (TP)에 있는 Cluster Observability Operator (COO)의 GA(General Availability) 릴리스까지 OpenShift Container Platform 4.14 이상에서 Logging UI 플러그인에 대해 Logging 6.0 이상을 사용하는 고객에게 지원을 제공합니다. COO에는 여전히 TP 기능이지만 로깅 UI 플러그인은 GA에 사용할 준비가 된 몇 가지 독립적인 기능이 포함되어 있기 때문에 이러한 지원 예외는 일시적입니다.