# OpenShift Container Platform 4.11

## 네트워크 관찰 기능

Network Observability Operator

Last Updated: 2024-02-09

# OpenShift Container Platform 4.11 네트워크 관찰 기능

Network Observability Operator

## 법적 공지



## 초록

이 문서에서는 OpenShift Container Platform 클러스터의 네트워크 트래픽 흐름을 관찰하고 분석하는 데 사용할 수 있는 Network Observability Operator를 사용하는 방법을 설명합니다.

# 차례


**1장.** ..... **4**
1.1. NETWORK OBSERVABILITY OPERATOR 1.4.2 4
1.2. NETWORK OBSERVABILITY OPERATOR 1.4.1 4
1.3. NETWORK OBSERVABILITY OPERATOR 1.4.0 4
1.4. NETWORK OBSERVABILITY OPERATOR 1.3.0 5
1.5. NETWORK OBSERVABILITY OPERATOR 1.2.0 6
1.6. NETWORK OBSERVABILITY OPERATOR 1.1.0 7

**2장.** ..... **8**
2.1. 8
2.2. 8
2.3. 8

**3장.** ..... **9**
3.1. 9
3.2. 9
3.3. 13
3.4. 14
3.5. KAFKA 설치 (선택 사항) 14
3.6. NETWORK OBSERVABILITY OPERATOR 설치 제거 15

**4장. OPENSHIFT CONTAINER PLATFORM의 NETWORK OBSERVABILITY OPERATOR** ..... **16**
4.1. 상태 보기 16
4.2. NETWORK OBSERVABLITY OPERATOR 아키텍처 17
4.3. NETWORK OBSERVABILITY OPERATOR 상태 및 구성 보기 18

**5장. NETWORK OBSERVABILITY OPERATOR 구성** ..... **20**
5.1. FLOWCOLLECTOR 리소스 보기 20
5.2. KAFKA를 사용하여 FLOW COLLECTOR 리소스 구성 22
5.3. 보강된 네트워크 흐름 데이터 내보내기 23
5.4. 흐름 수집기 리소스 업데이트 24
5.5. 빠른 필터 구성 24
5.6. SR-IOV 인터페이스 트래픽에 대한 모니터링 구성 25
5.7. 리소스 관리 및 성능 고려 사항 26

**6장. NETWORK POLICY** ..... **29**
6.1. 네트워크 OBSERVABILITY에 대한 네트워크 정책 생성 29
6.2. 네트워크 정책의 예 29

**7장. 네트워크 트래픽 관찰** ..... **31**
7.1. 개요 보기에서 네트워크 트래픽 관찰 31
7.2. 트래픽 흐름 보기에서 네트워크 트래픽 관찰 32
7.3. 토폴로지 보기에서 네트워크 트래픽 관찰 35
7.4. 네트워크 트래픽 필터링 36

**8장. NETWORK OBSERVABILITY OPERATOR 모니터링** ..... **38**
8.1. 상태 정보 보기 38
8.2. NETOBSERV 대시보드에 대한 LOKI 속도 제한 경고 생성 39

**9장. FLOWCOLLECTOR 구성 매개변수** ..... **40**
9.1. FLOWCOLLECTOR API 사양 40

**10장. 네트워크 흐름 형식 참조** ..... **79**
10.1. 네트워크 흐름 형식 참조 79

**11장. 네트워크 OBSERVABILITY 문제 해결 ..... 95**
11.1. MUST-GATHER 툴 사용 95
11.2. OPENSHIFT CONTAINER PLATFORM 콘솔에서 네트워크 트래픽 메뉴 항목 구성 95
11.3. FLOWLOGS-PIPELINE은 KAFKA를 설치한 후 네트워크 흐름을 사용하지 않습니다. 98
11.4. BR-INT 및 BR-EX 인터페이스 모두에서 네트워크 흐름을 볼 수 없음 98
11.5. NETWORK OBSERVABILITY 컨트롤러 관리자 POD가 메모리 부족 99
11.6. LOKI RESOURCEEXHAUSTED 오류 문제 해결 100
11.7. 리소스 문제 해결 101
11.8. LOKISTACK 속도 제한 오류 101

# 1장.

## 1.1. NETWORK OBSERVABILITY OPERATOR 1.4.2

-

### 1.1.1. CVE

- 2023-39325
- 2023-44487

## 1.2. NETWORK OBSERVABILITY OPERATOR 1.4.1

-

### 1.2.1. CVE

- 2023-44487
- 2023-39325
- 2023-29406
- 2023-29409
- 2023-39322
- 2023-39318
- 2023-39319
- 2023-39321

### 1.2.2. 버그 수정

- (**NETOBSERV-926**)
- (**NETOBSERV-1344**)

## 1.3. NETWORK OBSERVABILITY OPERATOR 1.4.0

- RHSA-2023:5379 Network Observability Operator 1.4.0

### 1.3.1.

### 1.3.2. 새로운 기능 및 개선 사항

#### 1.3.2.1. 주요 개선 사항

-

- 
- 
  - 
  - 
- 
- 

1.3.2.2.

1.3.2.3.

1.3.2.4.

1.3.2.5.

1.3.2.6.

### 1.3.3. 버그 수정

- (**NETOBSERV-1131**)
- (**NETOBSERV-1283**)
- (**NETOBSERV-1224**)
- (**NETOBSERV-975**)
- (NETOBSERV-1087)

### 1.3.4. 확인된 문제

- (**NETOBSERV-980**)
- (**NETOBSERV-1304**)
- (**NETOBSERV-1245**)
- (**NETOBSERV-926**)
- (**NETOBSERV-1293**

## 1.4. NETWORK OBSERVABILITY OPERATOR 1.3.0

- RHSA-2023:3905 Network Observability Operator 1.3.0

1.4.1.

### 1.4.2. 새로운 기능 및 개선 사항

#### 1.4.2.1.

- 

#### 1.4.2.2.

- 

#### 1.4.2.3.

- 

#### 1.4.2.4.

- 

### 1.4.3. 더 이상 사용되지 않는 기능

#### 1.4.3.1.

### 1.4.4. 버그 수정

- (NETOBSERV-1003)
- (NETOBSERV-976)
- (NETOBSERV-972)
- (NETOBSERV-971)
- (NETOBSERV-765)
- (NETOBSERV-1070)
- (NETOBSERV-773)
- (NETOBSERV-934)

### 1.4.5. 확인된 문제

- (NETOBSERV-1087)
- 

## 1.5. NETWORK OBSERVABILITY OPERATOR 1.2.0

- RHSA-2023:1817 Network Observability Operator 1.2.0

#### 1.5.1.

### 1.5.2. 새로운 기능 및 개선 사항

#### 1.5.2.1.

-

#### 1.5.2.2.

-

#### 1.5.2.3.

-

### 1.5.3. 버그 수정

- (NETOBSERV-774)
- (NETOBSERV-772)
- (NETOBSERV-755)
- (NETOBSERV-696)
- (NETOBSERV-617)

### 1.5.4. 알려진 문제

- (NETOBSERV-980)

### 1.5.5. 주요 기술 변경 사항

- (NETOBSERV-907)(NETOBSERV-956)

## 1.6. NETWORK OBSERVABILITY OPERATOR 1.1.0

-

### 1.6.1.

- (BZ#2169468)

# 2장.

## 2.1.

- 
- 
- 

## 2.2.

## 2.3.

### 2.3.1.

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 프로세서
- Operator

### 2.3.2.

### 2.3.3.

# 3장.

 참고

## 3.1.

표 3.1.

| | | |
|---|---|---|
| **내보내기** | ✓ | ✓ |
| | ✓ | ✓ |
| | ✓ | ✗ |
| | ✓ | ✗ |
| | ✓ | ✗ |

추가 리소스

- 

## 3.2.

사전 요구 사항

- 
- OpenShift Container Platform 4.10+
- Linux Kernel 4.18+

프로세스

1. OpenShift Container Platform 웹 콘솔에서 **Operator → OperatorHub**를 클릭합니다.
2. 
3. 

검증

1. 
2. 

 중요

### 3.2.1.

1.

2.

```
apiVersion: v1
kind: Secret
metadata:
  name: loki-s3
  namespace: netobserv 1
stringData:
  access_key_id: QUtJQUlPU0ZPRE5ON0VYQU1QTEUK
  access_key_secret:
d0phbHJYVXRuRkVNSS9LN01ERU5HL2JQeFJmaUNZRVhBTVBMRUtFWQo=
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

1

검증

-

추가 리소스

- 흐름 수집기 API 참조
- 흐름 수집기 샘플 리소스
-

### 3.2.2.

중요

프로세스

1.

2.

3.

4.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: loki
  namespace: netobserv 1
spec:
  size: 1x.small
  storage:
    schemas:
    - version: v12
```

```
    effectiveDate: '2022-06-01'
  secret:
    name: loki-s3
    type: s3
storageClassName: gp3 2
tenants:
  mode: openshift-network
```

1
2

중요

5. 생성을 클릭합니다.

### 3.2.2.1.

참고

표 3.2.

| 1x.small | | | |
|---|---|---|---|
| | | | |
| | | | |
| | 없음 | 2 | 3 |
| | | | |
| | | 63Gi | 139Gi |
| | 150Gi | 300Gi | 450Gi |

추가 리소스

- 

### 3.2.3.

```
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 40
        ingestionRate: 20
        maxGlobalStreamsPerTenant: 25000
      queries:
        maxChunksPerQuery: 2000000
        maxEntriesLimitPerQuery: 10000
        maxQuerySeries: 3000
```

### 3.2.4.

프로세스

1.

2.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: netobserv-reader
rules:
- apiGroups:
  - 'loki.grafana.com'
  resources:
  - network
  resourceNames:
  - logs
  verbs:
  - 'get'
```

1

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: netobserv-writer
rules:
- apiGroups:
  - 'loki.grafana.com'
  resources:
  - network
  resourceNames:
  - logs
  verbs:
  - 'create'
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: netobserv-writer-flp
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: netobserv-writer
subjects:
- kind: ServiceAccount
  name: flowlogs-pipeline
  namespace: netobserv
- kind: ServiceAccount
  name: flowlogs-pipeline-transformer
  namespace: netobserv
```

1

### 3.2.5.

사전 요구 사항

- 
- 
- 

프로세스

1. **$ oc adm policy add-cluster-role-to-user netobserv-reader user1**

## 3.3.

 중요

사전 요구 사항

- 
- **cluster-admin** 권한이 있어야 합니다.
- 
- 
- 

 참고

프로세스

1. **OpenShift Container Platform** 웹 콘솔에서 **Operator → OperatorHub**를 클릭합니다.
2. 
3. 
4. **Operators → 설치된 Operator**로 이동합니다.
5. a.
   b. i.
      ii.
      iii.
      iv.

v.

vi.

c.

d.

e. 생성을 클릭합니다.

검증

중요

## 3.4.

- 
- 
- 
- 
- 

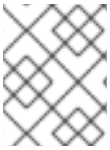
추가 리소스

**Flow Collector** 사양 및 **Network Observability Operator** 아키텍처 및 리소스 사용에 대한 자세한 내용은 다음 리소스를 참조하십시오.

- 흐름 수집기 **API** 참조
- 흐름 수집기 샘플 리소스
- 리소스 고려 사항
- **Network Observability** 컨트롤러 관리자 **Pod** 문제 해결
- 네트워크 **Observability** 아키텍처

## 3.5. KAFKA 설치 (선택 사항)

**Kafka Operator**는 대규모 환경에서 지원됩니다. **Kafka**는 보다 탄력적이고 확장 가능한 방식으로 네트워크 흐름 데이터를 전달하기 위해 높은 처리량과 대기 시간이 짧은 데이터 피드를 제공합니다. **Loki Operator** 및 **Network Observability Operator**가 설치된 것처럼 **Operator Hub**에서 **Kafka Operator**를 **Red Hat AMQ Streams** 로 설치할 수 있습니다. **Kafka**를 스토리지 옵션으로 구성하려면 " **FlowCollector** 리소스 구성"을 참조하십시오.



참고

**Kafka**를 설치 제거하려면 설치하는 데 사용한 방법과 일치하는 제거 프로세스를 참조하십시오.

추가 리소스

Kafka를 사용하여 FlowCollector 리소스 구성.

## 3.6. NETWORK OBSERVABILITY OPERATOR 설치 제거

OpenShift Container Platform 웹 콘솔 Operator Hub를 사용하여 Network Observability Operator를 설치 제거하고 Operator → 설치된 Operator 영역에서 작업할 수 있습니다.

프로세스

1. FlowCollector 사용자 지정 리소스를 제거합니다.
   a. 제공된 API 열에서 Network Observability Operator 옆에 있는 흐름 수집기 를 클릭합니다.
   b. 클러스터 의 옵션 메뉴 를 클릭하고 FlowCollector 삭제 를 선택합니다.
2. Network Observability Operator를 설치 제거합니다.
   a. Operator → 설치된 Operator 영역으로 돌아갑니다.
   b. Network Observability Operator 옆에 있는 옵션 메뉴 를 클릭하고 Operator 설치 제거 를 선택합니다.
   c. 홈 → 프로젝트 및 openshift-netobserv-operator선택
   d. 작업으로 이동하여 프로젝트 삭제를 선택합니다.
3. FlowCollector CRD(사용자 정의 리소스 정의)를 제거합니다.
   a. 관리 → 클러스터 리소스 정의로 이동합니다.
   b. FlowCollector 를 찾고 옵션 메뉴를 클릭합니다.
   c. Delete CustomResourceDefinition 을 선택합니다.

중요

Loki Operator 및 Kafka는 설치된 경우 그대로 유지되며 별도로 제거해야 합니다. 또한 오브젝트 저장소에 남아 있는 데이터와 제거해야 하는 영구 볼륨이 있을 수 있습니다.

# 4장. OPENSHIFT CONTAINER PLATFORM의 NETWORK OBSERVABILITY OPERATOR

Network Observability는 네트워크 Observability eBPF 에이전트에서 생성하는 네트워크 트래픽 흐름을 수집하고 보강하기 위해 모니터링 파이프라인을 배포하는 OpenShift Operator입니다.

## 4.1. 상태 보기

Network Observability Operator는 Flow Collector API를 제공합니다. 흐름 수집기 리소스가 생성되면 Pod 및 서비스를 배포하여 Loki 로그 저장소에 네트워크 흐름을 생성 및 저장하고 OpenShift Container Platform 웹 콘솔에서 대시보드, 메트릭 및 흐름을 표시합니다.

프로세스

1. 다음 명령을 실행하여 **FlowCollector** 의 상태를 확인합니다.

```
$ oc get flowcollector/cluster
```

출력 예

```
NAME      AGENT   SAMPLING (EBPF)   DEPLOYMENT MODEL   STATUS
cluster   EBPF    50                DIRECT             Ready
```

2. 다음 명령을 입력하여 **netobserv** 네임스페이스에서 실행 중인 Pod의 상태를 확인합니다.

```
$ oc get pods -n netobserv
```

출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
flowlogs-pipeline-56hbp             1/1     Running   0          147m
flowlogs-pipeline-9plvv             1/1     Running   0          147m
flowlogs-pipeline-h5gkb             1/1     Running   0          147m
flowlogs-pipeline-hh6kf             1/1     Running   0          147m
flowlogs-pipeline-w7vv5             1/1     Running   0          147m
netobserv-plugin-cdd7dc6c-j8ggp     1/1     Running   0          147m
```

**FlowLogs-pipeline** 포드는 흐름을 수집하고 수집된 흐름을 강화한 다음 Loki 스토리지로 흐름을 보냅니다. **NetObserv-plugin** Pod는 OpenShift Container Platform 콘솔의 시각화 플러그인을 생성합니다.

1. 다음 명령을 입력하여 네임스페이스 **netobserv-privileged** 에서 실행 중인 Pod의 상태를 확인합니다.

```
$ oc get pods -n netobserv-privileged
```

출력 예

```
NAME                          READY   STATUS    RESTARTS   AGE
netobserv-ebpf-agent-4lpp6    1/1     Running   0          151m
netobserv-ebpf-agent-6gbrk    1/1     Running   0          151m
```

```
netobserv-ebpf-agent-klpl9   1/1     Running   0          151m
netobserv-ebpf-agent-vrcnf   1/1     Running   0          151m
netobserv-ebpf-agent-xf5jh   1/1     Running   0          151m
```

**NetObserv-ebpf-agent Pod**는 노드의 네트워크 인터페이스를 모니터링하여 흐름을 가져오고**flowlogs-pipeline pod**로 보냅니다.

1. **Loki Operator**를 사용하는 경우 다음 명령을 입력하여 **openshift-operators-redhat** 네임스페이스에서 실행중인 **Pod**의 상태를 확인합니다.

```
$ oc get pods -n openshift-operators-redhat
```

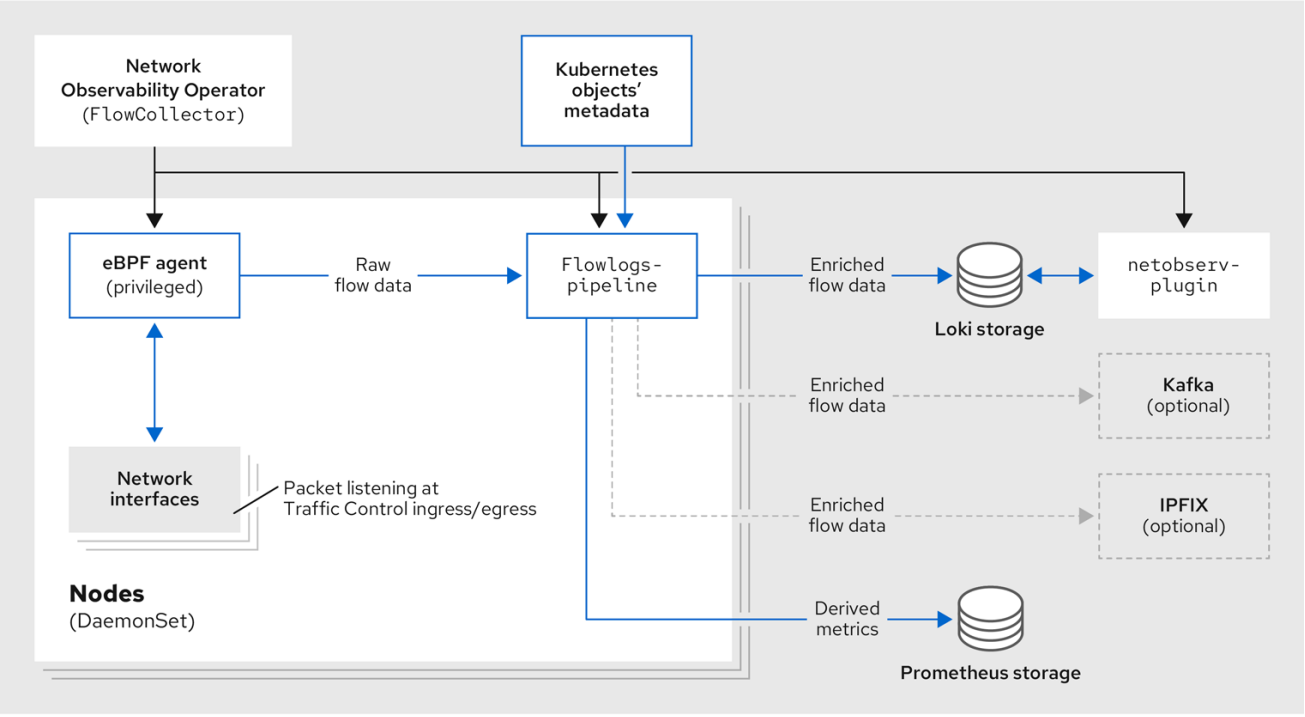
출력 예

```
NAME                                          READY   STATUS    RESTARTS   AGE
loki-operator-controller-manager-5f6cff4f9d-jq25h   2/2     Running   0          18h
lokistack-compactor-0                         1/1     Running   0          18h
lokistack-distributor-654f87c5bc-qhkhv        1/1     Running   0          18h
lokistack-distributor-654f87c5bc-skxgm        1/1     Running   0          18h
lokistack-gateway-796dc6ff7-c54gz             2/2     Running   0          18h
lokistack-index-gateway-0                     1/1     Running   0          18h
lokistack-index-gateway-1                     1/1     Running   0          18h
lokistack-ingester-0                          1/1     Running   0          18h
lokistack-ingester-1                          1/1     Running   0          18h
lokistack-ingester-2                          1/1     Running   0          18h
lokistack-querier-66747dc666-6vh5x            1/1     Running   0          18h
lokistack-querier-66747dc666-cjr45            1/1     Running   0          18h
lokistack-querier-66747dc666-xh8rq            1/1     Running   0          18h
lokistack-query-frontend-85c6db4fbd-b2xfb     1/1     Running   0          18h
lokistack-query-frontend-85c6db4fbd-jm94f     1/1     Running   0          18h
```

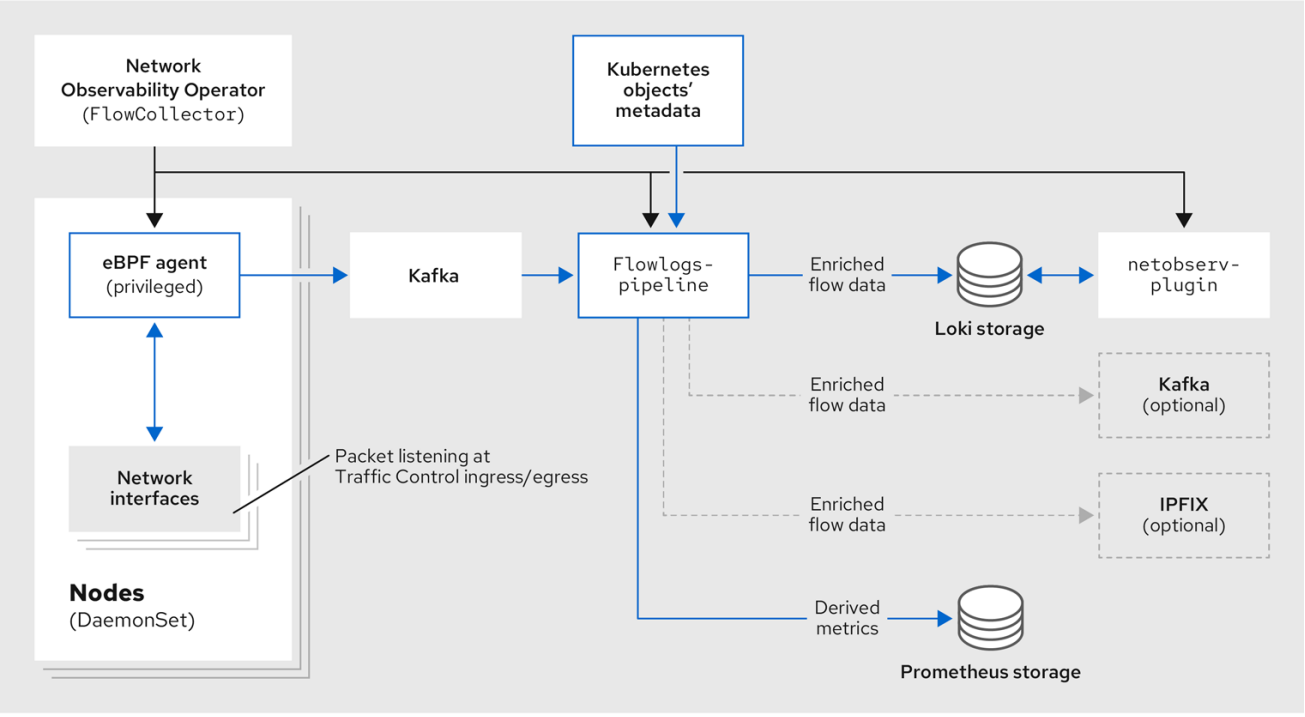
## 4.2. NETWORK OBSERVABLITY OPERATOR 아키텍처

**Network Observability Operator**는 설치 시 인스턴스화되고 **eBPF** 에이전트**, flowlogs-pipeline** 및 **netobserv-plugin** 구성 요소를 조정하도록 구성된**FlowCollector API**를 제공합니다. 클러스터당 하나의 **FlowCollector** 만 지원됩니다.

**eBPF** 에이전트 는 네트워크 흐름을 수집할 수 있는 일부 권한이 있는 각 클러스터 노드에서 실행됩니다. **flowlogs-pipeline** 은 네트워크 흐름 데이터를 수신하고 **Kubernetes** 식별자를 사용하여 데이터를 보강합니다. **Loki**를 사용하는 경우 **flowlogs-pipeline** 은 저장 및 인덱싱을 위해 흐름 로그 데이터를 **Loki**로 보냅니다. 동적 **OpenShift Container Platform** 웹 콘솔 플러그인인 **netobserv-plugin** 은 **Loki**를 쿼리하여 네트워크 흐름 데이터를 가져옵니다. **cluster-admins**는 웹 콘솔에서 데이터를 볼 수 있습니다.




**Kafka** 옵션을 사용하는 경우 **eBPF** 에이전트는 네트워크 흐름 데이터를 **Kafka**로 전송하고 **flowlogs-pipeline** 은 다음 다이어그램에 표시된 대로 **Loki**로 보내기 전에 **Kafka** 주제에서 읽습니다.




## 4.3. NETWORK OBSERVABILITY OPERATOR 상태 및 구성 보기

**oc describe** 명령을 사용하여 상태를 검사하고**FlowCollector** 의 세부 정보를 볼 수 있습니다.

프로세스

1. 다음 명령을 실행하여 **Network Observability Operator**의 상태 및 구성을 확인합니다.

```
$ oc describe flowcollector/cluster
```

# 5장. NETWORK OBSERVABILITY OPERATOR 구성

Flow Collector API 리소스를 업데이트하여 Network Observability Operator 및 해당 관리 구성 요소를 구성할 수 있습니다. 설치 중에 Flow Collector가 명시적으로 생성됩니다. 이 리소스는 클러스터 전체에서 작동하기 때문에 단일 **FlowCollector** 만 허용되며 **cluster** 라는 이름을 지정해야 합니다.

## 5.1. FLOWCOLLECTOR 리소스 보기

OpenShift Container Platform 웹 콘솔에서 직접 YAML을 보고 편집할 수 있습니다.

프로세스

1. 웹 콘솔에서 Operator → 설치된 Operator 로 이동합니다.
2. NetObserv Operator 의 제공된 API 제목에서 흐름 수집기 를 선택합니다.
3. 클러스터를 선택한 다음 YAML 탭을 선택합니다. 여기에서 **FlowCollector** 리소스를 수정하여 Network Observability Operator를 구성할 수 있습니다.

다음 예제에서는 OpenShift Container Platform Network Observability Operator의 샘플 **FlowCollector** 리소스를 보여줍니다.

샘플 **FlowCollector** 리소스

```
apiVersion: flows.netobserv.io/v1beta1
kind: FlowCollector
metadata:
  name: cluster
spec:
  namespace: netobserv
  deploymentModel: DIRECT
  agent:
    type: EBPF                    (1)
    ebpf:
      sampling: 50                (2)
      logLevel: info
      privileged: false
      resources:
        requests:
          memory: 50Mi
          cpu: 100m
        limits:
          memory: 800Mi
  processor:
    logLevel: info
    resources:
      requests:
        memory: 100Mi
        cpu: 100m
      limits:
        memory: 800Mi
    conversationEndTimeout: 10s
    logTypes: FLOWS               (3)
    conversationHeartbeatInterval: 30s
```

```
  loki:                                4
    url: 'https://loki-gateway-http.netobserv.svc:8080/api/logs/v1/network'
    statusUrl: 'https://loki-query-frontend-http.netobserv.svc:3100/'
    authToken: FORWARD
    tls:
      enable: true
      caCert:
        type: configmap
        name: loki-gateway-ca-bundle
        certFile: service-ca.crt
        namespace: loki-namespace        # 5
  consolePlugin:
    register: true
    logLevel: info
    portNaming:
      enable: true
      portNames:
        "3100": loki
    quickFilters:                        6
    - name: Applications
      filter:
        src_namespace!: 'openshift-,netobserv'
        dst_namespace!: 'openshift-,netobserv'
      default: true
    - name: Infrastructure
      filter:
        src_namespace: 'openshift-,netobserv'
        dst_namespace: 'openshift-,netobserv'
    - name: Pods network
      filter:
        src_kind: 'Pod'
        dst_kind: 'Pod'
      default: true
    - name: Services network
      filter:
        dst_kind: 'Service'
```

1 에이전트 사양인 **spec.agent.type** 은 CryostatPF 여야 합니다. eBPF는 지원되는 유일한 OpenShift Container Platform 옵션입니다.

2 Sampling 사양인 **spec.agent.ebpf.sampling** 을 설정하여 리소스를 관리할 수 있습니다. 샘플링 값이 작으면 많은 양의 컴퓨팅, 메모리 및 스토리지 리소스를 사용할 수 있습니다. 샘플링 비율 값을 지정하여 이를 완화할 수 있습니다. 값이 100이면 100개마다 하나의 흐름이 샘플링됩니다. 값이 0 또는 1이면 모든 흐름이 캡처됩니다. 값이 낮을수록 반환된 흐름의 증가 및 파생 메트릭의 정확도가 낮아집니다. 기본적으로 eBPF 샘플링은 값 50으로 설정되므로 50마다 하나의 흐름이 샘플링됩니다. 더 많은 샘플 흐름은 더 많은 스토리지가 필요하다는 것을 의미합니다. 클러스터를 관리할 수 있는 설정을 결정하려면 기본값으로 시작하고 경험적으로 구체화하는 것이 좋습니다.

3 선택적 **spec.processor.logTypes**,**spec.processor.conversationHeartbeatInterval** 및 **spec.processor.conversationEndTimeout** 을 설정하여 대화 추적을 활성화할 수 있습니다. 활성화하면 웹 콘솔에서 대화 이벤트를 쿼리할 수 있습니다. **spec.processor.logTypes** 의 값은 다음과 같습니다. **FLOWS CONVERSATIONS**,**ENDED_CONVERSATIONS**, **ALL**. 스토리지 요구 사항은 **ENDED _ CONVERSATIONS**의 경우 best 및 lowest입니다.

4

Loki 사양인 **spec.loki** 는 Loki 클라이언트를 지정합니다. 기본값은 **Loki Operator** 설치 섹션에 언급된 Loki 설치 경로와 일치합니다. Loki에 다른 설치 방법을 사용한 경우 설치에 적절한 클라이언트 정보

5 원래 인증서는 **Network Observability** 인스턴스 네임스페이스에 복사되고 업데이트를 확인합니다. 제공되지 않는 경우 네임스페이스의 기본값은 **"spec.namespace"**와 동일합니다. 다른 네임스페이스에 Loki를 설치하도록 선택한 경우 **spec.loki.tls.caCert.namespace** 필드에 지정해야 합니다. 마찬가지로 **spec.exporters.kafka.tls.caCert.namespace** 필드는 다른 네임스페이스에 설치된 Kafka에 사용할 수 있습니다.

6 **spec.quickFilters** 사양은 웹 콘솔에 표시되는 필터를 정의합니다.애플리케이션 필터 키,**src_namespace** 및 **dst_namespace** 는 negated (**!**)이므로 애플리케이션 필터 는 ***openshift-*** 또는 ***netobserv*** *네임스페이스의 대상이 아닌*모든 트래픽을 표시합니다. 자세한 내용은 아래의 빠른 필터 구성을 참조하십시오.

추가 리소스

대화 추적에 대한 자세한 내용은 대화 작업을 참조하십시오.

## 5.2. KAFKA를 사용하여 FLOW COLLECTOR 리소스 구성

처리량이 높고 대기 시간이 짧은 데이터 피드에 Kafka를 사용하도록 **FlowCollector** 리소스를 구성할 수 있습니다. Kafka 인스턴스를 실행해야 하며 **OpenShift Container Platform Network Observability** 전용 Kafka 주제를 해당 인스턴스에서 생성해야 합니다. 자세한 내용은 AMQ Streams를 사용한 Kafka 문서를 참조하십시오.

사전 요구 사항

- Kafka가 설치되어 있어야 합니다. Red Hat은 AMQ Streams Operator를 사용하여 Kafka를 지원합니다.

프로세스

1. 웹 콘솔에서 **Operator → 설치된 Operator** 로 이동합니다.
2. **Network Observability Operator**의 제공된 **API** 제목에서 흐름 수집기 를 선택합니다.
3. 클러스터를 선택한 다음 **YAML** 탭을 클릭합니다.
4. 다음 샘플 YAML과 같이 **OpenShift Container Platform Network Observability Operator**의 **FlowCollector** 리소스를 수정하여 Kafka를 사용합니다.

**FlowCollector** 리소스의 Kafka 구성 샘플

```
apiVersion: flows.netobserv.io/v1beta1
kind: FlowCollector
metadata:
  name: cluster
spec:
  deploymentModel: KAFKA                          1
  kafka:
    address: "kafka-cluster-kafka-bootstrap.netobserv"   2
    topic: network-flows                          3
    tls:
      enable: false                               4
```

- 

1. Kafka 배포 모델을 활성화하려면 **spec.deploymentModel** 을 **DIRECT** 대신 **KAFKA** 로 설정합니다.
2. **spec.kafka.address** 는 Kafka 부트스트랩 서버 주소를 나타냅니다. 필요한 경우 포트 9093에서 **TLS** 를 사용하기 위해 **kafka-cluster-kafka-bootstrap.netobserv:9093** 과 같이 포트를 지정할 수 있습니다.
3. **spec.kafka.topic** 은 Kafka에서 생성된 주제의 이름과 일치해야 합니다.
4. **spec.kafka.tls** 는 TLS 또는 mTLS를 사용하여 Kafka와의 모든 통신을 암호화하는 데 사용할 수 있습니다. 활성화된 경우 Kafka CA 인증서를 ConfigMap 또는 Secret으로 사용할 수 있어야 합니다. **flowlogs-pipeline** 프로세서 구성 요소가 배포되는 네임스페이스(기본값:**netobserv**)와 **eBPF** 에이전트가 배포되는 위치(기본값: **netobserv-privileged**). **spec.kafka.tls.caCert** 를 사용하여 참조해야 합니다. mTLS를 사용하는 경우 이러한 네임스페이스에서 클라이언트 시크릿을 사용할 수 있어야 합니다(예: AMQ Streams User Operator를 사용하여 생성할 수 있음) **spec.kafka.tls.userCert** 에서 참조됩니다.

## 5.3. 보강된 네트워크 흐름 데이터 내보내기

Kafka, IPFIX 또는 둘 다에 동시에 네트워크 흐름을 보낼 수 있습니다. Splunk, Elasticsearch 또는 Fluentd 와 같이 Kafka 또는 IPFIX 입력을 지원하는 모든 프로세서 또는 스토리지는 강화된 네트워크 흐름 데이터를 사용할 수 있습니다.

사전 요구 사항

- Kafka 또는 IPFIX 수집기 끝점은 Network Observability **flowlogs-pipeline** Pod에서 사용할 수 있습니다.

프로세스

1. 웹 콘솔에서 **Operator → 설치된 Operator** 로 이동합니다.
2. NetObserv Operator 의 제공된 **API** 제목에서 **흐름 수집기** 를 선택합니다.
3. 클러스터를 선택한 다음**YAML** 탭을 선택합니다.
4. 다음과 같이 **FlowCollector** 를 편집하여**spec.exporters** 를 구성합니다.

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  exporters:
  - type: KAFKA                 1
    kafka:
      address: "kafka-cluster-kafka-bootstrap.netobserv"
      topic: netobserv-flows-export  2
      tls:
        enable: false           3
  - type: IPFIX                 4
    ipfix:
```

```
targetHost: "ipfix-collector.ipfix.svc.cluster.local"
targetPort: 4739
transport: tcp or udp
```
5

2 Network Observability Operator는 모든 흐름을 구성된 Kafka 주제로 내보냅니다.

3 SSL/TLS 또는 mTLS를 사용하여 Kafka 간에 모든 통신을 암호화할 수 있습니다. 활성화하면 Kafka CA 인증서를 ConfigMap 또는 Secret으로 사용할 수 있어야 합니다. **flowlogs-pipeline** 프로세서 구성 요소가 배포되는 네임스페이스(기본값: **netobserv**). **spec.exporters.tls.caCert** 를 사용하여 참조해야 합니다. mTLS를 사용하는 경우 이러한 네임스페이스에서 클라이언트 시크릿을 사용할 수 있어야 합니다(예: AMQ Streams User Operator를 사용하여 생성할 수 있음) **spec.exporters.tls.userCert** 에서 참조해야 합니다.

1 4 흐름을 Kafka로 내보내거나 내보내기와 함께 IPFIX로 내보낼 수 있습니다.

5 전송을 지정하는 옵션이 있습니다. 기본값은 **tcp** 이지만 **udp** 도 지정할 수 있습니다.

5. 구성 후 네트워크 흐름 데이터를 JSON 형식의 사용 가능한 출력으로 전송할 수 있습니다. 자세한 내용은 *네트워크 흐름 형식 참조*를참조하십시오.

추가 리소스

흐름 형식을 지정하는 방법에 대한 자세한 내용은 네트워크 흐름 형식 참조를 참조하십시오.

## 5.4. 흐름 수집기 리소스 업데이트

OpenShift Container Platform 웹 콘솔에서 YAML을 편집하는 대신 **flowcollector** CR(사용자 정의 리소스)을 패치하여 eBPF 샘플링과 같은 사양을 구성할 수 있습니다.

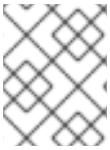
프로세스

1. 다음 명령을 실행하여 **flowcollector** CR을 패치하고**spec.agent.ebpf.sampling** 값을 업데이트합니다.

```
$ oc patch flowcollector cluster --type=json -p "[{"op": "replace", "path": "/spec/agent/ebpf/sampling", "value": <new value>}] -n netobserv"
```

## 5.5. 빠른 필터 구성

**FlowCollector** 리소스에서 필터를 수정할 수 있습니다. 정확한 일치는 값에 대한 double-quotes를 사용할 수 있습니다. 그렇지 않으면 부분 일치가 텍스트 값에 사용됩니다. 키 끝에 배치된 bang(!) 문자는 부정을 의미합니다. YAML 수정에 대한 자세한 내용은 샘플 **FlowCollector** 리소스를 참조하십시오.



참고

일치하는 필터 유형은 "all of" 또는 "any of"는 사용자가 쿼리 옵션에서 수정할 수 있는 UI 설정입니다. 이 리소스는 이 리소스 구성의 일부가 아닙니다.

다음은 사용 가능한 모든 필터 키 목록입니다.

표 5.1. 키 필터링

| Universal* | 소스 | 대상 | 설명 |
|---|---|---|---|
| 네임스페이스 | **src_namespace** | **dst_namespace** | 특정 네임스페이스와 관련된 트래픽을 필터링합니다. |
| name | **src_name** | **dst_name** | 특정 pod, 서비스 또는 노드(호스트 네트워크 트래픽의 경우)와 같은 지정된 리프 리소스 이름과 관련된 트래픽을 필터링합니다. |
| kind | **src_kind** | **dst_kind** | 지정된 리소스 유형과 관련된 트래픽을 필터링합니다. 리소스 종류에는 리프 리소스(Pod, 서비스 또는 노드) 또는 소유자 리소스(Deployment 및 StatefulSet)가 포함됩니다. |
| owner_name | **src_owner_name** | **dst_owner_name** | 지정된 리소스 소유자, 즉 워크로드 또는 Pod 세트와 관련된 트래픽을 필터링합니다. 예를 들어 배포 이름, StatefulSet 이름 등이 될 수 있습니다. |
| resource | **src_resource** | **dst_resource** | 고유하게 식별하는 표준 이름으로 표시되는 특정 리소스와 관련된 트래픽을 필터링합니다. canonical notation은 네임스페이스가 지정된 종류의 **kind.namespace.name** 또는 노드의 **node.name** 입니다. 예: **Deployment.my-namespace.my-web-server**. |
| address | **src_address** | **dst_address** | IP 주소와 관련된 트래픽을 필터링합니다. IPv4 및 IPv6이 지원됩니다. CIDR 범위도 지원됩니다. |
| mac | **src_mac** | **dst_mac** | MAC 주소와 관련된 트래픽을 필터링합니다. |
| port | **src_port** | **dst_port** | 특정 포트와 관련된 트래픽을 필터링합니다. |
| host_address | **src_host_address** | **dst_host_address** | Pod가 실행 중인 호스트 IP 주소와 관련된 트래픽을 필터링합니다. |
| 프로토콜 | 해당 없음 | 해당 없음 | TCP 또는 UDP와 같은 프로토콜과 관련된 트래픽을 필터링합니다. |

- 소스 또는 대상에 대해 **Universal keys filter for any of source or destination** 예를 들어, **name: 'my-pod '** 를 필터링하는 것은 모든 트래픽의 모든 트래픽을**my-pod** 및 **my-pod**로의 모든 트래픽을 의미합니다.

## 5.6. SR-IOV 인터페이스 트래픽에 대한 모니터링 구성

**SR-IOV(Single Root I/O Virtualization)** 장치가 있는 클러스터에서 트래픽을 수집하려면 **FlowCollector spec.agent.ebpf.privileged** 필드를 **true** 로 설정해야 합니다. 그런 다음 **eBPF** 에이전트는 기본적으로 모

니터링되는 호스트 네트워크 네임스페이스 외에도 다른 네트워크 네임스페이스를 모니터링합니다. **VF**(가상 기능) 인터페이스가 있는 **Pod**가 생성되면 새 네트워크 네임스페이스가 생성됩니다. **SRIOVNetwork** 정책 **IPAM** 구성을 지정하면 **VF** 인터페이스가 호스트 네트워크 네임스페이스에서 **Pod** 네트워크 네임스페이스로 마이그레이션됩니다.

사전 요구 사항

- SR-IOV 장치를 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.
- **SRIOVNetwork** CR(사용자 정의 리소스) **spec.ipam** 구성은 인터페이스에서 나열하거나 다른 플러그인에서 나열하는 범위의 IP 주소로 설정해야 합니다.

프로세스

1. 웹 콘솔에서 **Operator → 설치된 Operator** 로 이동합니다.
2. **NetObserv Operator** 의 제공된 **API** 제목에서 **흐름 수집기** 를 선택합니다.
3. 클러스터를 선택한 다음 **YAML** 탭을 선택합니다.
4. **FlowCollector** 사용자 지정 리소스를 구성합니다. 샘플 구성은 다음과 같습니다.

   SR-IOV 모니터링을 위한 **FlowCollector** 구성

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  namespace: netobserv
  deploymentModel: DIRECT
  agent:
    type: EBPF
    ebpf:
      privileged: true (1)
```

(1) SR-IOV 모니터링을 활성화하려면 **spec.agent.ebpf.privileged** 필드 값을 **true** 로 설정해야 합니다.

추가 리소스

**SriovNetwork** 사용자 정의 리소스 생성에 대한 자세한 내용은 CNI VRF 플러그인을 사용하여 추가 SR-IOV 네트워크 연결 생성을 참조하십시오.

## 5.7. 리소스 관리 및 성능 고려 사항

Network Observability에 필요한 리소스 양은 클러스터의 크기와 관찰성 데이터를 수집하고 저장하는 클러스터의 요구 사항에 따라 달라집니다. 리소스를 관리하고 클러스터의 성능 기준을 설정하려면 다음 설정을 구성하는 것이 좋습니다. 이러한 설정을 구성하면 최적의 설정 및 관찰 기능 요구 사항이 충족될 수 있습니다.

다음 설정을 사용하면 처음부터 리소스 및 성능을 관리하는 데 도움이 될 수 있습니다.

eBPF 샘플링

Sampling 사양인 **spec.agent.ebpf.sampling** 을 설정하여 리소스를 관리할 수 있습니다. 더 작은 샘플링 값은 많은 양의 컴퓨팅, 메모리 및 스토리지 리소스를 사용할 수 있습니다. 샘플링 비율 값을 지정하여 이를 완화할 수 있습니다. 값이 **100** 이면 100개마다 하나의 흐름이 샘플링됩니다. 값이**0** 또는**1** 이면 모든 흐름이 캡처됩니다. 값이 작으면 반환된 흐름이 증가하고 파생 메트릭의 정확도가 증가합니다. 기본적으로 **eBPF** 샘플링은 값 50으로 설정되므로 50마다 하나의 흐름이 샘플링됩니다. 더 많은 샘플 흐름은 더 많은 스토리지가 필요하다는 것을 의미합니다. 클러스터가 관리할 수 있는 설정을 결정하기 위해 기본값을 시작하고 실제적으로 구체화하는 것이 좋습니다.

인터페이스 제한 또는 제외

**spec.agent.ebpf.interfaces** 및 **spec.agent.ebpf.excludeInterfaces** 값을 설정하여 전체 관찰 트래픽을 줄입니다. 기본적으로 에이전트는 **excludeInterfaces** 및 **lo** (로컬 인터페이스)에 나열된 인터페이스를 제외하고 시스템의 모든 인터페이스를 가져옵니다. 인터페이스 이름은 사용된 **CNI(Container Network Interface)**에 따라 다를 수 있습니다.

다음 설정을 사용하여 잠시 동안 **Network Observability**가 실행된 후 성능을 미세 조정할 수 있습니다.

리소스 요구 사항 및 제한

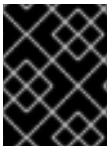
**spec.agent.ebpf.resources** 및 **spec.processor.resources** 사양을 사용하여 클러스터에서 예상되는 로드 및 메모리 사용량에 리소스 요구 사항 및 제한을 조정합니다. 대부분의 중간 크기의 클러스터에는 **800MB**의 기본 제한이 충분할 수 있습니다.

캐시 최대 흐름 시간 초과

**eBPF** 에이전트의 **spec.agent.ebpf.cacheMaxFlows** 및 **spec.agent.ebpf.cacheActiveTimeout** 사양을 사용하여 에이전트에서 보고하는 빈도를 제어합니다. 값이 클수록 에이전트가 더 적은 트래픽이 생성되므로 더 낮은 **CPU** 로드와 관련이 있습니다. 그러나 값이 클수록 메모리 사용량이 약간 길어지고 흐름 수집에서 대기 시간이 증가할 수 있습니다.

### 5.7.1. 리소스 고려 사항

다음 표에는 특정 워크로드 크기가 있는 클러스터의 리소스 고려 사항의 예가 요약되어 있습니다.



중요

표에 설명된 예제에서는 특정 워크로드에 맞는 시나리오를 보여줍니다. 각 예제를 워크로드 요구 사항을 충족하기 위해 조정할 수 있는 기준으로만 고려해 보십시오.

표 5.2. 리소스 권장 사항

| | 추가 소규모(10개 노드) | 소규모(25개 노드) | 중간(65 노드) [2] | 대규모(120개 노드) [2] |
|---|---|---|---|---|
| **작업자 노드 vCPU 및 메모리** | 4개의 vCPU| 16GiB mem [1] | 16개의 vCPU| 64GiB mem [1] | 16개의 vCPU| 64GiB mem [1] | 16개의 vCPU| 64GiB Mem [1] |
| LokiStack 크기 | **1x.extra-small** | **1x.small** | **1x.small** | **1x.medium** |
| **네트워크 Observability 컨트롤러 메모리 제한** | 400Mi(기본값) | 400Mi(기본값) | 400Mi(기본값) | 800Mi |
| **eBPF 샘플링 속도** | 50(기본값) | 50(기본값) | 50(기본값) | 50(기본값) |

| | 추가 소규모(10개 노드) | 소규모(25개 노드) | 중간(65 노드) [2] | 대규모(120개 노드) [2] |
|---|---|---|---|---|
| **eBPF 메모리 제한** | 800Mi (기본값) | 800Mi (기본값) | 2000Mi | 800Mi (기본값) |
| **CryostatP 메모리 제한** | 800Mi (기본값) | 800Mi (기본값) | 800Mi (기본값) | 800Mi (기본값) |
| **CryostatP Kafka 파티션** | 해당 없음 | 48 | 48 | 48 |
| **Kafka 소비자 복제본** | 해당 없음 | 24 | 24 | 24 |
| **Kafka 브로커** | 해당 없음 | 3 (기본값) | 3 (기본값) | 3 (기본값) |

1. **AWS M6i** 인스턴스로 테스트되었습니다.
2. 이 작업자와 컨트롤러 외에도 **3**개의 인프라 노드**(**크기 **M6i.12xlarge)** 및 **1** 워크로드 노드**( M6i.8xlarge**크기**)**를 테스트했습니다.

# 6장. NETWORK POLICY

**admin** 역할이 있는 사용자는**netobserv** 네임스페이스에 대한 네트워크 정책을 생성할 수 있습니다.

## 6.1. 네트워크 OBSERVABILITY에 대한 네트워크 정책 생성

**netobserv** 네임스페이스에 대한 수신 트래픽을 보호하려면 네트워크 정책을 생성해야 할 수 있습니다. 웹 콘솔에서 양식 보기를 사용하여 네트워크 정책을 생성할 수 있습니다.

프로세스

1. 네트워킹 → **NetworkPolicies** 로 이동합니다.
2. 프로젝트 드롭다운 메뉴에서 **netobserv** 프로젝트를 선택합니다.
3. 정책의 이름을 지정합니다. 이 예에서 정책 이름은 **allow-ingress** 입니다.
4. **Ingress** 규칙 추가를 세 번 클릭하여 수신 규칙을 생성합니다.
5. 양식에 다음을 지정합니다.
   a. 첫 번째 **Ingress** 규칙에 대해 다음 사양을 만듭니다.
      i. **Add allowed source** 드롭다운 메뉴에서 동일한 네임스페이스에서 **Pod** 허용을 선택합니다.
   b. 두 번째 **Ingress** 규칙에 대해 다음 사양을 만듭니다.
      i. **Add allowed source** 드롭다운 메뉴에서 클러스터 내부에서 **Pod** 허용을 선택합니다.
      ii. **+** 네임스페이스 선택기 추가를 클릭합니다.
      iii. **kubernetes.io/metadata.name** 레이블과 선택기 **openshift-console** 을 추가합니다.
   c. 세 번째 **Ingress** 규칙에 대해 다음 사양을 만듭니다.
      i. **Add allowed source** 드롭다운 메뉴에서 클러스터 내부에서 **Pod** 허용을 선택합니다.
      ii. **+** 네임스페이스 선택기 추가를 클릭합니다.
      iii. **kubernetes.io/metadata.name** 레이블과 선택기 **openshift-monitoring** 를 추가합니다.

검증

1. 모니터링 → 네트워크 트래픽 으로 이동합니다.
2. 트래픽 흐름 탭 또는 탭을 보고 데이터가 표시되는지 확인합니다.
3. 모니터링 → 대시보드 로 이동합니다. **NetObserv/Health** 선택에서 흐름이 수집되고 첫 번째 그래프에 표시되는 **Loki**로 전송되는지 확인합니다.

## 6.2. 네트워크 정책의 예

다음은 **netobserv** 네임스페이스의 예제 **NetworkPolicy** 오브젝트에 주석을 답니다.

네트워크 정책 샘플

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-ingress
  namespace: netobserv
spec:
  podSelector: {}          1
  ingress:
    - from:
        - podSelector: {}  2
          namespaceSelector: 3
            matchLabels:
              kubernetes.io/metadata.name: openshift-console
        - podSelector: {}
          namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: openshift-monitoring
  policyTypes:
    - Ingress
status: {}
```

1. 정책이 적용되는 Pod를 설명하는 선택기입니다. 정책 오브젝트는 **NetworkPolicy** 오브젝트를 정의하는 프로젝트에서 Pod만 선택할 수 있습니다. 이 문서에서는 **netobserv** serv 프로젝트인 Network Observability Operator가 설치된 프로젝트입니다.

2. 정책 오브젝트가 수신 트래픽을 허용하는 Pod와 일치하는 선택기입니다. 기본값은 선택기가 **NetworkPolicy** 와 동일한 네임스페이스의 Pod와 일치한다는 것입니다.

3. **namespaceSelector** 를 지정하면 선택기가 지정된 네임스페이스의 Pod와 일치합니다.

추가 리소스

CLI를 사용하여 네트워크 정책 생성

# 7장. 네트워크 트래픽 관찰

관리자는 OpenShift Container Platform 콘솔에서 네트워크 트래픽을 관찰하여 자세한 문제 해결 및 분석을 수행할 수 있습니다. 이 기능을 사용하면 트래픽 흐름의 다양한 그래픽 표현에서 통찰력을 얻을 수 있습니다. 네트워크 트래픽을 관찰하는 데 사용할 수 있는 여러 보기가 있습니다.

## 7.1. 개요 보기에서 네트워크 트래픽 관찰

개요 보기에는 클러스터의 네트워크 트래픽 흐름에 대한 전체 집계 지표가 표시됩니다. 관리자는 사용 가능한 표시 옵션을 사용하여 통계를 모니터링할 수 있습니다.

### 7.1.1. 개요 뷰 작업

관리자는 개요 보기로 이동하여 흐름 속도 통계의 그래픽 표시를 확인할 수 있습니다.

프로세스

1. 모니터링 → 네트워크 트래픽 으로 이동합니다.
2. 네트워크 트래픽 페이지에서 개요 탭을 클릭합니다.

메뉴 아이콘을 클릭하여 각 흐름 속도 데이터의 범위를 구성할 수 있습니다.

### 7.1.2. 개요 보기에 대한 고급 옵션 구성

고급 옵션을 사용하여 그래픽 보기를 사용자 지정할 수 있습니다. 고급 옵션에 액세스하려면 고급 옵션 표시를 클릭합니다. 표시 옵션 드롭다운 메뉴를 사용하여 그래프에서 세부 정보를 구성할 수 있습니다.사용 가능한 옵션은 다음과 같습니다.

- 메트릭 유형: **Cryostat** 또는 **Packets** 에 표시할 메트릭입니다. 기본값은 **Cryostat** 입니다.
- 범위: 네트워크 트래픽이 이동하는 구성 요소의 세부 정보를 선택합니다.노드,네임스페이스,소유자 또는 리소스로 범위를 설정할 수있습니다. 소유자는 리소스 집계입니다. 리소스는 호스트 네트워크 트래픽 또는 알 수 없는 **IP** 주소의 경우 **Pod**, 서비스, 노드일 수 있습니다. 기본값은 **Namespace** 입니다.
- **truncate** 레이블: 드롭다운 목록에서 레이블의 필요한 너비를 선택합니다. 기본값은**M** 입니다.

#### 7.1.2.1. 패널 관리

표시할 필요한 통계를 선택하고 순서를 다시 정렬할 수 있습니다. 열을 관리하려면 패널 관리를 클릭합니다.

#### 7.1.2.2. DNS 추적

개요 보기에서 네트워크 흐름의 **DNS(Domain Name System)** 추적을 그래픽으로 표시할 수 있습니다. **eBPF(Extended Berkeley Packet Filter)** 추적 후크와 함께 **DNS** 추적을 사용하면 다양한 용도로 사용할 수 있습니다.

- 네트워크 모니터링: **DNS** 쿼리 및 응답에 대한 인사이트를 제공하여 네트워크 관리자가 비정상적인 패턴, 잠재적인 병목 또는 성능 문제를 식별할 수 있도록 지원합니다.
- 보안 분석: 악성 코드가 사용하는 도메인 이름 생성 알고리즘**(DGA)**과 같은 의심스러운 **DNS** 활동을 감지하거나 보안 위반을 나타낼 수 있는 무단 **DNS** 확인을 식별합니다.

- 문제 해결: DNS 확인 단계를 추적하고 대기 시간을 추적하며 잘못된 구성을 식별하여 DNS 관련 문제를 디버깅합니다.

DNS 추적이 활성화되면 개요 의 차트에 표시되는 다음 메트릭을 확인할 수 있습니다. 이 뷰 활성화 및 작업에 대한 자세한 내용은 이 섹션의 *추가 리소스*를 참조하십시오.

- 평균 5개의 DNS 대기 시간
- 상위 5 DNS 응답 코드
- 총 5개의 DNS 응답 코드 스택

이 기능은 IPv4 및 IPv6 UDP 프로토콜에서 지원됩니다.

추가 리소스

- FlowCollector 에서 DNS를 구성하는 방법에 대한 자세한 내용은DNS 추적 작업을 참조하십시오.

## 7.2. 트래픽 흐름 보기에서 네트워크 트래픽 관찰

트래픽 흐름 보기에는 네트워크 흐름의 데이터와 테이블의 트래픽 양이 표시됩니다. 관리자는 트래픽 흐름 테이블을 사용하여 애플리케이션 간 트래픽 양을 모니터링할 수 있습니다.

### 7.2.1. 트래픽 흐름 보기 작업

관리자는 트래픽 흐름 테이블로 이동하여 네트워크 흐름정보를 볼 수 있습니다.

프로세스

1. 모니터링 → 네트워크 트래픽 으로 이동합니다.
2. 네트워크 트래픽 페이지에서 트래픽 흐름 탭을 클릭합니다.

각 행을 클릭하면 해당 흐름 정보를 얻을 수 있습니다.

### 7.2.2. 트래픽 흐름 보기에 대한 고급 옵션 구성

고급 옵션 표시를 사용하여 보기를 사용자 지정하고 내보낼 수 있습니다 표시 옵션 드롭다운 메뉴를 사용하여 행 크기를 설정할 수 있습니다. 기본값은 Normal 입니다.

#### 7.2.2.1. 열 관리

표시할 필수 열을 선택하고 순서를 다시 정렬할 수 있습니다. 열을 관리하려면 열 관리를 클릭합니다.

#### 7.2.2.2. 트래픽 흐름 데이터 내보내기

트래픽 흐름 보기에서 데이터를 내보낼 수 있습니다.

프로세스

1. 데이터 내보내기 를 클릭합니다.
2. 팝업 창에서 모든 데이터를 내보내려면 모든 데이터 내보내기 확인란을 선택하고 확인란의 선택을 해제하여 내보낼 필수 필드를 선택할 수 있습니다.

3. 내보내기 를 클릭합니다.

### 7.2.3. 대화 추적 작업

관리자는 동일한 대화의 일부인 네트워크 흐름을 그룹화할 수 있습니다. 대화는 IP 주소, 포트 및 프로토콜로 식별되는 피어 그룹화로 정의되므로 고유한 대화 ID가 생성됩니다. 웹 콘솔에서 대화 이벤트를 쿼리할 수 있습니다. 이러한 이벤트는 웹 콘솔에서 다음과 같이 표시됩니다.

- 대화 시작: 이 이벤트는 연결이 시작되거나 TCP 플래그가 인터셉트될 때 발생합니다.
- 대화 눈금: 이 이벤트는 연결이 활성화된 동안**FlowCollector spec.processor.conversationHeartbeatInterval** 매개변수에 정의된 각 지정된 간격으로 수행됩니다.
- 대화 종료: 이 이벤트는 **FlowCollector spec.processor.conversationEndTimeout** 매개변수에 도달하거나 TCP 플래그를 가로챌 때 발생합니다.
- **flow:** 지정된 간격 내에 발생하는 네트워크 트래픽 흐름입니다.

프로세스

1. 웹 콘솔에서 Operator → 설치된 Operator 로 이동합니다.
2. NetObserv Operator 의 제공된 API 제목에서 흐름 수집기 를 선택합니다.
3. 클러스터를 선택한 다음 YAML 탭을 선택합니다.
4. **spec.processor.logTypes,conversationEndTimeout** 및 **conversationHeartbeatInterval** 매개변수가 관찰 요구 사항에 따라 설정되도록 **FlowCollector** 사용자 지정 리소스를 구성합니다. 샘플 구성은 다음과 같습니다.

   대화 추적을 위한 **FlowCollector** 구성

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
 processor:
  conversationEndTimeout: 10s          1
  logTypes: FLOWS                      2
  conversationHeartbeatInterval: 30s   3
```

1. Conversation end 이벤트는 **conversationEndTimeout** 에 도달하거나 TCP 플래그가 가로채는 시점을 나타냅니다.
2. **logTypes** 를 **FLOWS** 로 설정하면 Flow 이벤트만 내보냅니다. 값을 **ALL** 로 설정하면 대화 및 흐름 이벤트가 모두 내보내 네트워크 트래픽 페이지에 표시됩니다. 대화 이벤트에만 집중하기 위해 대화 시작, 대화 틱 및 대화 종료 이벤트 또는 **ENDED_CONVERSATIONS** 내보내기를 대화 종료 이벤트를 내보내는 **CONVERSATIONS** 를 지정할 수 있습니다. 스토리지 요구 사항은 ENDED _ **CONVERSATIONS** 의 경우 best 및 lowest입니다.
3. Conversation tick 이벤트는 네트워크 연결이 활성화된 동안**FlowCollector conversationHeartbeatInterval** 매개변수에 정의된 각 지정된 간격을 나타냅니다.

참고

**logType** 옵션을 업데이트하면 이전 선택의 흐름이 콘솔 플러그인에서 명확하지 않습니다. 예를 들어 처음에는 오전 10일까지 **logType** 을 **CONVERSATIONS** 로 설정한 다음 **ENDED_CONVERSATIONS** 로 이동한 다음 콘솔 플러그인은 오전 10시 전의 모든 대화 이벤트를 표시하고 10 AM 이후의 대화만 종료합니다.

5. 트래픽 흐름 탭에서 네트워크 트래픽 페이지를 새로 고칩니다. 두 개의 새 열인**Event/Type** 및 **Conversation Id** 가 있습니다. 모든이벤트/유형 필드는**Flow** 가 선택한 쿼리 옵션인 경우**Flow** 입니다.

6. 쿼리 옵션을 선택하고로그 유형,대화 유형을 선택합니다. 이제 이벤트/유형 이 원하는 대화 이벤트를 모두 표시합니다.

7. 다음으로 특정 대화 ID를 필터링하거나 측면 패널에서 대화 및 흐름 로그 유형 옵션을 전환할 수 있습니다.

### 7.2.4. DNS 추적 작업

DNS 추적을 사용하면 네트워크를 모니터링하고 보안 분석을 수행하고 DNS 문제를 해결할 수 있습니다. 다음 YAML 예제의 사양으로 **FlowCollector** 를 편집하여 DNS를 추적할 수 있습니다.

중요

이 기능이 활성화되면 CPU 및 메모리 사용량 증가가 eBPF 에이전트에서 관찰됩니다.

프로세스

1. 웹 콘솔에서 **Operator → 설치된 Operator** 로 이동합니다.
2. NetObserv Operator 의 제공된 API 제목에서 흐름 수집기 를 선택합니다.
3. 클러스터를 선택한 다음YAML 탭을 선택합니다.
4. **FlowCollector** 사용자 지정 리소스를 구성합니다. 샘플 구성은 다음과 같습니다.

   DNS 추적을 위한 **FlowCollector** 구성

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  namespace: netobserv
  deploymentModel: DIRECT
  agent:
    type: EBPF
    ebpf:
      features:
       - DNSTracking          1
      privileged: true        2
```

1. **spec.agent.ebpf.features** 매개변수 목록을 설정하여 웹 콘솔에서 각 네트워크 흐름의 DNS 추적을 활성화할 수 있습니다.

2 DNS 추적을 활성화하려면 **spec.agent.ebpf.privileged** 사양 값이 **true** 여야 합니다.

5. 네트워크 트래픽 페이지를 새로 고칠 때개요 및 트래픽 흐름 보기 및 적용할 수 있는 새 필터에서 볼 수 있는 새로운 **DNS** 표현이 있습니다.

   a. 패널 관리에서 새 **DNS** 선택 사항을 선택하여개요 에 그래픽 시각화 및 **DNS** 지표를 표시합니다.

   b. 열 관리에서 새 선택 항목을 선택하여트래픽 흐름 보기에 **DNS** 열을 추가합니다.

   c. **DNS Id, DNS**대기 시간 및 **DNS** 응답 코드와 같은 특정 **DNS** 메트릭을 필터링하고 측면 패널에서 자세한 정보를 참조하십시오.

#### 7.2.4.1. 히스토그램 사용

히스토그램 표시를 클릭하여 흐름 기록을 가로 막대형 차트로 시각화하기 위한 도구 모음 보기를 표시할 수 있습니다. 히스토그램은 시간에 따른 로그 수를 보여줍니다. 히스토그램의 일부를 선택하여 도구 모음을 따르는 테이블에서 네트워크 흐름 데이터를 필터링할 수 있습니다.

## 7.3. 토폴로지 보기에서 네트워크 트래픽 관찰

토폴로지 보기에는 네트워크 흐름과 트래픽 양을 그래픽으로 표시할 수 있습니다. 관리자는토폴로지 보기를 사용하여 애플리케이션에서 트래픽 데이터를 모니터링할 수 있습니다.

### 7.3.1. 토폴로지 보기 작업

관리자는 토폴로지 보기로 이동하여 구성 요소의 세부 정보 및 지표를 확인할 수 있습니다.

프로세스

1. 모니터링 → 네트워크 트래픽 으로 이동합니다.
2. 네트워크 트래픽 페이지에서 토폴로지 탭을 클릭합니다.

토폴로지 에서 각 구성 요소를 클릭하여 구성 요소의 세부 정보 및 지표를 확인할 수 있습니다.

### 7.3.2. 토폴로지 보기의 고급 옵션 구성

고급 옵션 표시를 사용하여 보기를 사용자 지정하고 내보낼 수 있습니다고급 옵션 보기에는 다음과 같은 기능이 있습니다.

- 보기에서 찾기: 뷰에서 필요한 구성 요소를 검색하려면 다음을 수행합니다.
- **Display options:** 다음 옵션을 구성하려면 다음을 수행합니다.
  - **layout** : 그래픽 표현의 레이아웃을 선택합니다. 기본값은 **ColaNoForce** 입니다.
  - **scope:** 네트워크 트래픽이 이동하는 구성 요소의 범위를 선택합니다. 기본값은**Namespace** 입니다.
  - **groups:** 구성 요소를 그룹화하여 소유권에 대한 이해를 제공합니다. 기본값은**None** 입니다.
  - 그룹 축소: 그룹을 확장하거나 축소합니다. 그룹은 기본적으로 확장됩니다. **group**에 값이 **None** 인 경우 이 옵션이 비활성화됩니다.

- **show:** 표시할 필요가 있는 세부 정보를 선택합니다. 모든 옵션은 기본적으로 확인됩니다. 사용 가능한 옵션은 **Edges,Edges** 레이블 및 **Badges** 입니다.
- **truncate** 레이블: 드롭다운 목록에서 레이블의 필요한 너비를 선택합니다. 기본값은**M** 입니다.

#### 7.3.2.1. 토폴로지 보기 내보내기

뷰를 내보내려면 토폴로지 보기 내보내기 를 클릭합니다. 보기는 **PNG** 형식으로 다운로드됩니다.

## 7.4. 네트워크 트래픽 필터링

기본적으로 네트워크 트래픽 페이지는 **FlowCollector** 인스턴스에 구성된 기본 필터를 기반으로 클러스터의 트래픽 흐름 데이터를 표시합니다. 필터 옵션을 사용하여 사전 설정 필터를 변경하여 필요한 데이터를 관찰할 수 있습니다.

쿼리 옵션

다음과 같이 쿼리 옵션을 사용하여 검색 결과를 최적화할 수 있습니다.

- 사용 가능한 옵션 대화 및 흐름은흐름 로그, 새 대화, 완료된 대화 및 하트비트와 같은 로그유형별로 흐름을 쿼리하는 기능을 제공합니다. 이 기능은 긴 대화에 대한 업데이트가 포함된 주기적인 레코드입니다. 대화는 동일한 피어 간의 흐름을 집계하는 것입니다.
- 중복된 흐름: 여러 인터페이스에서 흐름이 보고될 수 있으며 소스 및 대상 노드 모두에서 흐름이 데이터에 여러 번 표시될 수 있습니다. 이 쿼리 옵션을 선택하면 중복된 흐름을 표시하도록 선택할 수 있습니다. 복제된 흐름은 포트를 포함한 동일한 소스 및 대상을 가지며 **Interface** 및 **Direction** 필드를 제외하고 동일한 프로토콜을 갖습니다. 중복은 기본적으로 숨겨집니다. 드롭다운 목록의 **Common** 섹션에서 **Direction** 필터를 사용하여 수신 트래픽과 송신 트래픽 사이를 전환합니다.
- **match** 필터: 고급 필터에서 선택한 다양한 필터 매개변수 간의 관계를 확인할 수 있습니다. 사용 가능한 옵션은 **all** 및 **match any**와 일치합니다. **match all**은 모든 값과 일치하는 결과를 제공하며 모든 값과 일치하면 입력된 값과 일치하는 결과가 제공됩니다. 기본값은 **all**과 일치합니다.
- **limit:** 내부 백엔드 쿼리의 데이터 제한입니다. 일치 및 필터 설정에 따라 트래픽 흐름 데이터 수가 지정된 제한 내에 표시됩니다.

빠른 필터

빠른 필터 드롭다운 메뉴의 기본값은**FlowCollector** 구성에 정의되어 있습니다. 콘솔에서 옵션을 수정할 수 있습니다.

고급 필터

드롭다운 목록에서 필터링할 매개변수를 선택하여 고급 필터, 공통,소스 또는대상 을 설정할 수 있습니다. 흐름 데이터는 선택 사항에 따라 필터링됩니다. 적용된 필터를 활성화하거나 비활성화하려면 필터 옵션 아래에 나열된 적용된 필터를 클릭하면 됩니다.

↑ 에서 한 가지 방법과 ↑ 뒤로 ↓ 필터링을 전환할 수 있습니다. ↑ 한 가지 방법 필터는 필터 선택에 따라소스 및 대상 트래픽만 표시합니다. 스왑을 사용하여 소스 및 대상 트래픽의 방향 보기를 변경할 수 있습니다. ↑ ↓ **Back and forth** 필터에는 소스 및 대상 필터가 있는 반환 트래픽이 포함됩니다. 네트워크 트래픽의 방향 흐름은 트래픽의 **Direction** 열에 **Ingress'** 또는 **'Egress for inter-node traffic** 및 **' Inner'** 트래픽의 경우 단일 노드 내의 트래픽으로 표시됩니다.

기본값 재설정 을 클릭하여 기존 필터를 제거하고**FlowCollector** 구성에 정의된 필터를 적용할 수 있습니다.

참고

텍스트 값을 지정하는 규칙을 이해하려면 자세히 알아보기 를 클릭합니다.

또는 해당 집계의 필터링된 데이터를 제공하는 네임스페이스,서비스,경로,노드, 워크로드 페이지의 네트워크 트래픽 탭에서 트래픽 흐름 데이터에 액세스할 수 있습니다.

추가 리소스

**FlowCollector** 에서 빠른 필터를 구성하는 방법에 대한 자세한 내용은빠른 필터 및 흐름 수집기 샘플 리소스 구성 을 참조하십시오.

# 8장. NETWORK OBSERVABILITY OPERATOR 모니터링

웹 콘솔을 사용하여 **Network Observability Operator**의 상태와 관련된 경고를 모니터링할 수 있습니다.

## 8.1. 상태 정보 보기

웹 콘솔의 대시보드 페이지에서 **Network Observability Operator**의 상태 및 리소스 사용량에 대한 메트릭에 액세스할 수 있습니다. 경고가 트리거될 경우 대시보드로 연결되는 상태 경고 배너는 네트워크 트래픽 및 홈 페이지에 표시될 수 있습니다. 다음과 같은 경우 경고가 생성됩니다.

- **Loki ingestion rate limit**에 도달한 경우와 같이 **flowlogs-pipeline** 워크로드가 **Loki** 오류로 인해 흐름을 삭제하면 **NetObservLokiError** 경고가 발생합니다.
- **NetObservNoFlows** 경고는 일정 시간 동안 흐름이 수집되지 않으면 발생합니다.

사전 요구 사항

- **Network Observability Operator**가 설치되어 있어야 합니다.
- **cluster-admin** 역할 또는 모든 프로젝트에 대한 보기 권한이 있는 사용자로 클러스터에 액세스할 수 있습니다.

프로세스

1. 웹 콘솔의 관리자 화면에서 모니터링 → 대시보드 로 이동합니다.
2. 대시보드 드롭다운에서 **Netobserv/Health** 를 선택합니다. **Operator** 상태에 대한 지표가 페이지에 표시됩니다.

### 8.1.1. 상태 경고 비활성화

**FlowCollector** 리소스를 편집하여 상태 경고를 비활성화할 수 있습니다.

1. 웹 콘솔에서 **Operator** → 설치된 **Operator** 로 이동합니다.
2. **NetObserv Operator** 의 제공된 **API** 제목에서 흐름 수집기 를 선택합니다.
3. 클러스터를 선택한 다음 **YAML** 탭을 선택합니다.
4. **spec.processor.metrics.disableAlerts** 를 추가하여 다음 **YAML** 샘플과 같이 상태 경고를 비활성화합니다.

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  processor:
    metrics:
      disableAlerts: [NetObservLokiError, NetObservNoFlows] 1
```

1 비활성화할 경고 유형을 모두 사용하여 하나 또는 목록을 지정할 수 있습니다.

## 8.2. NETOBSERV 대시보드에 대한 LOKI 속도 제한 경고 생성

Loki 속도 제한에 도달할 때 Netobserv 대시보드 메트릭에 대한 사용자 지정 규칙을 생성하여 경고를 트리거할 수 있습니다.

경고 규칙 구성 YAML 파일의 예는 다음과 같습니다.

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: loki-alerts
  namespace: openshift-operators-redhat
spec:
  groups:
  - name: LokiRateLimitAlerts
    rules:
    - alert: LokiTenantRateLimit
      annotations:
        message: |-
          {{ $labels.job }} {{ $labels.route }} is experiencing 429 errors.
        summary: "At any number of requests are responded with the rate limit error code."
      expr: sum(irate(loki_request_duration_seconds_count{status_code="429"}[1m])) by (job,
namespace, route) / sum(irate(loki_request_duration_seconds_count[1m])) by (job,
namespace, route) * 100 > 0
      for: 10s
      labels:
        severity: warning
```

추가 리소스

- 대시보드에서 볼 수 있는 경고를 만드는 방법에 대한 자세한 내용은 사용자 정의 프로젝트에 대한 경고 규칙 생성 을 참조하십시오.

# 9장. FLOWCOLLECTOR 구성 매개변수

FlowCollector는 기본 배포를 시험하고 구성하는 네트워크 흐름 컬렉션 API의 스키마입니다.

## 9.1. FLOWCOLLECTOR API 사양

설명

**FlowCollector** 는 기본 배포를 시험하고 구성하는 네트워크 흐름 컬렉션 API의 스키마입니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **apiVersion** | **string** | APIVersion은 버전이 지정된 이 오브젝트 표현의 스키마를 정의합니다. 서버는 인식된 스키마를 최신 내부 값으로 변환해야 하며 인식할 수 없는 값을 거부할 수 있습니다. 자세한 내용은 https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources |
| **kind** | **string** | kind는 이 오브젝트가 나타내는 REST 리소스에 해당하는 문자열 값입니다. 서버는 클라이언트가 요청을 제출하는 끝점에서 이를 유추할 수 있습니다. CamelCase로 업데이트할 수 없습니다. 자세한 내용은 https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds |
| 메타데이터 | **object** | 표준 오브젝트의 메타데이터입니다. 자세한 내용은 https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **spec** | **object** | FlowCollector 리소스의 원하는 상태를 정의합니다.<br><br>*: 이 문서 전체에서 "지원되지 않음" 또는 "더 이상 사용되지 않음"은 이 기능이 Red Hat에서 공식적으로 지원하지 않음을 의미합니다. 예를 들어 커뮤니티에서 기여하고 유지 관리를 위한 공식적인 동의 없이 수락되었을 수 있습니다. 제품 유지 관리자는 최상의 노력으로 이러한 기능에 대한 지원을 제공할 수 있습니다. |

### 9.1.1. .metadata

설명

표준 오브젝트의 메타데이터입니다. 자세한 내용은 https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

유형

**object**

### 9.1.2. .spec

설명

**FlowCollector** 리소스의 원하는 상태를 정의합니다.

***:** 이 문서 전체에서 "지원되지 않음" 또는 "더 이상 사용되지 않음"은 이 기능이 **Red Hat**에서 공식적으로 지원하지 않음을 의미합니다. 예를 들어 커뮤니티에서 기여하고 유지 관리를 위한 공식적인 동의 없이 수락되었을 수 있습니다. 제품 유지 관리자는 최상의 노력으로 이러한 기능에 대한 지원을 제공할 수 있습니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **agent** | **object** | 흐름 추출에 대한 에이전트 구성입니다. |
| **consolePlugin** | **object** | Console **Plugin** 은 사용 가능한 경우 OpenShift Container Platform 콘솔 플러그인과 관련된 설정을 정의합니다. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **deploymentModel** | **string** | **deploymentModel** 은 흐름 처리를 위해 원하는 유형의 배포를 정의합니다. 가능한 값은 다음과 같습니다.<br>- **DIRECT** (기본값) - 흐름 프로세서가 에이전트에서 직접 수신 대기하도록 합니다.<br>- 프로세서에서 사용하기 전에 Kafka 파이프라인으로 전송되는 흐름을 만드는 **KAFKA** 입니다. Kafka는 더 나은 확장성, 복원력 및 고가용성을 제공할 수 있습니다(자세한 내용은 https://www.redhat.com/en/topics/integration/what-is-apache-kafka)를 참조하십시오. |
| 내보내기 | **array** | 내보내기 는 사용자 정의 사용 또는 스토리지에 대한 추가 선택적 내보내기를 정의합니다. |
| **kafka** | **object** | Kafka 구성을 통해 Kafka를 흐름 컬렉션 파이프라인의 일부로 브로커로 사용할 수 있습니다. **spec.deploymentModel** 이 **KAFKA** 인 경우 사용할 수 있습니다. |
| **loki** | **object** | Loki, 흐름 저장소, 클라이언트 설정. |
| 네임스페이스 | **string** | 네트워크 Observability Pod가 배포되는 네임스페이스입니다. |
| 프로세서 | **object** | 프로세서는 에이전트에서 흐름을 수신하고, 보강하고, 메트릭을 생성하고, Loki 지속성 계층 및/또는 사용 가능한 내보내기로 전달하는 구성 요소의 설정을 정의합니다. |

### 9.1.3. .spec.agent

설명

흐름 추출에 대한 에이전트 구성입니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **ebpf** | **object** | **eBPF**는 **spec.agent.type** 이 Cryostat **PF** 로 설정된 경우 eBPF 기반 flow reporter와 관련된 설정을 설명합니다. |
| **ipfix** | **object** | **ipFI** X [deprecated (*)] - **spec.agent.type** 이 IPFIX로 설정된 경우 **IPFIX** 기반 흐름 보고기와 관련된 설정을 설명합니다. |
| **type** | **string** | **type** 은 흐름 추적 에이전트를 선택합니다. 가능한 값은 다음과 같습니다.<br>- Cryostat **PF** (기본값)는 네트워크 Observability eBPF 에이전트를 사용합니다.<br>- **IPFIX** [더 이상 사용되지 않는 (*)] - 레거시 IPFIX 수집기를 사용합니다.<br>**CryostatPF** 는 더 나은 성능을 제공하며 클러스터에 설치된 CNI와 관계없이 작동해야 합니다. **IPFIX** 는 OVN-Kubernetes CNI에서 작동합니다(IPFIX 내보내기를 지원하는 경우 다른 CNI가 작동할 수 있지만 수동 구성이 필요합니다). |

### 9.1.4. .spec.agent.ebpf

설명

**eBPF**는 **spec.agent.type** 이 Cryostat**PF** 로 설정된 경우 **eBPF** 기반 **flow reporter**와 관련된 설정을 설명합니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **cacheActiveTimeout** | **string** | **cacheActiveTimeout** 은 보고자가 전송하기 전에 집계하는 최대 기간입니다. **cacheMaxFlows** 및 **cacheActiveTimeout** 을 늘리면 네트워크 트래픽 오버헤드와 CPU 로드를 줄일 수 있지만 더 많은 메모리 소비와 흐름 컬렉션에서 대기 시간이 증가할 수 있습니다. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **cacheMaxFlows** | **integer** | **cacheMaxFlows** 는 집계의 최대 흐름 수입니다. 도달한 경우 보고자가 흐름을 보냅니다. **cacheMaxFlows** 및 **cacheActiveTimeout** 을 늘리면 네트워크 트래픽 오버헤드와 CPU 로드를 줄일 수 있지만 더 많은 메모리 소비와 흐름 컬렉션에서 대기 시간이 증가할 수 있습니다. |
| **debug** | **object** | **debug** 를 사용하면 eBPF 에이전트의 내부 구성의 일부 측면을 설정할 수 있습니다. 이 섹션은 GOGC 및 GOMAXPROCS env vars와 같은 디버깅 및 세분화된 성능 최적화를 위한 것입니다. 값을 설정하는 사용자는 자신의 위험에 이를 수행합니다. |
| **excludeInterfaces** | 배열(문자열) | **excludeInterfaces** 에는 흐름 추적에서 제외된 인터페이스 이름이 포함되어 있습니다. 항목은 **/br-/** 과 같은 슬래시로 묶고 정규식으로 일치합니다. 그렇지 않으면 대소문자를 구분하지 않는 문자열로 일치됩니다. |

| 속성 | 유형 | 설명 |
|---|---|---|
| **features** | 배열(문자열) | 활성화할 추가 기능 목록입니다. 모두 기본적으로 비활성화되어 있습니다. 추가 기능을 활성화하면 성능에 영향을 미칠 수 있습니다. 가능한 값은 다음과 같습니다.<br>- **PacketDrop**: 패킷 삭제 흐름 로깅 기능을 활성화합니다. 이 기능을 사용하려면 커널 디버그 파일 시스템을 마운트해야 하므로 eBPF Pod를 privileged로 실행해야 합니다. **spec.agent.eBPF.privileged** 매개변수가 설정되지 않은 경우 오류가 보고됩니다.<br>- **DNSTracking**: DNS 추적 기능을 활성화합니다. 이 기능을 사용하려면 커널 디버그 파일 시스템을 마운트해야 하므로 eBPF Pod를 권한으로 실행해야 합니다. **spec.agent.eBPF.privileged** 매개변수가 설정되지 않은 경우 오류가 보고됩니다.<br>- **FlowRTT** [unsupported (*)]: TCP handshakes 중에 eBPF 에이전트에서 flow latency (RTT) 계산을 활성화합니다. 이 기능은 샘플링 1에서 더 잘 작동합니다. |
| **imagePullPolicy** | **string** | **imagePullPolicy** 는 위에 정의된 이미지의 Kubernetes 가져오기 정책입니다. |
| 인터페이스 | 배열(문자열) | 인터페이스에 는 흐름이 수집되는 위치의 인터페이스 이름이 포함되어 있습니다. 비어 있는 경우 에이전트는 ExcludeInterfaces에 나열된 인터페이스를 제외하고 시스템의 모든 인터페이스를 가져옵니다. 항목은 **/br-/** 과 같은 슬래시로 묶여 정규식과 일치합니다. 그렇지 않으면 대소문자를 구분하지 않는 문자열로 일치됩니다. |
| **kafkaBatchSize** | **integer** | **kafkaBatchSize** 는 파티션으로 전송되기 전에 요청의 최대 크기를 바이트 단위로 제한합니다. Kafka를 사용하지 않을 때는 무시됩니다. 기본값: 10MB. |
| **logLevel** | **string** | **loglevel** 은 Network Observability eBPF Agent의 로그 수준을 정의합니다. |

| 속성 | 유형 | 설명 |
|---|---|---|
| **privileged** | **boolean** | eBPF 에이전트 컨테이너의 권한 모드입니다. 일반적으로 이 설정은 무시하거나 false로 설정할 수 있습니다. 이 경우 Operator는 세분화된 기능(BPF, PERFMON, NET_ADMIN, SYS_RESOURCE)을 컨테이너에 설정하여 올바른 작업을 활성화합니다. 어떤 이유로든 이러한 기능을 설정할 수 없는 경우(예: CAP_BPF를 모르는 이전 커널 버전이 사용 중인 경우 보다 글로벌 권한을 위해 이 모드를 켤 수 있습니다. |
| **resources** | **object** | 리소스는 이 컨테이너에 필요한 컴퓨팅 리소스입니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |
| **sampling** | **integer** | 흐름 보고기의 샘플링 속도입니다. 100은 100에 하나의 흐름이 전송되었음을 의미합니다. 0 또는 1은 모든 흐름이 샘플링됨을 의미합니다. |

### 9.1.5. .spec.agent.ebpf.debug

설명

**debug** 를 사용하면 **eBPF** 에이전트의 내부 구성의 일부 측면을 설정할 수 있습니다. 이 섹션은 **GOGC** 및 **GOMAXPROCS env vars**와 같은 디버깅 및 세분화된 성능 최적화를 위한 것입니다. 값을 설정하는 사용자는 자신의 위험에 이를 수행합니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **env** | 오브젝트(문자열) | **env** 를 사용하면 사용자 지정 환경 변수를 기본 구성 요소에 전달할 수 있습니다. GoGC 및 GOMAXPROCS와 같은 매우 구체적인 성능 튜닝 옵션을 전달하는 데 유용합니다. 이는 에지 디버그 또는 지원 시나리오에서만 유용하므로 FlowCollector 설명자의 일부로 공개적으로 노출해서는 안 됩니다. |

### 9.1.6. .spec.agent.ebpf.resources

설명

리소스는 이 컨테이너에 필요한 컴퓨팅 리소스입니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **limits** | **integer-or-string** | 제한은 허용되는 최대 컴퓨팅 리소스 양을 나타냅니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |
| **requests** | **integer-or-string** | 요청은 필요한 최소 컴퓨팅 리소스 양을 설명합니다. 컨테이너에 대한 Requests를 생략하면 구현 정의된 값을 제외하고 명시적으로 지정된 경우 기본값은 Limits로 설정됩니다. 요청은 제한을 초과할 수 없습니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |

### 9.1.7. .spec.agent.ipfix

설명

**ipFI X** [deprecated (*)] - **spec.agent.type** 이 **IPFIX**로 설정된 경우**IPFIX** 기반 흐름 보고기와 관련된 설정을 설명합니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **cacheActiveTimeout** | **string** | **cacheActiveTimeout** 은 보고자가 전송하기 전에 집계하는 최대 기간입니다. |
| **cacheMaxFlows** | **integer** | **cacheMaxFlows** 는 집계의 최대 흐름 수입니다. 도달한 경우 보고자가 흐름을 보냅니다. |
| **clusterNetworkOperator** | **object** | **clusterNetworkOperator** 는 사용 가능한 경우 OpenShift Container Platform Cluster Network Operator와 관련된 설정을 정의합니다. |
| **forceSampleAll** | **boolean** | **forceSampleAll** 을 사용하면 IPFIX 기반 흐름 보고기에서 샘플링을 비활성화할 수 있습니다. 클러스터 불안정성을 생성할 수 있으므로 IPFIX를 사용하여 모든 트래픽을 샘플링하지 않는 것이 좋습니다. 이렇게 하려면 이 플래그를 true로 설정합니다. at your own risk 입니다. true로 설정하면 샘플링 값이 무시됩니다. |
| **ovnKubernetes** | **object** | **OVNKubernetes** 는 사용 가능한 경우 OVN-Kubernetes CNI의 설정을 정의합니다. 이 구성은 OpenShift Container Platform 없이 OVN의 IPFIX 내보내기를 사용할 때 사용됩니다. OpenShift Container Platform을 사용하는 경우 대신 **clusterNetworkOperator** 속성을 참조하십시오. |
| **sampling** | **integer** | 샘플링 은 보고자의 샘플링 비율입니다. 100은 100에 하나의 흐름이 전송되었음을 의미합니다. 클러스터 안정성을 보장하기 위해 2 아래의 값을 설정할 수 없습니다. 클러스터 안정성에 영향을 미칠 수 있는 모든 패킷을 샘플링하려면 **forceSampleAll** 을 참조하십시오. 또는 IPFIX 대신 eBPF Agent를 사용할 수 있습니다. |

### 9.1.8. .spec.agent.ipfix.clusterNetworkOperator

설명

**clusterNetworkOperator** 는 사용 가능한 경우 OpenShift Container Platform Cluster Network Operator와 관련된 설정을 정의합니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| 네임스페이스 | **string** | 구성 맵을 배포할 네임스페이스입니다. |

### 9.1.9. .spec.agent.ipfix.ovnKubernetes

설명

**OVNKubernetes** 는 사용 가능한 경우 OVN-Kubernetes CNI의 설정을 정의합니다. 이 구성은 OpenShift Container Platform 없이 OVN의 IPFIX 내보내기를 사용할 때 사용됩니다. OpenShift Container Platform을 사용하는 경우 대신 **clusterNetworkOperator** 속성을 참조하십시오.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **containerName** | **string** | **containername** 은 IPFIX에 대해 구성할 컨테이너의 이름을 정의합니다. |
| **daemonSetName** | **string** | **daemonSetName** 은 OVN-Kubernetes Pod를 제어하는 DaemonSet의 이름을 정의합니다. |
| 네임스페이스 | **string** | OVN-Kubernetes Pod가 배포되는 네임스페이스입니다. |

### 9.1.10. .spec.consolePlugin

설명

**Console Plugin** 은 사용 가능한 경우 OpenShift Container Platform 콘솔 플러그인과 관련된 설정을 정의합니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| 자동 스케일러 | **object** | 플러그인 배포에 설정할 수평 Pod 자동 스케일러의 자동 스케일러 사양입니다. **HorizontalPodAutoscaler** 문서**(autoscaling/v2)**를 참조하십시오**.** |
| **enable** | **boolean** | 콘솔 플러그인 배포를 활성화합니다. spec.Loki.enable도 true여야 합니다. |
| **imagePullPolicy** | **string** | **imagePullPolicy** 는 위에 정의된 이미지의 Kubernetes 가져오기 정책입니다. |
| **logLevel** | **string** | 콘솔 플러그인 백엔드의 로그 수준 |
| **port** | **integer** | **port** 는 플러그인 서비스 포트입니다. 메트릭을 위해 예약된 9002를 사용하지 마십시오. |
| **portNaming** | **object** | **portNaming** 은 port-to-service 이름 변환의 구성을 정의합니다. |
| **quickFilters** | **array** | **quickFilters** 는 Console 플러그인에 대한 빠른 필터 사전 설정을 구성합니다. |
| 등록 | **boolean** | 등록 은 true로 설정하면 제공된 콘솔 플러그인을 OpenShift Container Platform Console Operator에 자동으로 등록할 수 있습니다. false로 설정하면 oc patch console.operator.openshift.io/cluster를 편집하여 수동으로 등록할 수 있습니다. **oc patch console.operator.openshift.io cluster --type='json' -p '[{"op": "add", "path": "/spec/plugins/-", "value": "netobserv-plugin"}]]** |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **replicas** | **integer** | **replicas** 는 시작할 복제본(Pod) 수를 정의합니다. |
| **resources** | **object** | 이 컨테이너에 필요한 컴퓨팅 리소스 측면에서 리소스입니다. 자세한 내용은 **https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/** |

### 9.1.11. .spec.consolePlugin.autoscaler

설명

플러그인 배포에 설정할 수평 **Pod** 자동 스케일러의 자동 스케일러 사양입니다. **HorizontalPodAutoscaler** 문서**(autoscaling/v2)**를 참조하십시오**.**

유형

**object**

### 9.1.12. .spec.consolePlugin.portNaming

설명

**portNaming** 은 **port-to-service** 이름 변환의 구성을 정의합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **enable** | **boolean** | 콘솔 플러그인 포트-서비스 이름 변환을 활성화 |
| **portNames** | 오브젝트**(**문자열**)** | **portNames** 는 콘솔에서 사용할 추가 포트 이름을 정의합니다(예 **: {"3100": "loki"}**). |

### 9.1.13. .spec.consolePlugin.quickFilters

설명

**quickFilters** 는 **Console** 플러그인에 대한 빠른 필터 사전 설정을 구성합니다**.**

유형

**array**

### 9.1.14. .spec.consolePlugin.quickFilters[]

설명

**QuickFilter** 는 콘솔의 빠른 필터에 대한 사전 설정 구성을 정의합니다**.**

유형

**object**

필수 항목

- **filter**
- **name**

| 속성 | 유형 | 설명 |
|---|---|---|
| **default** | **boolean** | **default** 는 이 필터가 기본적으로 활성화되어야 하는지 여부를 정의합니다. |
| **filter** | 오브젝트**(**문자열**)** | **filter** 는 이 필터를 선택할 때 설정할 키 및 값 집합입니다. 각 키는 쉼표로 구분된 문자열을 사용하여 값 목록과 연결할 수 있습니다(예: **filter: {"src_namespace": "namespace1,namespace2"}** ). |
| **name** | **string** | 콘솔에 표시되는 필터의 이름 |

### 9.1.15. .spec.consolePlugin.resources

설명

이 컨테이너에 필요한 컴퓨팅 리소스 측면에서 리소스입니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **limits** | **integer-or-string** | 제한은 허용되는 최대 컴퓨팅 리소스 양을 나타냅니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |
| **requests** | **integer-or-string** | 요청은 필요한 최소 컴퓨팅 리소스 양을 설명합니다. 컨테이너에 대한 Requests를 생략하면 구현 정의된 값을 제외하고 명시적으로 지정된 경우 기본값은 Limits로 설정됩니다. 요청은 제한을 초과할 수 없습니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |

### 9.1.16. .spec.exporters

설명

내보내기 는 사용자 정의 사용 또는 스토리지에 대한 추가 선택적 내보내기를 정의합니다.

유형

**array**

### 9.1.17. .spec.exporters[]

설명

**FlowCollectorExporter** 는 보강된 흐름을 보낼 추가 내보내기를 정의합니다.

유형

**object**

필수 항목

- **type**

| 속성 | 유형 | 설명 |
|---|---|---|
| **ipfix** | **object** | 강화된 IPFIX 흐름을 보내기 위한 IP 주소 및 포트와 같은 IPFIX 구성입니다. |
| **kafka** | **object** | 풍요한 흐름을 보내기 위한 주소 및 주제와 같은 Kafka 구성입니다. |
| **type** | **string** | **type** 은 내보내기 유형을 선택합니다. 사용 가능한 옵션은 **KAFKA** 및 **IPFIX** 입니다. |

### 9.1.18. .spec.exporters[].ipfix

설명

강화된 **IPFIX** 흐름을 보내기 위한 **IP** 주소 및 포트와 같은 **IPFIX** 구성입니다**.**

유형

**object**

필수 항목

- **targetHost**
- **targetPort**

| 속성 | 유형 | 설명 |
|---|---|---|
| **targetHost** | **string** | IPFIX 외부 수신자의 주소 |
| **targetPort** | **integer** | IPFIX 외부 수신자용 포트 |
| 전송 | **string** | IPFIX 연결에 사용되는 전송 프로토콜(**TCP** 또는 **UDP**)은 기본적으로 **TCP** 입니다. |

### 9.1.19. .spec.exporters[].kafka

설명

풍요한 흐름을 보내기 위한 주소 및 주제와 같은 **Kafka** 구성입니다**.**

유형

**object**

필수 항목

- **address**
- **topic**

| 속성 | 유형 | 설명 |
|---|---|---|
| **address** | **string** | Kafka 서버의 주소 |
| **SASL** | **object** | SASL 인증 구성. [지원되지 않음 (*)]. |
| **tls** | **object** | TLS 클라이언트 구성. TLS를 사용하는 경우 주소가 TLS에 사용되는 Kafka 포트(일반적으로 9093)와 일치하는지 확인합니다. |
| **topic** | **string** | 사용할 Kafka 주제입니다. 존재할 수 있어야 합니다. Network Observability는 이를 생성하지 않습니다. |

### 9.1.20. .spec.exporters[].kafka.sasl

설명

**SASL** 인증 구성**. [**지원되지 않음**(*)].**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **clientIDReference** | **object** | 클라이언트 ID가 포함된 시크릿 또는 구성 맵에 대한 참조 |
| **clientSecretReference** | **object** | 클라이언트 보안이 포함된 시크릿 또는 구성 맵에 대한 참조 |
| **type** | **string** | 사용할 SASL 인증 유형 또는 SASL을 사용하지 않는 경우 **DISABLED** |

### 9.1.21. .spec.exporters[].kafka.sasl.clientIDReference

설명

클라이언트 **ID**가 포함된 시크릿 또는 구성 맵에 대한 참조

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **file** | **string** | 구성 맵 또는 시크릿 내의 파일 이름 |
| **name** | **string** | 파일이 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 파일이 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 파일 참조에 대해 "configmap" 또는 "secret"을 입력합니다. |

### 9.1.22. .spec.exporters[].kafka.sasl.clientSecretReference

설명

클라이언트 보안이 포함된 시크릿 또는 구성 맵에 대한 참조

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **file** | **string** | 구성 맵 또는 시크릿 내의 파일 이름 |
| **name** | **string** | 파일이 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 파일이 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 파일 참조에 대해 "configmap" 또는 "secret"을 입력합니다. |

### 9.1.23. .spec.exporters[].kafka.tls

설명

**TLS** 클라이언트 구성**. TLS**를 사용하는 경우 주소가 **TLS**에 사용되는 **Kafka** 포트**(**일반적으로 **9093)**와 일치하는지 확인합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **caCert** | **object** | **cacert**는 인증 기관의 인증서 참조를 정의합니다. |
| **enable** | **boolean** | TLS 활성화 |
| **insecureSkipVerify** | **boolean** | **insecureSkipVerify** 를 사용하면 서버 인증서의 클라이언트 측 확인을 건너뛸 수 있습니다. true로 설정하면 **caCert** 필드가 무시됩니다. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **userCert** | **object** | **UserCert** 는 사용자 인증서 참조를 정의하고 mTLS에 사용됩니다 (단방향 TLS 사용 시 무시할 수 있음) |

### 9.1.24. .spec.exporters[].kafka.tls.caCert

설명

**cacert**는 인증 기관의 인증서 참조를 정의합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.25. .spec.exporters[].kafka.tls.userCert

설명

**UserCert** 는 사용자 인증서 참조를 정의하고 **mTLS**에 사용됩니다(단방향 **TLS** 사용 시 무시할 수 있음)

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.26. .spec.kafka

설명

**Kafka** 구성을 통해 **Kafka**를 흐름 컬렉션 파이프라인의 일부로 브로커로 사용할 수 있습니다**. spec.deploymentModel** 이 **KAFKA** 인 경우 사용할 수 있습니다**.**

유형

**object**

필수 항목

- **address**

- **topic**

| 속성 | 유형 | 설명 |
|---|---|---|
| **address** | **string** | Kafka 서버의 주소 |
| **SASL** | **object** | SASL 인증 구성. [지원되지 않음 (*)]. |
| **tls** | **object** | TLS 클라이언트 구성. TLS를 사용하는 경우 주소가 TLS에 사용되는 Kafka 포트(일반적으로 9093)와 일치하는지 확인합니다. |
| **topic** | **string** | 사용할 Kafka 주제입니다. 네트워크 관찰 기능(Network Observability)이 존재하지 않아야 합니다. |

### 9.1.27. .spec.kafka.sasl

설명

**SASL** 인증 구성**. [**지원되지 않음**(*)].**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **clientIDReference** | **object** | 클라이언트 ID가 포함된 시크릿 또는 구성 맵에 대한 참조 |
| **clientSecretReference** | **object** | 클라이언트 보안이 포함된 시크릿 또는 구성 맵에 대한 참조 |
| **type** | **string** | 사용할 SASL 인증 유형 또는 SASL을 사용하지 않는 경우 **DISABLED** |

### 9.1.28. .spec.kafka.sasl.clientIDReference

설명

클라이언트 **ID**가 포함된 시크릿 또는 구성 맵에 대한 참조

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **file** | **string** | 구성 맵 또는 시크릿 내의 파일 이름 |
| **name** | **string** | 파일이 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 파일이 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 파일 참조에 대해 "configmap" 또는 "secret"을 입력합니다. |

### 9.1.29. .spec.kafka.sasl.clientSecretReference

설명

클라이언트 보안이 포함된 시크릿 또는 구성 맵에 대한 참조

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **file** | **string** | 구성 맵 또는 시크릿 내의 파일 이름 |
| **name** | **string** | 파일이 포함된 구성 맵 또는 시크릿의 이름 |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| 네임스페이스 | **string** | 파일이 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 파일 참조에 대해 "configmap" 또는 "secret"을 입력합니다. |

### 9.1.30. .spec.kafka.tls

설명

**TLS** 클라이언트 구성**. TLS**를 사용하는 경우 주소가 **TLS**에 사용되는 **Kafka** 포트**(**일반적으로 **9093)**와 일치하는지 확인합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **caCert** | **object** | **cacert**는 인증 기관의 인증서 참조를 정의합니다. |
| **enable** | **boolean** | TLS 활성화 |
| **insecureSkipVerify** | **boolean** | **insecureSkipVerify** 를 사용하면 서버 인증서의 클라이언트 측 확인을 건너뛸 수 있습니다. true로 설정하면 **caCert** 필드가 무시됩니다. |
| **userCert** | **object** | **UserCert** 는 사용자 인증서 참조를 정의하고 mTLS에 사용됩니다 (단방향 TLS 사용 시 무시할 수 있음) |

### 9.1.31. .spec.kafka.tls.caCert

설명

**cacert**는 인증 기관의 인증서 참조를 정의합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.32. .spec.kafka.tls.userCert

설명

**UserCert** 는 사용자 인증서 참조를 정의하고 **mTLS**에 사용됩니다(단방향 **TLS** 사용 시 무시할 수 있음)

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.33. .spec.loki

설명

**Loki,** 흐름 저장소**,** 클라이언트 설정**.**

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **authToken** | **string** | **authtoken**은 **Loki**에 인증할 토큰 을 가져오는 방법을 설명합니다. - **DISABLED** 는 요청으로 토큰을 보내지 않습니다. - **FORWARD** 는 권한 부여를 위해 사용자 토큰을 전달합니다. - **HOST** [deprecated (*)] - 로컬 Pod 서비스 계정을 사용하여 Loki에 인증합니다. Loki Operator를 사용하는 경우 **FORWARD** 로 설정해야 합니다. |

| 속성 | 유형 | 설명 |
|---|---|---|
| **batchSize** | **integer** | **batchSize** 는 전송 전에 누적할 로그의 최대 배치 크기(바이트)입니다. |
| **batchWait** | **string** | **batchWait** 은 일괄 처리를 보내기 전에 대기할 최대 시간입니다. |
| **enable** | **boolean** | flow를 Loki에 저장하도록 **enable** 로 설정합니다. OpenShift Container Platform 콘솔 플러그인 설치에 필요합니다. |
| **maxBackoff** | **string** | **maxBackoff** 는 재시도 사이의 클라이언트 연결의 최대 백오프 시간입니다. |
| **maxRetries** | **integer** | **maxRetries** 는 클라이언트 연결에 대한 최대 재시도 횟수입니다. |
| **minBackoff** | **string** | **minBackoff** 는 재시도 사이의 클라이언트 연결의 초기 백오프 시간입니다. |
| **querierUrl** | **string** | **querierURL** 은 Loki querier 서비스의 주소를 지정합니다. 경우 Loki ingester URL과 다릅니다. 비어 있는 경우 URL 값이 사용됩니다( Loki ingester 및 querier가 동일한 서버에 있다고 가정). Loki Operator를 사용할 때 수집 및 쿼리에서 Loki 게이트웨이를 사용하므로 이를 설정하지 마십시오. |
| **staticLabels** | 오브젝트(문자열) | **staticLabels** 는 각 흐름에 설정할 공통 레이블의 맵입니다. |
| **statusTls** | **object** | Loki 상태 URL에 대한 TLS 클라이언트 구성 |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **statusUrl** | **string** | **statusURL** 은 Loki **/ready**,**/metrics** 및 **/config** 끝점의 주소를 지정합니다. 경우 Loki querier URL과 다릅니다. 비어 있는 경우 **querierURL** 값이 사용됩니다. 이 기능은 프런트 엔드에서 오류 메시지와 일부 컨텍스트를 표시하는 데 유용합니다. Loki Operator를 사용하는 경우 Loki HTTP 쿼리 프런트 엔드 서비스로 설정합니다(예: https://loki-query-frontend-http.netobserv.svc:3100/ ). **statusUrl** 이 설정된 경우 **statusTLS** 구성이 사용됩니다. |
| **tenantID** | **string** | **tenantId** 는 각 요청에 대해 테넌트를 식별하는 Loki **X-Scope-OrgID** 입니다. Loki Operator를 사용하는 경우 특수 테넌트 모드에 해당하는 네트워크 로 설정합니다. |
| **timeout** | **string** | 시간 초과 는 최대 시간 연결/요청 제한입니다. 시간 초과가 0이면 시간 초과가 없음을 의미합니다. |
| **tls** | **object** | Loki URL에 대한 TLS 클라이언트 구성 |
| **url** | **string** | **URL** 은 흐름을 푸시하는 기존 Loki 서비스의 주소입니다. Loki Operator를 사용할 때 경로에 설정된 네트워크 테넌트를 사용하여 Loki 게이트웨이 서비스로 설정합니다(예: https://loki-gateway-http.netobserv.svc:8080/api/logs/v1/network ). |

### 9.1.34. .spec.loki.statusTls

설명

**Loki** 상태 **URL**에 대한 **TLS** 클라이언트 구성

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **caCert** | **object** | **cacert**는 인증 기관의 인증서 참조를 정의합니다. |
| **enable** | **boolean** | TLS 활성화 |
| **insecureSkipVerify** | **boolean** | **insecureSkipVerify** 를 사용하면 서버 인증서의 클라이언트 측 확인을 건너뛸 수 있습니다. true로 설정하면 **caCert** 필드가 무시됩니다. |
| **userCert** | **object** | **UserCert** 는 사용자 인증서 참조를 정의하고 mTLS에 사용됩니다 (단방향 TLS 사용 시 무시할 수 있음) |

### 9.1.35. .spec.loki.statusTls.caCert

설명

**cacert**는 인증 기관의 인증서 참조를 정의합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.36. .spec.loki.statusTls.userCert

설명

**UserCert** 는 사용자 인증서 참조를 정의하고 **mTLS**에 사용됩니다(단방향 **TLS** 사용 시 무시할 수 있음)

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |

| 속성 | 유형 | 설명 |
|---|---|---|
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.37. .spec.loki.tls

설명

**Loki URL**에 대한 **TLS** 클라이언트 구성

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **caCert** | **object** | **cacert**는 인증 기관의 인증서 참조를 정의합니다. |
| **enable** | **boolean** | TLS 활성화 |
| **insecureSkipVerify** | **boolean** | **insecureSkipVerify** 를 사용하면 서버 인증서의 클라이언트 측 확인을 건너뛸 수 있습니다. true로 설정하면 **caCert** 필드가 무시됩니다. |
| **userCert** | **object** | **UserCert** 는 사용자 인증서 참조를 정의하고 mTLS에 사용됩니다 (단방향 TLS 사용 시 무시할 수 있음) |

### 9.1.38. .spec.loki.tls.caCert

설명

**cacert**는 인증 기관의 인증서 참조를 정의합니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.39. .spec.loki.tls.userCert

설명

**UserCert** 는 사용자 인증서 참조를 정의하고 **mTLS**에 사용됩니다(단방향 **TLS** 사용 시 무시할 수 있음)

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |

| 속성 | 유형 | 설명 |
|---|---|---|
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.40. .spec.processor

설명

프로세서는 에이전트에서 흐름을 수신하고, 보강하고, 메트릭을 생성하고, **Loki** 지속성 계층 및/또는 사용 가능한 내보내기로 전달하는 구성 요소의 설정을 정의합니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **clusterName** | **string** | **cluster** name은 흐름 데이터에 표시할 클러스터의 이름입니다. 이는 다중 클러스터 컨텍스트에서 유용합니다. OpenShift Container Platform을 사용하는 경우 자동으로 결정되도록 비워 둡니다. |
| **conversationEndTimeout** | **string** | **conversationEndTimeout** 은 대화가 종료되었다고 간주하기 위해 네트워크 흐름을 수신한 후 대기하는 시간입니다. 이 지연은 FIN 패킷이 TCP 흐름에 대해 수집될 때 무시됩니다(대신 **conversationTerminatingTimeout** 참조). |
| **conversationHeartbeatInterval** | **string** | **conversationHeartbeatInterval** 은 대화의 "틱" 이벤트 사이에 대기하는 시간입니다. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **conversationTerminatingTimeout** | **string** | **conversationTerminatingTimeout** 은 대화를 종료하기 위해 감지된 FIN 플래그에서 대기하는 시간입니다. TCP 흐름에만 관련이 있습니다. |
| **debug** | **object** | **debug** 를 사용하면 흐름 프로세서의 내부 구성의 일부 측면을 설정할 수 있습니다. 이 섹션은 GOGC 및 GOMAXPROCS env vars와 같은 디버깅 및 세분화된 성능 최적화를 위한 것입니다. 값을 설정하는 사용자는 자신의 위험에 이를 수행합니다. |
| **dropUnusedFields** | **boolean** | **dropUnusedFields** 를 사용하면 true로 설정하면 OVS에서 사용하지 않는 것으로 알려진 필드를 드롭하여 스토리지 공간을 절약할 수 있습니다. |
| **enableKubeProbes** | **boolean** | **enableKubeProbes** 는 Kubernetes 활성 및 준비 상태 프로브를 활성화하거나 비활성화하는 플래그입니다. |
| **healthPort** | **integer** | **healthPort** 는 상태 점검 API를 노출하는 Pod의 수집기 HTTP 포트입니다. |
| **imagePullPolicy** | **string** | **imagePullPolicy** 는 위에 정의된 이미지의 Kubernetes 가져오기 정책입니다. |
| **kafkaConsumerAutoscaler** | **object** | **kafkaConsumerAutoscaler** 는 Kafka 메시지를 사용하는 **flowlogs-pipeline-transformer** 에 대해 설정하는 수평 Pod 자동 스케일러의 사양입니다. 이 설정은 Kafka가 비활성화되면 무시됩니다. HorizontalPodAutoscaler 문서(autoscaling/v2)를 참조하십시오. |
| **kafkaConsumerBatchSize** | **integer** | **kafkaConsumerBatchSize** 는 브로커에게 소비자가 허용하는 최대 배치 크기(바이트)를 나타냅니다. Kafka를 사용하지 않을 때는 무시됩니다. 기본값: 10MB. |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **kafkaConsumerQueueCapacity** | **integer** | **kafkaConsumerQueueCapacity** 는 Kafka 소비자 클라이언트에서 사용되는 내부 메시지 큐의 용량을 정의합니다. Kafka를 사용하지 않을 때는 무시됩니다. |
| **kafkaConsumerReplicas** | **integer** | **kafkaConsumerReplicas** 는 Kafka 메시지를 사용하는 **flowlogs-pipeline-transformer** 용으로 시작할 복제본(pod) 수를 정의합니다. 이 설정은 Kafka가 비활성화되면 무시됩니다. |
| **logLevel** | **string** | 프로세서 런타임의 로그 수준 |
| **logTypes** | **string** | **logTypes** 는 생성할 원하는 레코드 유형을 정의합니다. 가능한 값은 다음과 같습니다.<br>- **FLOWS** (기본값) 일반 네트워크 흐름을 내보낼 수 있음<br>- 시작된 대화에 대한 이벤트를 생성하기 위해 **CONVERSATIONS** (시작된 대화) 및 주기적인 "tick" 업데이트<br>- **ENDED_CONVERSATIONS** to generate only ended conversation events<br>- **ALL** to generate both network flows and all conversation events |
| 메트릭 | **object** | 메트릭 은 메트릭과 관련된 프로세서 구성을 정의합니다. |
| **port** | **integer** | 흐름 수집기(호스트 포트)의 포트입니다. 일부 값은 허용되지 않습니다. 1024보다 크고 4500, 4789 및 6081과 달라야 합니다. |
| **profilePort** | **integer** | **profilePort** 를 사용하면 이 포트를 수신하는 Go pprof 프로파일러를 설정할 수 있습니다. |
| **resources** | **object** | 리소스는 이 컨테이너에 필요한 컴퓨팅 리소스입니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |

### 9.1.41. .spec.processor.debug

설명

**debug** 를 사용하면 흐름 프로세서의 내부 구성의 일부 측면을 설정할 수 있습니다. 이 섹션은 **GOGC** 및 **GOMAXPROCS env vars**와 같은 디버깅 및 세분화된 성능 최적화를 위한 것입니다. 값을 설정하는 사용자는 자신의 위험에 이를 수행합니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **env** | 오브젝트(문자열) | **env** 를 사용하면 사용자 지정 환경 변수를 기본 구성 요소에 전달할 수 있습니다. GoGC 및 GOMAXPROCS와 같은 매우 구체적인 성능 튜닝 옵션을 전달하는 데 유용합니다. 이는 에지 디버그 또는 지원 시나리오에서만 유용하므로 FlowCollector 설명자의 일부로 공개적으로 노출해서는 안 됩니다. |

### 9.1.42. .spec.processor.kafkaConsumerAutoscaler

설명

**kafkaConsumerAutoscaler** 는 **Kafka** 메시지를 사용하는 **flowlogs-pipeline-transformer** 에 대해 설정하는 수평 **Pod** 자동 스케일러의 사양입니다. 이 설정은 **Kafka**가 비활성화되면 무시됩니다. **HorizontalPodAutoscaler** 문서**(autoscaling/v2)**를 참조하십시오.

유형

**object**

### 9.1.43. .spec.processor.metrics

설명

메트릭 은 메트릭과 관련된 프로세서 구성을 정의합니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **disableAlerts** | 배열(문자열) | **disableAlerts** 는 비활성화해야 하는 경고 목록입니다. 가능한 값은 다음과 같습니다. **NetObservNoFlows** 에서는 특정 기간 동안 흐름이 관찰되지 않을 때 트리거됩니다. **NetObservLokiError**: Loki 오류로 인해 흐름을 삭제할 때 트리거됩니다. |
| **ignoreTags** | 배열(문자열) | **ignoreTags** 는 무시할 메트릭을 지정하는 태그 목록입니다. 각 메트릭은 태그 목록과 연결됩니다. 자세한 내용은 https://github.com/netobserv/network-observability-operator/tree/main/controllers/flowlogspipeline/metrics_definitions . 사용 가능한 태그는 **egress,ingress,flows,bytes,packet,namespaces,nodes,workloads,nodes-flows,namespaces-flows,workloads-flows** 입니다. 네임스페이스 기반 메트릭은 워크로드 및 네임스페이스 태그 둘 다에서 다루지 않으므로 항상 해당 중 하나를 무시하는 것이 좋습니다(분별한워크로드 제공). |
| **server** | **object** | Prometheus 스크랩에 대한 메트릭 서버 끝점 구성 |

### 9.1.44. .spec.processor.metrics.server

설명

**Prometheus** 스크랩에 대한 메트릭 서버 끝점 구성

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **port** | **integer** | prometheus HTTP 포트 |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **tls** | **object** | TLS 구성입니다. |

### 9.1.45. .spec.processor.metrics.server.tls

설명

**TLS** 구성입니다.

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **insecureSkipVerify** | **boolean** | **insecureSkipVerify** 를 사용하면 제공된 인증서의 클라이언트 측 확인을 건너뛸 수 있습니다. true로 설정하면 **providedCaFile** 필드가 무시됩니다. |
| 제공됨 | **object** | 유형이 **PROVIDED** 로 설정된 경우 TLS 구성입니다. |
| **providedCaFile** | **object** | **type** 이 **PROVIDED** 로 설정된 경우 CA 파일에 대한 참조입니다. |
| **type** | **string** | TLS 구성 유형<br>- 엔드포인트에 대해 TLS를 구성하지 않도록 **DISABLED** (기본값)를 선택합니다. - 인증서 파일과 키 파일을 수동으로 제공하는 **PROVIDED**. - **AUTO** 를 사용하여 주석을 사용하여 OpenShift Container Platform 자동 생성 인증서를 사용합니다. |

### 9.1.46. .spec.processor.metrics.server.tls.provided

설명

유형이 **PROVIDED** 로 설정된 경우 **TLS** 구성입니다.

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **certFile** | **string** | **cert** file은 구성 맵 또는 시크릿 내의 인증서 파일 이름의 경로를 정의합니다. |
| **certKey** | **string** | **certKey** 는 구성 맵 또는 시크릿 내의 인증서 개인 키 파일 이름의 경로를 정의합니다. 키가 필요하지 않은 경우 생략합니다. |
| **name** | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 이름 |
| 네임스페이스 | **string** | 인증서가 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 인증서 참조에 대한 유형: **configmap** 또는 **secret** |

### 9.1.47. .spec.processor.metrics.server.tls.providedCaFile

설명

**type** 이 **PROVIDED** 로 설정된 경우 **CA** 파일에 대한 참조입니다**.**

유형

**object**

| 속성 | 유형 | 설명 |
|---|---|---|
| **file** | **string** | 구성 맵 또는 시크릿 내의 파일 이름 |
| **name** | **string** | 파일이 포함된 구성 맵 또는 시크릿의 이름 |

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| 네임스페이스 | **string** | 파일이 포함된 구성 맵 또는 시크릿의 네임스페이스입니다. 생략하면 기본값은 Network Observability가 배포된 동일한 네임스페이스를 사용하는 것입니다. 네임스페이스가 다르면 필요에 따라 마운트할 수 있도록 구성 맵 또는 시크릿이 복사됩니다. |
| **type** | **string** | 파일 참조에 대해 "configmap" 또는 "secret"을 입력합니다. |

### 9.1.48. .spec.processor.resources

설명

리소스는 이 컨테이너에 필요한 컴퓨팅 리소스입니다. 자세한 내용은 **https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/**

유형

**object**

| 속성 | 유형 | 설명 |
| --- | --- | --- |
| **limits** | **integer-or-string** | 제한은 허용되는 최대 컴퓨팅 리소스 양을 나타냅니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |
| **requests** | **integer-or-string** | 요청은 필요한 최소 컴퓨팅 리소스 양을 설명합니다. 컨테이너에 대한 Requests를 생략하면 구현 정의된 값을 제외하고 명시적으로 지정된 경우 기본값은 Limits로 설정됩니다. 요청은 제한을 초과할 수 없습니다. 자세한 내용은 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ |

# 10장. 네트워크 흐름 형식 참조

이는 내부 및 Kafka로 흐름을 내보낼 때 네트워크 흐름 형식의 사양입니다.

## 10.1. 네트워크 흐름 형식 참조

이는 내부 및 Kafka로 흐름을 내보낼 때 사용되는 네트워크 흐름 형식의 사양입니다.

이 문서는 두 가지 주요 카테고리로 구성됩니다. *레이블* 과 일반 *필드*. 이 차이점은 Loki를 쿼리할 때만 중요합니다. *레이블* 은 *필드* 와 달리 스트림 선택기 에서 사용해야 하기 때문입니다.

이 사양을 Kafka 내보내기 기능에 대한 참조로 읽는 경우 모든 *라벨* 과 필드를 일반 필드로 처리하고 Loki와 관련된 구분을 무시해야 합니다.

### *10.1.1. 라벨*

***SrcK8S_Namespace***

- *선택적* ***SrcK8S_Namespace:string***

*소스 네임스페이스*

***DstK8S_Namespace***

- *선택적* ***DstK8S_Namespace:string***

*대상 네임스페이스*

***SrcK8S_OwnerName***

- *선택적* ***SrcK8S_OwnerName:string***

***Deployment, StatefulSet*** *등과 같은 소스 소유자.*

***DstK8S_OwnerName***

- *선택적* ***DstK8S_OwnerName:string***

*배포,* ***StatefulSet*** *등과 같은 대상 소유자.*

***FlowDirection***

- ***FlowDirection:FlowDirection (****다음 섹션 참조****, Enumeration: FlowDirection)***

*노드 관찰 지점의 흐름 방향*

***_RecordType***

- *선택 사항* ***_RecordType:RecordType***

*레코드 유형:* *일반 흐름 로그의 경우* ***'flowLog'*** *또는* ***'allConnections', 'newConnection', 'heartbeat', 'endConnection',*** *대화 추적을 위한* ***'endConnection'***

### ***10.1.2.*** *필드

***SrcAddr***

- ***srcAddr:string***

*소스 **IP** 주소(**ipv4** 또는 **ipv6**)*

***DstAddr***

- ***DstAddr: string***

*대상 **IP** 주소(**ipv4** 또는 **ipv6**)*

***SrcMac***

- ***srcMac:string***

*소스 **MAC** 주소*

***DstMac***

- ***DstMac: string***

*대상 **MAC** 주소*

***SrcK8S_Name***

- *선택적* ***SrcK8S_Name:string***

***Pod*** *이름,* *서비스 이름 등과 같은* ***Kubernetes*** *오브젝트와 일치하는 소스의 이름입니다.*

***DstK8S_Name***

- *선택 사항* ***DstK8S_Name:string***

***Pod*** *이름,* *서비스 이름 등과 같은* ***Kubernetes*** *오브젝트와 일치하는 대상의 이름입니다.*

***SrcK8S_Type***

- *선택적* ***SrcK8S_Type:string***

***Pod,*** *서비스 등과 같은* ***Kubernetes*** *오브젝트와 일치하는 소스의 종류입니다.*

***DstK8S_Type***

- *선택적* ***DstK8S_Type:string***

*Pod 이름, 서비스 이름 등과 같은 Kubernetes 오브젝트와 일치하는 대상의 종류입니다.*

***SrcPort***

- *선택적 SrcPort: 번호*

*소스 포트*

***DstPort***

- *선택적 DstPort: 번호*

*대상 포트*

***SrcK8S_OwnerType***

- *선택적 SrcK8S_OwnerType:string*

*Deployment, StatefulSet 등과 같은 소스 Kubernetes 소유자의 종류입니다.*

***DstK8S_OwnerType***

- *Optional DstK8S_OwnerType: string*

***Deployment, StatefulSet** 등과 같은 대상 **Kubernetes** 소유자의 종류입니다.*

***SrcK8S_HostIP***

- *선택적* ***SrcK8S_HostIP:string***

*소스 노드 **IP***

***DstK8S_HostIP***

- *선택적* ***DstK8S_HostIP:string***

*대상 노드 **IP***

***SrcK8S_HostName***

- *선택적* ***SrcK8S_HostName:string***

*소스 노드 이름*

***DstK8S_HostName***

- *선택 사항* ***DstK8S_HostName:string***

*대상 노드 이름*

***proto***

- *프로토**:**번호*

***L4** 프로토콜*

*인터페이스*

- *선택적 인터페이스**:**문자열*

*네트워크 인터페이스*

***IfDirection***

- *선택적* ***IfDirection:InterfaceDirection (****다음 섹션 참조****, Enumeration: InterfaceDirection)***

*네트워크 인터페이스 관찰 지점의 흐름 방향*

*플래그*

- *선택적 플래그:번호*

***TCP** 플래그*

*패킷*

- *선택적 패킷:번호*

*패킷 수*

***Packets_AB***

- *선택적 **Packets_AB:number***

*대화 추적에서 **A ~ B** 패킷에 대해 대화당 대응*

***Packets_BA***

- *선택적 **Packets_BA:number***

*대화 추적에서 **B**는 대화당 패킷 카운터로*

*바이트*

- *선택적* ***Cryostat:****번호*

*바이트 수*

***Bytes_AB***

- *선택적* ***Cryostat_AB:****번호*

*대화 추적에서* ***A~B****바이트 카운터는 대화당*

***Bytes_BA***

- *선택적* ***Cryostat_BA:****번호*

*대화 추적에서* ***B****에서* ***B****로 대화당 바이트 카운터*

***IcmpType***

- *선택적* ***IcmpType:number***

***ICMP*** *유형*

***IcmpCode***

- *선택적* ***IcmpCode:number***

***ICMP*** *코드*

***PktDropLatestState***

- ***Optional PktDropLatestState: string***

*드롭 다운의* ***PKT TCP*** *상태*

***PktDropLatestDropCause***

- ***Optional PktDropLatestDropCause: string***

*드롭 다운의* ***PKT*** *원인*

***PktDropLatestFlags***

- *선택적* ***PktDropLatestFlags:number***

*드롭 다운의* ***PKT TCP*** *플래그*

***PktDropPackets***

- *선택적* ***PktDropPackets:number***

*커널에 의해 삭제된 패킷 수*

***PktDropPackets_AB***

- ***Optional PktDropPackets_AB: number***

*대화 추적에서* ***A ~ B*** *패킷이 대화당 카운터를 삭제했습니다.*

***PktDropPackets_BA***

- ***Optional PktDropPackets_BA: number***

*대화 추적에서 패킷 간* ***B****는 대화당 카운터를 중단했습니다.*

***PktDropBytes***

- *선택적* ***PktDropBytes:****번호*

*커널에 의해 삭제된 바이트 수*

***PktDropBytes_AB***

- ***Optional PktDropBytes_AB: number***

*대화 추적에서 **A**에서 **B** 바이트로 대화당 카운터를 삭제했습니다.*

***PktDropBytes_BA***

- ***Optional PktDropBytes_BA: number***

*대화 추적에서 **B ~ A** 바이트가 대화당 카운터를 삭제했습니다.*

***DnsId***

- *선택적 **DnsId:**번호*

***DNS** 레코드 **ID***

***DnsFlags***

- *선택적 **DnsFlags:**번호*

*DNS 레코드의 DNS 플래그*

***DnsFlagsResponseCode***

- *선택적 **DnsFlagsResponseCode:string***

*구문 분석된 DNS 헤더 RCODEs 이름*

***DnsLatencyMs***

- *선택적 **DnsLatencyMs:number***

*응답과 요청 사이의 계산된 시간(밀리초)*

***TimeFlowStartMs***

- ***TimeFlowStartMs:**번호*

*이 흐름의 타임스탬프 시작(밀리초)*

***TimeFlowEndMs***

- ***TimeFlowEndMs: number***

*이 흐름의 종료 타임스탬프(밀리초)*

***TimeReceived***

- ***TimeReceived:*** *번호*

*흐름 수집기에서 이 흐름을 수신하고 처리하는 타임스탬프(초)*

***TimeFlowRttNs***

- *선택적* ***TimeFlowRttNs:*** *번호*

***flow Round Trip Time (RTT) in nanoseconds***

***_HashId***

- *선택 사항* ***_HashId:string***

*대화 추적에서 대화 식별자*

***_IsFirst***

- *선택 사항* ***_IsFirst:*** *문자열*

*대화 추적에서 첫 번째 흐름을 식별하는 플래그*

***numFlowLogs***

- *선택적* ***numFlowLogs:****번호*

*대화 추적에서, 대화당 흐름 로그의 카운터*

#### *10.1.3. enumeration: FlowDirection*

***Ingress***

- ***Ingress = "0"***

*노드 관찰 시점에서 들어오는 트래픽*

***Egress***

- ***egress = "1"***

*노드 관찰 시점에서 나가는 트래픽*

*내부*

- *내부* ***= "2"***

*동일한 소스 및 대상 노드가 있는 내부 트래픽*

# 11장. 네트워크 OBSERVABILITY 문제 해결

네트워크 Observability 문제 해결을 지원하기 위해 일부 문제 해결 작업을 수행할 수 있습니다.

## 11.1. MUST-GATHER 툴 사용

must-gather 툴을 사용하여 Network Observability Operator 리소스 및 Pod 로그, FlowCollector, Webhook 구성과 같은 클러스터 전체 리소스에 대한 정보를 수집할 수 있습니다.

프로세스

1. must-gather 데이터를 저장하려는 디렉터리로 이동합니다.
2. 다음 명령을 실행하여 클러스터 전체 must-gather 리소스를 수집합니다.

```
$ oc adm must-gather
 --image-stream=openshift/must-gather \
 --image=quay.io/netobserv/must-gather
```

## 11.2. OPENSHIFT CONTAINER PLATFORM 콘솔에서 네트워크 트래픽 메뉴 항목 구성

OpenShift Container Platform 콘솔의 Observe 메뉴에 네트워크 트래픽 메뉴 항목이 나열되지 않은 경우 OpenShift Container Platform 콘솔에서 네트워크 트래픽 메뉴 항목을 수동으로 구성합니다.

사전 요구 사항

- OpenShift Container Platform 버전 4.10 이상이 설치되어 있어야 합니다.

프로세스

1. 다음 명령을 실행하여 spec.consolePlugin.register 필드가 true 로 설정되어 있는지 확인합니다.

```
$ oc -n netobserv get flowcollector cluster -o yaml
```

출력 예

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  consolePlugin:
    register: false
```

2. *선택 사항:* ***Console Operator*** *구성을 수동으로 편집하여* ***netobserv-plugin*** *플러그인을 추가합니다.*

```
$ oc edit console.operator.openshift.io cluster
```

*출력 예*

```
...
spec:
  plugins:
  - netobserv-plugin
...
```

3. *선택 사항:* *다음 명령을 실행하여* ***spec.consolePlugin.register*** *필드를* ***true*** *로 설정합니다.*

```
$ oc -n netobserv edit flowcollector cluster -o yaml
```

*출력 예*

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
spec:
  consolePlugin:
    register: true
```

4. *다음 명령을 실행하여 콘솔* ***Pod****의 상태가 실행 중인지 확인합니다.*

```
$ oc get pods -n openshift-console -l app=console
```

5. *다음 명령을 실행하여 콘솔* ***Pod****를 다시 시작합니다.*

```
$ oc delete pods -n openshift-console -l app=console
```

6. *브라우저 캐시 및 기록을 지웁니다.*

7. *다음 명령을 실행하여* ***Network Observability*** *플러그인* ***Pod****의 상태를 확인합니다.*

```
$ oc get pods -n netobserv -l app=netobserv-plugin
```

*출력 예*

```
NAME                                READY   STATUS    RESTARTS   AGE
netobserv-plugin-68c7bbb9bb-b69q6   1/1     Running   0          21s
```

8. *다음 명령을 실행하여* ***Network Observability*** *플러그인* ***Pod****의 로그를 확인합니다.*

```
$ oc logs -n netobserv -l app=netobserv-plugin
```

*출력 예*

```
time="2022-12-13T12:06:49Z" level=info msg="Starting netobserv-console-plugin
[build version: , build date: 2022-10-21 15:15] at log level info" module=main
time="2022-12-13T12:06:49Z" level=info msg="listening on https://:9001"
module=server
```

### 11.3. FLOWLOGS-PIPELINE은 KAFKA를 설치한 후 네트워크 흐름을 사용하지 않습니다.

**deploymentModel: KAFKA** 를 사용하여 흐름 수집기를 먼저 배포한 다음 **Kafka**를 배포한 경우 흐름 수집기가 **Kafka**에 올바르게 연결되지 않을 수 있습니다. **Flowlogs-pipeline**이 **Kafka**의 네트워크 흐름을 사용하지 않는 **flow-pipeline Pod**를 수동으로 다시 시작합니다.

프로세스

1. 다음 명령을 실행하여 **flow-pipeline** 포드를 삭제하여 다시 시작합니다.

```
$ oc delete pods -n netobserv -l app=flowlogs-pipeline-transformer
```

### 11.4. BR-INT 및 BR-EX 인터페이스 모두에서 네트워크 흐름을 볼 수 없음

**br-ex'** 및 **br-int** 는 **OSI** 계층 **2**에서 작동하는 가상 브릿지 장치입니다. **eBPF** 에이전트는 **IP** 및 **TCP** 수준, 계층 **3** 및 **4**에서 각각 작동합니다. **eBPF** 에이전트는 네트워크 트래픽이 물리적 호스트 또는 가상 **pod** 인터페이스와 같은 다른 인터페이스에서 처리할 때 **br-ex** 및 **br-int** 를 통과하는 네트워크 트래픽을 캡처할 수 있습니다. **eBPF** 에이전트 네트워크 인터페이스를 **br-ex** 및 **br-int** 에만 연결하도록 제한하면 네트워크 흐름이 표시되지 않습니다.

인터페이스의 부분을 수동으로 제거하거나 네트워크 인터페이스를 **br-int** 및 **br-ex** 로 제한하는 **Interfaces**를 제외합니다.

프로세스

1. 인터페이스 제거: **['br-int', 'br-ex' ]** 필드. 이를 통해 에이전트는 모든 인터페이스에서 정보를 가져올 수 있습니다. 또는 **Layer-3** 인터페이스를 지정할 수 있습니다(예: **eth0** ). 다음 명령을 실행합니다.

```
$ oc edit -n netobserv flowcollector.yaml -o yaml
```

출력 예

```
apiVersion: flows.netobserv.io/v1alpha1
kind: FlowCollector
metadata:
  name: cluster
```

```
spec:
  agent:
    type: EBPF
    ebpf:
      interfaces: [ 'br-int', 'br-ex' ] 1
```

1

네트워크 인터페이스를 지정합니다.

## 11.5. NETWORK OBSERVABILITY 컨트롤러 관리자 POD가 메모리 부족

**Subscription** 오브젝트에서 **spec.config.resources.limits.memory** 사양을 편집하여 **Network Observability Operator**의 메모리 제한을 늘릴 수 있습니다.

프로세스

1. 웹 콘솔에서 **Operator** → 설치된 **Operator**로 이동합니다.
2. 네트워크 관찰 기능을 클릭한 다음 서브스크립션 을 선택합니다.
3. 작업 메뉴에서 서브스크립션 편집을 클릭합니다.

   a. 또는 **CLI**를 사용하여 다음 명령을 실행하여 **Subscription** 오브젝트에 대한 **YAML** 구성을 열 수 있습니다.

   ```
   $ oc edit subscription netobserv-operator -n openshift-netobserv-operator
   ```

4. **Subscription** 오브젝트를 편집하여 **config.resources.limits.memory** 사양을 추가하고 메모리 요구 사항을 고려하여 값을 설정합니다. 리소스 고려 사항에 대한 자세한 내용은 추가 리소스를 참조하십시오.

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

```
spec:
  channel: stable
  config:
    resources:
      limits:
        memory: 800Mi    (1)
      requests:
        cpu: 100m
        memory: 100Mi
  installPlanApproval: Automatic
  name: netobserv-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: <network_observability_operator_latest_version>    (2)
```

(1) 예를 들어 메모리 제한을 **800Mi** 로 늘릴 수 있습니다.

(2) 이 값은 편집해서는 안 되지만 **Operator**의 최신 릴리스에 따라 변경됩니다.

추가 리소스

- 리소스 고려 사항

### 11.6. LOKI RESOURCEEXHAUSTED 오류 문제 해결

네트워크 **Observability**에서 보낸 네트워크 흐름 데이터가 구성된 최대 메시지 크기를 초과하면 **Loki**가 **ResourceExhausted** 오류를 반환할 수 있습니다. **Red Hat Loki Operator**를 사용하는 경우 이 최대 메시지 크기는 **100MiB**로 구성됩니다.

프로세스

1. **Operator** → 설치된 **Operator** 로 이동하여 프로젝트 드롭다운 메뉴에서 모든 프로젝트를 확인합니다.
2. 제공된 **API** 목록에서 **Network Observability Operator**를 선택합니다.
3. 흐름 수집기 를 클릭한 다음 **YAML** 보기 탭을 클릭합니다.

a. *Loki Operator*를 사용하는 경우 **spec.loki.batchSize** 값이 **98MiB**를 초과하지 않는지 확인합니다.

b. *Grafana Loki*와 같은 **Red Hat Loki Operator**와 다른 **Loki** 설치 방법을 사용하는 경우 **grpc_server_max_recv_msg_size** **Loki** 서버 설정이 **FlowCollector** 리소스 **spec.loki.batchSize** 값보다 높은지 확인합니다. 그렇지 않은 경우 **grpc_server_max_recv_msg_size** 값을 늘리거나 제한보다 낮도록 **spec.loki.batchSize** 값을 줄여야 합니다.

4. **FlowCollector** 를 편집한 경우 저장을 클릭합니다.

## 11.7. 리소스 문제 해결

## 11.8. LOKISTACK 속도 제한 오류

**Loki** 테넌트에 배치된 속도 제한은 스트림 제한 초과**(limit:xMB/sec)**를 수집하는 동안 데이터 및 **429** 오류가 발생할 수 있습니다. 이 오류를 알리기 위해 경고를 설정하는 것이 좋습니다. 자세한 내용은 이 섹션의 추가 리소스의 **"NetObserv** 대시보드에 대한 **Loki** 속도 제한 경고 생성**"**을 참조하십시오.

다음 절차에 표시된 대로 **perStreamRateLimit** 및 **perStreamRateLimitBurst** 사양을 사용하여 **LokiStack CRD**를 업데이트할 수 있습니다.

프로세스

1. **Operator** → 설치된 **Operator** 로 이동하여 프로젝트 드롭다운에서 모든 프로젝트를 확인합니다.

2. **Loki Operator** 를 찾고 **LokiStack** 탭을 선택합니다.

3. **YAML** 보기를 사용하여 기존 **LokiStack** 인스턴스를 생성하거나 편집하여 **perStreamRateLimit** 및 **perStreamRateLimitBurst** 사양을 추가합니다.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: loki
  namespace: netobserv
spec:
```

```
limits:
  global:
    ingestion:
      perStreamRateLimit: 6 (1)
      perStreamRateLimitBurst: 30 (2)
tenants:
  mode: openshift-network
managementState: Managed
```

(1) ***perStreamRateLimit** 의 기본값은 **3** 입니다.*

(2) ***perStreamRateLimitBurst** 의 기본값은 **15** 입니다.*

4. *저장을 클릭합니다.*

*검증*

***perStreamRateLimit** 및 **perStreamRateLimitBurst** 사양을 업데이트하면 클러스터 재시작의 **pod**가 **429 rate-limit** 오류가 더 이상 발생하지 않습니다.*