```
  name: ingressnodefirewall
spec:
  interfaces:
  - eth0
  nodeSelector:
    matchLabels:
      <ingress_firewall_label_name>: <label_value> (1)
  ingress:
  - sourceCIDRs:
       - 172.16.0.0/12
    rules:
    - order: 10
      protocolConfig:
        protocol: ICMP
        icmp:
          icmpType: 8 #ICMP Echo request
      action: Deny
    - order: 20
      protocolConfig:
        protocol: TCP
        tcp:
          ports: "8000-9000"
      action: Deny
  - sourceCIDRs:
       - fc00:f853:ccd:e793::0/64
    rules:
    - order: 10
      protocolConfig:
        protocol: ICMPv6
        icmpv6:
          icmpType: 128 #ICMPV6 Echo request
      action: Deny
```

1. A <label_name> and a <label_value> must exist on the node and must match the **nodeselector** label and value applied to the nodes you want the **ingressfirewallconfig** CR to run on. The <label_value> can be **true** or **false**. By using **nodeSelector** labels, you can target separate groups of nodes to apply different rules to using the **ingressfirewallconfig** CR.

Zero trust Ingress Node Firewall rules object example
Zero trust Ingress Node Firewall rules can provide additional security to multi-interface clusters. For example, you can use zero trust Ingress Node Firewall rules to drop all traffic on a specific interface except for SSH.

A complete configuration of a zero trust Ingress Node Firewall rule set is specified in the following example:

IMPORTANT

Users need to add all ports their application will use to their allowlist in the following case to ensure proper functionality.

Example zero trust Ingress Node Firewall rules

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
```

```
kind: IngressNodeFirewall
metadata:
 name: ingressnodefirewall-zero-trust
spec:
 interfaces:
 - eth1 (1)
 nodeSelector:
   matchLabels:
     <ingress_firewall_label_name>: <label_value> (2)
 ingress:
 - sourceCIDRs:
      - 0.0.0.0/0 (3)
   rules:
   - order: 10
     protocolConfig:
       protocol: TCP
       tcp:
         ports: 22
     action: Allow
   - order: 20
     action: Deny (4)
```

1. Network-interface cluster
2. The <label_name> and <label_value> needs to match the **nodeSelector** label and value applied to the specific nodes with which you wish to apply the **ingressfirewallconfig** CR.
3. **0.0.0.0/0** set to match any CIDR
4. **action** set to **Deny**

IMPORTANT

eBPF Manager Operator integration is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

4.9.4. Ingress Node Firewall Operator integration

The Ingress Node Firewall uses eBPF programs to implement some of its key firewall functionality. By default these eBPF programs are loaded into the kernel using a mechanism specific to the Ingress Node Firewall. You can configure the Ingress Node Firewall Operator to use the eBPF Manager Operator for loading and managing these programs instead.

When this integration is enabled, the following limitations apply:

- The Ingress Node Firewall Operator uses TCX if XDP is not available and TCX is incompatible with bpfman.

- The Ingress Node Firewall Operator daemon set pods remain in the **ContainerCreating** state until the firewall rules are applied.
- The Ingress Node Firewall Operator daemon set pods run as privileged.

4.9.5. Configuring Ingress Node Firewall Operator to use the eBPF Manager Operator

The Ingress Node Firewall uses eBPF programs to implement some of its key firewall functionality. By default these eBPF programs are loaded into the kernel using a mechanism specific to the Ingress Node Firewall.

As a cluster administrator, you can configure the Ingress Node Firewall Operator to use the eBPF Manager Operator for loading and managing these programs instead, adding additional security and observability functionality.

Prerequisites

- You have installed the OpenShift CLI (**oc**).
- You have an account with administrator privileges.
- You installed the Ingress Node Firewall Operator.
- You have installed the eBPF Manager Operator.

Procedure

1. Apply the following labels to the **ingress-node-firewall-system** namespace:

```
$ oc label namespace openshift-ingress-node-firewall \
    pod-security.kubernetes.io/enforce=privileged \
    pod-security.kubernetes.io/warn=privileged --overwrite
```

2. Edit the **IngressNodeFirewallConfig** object named **ingressnodefirewallconfig** and set the **ebpfProgramManagerMode** field:

Ingress Node Firewall Operator configuration object

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
  name: ingressnodefirewallconfig
  namespace: openshift-ingress-node-firewall
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
  ebpfProgramManagerMode: <ebpf_mode>
```

where:

<ebpf_mode>: Specifies whether or not the Ingress Node Firewall Operator uses the eBPF Manager Operator to manage eBPF programs. Must be either **true** or **false**. If unset, eBPF Manager is not used.

4.9.6. Viewing Ingress Node Firewall Operator rules

Procedure

1. Run the following command to view all current rules :

```
$ oc get ingressnodefirewall
```

2. Choose one of the returned **<resource>** names and run the following command to view the rules or configs:

```
$ oc get <resource> <name> -o yaml
```

4.9.7. Troubleshooting the Ingress Node Firewall Operator

- Run the following command to list installed Ingress Node Firewall custom resource definitions (CRD):

```
$ oc get crds | grep ingressnodefirewall
```

Example output

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
ingressnodefirewallconfigs.ingressnodefirewall.openshift.io       2022-08-25T10:03:01Z
ingressnodefirewallnodestates.ingressnodefirewall.openshift.io    2022-08-25T10:03:00Z
ingressnodefirewalls.ingressnodefirewall.openshift.io             2022-08-25T10:03:00Z
```

- Run the following command to view the state of the Ingress Node Firewall Operator:

```
$ oc get pods -n openshift-ingress-node-firewall
```

Example output

```
NAME                                       READY  STATUS         RESTARTS  AGE
ingress-node-firewall-controller-manager   2/2    Running        0         5d21h
ingress-node-firewall-daemon-pqx56         3/3    Running        0         5d21h
```

The following fields provide information about the status of the Operator: **READY**, **STATUS**, **AGE**, and **RESTARTS**. The **STATUS** field is **Running** when the Ingress Node Firewall Operator is deploying a daemon set to the assigned nodes.

- Run the following command to collect all ingress firewall node pods' logs:

```
$ oc adm must-gather – gather_ingress_node_firewall
```

The logs are available in the sos node’s report containing eBPF **bpftool** outputs at **/sos_commands/ebpf**. These reports include lookup tables used or updated as the ingress firewall XDP handles packet processing, updates statistics, and emits events.

4.9.8. Additional resources

- About the eBPF Manager Operator

4.10. SR-IOV OPERATOR

4.10.1. Installing the SR-IOV Network Operator

You can install the Single Root I/O Virtualization (SR-IOV) Network Operator on your cluster to manage SR-IOV network devices and network attachments.

4.10.1.1. Installing the SR-IOV Network Operator

As a cluster administrator, you can install the Single Root I/O Virtualization (SR-IOV) Network Operator by using the OpenShift Container Platform CLI or the web console.

4.10.1.1.1. CLI: Installing the SR-IOV Network Operator

As a cluster administrator, you can install the Operator using the CLI.

Prerequisites

- A cluster installed on bare-metal hardware with nodes that have hardware that supports SR-IOV.
- Install the OpenShift CLI (**oc**).
- An account with **cluster-admin** privileges.

Procedure

1. Create the **openshift-sriov-network-operator** namespace by entering the following command:

```
$ cat << EOF| oc create -f -
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-sriov-network-operator
  annotations:
    workload.openshift.io/allowed: management
EOF
```

2. Create an **OperatorGroup** custom resource (CR) by entering the following command:

```
$ cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: sriov-network-operators
  namespace: openshift-sriov-network-operator
spec:
  targetNamespaces:
  - openshift-sriov-network-operator
EOF
```

3. Create a **Subscription** CR for the SR-IOV Network Operator by entering the following command:

```
$ cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: sriov-network-operator-subscription
  namespace: openshift-sriov-network-operator
spec:
  channel: stable
  name: sriov-network-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

4. Create an **SriovoperatorConfig** resource by entering the following command:

```
$ cat <<EOF | oc create -f -
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default
  namespace: openshift-sriov-network-operator
spec:
  enableInjector: true
  enableOperatorWebhook: true
  logLevel: 2
  disableDrain: false
EOF
```

Verification

- Check that the Operator is installed by entering the following command:

```
$ oc get csv -n openshift-sriov-network-operator \
  -o custom-columns=Name:.metadata.name,Phase:.status.phase
```

Example output

```
Name                                    Phase
sriov-network-operator.4.18.0-202406131906   Succeeded
```

4.10.1.1.2. Web console: Installing the SR-IOV Network Operator

As a cluster administrator, you can install the Operator using the web console.

Prerequisites

- A cluster installed on bare-metal hardware with nodes that have hardware that supports SR-IOV.
- Install the OpenShift CLI (**oc**).
- An account with **cluster-admin** privileges.

Procedure

1. Install the SR-IOV Network Operator:

    a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.

    b. Select **SR-IOV Network Operator** from the list of available Operators, and then click **Install**.

    c. On the **Install Operator** page, under **Installed Namespace**, select **Operator recommended Namespace**.

    d. Click **Install**.

2. Verify that the SR-IOV Network Operator is installed successfully:

    a. Navigate to the **Operators** → **Installed Operators** page.

    b. Ensure that **SR-IOV Network Operator** is listed in the **openshift-sriov-network-operator** project with a **Status** of **InstallSucceeded**.

    > **NOTE**
    >
    > During installation an Operator might display a **Failed** status. If the installation later succeeds with an **InstallSucceeded** message, you can ignore the **Failed** message.

    If the Operator does not appear as installed, to troubleshoot further:

    - Inspect the **Operator Subscriptions** and **Install Plans** tabs for any failure or errors under **Status**.
    - Navigate to the **Workloads** → **Pods** page and check the logs for pods in the **openshift-sriov-network-operator** project.
    - Check the namespace of the YAML file. If the annotation is missing, you can add the annotation **workload.openshift.io/allowed=management** to the Operator namespace with the following command:

      ```
      $ oc annotate ns/openshift-sriov-network-operator
      workload.openshift.io/allowed=management
      ```

      > **NOTE**
      >
      > For single-node OpenShift clusters, the annotation **workload.openshift.io/allowed=management** is required for the namespace.

4.10.1.2. Next steps

- Configuring the SR-IOV Network Operator

4.10.2. Configuring the SR-IOV Network Operator

The Single Root I/O Virtualization (SR-IOV) Network Operator manages the SR-IOV network devices and network attachments in your cluster.

4.10.2.1. Configuring the SR-IOV Network Operator

- Create a **SriovOperatorConfig** custom resource (CR) to deploy all the SR-IOV Operator components:

  a. Create a file named **sriovOperatorConfig.yaml** using the following YAML:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default 1
  namespace: openshift-sriov-network-operator
spec:
  disableDrain: false
  enableInjector: true 2
  enableOperatorWebhook: true 3
  logLevel: 2
  featureGates:
    metricsExporter: false
```

1 The only valid name for the **SriovOperatorConfig** resource is **default** and it must be in the namespace where the Operator is deployed.

2 The **enableInjector** field, if not specified in the CR or explicitly set to **true**, defaults to **false** or **<none>**, preventing any **network-resources-injector** pod from running in the namespace. The recommended setting is **true**.

3 The **enableOperatorWebhook** field, if not specified in the CR or explicitly set to true, defaults to **false** or **<none>**, preventing any **operator-webhook** pod from running in the namespace. The recommended setting is **true**.

  b. Create the resource by running the following command:

```
$ oc apply -f sriovOperatorConfig.yaml
```

4.10.2.1.1. SR-IOV Network Operator config custom resource

The fields for the **sriovoperatorconfig** custom resource are described in the following table:

Table 4.16. SR-IOV Network Operator config custom resource

Field	Type	Description
metadata.name	**string**	Specifies the name of the SR-IOV Network Operator instance. The default value is **default**. Do not set a different value.
metadata.namespace	**string**	Specifies the namespace of the SR-IOV Network Operator instance. The default value is **openshift-sriov-network-operator**. Do not set a different value.

Field	Type	Description
spec.configDaemonNodeSelector	**string**	Specifies the node selection to control scheduling the SR-IOV Network Config Daemon on selected nodes. By default, this field is not set and the Operator deploys the SR-IOV Network Config daemon set on worker nodes.
spec.disableDrain	**boolean**	Specifies whether to disable the node draining process or enable the node draining process when you apply a new policy to configure the NIC on a node. Setting this field to **true** facilitates software development and installing OpenShift Container Platform on a single node. By default, this field is not set. For single-node clusters, set this field to **true** after installing the Operator. This field must remain set to **true**.
spec.enableInjector	**boolean**	Specifies whether to enable or disable the Network Resources Injector daemon set.
spec.enableOperatorWebhook	**boolean**	Specifies whether to enable or disable the Operator Admission Controller webhook daemon set.
spec.logLevel	**integer**	Specifies the log verbosity level of the Operator. By default, this field is set to **0**, which shows only basic logs. Set to **2** to show all the available logs.
spec.featureGates	**map[string]bool**	Specifies whether to enable or disable the optional features. For example, **metricsExporter**.
spec.featureGates.metricsExporter	**boolean**	Specifies whether to enable or disable the SR-IOV Network Operator metrics. By default, this field is set to **false**.

Field	Type	Description
spec.featureGates.mellanoxFirmwareReset	**boolean**	Specifies whether to reset the firmware on virtual function (VF) changes in the SR-IOV Network Operator. Some chipsets, such as the Intel C740 Series, do not completely power off the PCI-E devices, which is required to configure VFs on NVIDIA/Mellanox NICs. By default, this field is set to **false**. IMPORTANT The **spec.featureGates.mellanoxFirmwareReset** parameter is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope

4.10.2.1.2. About the Network Resources Injector

The Network Resources Injector is a Kubernetes Dynamic Admission Controller application. It provides the following capabilities:

- Mutation of resource requests and limits in a pod specification to add an SR-IOV resource name according to an SR-IOV network attachment definition annotation.
- Mutation of a pod specification with a Downward API volume to expose pod annotations, labels, and huge pages requests and limits. Containers that run in the pod can access the exposed information as files under the **/etc/podnetinfo** path.

The Network Resources Injector is enabled by the SR-IOV Network Operator when the **enableInjector** is set to **true** in the **SriovOperatorConfig** CR. The **network-resources-injector** pod runs as a daemon set on all control plane nodes. The following is an example of Network Resources Injector pods running in a cluster with three control plane nodes:

```
$ oc get pods -n openshift-sriov-network-operator
```

Example output

```
NAME                                READY   STATUS    RESTARTS   AGE
network-resources-injector-5cz5p        1/1     Running   0          10m
network-resources-injector-dwqpx        1/1     Running   0          10m
network-resources-injector-lktz5        1/1     Running   0          10m
```

4.10.2.1.3. Disabling or enabling the Network Resources Injector

To disable or enable the Network Resources Injector, complete the following procedure.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.
- You must have installed the SR-IOV Network Operator.

Procedure

- Set the **enableInjector** field. Replace **<value>** with **false** to disable the feature or **true** to enable the feature.

```
$ oc patch sriovoperatorconfig default \
  --type=merge -n openshift-sriov-network-operator \
  --patch '{ "spec": { "enableInjector": <value> } }'
```

TIP

You can alternatively apply the following YAML to update the Operator:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default
  namespace: openshift-sriov-network-operator
spec:
  enableInjector: <value>
```

4.10.2.1.4. About the SR-IOV Network Operator admission controller webhook

The SR-IOV Network Operator Admission Controller webhook is a Kubernetes Dynamic Admission Controller application. It provides the following capabilities:

- Validation of the **SriovNetworkNodePolicy** CR when it is created or updated.
- Mutation of the **SriovNetworkNodePolicy** CR by setting the default value for the **priority** and **deviceType** fields when the CR is created or updated.

The SR-IOV Network Operator Admission Controller webhook is enabled by the Operator when the **enableOperatorWebhook** is set to **true** in the **SriovOperatorConfig** CR. The **operator-webhook** pod runs as a daemon set on all control plane nodes.

NOTE

Use caution when disabling the SR-IOV Network Operator Admission Controller webhook. You can disable the webhook under specific circumstances, such as troubleshooting, or if you want to use unsupported devices. For information about configuring unsupported devices, see Configuring the SR-IOV Network Operator to use an unsupported NIC.

The following is an example of the Operator Admission Controller webhook pods running in a cluster with three control plane nodes:

```
$ oc get pods -n openshift-sriov-network-operator
```

Example output

```
NAME                              READY   STATUS    RESTARTS   AGE
operator-webhook-9jkw6            1/1     Running   0          16m
operator-webhook-kbr5p            1/1     Running   0          16m
operator-webhook-rpfrl            1/1     Running   0          16m
```

4.10.2.1.5. Disabling or enabling the SR-IOV Network Operator admission controller webhook

To disable or enable the admission controller webhook, complete the following procedure.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.
- You must have installed the SR-IOV Network Operator.

Procedure

- Set the **enableOperatorWebhook** field. Replace **<value>** with **false** to disable the feature or **true** to enable it:

```
$ oc patch sriovoperatorconfig default --type=merge \
  -n openshift-sriov-network-operator \
  --patch '{ "spec": { "enableOperatorWebhook": <value> } }'
```

TIP

You can alternatively apply the following YAML to update the Operator:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default
  namespace: openshift-sriov-network-operator
spec:
  enableOperatorWebhook: <value>
```

4.10.2.1.6. About custom node selectors

The SR-IOV Network Config daemon discovers and configures the SR-IOV network devices on cluster nodes. By default, it is deployed to all the **worker** nodes in the cluster. You can use node labels to specify on which nodes the SR-IOV Network Config daemon runs.

4.10.2.1.7. Configuring a custom NodeSelector for the SR-IOV Network Config daemon

The SR-IOV Network Config daemon discovers and configures the SR-IOV network devices on cluster nodes. By default, it is deployed to all the **worker** nodes in the cluster. You can use node labels to specify on which nodes the SR-IOV Network Config daemon runs.

To specify the nodes where the SR-IOV Network Config daemon is deployed, complete the following procedure.

IMPORTANT

When you update the **configDaemonNodeSelector** field, the SR-IOV Network Config daemon is recreated on each selected node. While the daemon is recreated, cluster users are unable to apply any new SR-IOV Network node policy or create new SR-IOV pods.

Procedure

- To update the node selector for the operator, enter the following command:

```
$ oc patch sriovoperatorconfig default --type=json \
  -n openshift-sriov-network-operator \
  --patch '[{
      "op": "replace",
      "path": "/spec/configDaemonNodeSelector",
      "value": {<node_label>}
    }]'
```

Replace **<node_label>** with a label to apply as in the following example: **"node-role.kubernetes.io/worker": ""**.

TIP

You can alternatively apply the following YAML to update the Operator:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default
  namespace: openshift-sriov-network-operator
spec:
  configDaemonNodeSelector:
    <node_label>
```

4.10.2.1.8. Configuring the SR-IOV Network Operator for single node installations

By default, the SR-IOV Network Operator drains workloads from a node before every policy change. The Operator performs this action to ensure that there no workloads using the virtual functions before the reconfiguration.

For installations on a single node, there are no other nodes to receive the workloads. As a result, the Operator must be configured not to drain the workloads from the single node.

IMPORTANT

After performing the following procedure to disable draining workloads, you must remove any workload that uses an SR-IOV network interface before you change any SR-IOV network node policy.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.
- You must have installed the SR-IOV Network Operator.

Procedure

- To set the **disableDrain** field to **true** and the **configDaemonNodeSelector** field to **node-role.kubernetes.io/master: ""**, enter the following command:

```
$ oc patch sriovoperatorconfig default --type=merge -n openshift-sriov-network-operator --
patch '{ "spec": { "disableDrain": true, "configDaemonNodeSelector": { "node-
role.kubernetes.io/master": "" } } }'
```

TIP

You can alternatively apply the following YAML to update the Operator:

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
  name: default
  namespace: openshift-sriov-network-operator
spec:
  disableDrain: true
  configDaemonNodeSelector:
   node-role.kubernetes.io/master: ""
```

4.10.2.1.9. Deploying the SR-IOV Operator for hosted control planes

After you configure and deploy your hosting service cluster, you can create a subscription to the SR-IOV Operator on a hosted cluster. The SR-IOV pod runs on worker machines rather than the control plane.

Prerequisites

You must configure and deploy the hosted cluster on AWS.

Procedure

1. Create a namespace and an Operator group:

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-sriov-network-operator
```

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: sriov-network-operators
  namespace: openshift-sriov-network-operator
spec:
  targetNamespaces:
  - openshift-sriov-network-operator
```

2. Create a subscription to the SR-IOV Operator:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: sriov-network-operator-subsription
  namespace: openshift-sriov-network-operator
spec:
  channel: stable
  name: sriov-network-operator
  config:
    nodeSelector:
      node-role.kubernetes.io/worker: ""
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Verification

1. To verify that the SR-IOV Operator is ready, run the following command and view the resulting output:

```
$ oc get csv -n openshift-sriov-network-operator
```

Example output

```
NAME                                         DISPLAY                  VERSION              REPLACES
PHASE
sriov-network-operator.4.18.0-202211021237   SR-IOV Network Operator   4.18.0-
202211021237   sriov-network-operator.4.18.0-202210290517   Succeeded
```

2. To verify that the SR-IOV pods are deployed, run the following command:

```
$ oc get pods -n openshift-sriov-network-operator
```

4.10.2.2. About the SR-IOV network metrics exporter

The Single Root I/O Virtualization (SR-IOV) network metrics exporter reads the metrics for SR-IOV virtual functions (VFs) and exposes these VF metrics in Prometheus format. When the SR-IOV network metrics exporter is enabled, you can query the SR-IOV VF metrics by using the OpenShift Container Platform web console to monitor the networking activity of the SR-IOV pods.

When you query the SR-IOV VF metrics by using the web console, the SR-IOV network metrics exporter fetches and returns the VF network statistics along with the name and namespace of the pod that the VF is attached to.

The SR-IOV VF metrics that the metrics exporter reads and exposes in Prometheus format are described in the following table:

Table 4.17. SR-IOV VF metrics

Metric	Description	Example PromQL query to examine the VF metric
sriov_vf_rx_bytes	Received bytes per virtual function.	**sriov_vf_rx_bytes * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_tx_bytes	Transmitted bytes per virtual function.	**sriov_vf_tx_bytes * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_rx_packets	Received packets per virtual function.	**sriov_vf_rx_packets * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_tx_packets	Transmitted packets per virtual function.	**sriov_vf_tx_packets * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_rx_dropped	Dropped packets upon receipt per virtual function.	**sriov_vf_rx_dropped * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_tx_dropped	Dropped packets during transmission per virtual function.	**sriov_vf_tx_dropped * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_vf_rx_multicast	Received multicast packets per virtual function.	**sriov_vf_rx_multicast * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**

Metric	Description	Example PromQL query to examine the VF metric
sriov_vf_rx_broadcast	Received broadcast packets per virtual function.	**sriov_vf_rx_broadcast * on (pciAddr,node) group_left(pod,namespace,dev_type) sriov_kubepoddevice**
sriov_kubepoddevice	Virtual functions linked to active pods.	-

You can also combine these queries with the kube-state-metrics to get more information about the SR-IOV pods. For example, you can use the following query to get the VF network statistics along with the application name from the standard Kubernetes pod label:

```
(sriov_vf_tx_packets * on (pciAddr,node)  group_left(pod,namespace)  sriov_kubepoddevice) * on
(pod,namespace) group_left (label_app_kubernetes_io_name) kube_pod_labels
```

4.10.2.2.1. Enabling the SR-IOV network metrics exporter

The Single Root I/O Virtualization (SR-IOV) network metrics exporter is disabled by default. To enable the metrics exporter, you must set the **spec.featureGates.metricsExporter** field to **true**.

IMPORTANT

When the metrics exporter is enabled, the SR-IOV Network Operator deploys the metrics exporter only on nodes with SR-IOV capabilities.

Prerequisites

- You have installed the OpenShift CLI (**oc**).
- You have logged in as a user with **cluster-admin** privileges.
- You have installed the SR-IOV Network Operator.

Procedure

1. Enable cluster monitoring by running the following command:

   ```
   $ oc label ns/openshift-sriov-network-operator openshift.io/cluster-monitoring=true
   ```

   To enable cluster monitoring, you must add the **openshift.io/cluster-monitoring=true** label in the namespace where you have installed the SR-IOV Network Operator.

2. Set the **spec.featureGates.metricsExporter** field to **true** by running the following command:

   ```
   $ oc patch -n openshift-sriov-network-operator sriovoperatorconfig/default \
     --type='merge' -p='{"spec": {"featureGates": {"metricsExporter": true}}}'
   ```

Verification

1. Check that the SR-IOV network metrics exporter is enabled by running the following command:

```
$ oc get pods -n openshift-sriov-network-operator
```

Example output

```
NAME                                  READY   STATUS    RESTARTS   AGE
operator-webhook-hzfg4                1/1     Running   0          5d22h
sriov-network-config-daemon-tr54m     1/1     Running   0          5d22h
sriov-network-metrics-exporter-z5d7t  1/1     Running   0          10s
sriov-network-operator-cc6fd88bc-9bsmt 1/1    Running   0          5d22h
```

The **sriov-network-metrics-exporter** pod must be in the **READY** state.

2. Optional: Examine the SR-IOV virtual function (VF) metrics by using the OpenShift Container Platform web console. For more information, see "Querying metrics".

Additional resources

- Querying metrics for all projects with the monitoring dashboard
- Querying metrics for user-defined projects as a developer

4.10.2.3. Next steps

- Configuring an SR-IOV network device
- Optional: Uninstalling the SR-IOV Network Operator

4.10.3. Uninstalling the SR-IOV Network Operator

To uninstall the SR-IOV Network Operator, you must delete any running SR-IOV workloads, uninstall the Operator, and delete the webhooks that the Operator used.

4.10.3.1. Uninstalling the SR-IOV Network Operator

As a cluster administrator, you can uninstall the SR-IOV Network Operator.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have the SR-IOV Network Operator installed.

Procedure

1. Delete all SR-IOV custom resources (CRs):

```
$ oc delete sriovnetwork -n openshift-sriov-network-operator --all
$ oc delete sriovnetworknodepolicy -n openshift-sriov-network-operator --all
```

```
$ oc delete sriovibnetwork -n openshift-sriov-network-operator --all
```

```
$ oc delete sriovoperatorconfigs -n openshift-sriov-network-operator --all
```

2. Follow the instructions in the "Deleting Operators from a cluster" section to remove the SR-IOV Network Operator from your cluster.

3. Delete the SR-IOV custom resource definitions that remain in the cluster after the SR-IOV Network Operator is uninstalled:

```
$ oc delete crd sriovibnetworks.sriovnetwork.openshift.io
```

```
$ oc delete crd sriovnetworknodepolicies.sriovnetwork.openshift.io
```

```
$ oc delete crd sriovnetworknodestates.sriovnetwork.openshift.io
```

```
$ oc delete crd sriovnetworkpoolconfigs.sriovnetwork.openshift.io
```

```
$ oc delete crd sriovnetworks.sriovnetwork.openshift.io
```

```
$ oc delete crd sriovoperatorconfigs.sriovnetwork.openshift.io
```

4. Delete the SR-IOV Network Operator namespace:

```
$ oc delete namespace openshift-sriov-network-operator
```

Additional resources

- Deleting Operators from a cluster