TECHNOLOGY GUIDE

Intel Corporation

SR-IOV Spotlight: Networking and Observability with Kubernetes* Technology Guide

Authors

Killian Muldoon

Dave Cremins

1 Introduction

SR-IOV technology has been optimized in support of Intel® network interface cards (NICs), including the 700 and 800 series. These NICs have several built-in features that enable advanced use cases, improve manageability, and enhance observability of the hardware in a complex system.

This document shows how Kubernetes* orchestration can harness two of these features, SR-IOV virtual local area network (VLAN) trunking and SR-IOV virtual function (VF) telemetry, to power new kinds of networking workloads. It also describes how hardware-level metrics can be used as actionable information, such as about the current state of an application.

SR-IOV VLAN trunking is available in Intel NICs 700 series as a feature preview. SR-IOV virtual function telemetry is available in Intel NICs from the 700 and 800 series.

The features outlined in this guide help to enable cloud-native network functions (CNFs) - network functions that are designed and implemented in line with cloud-native architectural principles. For more on cloud native design in telecommunications see the Cloud Native Thinking for Telecommunications white paper by the Cloud Native Computing Foundation (CNCF). This guide can be helpful for developers and cluster operators who work in Kubernetes implementations of advanced high-performance networking, particularly those that deploy SR-IOV NICs.

This document is part of the Network Transformation Experience Kit, which is available at https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits.

Table of Contents

Figures

Tables

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
CNCF	Cloud Native Computing Foundation
CNF	Containerized Network Function
CNI	Container Network Interface
HPA	Horizontal Pod Autoscaler
K8s	Kubernetes
NIC	Network Interface Card
VF	Virtual function
VLAN	Virtual Local Area Network

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Cloud Native Thinking for Telecommunications	https://github.com/cncf/telecom-user-group/raw/master/whitepaper/CNCF%20_TUG_Cloud%20Native%20Thinking%20for%20Telecommunications.pdf
sriov-cni VLAN Trunking Feature Preview	https://github.com/k8snetworkplumbingwg/sriov-cni/tree/e507c01e4b3bfd423b043245a9f5c726bb22d8b2
Advanced Networking Features in Kubernetes* and Container Bare Metal Application Note	https://builders.intel.com/docs/networkbuilders/adv-network-features-in-kubernetes-app-note.pdf
sriov-cni [GitHub*]	https://github.com/k8snetworkplumbingwg/sriov-cni
Sriov network metrics exporter	https://github.com/intel/sriov-network-metrics-exporter
Telemetry Aware Scheduling – Automated Workload Optimization with Kubernetes* (K8s*) Technology Guide	https://networkbuilders.intel.com/solutionslibrary/telemetry-aware-scheduling-automated-workload-optimization-with-kubernetes-k8s-technology-guide

2 VLAN Trunking with Intel x700 NICs in Kubernetes

Intel SR-IOV NICs can use VLAN tags to create virtual networks within a single physical network. Setting the VLAN tag on an SR-IOV virtual function makes it part of the virtual network identified by that tag. Traffic that belongs to other virtual networks is not delivered to that virtual function.

VLAN trunking enables the use of multiple VLAN tags on a single virtual function. The workload using that virtual function can then operate as a "trunk" with multiple individual VLAN branches leading into it. This is essential for packet processing workloads that operate as trunk links. It allows traffic to flow through a network backbone regardless of the VLAN tag it is marked with.

For Kubernetes workloads, VLAN trunking is available as a feature preview in the SR-IOV container network interface (CNI). The implementation and documentation are available at sriov-cni VLAN Trunking Feature Preview. For more information on deploying workloads with SR-IOV data plane networks in a Kubernetes cluster see Advanced Networking Features in Kubernetes* and Container Bare Metal.

Note: VLAN trunking is currently only available in Intel 700 series NICs with an OOT i40e driver with a recommend version of 2.7.11 or greater. The SR-IOV changes required to configure VLAN trunking in Kubernetes are only available from a temporary branch as a feature preview.

2.1 Solution Description

VLAN tagging is enabled and managed in Kubernetes secondary SR-IOV interfaces using the Kubernetes SR-IOV system for Intel NICs. The overall system includes:

- **Multus CNI**: Enables secondary interfaces in Kubernetes managed containers
- **SR-IOV Network Device Plugin**: Manages SR-IOV virtual functions as resources for each node in a Kubernetes cluster
- **SR-IOV CNI**: Configures individual SR-IOV virtual functions, customizing them for individual workload needs

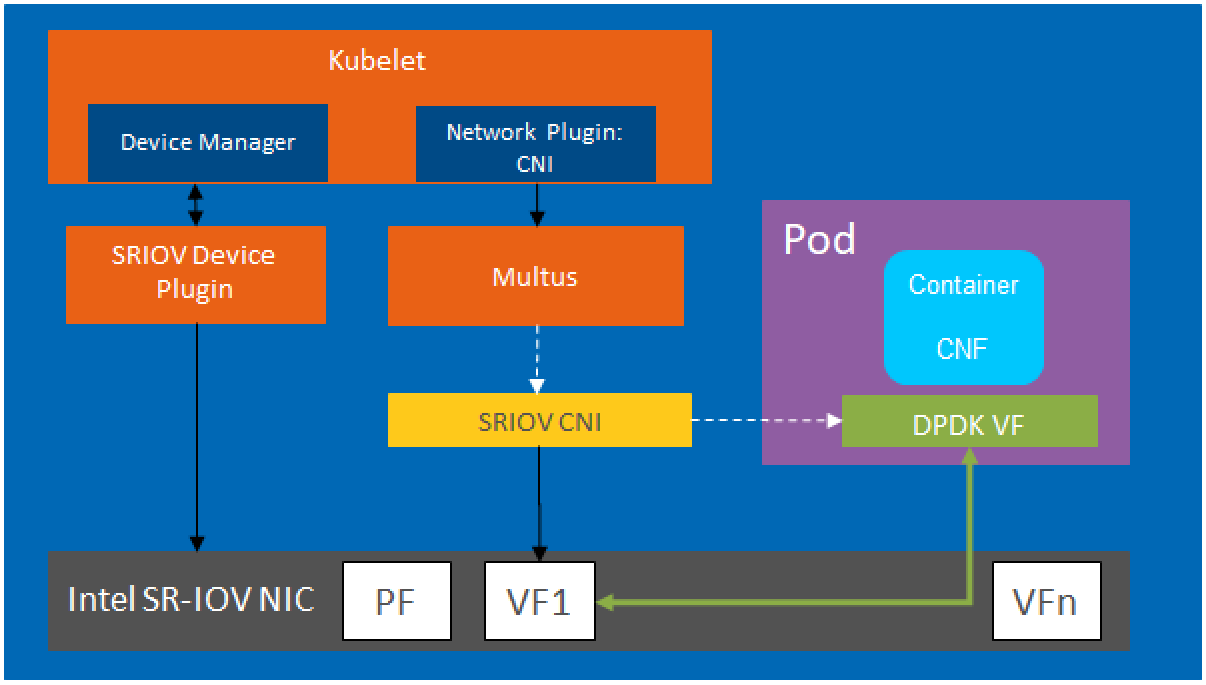

Figure 1. Components that Enable VLAN Tagging

When the above components are used to configure and manage SR-IOV virtual functions in Kubernetes, you can set VLAN tags and VLAN trunking by using built-in capabilities. The SR-IOV CNI configures VLAN trunking for the individual virtual functions on a node. VLAN trunking relies on the feature preview available at sriov-cni VLAN Trunking Feature Preview. When a new workload is created in Kubernetes, it can be configured with a secondary interface. The description of this interface has multiple variables, including base configuration settings for SR-IOV virtual functions. VLAN tags can be inserted into the network attachment definition configuration for each SR-IOV interface created.

For a workload that needs to act as a VLAN trunk, multiple VLAN tags can be added under the vlan_trunk field in the network configuration. This field sets the interface to accept either incoming or outgoing traffic marked with any of the supplied VLAN tags.

2.2 Benefits of Solution

VLAN management, enabled here in the NIC, allows the creation of virtual network partitions on a single physical network. In cloud-native packet processing workloads, this feature is extremely useful as many traffic flows are not forced to flow through a separate switch, but instead can flow between applications deployed on the same platform.

VLAN management, including VLAN trunking, can be enabled by a software switch, e.g., Open vSwitch*. Intel 700 series NICs allow this feature to be offloaded to the NIC. This preserves valuable CPU cores for packet processing. VLAN trunking allows workloads that need to operate on multiple different virtual networks to offload this function in the same way as single VLAN tags are managed.

Cloud-native deployments, particularly in multi-tenant environments, create several uses for virtual network partitions. Some examples include dividing network management traffic, tenant segregation, and isolating sensitive infrastructure. The traffic from multiple virtual networks may still need to traverse the same individual network functions.

VLAN trunking offload in Kubernetes manages network functions across virtual networks, reduces duplication of workloads, provides flexibility of network segregation, and upgrades existing network models to a cloud-native deployment paradigm.

2.3 Example: Network Gateway Function with SR-IOV VLAN Trunking

This example deployment includes two virtual networks defined by VLAN tags 20 and 40. Each of these virtual networks has one workload, A and B respectively in Figure 2, and are attached to a virtual function with the corresponding VLAN tag. In this deployment, however, there is a virtual firewall, marked Workload C, which acts as a gateway for traffic to both workloads.

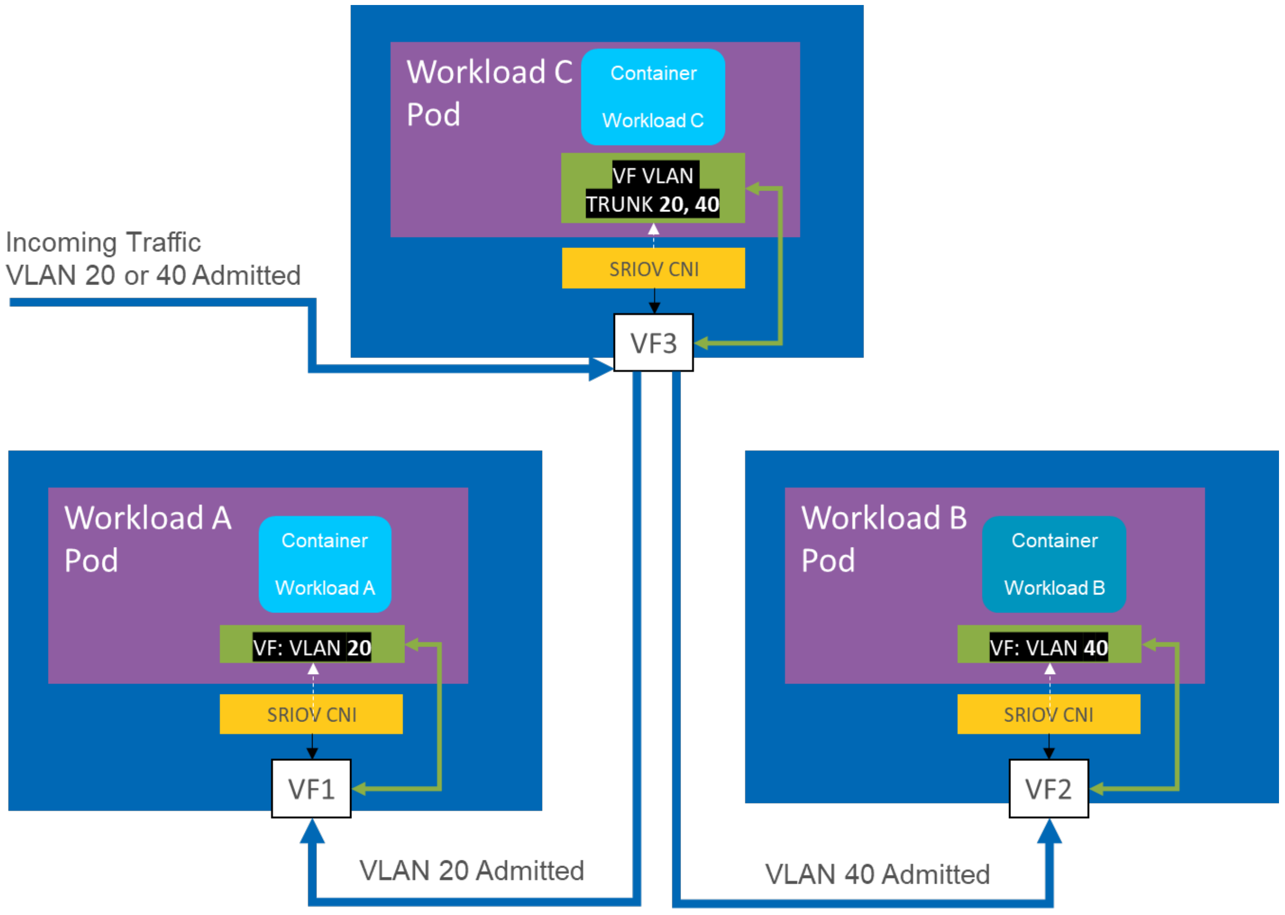

Figure 2. Network Gateway Function with SR-IOV VLAN Trunking

Figure 2 shows the problem solved with enablement of VLAN tagging for SR-IOV virtual functions in Kubernetes. The above deployment is agnostic to the physical deployment of the pods. They could be on a single machine, or spread across multiple machines, but network partitioning prevents a workload from seeing traffic not intended for it.

In Kubernetes, the above deployment would ordinarily require the creation of three distinct network attachment definitions. The first two, one each for A and B, would include their network parameters and a single VLAN classification. The third, used to configure the network for workload C, would define a VLAN trunk with VLAN tags 20 and 40.

VLAN trunking allows a single workload, the nominal firewall, to perform its function across a specified list of network functions. Sorting which traffic is allowed is done at the VF interface, which saves CPU cycles that would otherwise be needed to partition traffic.

VLAN trunking managed by SR-IOV can be enabled and used in use cases where traffic from multiple virtual networks needs to travel through the same network function. In addition to gateways and firewalls, trunking can be used to configure and deploy routers, load balancers, loggers, traffic analysis tools, and many other CNFs. The SR-IOV components listed above allow this to be managed and implemented in a cloud native manner using Kubernetes best practices.

3 Observability with SR-IOV Metrics Exporter in Kubernetes

Observability is a key consideration when designing distributed workloads. Userspace networking, where the ordinary kernel networking stack is bypassed to enable high-performance packet processing, sacrifices observability for performance. This trade-off can be avoided by using hardware level counters.

Intel 700 and 800 series NICs expose hardware metrics for each SR-IOV virtual function using the SR-IOV Network Metrics Exporter. These metrics can be used to check which workloads are under load, indicate the health of an interface, and compare current performance to performance expected under a benchmark or service level objective.

Note: The SR-IOV Network Metrics Exporter requires the i40e driver version to be 2.11 or higher. It depends on a Linux* kernel of version 4.4. or above.

3.1 Solution Description

The SR-IOV Network Metrics Exporter makes hardware counters available to a metrics database such as Prometheus, which is a time series database sponsored by the Cloud Native Computing Foundation (CNCF). This allows network metrics to be collected at the host rather than the application level. Instead of each application reporting its own interface metrics in a tailored format, all virtual functions expose metrics in a vendor-neutral standard that is controlled by the platform operator rather than the application developer.

The SR-IOV Network Metrics Exporter not only reads metrics exposed by the NIC driver, it also maps what devices are allocated to which workloads. That is, it exports which containers and pods in a Kubernetes cluster are using which specific virtual functions.

This mapping allows specific metrics, such as how many packets are being received by an interface, to be presented in terms of the application using the virtual function. Combined with queries at the database layer, this makes clear, unambiguous information about the current performance of workloads available using hardware counters.

Telemetry is enabled and managed in Kubernetes secondary SR-IOV interfaces using the following components:

- **Multus CNI**: Enables secondary interfaces in Kubernetes managed containers
- **SR-IOV Network Device Plugin**: Manages SR-IOV virtual functions as resources for each node in a Kubernetes cluster
- **SR-IOV CNI**: Configures individual SR-IOV virtual functions, customizing them for individual workload needs
- **SR-IOV Network Metrics Exporter**: Exports hardware counters and Kubernetes allocation of SR-IOV virtual functions
- **Prometheus**: Collects, collates, and queries time-series telemetry

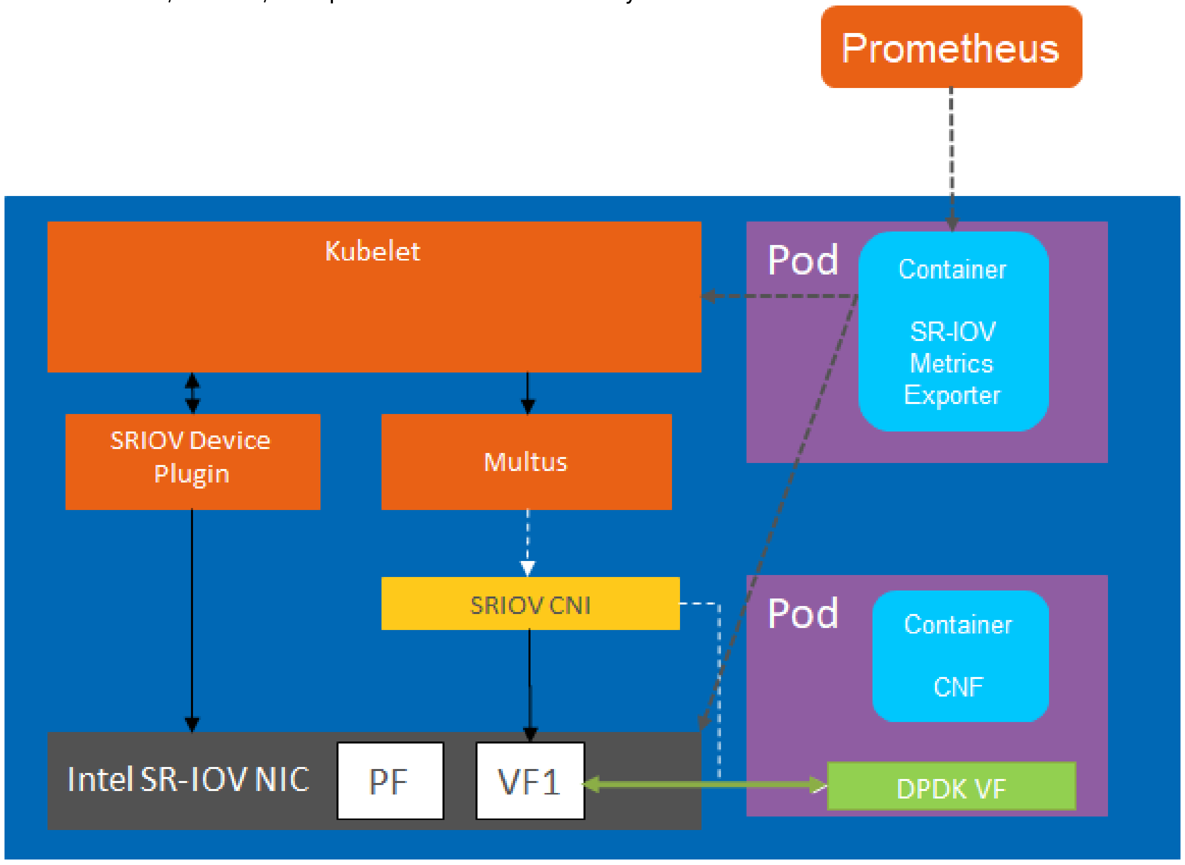

Figure 3. Components that Enable SR-IOV Telemetry Collection

The SR-IOV Network Metrics Exporter is a drop-in application in clusters already configured to use the above networking components to enable secondary networking in Kubernetes. It can be deployed as a DaemonSet inside a Kubernetes cluster.

After the exporter is up and running on the correct compute nodes, it collects information about existing SR-IOV virtual functions and any pods currently utilizing them. This information is made available at a network endpoint for collection to a metrics database – Prometheus in the above example. After metrics are available, querying can be used in order to analyze, visualize, or for automated actions based on the available metrics.

3.2 Benefits of Solution

The hardware counters that underpin the SR-IOV Network Metrics Exporter allow for application-agnostic metrics to be collected. Tailored metrics collection running in the same container as a workload can be replaced with a single independent metrics exporter for interface metrics.

Pushing metrics collection responsibilities to the hardware saves CPU cycles and moving network interface telemetry from inside the application to an independent exporter simplifies the deployment model.

The SR-IOV Network Metrics Exporter is built with Kubernetes and cloud native deployments in mind. It supports exporting virtual function metrics as an application-level indicator where appropriate.

Simpler deployments with more clearly defined responsibilities combined with offloading metrics collection to the NIC hardware empowers a cloud native approach to monitoring packet processing workloads in Kubernetes.

The metrics as exported can be fed to data analysis and machine learning applications to improve deployment reliability and performance. These metrics can also be used to drive autoscaling decisions through the Kubernetes Horizontal Pod Autoscaler (HPA) or workload placement decisions using Telemetry Aware Scheduling.

3.3 Example: Using SR-IOV Hardware Metrics to Scale Workloads

In the representative example deployment outlined below, workload A is a CNF processing network traffic. The SR-IOV Network Metrics Exporter runs on each node in this deployment, collecting raw metrics for each virtual function in use and matching running workloads with the interfaces attached to them through the Prometheus time series database. Prometheus can be installed inside the Kubernetes cluster as a pod, or entirely outside the cluster as part of an external monitoring solution.

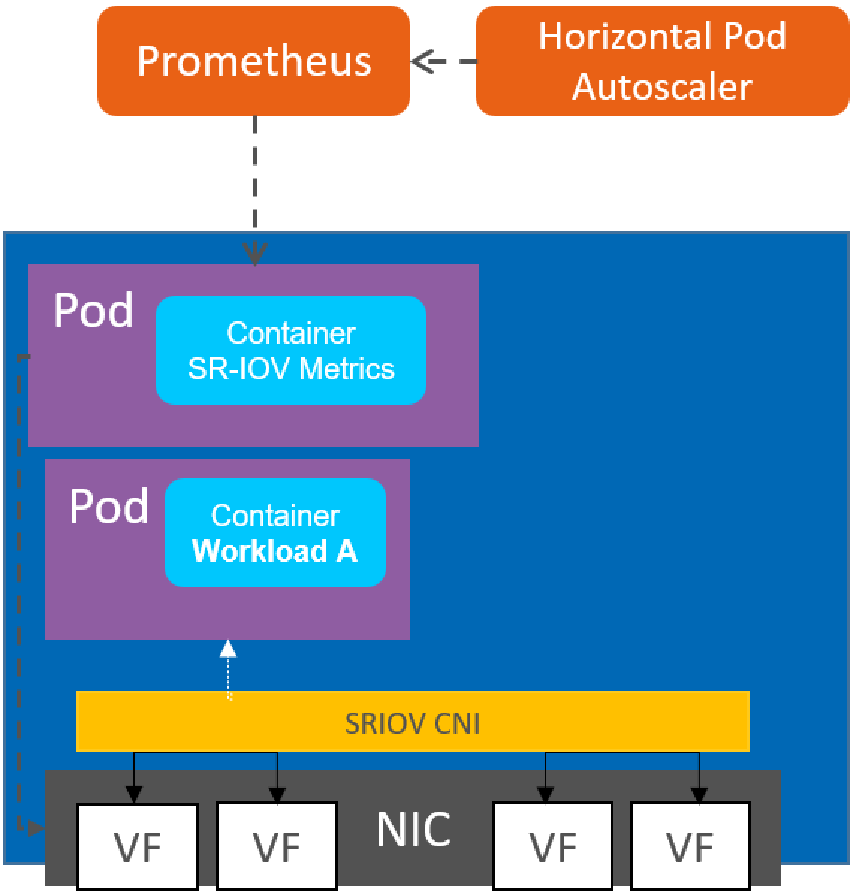

Figure 4. Workload A Does Not Exceed Traffic Capacity

As the workload has two virtual functions, one for incoming and a second for outgoing traffic, each has two sets of metrics exported by the SR-IOV Network Metrics Exporter. Traffic is monitored as it passes through each interface on the receive side and compared to the maximum traffic allocated to the interface producing a capacity metric.

This metric, titled pod_vf_rx_capacity and expressed as an integer percentage value, allows detection of a workload approaching overload.

The Kubernetes Horizontal Pod Autoscaler (HPA) is used to set an autoscaling policy to ensure that a single copy of this workload never exceeds capacity to the extent that it would drop traffic. As a rule, this workload is scaled up if traffic reaches more than 80 percent of the bandwidth available on the SR-IOV virtual function. HPA is a core part of a Kubernetes cluster and runs as part of the Kubernetes control plane.

Using the SR-IOV Network Metrics Exporter, the core part of metrics collection is offloaded to the NIC. The exporter makes the data available to Prometheus at regular intervals. A database query is used to express pod_vf_rx_capacity, the key performance indicator for this workload.

When this metric goes over 80 percent, the Kubernetes HPA increases the desired replica count, and the cluster resolves this by creating a new instance of workload A, called A2.

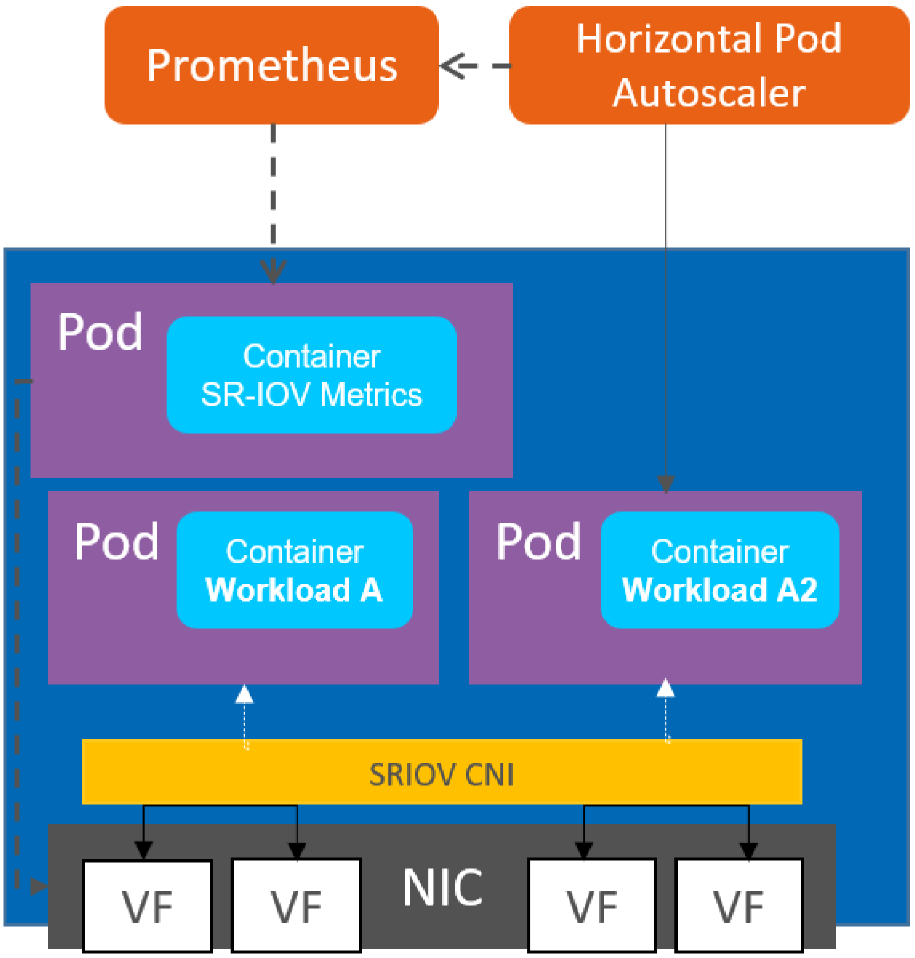

Figure 5. Workload A2 Is Created After Capacity of Workload A Exceeded

The above solution shows scaling in a single node cluster for simplicity. The same solution would be appropriate for large Kubernetes clusters. The Kubernetes HPA is just one of many use cases for the hardware SR-IOV metrics exposed by the SR-IOV Network Metrics Exporter.

4 Summary

SR-IOV VLAN trunking and SR-IOV Network Metrics Exporter enable hardware offload of key features to Intel NICs in an environment orchestrated by Kubernetes.

Each of these features is an example of the power of hardware offload in removing responsibilities from the CPU cores assigned to a cloud native network function. This allows greater flexibility in the design and deployment of network functions, while maintaining the same features for advanced workload management and observability.

The SR-IOV CNI, which enables VLAN trunking in Kubernetes, is an essential part of any networking deployment in Kubernetes that is powered by SR-IOV. Users interested in offloading metrics collection for cloud native automation and observability are encouraged to run SR-IOV Network Metrics Exporter.

0121/DN/WIPRO/PDF 634043-001US