Build a Demo of Integration of RHOCP and Apstra using RH Ansible Event-Driven Automation

Guy Davies

2025-02-11

Contents

List of Figures

Build a Demo of Integration of RHOCP and Apstra using RH Ansible Event-Driven Automation

Introduction

By default, when a network operator deploys workloads in Kubernetes, the connectivity between Pods is a single, Layer 2 domain provided by the primary Container Network Interface (CNI) plugin within the Kubernetes cluster. While this model works (relatively) well for most cloud-native applications, it is not sufficient for many enterprise applications that require more complex network topologies. In particular, some workloads require ultra-high performance, which is typically implemented using Single Root I/O Virtualization (SR-IOV).

SR-IOV typically exposes a particular VLAN to a particular Virtual Machine (VM) or Pod as a virtual Ethernet interface (with or without VLAN tag). The VLAN is then exposed directly to the fabric where it must be correctly configured to ensure that the VLAN is correctly bound into the associated bridge-domain or VRF. This has traditionally been done manually by the network operator, often by a different team than was responsible for deploying the VM or Pod. This leads to delays, misconfigurations, and outages.

The Apstra AOS platform can be used to automate the configuration of the network fabric to ensure that the VLANs are correctly bound to the correct EVPN bridge-domains or VRFs. Juniper Networks has developed a mechanism using Red Hat's Ansible Event-Driven Automation (EDA) to automatically respond to events in the Kubernetes cluster as Projects, Virtual Networks, Pods, and Virtual Machines are created, modified, or deleted. This mechanism can be used to automatically configure the network fabric to ensure that the VLANs are correctly bound to the correct EVPN bridge-domains or VRFs.

This document describes how to build a demo of the integration of the Apstra EDA module with Red Hat OpenShift Container Platform (RHOCP) using Red Hat Ansible Automation. The demo will show how the Apstra EDA module can be used to automatically configure the network fabric to ensure that the VLANs are correctly bound to the correct EVPN bridge-domains or VRFs as Projects, Virtual Networks, Pods, and Virtual

Machines are created, modified, or deleted in RHOCP.

Demo Environment

To build the demo, you will need the following:

- A minimum of 3 hosts (can be VMs) for the RHOCP cluster controllers
- A minimum of 2 hosts (must be BMS) for the RHOCP cluster compute
  - Each host must have at least one SR-IOV-capable NIC providing a minimum of two physical ports. E.g. Intel E810 or Intel XXV710
  - Each host must also have at least one other NIC for the management network and for the default Kubernetes network (primary CNI)
- One host (with connectivity outside of the cluster to the management network of the QFX switches) on which to run the Apstra AOS server VM and the Ansible Automation Platform VM
- At least two Juniper QFX leaf switches (QFX5120 or QFX5130)
- At least one Juniper QFX spine switch (QFX5210 or QFX5220)

Deploy the IP Fabric using Apstra with a simple 3-stage Clos topology with the QFX switches as the leaf and spine switches. A default blueprint should be fine with one spine and two leaves.

Attach the RHOCP compute hosts SR-IOV-capable NICs to the QFX leaf switches. Be consistent in the NICs used for the SR-IOV interfaces across the compute hosts. E.g. use ens2f0 for the first SR-IOV interface on all compute hosts (for the SR-IOV for Pods), ens2f1 for the second SR-IOV interface (for the SR-IOV for VMs), etc. Attach the management and default network NICs to the QFX leaf switches as well.

In Apstra, configure the different VRFs and bridge-domains for management and the default network. Do not configure anything for the SR-IOV networks. The Apstra EDA module will automatically configure the SR-IOV networks as part of the demo.

- Deploy RHOCP on the available hosts

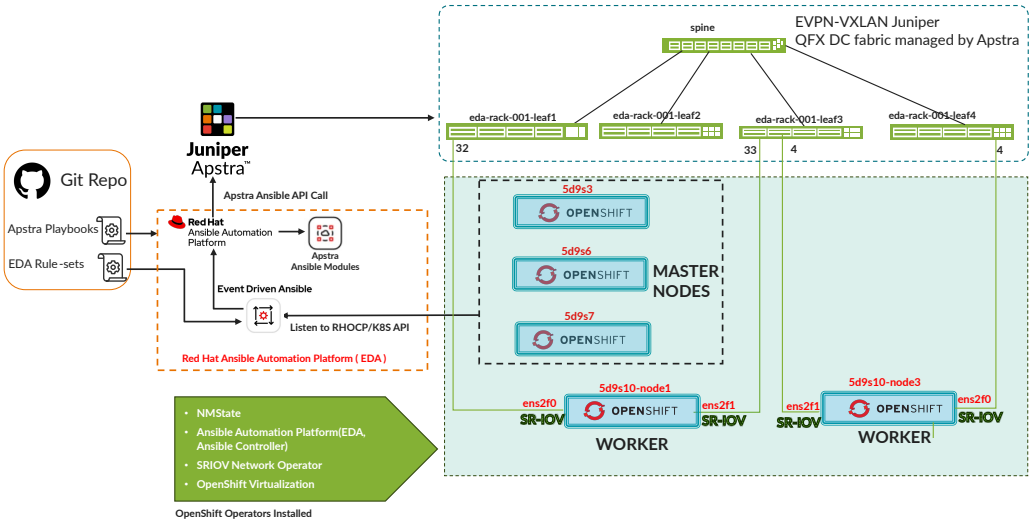

Figure 1: High Level Overview of an Example Demo Layout

- Deploy Red Hat Ansible Automation Platform operator onto the RHOCP Cluster. This includes Red Hat Event Driven Ansible.
- Deploy the Apstra EDA package onto the RHOCP cluster & RHAAP EDA. These can be pulled from Juniper's Github repository

Demo Steps

NOTE: All the excerpts shown in this README are available in the /Artifacts folder of this repository

The following demo builds the setup as shown in the diagram below.

Step 1: Create SRIOV Network Node Policies

In order to correctly differentiate between SR-IOV NICs used for Pods and those to be used for VMs, two `SriovNetworkNodePolicy` objects must be created. The in the first example manifest, the `deviceType` is `netdevice`, which is appropriate for Pods in Openshift/Kubernetes.

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
```

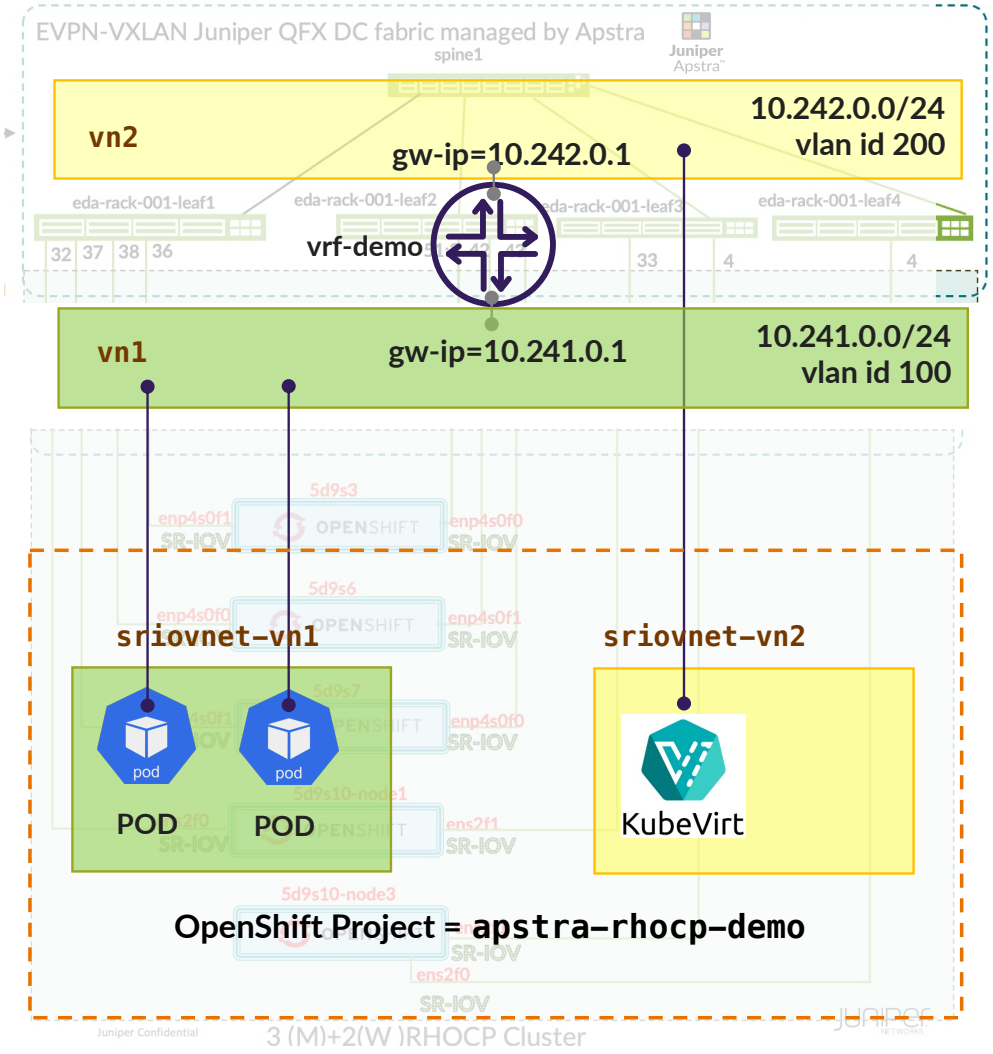

Figure 2: Demo Setup

```
  name: ens2f0-netdev-policy
  namespace: openshift-sriov-network-operator
spec:
  deviceType: netdevice
  isRdma: false
  needVhostNet: true
  nicSelector:
    pfNames:
      - ens2f0
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: 'true'
  numVfs: 8
  priority: 99
  resourceName: pod_vfs_ens2f0
```

In the second example manifest, the `deviceType` is `vfio-pci`, which is required for Openshift Virtualization (Kubevirt).

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: ens2f1-vfio-pci-policy
  namespace: openshift-sriov-network-operator
spec:
  deviceType: vfio-pci
  isRdma: false
  nicSelector:
    pfNames:
      - ens2f1
    rootDevices:
      - '0000:81:00.1'
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: 'true'
  numVfs: 8
  priority: 99
  resourceName: vm_vfs_ens2f1
```

Each set of SR-IOV resources should be assigned a `resourceName`, which identifies the type of workload that can use them. This label is used when

requesting SR-IOV interfaces to be attached to a workload to identify the type of workload being instantiated.

Step 2: Create a Project in RHOCP

A Project in RHOCP corresponds to a VRF in the network fabric. When a Project is created, the Apstra EDA module will automatically create the corresponding VRF in the network fabric. Within the context of the Project all Virtual Networks are Layer 2 domains, with automatically configured Layer 3 connectivity between those Virtual Networks.

This can be achieved through the Openshift UI or via the CLI as shown below.

```
oc new-project eda-demo-vrf \
    --description="Project to demo Apstra EDA integration with RHOCP" \
    --display-name="eda-demo-vrf"
```

Step 3: Create two Virtual Networks in RHOCP

Once the Project is created, you must create two Virtual Networks, which will consume resources assigned for different SR-IOV pools.

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetwork
metadata:
  name: sriovnet-vn1
  namespace: openshift-sriov-network-operator
  labels:
    type: eda
    pfname: ens2f0
  annotations:
    apstra.juniper.net/vnet: '[
      {
        "vnetName": "vn1",
        "vrfName": "eda-demo-vrf"
      }
    ]'
spec:
```

```
  ipam: |
    {
      "type": "host-local",
      "subnet": "10.241.0.0/24",
      "rangeStart": "10.241.0.101",
      "rangeEnd": "10.241.0.200",
      "gateway": "10.241.0.1",
      "routes": [
        {
          "dst": "10.242.0.0/16",
          "gw": "10.241.0.1"
        }
      ]
    }
  networkNamespace: apstra-rhocp-demo
  resourceName: pod_vfs_ens2f0
  vlan: 100
```

The `resourceName` in the above YAML is required to ensure that vn1 uses the set of SR-IOV interfaces assigned for the deployment of Pods with the correct driver.

```
oc apply -f SRIOVNET_VN1.yaml
```

When the SR-IOV Virtual Network is created in Openshift, the EDA Plugin will trigger an Ansible Playbook running in Ansible Automation Platform, which makes API calls to Apstra requesting a matching virtual-network be created in the IP fabric. A Virtual Network with the name matching `vnetName` (`vn1`) will be created inside the VRF with the name matching `vrfName` (`eda-demo-vrf`). It will have a matching `vlan` (`100`). The VNI will be automatically selected by Apstra from a pool of available VNIs.

This process can be seen in the Ansible Automation Platform UI, where the create_vnet.yaml Playbook will be triggered. In the Apstra UI, the new VN will appear but will have no active Connectivity Templates and no associated interfaces.

A second SR-IOV Virtual Network should be created to which VMs will attach, as shown below.

```
apiVersion: sriovnetwork.openshift.io/v1
```

```
kind: SriovNetwork
metadata:
  name: sriovnet-vn2
  namespace: openshift-sriov-network-operator
  labels:
    type: eda
    pfname: ens2f1
  annotations:
    apstra.juniper.net/vnet: '[
      {
        "vnetName": "vn2",
        "vrfName": "eda-demo-vrf"
      }
    ]'
spec:
  ipam: |
    {
      "type": "host-local",
      "subnet": "10.242.0.0/24",
      "rangeStart": "10.242.0.101",
      "rangeEnd": "10.242.0.200",
      "gateway": "10.242.0.1",
      "routes": [
        {
          "dst": "10.241.0.0/16",
          "gw": "10.242.0.1"
        }
      ]
    }
  networkNamespace: apstra-rhocp-demo
  resourceName: vm_vfs_ens2f1
  vlan: 200
```

The `resourceName` in the above YAML is required to ensure that vn1 uses the set of SR-IOV interfaces assigned for the deployment of VMs with the correct driver.

```
oc apply -f SRIOVNET-VN2.yaml
```

The same process can be observed in the Ansible Automation Platform UI and in the Apstra UI, resulting in a new VN in Apstra with no attached resources.

Step 4: Create a Pod in RHOCP

Once the SR-IOV Virtual Networks have been built, workloads can be attached. For the purposes of the demo, you can create a single instance of a simple Pod using the manifest below.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vn1-deployment
  namespace: apstra-rhocp-demo
  labels:
    type: eda
    vnet: vn1
spec:
  replicas: 1
  selector:
    matchLabels:
      type: eda
      vnet: vn1
  template:
    metadata:
      name: vn1-deployment
      labels:
        type: eda
        vnet: vn1
      annotations:
        apstra.juniper.net/ep: '[
          {
            "vnetName": "vn1"
          }
        ]'
        k8s.v1.cni.cncf.io/networks: '[
          {
```

```
          "name": "sriovnet-vn1",
          "namespace": "apstra-rhocp-demo",
          "interface": "ext0"
        }
      ]'
  spec:
    containers:
      - name: iperf3
        image: centos/tools
        command: [ "/bin/bash", "-c", "--" ]
        args: [ "while true; do sleep 300000; done;" ]
        ports:
          - containerPort: 5201
```

You'll note that the annotations under `apstra.juniper.net/ep` identify the `vnetName` (`vn1`) to which the secondary interface should be attached in Apstra. The system will use the data from LLDP to identify which switch and which port should be configured. The VLAN will be taken from the configuration of the virtual network `vn1`.

```
oc apply -f DEPLOYMENT-POD.yaml
```

On issuing the above command, if you look again in the Ansible Automation Platform UI, you will see a `create_pod.yaml` Playbook being triggered. This Pod will be assigned to one of the available compute nodes, and will consume a VF on the NIC associated with `sriovnet-vn1`. This will trigger the creation of a Connection Template, and the configuration of the interface on the appropriate leaf switch, as identified by the LLDP data. Looking at the Apstra UI after a couple of minutes, you will see in the details of the virtual network, that the connected workloads changes from 0 to 1 and from yellow to green.

Step 5: Scale the Pod Deployment in RHOCP

In the Openshift UI, if you then look at the description of the Pod Deployment, you will see the current count at 1. Clicking on the counter in the circle in the top left, you can increase the number from 1 to 2. This will trigger the deployment of a second Pod on a different compute node. This, in turn will trigger another run of the `create_pod.yaml` Playbook in

Ansible Automation Platform, which in turn will trigger the same process in Apstra resulting in the count of attached nodes for `vn1` increasing from 1 to 2. Looking at the details, you will see that now two interfaces are selected on different switches for the connections of the two Pods.

Step 6: Create a Virtual Machine in RHOCP using Kubevirt

Next, we can deploy a virtual machine in RHOCP using Openshift Virtualization (Kubevirt). The following example manifest would be sufficient.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: kubevirt-vm-vn2
  namespace: apstra-rhocp-demo
  labels:
    type: eda
spec:
  running: true
  template:
    metadata:
      labels:
        kubevirt.io/size: large
        kubevirt.io/domain: kubevirt-vm-vn2
        type: eda
      annotations:
        apstra.juniper.net/ep: '[
          {
            "vnetName": "vn2"
          }
        ]'
        k8s.v1.cni.cncf.io/networks: '[
          {
            "name": "sriovnet-vn2",
            "namespace": "apstra-rhocp-demo"
          }
        ]'
```

```
spec:
  domain:
    cpu:
      cores: 16
    devices:
      disks:
      - disk:
          bus: virtio
        name: rootfs
      - cdrom:
          bus: sata
        name: cloudinitvolume
      interfaces:
      - name: default
        masquerade: {}
      - name: private-net
        sriov: {}
    machine:
      type: q35
    resources:
      requests:
        memory: 16G
  volumes:
    - containerDisk:
        image: s-artifactory.juniper.net/atom-docker/kubevirt-images/ubuntu-22.04-k8
      name: rootfs
    - cloudInitConfigDrive:
        userData: |-
          #cloud-config
          chpasswd:
            list: |
              ubuntu:ubuntu
              root:root
            expire: False
          write_files:
          - encoding: plain
            content: |
              network:
```

```
                    version: 2
                    ethernets:
                      enp1s0:
                        routes:
                        -   to: 0.0.0.0/0
                            via: 10.0.2.1
                            table: 10
                        routing-policy:
                        -   from: 10.0.2.1/24
                            table: 10
                      enp7s0:
                        addresses: [10.242.0.113/24]
                        routes:
                        -   to: 0.0.0.0/0
                            via: 10.242.0.1
                  owner: root:root
                  path: /etc/netplan/source-based-routing.yaml
                  permissions: '0644'
                bootcmd:
                  - dhclient
                runcmd:
                  - dhclient
                  - netplan generate
                  - netplan apply
                users:
                - name: ubuntu
                  ssh-authorized-keys:
                    - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCtgGlrOj4eOALNutALv+Vs0H+PzP9
            name: cloudinitvolume
        networks:
        - name: default
          pod: {}
        - multus:
            networkName: sriovnet-vn2
          name: private-net
---
apiVersion: v1
kind: Service
```

```
metadata:
  name: kubevirt-vn2-svc
  namespace: apstra-rhocp-demo
spec:
  ports:
  - port: 22
    targetPort: 22
    nodePort: 30254
  selector:
    kubevirt.io/size: large
    kubevirt.io/domain: kubevirt-vm-vn2
    type: eda
  type: NodePort
```

```
oc apply -f DEPLOYMENT-KUBEVIRT.yaml
```

In a similar way to for the Pods, when the above command is run, the Ansible Automation Platform will show the `create_vm.yaml` Playbook running. This will use the same approach to identify the required switch and interface onto which to configure the VLAN ID (`200`) associated with `vn2`. This will be communicated to Apstra, which will bind that port and VLAN on the identified leaf switch to the correct VN. As with for the Pods, you can follow the process on the Ansible Automation Platform UI, then the Apstra UI and you will see that the VM will use a different physical interface than the two Pods used on their respective hosts.

Step 7: Test the Default Connectivity Between the Pods and Between a Pod and the Virtual Machine

Once all the workloads are up and running, you can check the connectivity within a VN and between VNs.

Use the tools provided in the attached git repository at `<repo-base>/Scripts/` to get an easy overview of your workloads.

```
./Scripts/list-pods
./Scripts/list-kubevirt-vms
```

Pick one of the Pods and run a ping between it and the other Pod. They share a VN so should be on the same subnet.

```
oc exec -itn apstra-rhocp-demo <pod1-name> -- ping -c 5 <pod2-sriov-if-ip-address>
```

Now grab the IP address of the SR-IOV interface on the VM and ping that from pod1. This should be on a different subnet than the two Pods, since it is in a different L2 domain.

```
oc exec -itn apstra-rhocp-demo <pod1-name> -- ping -c 5 <vm-sriov-if-ip-address>
```

These should both work.