Connecting virtual machines in the cloud age
What is this about?

- Advanced networking use-cases on Kubernetes and their **challenges**
- Cluster networking **evolution**
- Four pillars of **KubeVirt networking**
Why should I care?

- Beauty of Kubernetes networking

- Tools for complex networking use-cases, both for Pods and VMs
Networking Evolution
Networking Evolution

How the approach to networking changed since the 80s
Evolution
physical ages
Evolution

physical ages

Router

Switch

Switch

Machine

eth0

Machine

eth0

Machine

eth0

Machine

eth0
Evolution

physical ages

Router

Switch

eth1
eth0

Service
Machine

Switch

eth0

Service
Machine

Switch

eth0

Service
Machine

Switch

eth0

Service
Machine
Evolution

cloud ages

overlay network
Evolution of cloud ages

- Private switch
- Storage network
- Overlay network

Diagram showing nodes and services connected through different interfaces (eth2, eth3) and networks (private, storage, overlay).
Four Pillars of Networking
Four Pillars of Networking

KubeVirt Razor: "If something can be used for Pods, it should not be implemented only for VMs"
Four Pillars of Networking

KubeVirt Razor: "If something can be used for Pods, it should not be implemented only for VMs"
Four Pillars of Networking

KubeVirt Razor: "If something can be used for Pods, it should not be implemented only for VMs"
# Four Pillars of Networking

KubeVirt Razor: "If something can be used for Pods, it should not be implemented only for VMs"

<table>
<thead>
<tr>
<th>Node</th>
<th>Logical</th>
<th>Smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Network</td>
<td>Scheduling</td>
</tr>
<tr>
<td>Configuration</td>
<td>Definition</td>
<td></td>
</tr>
</tbody>
</table>
Four Pillars of Networking

KubeVirt Razor: "If something can be used for Pods, it should not be implemented only for VMs"
Node Network Configuration

- Dynamic configuration
- SR-IOV, L2, ...
- Heterogeneous cluster
Node Network Configuration

- kubernetes-nmstate

- **Declarative** network management across cluster nodes

- NodeNetworkState

- NodeNetworkConfigurationPolicy
Node Network Configuration

- eth0
- eth1
Node Network Configuration

- Node: eth0
- Node: eth0, eth1
- Node: eth0, eth1
- Switch
Node Network Configuration

```yaml
apiverison: nmstate.io/v1
kind: nodenetworkstate
metadata:
  name: node01
...
spec:
...
status:
currentstate:
capabilities: null
interfaces:
  - ifindex: 3
    name: eth1
    ipv4:
      ...
    ipv6:
      ...
    mtu: 1500
    state: up
    type: ethernet
...
Node Network Configuration

apiVersion: nmstate.io/v1
kind: nodenetworkstate
metadata:
  name: node01
...
spec:
desiredState:
  managed: true
  interfaces:
    - description: br1 bridge downlink
      name: eth1
      type: ethernet
      state: up
      ipv4:
        enabled: false
      ipv6:
        enabled: false
    - description: Bridge for pod networks
      name: br1
      type: ovs-bridge
      state: up
      bridge:
        options:
        ...  
        port:
          - name: eth1
            type: system
      ipv4:
        enabled: false
      ipv6:
        enabled: false
      status:
        ...
Node Network Configuration

- Node
- eth0
- eth1
- Switch
- eth0
- eth1
- Node
- eth0
- eth1
- Node
- eth0
- eth1
Node Network Configuration
Node Network Configuration

- Some network workloads might require node to be **configured before the Pod is scheduled** (bridges, SR-IOV, …)

- **Requirements change** and all nodes are not the same

- **kubernetes-nmstate** allows administrator to do day 2 network configuration in declarative and centralized manner
Logical Network Definition

- For user: Logical network is "connectivity" requested for a Pod

- For admin: **Specifies how** should be a Pod connected to a network

- **Decoupled** from node network configuration

- Kubernetes Network Custom Resource Definition [De-facto Standard](#)

- Multus
Logical Network Definition

kubelet

CNI

flannel
Logical Network Definition

kubelet

CNI

multus
Logical Network Definition

kubelet

CNI

multus

1. CNI

flannel
Logical Network Definition

kubelet

CNI

1. CNI

multus

(2. CNI)

flannel

ovs
Logical Network Definition

1. CNI
   multus
   1. CNI
   flannel
   2. CNI
   ovs
   3. CNI
   …
Logical Network Definition

Switch "blue net"

Node

eth0

Node

eth0 eth1

br1

Node

eth0 eth1

br1
Logical Network Definition

Switch "blue net"

Node

Node

Node
Logical Network Definition

- Node: eth0
- Node: eth0, eth1
- Node: eth0, eth1
- Pod: eth0, eth1
- Switch: "blue net"

Diagram shows connectivity between nodes and a switch named "blue net".
Logical Network Definition
Logical Network Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: blue-network
annotations:
  k8s.v1.cni.cncf.io/resourceName: ovs-cni.network.kubevirt.io/br1
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "ovs",
    "bridge": "br1",
    "vlan": 100
  }'
```
Logical Network Definition

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: blue-network
annotations:
  k8s.v1.cni.cncf.io/resourceName: ovs-cni.network.kubevirt.io/br1
spec:
  config: '
    "cniVersion": "0.3.1",
    "type": "ovs",
    "bridge": "br1",
    "vlan": 100
  '
Logical Network Definition

apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  annotations:
    k8s.v1.cni.cncf.io/networks: blue-network
spec:
  containers:
  - name: samplepod
    command: ["sleep", "infinity"]
    image: fedora
Logical Network Definition

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
annotations:
    k8s.v1.cni.cncf.io/networks: blue-network
spec:
  containers:
  - name: samplepod
    command: ["sleep", "infinity"]
    image: fedora
```
Logical Network Definition

- **Representation** of network connectivity
- NetworkAttachmentDefinition describes **how to connect** Pod to a network
Smart Scheduling

- Clusters are heterogeneous

- Node selector vs. de-facto standard
Smart Scheduling

pod

Node

eth0

Node

eth0  eth1

Node

eth0  eth1  br1

Switch "blue net"
Smart Scheduling

nets: blue-net
pod

Switch "blue net"
Smart Scheduling

nets: blue-net
pod

Node
eth0

Switch
"blue net"

Node
eth0
eth1
br1

Node
eth0
eth1
br1
Smart Scheduling

nets: blue-net
pod

Switch
"blue net"
Smart Scheduling

nets: blue-net
pod

Node

eth0

Node

eth0  eth1

Switch
"blue net"

eth0  eth1

br1

Node
Smart Scheduling

Switch "blue net"

Node

eth0

Node

eth0 eth1

br1

nets: blue-net

pod

Node

eth0 eth1

br1

nets: blue-net
Smart Scheduling

- Node resources

- Kubernetes takes care of resource allocation and release

Capacity:
  cpu: 2
  memory: 2049008Ki
  ovs-cni.network.kubevirt.io/br1: 1Gi
  intel.com/sriov_net_A: 8
Smart Scheduling

- Unlimited or shareable resources

- "Extended resources"

- Kubernetes tracks available volume

- e.g. L2 connectivity via Linux bridge
Smart Scheduling

- Limited and countable resources
- "Device Plugin"
- Kubernetes keeps track of every single device providing a resource
- e.g. L2 connectivity using SR-IOV virtual functions
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: blue-network
  annotations:
    k8s.v1.cni.cncf.io/resourceName: ovs-cni.network.kubevirt.io/br1
spec:
  config: '{
      "cniVersion": "0.3.1",
      "type": "ovs",
      "bridge": "br1",
      "vlan": 100
    }'
Smart Scheduling

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
    name: blue-network
annotations:
    k8s.v1.cni.cncf.io/resourceName: ovs-cni.network.kubevirt.io/br1
spec:
    config: '{
        "cniVersion": "0.3.1",
        "type": "ovs",
        "bridge": "br1",
        "vlan": 100
    }'
Smart Scheduling

- Kubernetes provides tools to **expose available resources**

- Multus glues **Logical Network Definition and Node resources** together
VM Binding Mechanism

- Virtual machines is just another process running inside a Pod
- KubeVirt pods are treated as any other Kubernetes pod
- The only KubeVirt specific pillar
- Other components are just being consumed
VM Binding Mechanism
VM Binding Mechanism

- eth0
  - 10.10.0.1

pod
VM Binding Mechanism

container

pod

eth0
10.10.0.1
VM Binding Mechanism

- pod
- eth0 10.10.0.1
- a process
- container
- pod
VM Binding Mechanism

- VM (virtual machine)
- Container
- Pod
- eth0
  - 10.10.0.1
VM Binding Mechanism

container

virtual machine

pod

eth0

10.10.0.1
VM Binding Mechanism

[Diagram showing a virtual machine within a container, labeled "eth0" and connected to a pod with an IP address of 10.10.0.1.]
VM Binding Mechanism

- L2 binding using a Linux bridge
- Virtual machine takes over Pod network
VM Binding Mechanism

code:

- pod
- eth0
- 10.10.0.1
- container
- virtual machine
- eth0

Diagram:
- Pod containing a container with virtual machine
- Container with eth0 interface
- Pod IP: 10.10.0.1
- Cloud icon pointing to pod

Explanation:
- The diagram illustrates the binding mechanism between a pod, container, and virtual machine. The pod contains a container, which in turn has a virtual machine with an eth0 interface.
- The pod is connected via its IP address (10.10.0.1) to the cloud, indicating how it communicates with other pods or external networks.
VM Binding Mechanism

- Virtual machine
- Container
- Pod
- eth0
- br1
- DHCP server
  - offering 10.10.0.1 on br1
VM Binding Mechanism

- virtual machine
- container
- pod
- eth0
- br1
- DHCP server
  - offering 10.10.0.1
  - on br1
- DHCP server offering 10.10.0.1 on br1
VM Binding Mechanism

- Virtual machine
  - eth0: 10.10.0.1
- Container
- Pod
- dhcp-server
  - offering: 10.10.0.1
  - on: br1
VM Binding Mechanism

apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: vmi-test
  ...

spec:
  networks:
    - name: default
      pod: {}
  domain:
    devices:
      ...
      interfaces:
        - name: default
          bridge: {}
      ...
  ...
  ...

VM Binding Mechanism

- L4 **NAT** binding

- Traffic addressed **for specific port** is forwarded to virtual machine

- Allows **side containers** to access Kubernetes network
VM Binding Mechanism

- **container**
  - **virtual machine**
    - eth0
      - 10.10.0.1

- **pod**
  - br1
  - DHCP server
    - offering 192.168.0.1 on br1
VM Binding Mechanism

- pod
  - eth0 10.10.0.1
  - 192.168.0.1
- container
  - virtual machine
  - eth0 192.168.0.1
- DHCP server
  - offering 192.168.0.1 on br1
VM Binding Mechanism

- **eth0**
  - 10.10.0.1

- **container**

- **virtual machine**
  - eth0
  - 192.168.0.1

- **pod**

- **iptables**
  - if dst.port == 80
  - forward to br1 192.168.0.1

- **DHCP server**
  - offering 192.168.0.1 on br1
VM Binding Mechanism

- **pod**
- **eth0**
- **10.10.0.1**
- **container**
- **virtual machine**
- **iptables**
- if **dst.port == 80**
- forward to **br1 192.168.0.1**
- **DHCP server**
- offering **192.168.0.1**
- on **br1**
- **br1**
- **if dst.port == 80**
- forward to **br1 192.168.0.1**
- **eth0**
- **192.168.0.1**
- **virtual machine**
- pod
- **container**
- **a process**
- **side container**
VM Binding Mechanism

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: vmi-test
...

spec:
  networks:
    - name: default
      pod: {}
  domain:
    devices:
      ...
    interfaces:
      - name: default
        masquerade: {}
        ports:
          - 80
      ...
```
VM Binding Mechanism

- Secondary network interfaces provided by Multus
- The same binding process
VM Binding Mechanism

apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: vmi-test

spec:
  networks:
    - name: default
      pod: {}
    - name: blue
      multus:
        networkName: blue-network
  domain:
    devices:
      ...
    interfaces:
      - name: default
        bridge: {}
      - name: blue
        bridge: {}
    ...
    ...
    ...

VM Binding Mechanism

- Virtual machine is **just another process** running in Kubernetes Pod

- **Performance and capabilities** vary based on selected binding mechanism
Takeaways

- Container clouds changed **view on networking**
- It is possible to run **networking heavy workloads** on Kubernetes
- **Available tools and mechanisms** to handle such use-cases
So long
and
thanks for listening!
KubeVirt https://kubevirt.io/

Kubernetes https://kubernetes.io/

Multus CNI https://github.com/intel/multus-cni

Open vSwitch CNI https://github.com/kubevirt/ovs-cni

NMState http://nmstate.io

NMState https://github.com/nmstate/nmstate

Kubernetes NMState https://github.com/nmstate/kubernetes-nmstate

Kubernetes Node Network Configuration design document https://tinyurl.com/y7otybg9