Red Hat
OpenShift
Container Platform

KubeVirt Networking

ONIC Japan 2024 BOF

Manabu Ori
Red Hat

v1.7

Red Hat

KubeVirtとは

- KVM仮想マシンをKubernetes上のPodとして動かす仕組み
  - コンテナワークロードと仮想化ワークロードがKubernetes上で共存
- CNCF Incubating Project
- 開発状況
  - コミュニティでアクティブに開発が進行中
  - 2023年時点での実績
    - 200以上の企業が開発に貢献
      - 2022年から50%増
    - 60リリース
    - CNCFのアクティブなプロジェクトトップ10
- 2010年始めくらいに、k8s上でVMを動かす勢がいくつか出てきたのですが、生き残ったのはKubeVirtだけ…のはず
  - e.g. Mirantis Virtlet https://www.mirantis.com/blog/virtlet-run-vms-as-kubernetes-pods

企業別貢献度ランキング

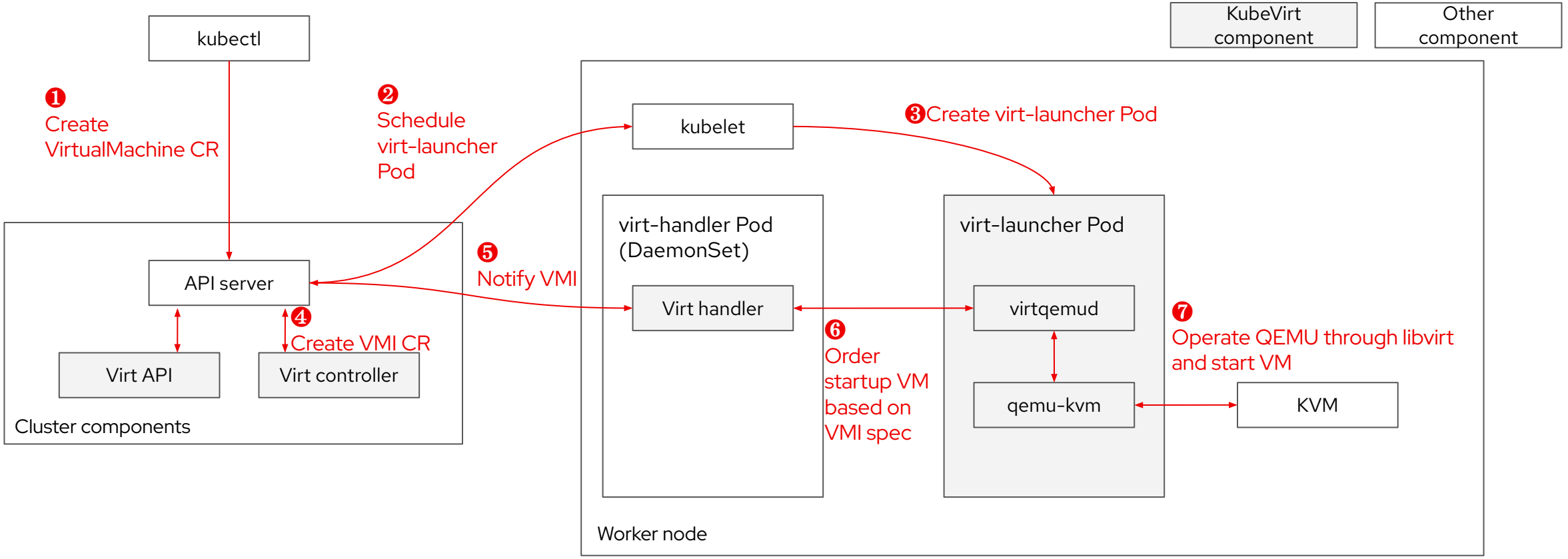
https://all.devstats.cncf.io/d/4/company-statistics-by-repository-group?orgId=1&var-period=m&var-metric=contributions&var-repogroup_name=KubeVirt&var-companies=All&from=now-4y&to=now

Red Hat

KubeVirtとは

- Kubernetesの仕組みを享受
  - ネットワーク
    - CNIプラグインによる外部との接続
    - Serviceによるサービスディスカバリ、ロードバランス
    - Ingress, LoadBalancer Serivice, NodePort等による外部からのアクセス
    - Network Policyによるアクセス制御
  - ストレージ
    - CSIプラグインによる外部ストレージとの連携
    - 仮想マシンのディスクはk8sのPersistent Volumeとして提供
- VMスナップショット(CSI Snapshotを利用)
- ライブマイグレーション
  - 仮想マシンのボリュームはRWXのPVであることが必須
- HA
  - Medik8s

仮想マシンのデプロイの流れ



Red Hat

登場人物

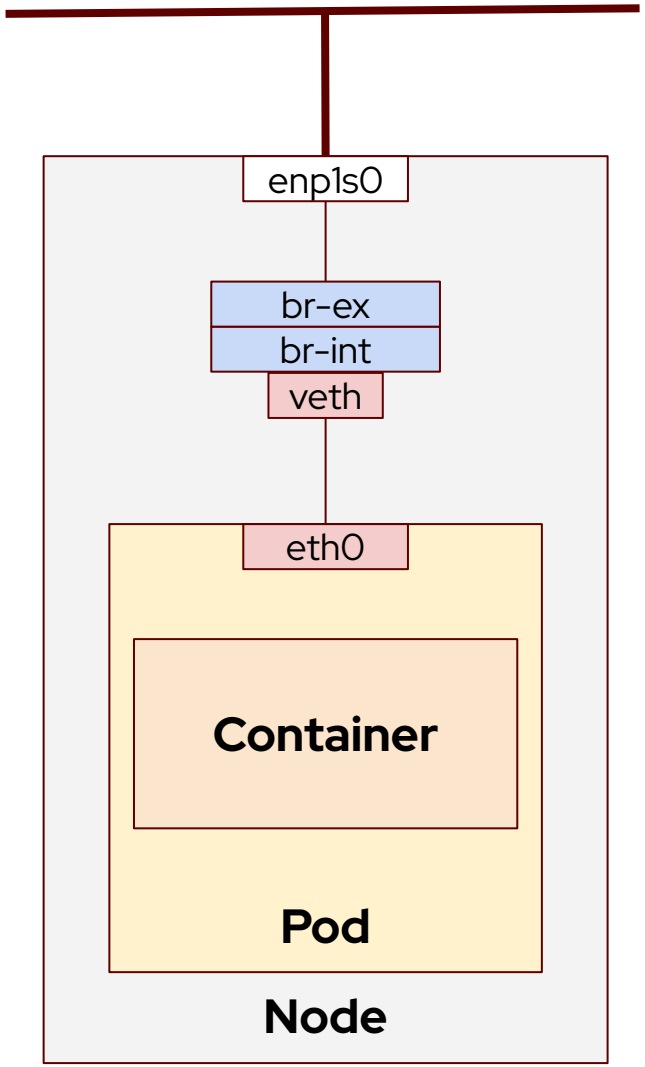
- hco-operator
  - **virt-operator**
  - cdi-operator
    - Containerized Data Importer
  - cluster-network-addon-operator
  - host-path-provisioner-operator
  - ssp-operator
    - Scheduling, Scale and Performance

(最低限 KubeVirt Operator (virt-operator) がいれば検証できます)

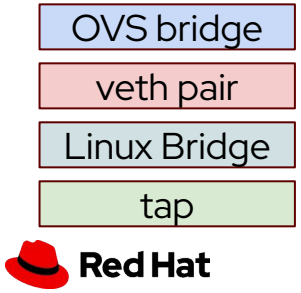
OpenShift v4.16で動く
KubeVirt関連Pod

```
$ oc -n openshift-cnv get pod
NAME                                               READY   STATUS    RESTARTS        AGE
aaq-operator-596cfd7fdb-6q6cf                      1/1     Running   0               2d17h
bridge-marker-8xxc8                                1/1     Running   0               2d17h
bridge-marker-dz4gb                                1/1     Running   0               2d17h
bridge-marker-shj6m                                1/1     Running   0               2d17h
cdi-apiserver-6f47789d9f-zj49l                     1/1     Running   0               2d17h
cdi-deployment-856f4df8bb-ddzwn                    1/1     Running   0               2d17h
cdi-operator-6b95fb447-kchsw                       1/1     Running   0               2d17h
cdi-uploadproxy-64d7ccb9fc-lb8jj                   1/1     Running   0               2d17h
cluster-network-addons-operator-59c8d6b878-cz2j6   2/2     Running   0               2d17h
hco-operator-58799b7b9b-8gtnk                      1/1     Running   0               2d17h
hco-webhook-7d4d997fc7-nwf2s                       1/1     Running   0               2d17h
hostpath-provisioner-operator-9884b8b8-kv7h8       1/1     Running   0               2d17h
hyperconverged-cluster-cli-download-79dd68654d-j95mg   1/1     Running   0           2d17h
kube-cni-linux-bridge-plugin-4s9fd                 1/1     Running   0               2d17h
kube-cni-linux-bridge-plugin-hmd2j                 1/1     Running   0               2d17h
kube-cni-linux-bridge-plugin-vgh5w                 1/1     Running   0               2d17h
kubemacpool-cert-manager-cd8bd9b-kwbn7             1/1     Running   0               2d17h
kubemacpool-mac-controller-manager-687bb69558-8kmbp   2/2     Running   0            2d17h
kubevirt-apiserver-proxy-6cf95886bb-2z6gt          1/1     Running   0               2d17h
kubevirt-apiserver-proxy-6cf95886bb-fx95q          1/1     Running   0               2d17h
kubevirt-console-plugin-66748d85cb-5flv9           1/1     Running   0               2d17h
kubevirt-console-plugin-66748d85cb-cwfnp           1/1     Running   0               2d17h
mtq-operator-5584b589d9-wx8dl                      1/1     Running   0               2d17h
ssp-operator-67657bdc9f-bddhj                      1/1     Running   1 (2d17h ago)   2d17h
virt-api-77cc94cddc-72blk                          1/1     Running   0               2d17h
virt-api-77cc94cddc-tl6sk                          1/1     Running   0               2d17h
virt-controller-b645f9-f7nv8                       1/1     Running   0               2d17h
virt-controller-b645f9-zqlfm                       1/1     Running   0               2d17h
virt-exportproxy-64bcb95d78-jc7b4                  1/1     Running   0               2d17h
virt-exportproxy-64bcb95d78-shwb7                  1/1     Running   0               2d17h
virt-handler-kt4k8                                 1/1     Running   0               2d17h
virt-handler-lnf6x                                 1/1     Running   0               2d17h
virt-handler-mbxvl                                 1/1     Running   0               2d17h
virt-operator-7484fdc485-86mqh                     1/1     Running   0               2d17h
virt-operator-7484fdc485-pn26f                     1/1     Running   0               2d17h
virt-template-validator-5c975bd855-xmlsn           1/1     Running   0               2d17h
virt-template-validator-5c975bd855-zrst2           1/1     Running   0               2d17h
```

k8sネットワークおさらい: Pod/コンテナのネットワーク接続

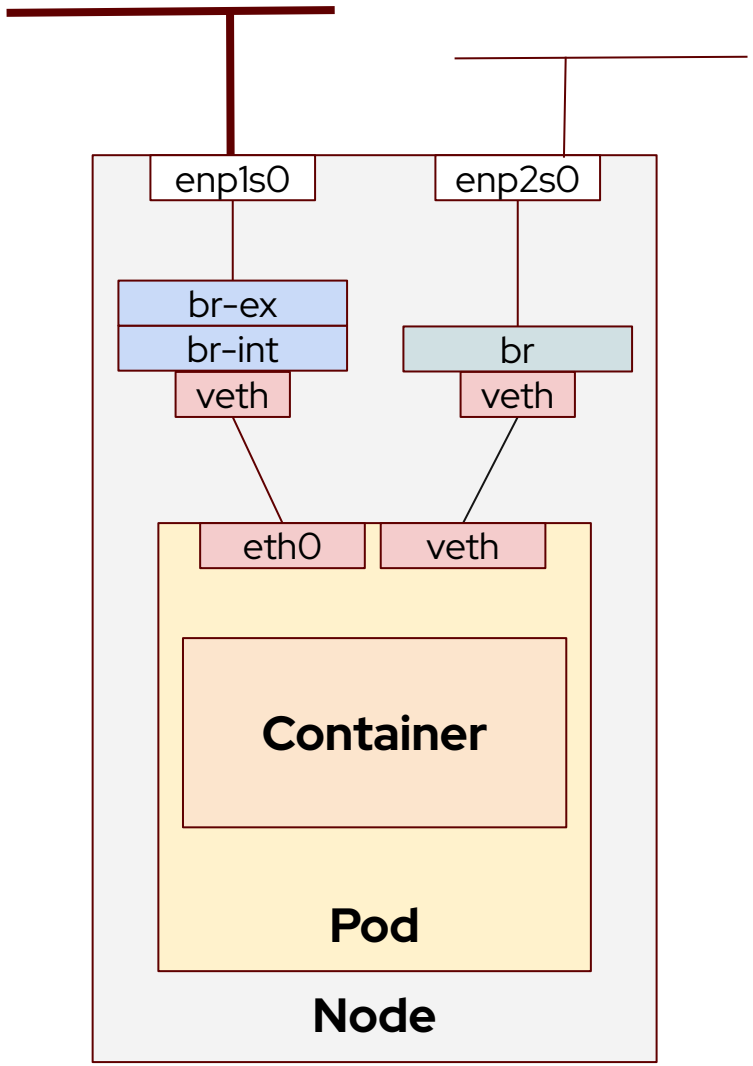


- ノードの物理インターフェースとPodの接続はCNIプラグインが管理
- IPアドレスはCNIプラグインのIPAMが管理
- 通常、KubernetesのPodが持つインターフェースはひとつだけ
  - Multusを使うとPodを複数ネットワークに接続可



Red Hat

k8sネットワークおさらい: Multus使用時



- Multus: 複数のCNIプラグインをまとめるメタCNIプラグイン
- デフォルトCNI
  - 従来のPodネットワークに接続するCNI
- セカンダリCNI
  - 別ネットワークに接続するためのCNI
- セカンダリCNI側は、できることが限られる
  - Service (upstreamで実験的な実装はあったけど...)
  - Network Policy (Multi NetworkPolicy)
  - IPAM (DHCP Plugin or CNI embedded IPAM)

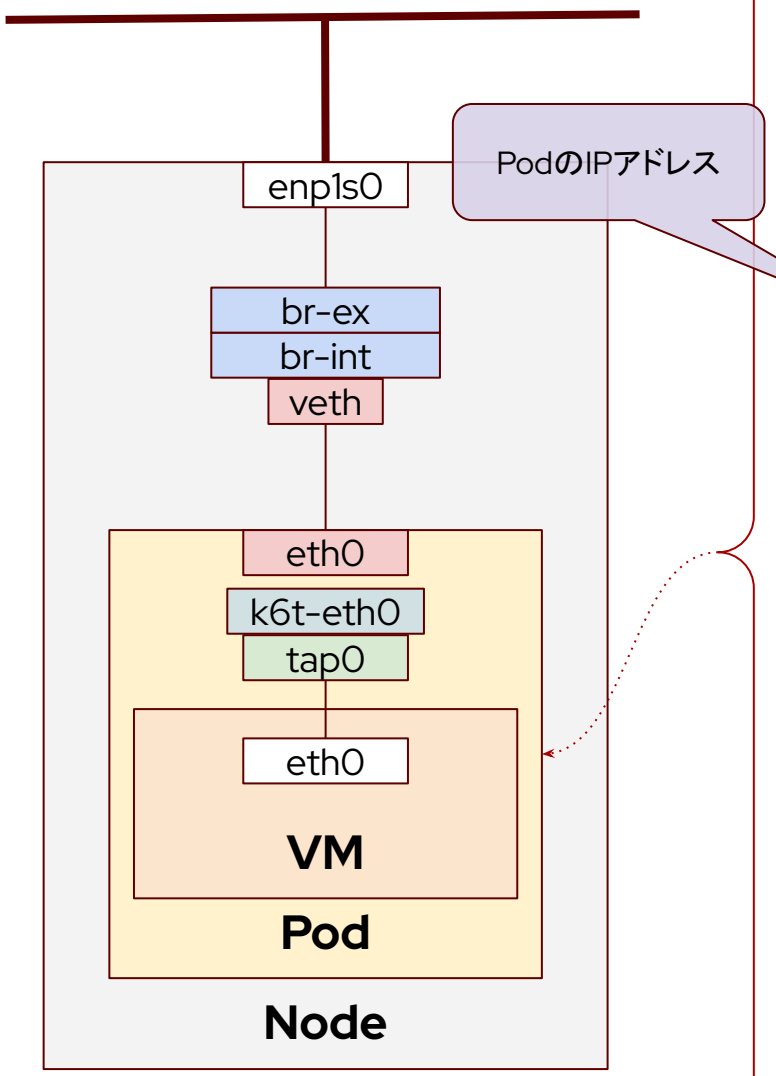
OVS bridge
veth pair
Linux Bridge
tap

Red Hat

```
$ kubectl get pod -o json virt-launcher-fedora-0-8twvt | jq -r .status.podIP
10.129.2.132
```

仮想マシンのネットワーク接続 (masquerade)

Podのインターフェース




```
localhost:~$ oc exec virt-launcher-fedora-0-8twvt -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if165: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdis
state UP group default
    link/ether 0a:58:0a:81:02:84 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.129.2.132/23 brd 10.129.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::858:aff:fe81:284/64 scope link
       valid_lft forever preferred_lft forever
3: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc
UP group default qlen 1000
    link/ether 02:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.1/24 brd 10.0.2.255 scope global k6t-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ff:fe00:0/64 scope link
       valid_lft forever preferred_lft forever
4: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_c
k6t-eth0 state UP group default qlen 1000
    link/ether 4e:0c:58:54:2b:f8 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::4c0c:58ff:fe54:2bf8/64 scope link
       valid_lft forever preferred_lft forever
```

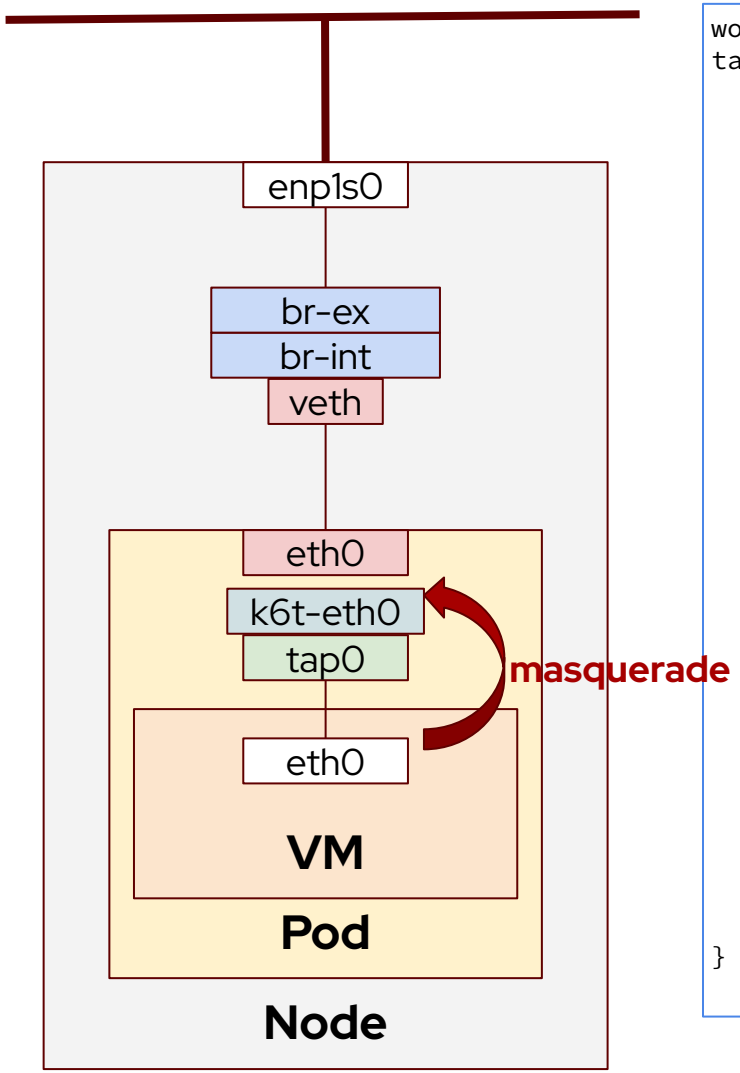
veth

bridge

tap

```
localhost:~$ oc exec virt-launcher-fedora-0-8twvt -- ip r
default via 10.129.2.1 dev eth0
10.0.2.0/24 dev k6t-eth0 proto kernel scope link src 10.0.2.1
10.128.0.0/14 via 10.129.2.1 dev eth0
10.129.2.0/23 dev eth0 proto kernel scope link src 10.129.2.132
100.64.0.0/16 via 10.129.2.1 dev eth0
172.30.0.0/16 via 10.129.2.1 dev eth0
```

仮想マシンのネットワーク接続 (masquerade)



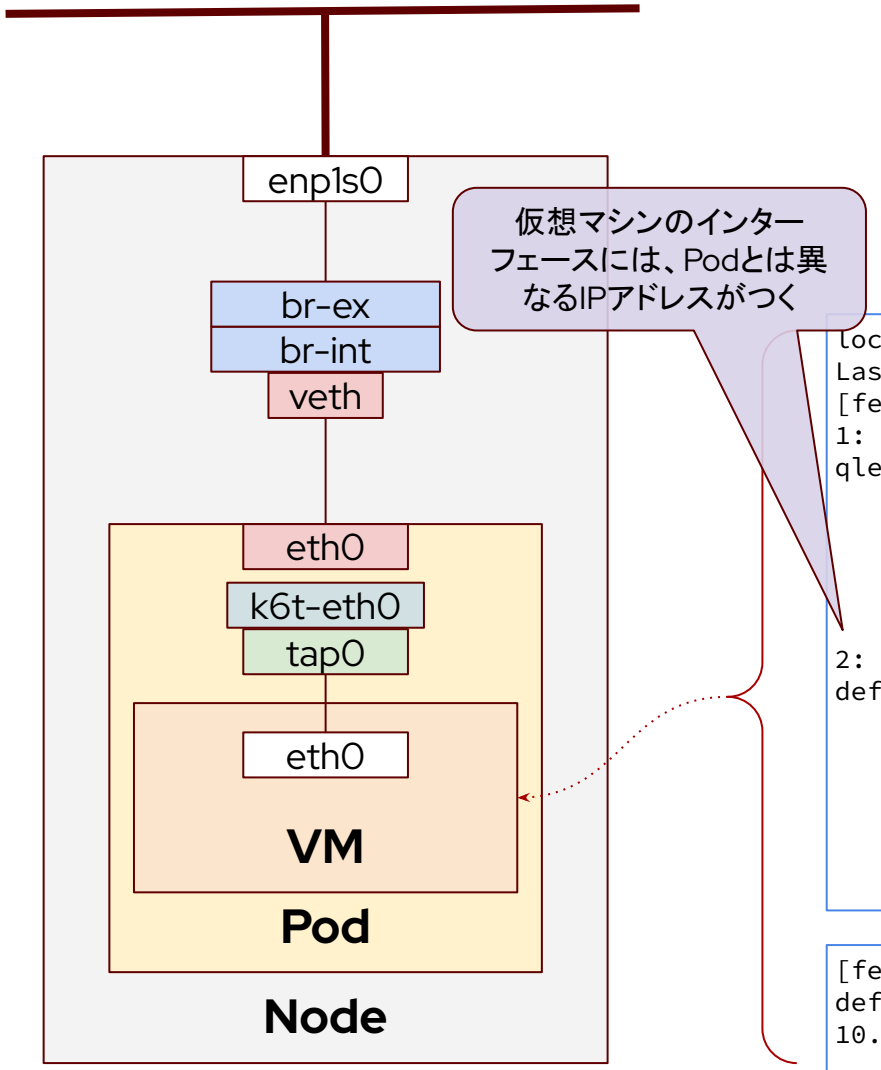
Pod内のnftablesルール

```
localhost:~$ oc exec virt-launcher-fedora-0-8twvt -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
worker-0:~$ sudo nsenter -t 2062231 -n nft list table nat
table ip nat {
        chain prerouting {
                type nat hook prerouting priority dstnat; policy accept;
                iifname "eth0" counter packets 2 bytes 120 jump KUBEVIRT_PREINBOUND
        }

        chain input {
                type nat hook input priority srcnat; policy accept;
        }

        chain output {
                type nat hook output priority dstnat; policy accept;
                ip daddr 127.0.0.1 counter packets 0 bytes 0 dnat to 10.0.2.2
        }

        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                ip saddr 10.0.2.2 counter packets 157 bytes 11804 masquerade
                oifname "k6t-eth0" counter packets 4 bytes 579 jump KUBEVIRT_POSTINBOUND
        }

        chain KUBEVIRT_PREINBOUND {
                counter packets 2 bytes 120 dnat to 10.0.2.2
        }

        chain KUBEVIRT_POSTINBOUND {
                ip saddr 127.0.0.1 counter packets 0 bytes 0 snat to 10.0.2.1
        }
}
```

```
default via 10.129.2.1 dev eth0
10.0.2.0/24 dev k6t-eth0 proto kernel scope link src 10.0.2.1
10.128.0.0/14 via 10.129.2.1 dev eth0
10.129.2.0/23 dev eth0 proto kernel scope link src 10.129.2.132
100.64.0.0/16 via 10.129.2.1 dev eth0
172.30.0.0/16 via 10.129.2.1 dev eth0
```


Red Hat

仮想マシンのネットワーク接続 (masquerade)



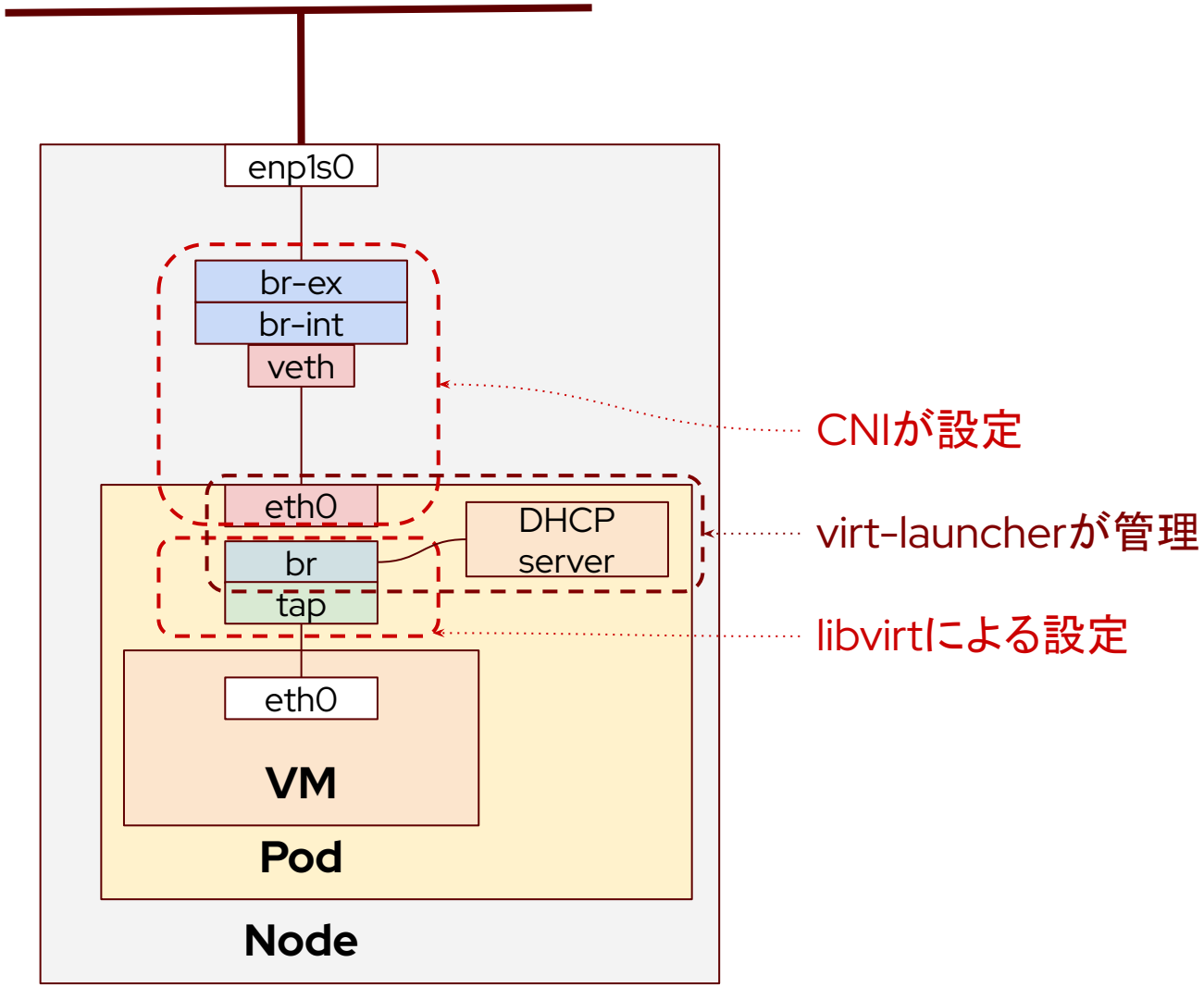
仮想マシンのインターフェースには、Podとは異なるIPアドレスがつく

VMのインターフェース

```
localhost: ~$ oc virt ssh fedora-0
Last login: Thu Oct 10 04:34:07 2024 from 10.131.0.57
[fedora-0: ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UP group
default qlen 1000
    link/ether 02:b3:f0:00:00:45 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 10.0.2.2/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 86306184sec preferred_lft 86306184sec
    inet6 fe80::b3:f0ff:fe00:45/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

```
[fedora-0: ~]$ ip r
default via 10.0.2.1 dev eth0 proto dhcp src 10.0.2.2 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.2 metric 100
```

仮想マシンのネットワーク接続 (masquerade)



- VMのIPアドレスはDHCPで付与
- virt-launcherが自前DHCPサーバをGo routineとして起動

OVS bridge
veth pair
Linux Bridge
tap

Red Hat

VirtualMachineカスタムリソースを見てみる

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...
spec:
  template:
    spec:
...
      domain:
        devices:
          interfaces:
          - name: default
            masquerade: {}
...
      networks:
      - name: default
        pod: {}
...
```

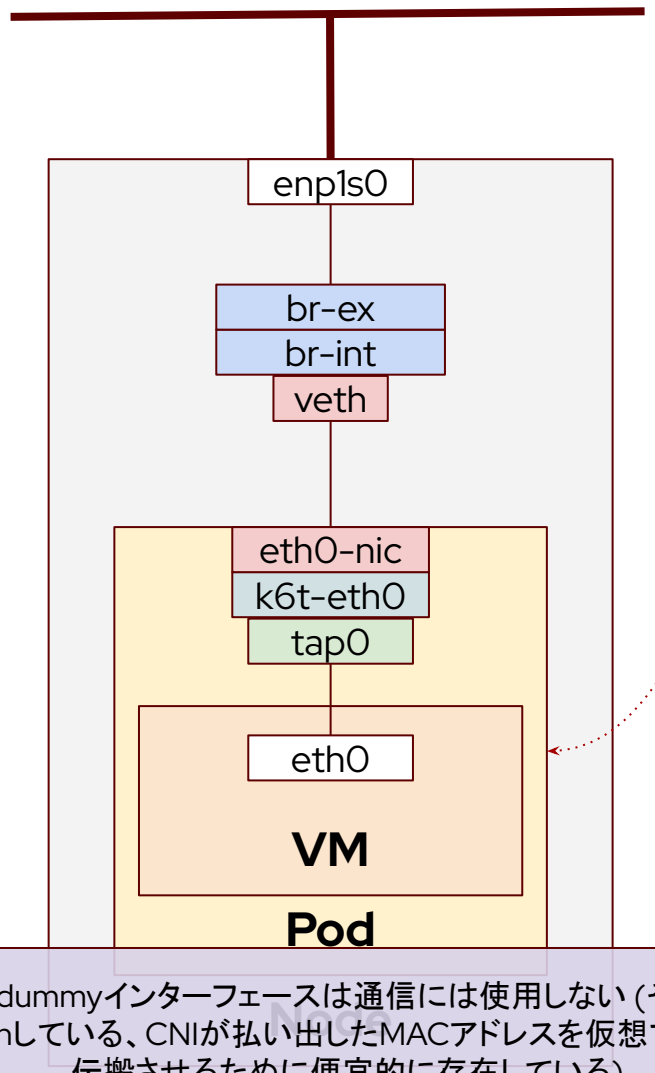
frontend
(仮想NIC)

backend
(接続先)

- VirtualMachineのネットワーク設定は2段階
  - VMの仮想NIC (frontend)
    - **pod**
    - multus
  - 仮想NICの接続先 (backend)
    - bridge
    - **masquerade**
    - slirp
    - sriov

※ p.9～p.12の絵は、Pod Networkにmasqueradeで接続した構成です

cf. Pod Networkにbridge接続する場合



Podのインターフェース

```
localhost:~$ oc exec virt-launcher-fedora-9-cbf8r -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0-nic@if310: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 q
master k6t-eth0 state UP group default
    link/ether 9e:d8:2d:12:d8:ca brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::9cd8:2dff:fe12:d8ca/64 scope link
       valid_lft forever preferred_lft forever
3: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state
UP group default qlen 1000
    link/ether 8e:4c:fb:01:c6:3a brd ff:ff:ff:ff:ff:ff
    inet 169.254.75.10/32 scope global k6t-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::9cd8:2dff:fe12:d8ca/64 scope link
       valid_lft forever preferred_lft forever
4: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel master
k6t-eth0 state UP group default qlen 1000
    link/ether 8e:4c:fb:01:c6:3a brd ff:ff:ff:ff:ff:ff
    inet6 fe80::8c4c:fbff:fe01:c63a/64 scope link
       valid_lft forever preferred_lft forever
5: eth0: <BROADCAST,NOARP> mtu 1400 qdisc noop state DOWN group default
qlen 1000
    link/ether 0a:58:0a:83:01:0e brd ff:ff:ff:ff:ff:ff
    inet 10.131.1.14/23 brd 10.131.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::858:aff:fe83:10e/64 scope link
       valid_lft forever preferred_lft forever
```

veth

bridge

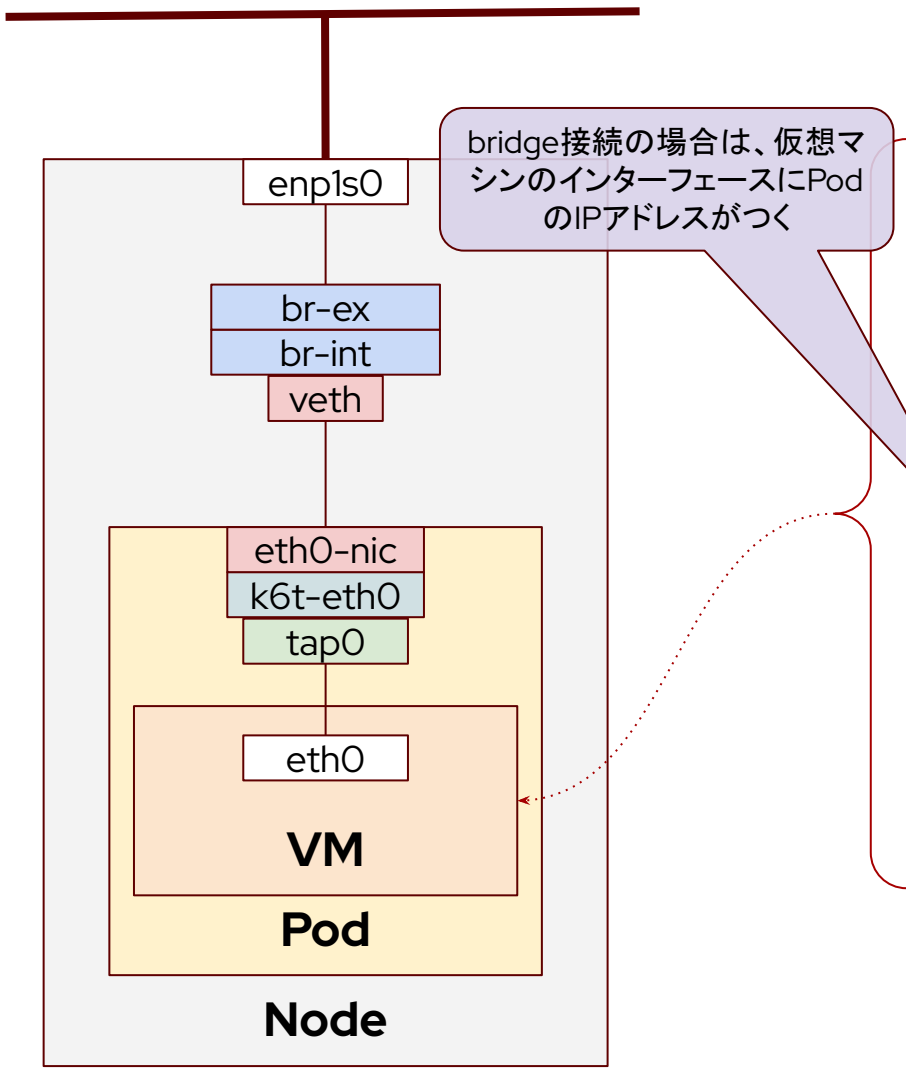
tap

dummy

このdummyインターフェースは通信には使用しない (そもそもDownしている、CNIが払い出したMACアドレスを仮想マシンに伝搬させるために便宜的に存在している)

```
localhost:~$ oc exec virt-launcher-fedora-9-cbf8r -- ip r
10.131.0.0/23 dev eth0 proto kernel scope link src 10.131.1.14
```

cf. Pod Networkにbridge接続する場合



bridge接続の場合は、仮想マシンのインターフェースにPodのIPアドレスがつく

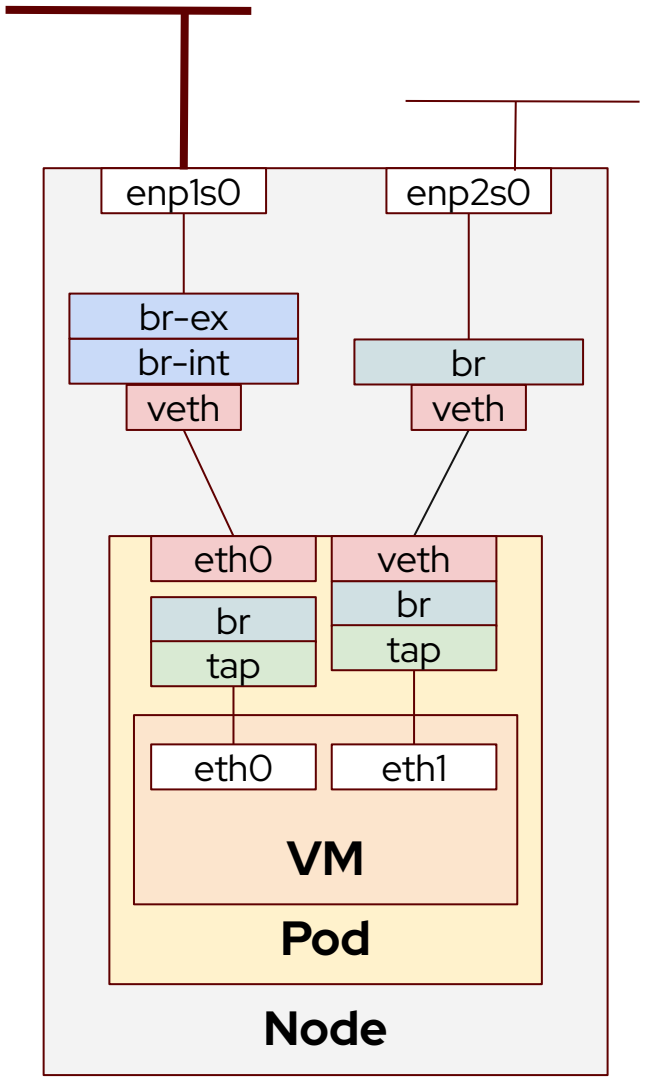
VMのインターフェース

```
[fedora-9 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UP
group default qlen 1000
    link/ether 0a:58:0a:83:01:0e brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 10.131.1.14/23 brd 10.131.1.255 scope global dynamic noprefixroute eth0
       valid_lft 86305978sec preferred_lft 86305978sec
    inet6 fe80::858:aff:fe83:10e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

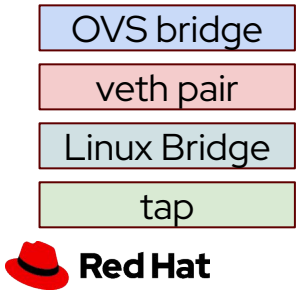
```
[fedora-9 ~]$ ip r
default via 10.131.0.1 dev eth0 proto dhcp src 10.131.1.14 metric 100
10.131.0.0/23 dev eth0 proto kernel scope link src 10.131.1.14 metric 100
100.64.0.0/16 via 10.131.0.1 dev eth0 proto dhcp src 10.131.1.14 metric 100
172.30.0.0/16 via 10.131.0.1 dev eth0 proto dhcp src 10.131.1.14 metric 100
```


Red Hat

仮想マシンを複数のネットワークに接続

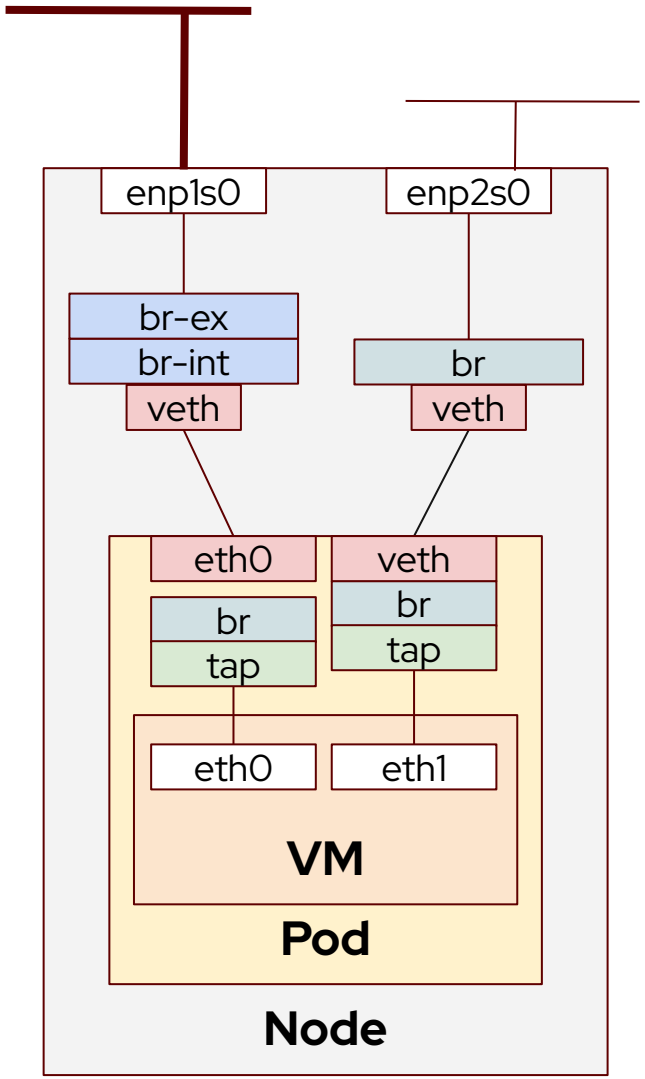


- MultusでセカンダリCNIを設定
  - 必要に応じて各ノードにLinux Bridge等を作成
- カスタムリソースNetworkAttachmentDefinitionで追加ネットワークを定義
- カスタムリソースVirtualMachineで追加NICおよび接続先を記述



Red Hat

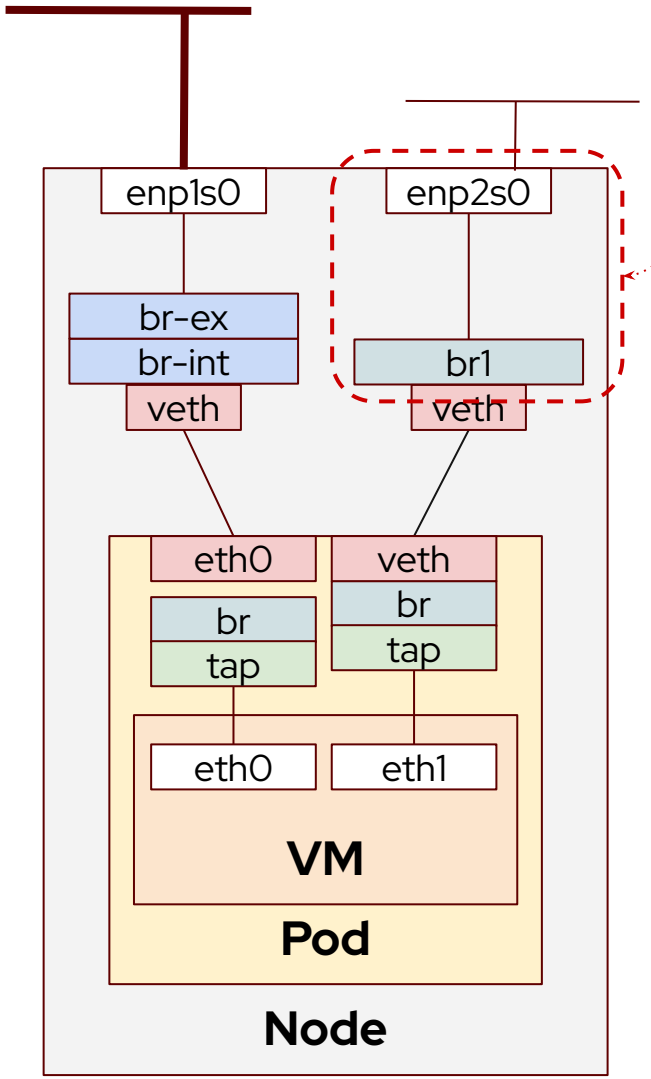
仮想マシンを複数のネットワークに接続




```
kind: NetworkAttachmentDefinition
metadata:
  name: virt-bridge
  annotations:
    k8s.v1.cni.cncf.io/resourceName:
bridge.network.kubevirt.io/br1
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "virt-bridge",
    "type": "bridge",
    "bridge": "br1",
    "macspoofchk": false,
    "vlan": 33
  }'
```

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...
spec:
  template:
    spec:
      domain:
        devices:
          interfaces:
          - name: default
            masquerade: {}
          - name: nic-add1
            bridge: {}
...
      networks:
      - name: default
        pod: {}
      - name: nic-add1
        multus:
          networkName: virt-bridge
      volumes:
      - dataVolume:
          name: fedora-1
        name: rootdisk
...
      - cloudInitNoCloud:
          networkData: |
            ethernets:
              eth1:
                addresses:
                  - 172.17.33.250/24
              version: 2
```

余談: 追加接続用のブリッジの設定が結構めんどくさい...



nmstate operator

- Multusでセカンダリ CNIを設定
  - **必要に応じて各ノードに Linux Bridge等を作成**

⇩

これが結構めんどくさい

⇩

nmstate operatorがお役に立てるかも

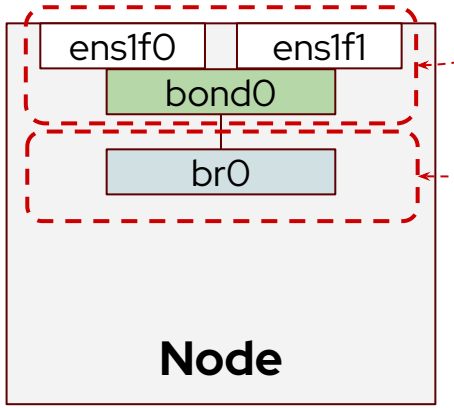
OVS bridge
veth pair
Linux Bridge
tap

Red Hat

余談: 追加接続用のブリッジの設定が結構めんどくさい...

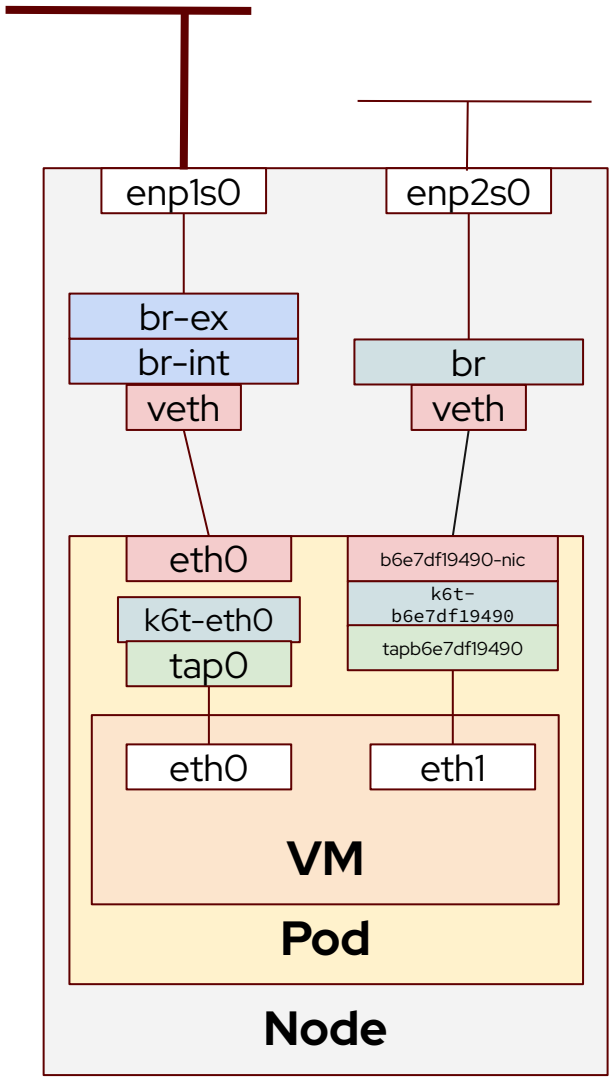
KubeVirtとは直接関係ないですが...

- nmstate https://nmstate.io/
  - 宣言的なネットワーク管理API・ツール
  - NetworkManagerを使用
- nmstate operator https://nmstate.io/kubernetes-nmstate/
  - nmstateを使ってノードのNICを設定するOperator
  - netplanのrendererをNetworkManagerにすればUbuntuでも動くはず？




```
spec:
  nodeSelector:
    node-role.kubernetes.io/virt: ''
  desiredState:
    interfaces:
      - name: br0
        state: up
        type: linux-bridge
        bridge:
          options:
            stp:
              enabled: true
          port:
            - name: bond0
        ipv4:
          enabled: true
          dhcp: false
          address:
            - ip: 172.16.1.21
              prefix-length: 24

      - name: bond0
        state: up
        type: bond
        link-aggregation:
          mode: 802.3ad
          port:
            - ens1f0
            - ens1f1
        ipv4:
          enabled: false
        ipv6:
          enabled: false
```

Podのインターフェース




```
$ oc exec virt-launcher-fedora-1-rkd8z -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if125: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default
    link/ether 0a:58:0a:83:00:56 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.131.0.86/23 brd 10.131.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::858:aff:fe83:56/64 scope link
       valid_lft forever preferred_lft forever
3: b6e7df19490-nic@if126: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue master
k6t-b6e7df19490 state UP group default
    link/ether 5e:c1:7d:9f:f8:6f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::5cc1:7dff:fe9f:f86f/64 scope link
       valid_lft forever preferred_lft forever
4: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default qlen 1000
    link/ether 02:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.1/24 brd 10.0.2.255 scope global k6t-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ff:fe00:0/64 scope link
       valid_lft forever preferred_lft forever
5: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel master k6t-eth0 state UP group
default qlen 1000
    link/ether ea:42:e7:23:ae:39 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::e842:e7ff:fe23:ae39/64 scope link
       valid_lft forever preferred_lft forever
6: k6t-b6e7df19490: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default qlen
1000
    link/ether 5e:c1:7d:9f:f8:6f brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5cc1:7dff:fe9f:f86f/64 scope link
       valid_lft forever preferred_lft forever
7: tapb6e7df19490: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel master k6t-b6e7df19490
state UP group default qlen 1000
    link/ether 9e:b4:a4:8d:8e:6d brd ff:ff:ff:ff:ff:ff
    inet6 fe80::9cb4:a4ff:fe8d:8e6d/64 scope link
       valid_lft forever preferred_lft forever
8: podb6e7df19490: <BROADCAST,NOARP> mtu 1400 qdisc noop state DOWN group default qlen 1000
    link/ether 02:b3:f0:00:00:3c brd ff:ff:ff:ff:ff:ff
    inet6 fe80::b3:f0ff:fe00:3c/64 scope link
       valid_lft forever preferred_lft forever
```

veth

veth

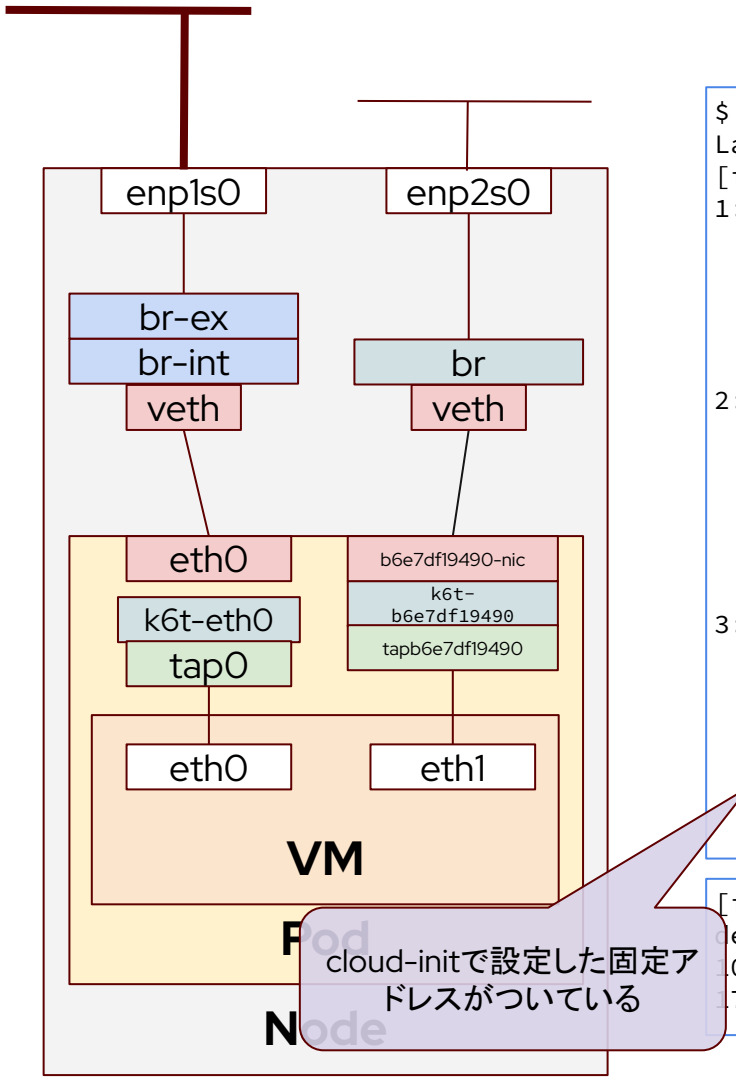
bridge

tap

bridge

tap

dummy



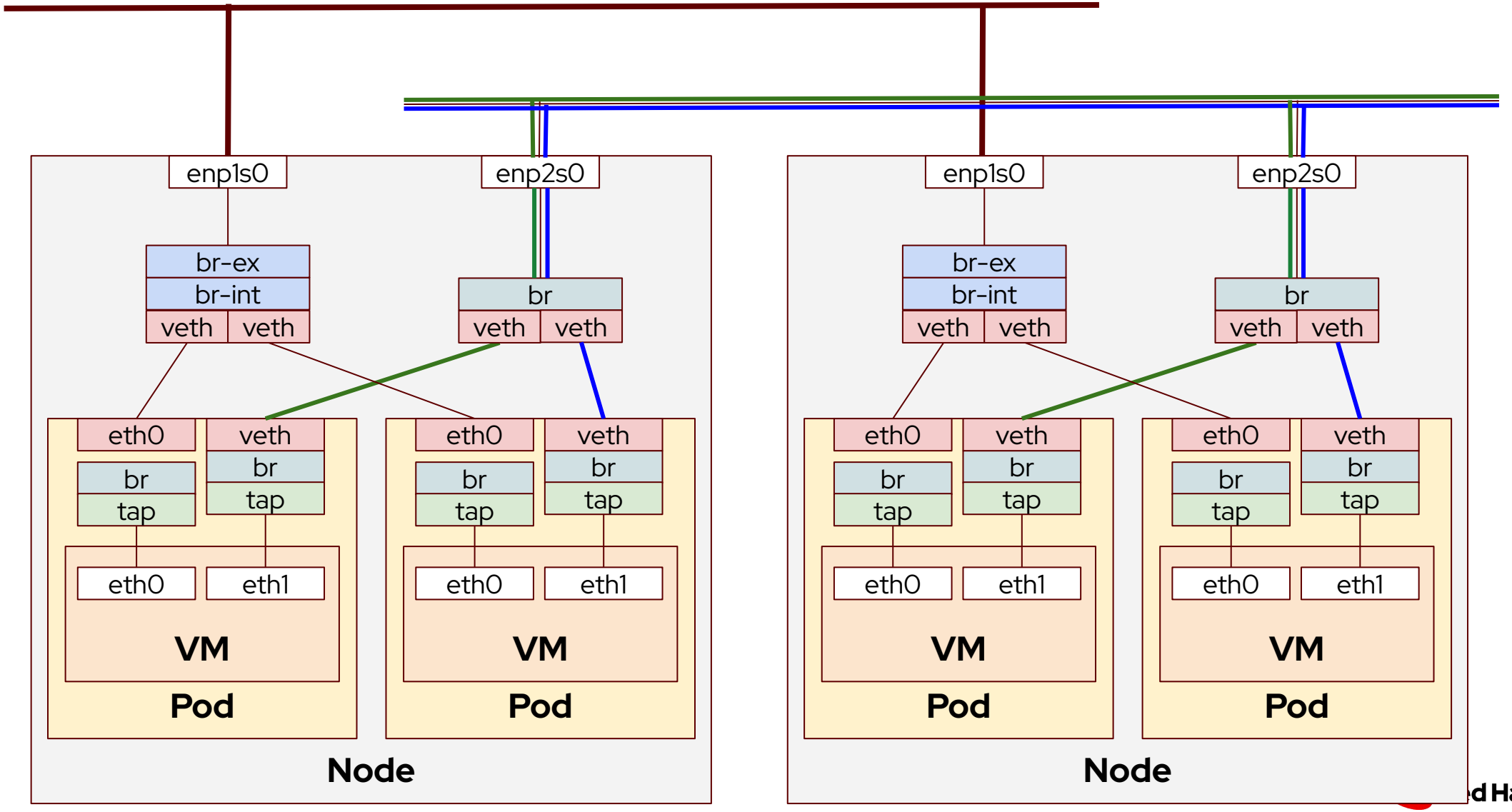
VMのインターフェース

```
$ oc virt ssh fedora-1 -i ~/.ssh/gpscloud_id_rsa -l fedora
Last login: Thu Oct 10 06:28:33 2024 from 10.131.0.57
[fedora@fedora-1 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:b3:f0:00:00:3b brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 10.0.2.2/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 86142178sec preferred_lft 86142178sec
    inet6 fe80::b3:f0ff:fe00:3b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:b3:f0:00:00:3c brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 172.17.33.250/24 brd 172.17.33.255 scope global noprefixroute eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::b3:f0ff:fe00:3c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

cloud-initで設定した固定アドレスがついている

```
[fedora@fedora-1 ~]$ ip r
default via 10.0.2.1 dev eth0 proto dhcp src 10.0.2.2 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.2 metric 100
172.17.33.0/24 dev eth1 proto kernel scope link src 172.17.33.250 metric 101
```


Red Hat


enp1s0
enp2s0
br-ex
br-int
veth
veth
br
veth
veth
eth0
veth
br
tap
br
tap
eth0
eth1
VM
Pod
Node

ライブマイグレーション

- 仮想ディスクのPVのアクセスモードはRWXであるが必須
  - RWXと言えばNFSですが… パフォーマンスの観点からはブロックストレージが推奨
  - RWXなブロックストレージが利用できるかはストレージ(CSIドライバ) 次第
- カスタムリソースVirtualMachineにおける、Podネットワークへ接続設定はmasqueradeにする
- PodのIPアドレスに注意が必要
  - デフォルトCNI側のIPアドレスはライブマイグレーションすると変わる
  - セカンダリCNI側のIPアドレスをCNIのIPAMから払い出すと…たぶん移行後はアドレスが変わる
  - セカンダリCNI側のIPアドレスをcloud-initから固定でつけておくと、移行後も同じアドレスでアクセスできる
  -

OVN-Kubernetesを
セカンダリCNIとして使う

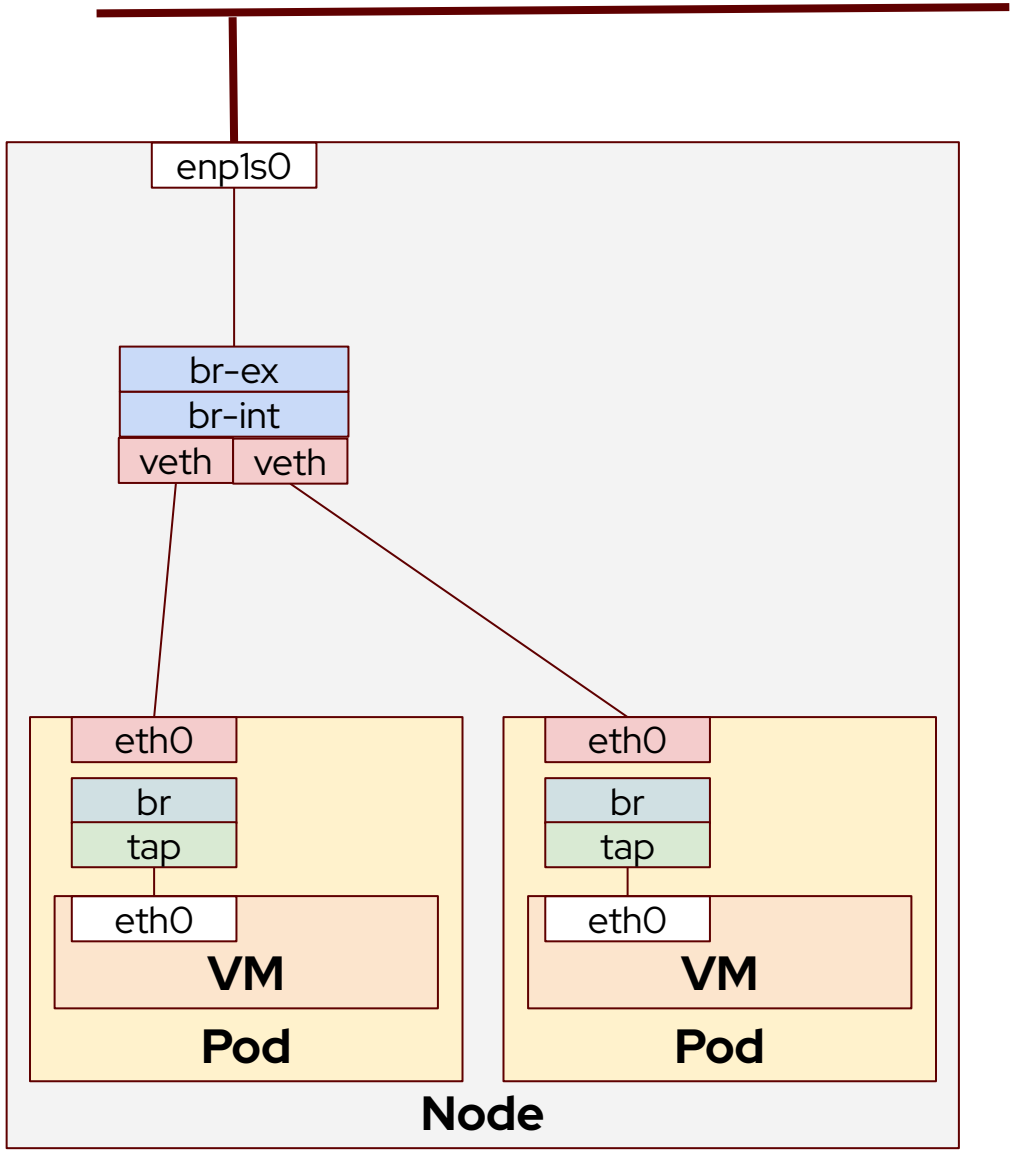
※ ONICのセッションではお話していない内容です

Red Hat

OVN-KubernetesをセカンダリCNIとして使用する

- OVN-Kubernetes:
  - 主にOpen vSwitchのコミュニティで開発しているCNIプラグイン
  - OVNをSDNコアとして使用
  - 主なユーザー: Red Hat, NVIDIA
- OVN-Kubernetesは、Multusのセカンダリ CNIとして使うことが可能
  - セカンダリCNIとして使う場合、2つのモードがある
    - layer2
    - localnet

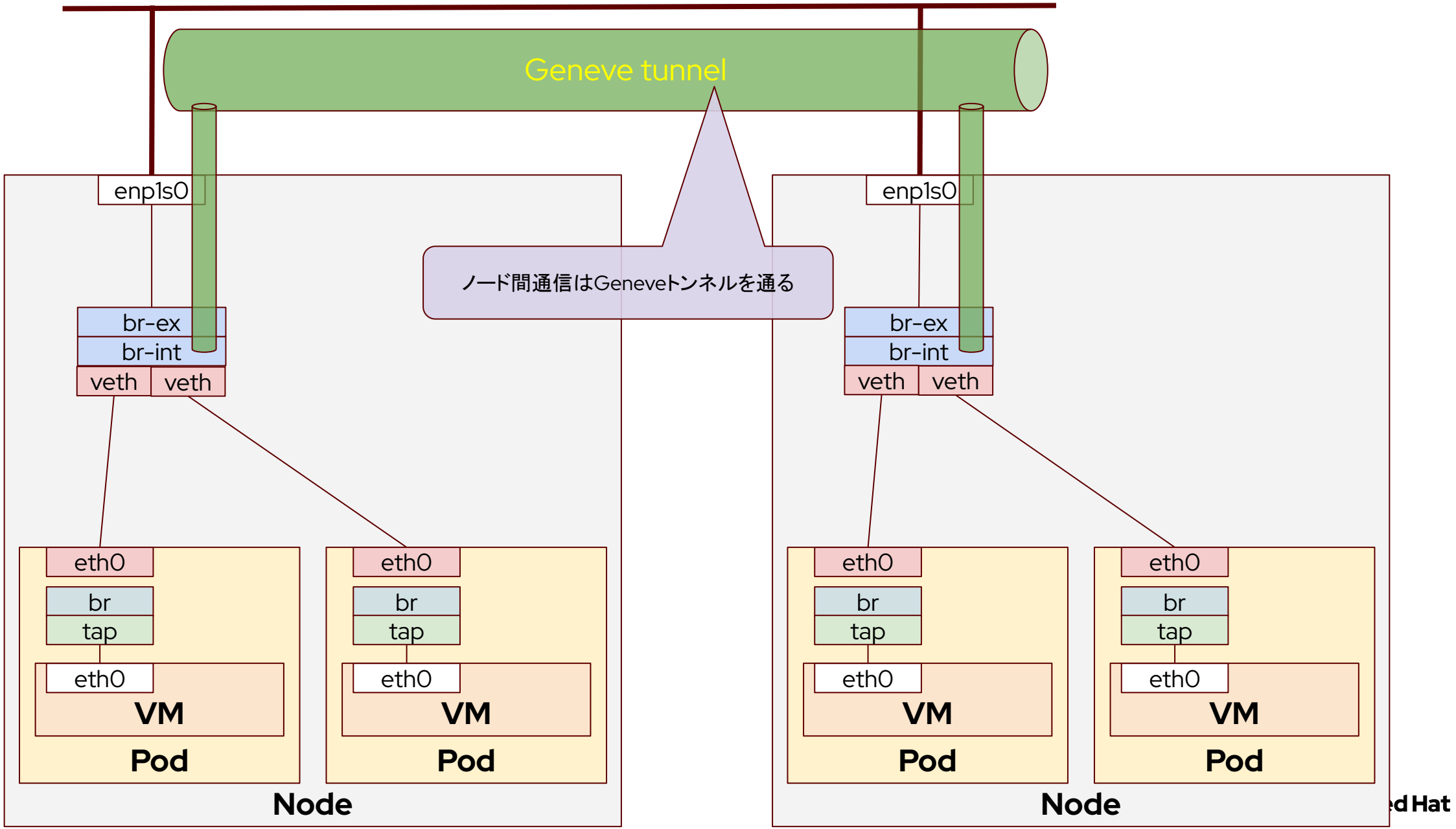
おさらい: プライマリCNIとしてのOVN-Kubernetes



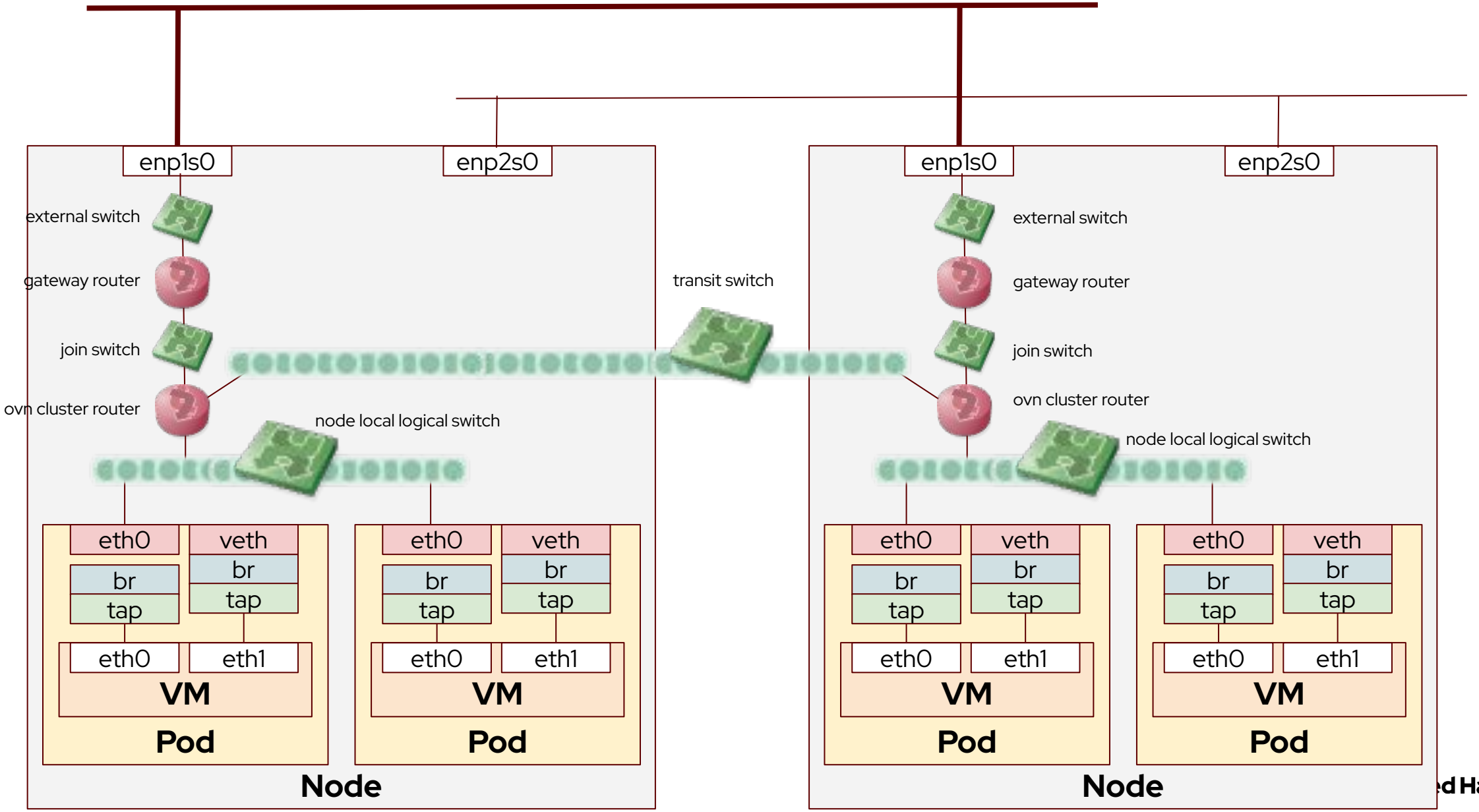
- 2つのOVSブリッジ: br-int, br-ex
- Podはbr-intにvethで接続

Red Hat

おさらい: プライマリCNIとしてのOVN-Kubernetes



おさらい: OVN-Kubernetesのオーバーレイネットワーク (プライマリCNI)



OVN-Kubernetes セカンダリCNIの2つの動作モード

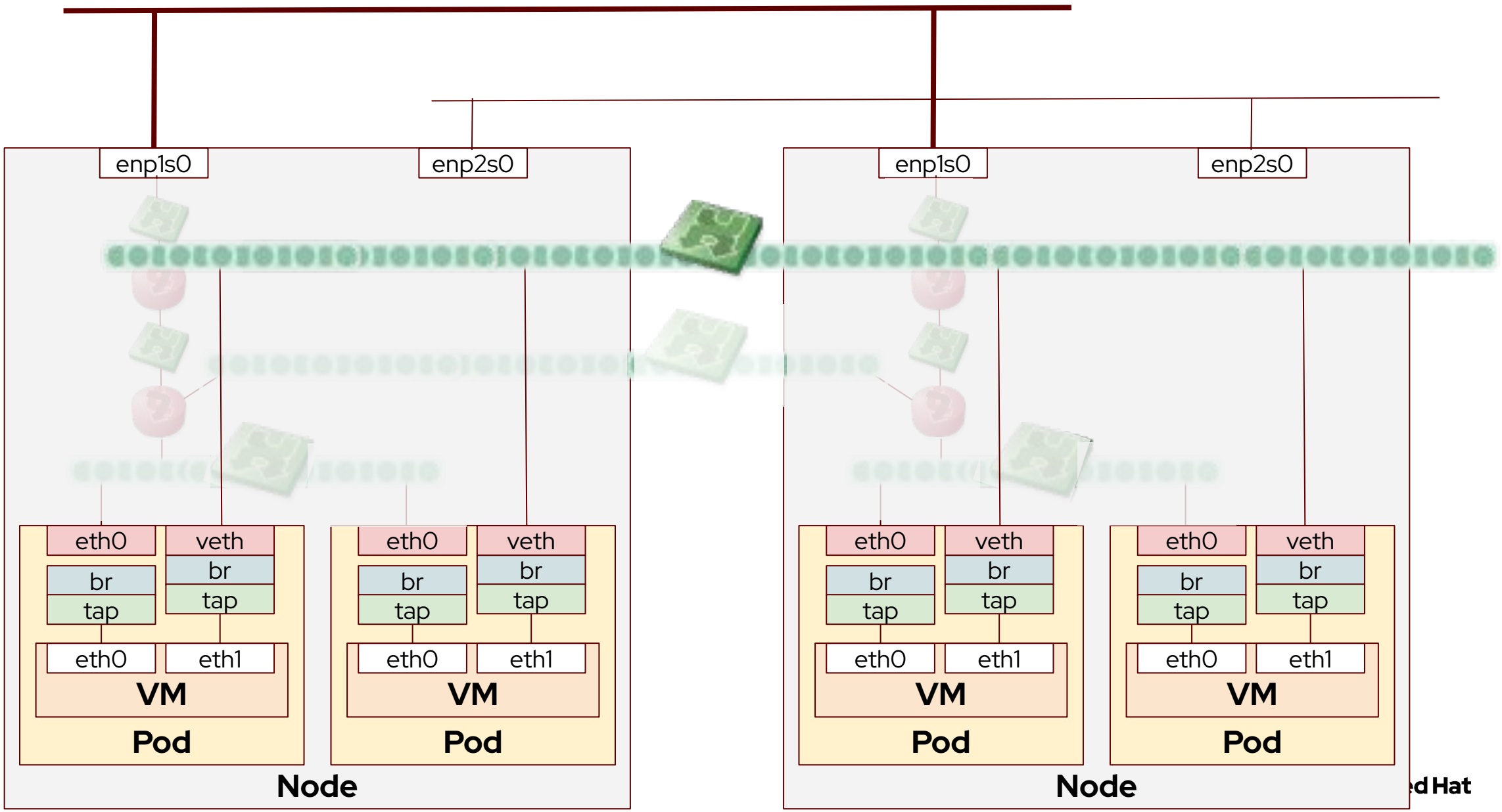
- layer2 モード
  - クラスターにまたがったオーバーレイL2ネットワークを構成
  - クラスター内に閉じており、外部(アンダーレイ)には接続できない
  - ノードをまたがる通信はGeneveトンネルを通る
- localnet モード
  - クラスターにまたがったオーバーレイL2ネットワークを構成
  - OVSブリッジを介して外部(アンダーレイ)にVLAN接続可能
  - ノードをまたがる通信はアンダーレイVLANを通る

layer2モード

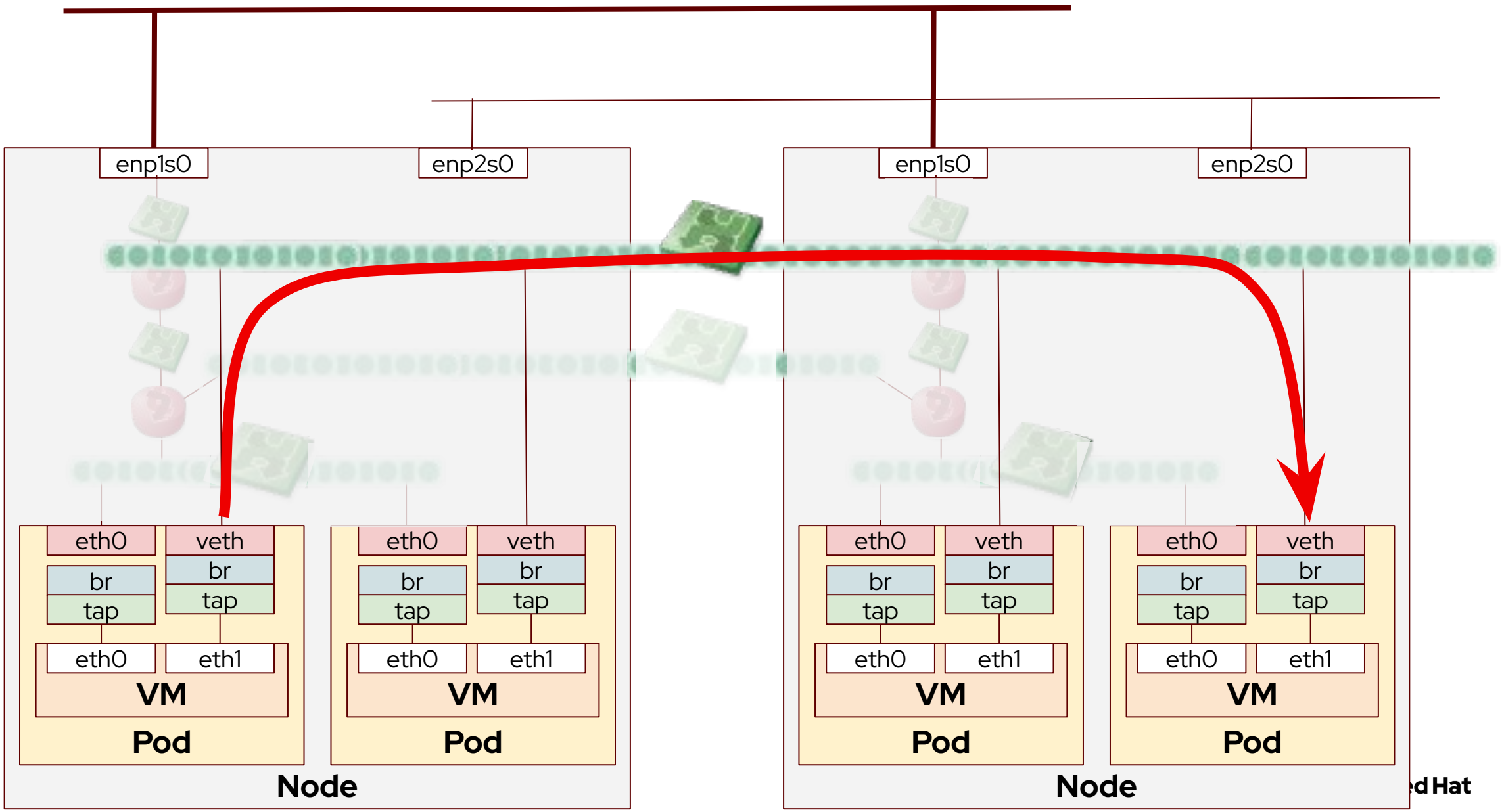
- クラスターにまたがったオーバーレイL2ネットワークを構成
- クラスター内に閉じており、外部(アンダーレイ)には接続できない
- ノードをまたがる通信はbr-intから出るGeneveトンネルを通る
- オーバーレイでL2ネットワークを作るだけ、なのでブリッジの指定とか NodeNetworkConfigurationPolicyの作成は必要なく、いきなりNetworkAttachDefinitionを作れば使える

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: nad-layer2
spec:
  config: |-
    {
      "cniVersion": "0.3.1",
      "name": "l2-network",
      "type": "ovn-k8s-cni-overlay",
      "topology":"layer2",
      "netAttachDefName": "test/nad-layer2"
    }
```

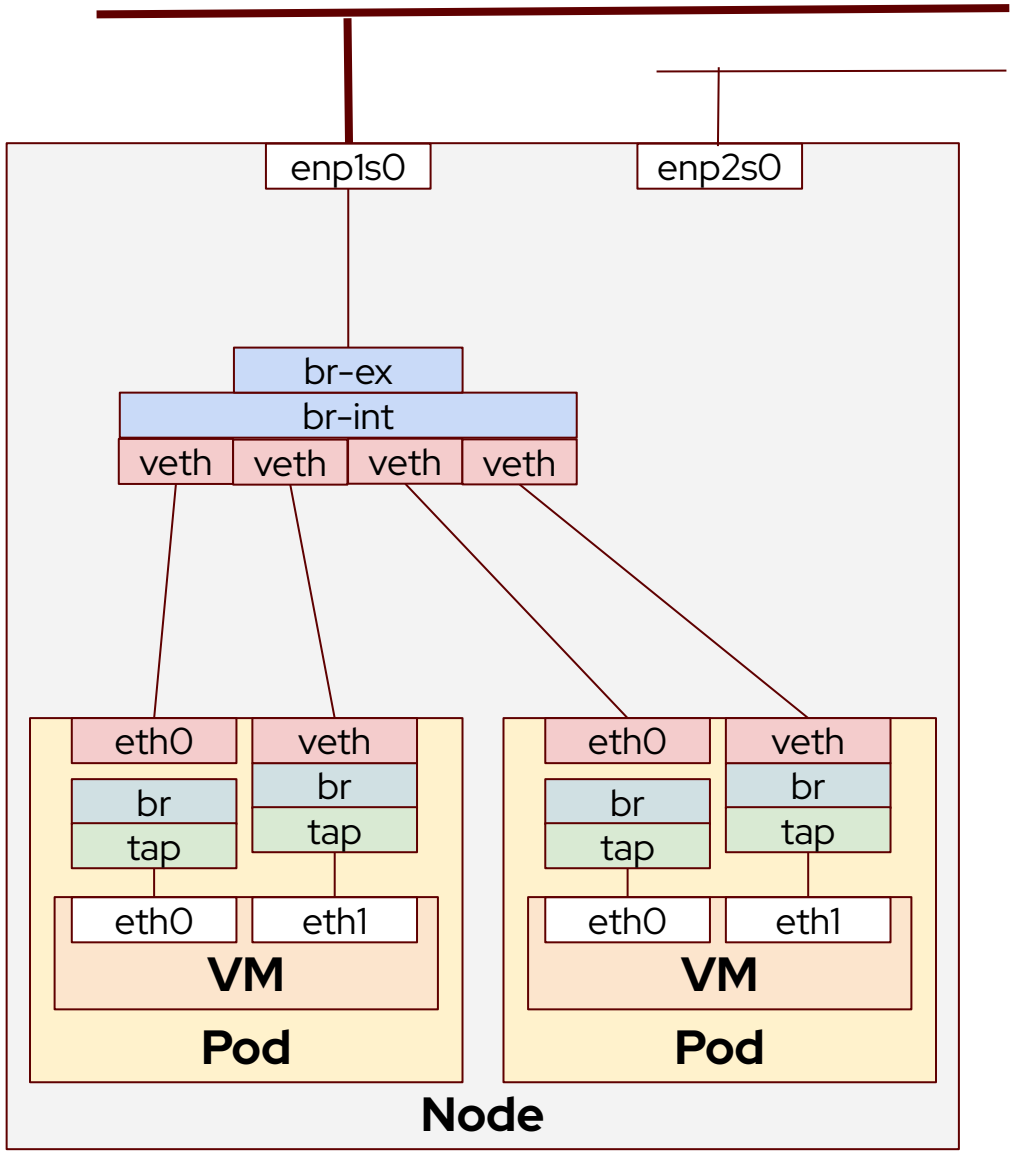
layer2モードのOVN-K追加ネットワーク (オーバーレイ)



layer2モードのOVN-K追加ネットワーク (オーバーレイ)



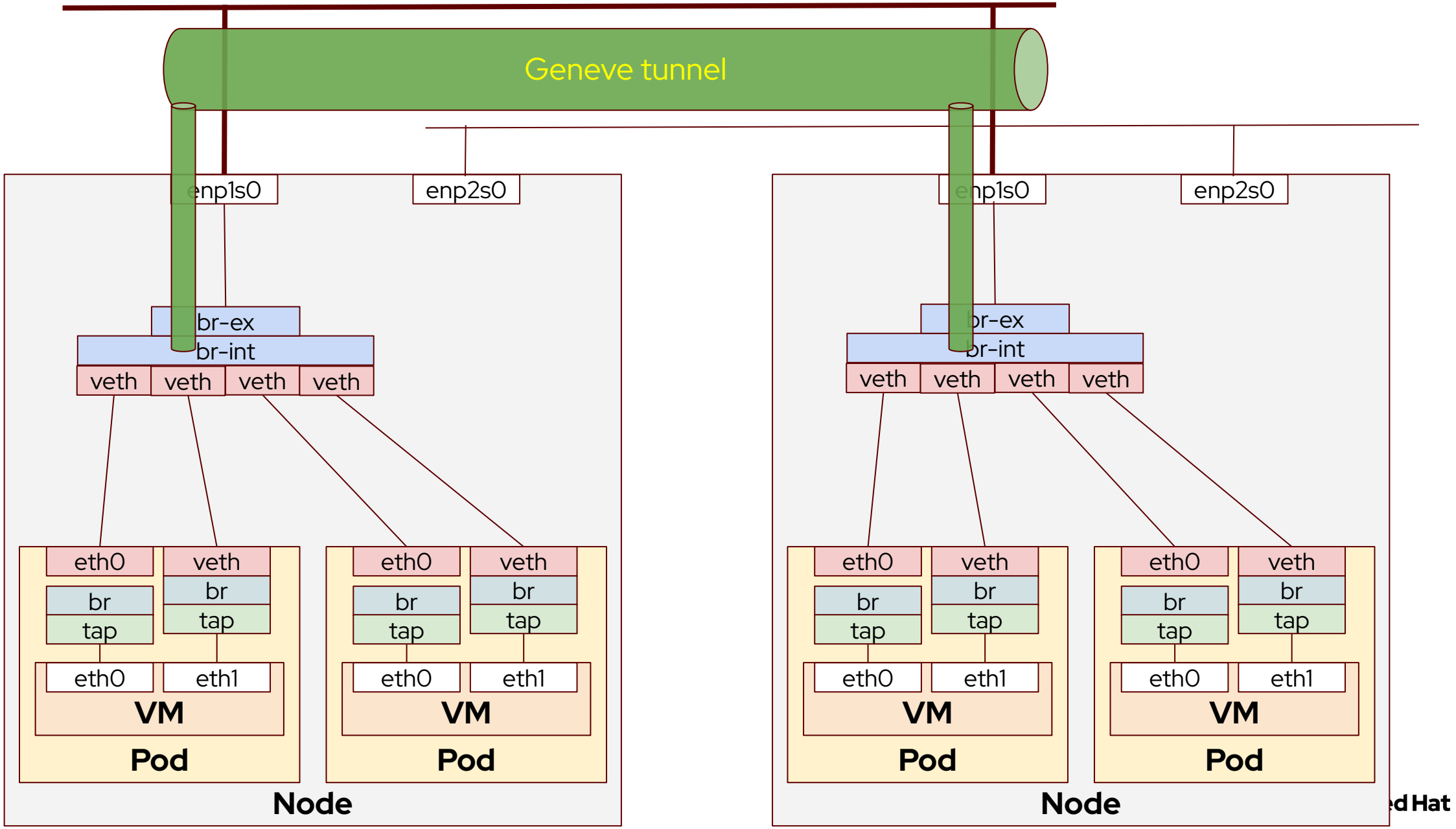
layer2モードのOVN-K追加ネットワーク (アンダーレイ)



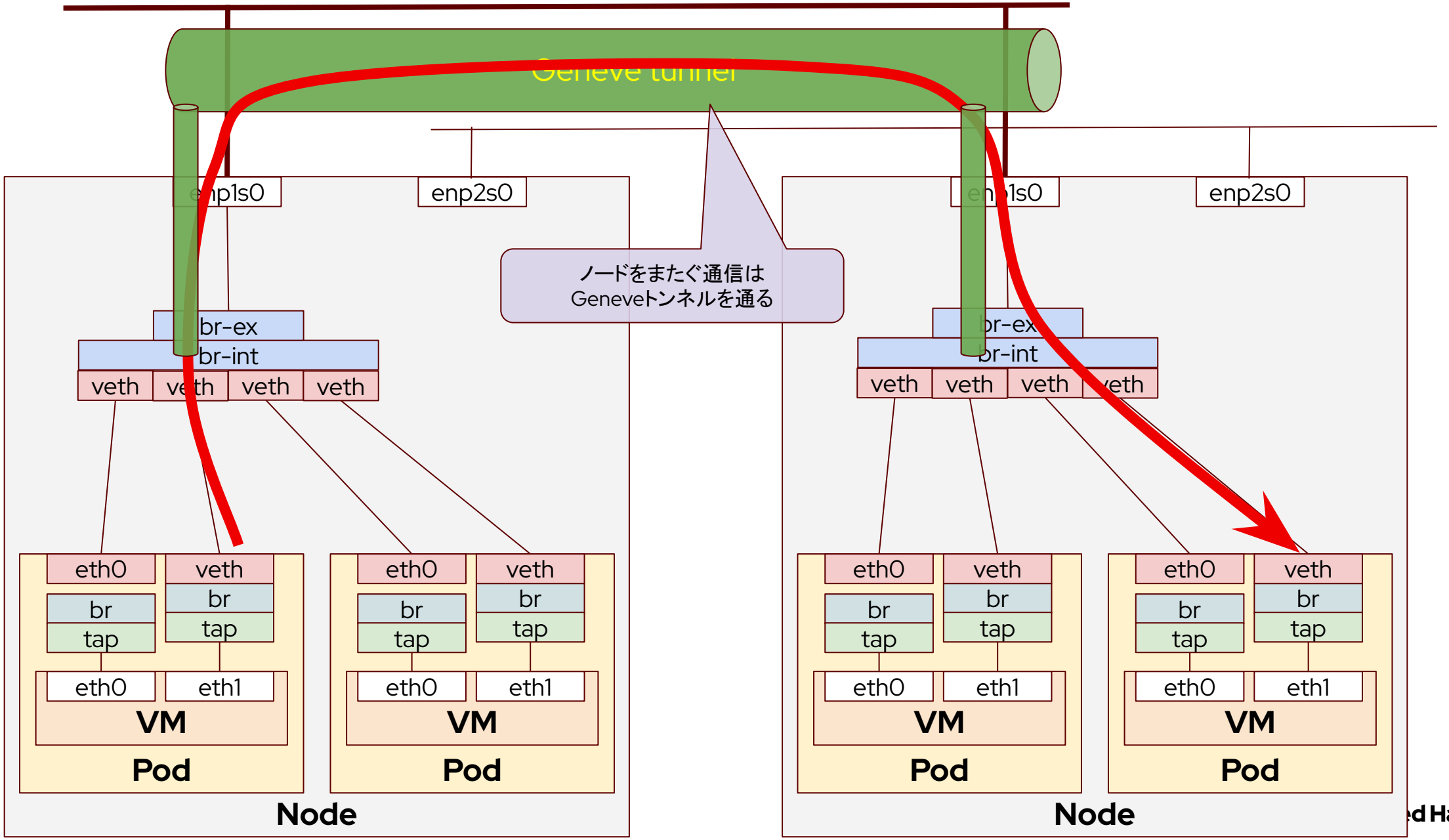
- アンダーレイとしては、Pod/VMの追加インターフェースもbr-intに接続する
- ノードをまたぐ通信は、Geneveトンネルを通る

Red Hat

layer2モードのOVN-K追加ネットワーク (アンダーレイ)



layer2モードのOVN-K追加ネットワーク (アンダーレイ)



localnetモード (br-ex共用)

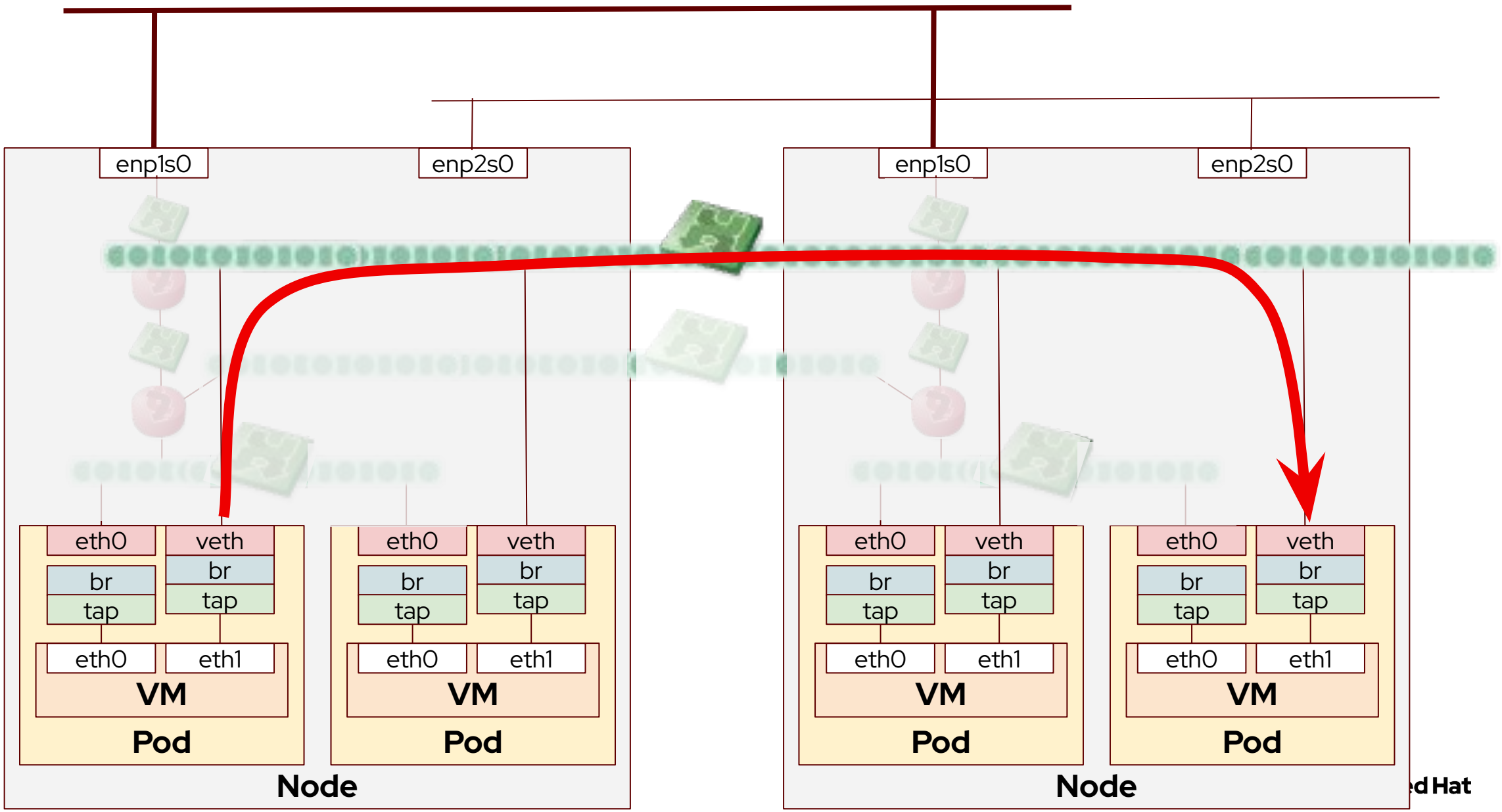
- (nmstate operatorのCRである)NodeNetworkConfigurationPolicyにおいて、bridgeとして `br-ex` を指定
- セカンダリCNI側でノードをまたいだ通信をする場合は、(Geneveトンネルではなく) `br-ex` につながる物理NICを使用
  - 外にでるときにNetworkAttachmentDefinitionで指定したVLAN IDを付与

    → 外部とVLAN通信が可能

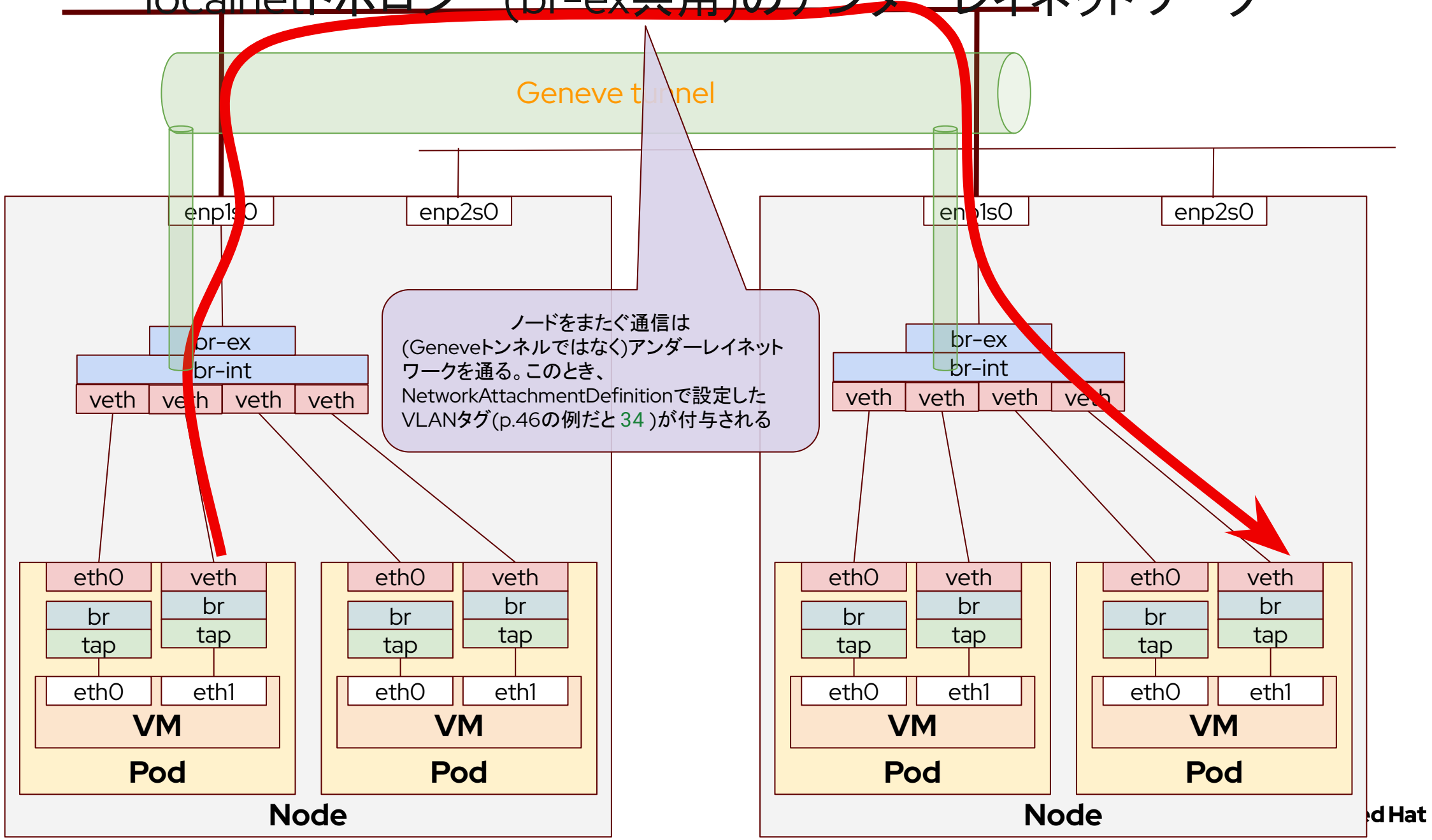
```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br-ex-multiple-networks
spec:
  nodeSelector:
    node-role.kubernetes.io/worker-virt: ''
  desiredState:
    ovn:
      bridge-mappings:
      - localnet: localnet1
        bridge: br-ex
        state: present
```

```
kind: NetworkAttachmentDefinition
metadata:
  name: nad-localnet1
spec:
  config: |-
    {
      "cniVersion": "0.3.1",
      "name": "localnet1",
      "type": "ovn-k8s-cni-overlay",
      "topology":"localnet",
      "vlanID": 34,
      "netAttachDefName": "test/nad-localnet1"
    }
```

localnetトポロジー(br-ex共用)のオーバーレイネットワーク



localnetトポロジー(br-ex共用)のアンダーレイネットワーク



localnetモード (別NIC使用)

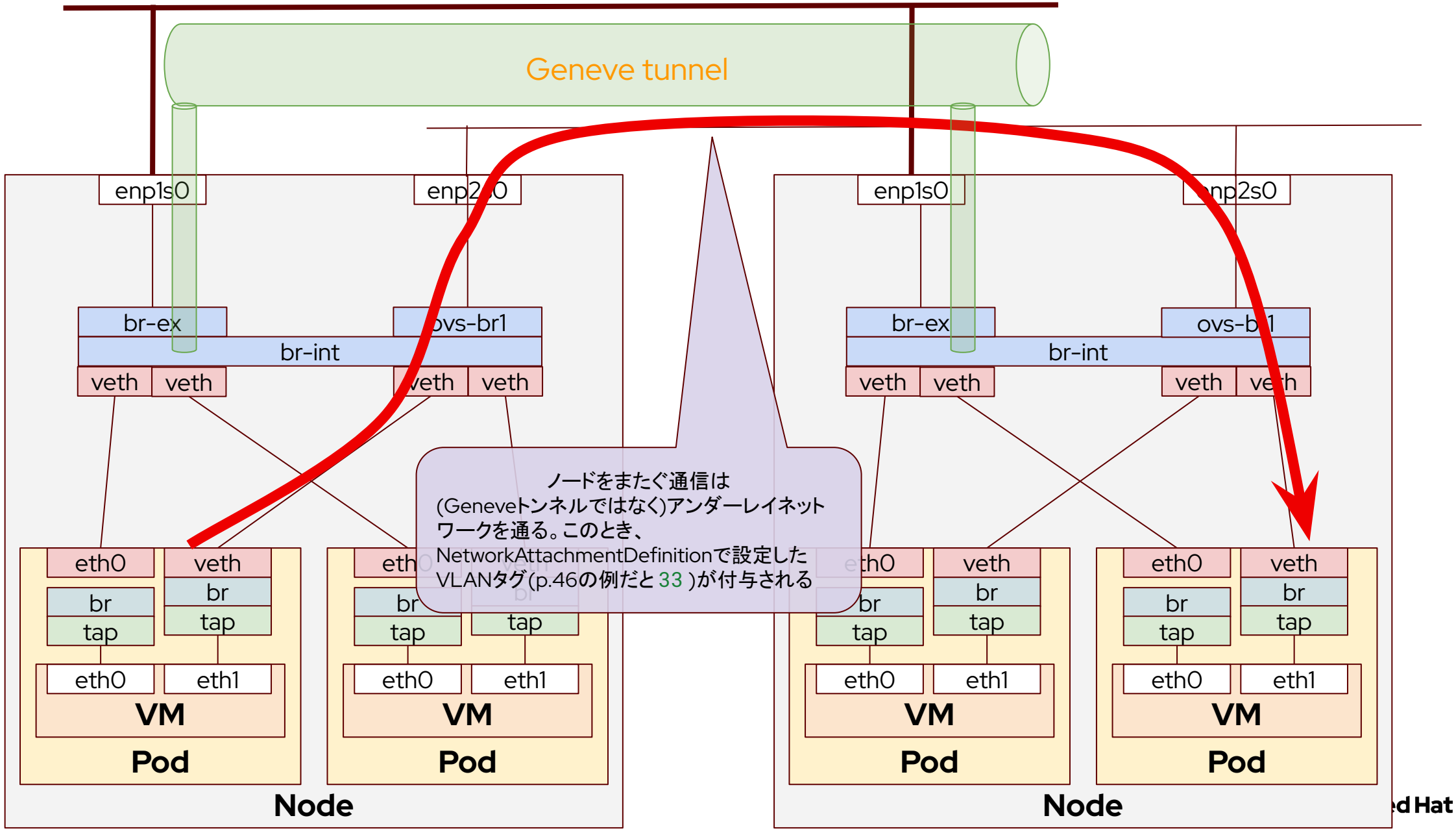
- (nmstate operatorのCRである)NodeNetworkConfigurationPolicyにおいて、bridgeとして `ovs-br1` を指定
  - 物理インターフェース `enp2s0` を `ovs-br1` に接続
- セカンダリCNI側でノードをまたいだ通信をする場合は、(Geneveトンネルではなく) `ovs-br1` につながる物理NIC `enp2s0` を使用
  - 外にでるときにNetworkAttachmentDefinitionで指定したVLAN IDを付与

    → 外部とVLAN通信が可能

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ovs-br1-multiple-networks
spec:
  nodeSelector:
    node-role.kubernetes.io/worker-virt: ''
  desiredState:
    interfaces:
    - name: ovs-br1
      type: ovs-bridge
      state: up
      bridge:
        options:
          stp: true
        port:
        - name: enp2s0
    ovn:
      bridge-mappings:
      - localnet: localnet2
        bridge: ovs-br1
        state: present
```

```
kind: NetworkAttachmentDefinition
metadata:
  name: nad-localnet2
spec:
  config: |-
    {
      "cniVersion": "0.3.1",
      "name": "localnet2",
      "type": "ovn-k8s-cni-overlay",
      "topology":"localnet",
      "vlanID": 33,
      "netAttachDefName": "test/nad-localnet2"
    }
```

localnetトポロジー(別NIC使用時)のアンダーレイネットワーク



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat