Cloud Native Contrail Networking ユーザガイド

Juniper Networks

Rev.1.0 / 2022/07/01

Juniper Networks | Driven by Experience™

CONFIDENTIALITY AND LEGAL NOTICE

This material contains information that is confidential and proprietary to Juniper Networks, Inc. Recipient may not distribute, copy, or repeat information in the document.

This statement of product direction sets forth Juniper Networks' current intention and is subject to change at any time without notice. No purchases are contingent upon Juniper Networks delivering any feature or functionality depicted in this presentation.

subject to a license agreement that describes program terms and conditions.

本資料は融資がベストエフォートで記載している資料となります。
内容に不備がある場合はご了承ください。
最新の状況などは公式のマニュアルをご確認ください。
また、内容は予告なしに変更になる場合があります。

技術詳細は公式ドキュメントを参照ください。
https://www.juniper.net/documentation/product/us/en/cloud-native-contrail-networking
各サンプルファイルはgithubも参照ください。
https://github.com/jnpr-jp-crdc/CN2

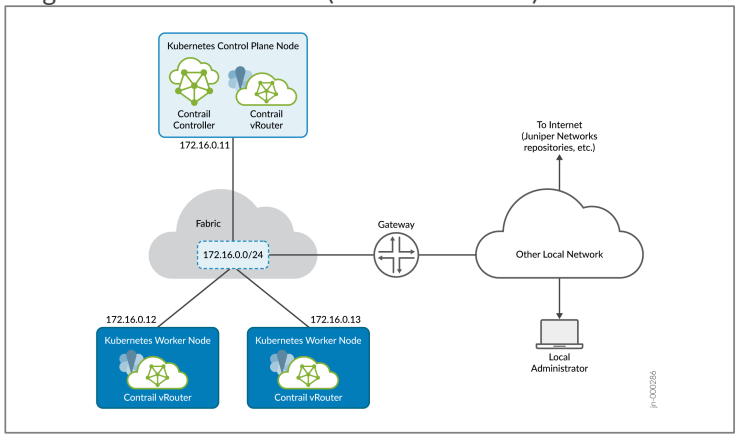
コンテンツ

0. CN2 Install

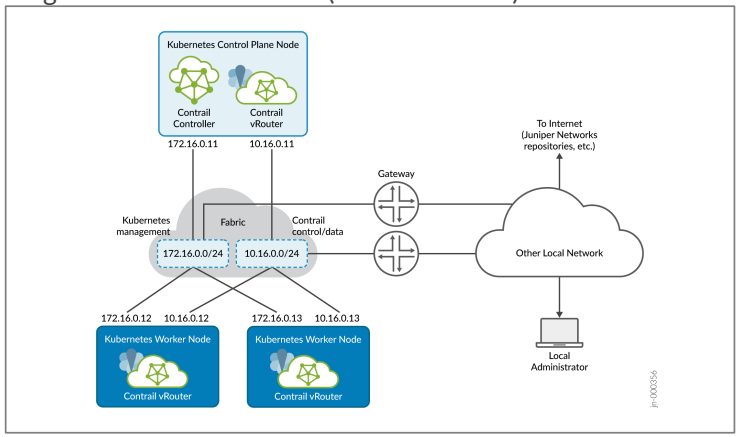
Contrail Install

- Contrailでは以下の3つの構成をサポートしています。
  - Single Kubernetes Cluster (Shared Network)
    - Kubernetes/Contrail共に共通のInterfaceを使用
  - Single Kubernetes Cluster (Multi Network)
    - KubernetesとContrailの通信を分離
  - Multi Kubernetes Cluster
    - 複数のKubernetes Clusterを1つのContrailで管理
    - Center ClusterにのみContrail Controllerをデプロイし、他Clusterにはデプロイしない
    - Center ClusterのContrail Controllerから他ClusterのvRouterを管理

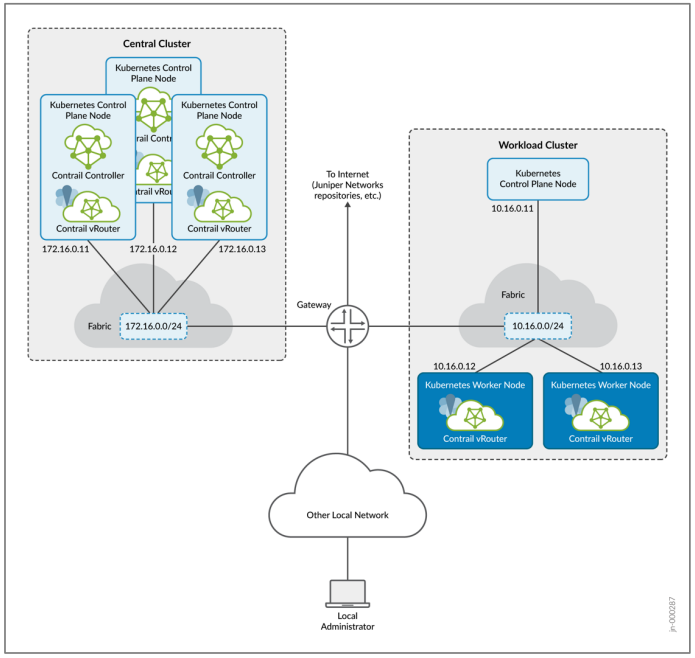
Single Kubernetes Cluster (Shared Network)



Single Kubernetes Cluster (Multi Network)



Multi Kubernetes Cluster



Contrail Install Steps

URL LinkはVersion 22.2リンクとなります
インストールするVersionに合ったドキュメントを参照ください

1. Free Trialサイトにアクセスし、hub.juniper.netのアカウント取得
   https://www.juniper.net/jp/ja/forms/cn2-free-trial.html

2. Kubernetesのインストール
   - Kubernetesのインストール方法に制限はありません
   - Kubernetesのインストールが初めての方は、以下を参考にしてください
     https://www.juniper.net/documentation/us/en/software/cn-cloud-native22/cn-cloud-native-k8s-install-and-lcm/topics/task/cn-cloud-native-k8s-create-kubernetes-cluster.html
     https://kubernetes.io/ja/docs/setup/production-environment/tools/kubeadm/
   - Contrail Requirements
     https://www.juniper.net/documentation/en_US/release-independent/contrail-cloud-native/topics/reference/cloud-native-contrail-supported-platforms.pdf
     https://www.juniper.net/documentation/us/en/software/cn-cloud-native22/cn-cloud-native-k8s-install-and-lcm/topics/reference/cn-cloud-native-system-requirements.html

3. 以下のサイトからContrail Deployment ManifestsのDownload
   - https://support.juniper.net/support/downloads/?p=contrail-networking
   - 権限のない方はJuniper担当者にお問い合わせください

4. Contrail インストール
   - “kubectl –f apply deployer.yaml”でContrailがインストールされます。
   - 詳細なステップは以下を参照してください
   https://www.juniper.net/documentation/us/en/software/cn-cloud-native22/cn-cloud-native-k8s-install-and-lcm/topics/concept/cn-cloud-native-k8s-install-overview.html

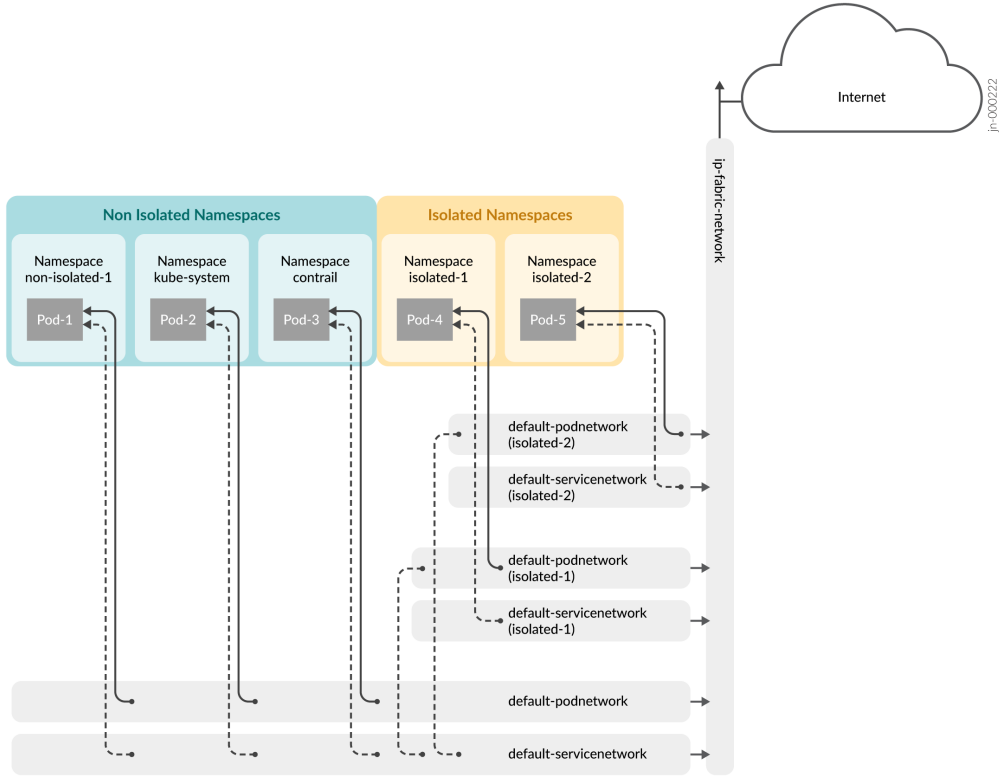
5. Contrail Analytics インストール (Option)
   https://www.juniper.net/documentation/us/en/software/cn-cloud-native22/cn-cloud-native-k8s-install-and-lcm/topics/topic-map/cn-cloud-native-k8s-install-contrail-analytics.html

1. Namespace

Namespace

Contrailでは2つのタイプのNamespaceを利用することが可能

- Non-Isolated Namespace
  - default-pod-network, default-service-networkはNamespace間で共有
  - Podはクラスタ内の他のPODとNATなしで接続が可能
  - Podは”IP Fabric Forwarding”, “Fabric Source NAT”機能を利用することでUnderlay Networkへ接続が可能
- Isolated Namespace
  - deault-pod-network, default-service-networkはNamespace毎に自動生成される
  - Podは同一Namespace内のPODのみ接続が可能
  - PodはNon-Isolated NamespaceのServiceにアクセス可能
  - Pod/ServiceのIP Addressはクラスタの Pod/Service Subnetと同じSubnetが使用される
  - Podは”IP Fabric Forwarding”, “Fabric Source NAT”機能を利用することでUnderlay Networkへ接続が可能
  - “Virtual Network Router(VNR)”機能を利用することでNamespace間接続が可能



Namespace - Isolated Namespace

Sample: Isolated Namespace

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
```

<-- Namespace名

<-- Isolated Namespaceを有効化

Sample: Isolated Namespace with IP fabric Forwarding / Fabric SNAT

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
  annotations:
    core.juniper.net/forwarding-mode: "ip-fabric | fabric-snat"
```

<-- Namespace名

<-- Isolated Namespaceを有効化

<-- “ip-fabric”はNATなしでUnderlayへ接続。”fabric-snat”はWorkerNodeのIPでSNATされUnderlayへ接続
本設定はdefault-pod-networkに対してのみ有効

Namespace - Isolated Namespace

新規作成したNameSpaceにdefault-podnetwork, default-servicenetworkが作成される

```
# kubectl get vn -A
NAMESPACE                                                NAME                    VNI   IP FAMILIES   STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-podnetwork      2     v6,v4         Success   78m
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-servicenetwork  4     v6,v4         Success   78m
contrail                                                 ip-fabric               3                   Success   78m
contrail                                                 link-local              1                   Success   78m
ns1                                                      default-podnetwork      5     v6,v4         Success   10m
ns1                                                      default-servicenetwork  6     v6,v4         Success   10m
```

cluster namespaceのsubnetが使用されるため、Isolated Namespace毎にsubnetは作成されない

```
# kubectl get subnet -A
NAMESPACE                                                NAME                              CIDR                          USAGE   STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-podnetwork-pod-v4-subnet      10.233.64.0/18                0.15%   Success   99m
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-podnetwork-pod-v6-subnet      fd85:ee78:d8a6:8607::1:0/112  0.04%   Success   99m
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-servicenetwork-pod-v4-subnet  10.233.0.0/18                 0.12%   Success   99m
contrail-k8s-kubemanager-cluster1-juniper-local-contrail   default-servicenetwork-pod-v6-subnet  fd85:ee78:d8a6:8607::1000/116 0.07%   Success   99m
```

Namespace – POD作成

Sample: Isolated Namespace内のPOD作成

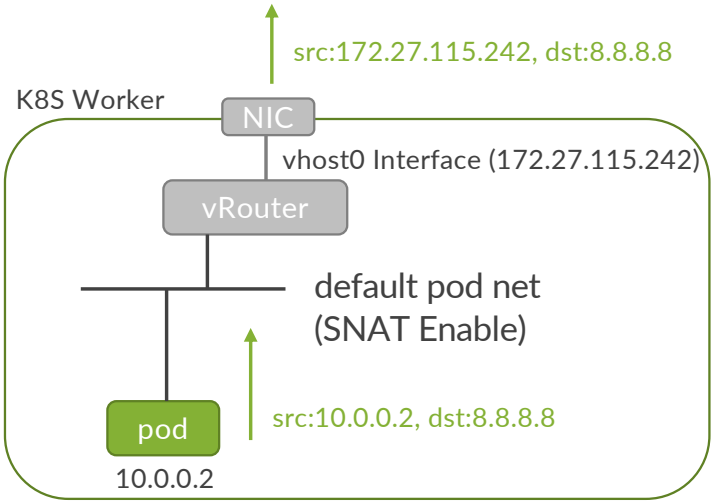
```
apiVersion: v1
kind: Pod
metadata:
  name: centos1
  namespace: ns1
spec:
  containers:
  - name: centos1
    image: centos
    command: [ "/bin/bash", "-c" ]
    args:
      - sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

default-podnetworkに接続される

```
# kubectl describe pod centos1 -n ns1
Name:         centos1
Namespace:    ns1
Priority:     0
Node:         cn2-worker1/172.27.115.242
Start Time:   Mon, 20 Jun 2022 01:32:56 +0000
Labels:       <none>
Annotations:  kube-manager.juniper.net/vm-uuid: 2b940bcb-41e2-4b70-934f-f29b06023966
Status:       Running
IP:           10.233.65.8
IPs:
  IP:  10.233.65.8
  IP:  fd85:ee78:d8a6:8607::1:108
```

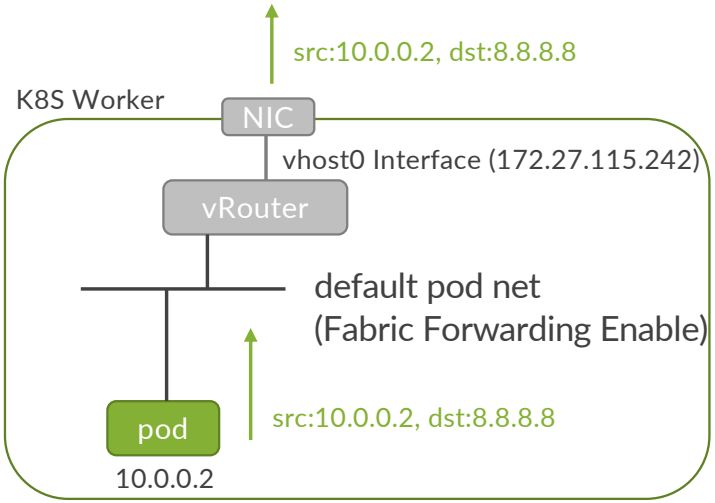
Namespace– Fabric SNAT

- Fabric SNATをtrueにセットした場合、PODから外部への通信はPODが稼働するWorkerNodeのInterface IP(vhost0 Interface IP)でSNATされる
- SNATのため、外部からPODへの接続は不可



Namespace – Fabric Forwarding

- Fabric Forwardingをtrueにセットした場合、PODから外部への通信はそのままUnderlayへ接続される
- Contrail Control Nodeと外部ルータ間でBGP接続をしている場合、PODのホストルートがBGP/inet unicastにより外部ルータへAdvertiseされる
  - 詳細は9. 外部ルータ接続　参照

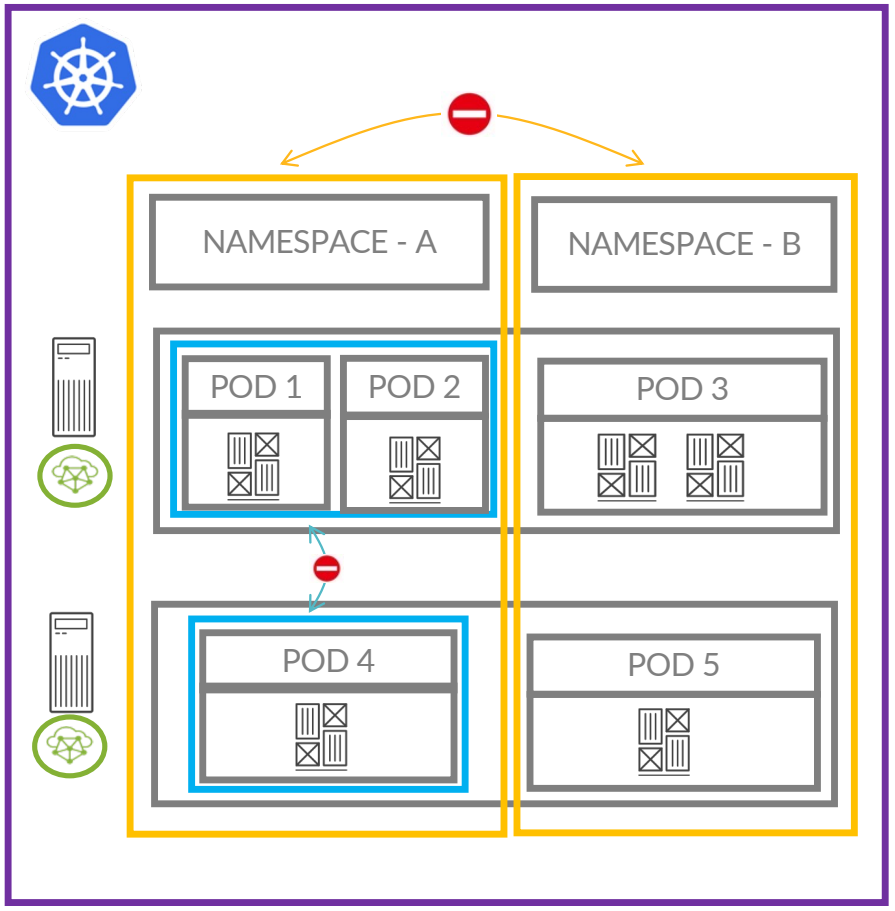


2. Virtual Network

Virtual Network

- default-podnetwork, default-servicenetwork以外にNamespace内に1つ以上のVirtual Networkを作成可能
- Virtual NetworkはRoutingInstance(VRF)が分かれており、Virtual Network間の接続は不可
- VNR(後述) を使用することでVirtual Network間接続が可能
- Virtual Networkに紐づくSubnet(IPv4/IPv6)を作成
- Virtual Networkには以下の3つのForwardingModeがある
  - l2_l3 : IP FIB, MAC FIBをLookup (Default Mode)
  - l2 : MAC FIBのみLookup
  - l3 : IP FIBのみLookup
- Default-podnetwork同様にIP Fabric Forwarding, Fabric SNATの設定が可能

デフォルトの分割単位
ネームスペース単位の分割
仮想ネットワーク単位の分割



Virtual Network – NAD(Network Attachment Definition)

Sample: IPv4 NAD

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: nad1
  namespace: ns1
  annotations:
    juniper.net/networks: '{
      "ipamV4Subnet": "10.0.0.0/24",
      "ipamV6Subnet": "2001:db8::/64",
      "routeTargetList": ["target:64512:1"],
      "importRouteTargetList": ["target:10.2.2.2:561"],
      "exportRouteTargetList": ["target:10.1.1.1:561"],
      "fabricSNAT": true
    }'
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": ”nad1",
    "type": "contrail-k8s-cni"
  }'
```

← Optional
← Optional
← Optional
← Optional
← Optional
← Optional

NADを作成することでVirtual NetworkとSubnetが生成される

```
# kubectl get vn -n ns1
NAME                      VNI   IP FAMILIES   STATE     AGE
default-podnetwork        5     v6,v4         Success   3d20h
default-servicenetwork    6     v6,v4         Success   3d20h
nad1                      9     v6,v4         Success   11s

# kubectl get subnet -n ns1
NAME      CIDR             USAGE   STATE     AGE
nad1-v4   10.0.0.0/24      1.17%   Success   15s
nad1-v6   2001:db8::/64    0.00%   Success   15s
```

Virtual Network – POD作成

Sample: Isolated Namespace内のvn1にPOD作成

```
apiVersion: v1
kind: Pod
metadata:
  name: centos1
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: ns1/vn1
spec:
  containers:
  - name: centos1
    image: centos
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

← 接続するVirtual Networkを指定

← PODはMulti-Interfaceとなるため、Virtual Network側へのdefault routeを指定

Virtual Network – POD作成

PODはdefault-podnetworkとvn1に接続される

```
# kubectl describe pod centos1 -n ns1
Name:         centos1
Namespace:    ns1
Priority:     0
Node:         cn2-worker1/172.27.115.242
Start Time:   Mon, 20 Jun 2022 01:44:34 +0000
Labels:       <none>
Annotations:  k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "ns1/default-podnetwork",
                    "interface": "eth0",
                    "ips": [
                        "10.233.65.8",
                        "fd85:ee78:d8a6:8607::1:108"
                    ],
                    "mac": "02:92:8e:f7:8c:40",
                    "default": true,
                    "dns": {}
                },{
                    "name": "ns1/nad1",
                    "interface": "eth1",
                    "ips": [
                        "10.0.0.2",
                        "2001:db8::2"
                    ],
                    "mac": "02:34:d0:6c:e7:0b",
                    "dns": {}
                }]
              k8s.v1.cni.cncf.io/networks: ns1/nad1
              kube-manager.juniper.net/vm-uuid: 8a08bdf0-5bec-40cb-8e25-
c1921e351c93
Status:       Running
IP:           10.233.65.8
IPs:
  IP:  10.233.65.8
  IP:  fd85:ee78:d8a6:8607::1:108
```

Virtual Network – POD作成 – Multi Virtual Net

Sample: vn1, vn2に接続するPOD作成

```
apiVersion: v1
kind: Pod
metadata:
  name: centos1
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: ns1/vn1, ns1/vn2
spec:
  containers:
  - name: centos1
    image: centos
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

```
# kubectl describe pod centos1 -n ns1
Name:         centos1
Namespace:    ns1
Priority:     0
Node:         cn2-worker1/172.27.115.242
Start Time:   Wed, 22 Jun 2022 00:35:11 +0000
Labels:       <none>
Annotations:  k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "ns1/default-podnetwork",
                    "interface": "eth0",
                    "ips": [
                        "10.233.65.8",
                        "fd85:ee78:d8a6:8607::1:108"
                    ],
                    "mac": "02:f1:0f:c3:ef:17",
                    "default": true,
                    "dns": {}
                },{
                    "name": "ns1/vn1",
                    "interface": "eth1",
                    "ips": [
                        "10.0.0.2"
                    ],
                    "mac": "02:73:fa:01:d7:a3",
                    "dns": {}
                },{
                    "name": "ns1/vn2",
                    "interface": "eth2",
                    "ips": [
                        "20.0.0.2"
                    ],
                    "mac": "02:b2:0f:c0:c6:fb",
                    "dns": {}
                }]
              k8s.v1.cni.cncf.io/networks: ns1/vn1,
ns1/vn2
              kube-manager.juniper.net/vm-uuid:
fbf38889-0e83-4c37-8e59-29e7672a53c9
Status:       Running
IP:           10.233.65.8
IPs:
  IP:  10.233.65.8
  IP:  fd85:ee78:d8a6:8607::1:108
```

Virtual Network – POD作成 – その他Network Option

Sample Yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: centos1
  namespace: ns1
k8s.v1.cni.cncf.io/networks: |-
  [
    {
      "name": "vn1",
      "namespace": "ns1",
      "cni-args": null,
      "ips": ["10.0.0.100"],
      "mac": "de:ad:00:00:be:ef",
      "interface": "tap1”
    },
    {
      "name": "vn2",
      "namespace": "ns1",
      "cni-args": null,
      "ips": ["20.0.0.100"],
      "mac": "de:ad:00:00:be:ee",
      "interface": "tap2"
    }
  ]
```

← PODのIP指定
← PODのMAC指定

```
spec:
  containers:
  - name: centos1
    image: centos
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

Virtual Network – Subnet IPv4

- Network Attachment Definition(NAD)以外にSubnet, Virtual Networkを個別に設定しPODにアタッチ可能
- Subnet, Virtual NetworkではNADより細かいパラメータを指定可能

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet1-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet1-v4
spec:
  cidr: 10.0.0.0/24
  defaultGateway: 10.0.0.1                          ← PODに付与するIP Range
  dnsNameservers: [8.8.8.8]
  bgpaasPrimaryIP: 10.0.0.2                         ← BGPaaSで使用するIP(後述)
  bgpaasSecondaryIP: 10.0.0.3
  disableBGPaaSIPAutoAllocation: true | false       ← trueにセットした場合、bgpaasPrimaryIP, bgpaasSecondaryIPは必須
  fqName: <string>
```

Virtual Network – Subnet IPv6

- Network Attachment Definition(NAD)以外にSubnet, Virtual Networkを個別に設定しPODにアタッチ可能
- Subnet, Virtual NetworkではNADより細かいパラメータを指定可能

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet1-v6
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet1-v6
spec:
  cidr: 2001:db8::/64
  defaultGateway: 2001:db8::1                          ← PODに付与するIP Range
  dnsNameservers: [2001:db8::8:8:8:8]
  bgpaasPrimaryIP: 2001:db8::2                         ← BGPaaSで使用するIP(後述)
  bgpaasSecondaryIP: 2001:db8::3
  disableBGPaaSIPAutoAllocation: true | false          ← trueにセットした場合、bgpaasPrimaryIP, bgpaasSecondaryIPは必須
  fqName: <string>
```

Virtual Network

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1-v4
  v6SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1-v6
  routeTargetList:
    - target:23:4561
    - target:21:7000
    - target:871:6540
  importRouteTargetList:
    - target:10.2.2.2:561
    - target:97:651
  exportRouteTargetList:
    - target:10.1.1.1:561
    - target:97:651
  virtualNetworkProperties:
    forwardingMode: l2_l3
    rpf: disable
  fabricForwarding: false
  fabricSNAT: false
  isProviderNetwork: false
```

- Virtual Networkを作成し、Subnetを紐づける

← IPv4 Subnetを指定

← IPv6 Subnetを指定
← Option

← Option

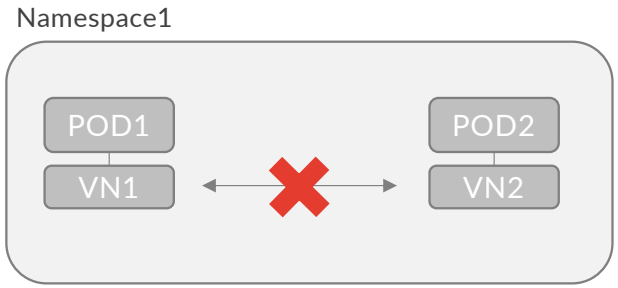
← Option

← Option
← Option: l2 or l3 or l2_l3(default)
← Option (ReversePathForwarding) Default: disable (true: SourceIPのRouteTable Validationチェックが有効)
← Option default false
← Option default false
← Option default false (trueにした場合、後から変更不可)

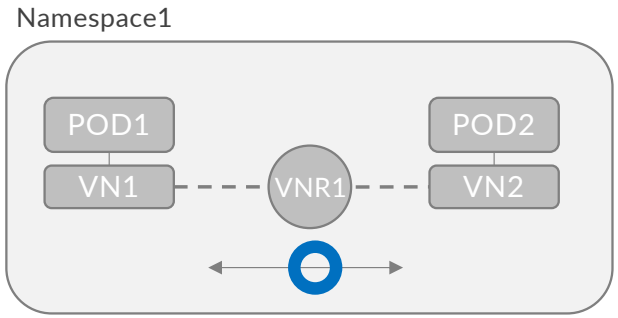
3. Virtual Network Router

Virtual Network Router (VNR)

- VNRはRouteLeakによりVirtual Network間を接続
- 1つのVNRに複数のVirtual Networkを接続可能
- Virtual Networkは複数のVNRに接続可能
- VNR Type
  - Mesh:
    - 全てのPODは互いに通信可能
  - Hub:
    - Spoke/Hub間のVNRは互いに通信可能
    - Type Spokeとの間でRouteLeakされる
    - Hub内のVirtual Netowrk間は通信不可
  - Spoke:
    - Spoke/Hub間のVNRは互いに通信可能
    - Type Spoke同士はRouteLeakされない
    - Spoke内のVirtual Network間は通信不可



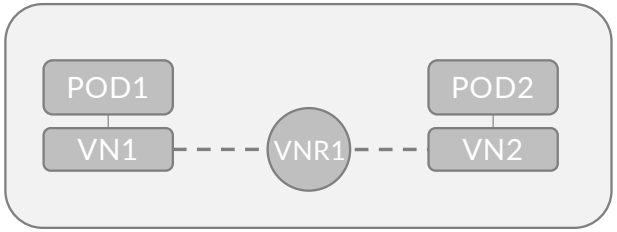
DefaultではVirtual Network間の通信は不可



VNRに接続することによりVirtual Network間接続が可能

Virtual Network Router (VNR) – Mesh – Case1

Namespace1



- 同一Namespace内にあるVN1, VN2をVNR1(Mesh)にて接続
- Virtual Networkに設定したLabelにマッチした経路をImport

Sample Yaml

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

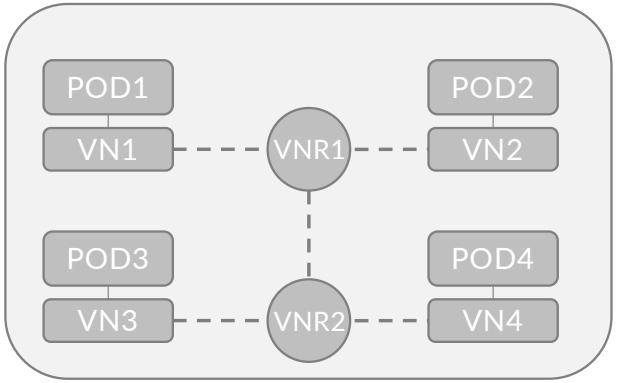
Virtual Network Router (VNR) – Mesh – Case1

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
      vnr: web
spec:
  type: mesh
  virtualNetworkSelector:
  matchExpressions:
    - key: vn
      operator: In
      values: [web, db]
```

Virtual Network Router (VNR) – Mesh – Case2

Namespace1



- 同一Namespace内で異なるVNRに接続されているVirtual Network間を接続
- VNRにてVNRに設定したLabelをImportすることでVNR間を接続

Sample Yaml

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

Virtual Network Router (VNR) – Mesh – Case2

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
    vnr: web
spec:
  type: mesh
  virtualNetworkSelector:
    matchLabels:
      vn: web
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnr: db
---
```

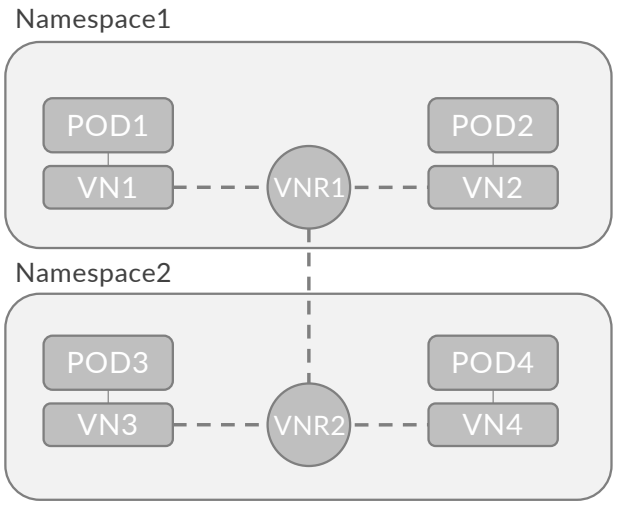
← VNR2のLabel(vnr: db)をVNR1にImport

Virtual Network Router (VNR) – Mesh – Case2

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr2
  annotations:
    core.juniper.net/display-name: vnr2
  labels:
      vnr: db
spec:
  type: mesh
  virtualNetworkSelector:
    matchLabels:
      vn: db
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnr: web
---
```

← VNR1のLabel(vnr: web)
をVNR2にImport

Virtual Network Router (VNR) – Mesh – Case3



- 異なるNamespaceで異なるVNRに接続されているVirtual Network間を接続
- VNRにてNamespaceに設定したLabelをImportすることで異なるNamespaceのVNR間を接続

Sample Yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
    ns: ns1
---
apiVersion: v1
kind: Namespace
metadata:
  name: ns2
  labels:
    core.juniper.net/isolated-namespace: "true"
    ns: ns2
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
```

Virtual Network Router (VNR) – Mesh – Case3

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns2
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns2
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns2
    name: subnet2
---
```

Virtual Network Router (VNR) – Mesh – Case3

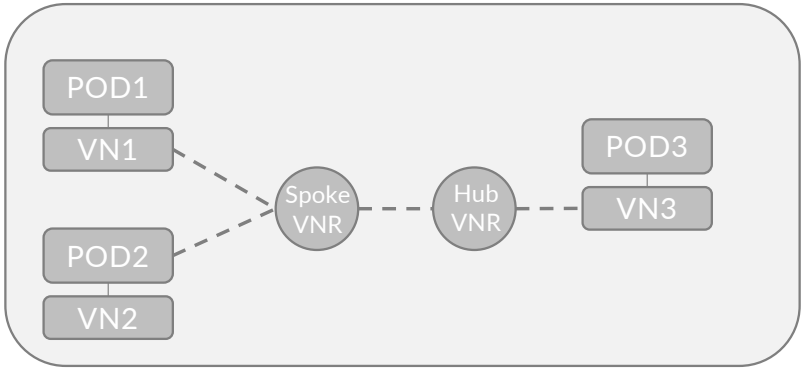
```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
    vnr: web
spec:
  type: mesh
  virtualNetworkSelector:
    matchLabels:
      vn: web
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnr: db
        namespaceSelector:
          matchLabels:
            ns: ns2
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns2
  name: vnr2
  annotations:
    core.juniper.net/display-name: vnr2
  labels:
    vnr: db
spec:
  type: mesh
  virtualNetworkSelector:
    matchLabels:
      vn: db
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnr: web
        namespaceSelector:
          matchLabels:
            ns: ns1
---
```

← Namespaceに設定したlabelをImport

Virtual Network Router (VNR) – Hub&Spoke – Case1

Namespace1



- 同一Namespace内でHub&Spoke
- VN1, VN2はSpoke VNRに接続、VN3はHub VNRに接続
- VN1 / VN3間及び VN2 / VN3間は互いに通信可能
- VN1 / VN2間は通信不可

Sample Yaml

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

Virtual Network Router (VNR) – Hub&Spoke – Case1

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2
---
```

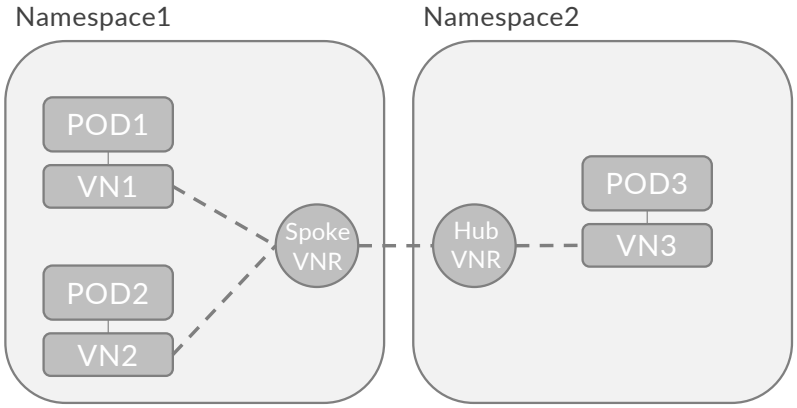
```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet3
  annotations:
    core.juniper.net/display-name: subnet3
spec:
  cidr: "30.0.0.0/24"
  defaultGateway: 30.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn3
  annotations:
    core.juniper.net/display-name: vn3
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet3
---
```

Virtual Network Router (VNR) – Hub&Spoke – Case1

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
    vnrgroup: spokes
spec:
  type: spoke
  virtualNetworkSelector:
    matchLabels:
      vn: web
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: hubs
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr2
  annotations:
    core.juniper.net/display-name: vnr2
  labels:
    vnrgroup: hubs
spec:
  type: hub
  virtualNetworkSelector:
    matchLabels:
      vn: db
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: spokes
```

Virtual Network Router (VNR) – Hub&Spoke – Case2



- 異なるNamespace間でHub&Spoke
- VN1, VN2はSpoke VNRに接続、VN3はHub VNRに接続
- VN1 / VN3間及び VN2 / VN3間は互いに通信可能
- VN1 / VN2間は通信不可

Sample Yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
    ns: ns1
---
apiVersion: v1
kind: Namespace
metadata:
  name: ns2
  labels:
    core.juniper.net/isolated-namespace: "true"
    ns: ns2
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
```

Virtual Network Router (VNR) – Hub&Spoke – Case2

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns2
  name: subnet3
  annotations:
    core.juniper.net/display-name: subnet3
spec:
  cidr: "30.0.0.0/24"
  defaultGateway: 30.0.0.1
---
```

Virtual Network Router (VNR) – Hub&Spoke – Case2

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns2
  name: vn3
  annotations:
    core.juniper.net/display-name: vn3
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns2
    name: subnet3
---
```

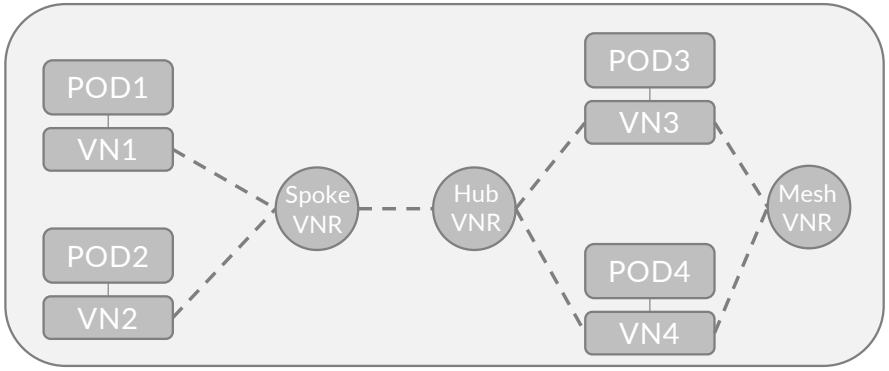
```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
    vnrgroup: spokes
spec:
  type: spoke
  virtualNetworkSelector:
    matchLabels:
      vn: web
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: hubs
        namespaceSelector:
          matchLabels:
            ns: ns2
---
```

Virtual Network Router (VNR) – Hub&Spoke – Case2

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns2
  name: vnr2
  annotations:
    core.juniper.net/display-name: vnr2
  labels:
    vnrgroup: hubs
spec:
  type: hub
  virtualNetworkSelector:
    matchLabels:
      vn: db
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: spokes
        namespaceSelector:
          matchLabels:
            ns: ns1
```

Virtual Network Router (VNR) – Mesh&Hub&Spoke

Namespace1



- 異なるNamespace間でHub&Spoke
- VN1, VN2はSpoke VNRに接続、VN3, VN4はHub VNRとMesh VNRに接続
- VN1 - VN3/VN4間、VN2 - VN3/VN4間は互いに通信可能
- VN3, VN4をMesh VNRにも接続することにより、VN3-VN4間も接続可能となる

Sample Yaml

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

Virtual Network Router (VNR) – Mesh&Hub&Spoke

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet2
  annotations:
    core.juniper.net/display-name: subnet2
spec:
  cidr: "20.0.0.0/24"
  defaultGateway: 20.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
  annotations:
    core.juniper.net/display-name: vn2
  labels:
    vn: web
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet3
  annotations:
    core.juniper.net/display-name: subnet3
spec:
  cidr: "30.0.0.0/24"
  defaultGateway: 30.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn3
  annotations:
    core.juniper.net/display-name: vn3
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet3
---
```

Virtual Network Router (VNR) – Mesh&Hub&Spoke

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet4
  annotations:
    core.juniper.net/display-name: subnet4
spec:
  cidr: "40.0.0.0/24"
  defaultGateway: 40.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn4
  annotations:
    core.juniper.net/display-name: vn4
  labels:
    vn: db
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet4
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr1
  annotations:
    core.juniper.net/display-name: vnr1
  labels:
    vnrgroup: spokes
spec:
  type: spoke
  virtualNetworkSelector:
    matchLabels:
      vn: web
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: hubs
---
```

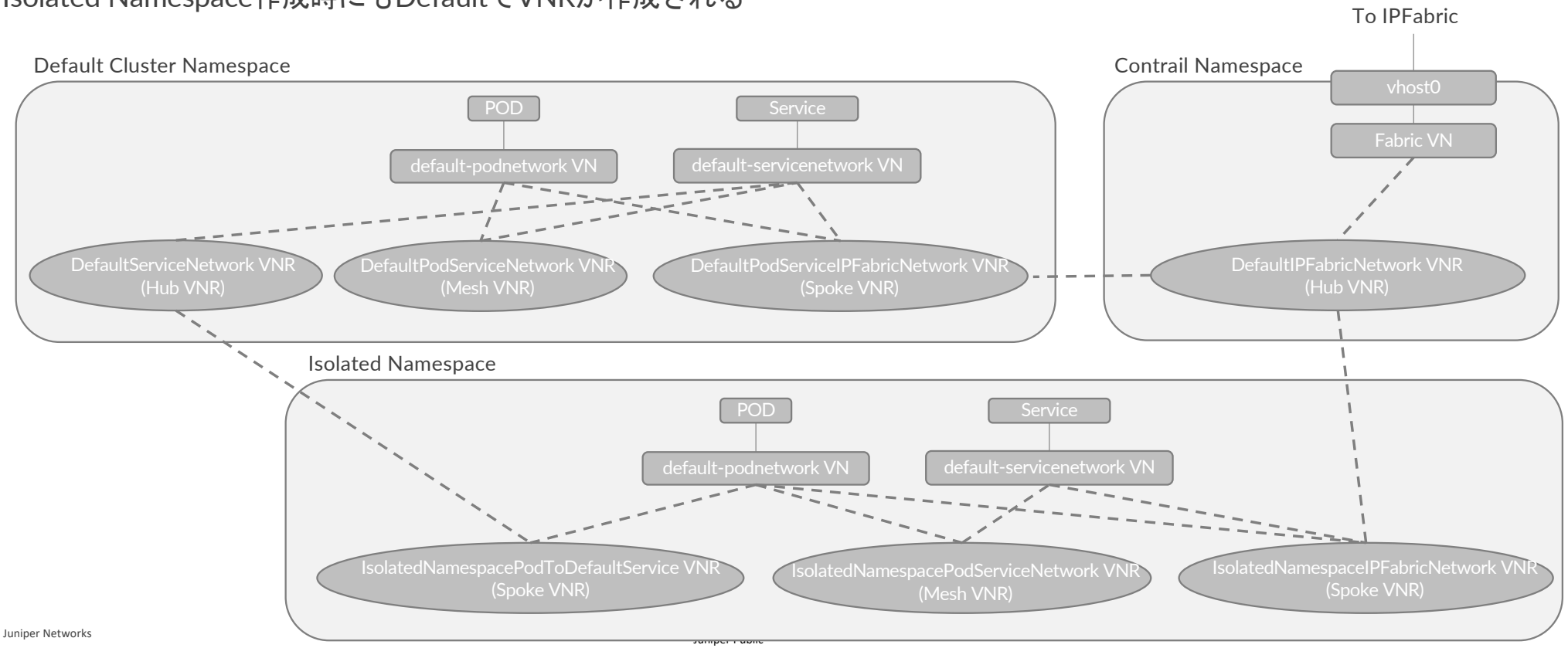
Virtual Network Router (VNR) – Mesh&Hub&Spoke

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr2
  annotations:
    core.juniper.net/display-name: vnr2
  labels:
    vnrgroup: hubs
spec:
  type: hub
  virtualNetworkSelector:
    matchLabels:
      vn: db
  import:
    virtualNetworkRouters:
      - virtualNetworkRouterSelector:
          matchLabels:
            vnrgroup: spokes
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
  namespace: ns1
  name: vnr3
  annotations:
    core.juniper.net/display-name: vnr3
spec:
  type: mesh
  virtualNetworkSelector:
    matchLabels:
      vn: db
```

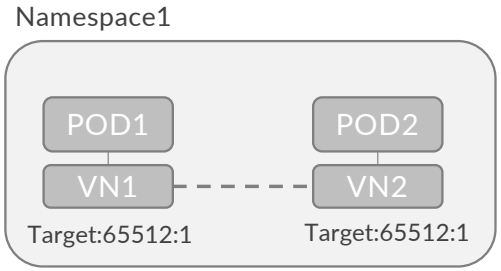
Virtual Network Router (VNR) - Default

- Default Cluster Namespace, Contrail NamespaceにはDefaultで定義されているVNRがあり、default-podnetwork, default-servicenetwork間の接続、Fabric Networkへの接続が定義されている
- Isolated Namespace作成時にもDefaultでVNRが作成される



4. Route Leak

Route Leak – Same Route Target

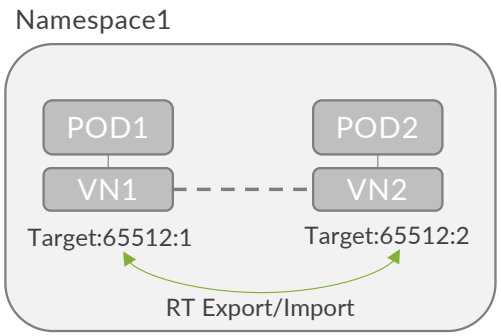


- VNRを使用せず、Virtual NetworkのRoute Targetを揃えることでVirtual Network間接続が可能
- Route TargetによるRoute LeakはVirutal Network間だけでなく、外部ルータの経路もImport可能

Sample Yaml

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vn1
  namespace: ns1
  annotations:
    juniper.net/networks: '{
      "ipamV4Subnet": "10.0.0.0/24",
      "routeTargetList": ["target:64512:1"]
    }'
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "vn1",
    "type": "contrail-k8s-cni"
  }'
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vn2
  namespace: ns1
  annotations:
    juniper.net/networks: '{
      "ipamV4Subnet": "20.0.0.0/24",
      "routeTargetList": ["target:64512:1"]
    }'
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "vn2",
    "type": "contrail-k8s-cni"
  }'
```

Route Leak – Route Target Import



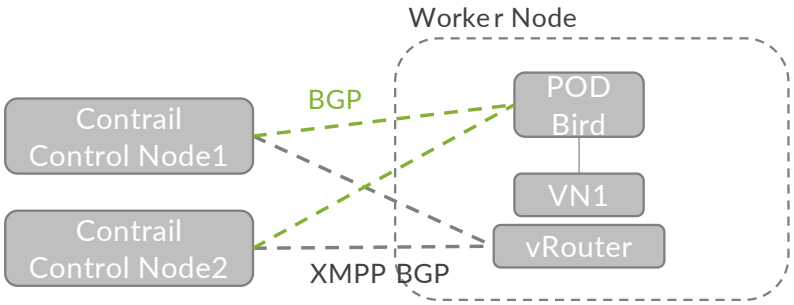
- VNRを使用せず、Virtual NetworkのRoute TargetをExport/ImportすることでVirtual Network間接続が可能
- Route TargetによるRoute LeakはVirutal Network間だけでなく、外部ルータの経路もImport可能

Sample Yaml

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vn1
  namespace: ns1
  annotations:
    juniper.net/networks: '{
      "ipamV4Subnet": "10.0.0.0/24",
      "routeTargetList": ["target:64512:1"],
      "importRouteTargetList": ["target:64512:2"],
      "exportRouteTargetList": ["target:64512:1"]
    }'
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "vn1",
    "type": "contrail-k8s-cni"
  }'
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vn2
  namespace: ns1
  annotations:
    juniper.net/networks: '{
      "ipamV4Subnet": "20.0.0.0/24",
      "routeTargetList": ["target:64512:2"],
      "importRouteTargetList": ["target:64512:1"],
      "exportRouteTargetList": ["target:64512:2"]
    }'
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "vn2",
    "type": "contrail-k8s-cni"
  }'
```

5. BGPaaS

BGPaaS



- BGPaaSにより、PODとContrail Control Node間でBGP接続が可能
- PODはContrail Control Nodeとの間でRouteのImport/Exportが可能
- BGP Peerを張るPODの指定方法は2つ
  - virtualMachineInterfacesSelector
    - PODに設定したLabelを元にPeerを決定
  - virtualMachineInterfacesReferences
    - PODのVMIを指定しPeering

BGPaaS - virtualMachineInterfacesSelector

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet1-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet1-v4
spec:
  cidr: 10.0.0.0/24
  defaultGateway: 10.0.0.1
  bgpaasPrimaryIP: 10.0.0.2
  bgpaasSecondaryIP: 10.0.0.3
  disableBGPaaSIPAutoAllocation: true
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1-v4
---
```

← Default false(BGPaaS IPはBGPaaSを最初に作成した際に自動生成)

trueに指定した場合、bgpaasPrimaryIPはMandatory

BGPaaS - virtualMachineInterfacesSelector

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bird-config
  namespace: ns1
  labels:
    app: bird
    type: config
data:
  bird.conf: |+
    router id 10.0.0.4;
    protocol device { scan time 10; }
    protocol direct { }
    protocol kernel {
      persist;
      scan time 20;
      ipv4 {
        import all;
        export all;
      };
    }
    protocol bgp bgp1_1 {
      local as 10;
      neighbor 10.0.0.2 as 64512;
      ipv4 {
        import all;
        export all;
      };
    }
---
```

```
apiVersion: v1
kind: Pod
metadata:
  name: bird
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: vn1
    core.juniper.net/bgpaas-networks: vn1
spec:
  containers:
    - image: pierky/bird:2.0.9
      imagePullPolicy: IfNotPresent
      name: bird
      volumeMounts:
        - name: config-volume
          mountPath: /etc/bird
  volumes:
    - name: config-volume
      configMap:
        name: bird-config
---
```

BGPaaS - virtualMachineInterfacesSelector

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: BGPAsAService
metadata:
  namespace: ns1
  name: bgpaas-test
spec:
  shared: false
  autonomousSystem: 10
  bgpAsAServiceSessionAttributes:
    loopCount: 2
    routeOriginOverride:
      origin: EGP
    addressFamilies:
      family:
        - inet
        - inet6
  virtualMachineInterfacesSelector:
    - matchLabels:
        core.juniper.net/bgpaasVN: vn1
```

← EGP or IGP

← PODに設定したLabel

Bird BGP Status

```
bird> show protocols
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     02:35:15.724
direct1    Direct     ---        up     02:35:15.724
kernel1    Kernel     master4    up     02:35:15.724
bgp1_1     BGP        ---        up     02:35:20.312  Established

bird> show route
Table master4:
10.0.0.5/32          unicast [bgp1_1 02:38:09.537 from 10.0.0.2] !
(100) [AS64512?]
            via 10.0.0.1 on eth1
10.0.0.4/32          unicast [bgp1_1 02:35:20.315 from 10.0.0.2] !
(100) [AS64512?]
            via 10.0.0.1 on eth1
```

Contrail Status

```
# kubectl get BGPRouter -n ns1
NAME                    TYPE            IDENTIFIER   STATE     AGE
ns1-vn1-bgpaas-server   bgpaas-server                Success
6m37s
ns1-vn1-bird-cb1d9a9c   bgpaas-client   10.0.0.4     Success
6m37s
```

BGPaaS - virtualMachineInterfacesReferences

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
  labels:
    core.juniper.net/isolated-namespace: "true"
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet1-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet1-v4
spec:
  cidr: 10.0.0.0/24
  defaultGateway: 10.0.0.1
  bgpaasPrimaryIP: 10.0.0.2
  bgpaasSecondaryIP: 10.0.0.3
  disableBGPaaSIPAutoAllocation: true
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1-v4
---
```

BGPaaS - virtualMachineInterfacesSelector

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bird-config
  namespace: ns1
  labels:
    app: bird
    type: config
data:
  bird.conf: |+
    router id 10.0.0.4;
    protocol device { scan time 10; }
    protocol direct { }
    protocol kernel {
      persist;
      scan time 20;
      ipv4 {
        import all;
        export all;
      };
    }
    protocol bgp bgp1_1 {
      local as 10;
      neighbor 10.0.0.2 as 64512;
      ipv4 {
        import all;
        export all;
      };
    }
---
```

```
apiVersion: v1
kind: Pod
metadata:
  name: bird
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: vn1
spec:
  containers:
    - image: pierky/bird:2.0.9
      imagePullPolicy: IfNotPresent
      name: bird
      volumeMounts:
        - name: config-volume
          mountPath: /etc/bird
  volumes:
    - name: config-volume
      configMap:
        name: bird-config
```

BGPaaS - virtualMachineInterfacesSelector

- POD作成後、VMI名を確認

```
# kubectl get vmi -n ns1
CLUSTERNAME                                        NAME            NETWORK              PODNAME   IFCNAME   STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local    bird-cb1d9a9c   vn1                  bird      eth1      Success   8m12s
contrail-k8s-kubemanager-cluster1-juniper-local    bird-d655861b   default-podnetwork   bird      eth0      Success   8m12s
```

BGPaaS - virtualMachineInterfacesSelector

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: BGPAsAService
metadata:
  namespace: ns1
  name: bgpaas-test
spec:
  shared: false
  autonomousSystem: 10
  bgpAsAServiceSessionAttributes:
    loopCount: 2
    routeOriginOverride:
      origin: EGP
    addressFamilies:
      family:
        - inet
        - inet6
 virtualMachineInterfaceReferences:
    - apiVersion: core.contrail.juniper.net/v1alpha1
      kind: VirtualMachineInterface
      namespace: ns1
      name: bird-cb1d9a9c
```

← 作成したPODのVMIを指定

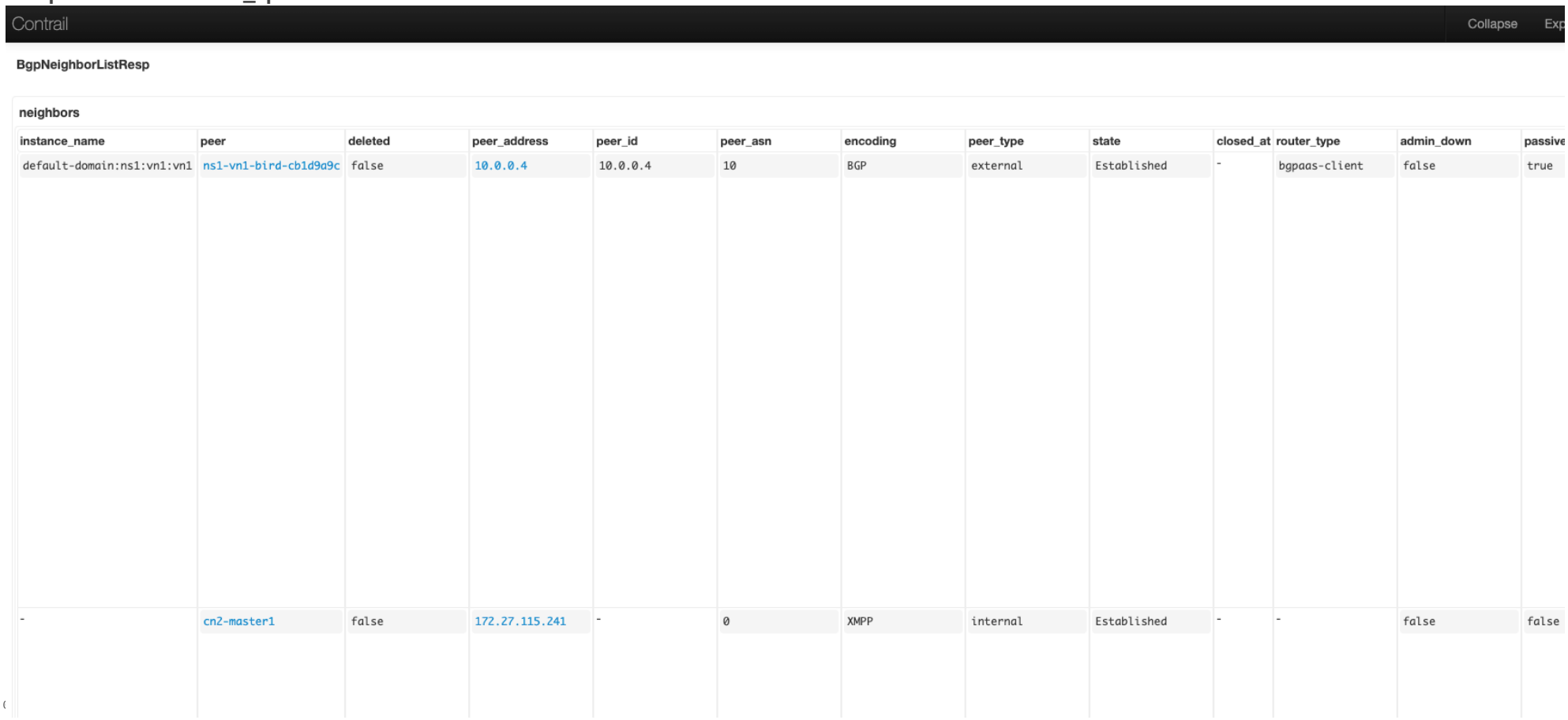
Bird BGP Status

```
bird> show protocols
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     02:35:15.724
direct1    Direct     ---        up     02:35:15.724
kernel1    Kernel     master4    up     02:35:15.724
bgp1_1     BGP        ---        up     02:35:20.312  Established

bird> show route
Table master4:
10.0.0.5/32          unicast [bgp1_1 02:38:09.537 from 10.0.0.2] !
(100) [AS64512?]
            via 10.0.0.1 on eth1
10.0.0.4/32          unicast [bgp1_1 02:35:20.315 from 10.0.0.2] !
(100) [AS64512?]
            via 10.0.0.1 on eth1
```

BGPaaS

- Contrail側のBGP Peer、RouteなどはGUIから確認可能

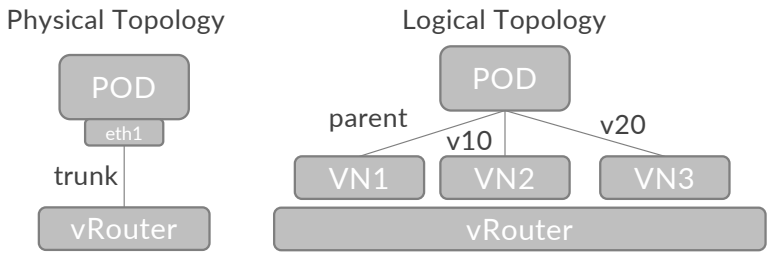
http://controller_ip:8083



instance_name	peer	deleted	peer_address	peer_id	peer_asn	encoding	peer_type	state	closed_at	router_type	admin_down	passive
default-domain:ns1:vn1:vn1	ns1-vn1-bird-cb1d9a9c	false	10.0.0.4	10.0.0.4	10	BGP	external	Established	-	bgpaas-client	false	true
-	cn2-master1	false	172.27.115.241	-	0	XMPP	internal	Established	-	-	false	false

6. VLAN Sub Interface

VLAN Sub Interface



- PODとvRouter間でTrunk接続
- PODにはParent Interface(Tagなし)とSub Interface(Tagあり)を設定
- Parenet / Sub InterfaceはInterface Groupによって紐付ける
- Network PolicyはWhiteList形式(”except” optionはDeny Ruleとなる)

VLAN Sub Interface

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet1-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet1-v4
spec:
  cidr: 10.0.0.0/24
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1-v4
---
```

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet2-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet2-v4
spec:
  cidr: 20.0.0.0/24
  defaultGateway: 20.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn2
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet2-v4
---
```

VLAN Sub Interface

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet3-v4
  namespace: ns1
  annotations:
    core.juniper.net/display-name: subnet3-v4
spec:
  cidr: 30.0.0.0/24
  defaultGateway: 30.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn3
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet3-v4
---
```

```
apiVersion: v1
kind: Pod
metadata:
  name: centos1
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vn1",
          "namespace": "ns1",
          "cni-args": {
              "net.juniper.contrail.interfacegroup": "eth1"
          }
        },
        {
          "name": "vn2",
          "namespace": "ns1",
          "cni-args": {
              "net.juniper.contrail.vlan": "100",
              "net.juniper.contrail.interfacegroup": "eth1"
          }
        },
```

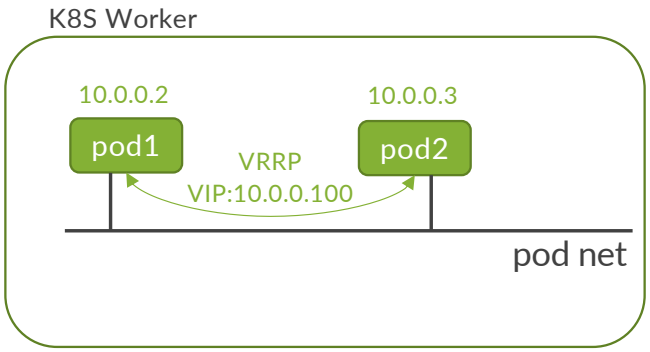
VLAN Sub Interface

```
        {
          "name": "vn3",
          "namespace": "ns1",
          "cni-args": {
              "net.juniper.contrail.vlan": "200",
              "net.juniper.contrail.interfacegroup": "eth1"
          }
        }
      ]
spec:
  containers:
  - name: centos1
    image: centos
    command: [ "/bin/bash", "-c" ]
    args:
     - sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

```
[root@centos1 /]# ip a
2: eth1.100@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
    link/ether 02:3e:63:aa:a4:f3 brd ff:ff:ff:ff:ff:ff
    inet 20.0.0.2/24 brd 20.0.0.255 scope global eth1.100
       valid_lft forever preferred_lft forever
    inet6 fe80::a816:32ff:feaa:1ac4/64 scope link
       valid_lft forever preferred_lft forever
3: eth1.200@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
    link/ether 02:54:6a:9e:f3:c5 brd ff:ff:ff:ff:ff:ff
    inet 30.0.0.2/24 brd 30.0.0.255 scope global eth1.200
       valid_lft forever preferred_lft forever
    inet6 fe80::a816:32ff:feaa:1ac4/64 scope link
       valid_lft forever preferred_lft forever
401: eth0@if402: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
    link/ether 02:16:89:02:08:2f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.233.65.8/18 brd 10.233.127.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fd85:ee78:d8a6:8607::1:108/112 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::40f:acff:fed2:5e46/64 scope link
       valid_lft forever preferred_lft forever
403: eth1@if404: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
    link/ether 02:dc:c7:d9:ae:0f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a816:32ff:feaa:1ac4/64 scope link
       valid_lft forever preferred_lft forever
```

7. Allowed Address Pair

Allowed Address Pairs



- Allowed Address Pair(VIP)をPODに付与
- Address Modeとして以下の2つを設定可能
  - active-active: VIPへのアクセスはECMPにより分散
  - active-standby: VRRP

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

Allowed Address Pairs

```
apiVersion: v1
kind: Pod
metadata:
  name: keepalived1
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vn1",
          "namespace": "ns1",
          "cni-args": {
            "net.juniper.contrail.allowedAddressPairs": "[{¥"ip¥": ¥"10.0.0.100/32¥",¥"mac¥":¥"X:X:X:X:X:X¥",¥"addressMode¥":
¥"active-standby¥"}]"
          }
        }
      ]
spec:
  containers:
  - name: keepalived1
    image: my-keepalived
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

← macはOption

Allowed Address Pairs

```
apiVersion: v1
kind: Pod
metadata:
  name: keepalived2
  namespace: ns1
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vn1",
          "namespace": "ns1",
          "cni-args": {
            "net.juniper.contrail.allowedAddressPairs": "[{¥"ip¥": ¥"10.0.0.100/32¥",¥"mac¥":¥"X:X:X:X:X:X¥",¥"addressMode¥":
¥"active-standby¥"}]"
          }
        }
      ]
spec:
  containers:
  - name: keepalived2
    image: my-keepalived
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
    securityContext:
      privileged: true
      capabilities:
        add:
          - NET_ADMIN
```

← macはOption

Allowed Address Pairs

Sample: keepalived1

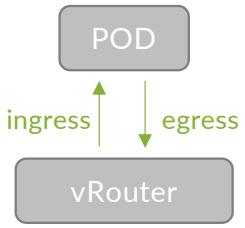
```
global_defs {
    vrrp_garp_master_refresh 60
}
vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 150
    advert_int 1
    virtual_ipaddress {
        10.0.0.100
    }
}
```

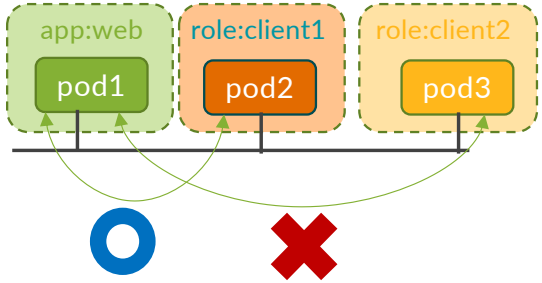
Sample: keepalived2

```
global_defs {
    vrrp_garp_master_refresh 60
}
vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 100
    advert_int 1
    virtual_ipaddress {
        10.0.0.100
    }
}
```

8. Network Policy

Network Policy





- Namespace, Virtual Networkを跨る通信及び、Namespace, Virtual Network内通信、外部通信に対してNetwork Policy(Firewall)を適用
- Network PolicyはPODのLabelに紐付く
- Ingress, Egressの設定が可能(指定しない場合はIngressが適用される)

Network Policy

PODに設定したLabelにマッチした場合、NetworkPolicyがPODに適用される →

Network Policyを適用するPOD向けの通信制御→
送信元にマッチする条件を指定→
ipBlock, namespaceSelector, podSelectorはor条件 →

fromとportsはand条件(両方がマッチしたトラヒックが許可される) →

Network Policyを適用するPOD向け通信の送信Port →
Network Policyを適用するPOD発信の通信制御→

toとportsはand条件(両方がマッチしたトラヒックが許可される) →

Network Policyを適用するPOD発信の送信Port →

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy1
  namespace: ns1
spec:
  podSelector:
    matchLabels:
      role: webserver
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 10.0.0.0/24
        except:
        - 10.0.0.3/32
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: client
    ports:
    - protocol: TCP
      port: 80
  egress:
  - to:
    - podSelector:
        matchLabels:
          role: webserver
    ports:
    - protocol: TCP
      port: 80
```

Network Policy – K8S / Contail マッピング

NETWORK POLICY

CONTRAIL FIREWALL POLICY

NETWORK POLICY	CONTRAIL FIREWALL POLICY
LABEL	Custom Tag (Label毎)
NAMESPACE	Custom Tag (Namespace毎)
NETWORK POLICY	FW Policy (Network Policy毎)
INGRESS RULE	FW Rule (Ingress Rule毎)
INGRESS CIDR RULES	Address Group
CLUSTER	Default Application Policy Set

For standalone K8s cluster, Contrail Objects created with Global scope

Network Policy – Policy作成時に生成されるObject

Address Group

```
# kubectl get addressgroups -n ns1
NAME          SUBNETS   STATE     AGE
10.0.0.0-24   1         Success   39m
10.0.0.3-32   1         Success   39m
```

Tag

```
# kubectl get tags -n ns1
NAME             TAGTYPE     TAGVALUE       ID        STATE     AGE
tag-7965fbc5d9   role        webserver      1507334   Success   43m
tag-79f7d87f89   role        client         1507335   Success   43m
```

Firewall Policy

```
# kubectl get firewallpolicies -n ns1
NAME             RULES   STATE     AGE
k8s-allow-all    2       Success   49m
k8s-deny-all     2       Success   49m
ns1-policy1      5       Success   49m
```

Firewall Rule

```
# kubectl get firewallrules -n ns1
NAME                                        SERVICE                 ENDPOINT1   DIRECTION   ENDPOINT2   ACTION   STATE     AGE
default-egress-ns1-policy1-denyall          any/0:65535->0:65535                >                       deny     Success   47m
default-ingress-ns1-policy1-denyall         any/0:65535->0:65535                <                       deny     Success   47m
k8s-Namespace-ns1-egress                    any/0:65535->0:65535                >                       pass     Success   47m
k8s-Namespace-ns1-ingress                   any/0:65535->0:65535                <                       pass     Success   47m
ns1-egress-policy1-0-PodSelector-0          tcp/0:65535->80:80                  >                       pass     Success   47m
ns1-ingress-policy1-0-PodSelector-2         tcp/0:65535->80:80                  <                       pass     Success   47m
ns1-ingress-policy1-0-namespaceSelector-1   tcp/0:65535->80:80                  <                       pass     Success   47m
ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.0-24   tcp/0:65535->80:80            <                       pass     Success   47m
ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.3-32   tcp/0:65535->80:80            <                       deny     Success   47m
```

Network Policy – Firewall Rule シーケンス

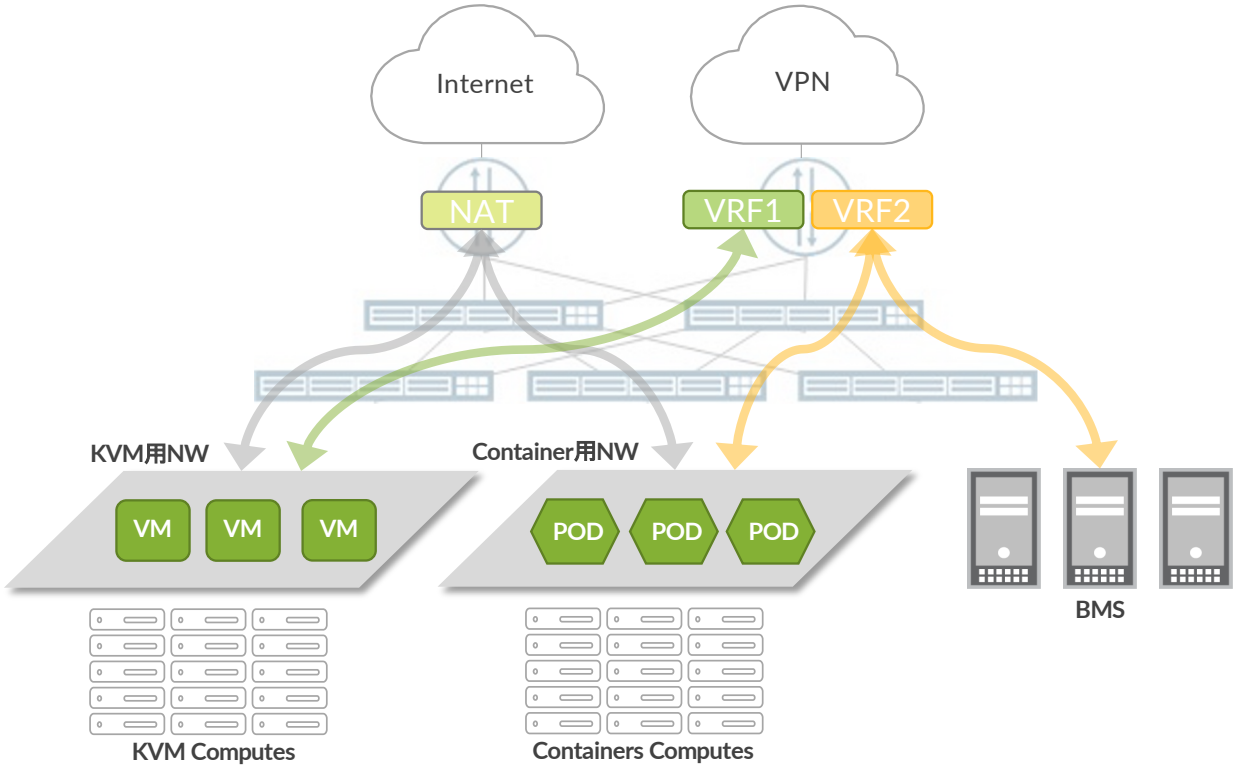
Address Group

```
# kubectl describe firewallpolicies ns1-policy1 -n ns1
Name:         ns1-policy1
--- snip ----
Spec:
  Firewall Rule References:
    API Version:  core.contrail.juniper.net/v1alpha1
    Attributes:
      Sequence:  1.000000
    Fq Name:
      default-domain
      ns1
      ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.3-32
    Kind:              FirewallRule
    Name:              ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.3-32
    Namespace:         ns1
    Resource Version:  5427018
    UID:               8735915c-584e-42f5-917c-19555b6ae8e3
    API Version:       core.contrail.juniper.net/v1alpha1
    Attributes:
      Sequence:  2.000000
    Fq Name:
      default-domain
      ns1
      ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.0-24
    Kind:              FirewallRule
    Name:              ns1-ingress-policy1-ipBlock-0-cidr--10.0.0.0-24
    Namespace:         ns1
    Resource Version:  5427032
    UID:               caf01e6a-b4ee-4b19-8141-db65ec0ff1ec
    API Version:       core.contrail.juniper.net/v1alpha1
    Attributes:
      Sequence:  3.000000
    Fq Name:
      default-domain
      ns1
      ns1-ingress-policy1-0-namespaceSelector-1
    Kind:              FirewallRule
    Name:              ns1-ingress-policy1-0-namespaceSelector-1
    Namespace:         ns1
    Resource Version:  5427035
    UID:               5885364e-2ace-4c71-a62e-772fdc1dae22
    API Version:       core.contrail.juniper.net/v1alpha1
    Attributes:
      Sequence:  4.000000
    Fq Name:
      default-domain
      ns1
      ns1-ingress-policy1-0-PodSelector-2
    Kind:              FirewallRule
    Name:              ns1-ingress-policy1-0-PodSelector-2
    Namespace:         ns1
    Resource Version:  5427042
    UID:               068e72cd-d6f8-49ed-92d9-d1740516cb27
    API Version:       core.contrail.juniper.net/v1alpha1
    Attributes:
      Sequence:  5.000000
    Fq Name:
      default-domain
      ns1
      ns1-egress-policy1-0-PodSelector-0
    Kind:              FirewallRule
    Name:              ns1-egress-policy1-0-PodSelector-0
    Namespace:         ns1
    Resource Version:  5427047
    UID:               76982eee-0e5c-491c-9fb5-63d4c58c1478
  Fq Name:
    default-domain
    ns1
    ns1-policy1
```

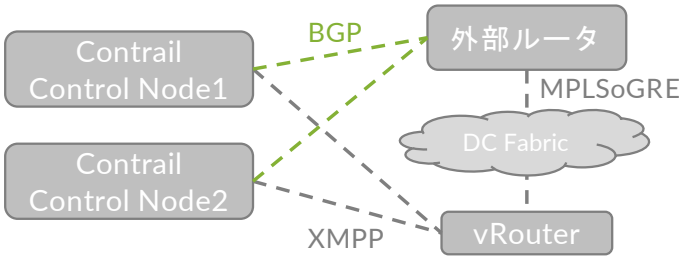
9. 外部ルータ接続

外部ルータ接続

- 仮想ネットワークの外部接続には物理ルータを使用可能となり、ソフトウェアGWのボトルネックを解消
- 仮想ネットワークのVPN網への延伸、NATによるInternet接続が可能



外部ルータ接続 – ルータ定義



- Contrail Control Nodeと外部ルータ間でBGP Peeringし、Contrailで作成したVirtual Networkの経路を外部ルータへAdvertise

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: BGPRouter
metadata:
  namespace: contrail
  name: vmx1
spec:
  parent:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: RoutingInstance
    namespace: contrail
    name: default
  bgpRouterParameters:
    vendor: contrail
    routerType: control-node
    address: 1.1.1.1
    identifier: 1.1.1.1
    routerType: router
    addressFamilies:
      family:
        - route-target
        - inet-vpn
        - e-vpn
        - inet
```

← 外部ルータのIP

外部ルータ接続 – ルータ定義

```
# kubectl get bgprouters -A
NAMESPACE   NAME          TYPE           IDENTIFIER       STATE     AGE
contrail    cn2-master1   control-node   172.27.115.241   Success   12d
contrail    vmx1          router         1.1.1.1          Success   14m

# kubectl describe bgprouters vmx1 -n contrail
-- snip --
Spec:
  Bgp Router Parameters:
    Address:  1.1.1.1
    Address Families:
      Family:
        route-target
        inet-vpn
        e-vpn
        inet
    Auth Data:
    Autonomous System:  64512
    Identifier:         1.1.1.1
    Port:               179
    Router Type:        router
    Vendor:             contrail

root@vmx1> show bgp summary
172.27.115.241        64512         41         43       0       0       18:09 Establ
  bgp.rtarget.0: 15/15/15/0
  bgp.l3vpn.0: 0/0/0/0
```

← 外部ルータのBGPエントリー

← 外部ルータ登録することにより自動的にContrail Control NodeにPeerが追加される

← 外部ルータ側でControl NodeとBGP Peerが張れていることを確認

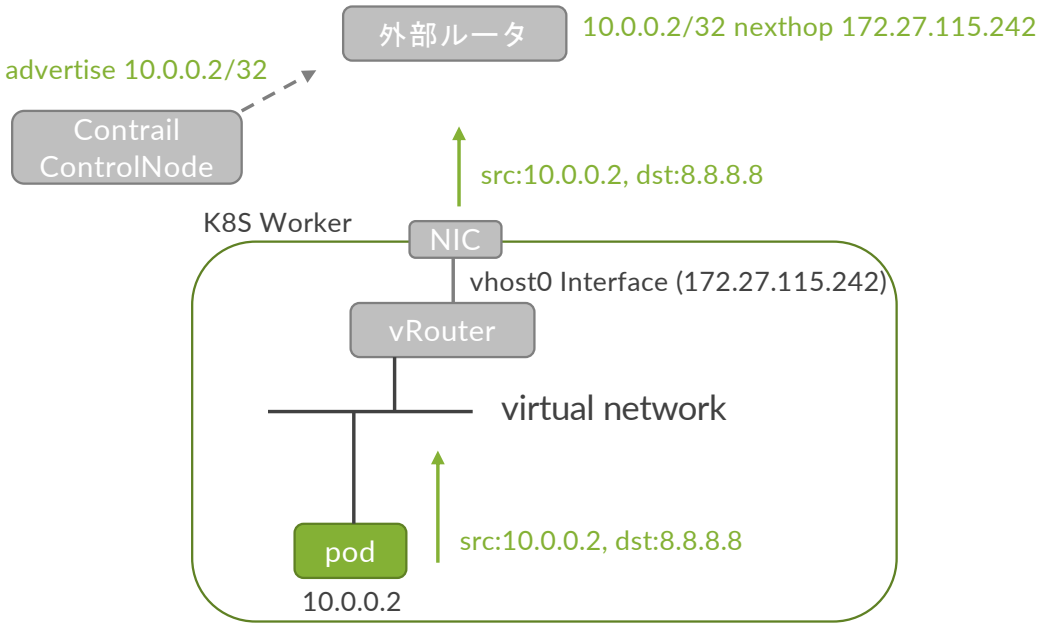
外部ルータ接続 – 外部ルータ設定

```
set interfaces ge-0/0/0 unit 0 family inet address 172.27.115.245/22
set interfaces lo0 unit 0 family inet address 1.1.1.1/32
set interfaces lo0 unit 1 family inet address 100.0.0.1/32
set routing-options route-distinguisher-id 1.1.1.1
set routing-options resolution rib bgp.rtarget.0 resolution-ribs inet.0
set routing-options resolution rib bgp.l3vpn.0 resolution-ribs inet.3
set routing-options resolution rib bgp.l3vpn.0 resolution-ribs inet.0
set routing-options router-id 1.1.1.1
set routing-options autonomous-system 64512
set routing-options autonomous-system loops 2
set routing-options dynamic-tunnels contrail_gre_tunnel source-address 1.1.1.1
set routing-options dynamic-tunnels contrail_gre_tunnel gre
set routing-options dynamic-tunnels contrail_gre_tunnel destination-networks 172.27.112.0/22
set protocols router-advertisement interface fxp0.0
set protocols router-advertisement interface lo0.0
set protocols bgp group contrail_asn-64512 type internal
set protocols bgp group contrail_asn-64512 local-address 1.1.1.1
set protocols bgp group contrail_asn-64512 hold-time 90
set protocols bgp group contrail_asn-64512 family inet unicast
set protocols bgp group contrail_asn-64512 family inet-vpn unicast
set protocols bgp group contrail_asn-64512 family route-target
set protocols bgp group contrail_asn-64512 local-as 64512
set protocols bgp group contrail_asn-64512 local-as loops 2
set protocols bgp group contrail_asn-64512 multipath
set protocols bgp group contrail_asn-64512 neighbor 172.27.115.241 peer-as 64512
set protocols bgp group contrail_asn-64512 vpn-apply-export
```

← MPLSoGREの接続先(worker node(vRouter)が接続されているセグメント)

← Contrail Control Node IP

外部ルータ接続 – Fabric Forwarding



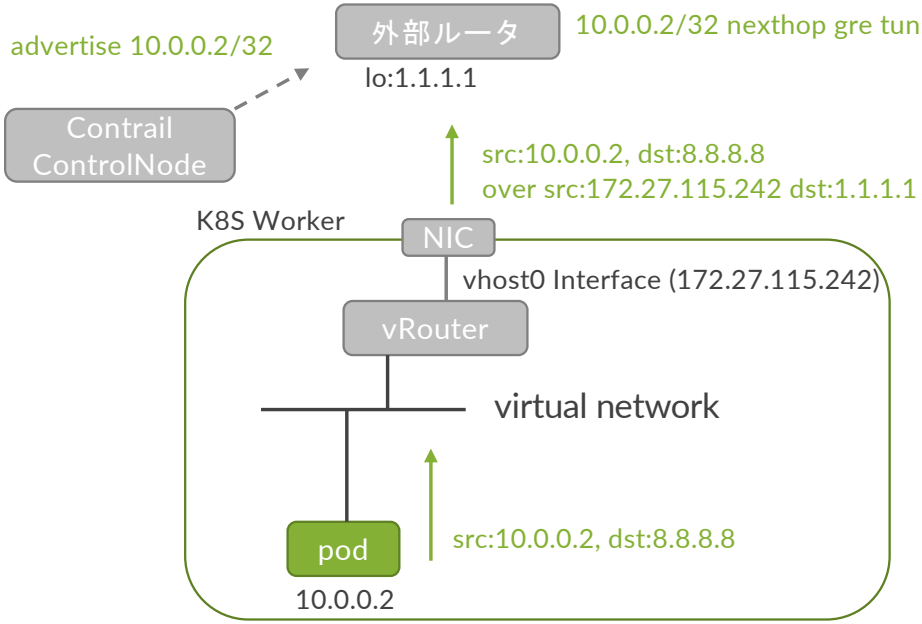

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
  fabricForwarding: true
```

- Contrail Control Nodeと外部ルータ間で BGP/IPv4 Peeringし、Contrailで作成したVirtual Networkの経路を外部ルータへAdvertise
- PODのhost routeが外部ルータへadvertiseされる

外部ルータ接続 – Fabric Forwarding

```
root@vmx1> show route table inet.0

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:08:02
                    >  to 172.27.112.1 via ge-0/0/0.0
1.1.1.1/32         *[Direct/0] 22:58:03
                    >  via lo0.0
10.0.0.2/32        *[BGP/170] 04:07:09, MED 100, localpref 200, from 172.27.115.241
                      AS path: ?, validation-state: unverified
                    >  to 172.27.115.242 via ge-0/0/0.0
```

外部ルータ接続 – Virtual Network




```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
  routeTargetList:
    - target:64512:1
```

- Contrail Control Nodeと外部ルータ間で BGP/VPNv4 Peeringし、Contrailで作成した Virtual Networkの経路を外部ルータへAdvertise
- PODのhost routeが外部ルータへadvertiseされる

外部ルータ接続 – Virtual Network - 外部ルータ設定

外部ルータでRoutingInstanceを作成し、Contrail Virtual Networkの経路をImport

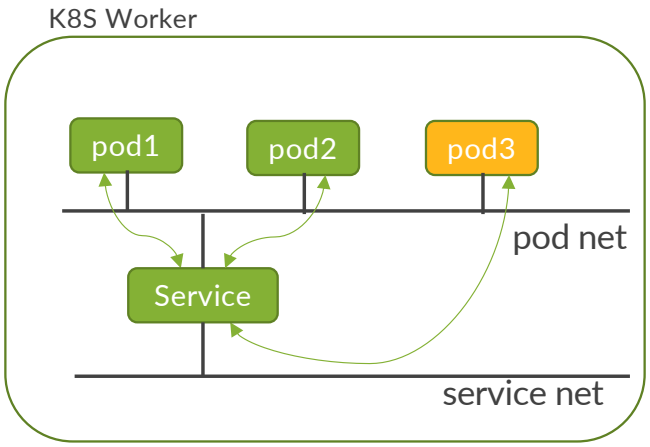
```
set routing-instances vn1 instance-type vrf
set routing-instances vn1 interface lo0.1
set routing-instances vn1 vrf-target target:64512:1
set routing-instances vn1 vrf-table-label
```

← Virtual NetworkのRoute Targetと揃える

```
root@vmx1> show route table vn1.inet.0

vn1.inet.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.2/32        *[BGP/170] 00:01:16, MED 100, localpref 200, from 172.27.115.241
                      AS path: ?, validation-state: unverified
                    >  via gr-0/3/0.32770, Push 67
99.0.0.1/32        *[Direct/0] 17:16:05
                    >  via lo0.1

root@vmx1> show route table inet.3

inet.3: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.27.112.0/22    *[Tunnel/305] 18:20:48
                       Tunnel
172.27.115.241/32  *[Tunnel/300] 00:01:09
                    >  via gr-0/3/0.32771
172.27.115.242/32  *[Tunnel/300] 17:51:57
                    >  via gr-0/3/0.32770
172.27.115.243/32  *[Tunnel/300] 17:51:57
                    >  via gr-0/3/0.32769
```

10. ClusterIP

ClusterIP




```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: ns1
spec:
selector:
    app: web
  ports:
  - port: 80
    targetPort: 80
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: ns1
  labels:
    app: web
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
    ports:
    - containerPort: 80
```

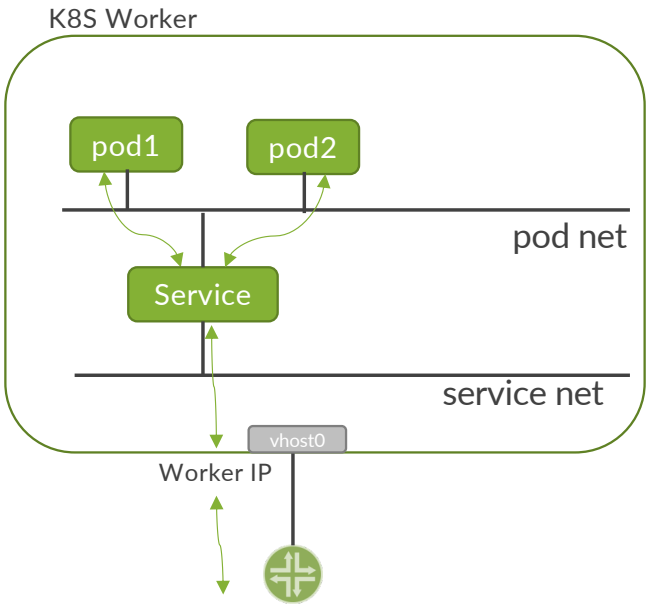
- Service(ClusterIP)を定義することにより、ECMPベースで複数PODにアクセスが可能
- Contrail環境ではdefault-podnetworkとdefault-servicenetworkはVNRにより互いにRouteLeakしているため、PODからClusterIPへ接続可
- 新規作成したVirtual Networkからdefault-servicenetworkへの接続は別途RouteLeakが必要

ClusterIP

```
# kubectl get services -n ns1
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
my-service   ClusterIP   10.233.54.224   <none>        80/TCP    77s

# kubectl get floatingips -n ns1
NAME                                                          IPADDRESS       INTERFACES   PORTMAPPING   STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-my-service-28b76502  10.233.54.224   1            TCP/80->80    Success   8m30s

[root@centos1 /]# curl 10.233.54.224
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
--- snip ---

[root@centos1 /]# curl my-service.ns1.svc.cluster1.juniper.local
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
--- snip ---
```

11. NodePort

NodePort



Range: 30000-32767 →

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: ns1
spec:
  type: NodePort
  selector:
    app: web
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30010
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: ns1
  labels:
    app: web
  annotations:
    k8s.v1.cni.cncf.io/networks: ns1/vn1
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
    ports:
    - containerPort: 80
```

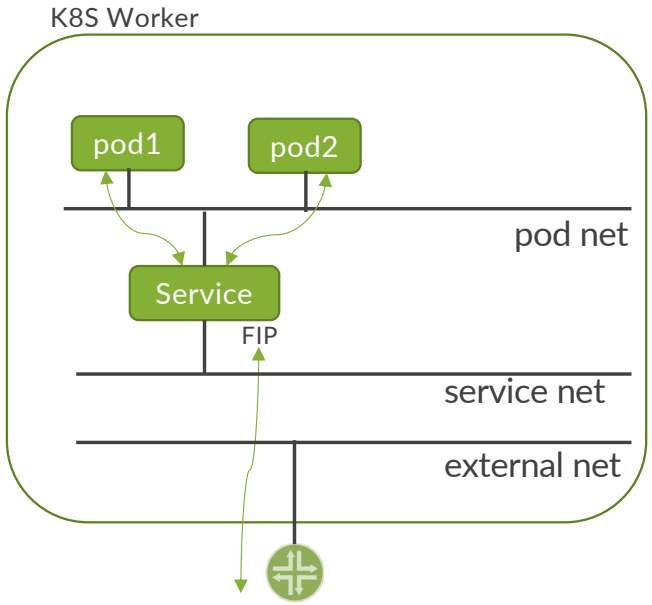
- K8S Cluster外部からWorkerNode IPでPortForwardingによる接続
- Contrail環境ではFloatingIPが生成され、VMIにアタッチされる

NodePort

```
# kubectl get services -n ns1
NAME         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
my-service   NodePort   10.233.12.191   <none>        80:30010/TCP   8m24s

# kubectl get floatingips -n ns1
NAME                                                          IPADDRESS       INTERFACES   PORTMAPPING   STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-my-service-28b76502  10.233.12.191   1            TCP/80->80    Success   21m

# curl worker_node_vhost0_ip:30010
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
--- snip ---
```

12. Loadbalancer

Loadbalancer



- Service(LoadBalancer)を定義することにより、FloatingIPを使用し外部からPODへECMPアクセス可能
- FloatingIPはExternal Netから払い出され、ServiceIPにアタッチされる
- FloatingIPは外部RouterへAdvertise可能
  - External Netのfabricforwardingをtrueにすることで、Contrail Controller/外部Router間でBGP/ipv4経路がAdvertiseされ、Underlay接続となる
  - External Netのfabricforwardingをfalseにし、外部RouterのRoutingInstanceとRouteLeakすることにより、Contrail Controller/外部Router間でBGP/vpnv4経路がAdvertiseされ、Overlay接続となる

Loadbalancer – External Netの定義

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet-external1
  namespace: ns1
spec:
  cidr: 100.0.0.0/24
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn-external1
  labels:
    service.contrail.juniper.net/externalNetwork: default-external
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet-external1
  fabricForwarding: true
```

```
# kubectl edit kubemanager contrail-k8s-kubemanager -n contrail
Spec箇所に以下を追記

spec:
  externalNetworkSelectors:
    default-external:
      networkSelector:
        matchLabels:
          service.contrail.juniper.net/externalNetwork: default-external
      namespaceSelector:
        matchLabels:
          ns: ns1
```

※Defaultで使用するExternal Netを定義

Loadbalancer – Loadbalancerの定義

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: ns1
spec:
  ipFamilies: ["IPv4", "IPv6"]
  type: LoadBalancer
  selector:
    app: web
  ports:
  - port: 8080
    targetPort: 80
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: ns1
  labels:
    app: web
  annotations:
    k8s.v1.cni.cncf.io/networks: ns1/vn1
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
    ports:
    - containerPort: 80
```

← 指定しない場合IPv4のみ

Loadbalancer

```
# kubectl get svc -n ns1
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
my-service   LoadBalancer   10.233.33.83  100.0.0.4     8080:30597/TCP   5m21s

# kubectl get fip -n ns1
NAME                                                          IPADDRESS     INTERFACES   PORTMAPPING     STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-my-service-28b76502   10.233.33.83   1            TCP/8080->80    Success   5m16s
contrail-k8s-kubemanager-cluster1-juniper-local-my-service-d7fd4d39   100.0.0.4      1            TCP/8080->80    Success   5m12s

# curl 100.0.0.4:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
--- snip ---
```

外部ルータ接続している場合は、FloatingIPが自動的に外部ルータへ広報される

```
root@vmx1> show route table inet.0

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

--- snip ---

100.0.0.4/32       *[BGP/170] 00:01:42, MED 100, localpref 200, from 172.27.115.241
                      AS path: ?, validation-state: unverified
                    >  to 172.27.115.242 via ge-0/0/0.0
```

Loadbalancer – External Network Selector

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet-external2
  namespace: ns1
spec:
  cidr: 200.0.0.0/24
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn-external2
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet-external2
  fabricForwarding: true
```

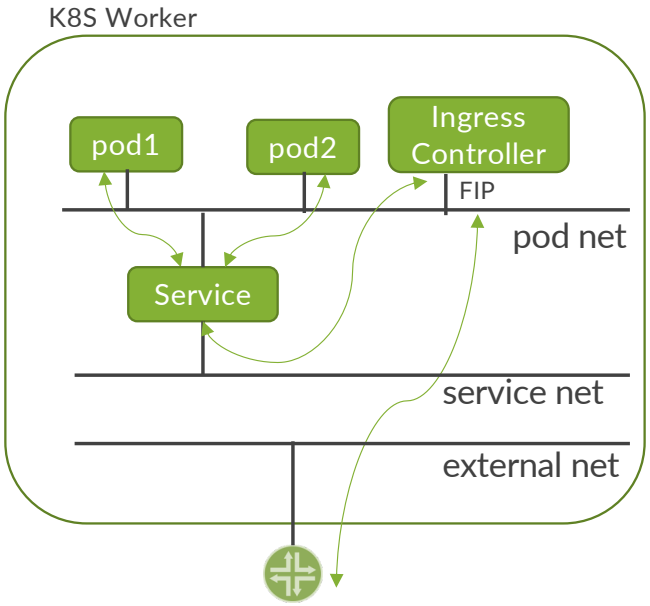
```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: ns1
  annotations:
    service.contrail.juniper.net/externalNetwork: ns1/vn-external2
spec:
  type: LoadBalancer
  selector:
    app: web
  ports:
  - port: 8080
    targetPort: 80
```

※annotaionsにてexternal Networkを指定することで特定のVirtual Networkを使用可能

```
# kubectl get svc -n ns1
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
my-service   LoadBalancer   10.233.27.32   200.0.0.2     8080:31670/TCP   2m29s
```

13. Ingress

Ingress



- Ingressを定義することによりL7 LoadBalancingが可能
- Contrailでは、HAProxy, NGINX, ContourのIngress Controllerをサポート
- FloatingIPは外部RouterへAdvertise可能
  - External Netのfabricforwardingをtrueにすることで、Contrail Controller/外部Router間でBGP/ipv4経路がAdvertiseされ、Underlay接続となる
  - External Netのfabricforwardingをfalseにし、外部RouterのRoutingInstanceとRouteLeakすることにより、Contrail Controller/外部Router間でBGP/vpnv4経路がAdvertiseされ、Overlay接続となる

Ingress – External Netの定義

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  name: subnet-external1
  namespace: ns1
spec:
  cidr: 100.0.0.0/24
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn-external1
  labels:
    service.contrail.juniper.net/externalNetwork: default-external
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet-external1
  fabricForwarding: true
```

※Loadbalancerセクションで設定済みであれば省略

```
# kubectl edit kubemanager contrail-k8s-kubemanager -n contrail
Spec箇所に以下を追記

spec:
  externalNetworkSelectors:
    default-external:
      networkSelector:
        matchLabels:
          service.contrail.juniper.net/externalNetwork: default-external
      namespaceSelector:
        matchLabels:
          ns: ns1
```

※Defaultで使用するExternal Netを定義

Ingress – Ingressの定義(NGINX)

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: Subnet
metadata:
  namespace: ns1
  name: subnet1
  annotations:
    core.juniper.net/display-name: subnet1
spec:
  cidr: "10.0.0.0/24"
  defaultGateway: 10.0.0.1
---
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetwork
metadata:
  namespace: ns1
  name: vn1
  annotations:
    core.juniper.net/display-name: vn1
spec:
  v4SubnetReference:
    apiVersion: core.contrail.juniper.net/v1alpha1
    kind: Subnet
    namespace: ns1
    name: subnet1
---
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress1
  namespace: ns1
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - http:
        paths:
          - path: /testpath
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: web
  namespace: ns1
spec:
  ports:
  - port: 8080
    targetPort: 80
  selector:
    app: web
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: ns1
  labels:
    app: web
  annotations:
    k8s.v1.cni.cncf.io/networks: ns1/vn1
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
    ports:
    - containerPort: 80
```

Ingress

```
# kubectl get svc -n ns1
NAME   TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
web    ClusterIP  10.233.49.40  <none>        8080/TCP   7m27s

# kubectl get ingress -n ns1
NAME       CLASS   HOSTS   ADDRESS     PORTS   AGE
ingress1   nginx   *       100.0.0.5   80      8m7s

# kubectl get fip -n ns1
NAME                                                                                IPADDRESS    INTERFACES   PORTMAPPING                 STATE     AGE
contrail-k8s-kubemanager-cluster1-juniper-local-ingress-nginx-controller-bf14c2bc   100.0.0.5    1            TCP/80->80,TCP/443->443     Success   118m

# curl 100.0.0.5/testpath
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
--- snip ---
```

外部ルータ接続している場合は、FloatingIPが自動的に外部ルータへ広報される

```
root@vmx1> show route table inet.0

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

--- snip ---

100.0.0.5/32       *[BGP/170] 00:01:42, MED 100, localpref 200, from 172.27.115.241
                      AS path: ?, validation-state: unverified
                    >  to 172.27.115.242 via ge-0/0/0.0
```

14. KubeVirt

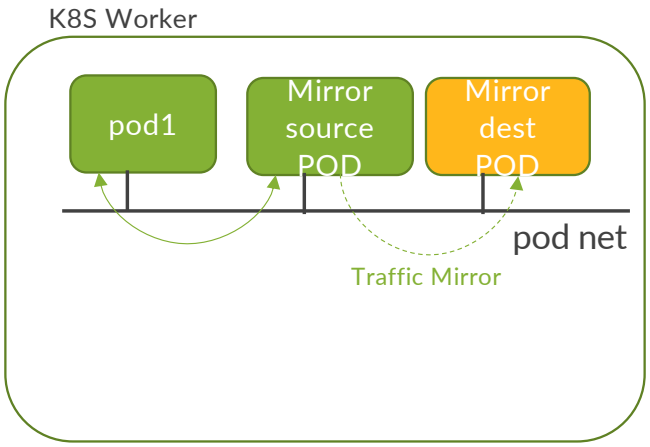
Update soon

15. DPDK

Update soon

16. Port base Mirroring

Port base Mirroring



- PODの通信を特定のPODへMirror

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: MirrorDestination
metadata:
  name: mirrordestinationprofile1
  namespace: ns1
  labels:
    core.juniper.net/analyzer-pod-selector: analyzerpod
spec:
  trafficDirection: both
  juniperHeader: false
  udpPort: 9098
  nextHopMode: dynamic
  staticNhHeader: <null for dynamic nh|vtep tunnel destip, mac,
vxlanid for static nh>
  nextHopMode: <static|dynamic>
  nicAssistedMirroring: <boolean>
   nicAssistedVlanID:
  staticNextHopHeader:
   vTEPDestinationIP:
   vTEPDestinationMac:
   vxlanID:
---
```

← TrafficのMirror元PODに紐付ける名前

← TrafficのMirror先のPODの識別

← both or ingress or egress

← Udp headeを付与する場合はtrue

← dynamic or static

static(Mirror先を指定)の場合、
以降のオプションを設定

Port base Mirroring

```
apiVersion: v1
kind: Pod
metadata:
  name: analyzerpod
  namespace: ns1
  labels:
    core.juniper.net/analyzer-pod: analyzerpod
  annotations:
    core.juniper.net/analyzer-interface: "true"
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vn1",
          "namespace": "ns1",
          "cni-args": {
            "core.juniper.net/analyzer-interface": "true"
          }
        }
      ]
spec:
  containers:
  - name: analyzerpod
    image: centos
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
```

← TrafficのMirror先のPOD

← analyzer-pod-selectorで定義したLabel名

← default-podnetworkでMirror Trafficを受信

← VirtualNetworkでMirror Trafficを受信

Port base Mirroring

```
apiVersion: v1
kind: Pod
metadata:
  name: mirrored-pod
  namespace: ns1
  annotations:
    core.juniper.net/mirror-destination: "mirrordestinationprofile1"
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vn1",
          "namespace": "ns1",
          "cni-args": {
            "core.juniper.net/mirror-destination": "mirrordestinationprofile1"
          }
        }
      ]
spec:
  containers:
  - name: mirrored-pod
    image: centos
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c" ]
    args:
      - ip route del default;
        ip route add default via 10.0.0.1;
        sleep infinity;
```

← TrafficのMirror元のPOD

← default-podnetworkのTrafficをMirror先へ送信
MirrorDestinationリソースの名前を指定

← Virtual NetworkのTrafficをMirror先へ送信
MirrorDestinationリソースの名前を指定

17. Contrail Status

Contrail Status

“kubectl contrail status”でContrailコンポーネントの状態確認が可能

```
# kubectl contrailstatus --all
PODNAME(CONFIG)                          STATUS    NODE          IP              MESSAGE
contrail-k8s-apiserver-5f7c4cb9f9-bnnvx            ok            cn2-master1     172.27.115.241
contrail-k8s-controller-687f5565f4-x9fhv           ok            cn2-master1     172.27.115.241
contrail-k8s-kubemanager-6998b8448f-s5dvg          ok            cn2-master1     172.27.115.241


PODNAME(CONTROL)    STATUS    NODE          IP              MESSAGE
contrail-control-0  ok        cn2-master1   172.27.115.241


LOCAL BGPROUTER     NEIGHBOR BGPROUTER  ENCODING  STATE        POD
cn2-master1         vmx1                BGP       Established  ok  contrail-control-0
cn2-master1         cn2-master1         XMPP      Established  ok  contrail-control-0
cn2-master1         cn2-worker1         XMPP      Established  ok  contrail-control-0
cn2-master1         cn2-worker2         XMPP      Established  ok  contrail-control-0


PODNAME(DATA)                    STATUS    NODE          IP              MESSAGE
contrail-vrouter-masters-ptjng   ok        cn2-master1   172.27.115.241
contrail-vrouter-nodes-2tpnt     ok        cn2-worker2   172.27.115.243
contrail-vrouter-nodes-xv425     ok        cn2-worker1   172.27.115.242
```

Contrail Status

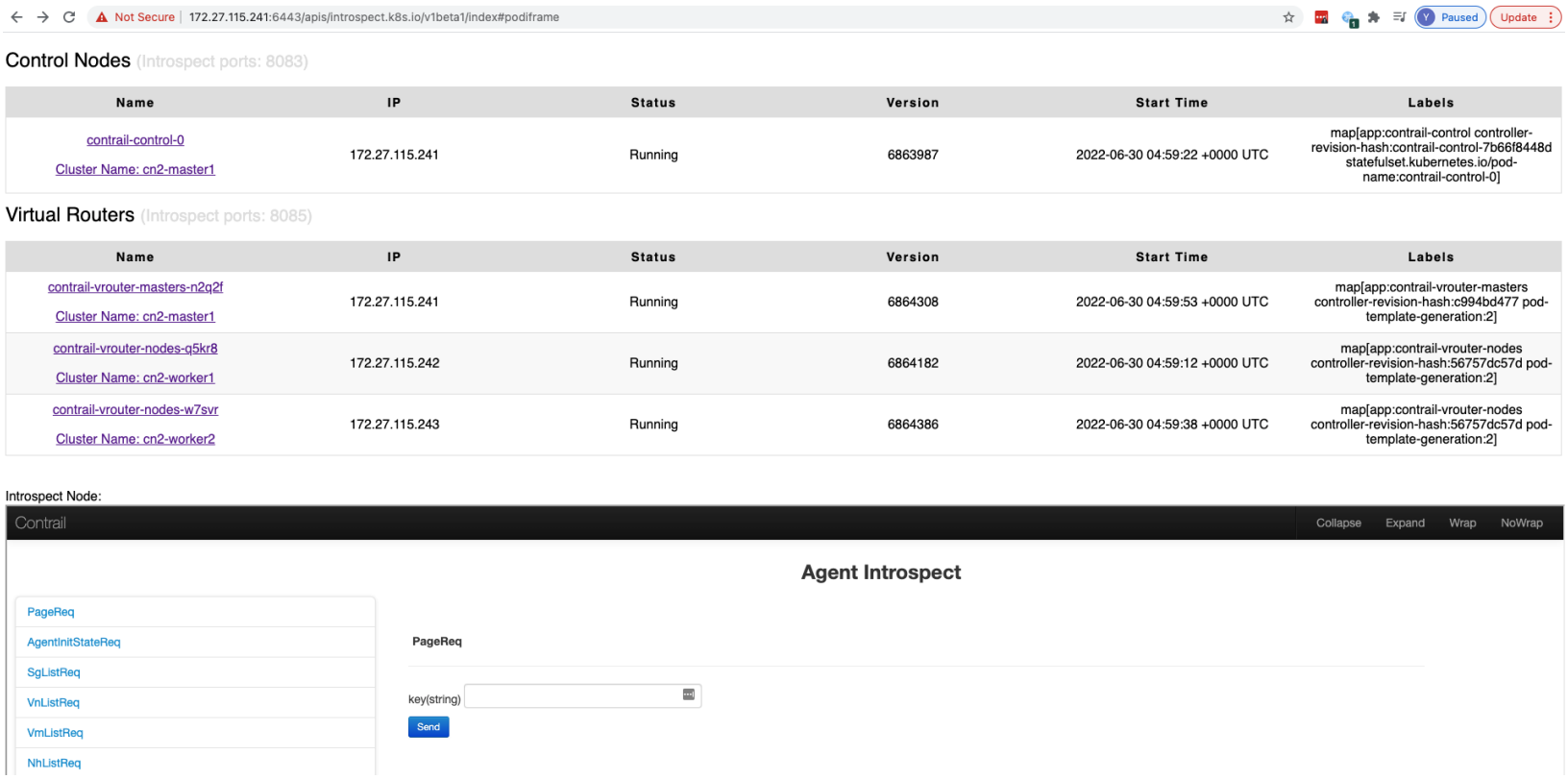
“kubectl contrailstatus”でContrailリソースの確認が可能

```
# kubectl contrailstatus resource routinginstance
NAME                                    NAMESPACE                                                   STATUS
DefaultIPFabricNetwork                  contrail                                                    ok
DefaultPodServiceIPFabricNetwork        contrail-k8s-kubemanager-cluster1-juniper-local-contrail    ok
DefaultPodServiceNetwork                contrail-k8s-kubemanager-cluster1-juniper-local-contrail    ok
DefaultServiceNetwork                   contrail-k8s-kubemanager-cluster1-juniper-local-contrail    ok
IsolatedNamespaceIPFabricNetwork        ns1                                                         ok
IsolatedNamespacePodServiceNetwork      ns1                                                         ok
IsolatedNamespacePodToDefaultService    ns1                                                         ok
default                                 contrail                                                    ok
default-podnetwork                      ns1                                                         ok
default-podnetwork                      contrail-k8s-kubemanager-cluster1-juniper-local-contrail    ok
default-servicenetwork                  contrail-k8s-kubemanager-cluster1-juniper-local-contrail    ok
default-servicenetwork                  ns1                                                         ok
ip-fabric                               contrail                                                    ok
link-local                              contrail                                                    ok
vn-external1                            ns1                                                         ok
vn1                                     ns1                                                         ok

# kubectl contrailstatus resource -h
This command can be used to probe statuses of contrail resources.

Available resources:
  bgprouter, virtualnetwork, routinginstance, globalsystemconfig
```

18. Analytics

Introspection

- Introspectionにより、Contrailのより詳細なDebuggingが可能
- 事前にContrail-Analyticsをインストールしておく必要がある

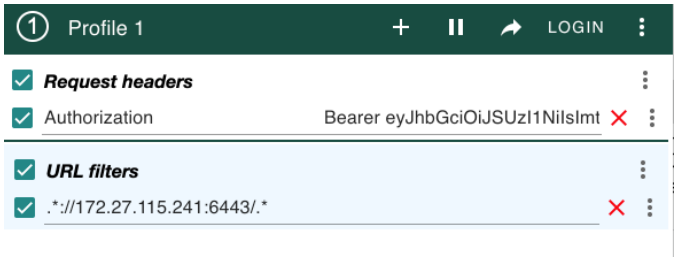


Introspection – アクセス方法

1. Google Chrome の ModHeader Extentionをインストール
2. K8S Masterで以下を実行し、TokenをGet

```
SECRET_NAME=$(kubectl get serviceaccount introspect -n contrail-analytics -o jsonpath='{.secrets[0].name}'); TOKEN=$(kubectl get secret $SECRET_NAME -n contrail-analytics -o jsonpath='{.data.token}' | base64 --decode); echo $TOKEN
```

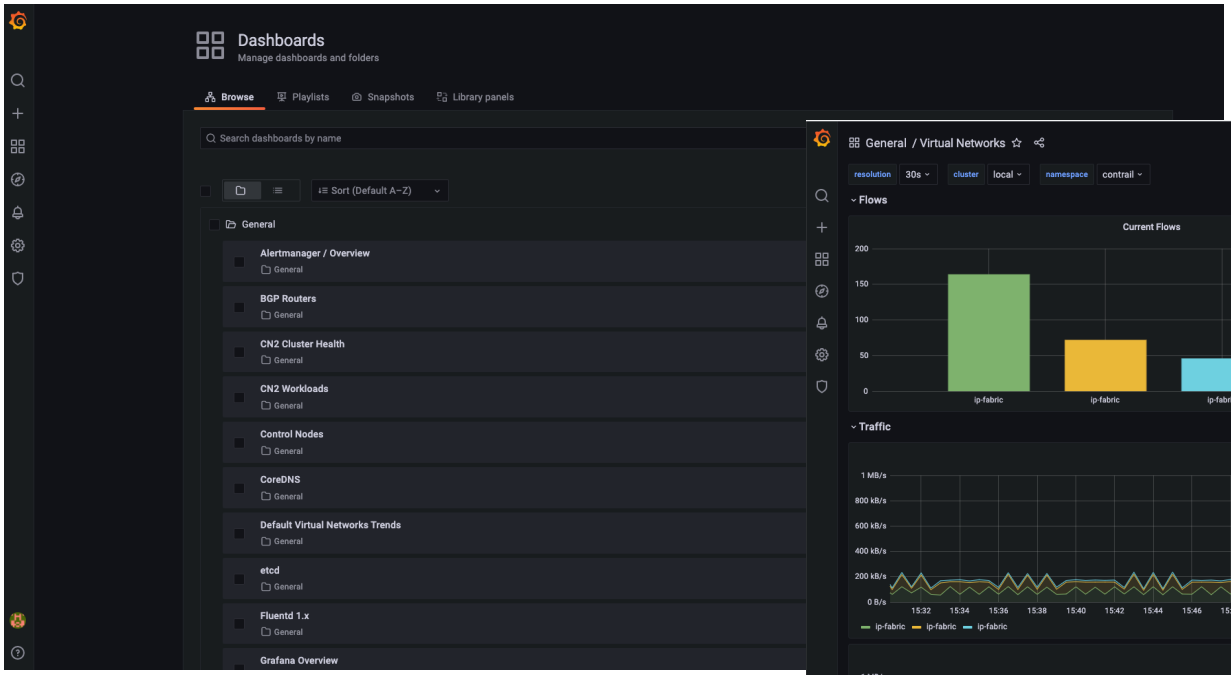
3. ModHeaderに以下を設定

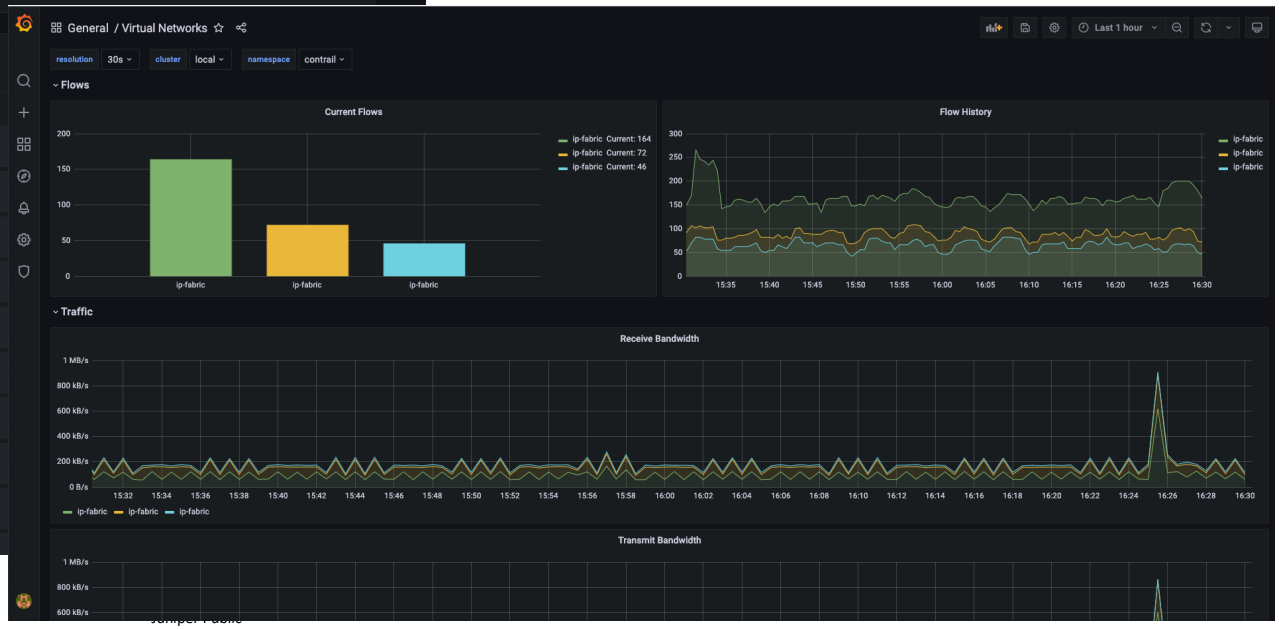


4. https://K8S_API_ServerIP:6443/apis/introspect.k8s.io/v1beta1/index

Grafana/Prometheus

- Grafanaにて各種リソース、トラヒックの使用状況のモニタリングが可能
- 事前にContrail-Analyticsをインストールしておく必要がある
- アクセス方法
  - kubectl port-forward --address K8S_MASTER_IP service/grafana -n contrail-analytics 80:80
  - http://K8S_MASTER_IPにアクセス (User:admin, Pass:prom-operator)

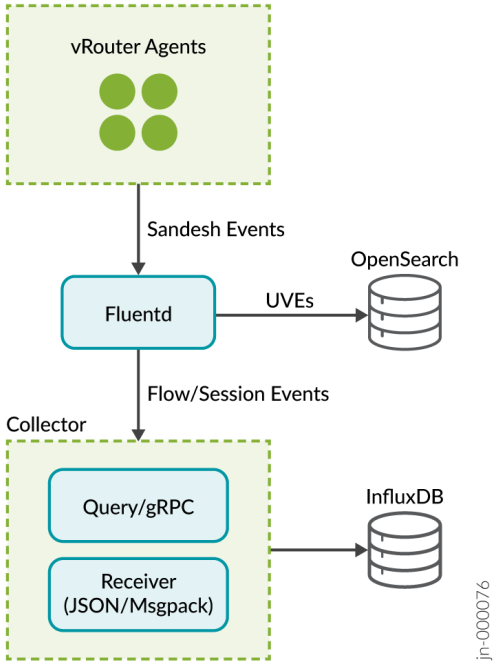




vRouter Session Analytics

- vRouter Agentは、user visible entities (UVEs) 及び traffic information (session)をコレクターにExport
- InfluxDBにクエリすることで以下の情報取得が可能

Column	Filterable	Detail
vn	Yes	Client Virtual Network
vmi	Yes	Interface
remote_vn	Yes	Server Virtual Network
vrouter_ip	Yes	Agent IP
local_ip	Yes	Client IP
client_port	Yes	Client Port
remote_ip	Yes	Server IP
server_port	Yes	Server Port
protocol	Yes	Protocol
label.local.<label-name>	Yes	Client Pod Labels (E.g. client pod with label site maps to label.local.site tag in db)
label.remote.<label-name>	Yes	Server Pod Labels
forward_sampled_bytes	No	Bytes Sent
forward_sampled_pkts	No	Packets Sent
reverse_sampled_bytes	No	Bytes Received
reverse_sampled_pkts	No	Packaets Received
total_bytes	No	Total Bytes Exchanged



vRouter Session Analytics – 設定

1. Fluentdのcluster_ip確認

   kubectl get svc -n contrail-analytics | grep fluentd-loadbalancer

2. vRouter Agentのコレクタ設定(CLUSTER_IP箇所編集)

   kubectl -n contrail patch vrouter contrail-vrouter-masters --type=merge -p '{"spec":{"agent":{"default":{"collectors":[”CLUSTER_IP:24224"]}}}}'

   kubectl -n contrail patch vrouter contrail-vrouter-nodes --type=merge -p '{"spec":{"agent":{"default":{"collectors":[”CLUSTER_IP:24224"]}}}}'

   kubectl -n contrail patch gvc default-global-vrouter-config --type=merge -p '{"spec":{"flowExportRate": 10000}}'

3. InfluxDBのPassword取得

   echo $(kubectl -n contrail-analytics get secret influxdb-auth -o json | jq -M '.data."admin-password"' | cut -f2 -d¥" | base64 -d)

4. PortForward設定

   kubectl port-forward --address K8S_MASTER_IP pods/influxdb-0 -n contrail-analytics 8086:8086

5. InfluxDBへアクセス

   http://K8S_MASTER_IP:8086 (User:admin, Pass:上記3のPass)

6. LogLevel変更(Option)

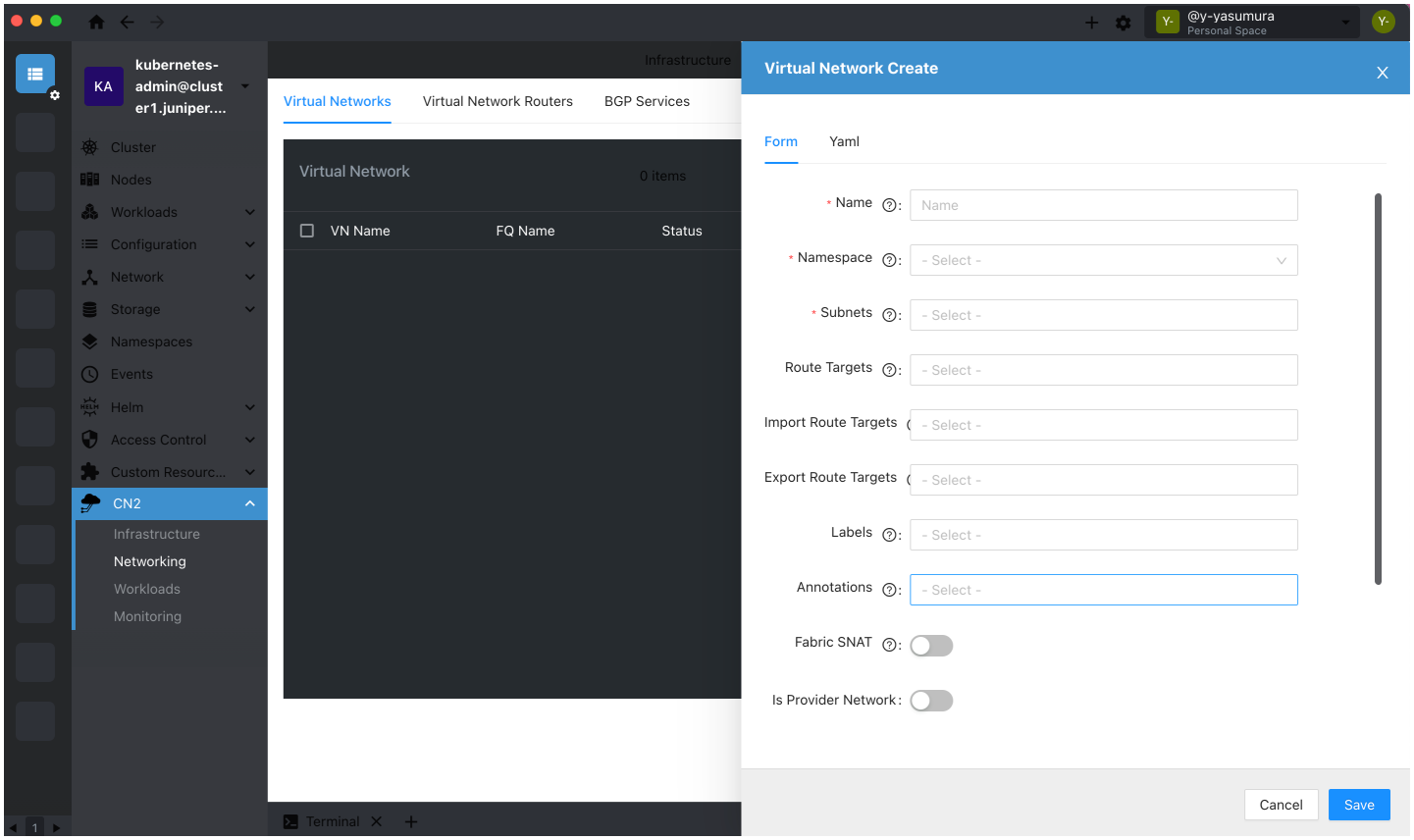
   kubectl -n contrail-analytics edit cm collector

```
data:
  config.yaml: |-
    log:
      caller: true
      level:
        default: debug
```

19. Lens Extension

Lens Extension

- Lens(K8S IDE)のContrail Extensionをインストールすることで、Contrailのオペレーションやリソースモニタリングをの GUIから利用可能
- リソースはYaml形式の他にForm形式で設定が可能



Lens Extension – インストール

- Lens SoftwareをClientPCにインストール(Contrail R22.2時点ではLens 5.4.4推奨)
  - https://k8slens.dev/
- Cluster追加
  - Catalog->Categories->ClustersからClusterを追加
  - Kubeconfigは”/root/.kube/config”をコピペ(serve箇所のIPを環境に合わせて変更)
- Contrail ExtensionをLensにインストール
  - Lens->Extensionsにてcn2_lens_extension-X.X.X.tgzをインストール
  - CN2 Tabが見えない場合、Lensを再起動

20. Contrail Pipeline

Update soon

21. Multi-Cluster

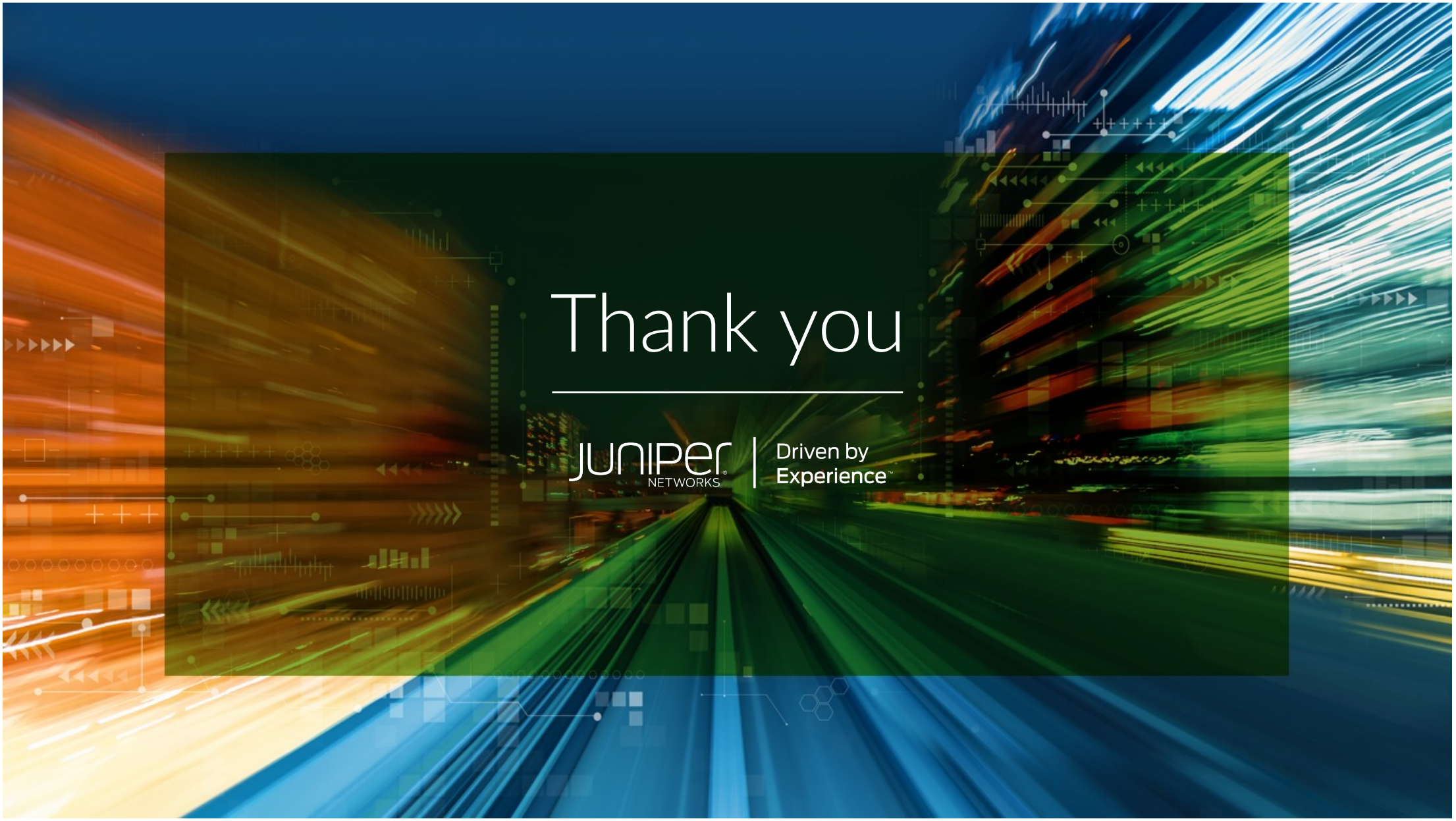
Update soon

Appendix

参考リンク

- 製品概要ページ
  https://www.juniper.net/us/en/products/sdn-and-orchestration/contrail/cloud-native-contrail-networking.html
- 製品技術ドキュメント
  https://www.juniper.net/documentation/product/us/en/cloud-native-contrail-networking
- CN2無料トライアル申請ページ
  https://www.juniper.net/jp/ja/forms/cn2-free-trial.html
- Introduction to the Juniper Contrail Networking Solution (consera動画)
  https://www.coursera.org/learn/juniper-contrail-networking
- Juniper Communities Cloud-Native Contrail Networking (CN2)
  https://community.juniper.net/answers/communities/community-home?communitykey=33a620e9-dd0e-4df4-ba5c-200cbf73b292

- Cloud Native Contrail Networking(CN2)概要およびチュートリアル(Juniper Japan融資)
  https://github.com/jnpr-jp-crdc/CN2


Thank you
JUNIPER NETWORKS
Driven by Experience™