Open Networking Conference Japan 2018

こんなに違う コンテナ・ネットワーキング あれこれ

Open Networking Conference 2018

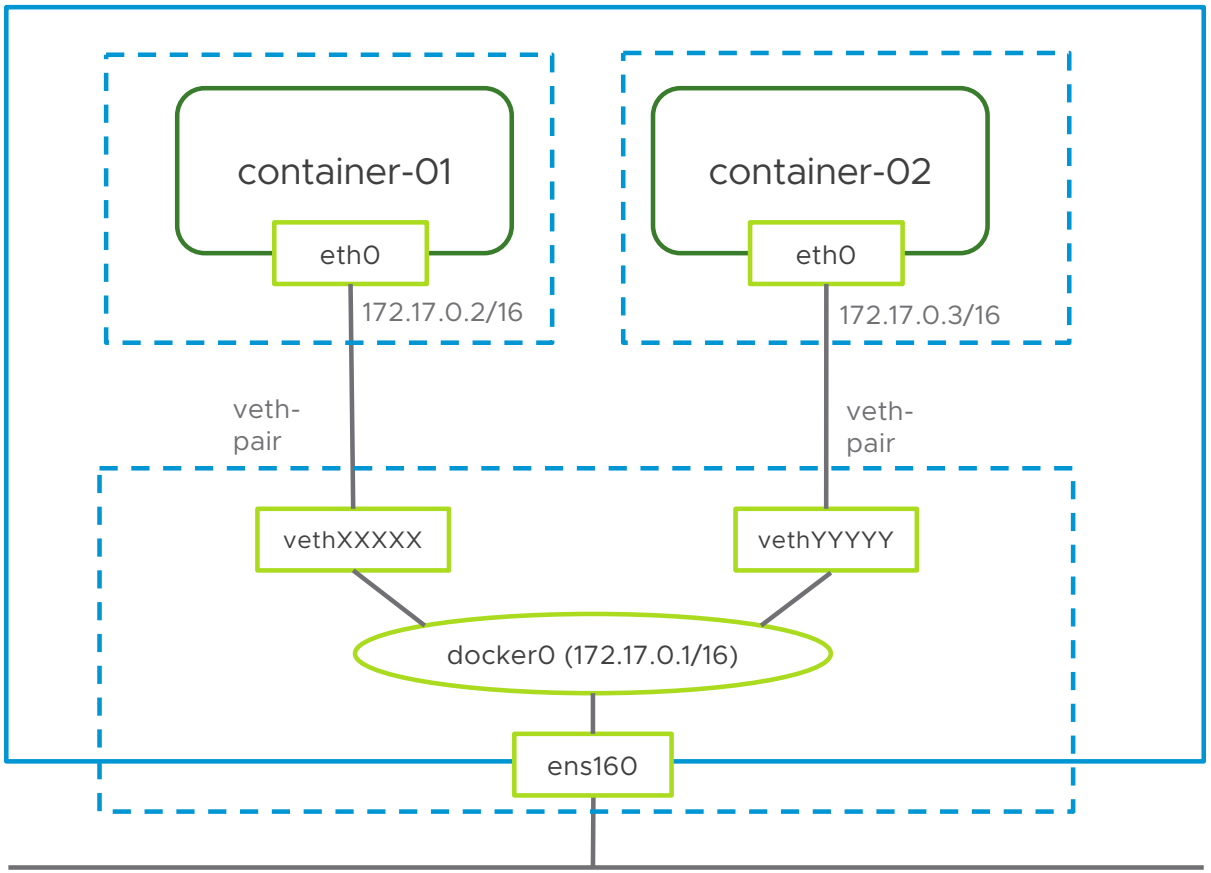
Oct. 19, 2018

CTO, North Asia (Japan, Korea and Greater China)

Motonori Shindo

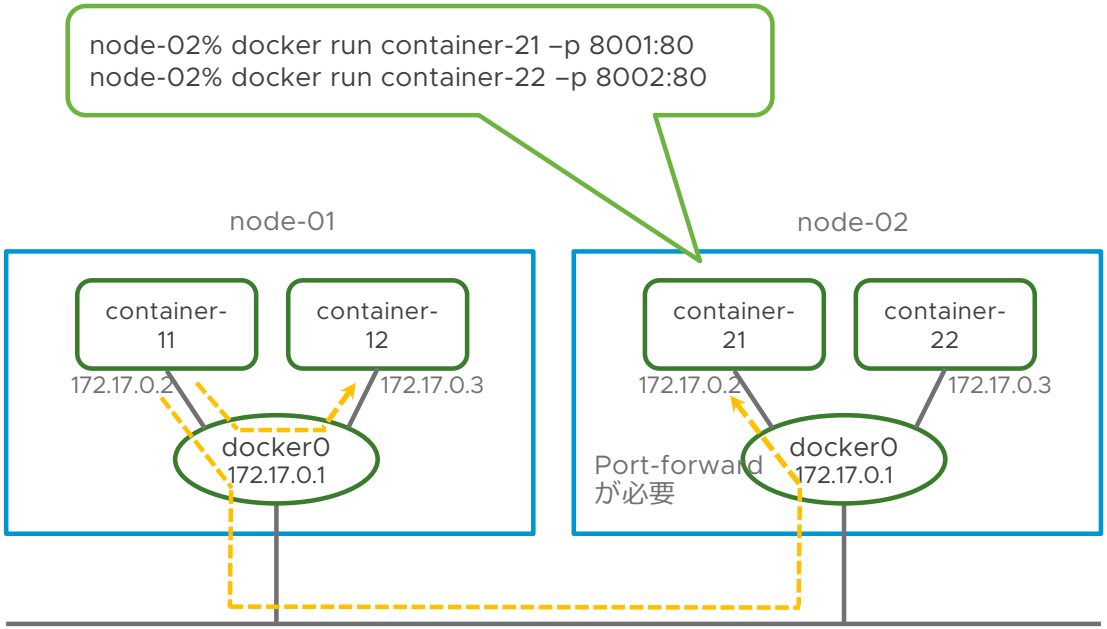
vmware®

Docker Network



- コンテナとホストの間に veth pair が作られ、それぞれ別の namespace に置かれる
- ホスト側に docker0 という bridge が作られ、物理インターフェースと、veth が接続される
  - docker0 にはデフォルトで 172.17.0.1/16 が割り当てられる
- コンテナ側の eth0 には docker0 と同じサブネットから IP アドレスが振られていく
- コンテナの default gateway は docker0 を向く
- Docker0 に割り当てられたサブネットからのパケットが外に出る場合は SNAT (IP masquerade) されて出ていく

Docker Network の課題



- 同一ノード上のコンテナは素直に通信できる
- 複数ノードがある場合、各ノードには同じ IP アドレスが振られる可能性がある（高い）
  - ノードをまたがる通信には port-forward が必要になる
  - Port-forward の管理が大変（これをオーケストレーションする人がいない）

Docker により Container Network Model (CNM) が提案されたが、Kubernetes は採用せず（参考：Why Kubernetes doesn't use libnetwork）

CNI (Container Network Interface)

CoreOS によって提唱、現在は CNCF に移管されている

Linux コンテナが作成された際のネットワーク接続性の確保とコンテナが削除された際のリソース解放を行う

- QoS やネットワーク・ポリシーは範疇外（現状）

コンテナランタイム

- Rkt, Kubernetes, OpenShift, Cloud Foundry, Apache Mesos, Amazon ECS

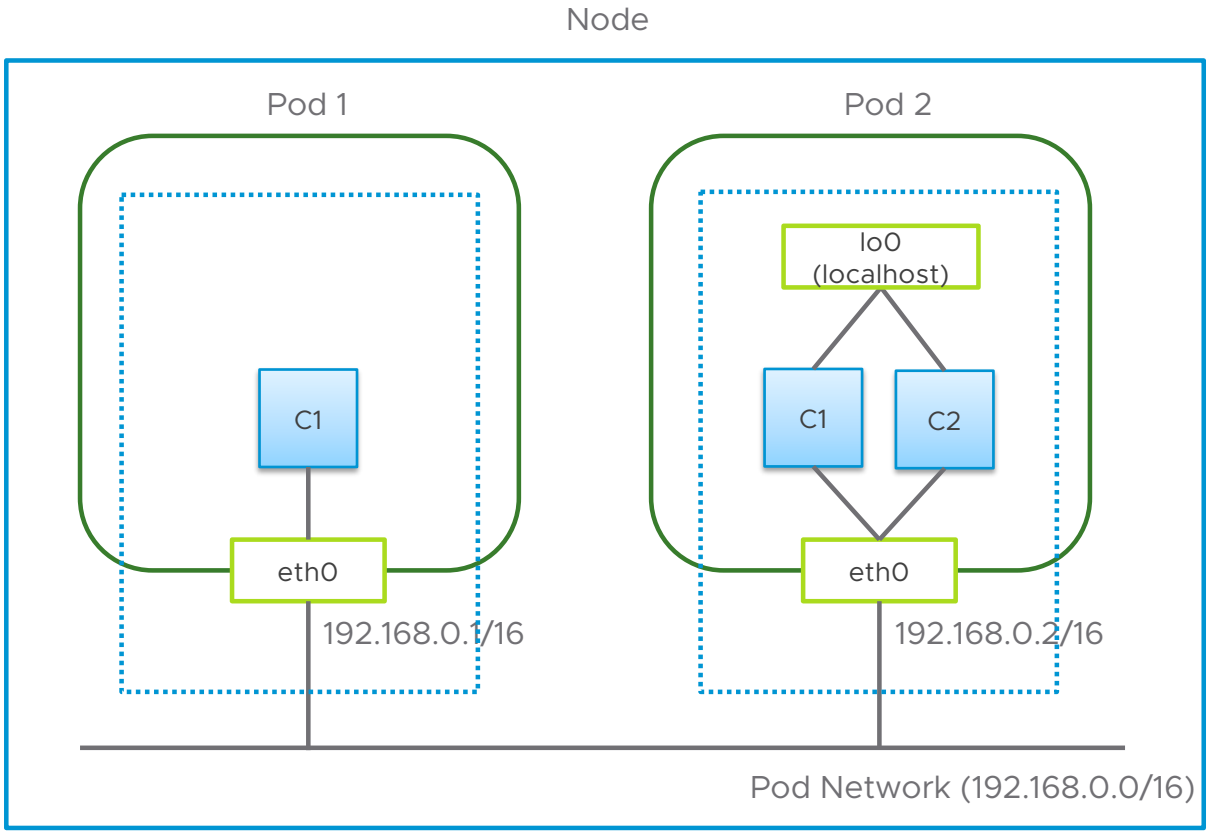
3rd Party プラグイン

- Project Calico, Weave, Contiv, Cilium, Nuage CNI, Silk, ovn-kubernetes, Juniper Contrail/TungstenFabric, NSX-T Container Plugin (NCP), etc.

Kubernetes がネットワークに求める要件

1. 全てのコンテナは他コンテナと NAT なしで通信できること
2. 全てのノードは全てのコンテナと NAT なしで通信できること
3. コンテナから見える自分の IP アドレスと他から見た場合と同じであること

Kubernetes のネットワークモデル（ノード内）



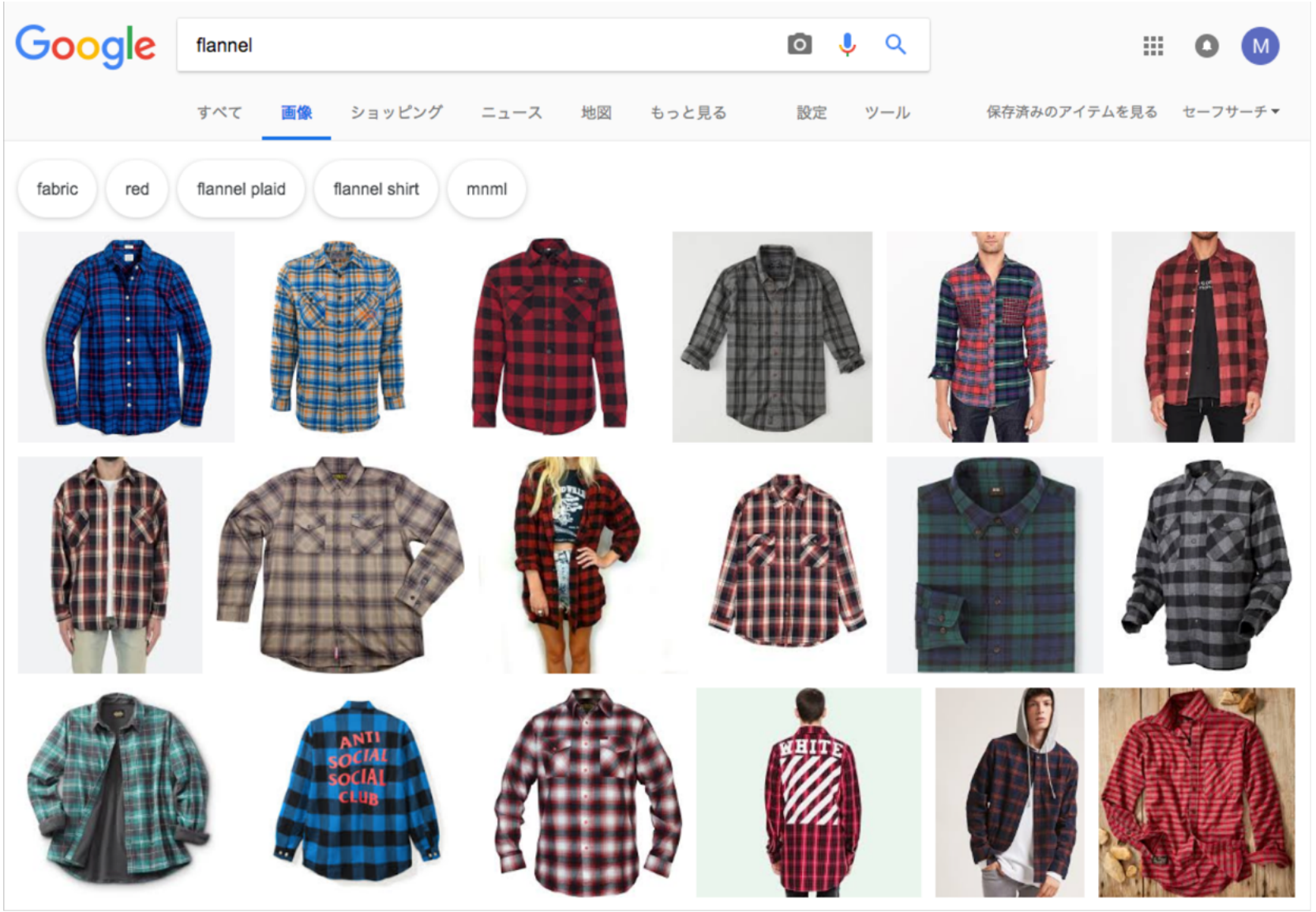
- IP アドレスは Pod に対して振られ、Pod Network の中からアドレスが割り当てられる
- 1 Pod 内に複数のコンテナがある場合でも、それらは同じ namespace 内にあり network interface は共有される
- Pod 内の複数コンテナは localhost を使って通信可能
  - Pod 内のポート番号の調停は必要だが、Pod 間で調停をする必要はない

Flannel

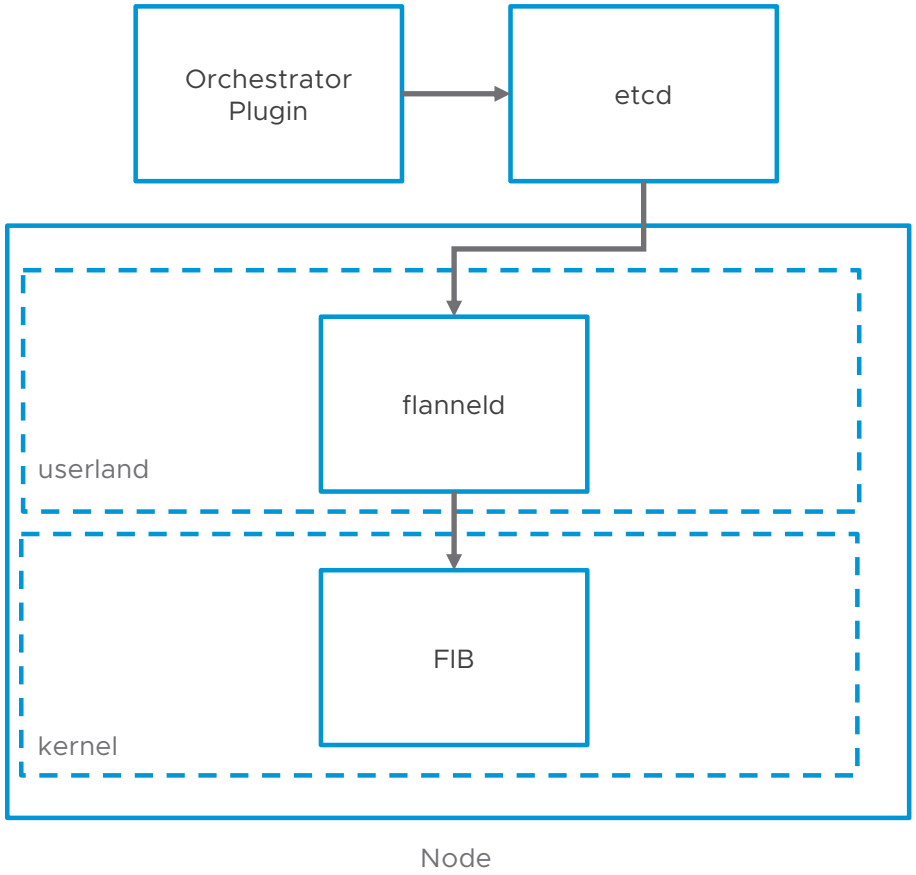
機能
- ネットワーク接続性

特徴
- さまざまな back end をサポート
  - Official: VXLAN, host-gw, UDP
  - Experimental: AliVPC, Alloc, AWS VPCs, GCE, IPIP, IPsec

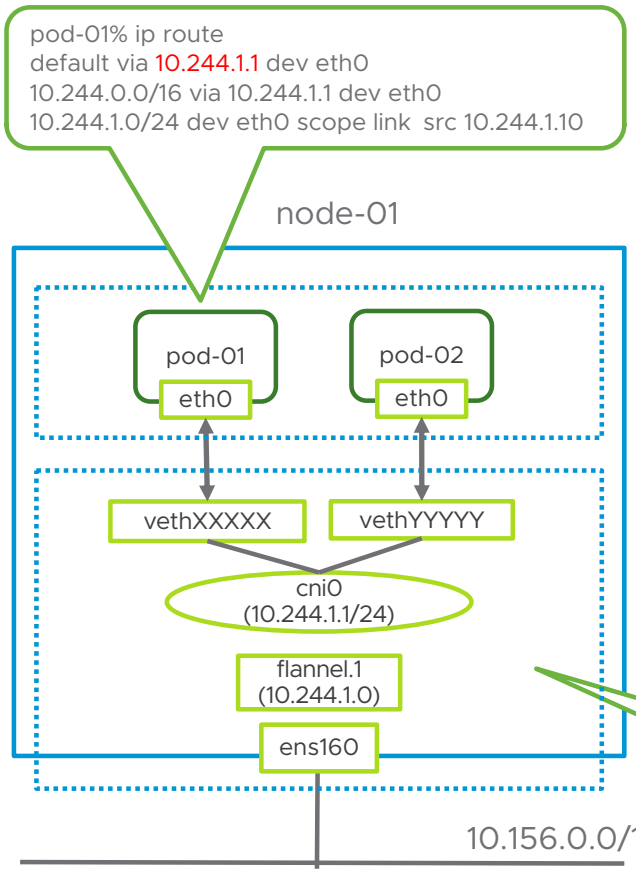

ログイン
Pro
ProLite
▸データ提供：EDP
flannel
検索
クリア
お知らせ あなたのスマホに入ってる？アプリ「英辞郎 on the WEB」（DLは無料です）
変化形：flannels , flanneling , flannelling , flanneled , flannelled
flannel
【自動】
〈英話〉お世辞を言う、ごまをする
【他動】
1. ～をフランネルの生地で覆う
2. 〈英話〉（人）にお世辞を言う、（人）にごまをする
3. 〔過度にもしくは偽善で〕（人）を褒めそやす、（人）にお世辞を言う、（人）にこびへつらう
【名】
1. フランネル、フランネル製のズボン［衣類］
2. 《flannels》フランネルの布、フランネル製品
3. 〈英〉浴用タオル
【形】
フランネル（製）の、フランネル製のズボンをはいた
発音 flǽnl、 カナ フランネル、 変化 《動》flannels | flanneling | flannelling | flanneled | flannelled、 分節
flan・nel



Flannel のアーキテクチャ



Flannel によるネットワーク



- Pod Network CIDR (/16) は各ノード毎に /24 に分割され、各ノードに割り当てられる
- flannel.1 という VXLAN (VTEP) interface が作成され、.0/32 のアドレスが振られる
- cni0 という bridge interface が作成され、.1/24 のアドレスが振られる
- Pod には eth0 が作られ、それとペアとなる veth interface が作られ、cni0 ブリッジに接続される
- 各ノードの Pod Network の next-hop は、そのノードの flannel.1 interface のアドレスの ”onlink” で作られる
- 他ホストの flannel.1 interface の MAC アドレスが静的に ARP テーブルに書かれる
  - ググると L2 / L3 MISS を拾う、という記述が散見されるが、2017年2月の commit 5d3d66425 で大幅に書き換えられており、現在は L2/L3 MISS をフックするような動作はしなくなっている

```
node-01% ip route | grep 10.244
10.244.0.0/24 via 10.244.0.0 dev flannel.1 onlink
10.244.1.0/24 dev cni0 proto kernel scope link src 10.244.1.1
10.244.2.0/24 via 10.244.2.0 dev flannel.1 onlink
node-01% arp -an | grep 10.244
? (10.244.2.0) at 1a:39:5c:fa:9a:f1 [ether] PERM on flannel.1
? (10.244.1.9) at 0a:58:0a:f4:01:09 [ether] on cni0
? (10.244.0.0) at 22:b3:69:58:45:f1 [ether] PERM on flannel.1
```

kubuadm init --pod-network-cidr=10.244.0.0/16

Project Calico

機能

- ネットワーク接続性
- ネットワークポリシー

特徴

- 複数のオーケストレーションに対応（OpenStack, Kubernetes, etc.）
- Encapsulation が必要ない（IP-in-IP を使うことはできる）
- BGP（BIRD, gobgp）で経路を広告
- ポリシーは iptables で実現
- ポリシー部分だけ使うことも可能（e.g. Canal）
- IPv6 対応

ログイン Pro ProLite ▸データ提供：EDP

calico 検索 クリア

お知らせ 「Pro」の単語帳はなんと20,000件！あなただけの単語・例文帳で仕事効率アップ

検索文字列 calico

該当件数：13件

＊ データの転載は禁じられています。

▼例文付きの検索結果を見る

変化形：《複》calicoes , calicos

calico

【名】

1. キャラコ、キャリコ、更紗、サラサ◆光沢のある綿生地で、花や動物柄などの鮮やかなプリントが施されているもの。◆【語源】最初にイギリスに輸出したインドのCalicut市からで、当初は東インドから輸出される綿布全体を指す言葉であった。

2. 〈英〉キャラコ、キャリコ◆安価な無地の綿生地でモスリンより重く目の粗いもの。

3. 〔ネコなどの〕白地に赤と黒のまだら模様の動物、三毛の動物

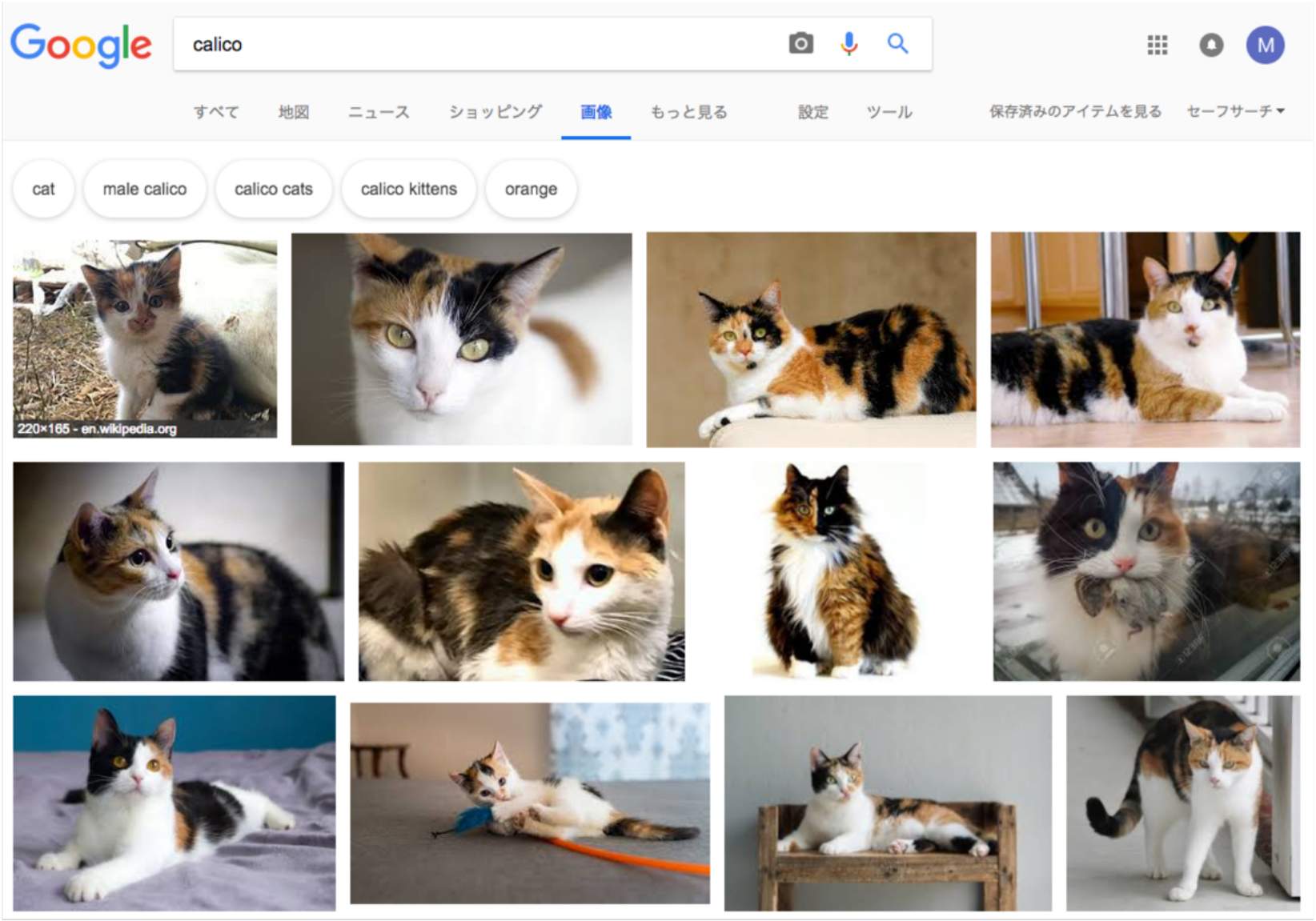
4. 〔金魚の〕キャリコ◆キャリコリュウキン、シュブンキンなどまだら模様のものを指す。

【形】

1. キャラコの、更紗の

2. 〔動物が〕白地に赤と黒のまだら模様の、三毛の

発音 kǽlikòu、 カナ キャラコ、キャリコ、 分節 cal・i・co



ログイン Pro ProLite ▸データ提供：EDP

calico 検索 クリア

お知らせ 「Pro」の単語帳はなんと20,000件！あなただけの単語・例文帳で仕事効率アップ

検索文字列 calico

該当件数：13件

＊ データの転載は禁じられています。

▼例文付きの検索結果を見る

変化形：《複》calicoes , calicos

calico

【名】

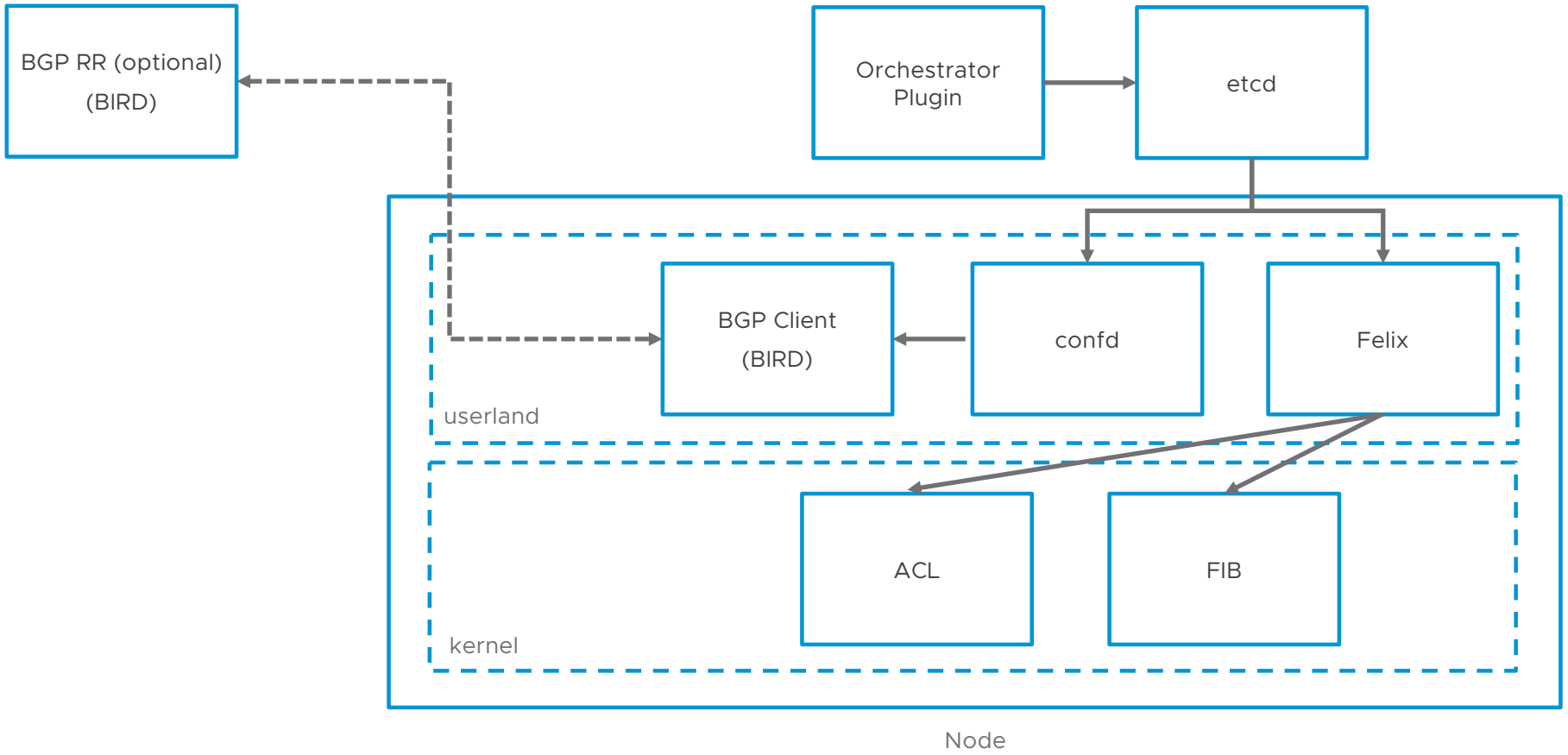
1. キャラコ、キャリコ、更紗、サラサ◆光沢のある綿生地で、花や動物柄などの鮮やかなプリントが施されているもの。
◆【語源】最初にイギリスに輸出したインドのCalicut市からで、当初は東インドから輸出される綿布全体を指す言葉であった。
2. 〔英〕キャラコ、キャリコ◆安価な無地の綿生地でモスリンより重く目の粗いもの。
3. 〔ネコなどの〕白地に赤と黒のまだら模様の動物、三毛の動物
4. 〔金魚の〕キャリコ◆キャリコサバキン、シュブンキンなどまだら模様のものを指す。

【形】

1. キャラコの、更紗の
2. 〔動物が〕白地に赤と黒のまだら模様の、三毛の

発音 kæ̀likòu、 カナ キャラコ、キャリコ、 分節 cal・i・co

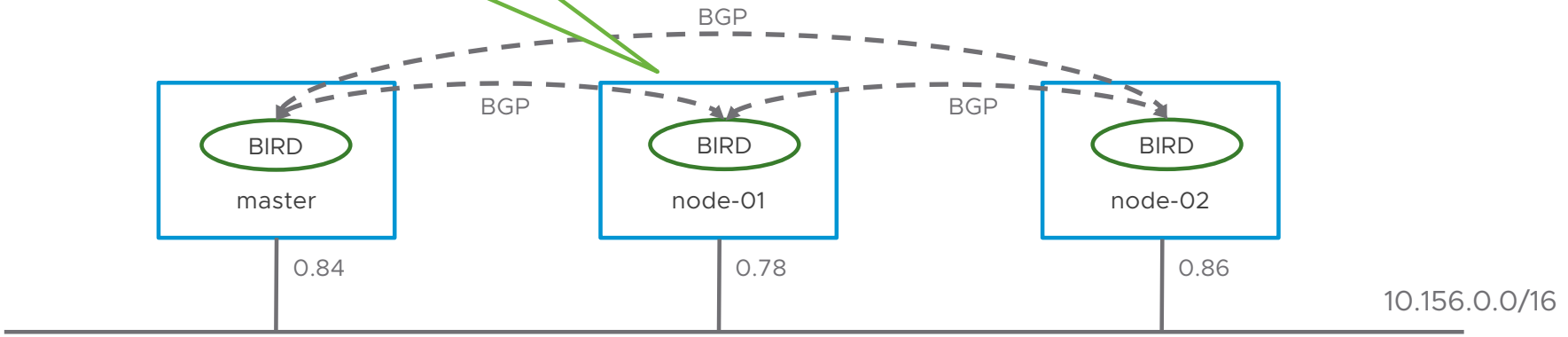
Calico アーキテクチャ（コンポーネント）



Calico によるネットワーク（ノード間）

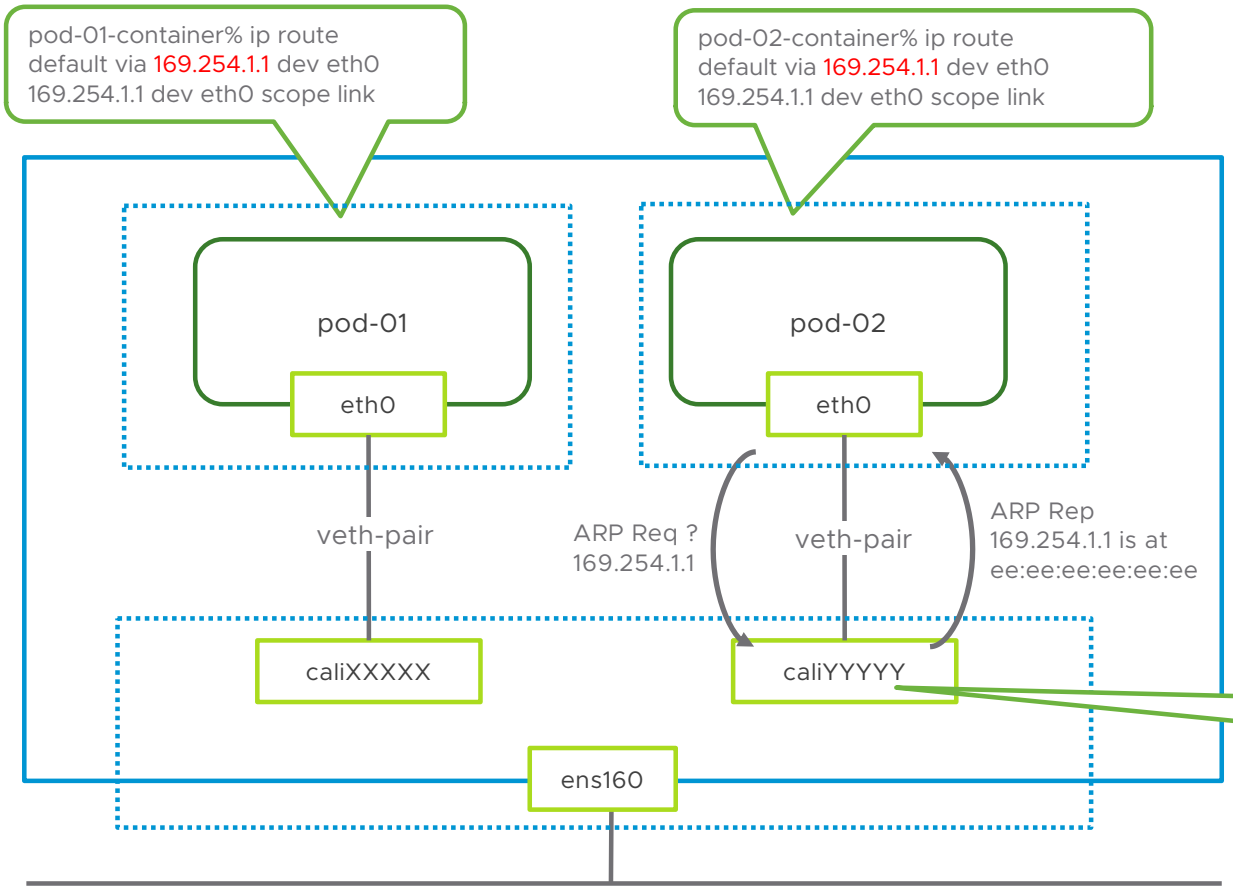
```
node-01% ip route | grep 192.168
blackhole 192.168.0.128/26 proto bird
192.168.0.144 dev caliae434558fa1 scope link
192.168.0.145 dev cali436b19e8d7e scope link
192.168.0.146 dev calid93c954af3f scope link
192.168.221.192/26 via 10.156.250.84 dev ens160 proto bird
192.168.238.0/26 via 10.156.250.86 dev ens160 proto bird
```

- Pod Network CIDR (/16) は各 node 毎に /26 に分割して BGP で広告
  - ググるとホストルート（/32）が広告される、という記述がたくさん見つかるが、少なくとも現在はそうなっていない
- ノードに deploy された Pod 毎に caliXXXXXXX という veth interface が作成される
- 他ノードへの経路は IP-in-IP を使う場合は dev tunl0 に、そうでない場合は通常の ethernet i/f (e.g. dev ens160) を向く



kubuadm init --pod-network-cidr=192.168.0.0/16

Calico によるネットワーク（ノード内）



- どの Pod も default route が link local である 169.254.1.1 を向く
- しかし、誰も 169.254.1.1 というアドレスを持った interface はどこにもない
- なぜ動くか？
  - Host 側の interface (caliXXXXX) に proxy ARP の設定がされている！

```
node-01 % cat \
/proc/sys/net/ipv4/conf/caliYYYYY/proxy_arp
1
```

Kubernetes ネットワークポリシー

いわゆる ACL（Access Control List）機能

- 方向：　ingress / egress
- 対象：　IP Block、Pod (label-based)

Pod 単位にかけられるので、Pod の IP アドレスが変わっても追従することができる（いわゆる Micro Segmentation）

iptablesで実装するケースが多い

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
  namespace: policy-demo
spec:
  podSelector:
    matchLabels:
      run: nginx
  ingress:
    - from:
      - podSelector:
          matchLabels:
            run: access
```

Calico の Kubernetes ネットワーク

```
mshindo@kube-node01:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
cali-INPUT  all  --  anywhere             anywhere             /* cali:Cz_u1IQiXIMmKD4c */
KUBE-EXTERNAL-SERVICES  all  --  anywhere             anywhere             ctstate NEW /* kubernetes externally-visible service portals */
KUBE-FIREWALL  all  --  anywhere             anywhere

(snipped)

Chain cali-FORWARD (1 references)
target     prot opt source               destination
MARK       all  --  anywhere             anywhere             /* cali:vjrMJCRpqwy5oRoX */ MARK and 0xfff1ffff
cali-from-hep-forward  all  --  anywhere             anywhere             /* cali:A_sPAO0mcxbT9mOV */ mark match 0x0/0x10000
cali-from-wl-dispatch  all  --  anywhere             anywhere             /* cali:8ZoYfO5HKXWbB3pk */
cali-to-wl-dispatch  all  --  anywhere             anywhere             /* cali:jdEuaPBe14V2hutn */
cali-to-hep-forward  all  --  anywhere             anywhere             /* cali:12bc6HljsMKsmfr- */
ACCEPT     all  --  anywhere             anywhere             /* cali:MH9kMp5aNICL-Olv */ /* Policy explicitly accepted packet. */ mark match 0x10000/0x10000


Chain cali-INPUT (1 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             /* cali:msRIDfJRWnYwzW4g */ mark match 0x10000/0x10000
ACCEPT     ipencap--  anywhere             anywhere             /* cali:1IRlRis1-pHsGnX5 */ /* Allow IPIP packets from Calico hosts */ match-set cali40all-hosts-net src ADDRTYPE match dst-type LOCAL
DROP       ipencap--  anywhere             anywhere             /* cali:jX63A0VGotRJWnUL */ /* Drop IPIP packets from non-Calico hosts */
cali-wl-to-host  all  --  anywhere             anywhere             [goto]  /* cali:Dit8xicL3zTIYYIp */
MARK       all  --  anywhere             anywhere             /* cali:LCGWUV2ju3tJmfW0 */ MARK and 0xfff0ffff
cali-from-host-endpoint  all  --  anywhere             anywhere             /* cali:x-gEznubq2huN2Fo */
ACCEPT     all  --  anywhere             anywhere             /* cali:m27NaAhoKHLs1pID */ /* Host endpoint policy accepted packet. */ mark match 0x10000/0x10000


Chain cali-OUTPUT (1 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             /* cali:Mq1_rAdXXH3YkrzW */ mark match 0x10000/0x10000
RETURN     all  --  anywhere             anywhere             /* cali:69FkRTJDvD5Vu6Vl */
ACCEPT     ipencap--  anywhere             anywhere             /* cali:AnEsmO6bDZbQntWW */ /* Allow IPIP packets to other Calico hosts */ match-set cali40all-hosts-net dst ADDRTYPE match src-type LOCAL
MARK       all  --  anywhere             anywhere             /* cali:9e9Uf3GU5tX--Lxy */ MARK and 0xfff0ffff
cali-to-host-endpoint  all  --  anywhere             anywhere             /* cali:OB2pzPrvQn6PC89t */
ACCEPT     all  --  anywhere             anywhere             /* cali:tvSSMDBWrme3CUqM */ /* Host endpoint policy accepted packet. */ mark match 0x10000/0x10000


Chain cali-failsafe-in (0 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             /* cali:wWFQM43tJU7wwnFZ */ multiport dports ssh
ACCEPT     udp  --  anywhere             anywhere             /* cali:LwNV--R8MjeUYacw */ multiport dports bootpc
ACCEPT     tcp  --  anywhere             anywhere             /* cali:QOO5NUOqOSS1_Iw0 */ multiport dports bgp
ACCEPT     tcp  --  anywhere             anywhere             /* cali:cwZWoBSwVeIAZmVN */ multiport dports 2379
ACCEPT     tcp  --  anywhere             anywhere             /* cali:7FbNXT91kugE_upR */ multiport dports 2380
ACCEPT     tcp  --  anywhere             anywhere             /* cali:ywE9WYUBEpve70WT */ multiport dports 6666
ACCEPT     tcp  --  anywhere             anywhere             /* cali:I-WQSVBf_IygPR0J */ multiport dports kcd
```

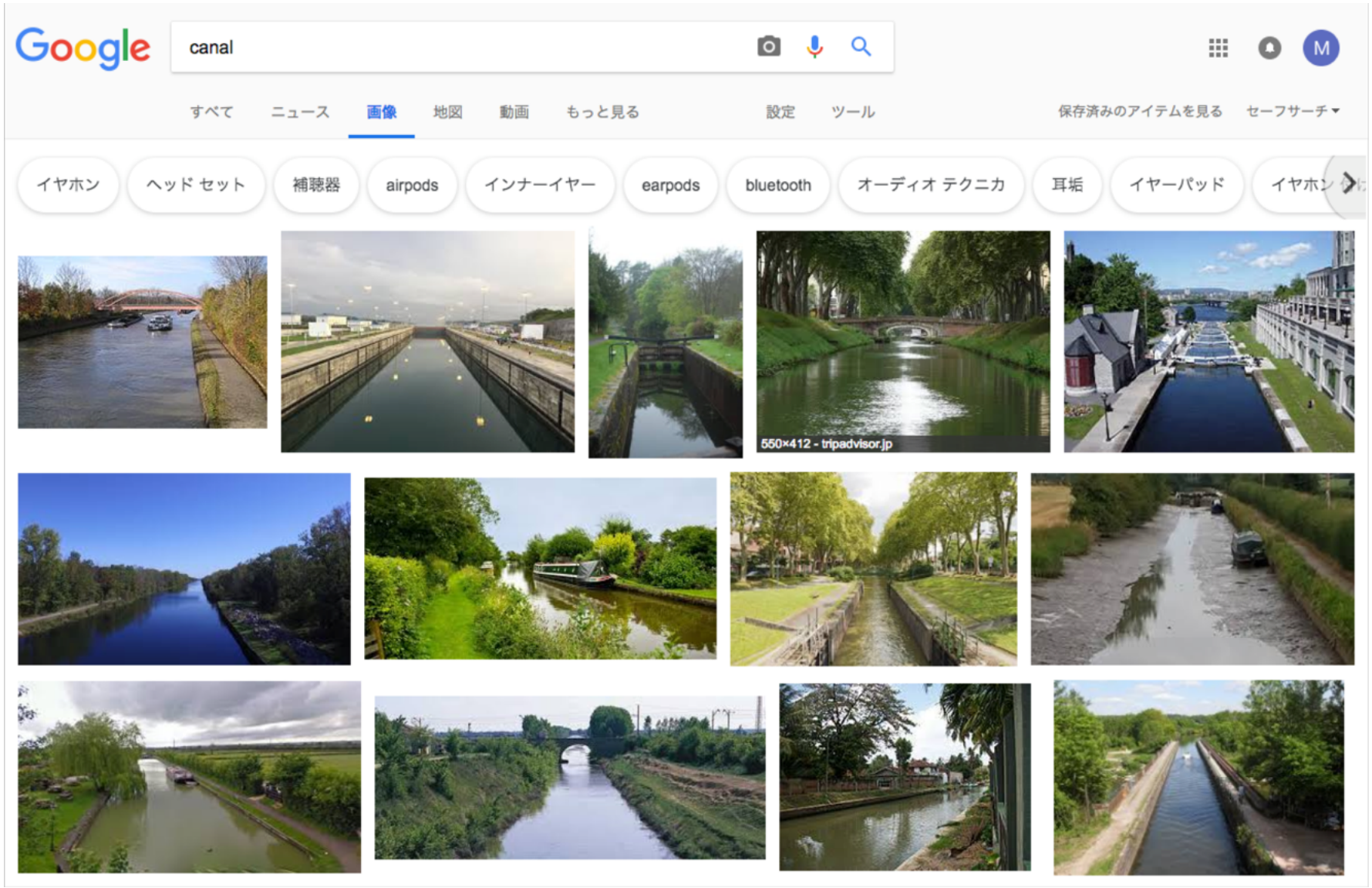
Canal

Calico の ポリシー機能と Flannel の接続性機能を組み合わせた「利用パターン」

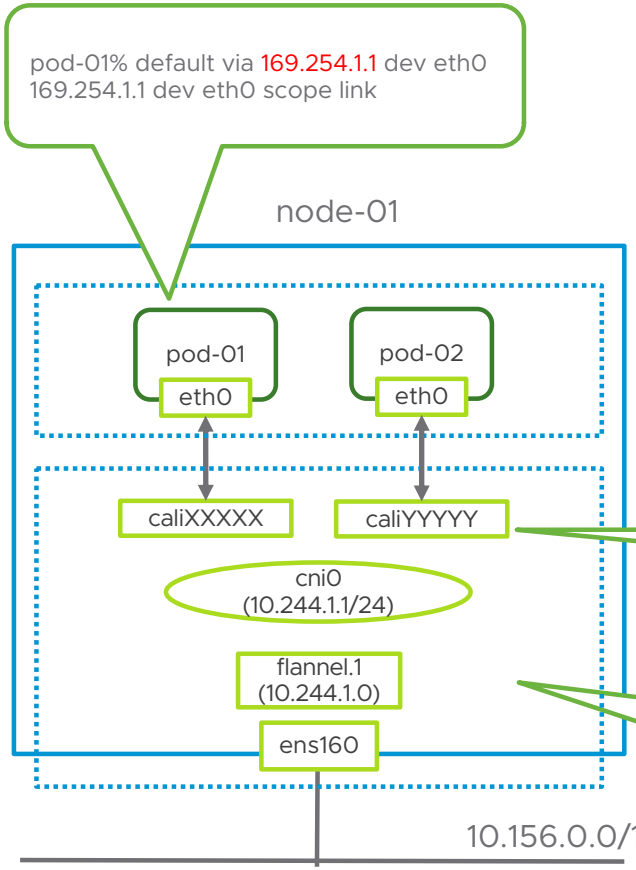
- Calico / Flannel には特に何も手を入れず、そのまま利用

VMware Kubernetes Engine (VKE) は現状 Canal を使っている（OVN に移行する可能性あり）

```
% kubectl get pod -n vke-system
NAME                                                READY     STATUS    RESTARTS   AGE
canal-node-1-10-2-62-87l9w                          3/3       Running   0          1h
canal-node-1-10-2-62-fkbjs                          3/3       Running   0          1h
cluster-autoscaler-1-10-2-62-b77d598d-7l7cx         1/1       Running   0          1h
kube-dns-1-10-2-62-6b898964fc-l6pbf                 3/3       Running   0          1h
kubernetes-dashboard-1-10-2-62-5b46d8b9d6-7lhw9     2/2       Running   0          1h
nginx-ingress-deployment-1-10-2-62-7cf5897767-pdr6l 1/1       Running   0          1h
node-monitor-ds-1-10-2-62-glv2r                     1/1       Running   0          1h
update-controller-ds-1-10-2-62-52flf                1/1       Running   0          1h
```



Canal によるネットワーク



- Pod の eth0 と veth (caliXXXXX) interface のアーキテクチャは Calico のまま
  - デフォルトが 169.254.1.1 を向き、proxy ARP
- cni0 bridge interface は作られるが caliXXXXX interface は繋ぎ込まれない
- 他ノードへの経路のルーティングアーキテクチャは flannel のまま
  - 他ノードへの onlink な経路と静的 ARP 設定
- 「Calico の オーバーレイネットワーク部分を flannel で置き換えたアーキテクチャ」と考えたほうが正しいかも

```
node-01 % cat \
/proc/sys/net/ipv4/conf/caliYYYYY/proxy_arp
1
```

```
node-01% ip route | grep 10.244
10.244.0.0/24 via 10.244.0.0 dev flannel.1 onlink
10.244.1.0/24 dev cni0 proto kernel scope link src 10.244.1.1 linkdown
10.244.1.2 dev calia2a0356c6f4 scope link
10.244.1.3 dev cali4f3e8bf5e26 scope link
10.244.2.0/24 via 10.244.2.0 dev flannel.1 onlink
node-01% arp -an | grep 10.244
? (10.244.2.0) at 1a:39:5c:fa:9a:f1 [ether] PERM on flannel.1
? (10.244.0.0) at 22:b3:69:58:45:f1 [ether] PERM on flannel.1
? (10.244.1.9) at 0a:58:0a:f4:01:09 [ether] on cni0
```

kubuadm init --pod-network-cidr=10.244.0.0/16

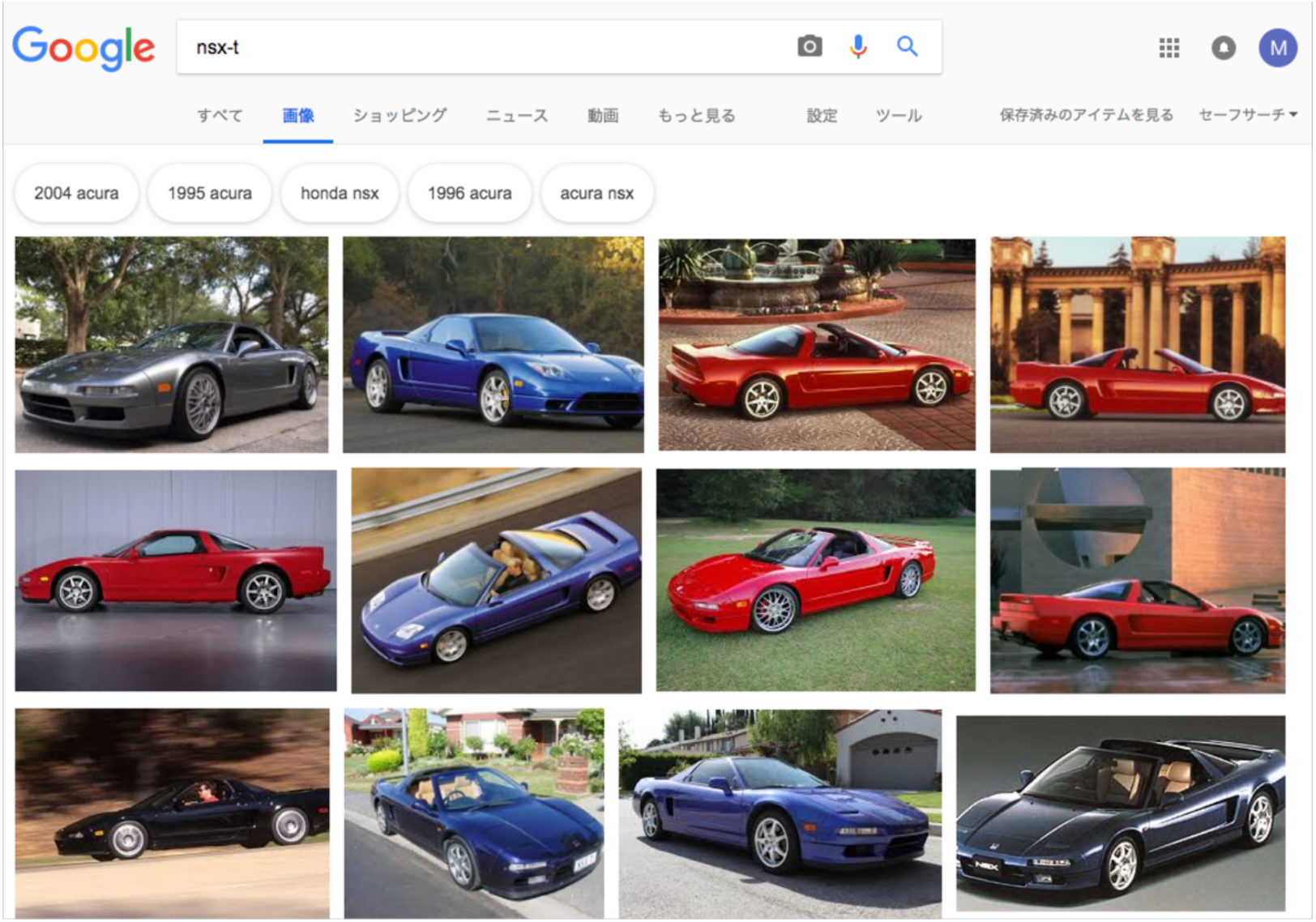
NSX-T Container Plugin (NCP)

機能

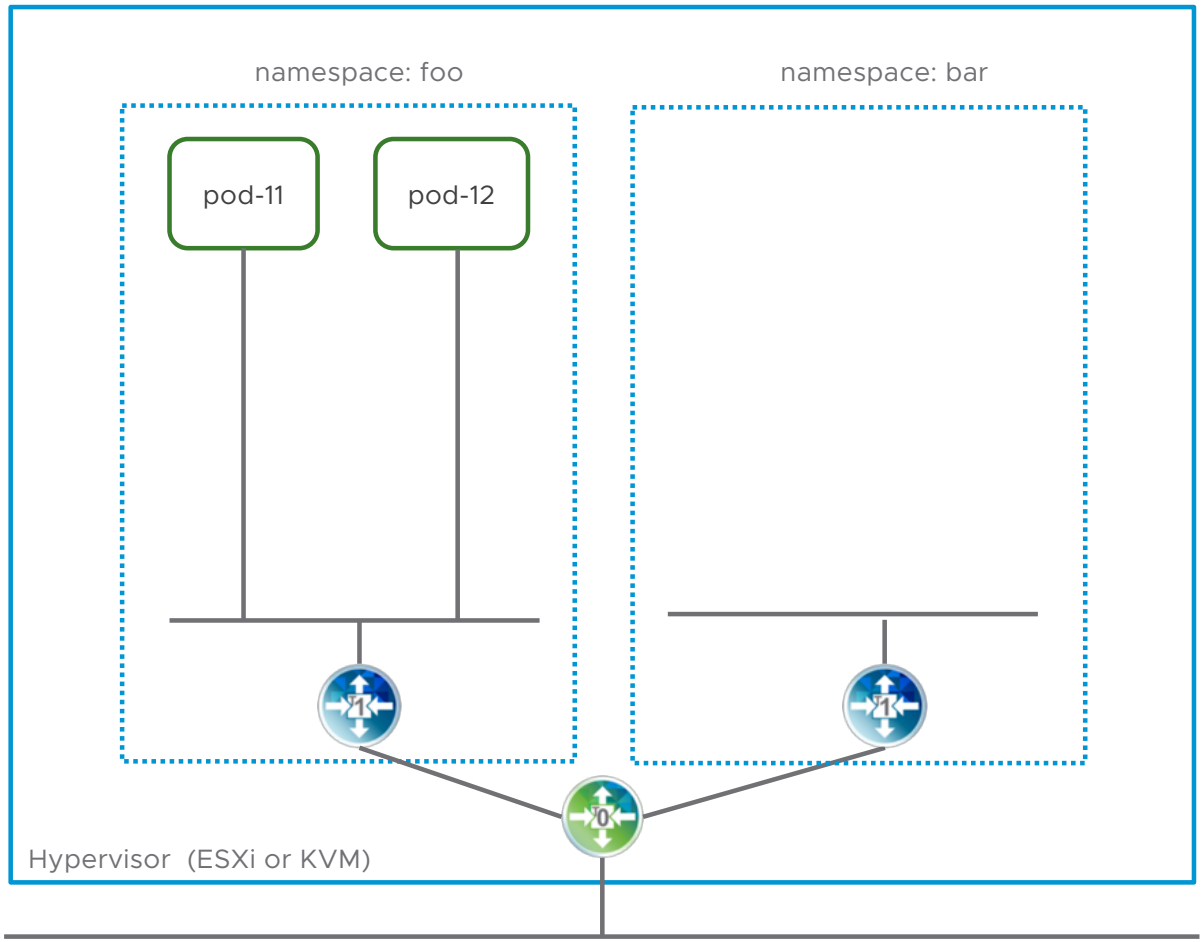
- ネットワーク接続性
- ネットワークポリシー
- その他

特徴

- ネームスペースとの連動
- 可視化＆トラブルシュート機能（カウンタ、ミラーリング、Traceflow、等）
- VM ワークロードとの共存
- Load Balancer を提供

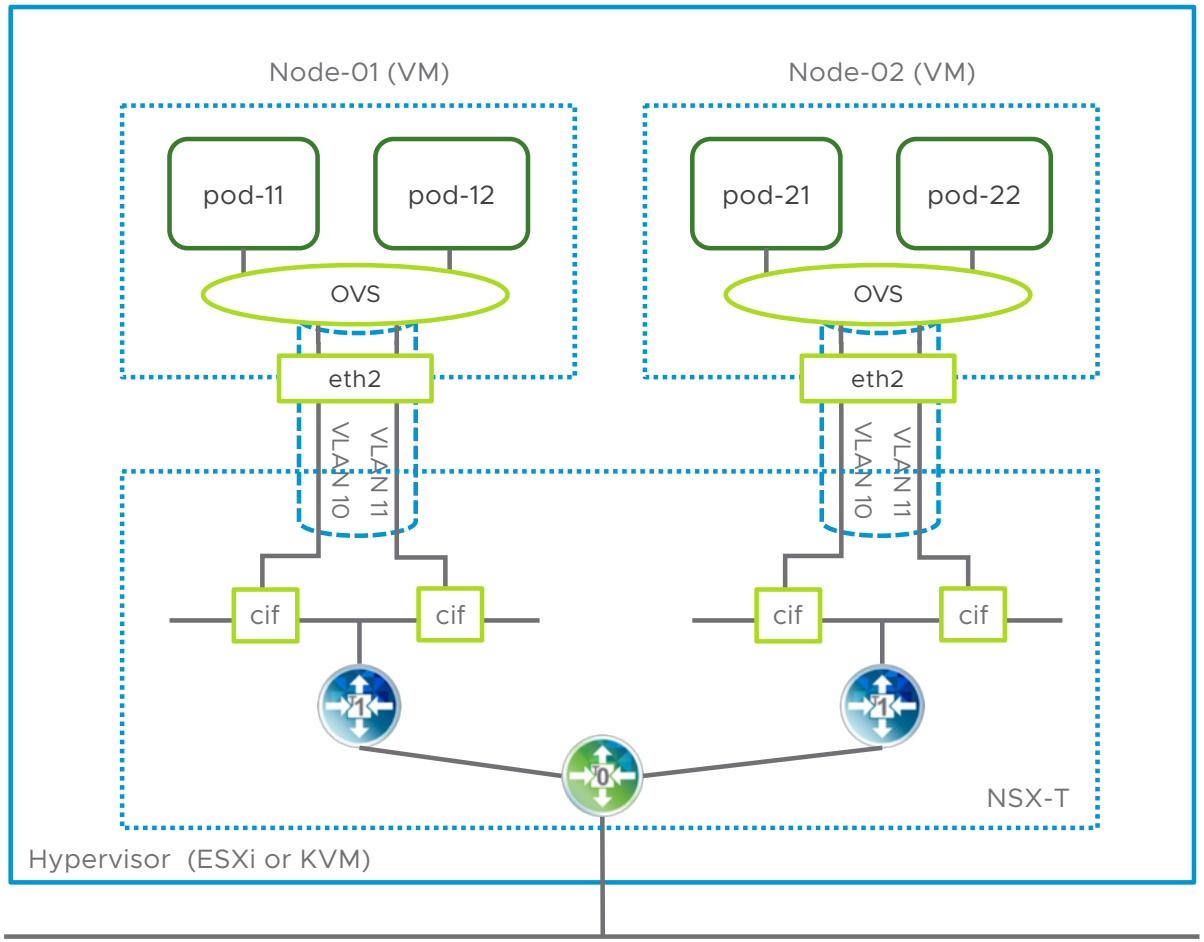


NSX-T NCP によるネットワーク（namespace との連動）



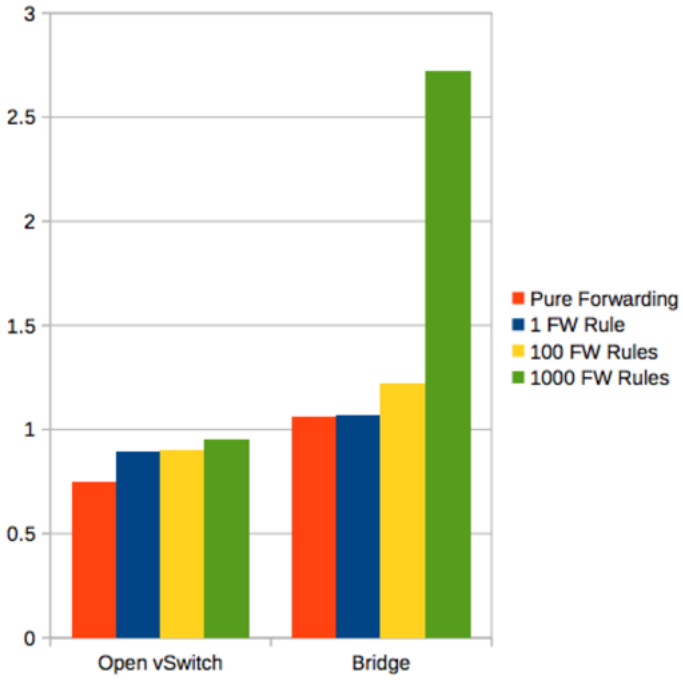
- NCP は K8S の namespace を監視
  1. namespace が作成される
  2. NSX-T が持つ IP block からサブネットが割り振られる
  3. 論理スイッチが作成される
  4. T1 ルータが作成され T0 ルータと接続される
  5. T1 にルータポートが作成され、論理スイッチに接続され、IP アドレスがサブネットから振られる
  6. NAT するなら、T0 ルータに SNAT のルールを設定する

NSX-T NCP によるネットワーク（wiring）



- Node 外に仮想ネットワーク機能 (NSX-T) があるので、Node でオーバーレイを終端する必要がない
- Node VM の中で動作している OVS に VLAN Trunking してやるだけ
  - OVS は Standalone モードで動作し、NCP によりプログラムされる
- Cif (Container I/F) に対して DFW を適用できるので、Pod 単位に Micro-Segmenation できる

OVS vs iptables パフォーマンス比較



Number of gigacycles per second required to saturate a 10Gbps link with netperf TCP_STREAM. (Lower is better)

- Preliminary results are quite promising
  - OVS+conntrack uses a nearly consistent rate regardless of number of rules
  - Bridge+iptables uses more CPU as rule count increases

Open vSwitch

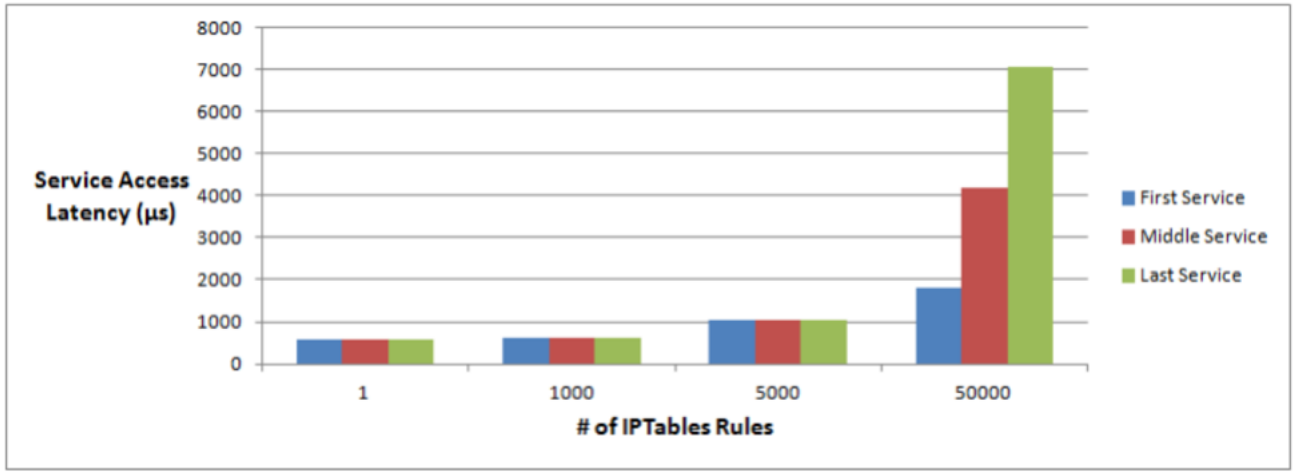
出典：http://www.openvswitch.org/support/ovscon2014/17/1030-conntrack_nat.pdf

iptables による Service Routing のパフォーマンス

Where is latency generated?

- Long list of rules in a chain
- Enumerate through the list to find a service and pod

In this test, there is one entry per service in KUBE-SERVICES chain.



	1 Service (µs)	1000 Services (µs)	10000 Services (µs)	50000 Services (µs)
First Service	575	614	1023	1821
Middle Service	575	602	1048	4174
Last Service	575	631	1050	7077

出典： https://www.slideshare.net/LCChina/scale-kubernetes-to-support-50000-services

iptables による Service Routing のパフォーマンス

- Where is the latency generated?
  - not incremental
  - copy all rules
  - make changes
  - save all rules back
  - IPTables locked during rule update
- Time spent to add one rule when there are 5k services (40k rules): 11 minutes
- 20k services (160k rules): 5 hours

出典： https://www.slideshare.net/LCChina/scale-kubernetes-to-support-50000-services

ovn-kubernetes

OVN （Open Virtual Network）は、Open vSwitch（OVS）チームが中心にオープンソースで開発が進められている仮想ネットワークスタック

機能

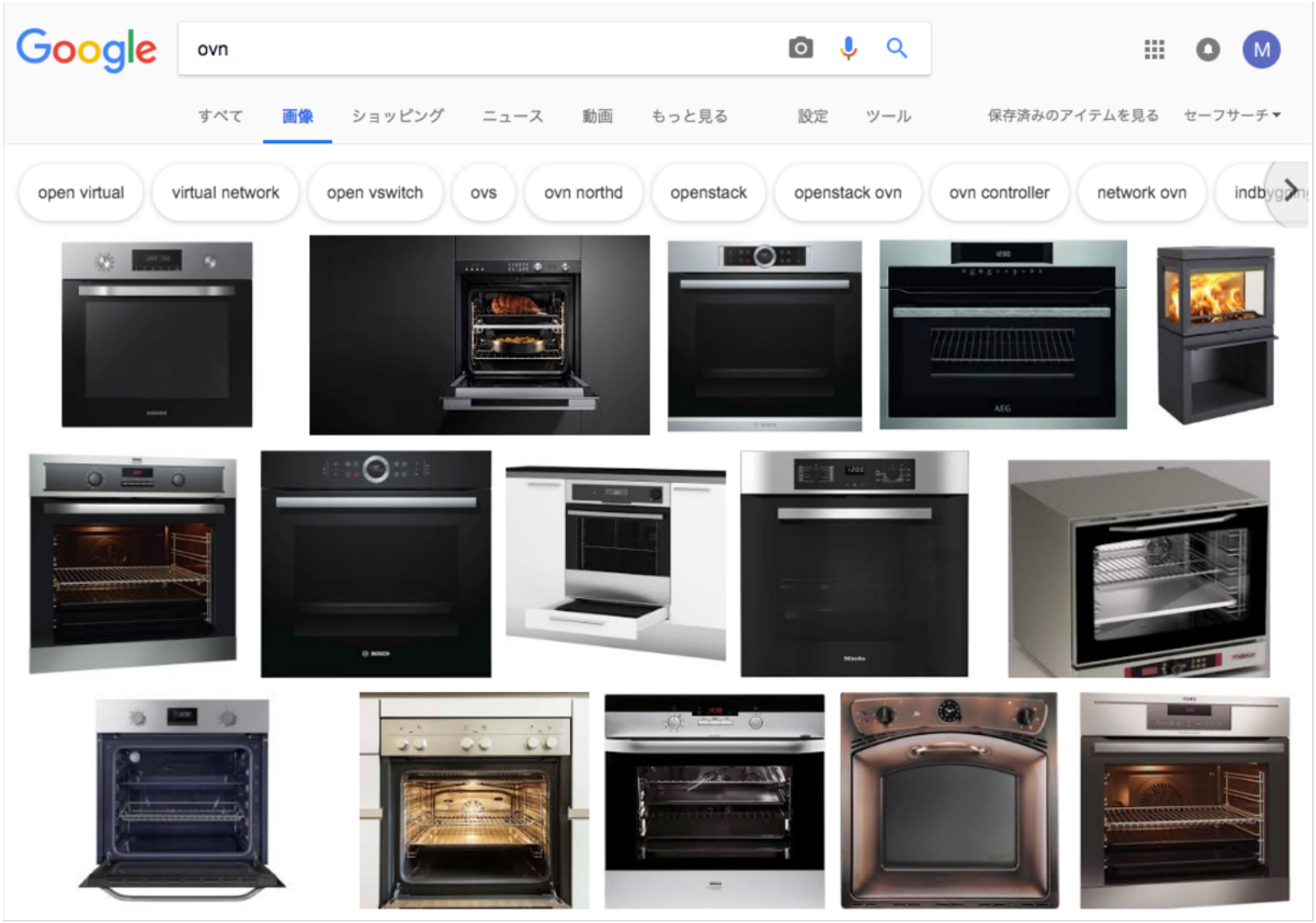
- ネットワーク接続機能

特徴

- シンプルなアーキテクチャ
  - 仮想スイッチをノードごとに1つ作り、それらを仮想ルータで束ねる

OVN の動作

力尽きてしまいしました・・・・・



比較表

		Flannel	Calico Project	Canal	NSX-T Container Plugin (NCP)
Datastore		K8S API (recommended), etcd	etcd、K8S API (beta)	K8S API (recommended), etcd	NSX Manager
Overlay		Production: VXLAN, none (host-gw)、Experimental: IP-in-IP, IPsec	None, IP-in-IP	VXLAN, IP-in-IP, VCP routing (not recommended)	Geneve (outside of worker node)
Network Policy	Mechanism	-	Iptables / Istio	Iptables / Istio	OVS / (n)vDS
	K8S Policy	-	Yes	Yes	Yes
	App Policy	-	HTTP method/path (requires Istio)	HTTP method/path (requires Istio)	No
Load Balancer	N-S	-	-	-	NSX LB
	E-W	iptables	iptables	iptables	OVS datapath
VM Support		No	No	No	Yes
Visibility Tools		External	External (prometheus)	External (prometheus)	Built-in (Traceflow, Mirroing, etc.)
Commercial Support		No	Yes	No?	Yes

Key Takeaways

コンテナ・ネットワーキング、闇深すぎ ☺

動きが早いので、Web の情報を信用してはいけない

スケーラビリティには注意しよう！

今回テストに使ったバージョン

Ubuntu: 18.04 (bionic)

Kubernetes: v1.11.2

Docker: 18.06.1-ce

Flannel: v0.10.0

Calico: 3.2.1

Canal: Calico 2.6.2 + Flannel v0.9.1

NSX-T: 2.1.0.0.7395503

ご静聴ありがとうございました

vmware