

Open Networking Conference Japan 2020

潜水艦に乗ってトンネルをくぐろう ～Submarinerによる マルチKubernetesクラスターネットワーキング～

v1.2

Manabu Ori (@orimanabu)

Red Hat

October 15th, 2020

目次

- Submarinerとは
- アーキテクチャ
- クラスタをまたいだPod間通信
- インストール方法
- (時間があればデモ)
- Submarinerの今後



※ 本資料は、Submariner v0.6.1を対象に調査した内容に基づいて作成しました

自己紹介

- 氏名: 織 学 (@orimanabu)
- 所属: Red Hat
- 仕事: OpenStack, Kubernetes(OpenShift), Ansible等のコンサルティング

Submarinerとは



ハイブリッドクラウドしてますか？

- オンプレミス、プライベートクラウド、パブリッククラウド、適材適所で使い分けたり、連携させたい ...ですよね
 - いいものを安くつまみ食いしたい (特にマネージドサービス)
 - パブリッククラウド、たまに「クラウド全体」障害が発生して怖い
 - パブリッククラウド使うと(いろいろ便利すぎて)ロックインされそうで...
 - ある種のデータはやっぱり手元に置いておきたい
 - 特殊なハードウェア構成がどうしても必要
 - 基本オンプレだけど、負荷のスパイク時だけパブリッククラウドのリソースを使いたい

KubernetesクラスターをまたいだPod間通信

- ユースケース
 - スケーラビリティの向上
 - 災害対策
 - セキュリティ
 - Istio Multicluster
 - Shared control plane
 - FrontendはEdge、DBはオンプレ
 - パブリッククラウドのマネージドサービスと連携
- 今できること
 - SDNソリューションいろいろ
 - Calico
 - VMware NSX-T? , Juniper Contrail? , etc...
 - 全クラスターで同じCNI Pluginを使っていれば...
 - そしてSubmariner

Submariner

- 複数のKubernetesクラスターにまたがったPod/Service間通信を実現する仕組み
- 各クラスターにGatewayノードを設置し、Gateway間でIPsecトンネルを張る
 - IPsec周りの設定はプラグイン形式
 - strongSwan
 - LibreSwan
 - Wireguard VPN
- CNI Pluginに依存しない
 - Weave, Calico, Canal, Flannel, OpenShift-SDN 等でテスト済み
- 複数クラスターにまたがるサービスディスカバリ (Lighthouse)
- 各クラスターでPod/Serviceネットワークのアドレスブロックが重複しても大丈夫 (GlobalNet)
- Kubernetes Multi Cluster Sigのプロジェクトに登録してもらうよう提案中

- L3 connectivityを提供するのみ、Service Meshではない

歴史

- 2017: コンセプト考案 (Rancher)
- 2018: 最初のプロトタイプ実装 (Rancher)
- 2019年3月: Submariner v0.0.1リリース
- (この資料は v0.6.1 を対象に作成しました)

Submariner

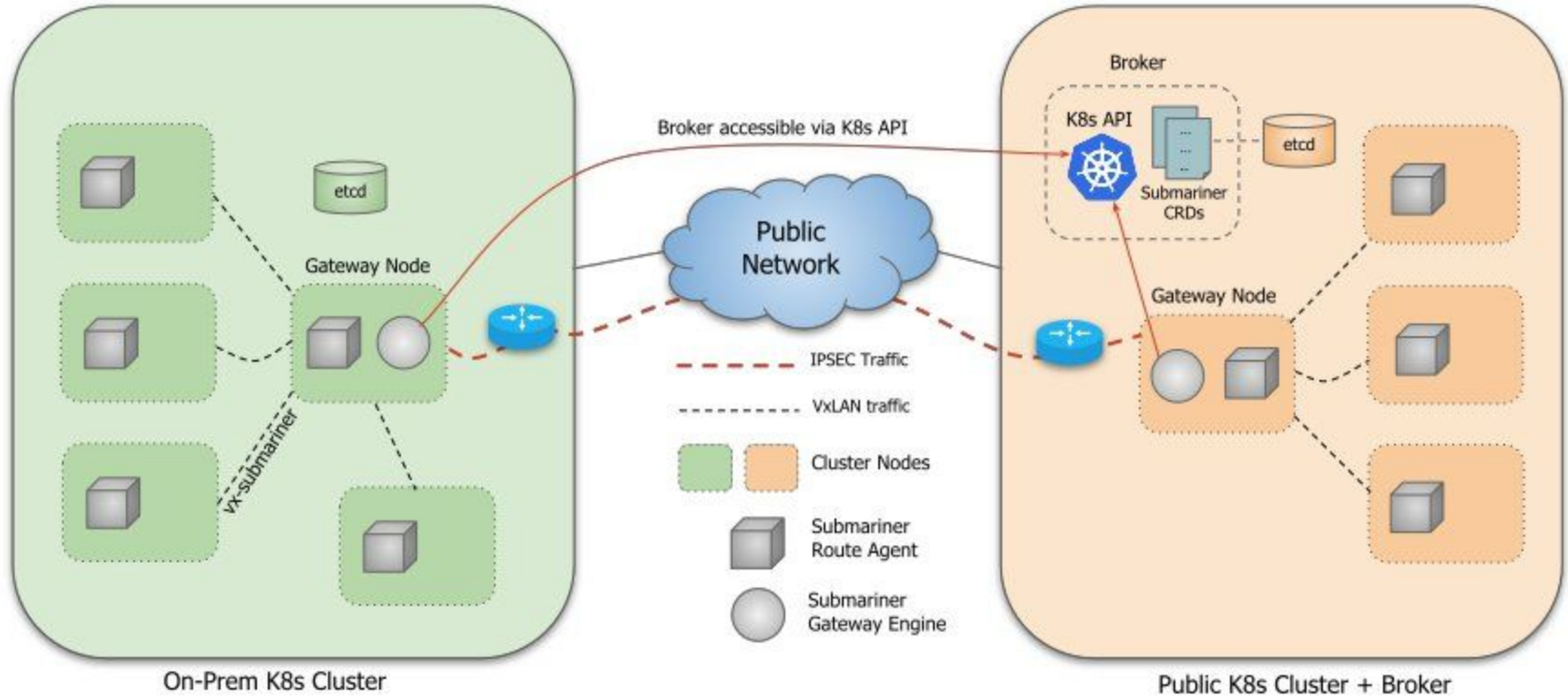
End to End Tests passing Unit Tests passing Linting passing Release Images passing Periodic passing

Submariner is a tool built to connect overlay networks of different Kubernetes clusters. While most testing is performed against Kubernetes clusters that have enabled Flannel/Calico/Canal/Weave/OpenShiftSDN, Submariner should be compatible with any CNI cluster network provider, as it utilizes off-the-shelf components to establish encrypted tunnels between each Kubernetes cluster.

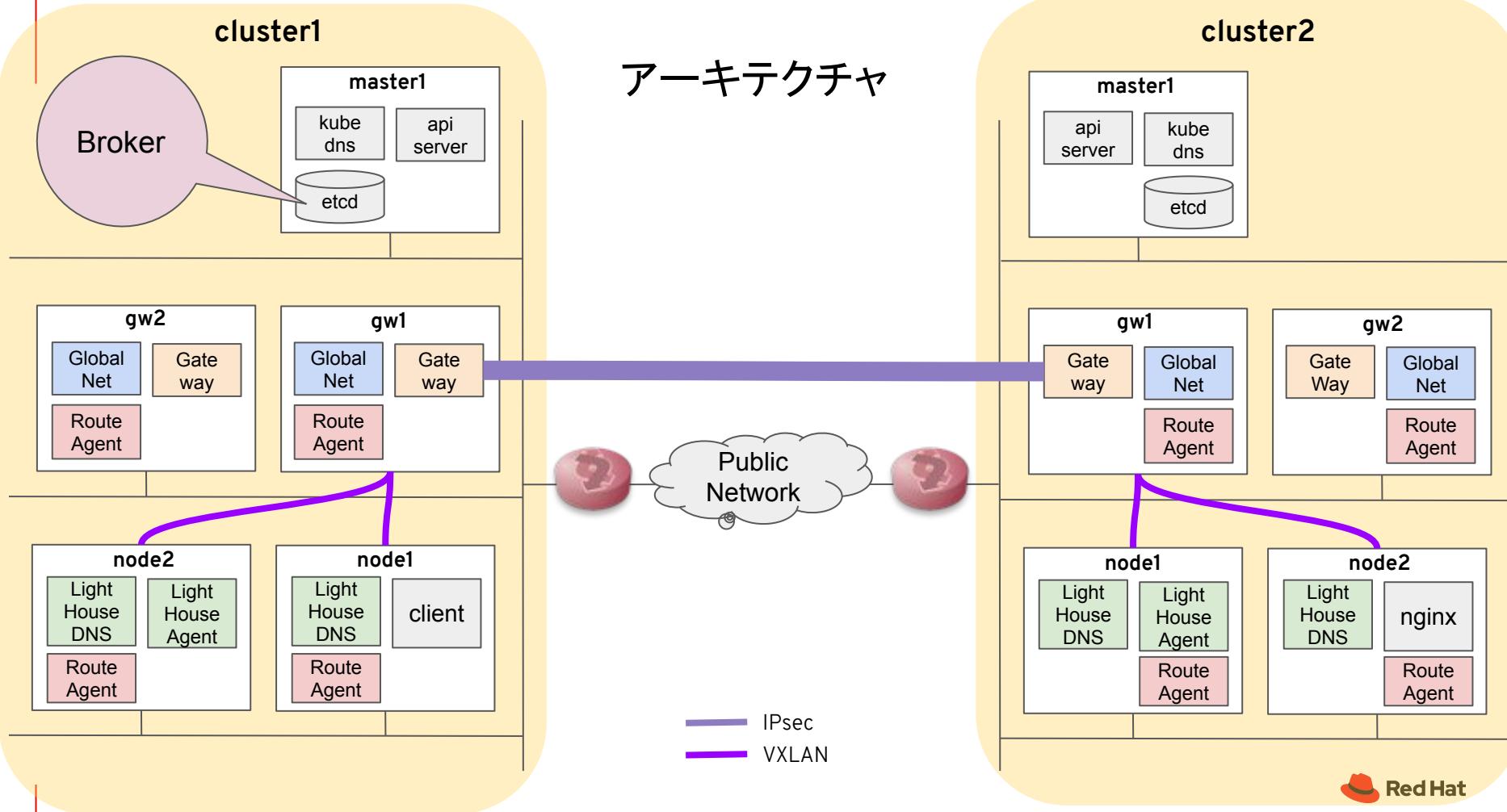
Note that Submariner is in the **pre-alpha** stage, and should not be used for production purposes. While we welcome usage and experimentation, it is quite possible that you could run into bugs.

アーキテクチャ

アーキテクチャ



アーキテクチャ



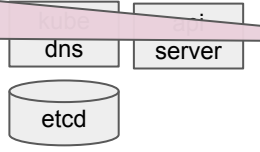
— IPsec
— VXLAN

Broker

- クラスタ間でメタデータ情報を交換するための場所 /API
 - 具体的には、KubernetesのAPIサーバー、CRD、etcd
- Submarinerで接続するクラスタの中にあっても独立していてもよい
 - ただし、Submarinerで接続するクラスタの GatewayノードからAPIアクセスできる必要がある
- “submariner-k8s-broker” namespace
 - Cluster
 - MultiClusterService
 - ServiceImport
 - Endpoint
 - RBAC関連

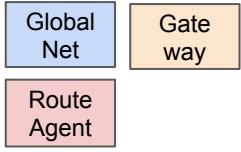
cluster1

master1

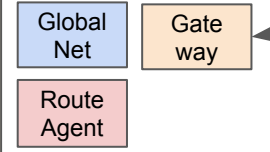


Brokerは独立した
クラスターでも
よい

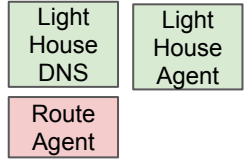
gw2



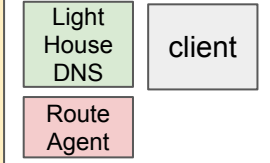
gw1



node2

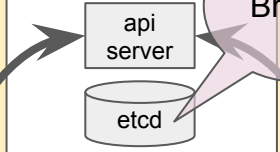


node1



cluster3

master



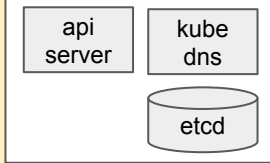
Broker

Public Network

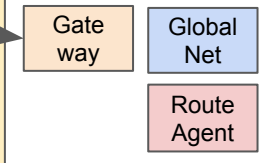
Gatewayノードから
APIアクセスできる
必要がある

cluster2

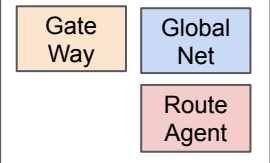
master1



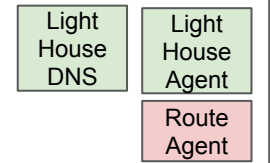
gw1



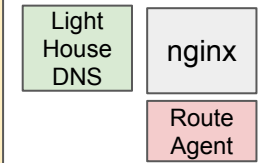
gw2



node1

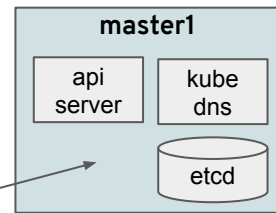
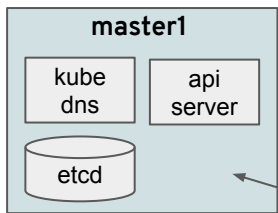


node2

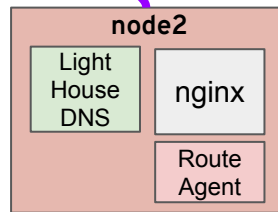
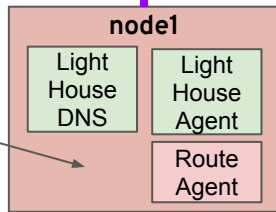
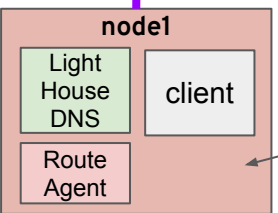
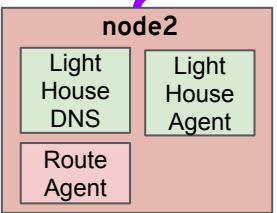
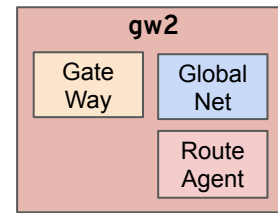
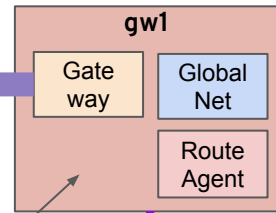
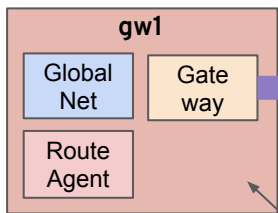
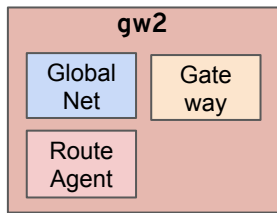


cluster1

cluster2



Master node



Worker node



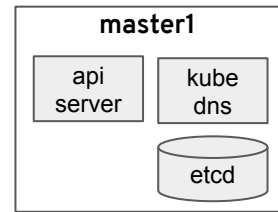
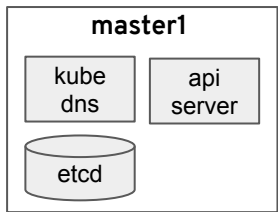
- IPsec
- VXLAN

Gatewayノード

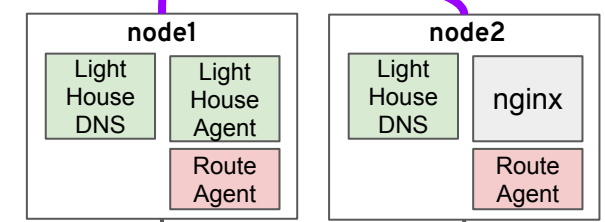
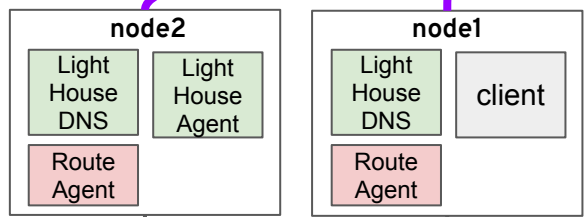
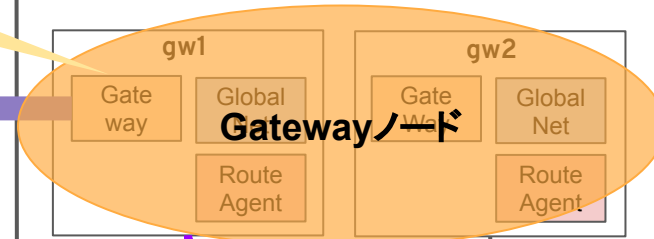
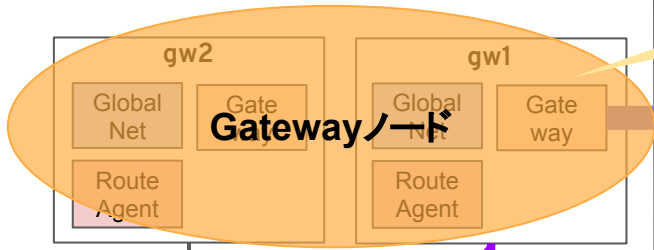
- IPsecの接続を行うゲートウェイとなるノード
 - クラスタをまたがった通信は全てここを経由する
 - Gatewayノードにスケジュールされる Pod
 - Gateway Pod: IPsec接続を行う
 - GlobalNet Pod: GlobalNetの機能を担う (後述)
 - “submariner.io/gateway=true” というラベルをつけたノードが Gatewayノードとなる
 - クラスタ内に複数の Gatewayノードが存在する場合は、Leader Electionの仕組みにより1台だけActiveとなり、このノードがIPsec接続を行う
 - 他のWorkerノードは、ActiveなGatewayノードとVXLANのトンネルを張っている
- 障害発生時の動き
 - 障害検知 → Leader Electionで新しいLeaderを選出
 - 各クラスタの Gatewayノードは、新しいLeaderとIPsecトンネルを張り直す
 - クラスタ内の Workerノードは、新しいLeaderとVXLANトンネルを張り直す

cluster1

cluster2



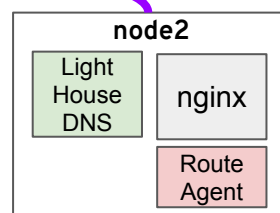
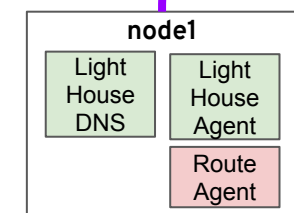
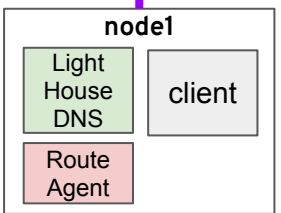
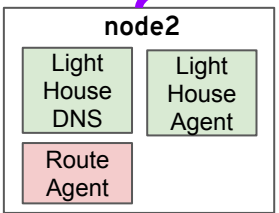
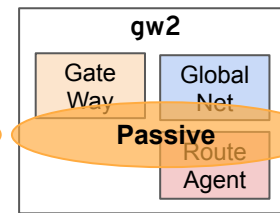
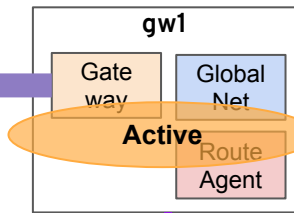
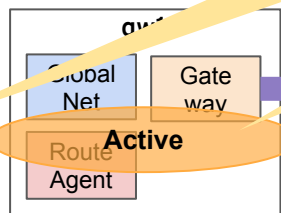
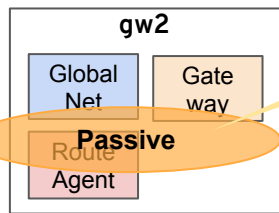
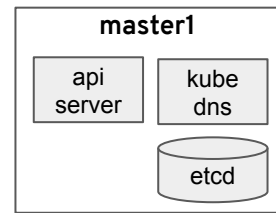
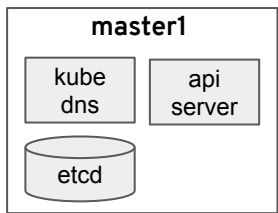
“submariner.io/gateway =true” というラベルが付与されたノードはGatewayノードとなる



- IPsec
- VXLAN

cluster1

cluster2

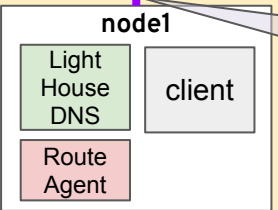
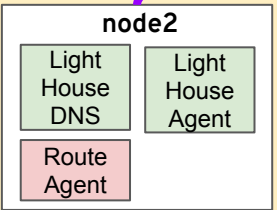
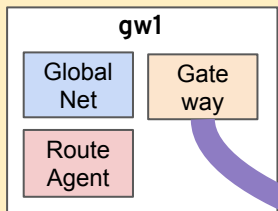
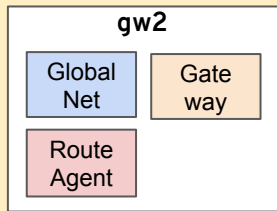
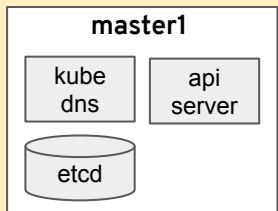


クラスター内に複数のGateway
ノードが存在する場合は、Leader
Electionの仕組みにより1台Active
ノードが選定される

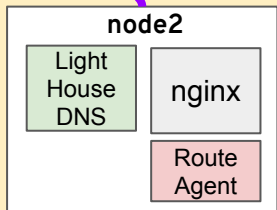
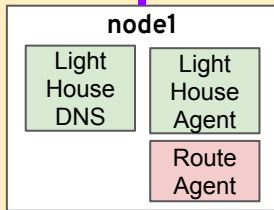
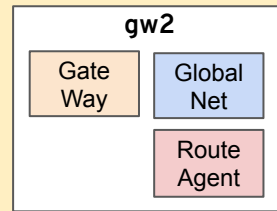
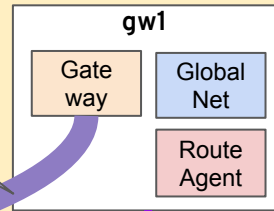
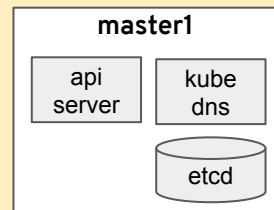


— IPsec
— VXLAN

cluster1



cluster2



ActiveなGatewayノードのsubmariner-gateway Pod間でIPsec接続

Public Network

各WorkerノードはActiveなGatewayノードとVXLAN接続

- IPsec
- VXLAN

Gateway Engine

- IPsecの接続を実施するPod
- Host Networkを使用
- IPsec接続の実際の処理は、プラグイン形式で選択できる (“cable driver”)
 - strongSwan
 - Submarinerのデフォルト
 - Vici使ってUnix Socket経由でcharonを操作する
 - LibreSwan
 - whack, plutoコマンドで設定する
 - WireGuard
 - wgctrlを使ってNetLink経由で設定する

GlobalNet

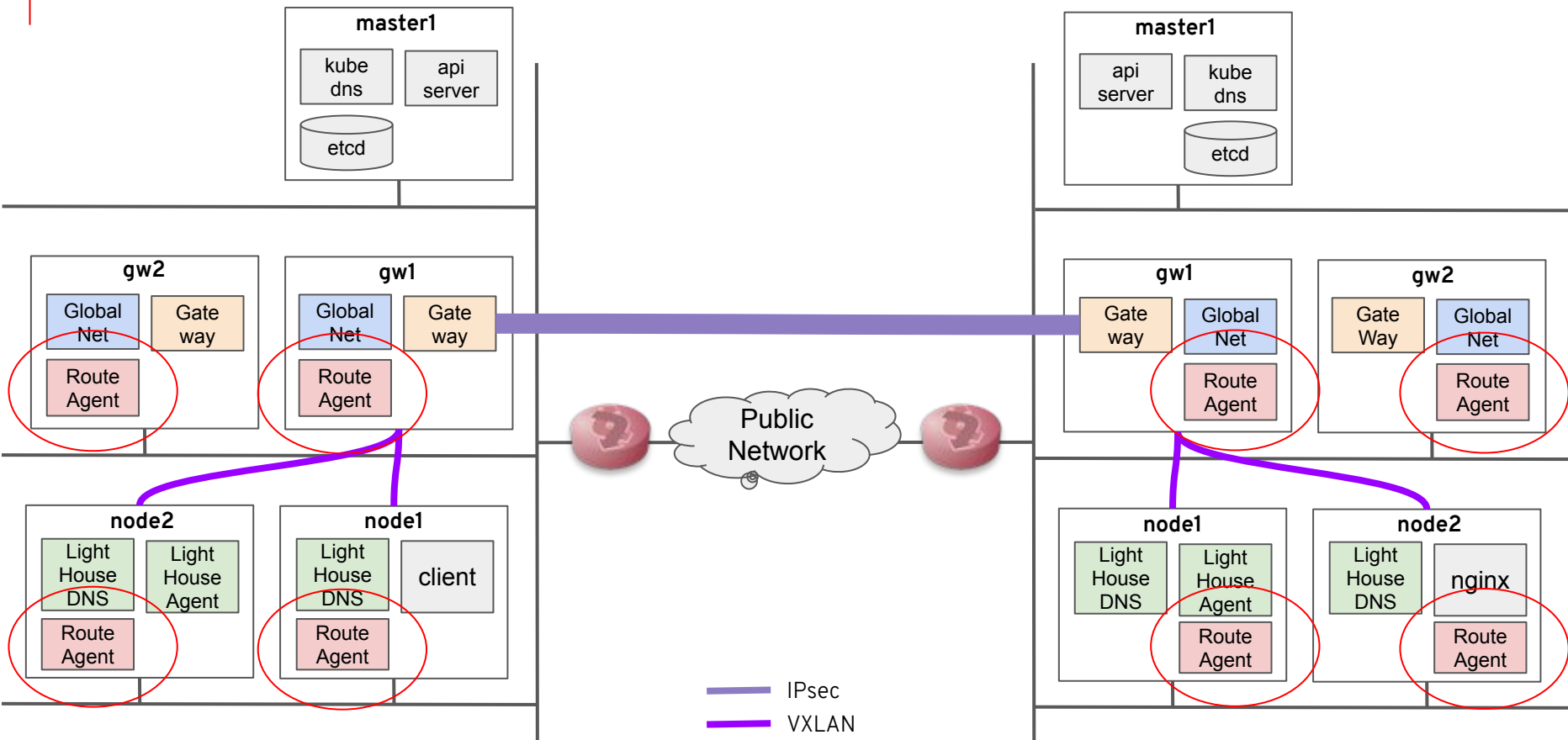
- GlobalNetとは: 複数クラスターにまたがったプライベートアドレスブロック
 - Serviceを他のクラスターに公開すると、GlobalNetからIPアドレスをアサイン (Global IP)
 - 他のクラスターから公開 Serviceにアクセスする場合は、Global IP宛てに通信する
 - GlobalNet PodがGlobal IP ⇔ ClusterIPのアドレス変換ルールを iptablesで設定
 - デフォルトは169.254.0.0/16
 - Calicoを使う場合はGlobalNetのアドレスブロックを変えること
 - Pod/Service用のアドレスレンジがクラスター間で重複していても大丈夫！
 - pod_network_cidrとかservice_cidrとか、デフォルトのまま使うことが多いですよね ...
- GlobalNetを使う場合と使わない場合の細かい違い
 - 使う場合: Service宛てにしかクラスターまたぎの通信はできない
 - 使わない場合: Service宛てでもPod直接でもどちらでもクラスターまたぎの通信ができる

GlobalNet controller

- GlobalNetを管理するPod
 - GlobalNet機能が有効なときのみ、activeなGatewayノードで稼動
- 2つの機能
 - IPAM
 - 各ノードごとにGlobalNetから切り出されたアドレスレンジから、Pod/Service生成時にGlobalIPを割り当て
 - GlobalIPを割り当てたPod/Serviceにannotateを付与
 - submariner.io/globalip=<global-ip>
 - Pod/Service消滅時は割り当てたGlobalIPをプールに戻す
 - アドレス変換
 - Gatewayノードにおいて、iptablesルールを追加
 - PodからのソースIPアドレスをそのPodのGlobalIPに変換するEgress SNATルール
 - Serviceに割り当てたGlobalIP宛ての通信をkube-proxyのService用chainに向けるルール

cluster1

cluster2

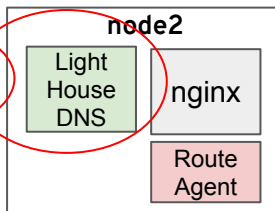
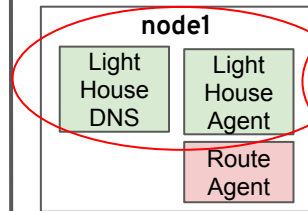
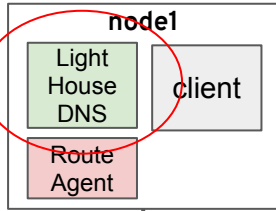
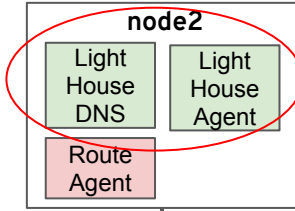
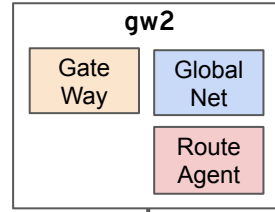
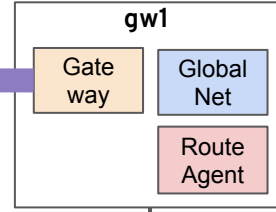
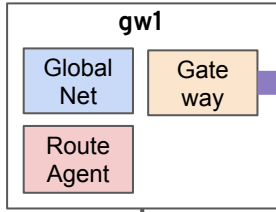
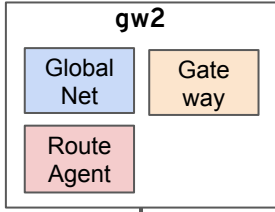
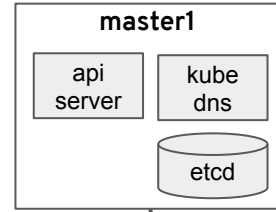
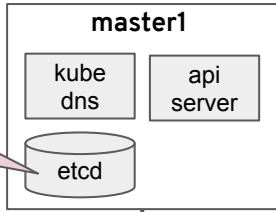
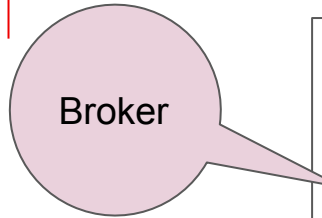


Route Agent

- 全てのWorkerノードで稼動するPod (DaemonSet)
- 各WorkerノードとActiveなGatewayノードとの間でVXLANを張り、別クラスター宛ての通信をGatewayノードにルーティングする
 - VXLANインターフェースのMTUを「ホストインターフェースのMTU-VXLANヘッダ」に設定する
- BrokerのEndpointリソースを使ってルーティングとiptablesの設定を行う
- Gatewayノードのfailoverが発生すると、Route Agentは新たなActive GatewayノードとVXLANを張り直し、ルーティングテーブルの設定を更新する

cluster1

cluster2

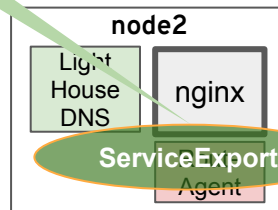
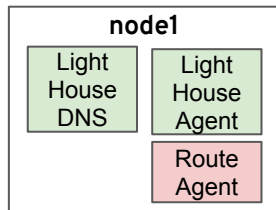
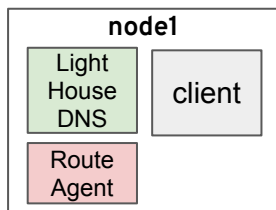
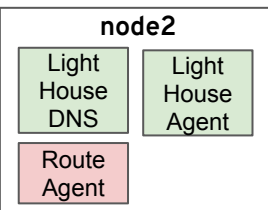
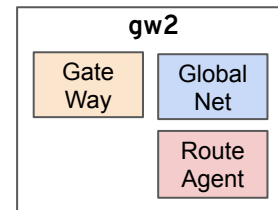
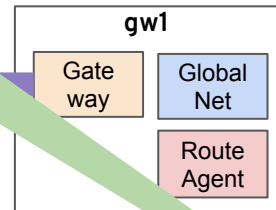
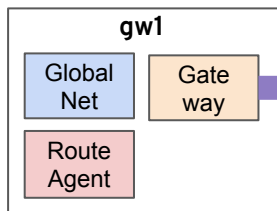
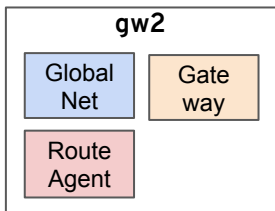
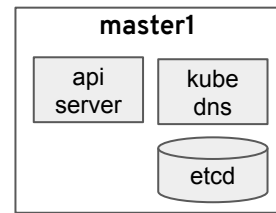
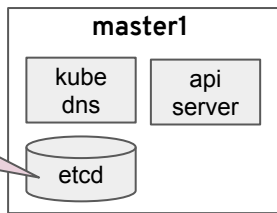
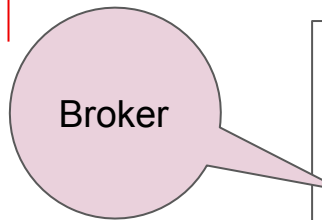


Service Discovery

- クラスタ内のサービスディスカバリ
 - KubeDNSを使って、Serviceを名前解決できるようにする
 - `<service>.<namespace>.svc.cluster.local`
- Submarinerによるサービスディスカバリの拡張
 - 他クラスタで「公開」設定した Serviceを、別クラスタからDNSで名前解決できるようにする仕組み
 - `<service>.<namespace>.svc.clusterset.local`
 - (古いバージョンのSubmarinerだと "supercluster.local")
- 登場人物
 - Lighthouse Agent
 - Lighthouse DNS Server

cluster1

cluster2



公開したいServiceに関して、あらかじめServiceExportを作成しておく

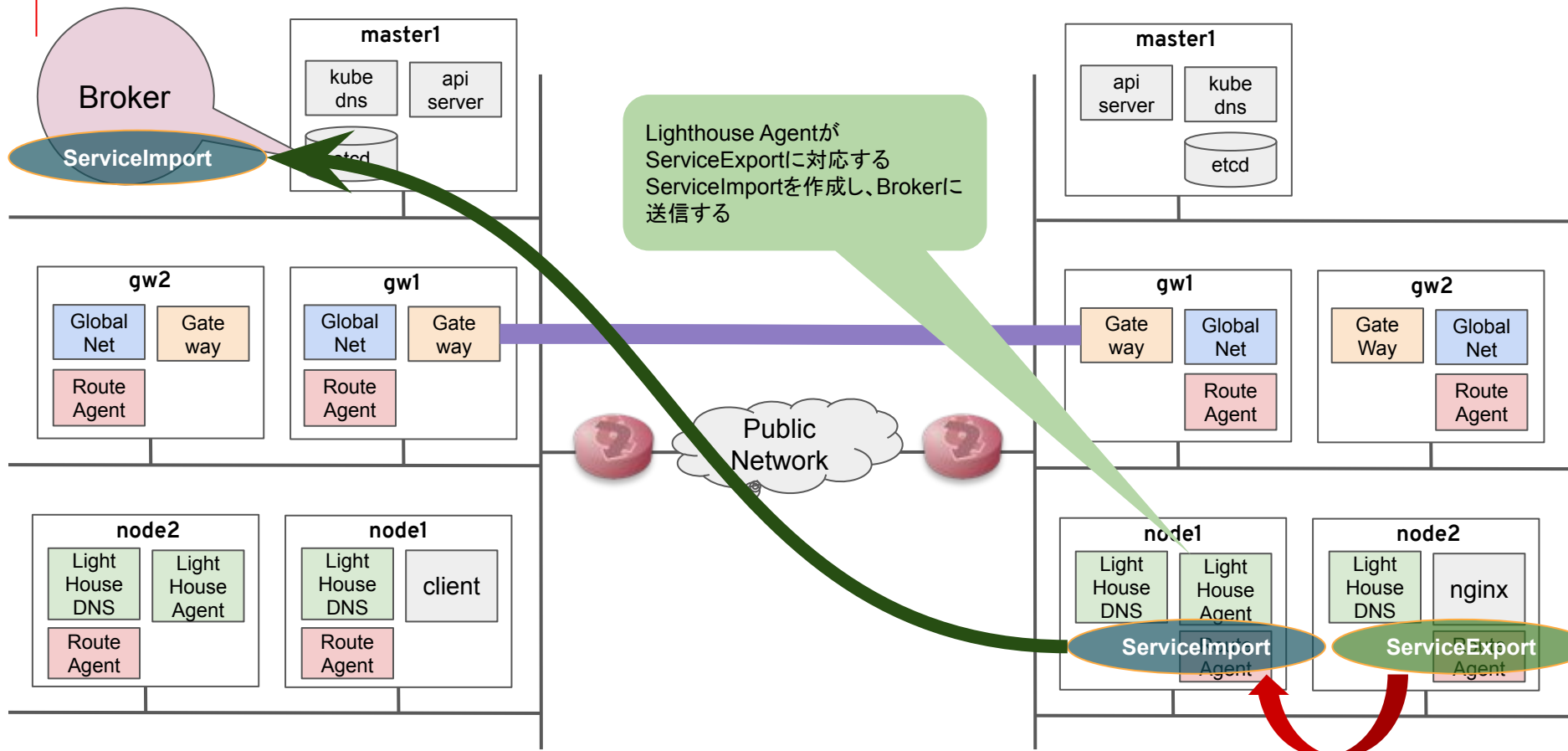
```
subctl export service ${service}
```



※ ServiceExport, ServiceImportオブジェクトは、実際はetcdに保存されます

cluster1

cluster2

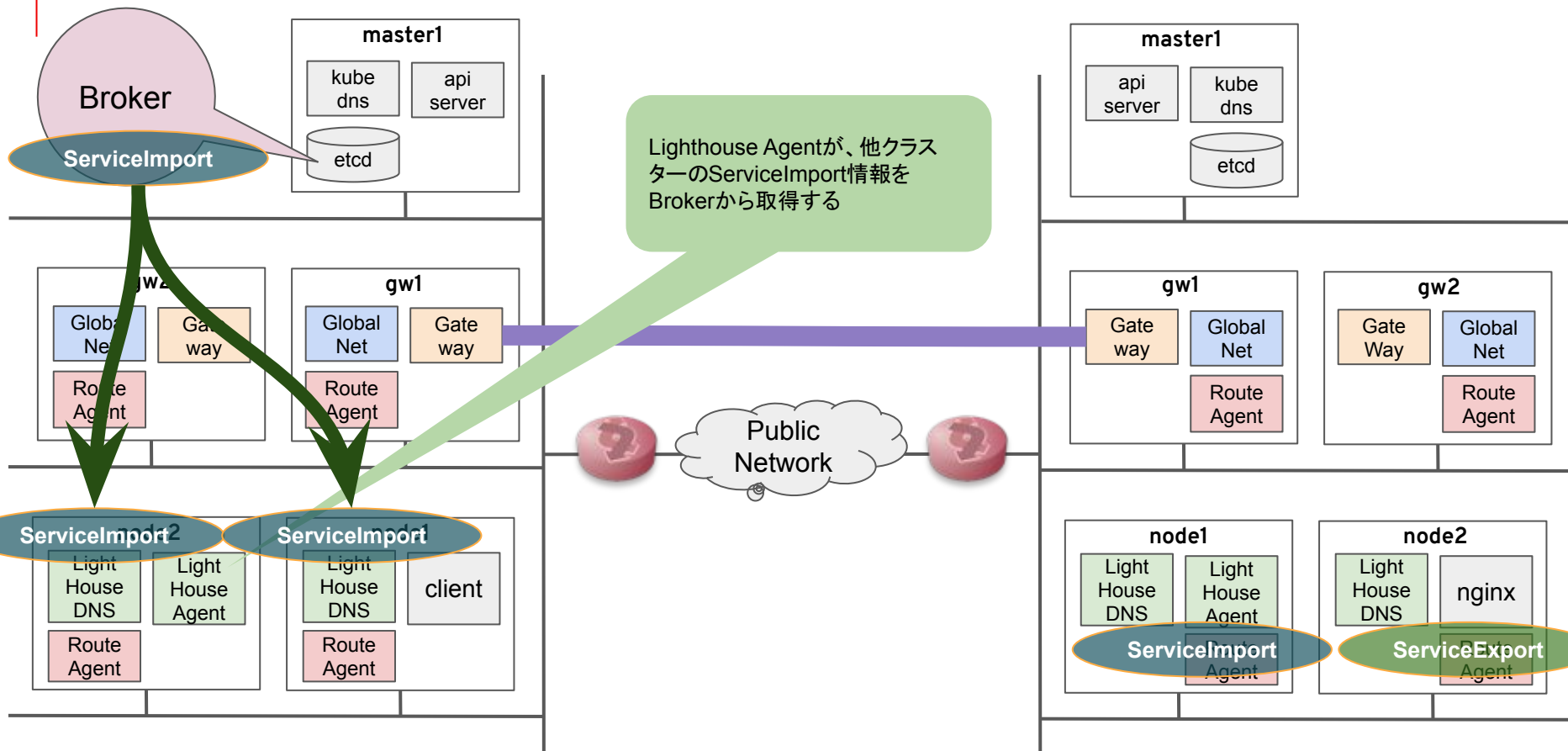


※ ServiceExport, ServiceImportオブジェクトは、実際はetcdに保存されます



cluster1

cluster2



※ ServiceExport, ServiceImportオブジェクトは、実際はetcdに保存されます



Service Discovery

- Lighthouse Agent
 - 各クラスターで1台稼動するPod
 - BrokerのKubernetes APIと通信し、クラスター越しにアクセスされる Serviceに関するメタデータ情報を交換する
 - 自クラスターの公開 Serviceのメタデータを Brokerに送る
 - ユーザーはサービスを公開するために、ServiceExportを作成する
 - LH AgentはServiceExportができると、対応するServiceImportを作成し、Brokerに送る
 - 他クラスターの公開 Serviceのメタデータを Brokerから取得する
 - Brokerに他クラスターの ServiceImportが追加されると、LH Agentはそれを手元にコピーする

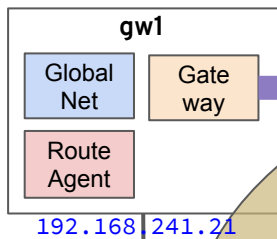
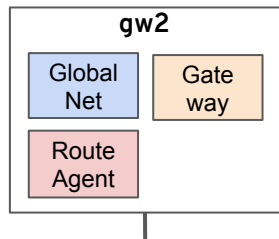
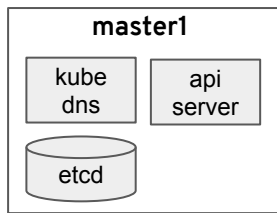
Service Discovery

- Lighthouse DNS Server
 - KubeDNSのupstream DNSサーバとして、各クラスターで2台稼働する
 - ServiceImportの情報をもとに、"clusterset.local"ドメインのレコードを構成する
- ワークフロー
 - ユーザーPodがclusterset.localドメインの名前を解決しようとする
 - KubeDNSはそのクエルをLighthouse DNSサーバにフォワードする
 - Lighthouse DNSサーバは、ServiceImportのキャッシュがあればそのレコードを返し、なければNXDOMAINを返す

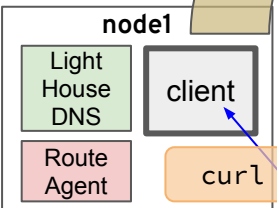
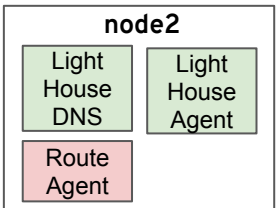
クラスターを またいだPod間通信

GlobalNet: 169.254.0.0/19

cluster1



192.168.241.21

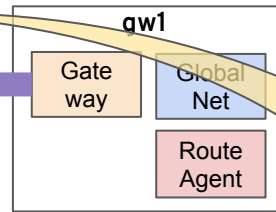
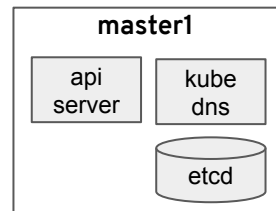


192.168.241.11

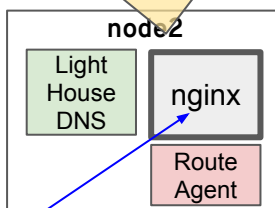
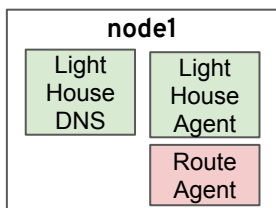
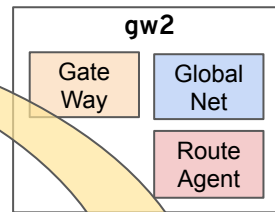
client Pod: 10.241.0.4 (169.254.18.25)
 (Pod IP address) (Global IP for Pod)

GlobalNet: 169.254.32.0/19

cluster2

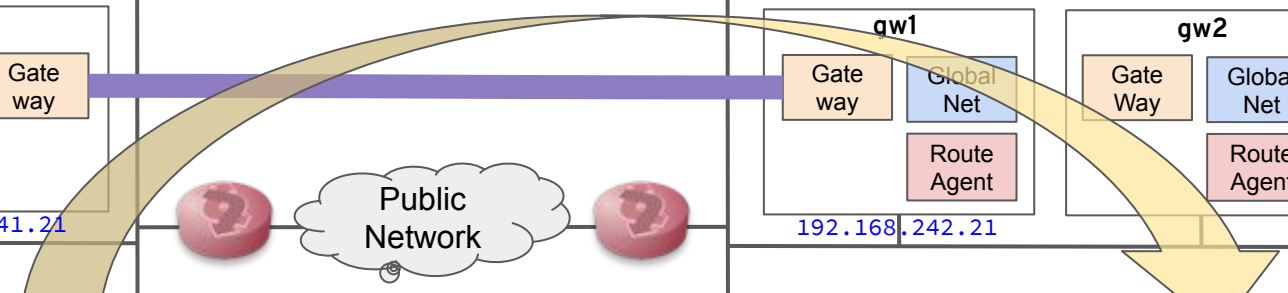


192.168.242.21



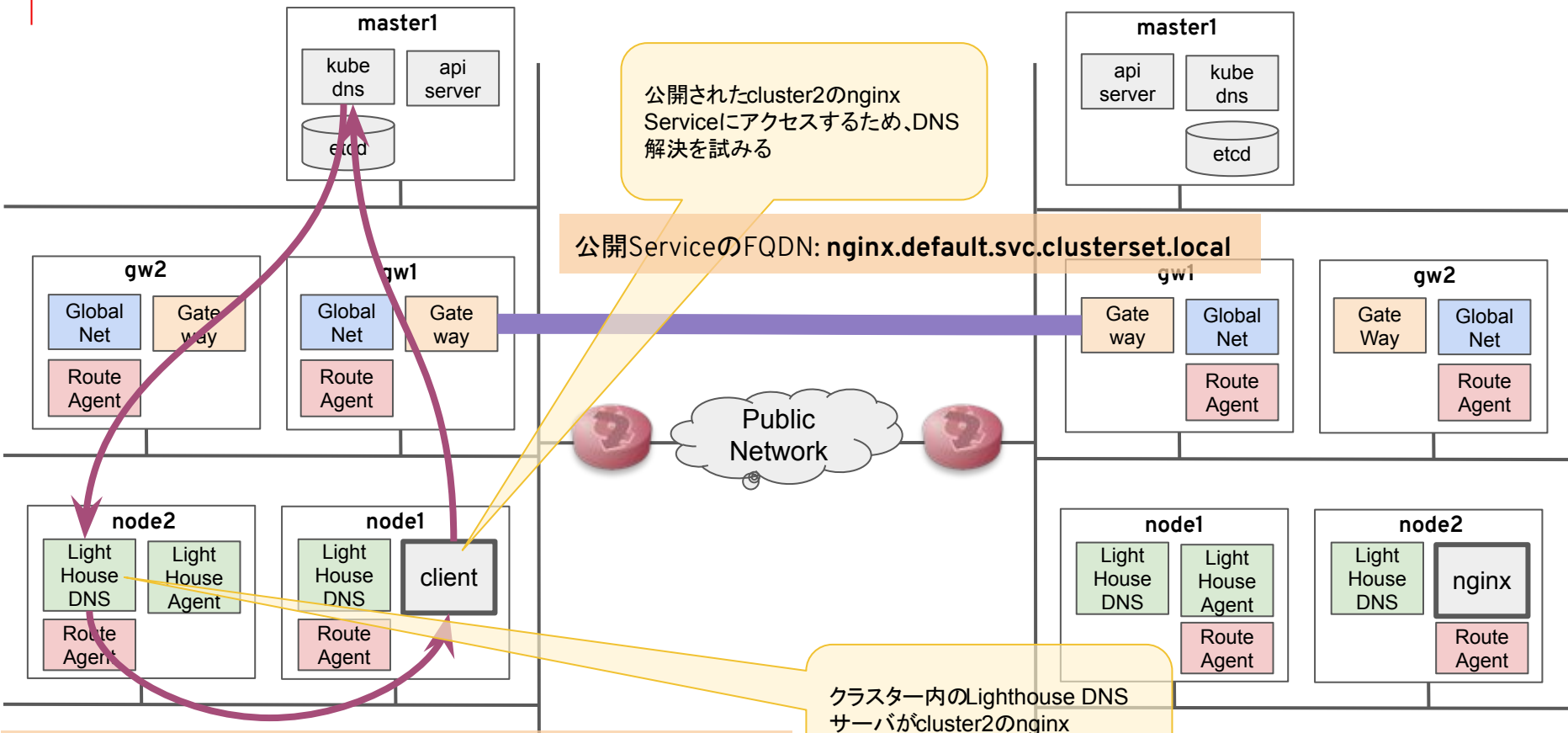
192.168.242.12

target pod: 10.242.213 (169.254.32.40)
 target svc: 10.142.73.136 (169.254.33.168)
 name: nginx, namespace: default



cluster1

cluster2



公開されたcluster2のnginx Serviceにアクセスするため、DNS 解決を試みる

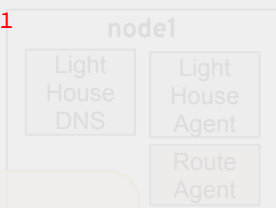
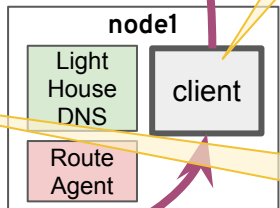
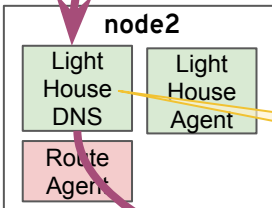
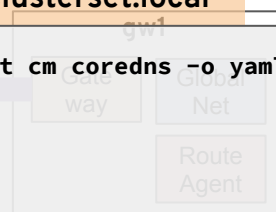
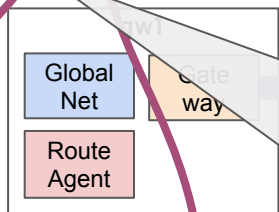
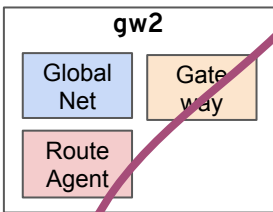
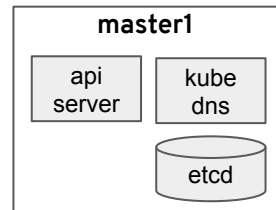
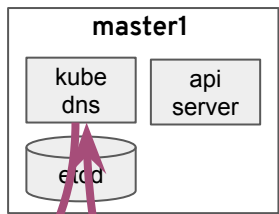
公開ServiceのFQDN: `nginx.default.svc.clusterset.local`

クラスター内のLighthouse DNS サーバがcluster2のnginx ServiceのGlobal IPを返す

`nginx.default.svc.clusterset.local`のIPアドレス: **169.254.33.168**

cluster1

cluster2



公開されたcluster2のnginx Serviceにアクセスするため、DNS 解決を試みる

公開ServiceのFQDN: nginx.default.svc.clusterset.local

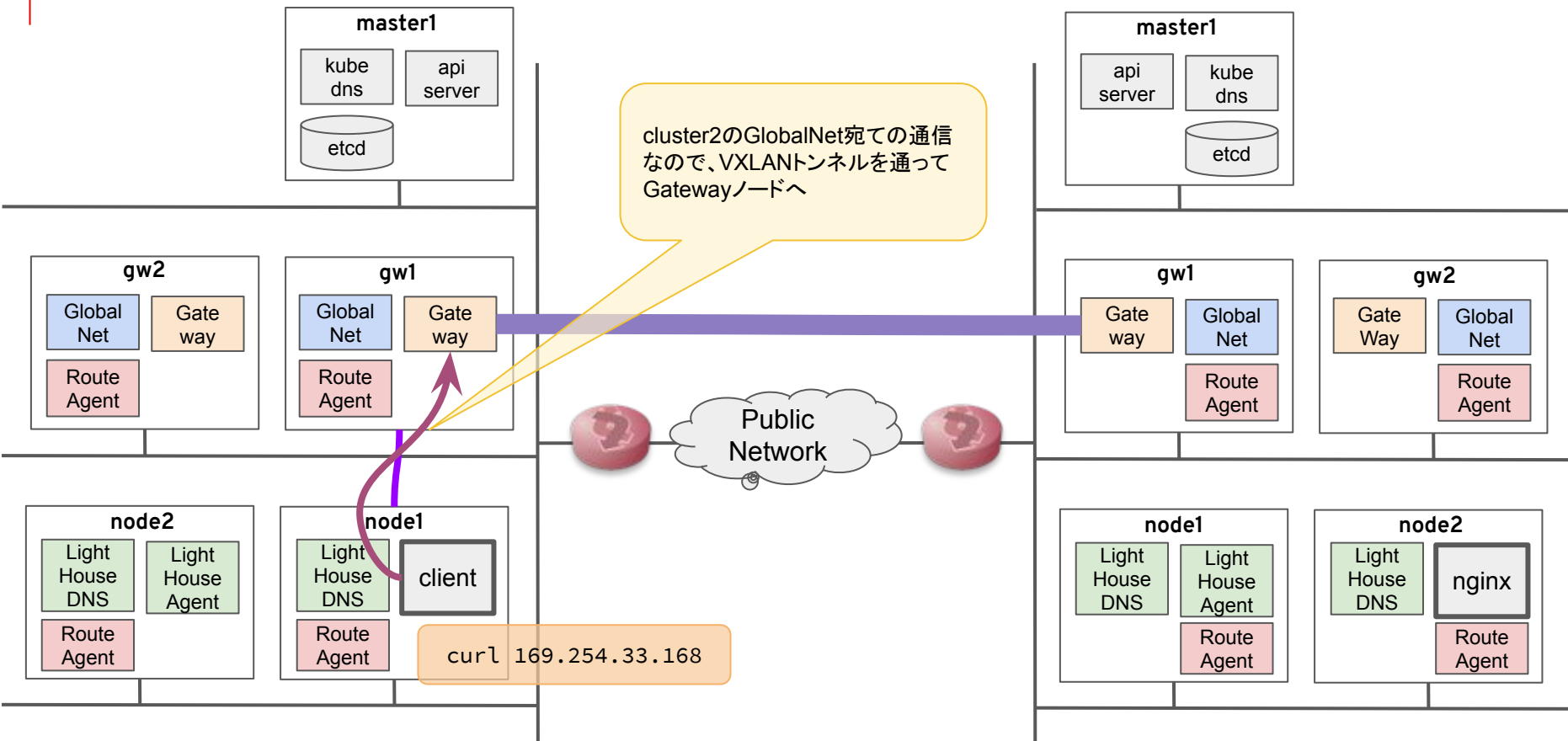
```
$ kubectl -n kube-system get cm coredns -o yaml | head -n 15
apiVersion: v1
data:
  Corefile: |
    #lighthouse
    clusterset.local:53 {
      forward . 10.141.162.221
    }
    supercluster.local:53 {
      forward . 10.141.162.221
    }
    .:53 {
      errors
      health {
        lameduck 5s
      }
    }

$ kubectl -n submariner-operator get svc
NAME                                TYPE          CLUSTER-IP      EXTERNAL-I
submariner-lighthouse-coredns      ClusterIP     10.141.162.221  <none>
submariner-operator-metrics        ClusterIP     10.141.85.172   <none>
```

nginx.default.svc.clusterset.localのIPアドレス: 169.254.33.16

cluster1

cluster2



cluster2のGlobalNet宛での通信
なので、VXLANトンネルを通じて
Gatewayノードへ

curl 169.254.33.168

```
$ ip -4 addr show dev vx-submariner
```

```
13: vx-submariner: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default  
inet 240.168.241.21/8 brd 240.255.255.255 scope global vx-submariner  
valid_lft forever preferred_lft forever
```

cluster2のGlobalNet宛での通信
なので、VXLANトンネルを通じて
Gatewayノードへ

```
$ ip route show
```

```
default via 192.168.241.1 dev eth0  
10.241.0.0/16 dev weave proto kernel scope link src 10.241.0.1  
169.254.0.0/16 dev eth0 scope link metric 1002  
169.254.32.0/19 via 240.168.241.21 dev vx-submariner proto static  
192.168.241.0/24 dev eth0 proto kernel scope link src 192.168.241.11  
240.0.0.0/8 dev vx-submariner proto kernel scope link src 240.168.241.1
```

```
$ ip -d link show dev vx-submariner
```

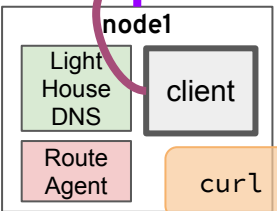
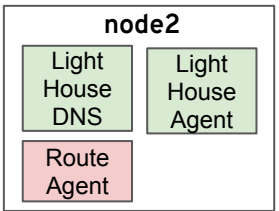
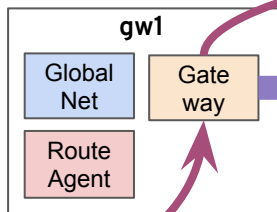
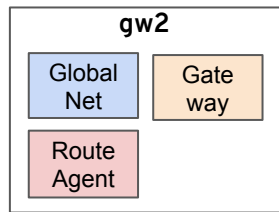
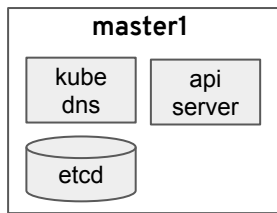
```
15: vx-submariner: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue  
DEFAULT group default  
link/ether 92:34:a2:38:15:ff brd ff:ff:ff:ff:ff:ff promiscuity 0  
vxlan id 100 remote 192.168.241.21 srcport 0 0 dstport 4800 nolearn
```

client Pod: 10.241.0.4 (169.254.18.25)

target pod: 10.242.2.3 (169.254.32.40)
target svc: 10.142.73.136 (169.254.33.168)

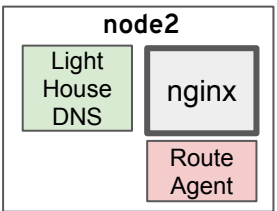
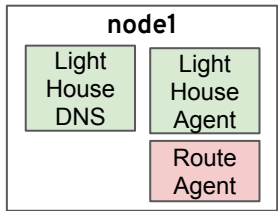
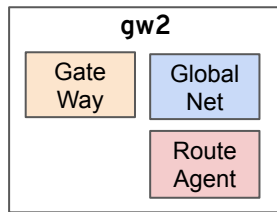
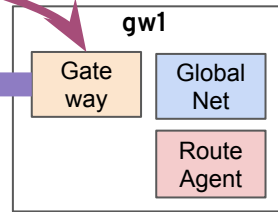
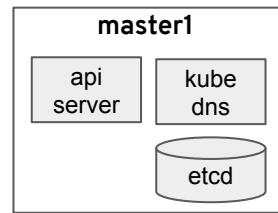


cluster1



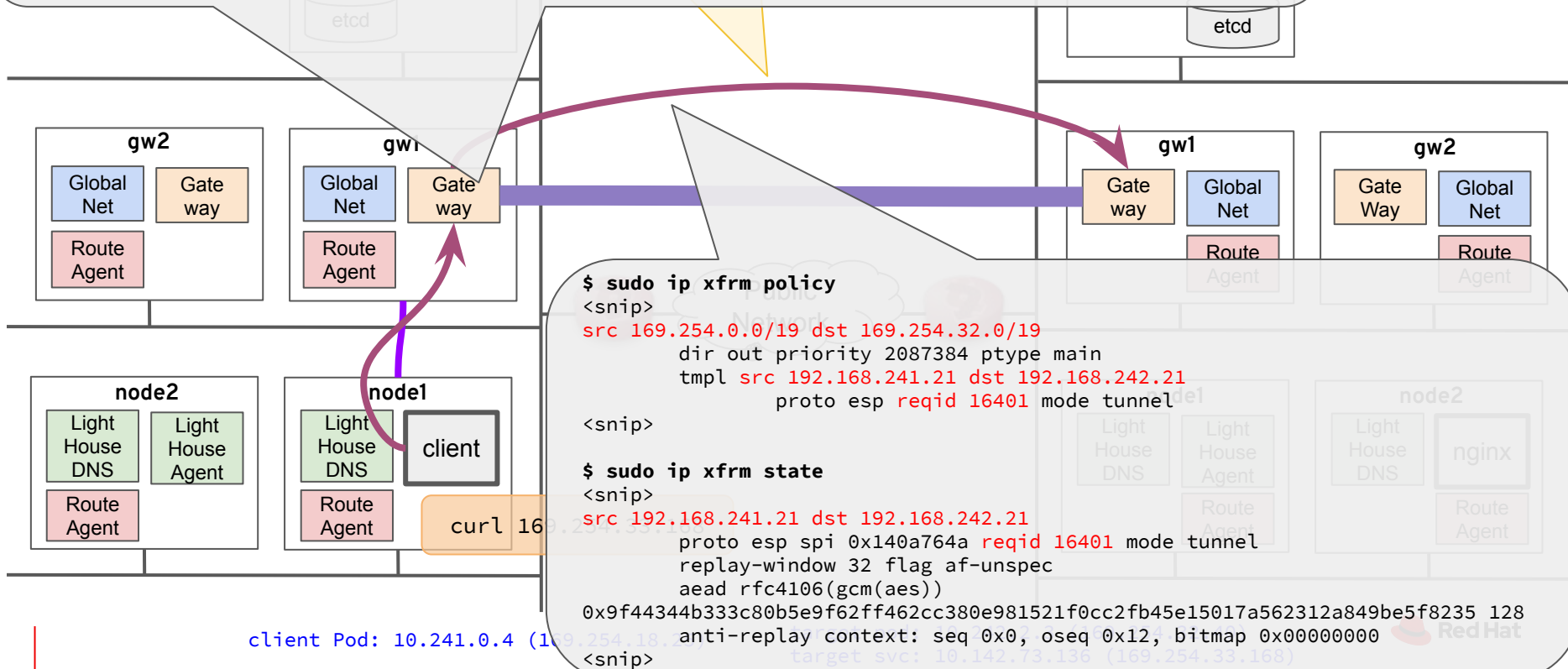
ソースアドレスをPodのアドレスに対応するGlobal IPに変換し、IPsecのXFRM Policyにしたがってcluster2のGatewayノードへ

cluster2



```
$ sudo iptables -S -t nat
```

```
<snip>  
-A POSTROUTING -j SUBMARINER-POSTROUTING  
-A SUBMARINER-POSTROUTING -j SUBMARINER-GN-EGRESS  
-A SUBMARINER-GN-EGRESS -j SUBMARINER-GN-MARK  
-A SUBMARINER-GN-MARK -d 169.254.32.0/19 -j MARK --set-xmark 0xc0000/0xc0000  
-A SUBMARINER-GN-EGRESS -s 10.241.0.4/32 -m mark --mark 0xc0000/0xc0000 -j SNAT --to-source 169.254.18.25  
<snip>
```



```
$ sudo ip xfrm policy
```

```
<snip>  
src 169.254.0.0/19 dst 169.254.32.0/19  
dir out priority 2087384 ptype main  
tmpl src 192.168.241.21 dst 192.168.242.21  
proto esp reqid 16401 mode tunnel  
<snip>
```

```
$ sudo ip xfrm state
```

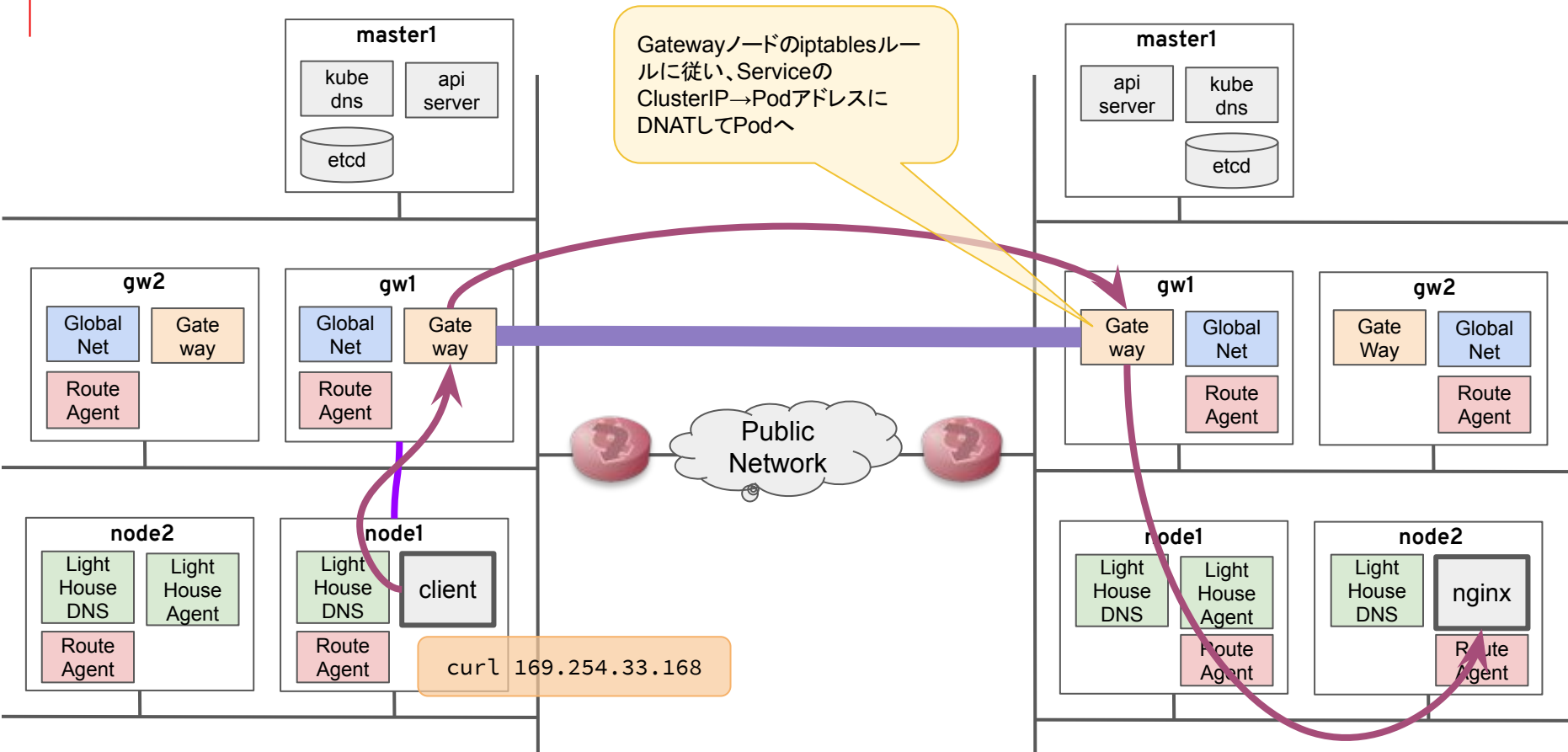
```
<snip>  
src 192.168.241.21 dst 192.168.242.21  
proto esp spi 0x140a764a reqid 16401 mode tunnel  
replay-window 32 flag af-unspec  
aad rfc4106(gcm(aes))  
0x9f44344b333c80b5e9f62ff462cc380e981521f0cc2fb45e15017a562312a849be5f8235 128  
anti-replay context: seq 0x0, oseq 0x12, bitmap 0x00000000  
<snip>
```

client Pod: 10.241.0.4 (169.254.18.25)



cluster1

cluster2



curl 169.254.33.168

client Pod: 10.241.0.4 (169.254.18.25)

target pod: 10.242.2.3 (169.254.32.40)
target svc: 10.142.73.136 (169.254.33.168)



```

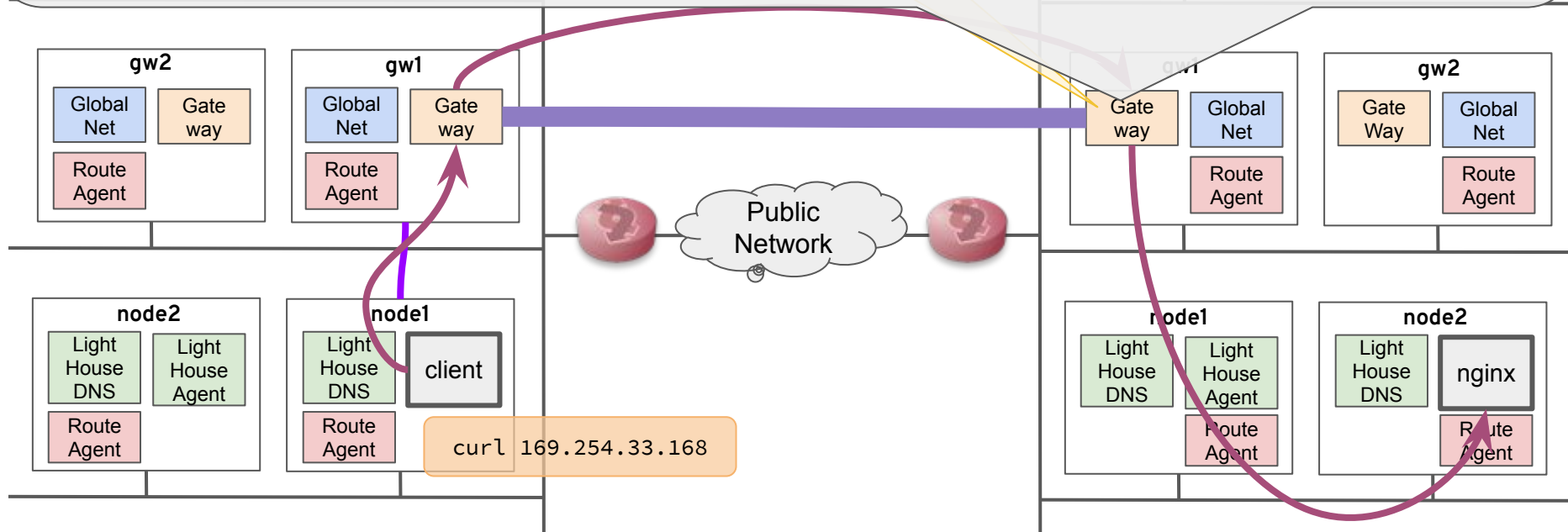
$ sudo iptables -S -t nat: 169.254.0.0/19
-A PREROUTING -j SUBMARINER-GN-INGRESS
-A PREROUTING -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
<snip>
-A KUBE-SERVICES -d 10.142.73.136/32 -p tcp -m comment --comment "default/nginx:http cluster IP" -m tcp --dport 80 -j KUBE-SVC-W74WBVT47KWK7FLX
<snip>
-A KUBE-SEP-4I5XE2BCCVWHDHTF -p tcp -m comment --comment "default/nginx:http" -m tcp -j DNAT --to-destination 10.242.4.4:8080
-A KUBE-SEP-A4FGUEZKMLCAJVUL -p tcp -m comment --comment "default/nginx:http" -m tcp -j DNAT --to-destination 10.242.2.3:8080
-A KUBE-SEP-FB5TTVHWC6TEALPV -p tcp -m comment --comment "default/nginx:http" -m tcp -j DNAT --to-destination 10.242.1.4:8080
<snip>
-A KUBE-SVC-W74WBVT47KWK7FLX -m comment --comment "default/nginx:http" -m statistic --mode random --probability 0.33333333349 -j KUBE-SEP-FB5TTVHWC6TEALPV
-A KUBE-SVC-W74WBVT47KWK7FLX -m comment --comment "default/nginx:http" -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-A4FGUEZKMLCAJVUL
-A KUBE-SVC-W74WBVT47KWK7FLX -m comment --comment "default/nginx:http" -j KUBE-SEP-4I5XE2BCCVWHDHTF
<snip>
-A SUBMARINER-GN-INGRESS -d 169.254.33.168/32 -j KUBE-SVC-W74WBVT47KWK7FLX

```

GlobalNet: 169.254.32.0/19
cluster2

kube-proxyによるルール

submarinerによるルール



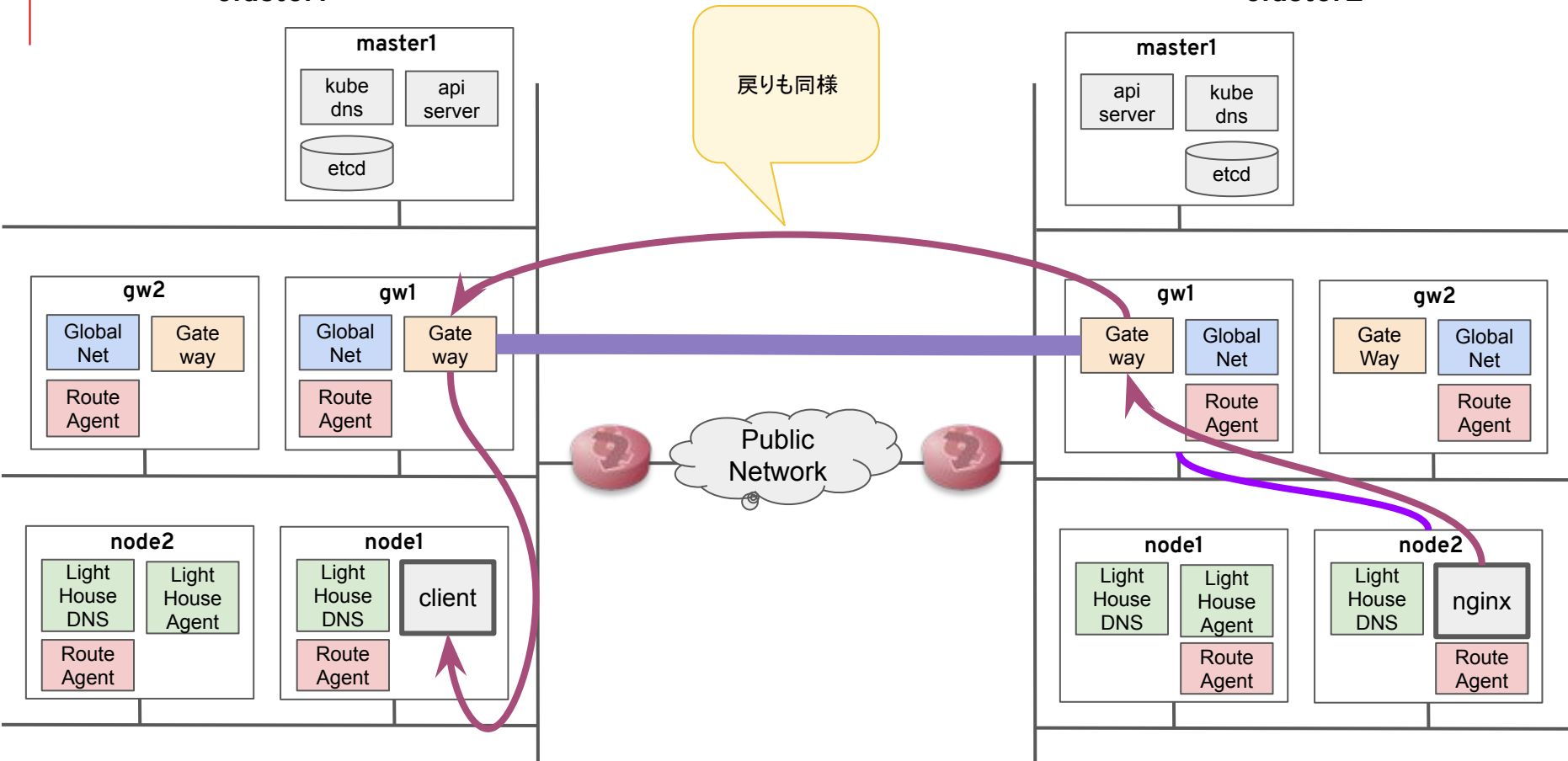
client Pod: 10.241.0.4 (169.254.18.25)

target pod: 10.242.2.3 (169.254.32.40)
target svc: 10.142.73.136 (169.254.33.168)



cluster1

cluster2



client Pod: 10.241.0.4 (169.254.18.25)

target pod: 10.242.2.3 (169.254.32.40)
target svc: 10.142.73.136 (169.254.33.168)

インストール

セットアップ

- インストール方法
 - subctl
 - helm
- subctlを使ったインストール
 - subctlをダウンロード
 - `curl -Ls https://get.submariner.io | bash`
 - Brokerの構築
 - `subctl deploy-broker --kubeconfig <KUBECONFIG>`
 - broker-info.submが生成されるので、Submariner接続する各クラスターにばらまく
 - Submarinerのデプロイ
 - 各参加クラスターで実行
 - `subctl join --kubeconfig <KUBECONFIG> broker-info.subm --clusterid <ID>`
 - このコマンドでsubmariner-operatorがデプロイされる
 - あとはsubmariner-operatorが全部やってくれる

デモ

環境

- Kubernetesクラスター#1
 - CentOS7
 - Kubernetes v1.19.2
 - Submariner v0.6.1
 - Weave
- Kubernetesクラスター#2
 - CentOS7
 - Kubernetes v1.19.2
 - Submariner v0.6.1
 - Flannel

環境 (クラスター#1)

```
[centos@site1-master1 setup]$ subctl show all
```

```
Showing information for cluster "kubernetes-admin@kubernetes":
```

```
  Discovered network details:
```

```
    Network plugin: weave-net
    Service CIDRs:  [10.141.0.0/16]
    Cluster CIDRs:  [10.241.0.0/16]
    Global CIDR:    169.254.0.0/19
```

CLUSTER ID	ENDPOINT IP	PUBLIC IP	CABLE DRIVER	TYPE
site1	192.168.241.21		strongswan	local
site2	192.168.242.21		strongswan	remote
site1	192.168.241.22		strongswan	local

GATEWAY	CLUSTER	REMOTE IP	CABLE DRIVER	SUBNETS	STATUS
site2-gw1	site2	192.168.242.21	strongswan	169.254.32.0/19	connected

NODE	HA STATUS	SUMMARY
site1-gw1	active	All connections (1) are established
site1-gw2	passive	There are no connections

COMPONENT	REPOSITORY	VERSION
submariner	quay.io/submariner	0.6.1
submariner-operator	quay.io/submariner	0.6.1
service-discovery	quay.io/submariner	0.6.1

環境 (クラスター#1)

NAME	STATUS	ROLES	AGE	VERSION
site1-gw1	Ready	<none>	10h	v1.19.2
site1-gw2	Ready	<none>	10h	v1.19.2
site1-master1	Ready	master	10h	v1.19.2
site1-node1	Ready	<none>	10h	v1.19.2
site1-node2	Ready	<none>	10h	v1.19.2

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE
READINESS GATES									
default	centos-69ccb76b8-4d9nd	1/1	Running	0	9h	10.241.240.2	site1-node1	<none>	<none>
default	centos-69ccb76b8-hw2l6	1/1	Running	0	9h	10.241.96.3	site1-gw2	<none>	<none>
default	centos-69ccb76b8-s4x9l	1/1	Running	0	9h	10.241.192.3	site1-node2	<none>	<none>
kube-system	coredns-f9fd979d6-8p7j8	1/1	Running	0	10h	10.241.192.1	site1-node2	<none>	<none>
kube-system	coredns-f9fd979d6-rlz25	1/1	Running	0	10h	10.241.96.1	site1-gw2	<none>	<none>
kube-system	etcd-site1-master1	1/1	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	kube-apiserver-site1-master1	1/1	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	kube-controller-manager-site1-master1	1/1	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	kube-proxy-29bz9	1/1	Running	0	10h	192.168.241.11	site1-node1	<none>	<none>
kube-system	kube-proxy-4lwhf	1/1	Running	0	10h	192.168.241.21	site1-gw1	<none>	<none>
kube-system	kube-proxy-bcp8d	1/1	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	kube-proxy-psjst	1/1	Running	0	10h	192.168.241.12	site1-node2	<none>	<none>
kube-system	kube-proxy-rh6l2	1/1	Running	0	10h	192.168.241.22	site1-gw2	<none>	<none>
kube-system	kube-scheduler-site1-master1	1/1	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	weave-net-2wjc6	2/2	Running	0	10h	192.168.241.10	site1-master1	<none>	<none>
kube-system	weave-net-mvn9n	2/2	Running	0	10h	192.168.241.21	site1-gw1	<none>	<none>
kube-system	weave-net-wlcvp	2/2	Running	0	10h	192.168.241.12	site1-node2	<none>	<none>
kube-system	weave-net-wpg4w	2/2	Running	0	10h	192.168.241.22	site1-gw2	<none>	<none>
kube-system	weave-net-x2tnf	2/2	Running	0	10h	192.168.241.11	site1-node1	<none>	<none>
submariner-operator	submariner-gateway-rcm82	1/1	Running	0	9h	192.168.241.22	site1-gw2	<none>	<none>
submariner-operator	submariner-gateway-wv6ks	1/1	Running	0	10h	192.168.241.21	site1-gw1	<none>	<none>
submariner-operator	submariner-globalnet-dht4f	1/1	Running	0	9h	192.168.241.22	site1-gw2	<none>	<none>
submariner-operator	submariner-globalnet-khw5m	1/1	Running	0	10h	192.168.241.21	site1-gw1	<none>	<none>
submariner-operator	submariner-lighthouse-agent-6449c8786f-mztcs	1/1	Running	0	10h	10.241.240.1	site1-node1	<none>	<none>
submariner-operator	submariner-lighthouse-coredns-869db5bfb8-ghg6r	1/1	Running	0	10h	10.241.192.2	site1-node2	<none>	<none>
submariner-operator	submariner-lighthouse-coredns-869db5bfb8-nkfm2	1/1	Running	0	10h	10.241.96.2	site1-gw2	<none>	<none>
submariner-operator	submariner-operator-5954fd57b9-zqx9p	1/1	Running	0	10h	10.241.0.2	site1-gw1	<none>	<none>
submariner-operator	submariner-routeagent-fn46d	1/1	Running	0	10h	192.168.241.11	site1-node1	<none>	<none>
submariner-operator	submariner-routeagent-nfpt6	1/1	Running	0	10h	192.168.241.22	site1-gw2	<none>	<none>
submariner-operator	submariner-routeagent-pkk6m	1/1	Running	0	10h	192.168.241.21	site1-gw1	<none>	<none>
submariner-operator	submariner-routeagent-sffds	1/1	Running	0	10h	192.168.241.12	site1-node2	<none>	<none>

環境 (クラスター#2)

```
[centos@site2-master1 setup]$ subctl show all
```

```
Showing information for cluster "kubernetes-admin@kubernetes":
```

```
  Discovered network details:
```

```
    Network plugin: generic
    Service CIDRs:  [10.142.0.0/16]
    Cluster CIDRs:  [10.242.0.0/16]
```

CLUSTER ID	ENDPOINT IP	PUBLIC IP	CABLE DRIVER	TYPE
site2	192.168.242.21		strongswan	local
site1	192.168.241.21		strongswan	remote
site2	192.168.242.22		strongswan	local

GATEWAY	CLUSTER	REMOTE IP	CABLE DRIVER	SUBNETS	STATUS
site1-gw1	site1	192.168.241.21	strongswan	169.254.0.0/19	connected

NODE	HA STATUS	SUMMARY
site2-gw1	active	All connections (1) are established
site2-gw2	passive	There are no connections

COMPONENT	REPOSITORY	VERSION
submariner	quay.io/submariner	0.6.1
submariner-operator	quay.io/submariner	0.6.1
service-discovery	quay.io/submariner	0.6.1

環境 (クラスター#2)

NAME	STATUS	ROLES	AGE	VERSION
site2-gw1	Ready	<none>	10h	v1.19.2
site2-gw2	Ready	<none>	10h	v1.19.2
site2-master1	Ready	master	10h	v1.19.2
site2-node1	Ready	<none>	10h	v1.19.2
site2-node2	Ready	<none>	10h	v1.19.2

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE
READINESS GATES									
default	hello-site2-5c48dfcd96-6mk2z	1/1	Running	0	9h	10.242.2.4	site2-node1	<none>	<none>
default	hello-site2-5c48dfcd96-cq8pl	1/1	Running	0	9h	10.242.3.4	site2-node2	<none>	<none>
default	hello-site2-5c48dfcd96-f5zbg	1/1	Running	0	9h	10.242.4.4	site2-gw2	<none>	<none>
kube-system	coredns-f9fd979d6-68z88	1/1	Running	0	10h	10.242.3.2	site2-node2	<none>	<none>
kube-system	coredns-f9fd979d6-rrwcx	1/1	Running	0	10h	10.242.4.2	site2-gw2	<none>	<none>
kube-system	etcd-site2-master1	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
kube-system	kube-apiserver-site2-master1	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
kube-system	kube-controller-manager-site2-master1	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
kube-system	kube-flannel-ds-62ljs	1/1	Running	0	10h	192.168.242.22	site2-gw2	<none>	<none>
kube-system	kube-flannel-ds-9gxp6	1/1	Running	0	10h	192.168.242.12	site2-node2	<none>	<none>
kube-system	kube-flannel-ds-f28v5	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
kube-system	kube-flannel-ds-q72qc	1/1	Running	0	10h	192.168.242.21	site2-gw1	<none>	<none>
kube-system	kube-flannel-ds-rwstq	1/1	Running	0	10h	192.168.242.11	site2-node1	<none>	<none>
kube-system	kube-proxy-4vdzv	1/1	Running	0	10h	192.168.242.11	site2-node1	<none>	<none>
kube-system	kube-proxy-g5cj2	1/1	Running	0	10h	192.168.242.22	site2-gw2	<none>	<none>
kube-system	kube-proxy-kg2jr	1/1	Running	0	10h	192.168.242.12	site2-node2	<none>	<none>
kube-system	kube-proxy-l74q6	1/1	Running	0	10h	192.168.242.21	site2-gw1	<none>	<none>
kube-system	kube-proxy-sfszz	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
kube-system	kube-scheduler-site2-master1	1/1	Running	0	10h	192.168.242.10	site2-master1	<none>	<none>
submariner-operator	submariner-gateway-92nr4	1/1	Running	0	9h	192.168.242.21	site2-gw1	<none>	<none>
submariner-operator	submariner-gateway-dtq42	1/1	Running	0	9h	192.168.242.22	site2-gw2	<none>	<none>
submariner-operator	submariner-globalnet-4jrkf	1/1	Running	0	9h	192.168.242.22	site2-gw2	<none>	<none>
submariner-operator	submariner-globalnet-8tsbg	1/1	Running	0	9h	192.168.242.21	site2-gw1	<none>	<none>
submariner-operator	submariner-lighthouse-agent-74cddbdf59-wlkb9	1/1	Running	0	9h	10.242.4.3	site2-gw2	<none>	<none>
submariner-operator	submariner-lighthouse-coredns-869db5bfb8-4jt8g	1/1	Running	0	9h	10.242.2.3	site2-node1	<none>	<none>
submariner-operator	submariner-lighthouse-coredns-869db5bfb8-ln9jm	1/1	Running	0	9h	10.242.3.3	site2-node2	<none>	<none>
submariner-operator	submariner-operator-5954fd57b9-t54zz	1/1	Running	0	9h	10.242.2.2	site2-node1	<none>	<none>
submariner-operator	submariner-routeagent-ghpfl	1/1	Running	0	9h	192.168.242.11	site2-node1	<none>	<none>
submariner-operator	submariner-routeagent-jxx5n	1/1	Running	0	9h	192.168.242.21	site2-gw1	<none>	<none>
submariner-operator	submariner-routeagent-sxhsw	1/1	Running	0	9h	192.168.242.22	site2-gw2	<none>	<none>
submariner-operator	submariner-routeagent-z5576	1/1	Running	0	9h	192.168.242.12	site2-node2	<none>	<none>




Submarinerの今後


ロードマップ

- Support more CNI Plugins
 - ovn-kubernetes, cilium, xKE/xKS, etc...
- More flexible IPsec topology
- More tunneling options
 - OpenVPN, VXLAN, IP in IP, ..
- Network Policy support accross clusters
 - coastguard <https://github.com/submariner-io/coastguard>
- ...

Thank you



 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

Red Hat is here to help

Responding to COVID-19 requires collaboration, transparency, and the free exchange of expertise.

[Ways to contact us](#)

