プラットフォームにおける組織間対話の重要性
- 実例から見るNWにおけるソフトウェア運用 -

JANOG58 2026.07.15

cybozu

プラットフォーム運用における、組織間の対話や交流の大切さを議論したい

- J57 にて、ネットワークエンジニアとプラットフォームエンジニア（k8s 等の基盤エンジニア）との対話に関する発表がありました。
  - 今こそ学びたいKubernetesネットワーク ～CNIが繋ぐNWとプラットフォームの「フラッと」な対話
  - https://www.janog.gr.jp/meeting/janog57/k8s/
- CNI という技術カットで、組織間の技術交流の重要性について提起した発表だと思っています。
- 弊社は Neco というプラットフォームを運用しており、組織間対話の重要性を多方面で感じています。
- 一つの企業からのアンサーソングとして、今回は「プラットフォーム運用」というカットで、組織間対話についての弊社の経験や考え、悩みをお伝えしたいと思っています。
- AI 時代、運用が大きく変わっていく中で、皆様から幅広い議論をお願いしたいと思っています。

cybozu.com と Neco プラットフォーム

- 弊社の提供するサービス（kintone等）は、多くが cybozu.com というドメインで運用されています。
- 転じて、社内ではこの枠で運用されているシステムを系として cybozu.com と呼ぶことがあります。
- cybozu.com は一部のサブシステムを除いて、大半がオンプレミス機材の上で運用されています。
- 更に、システムを効率的に運用するために、サービス提供用のプラットフォームを作っています。

今回登場するチーム

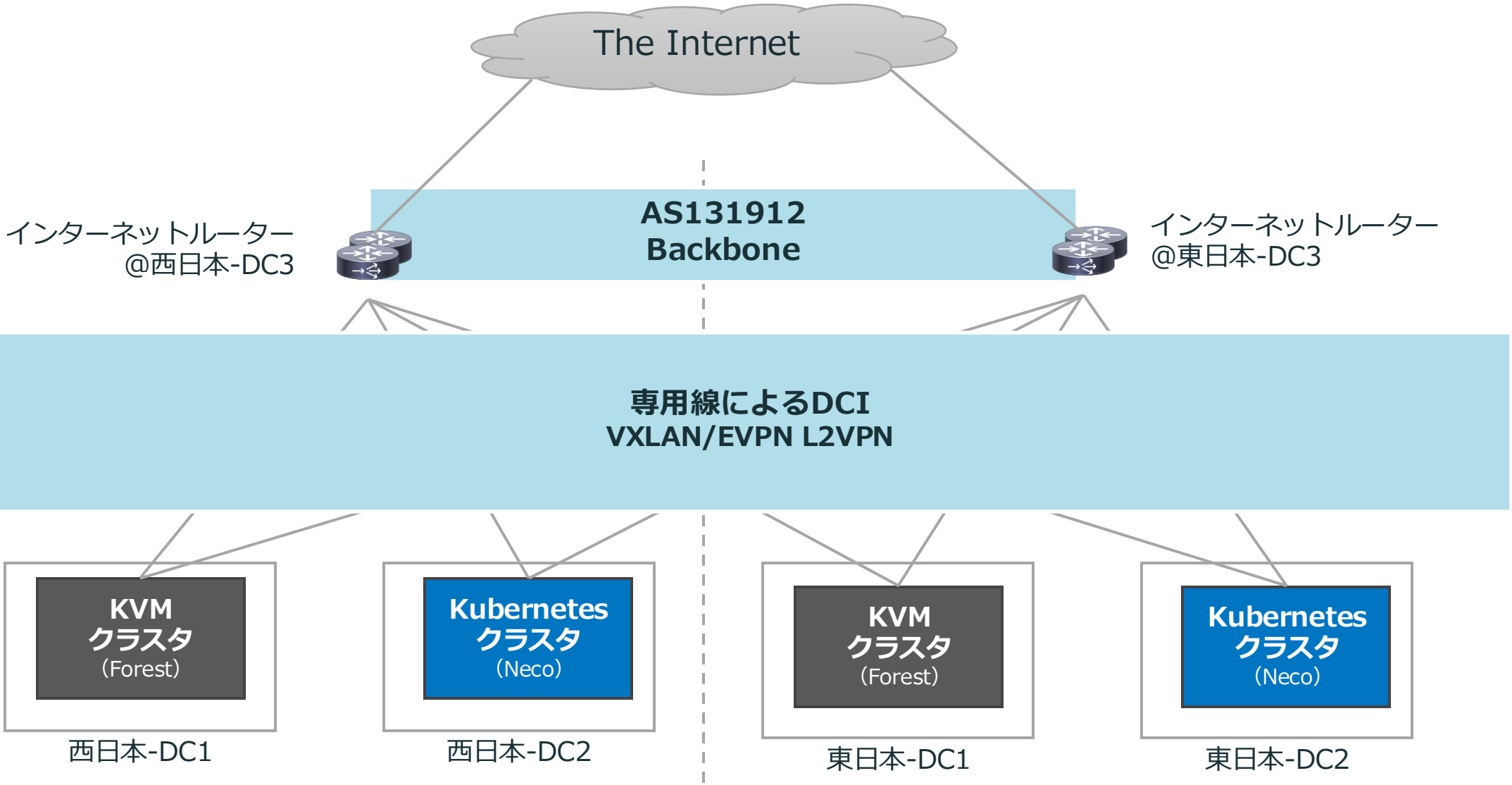
- NW＆DC チーム（尾崎の所属しているチーム）
  - cybozu.com だけではなく、社内の全ネットワークを担当するチーム
  - エンタープライズ NW 製品を組み合わせた、伝統的な NW 運用をしている
  - KVM をベースとした Forest 基盤の NW 運用は、こちらのチームで行なっていた
- Kubernetes ネットワーク チーム（寺嶋の所属しているチーム）
  - Neco の導入と共に発展、基本的に Neco に専属のチームとなっている
  - Linux をベースとしたソフトウェアによるネットワーク構築をしている
  - Kubernetes のサーバー側を担当しているチームとも密に行動している

サイボウズ の ネットワーク

サイボウズのネットワーク概要

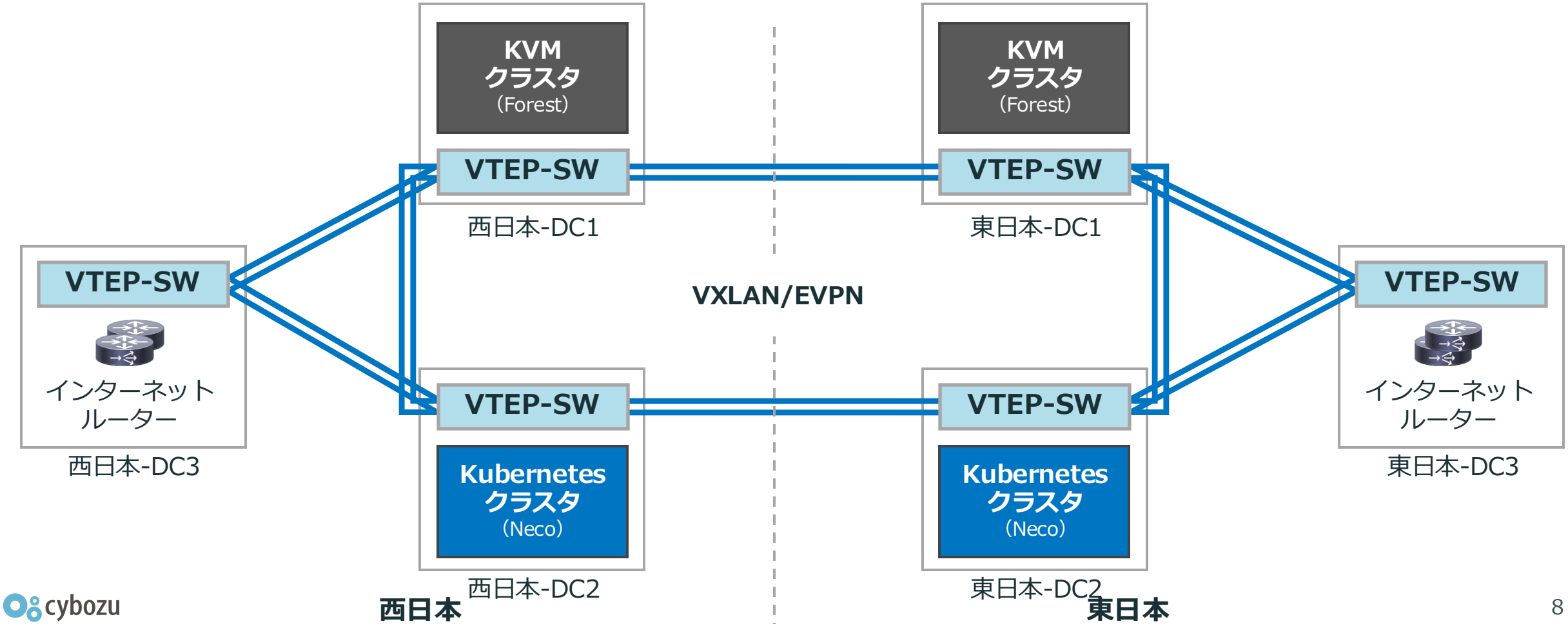
- **AS131912**
  - IX等の各種メンテナンス対応、ピアリング交渉、ROAや番号資源などの管理
- **オフィスネットワーク**
  - 国内外の全拠点
- **cybozu.com が稼働するネットワーク**
  - Forest（KVM ベース）向けのネットワーク
    - サーバーは MC-LAG で ToR スイッチに接続
    - コアスイッチで定義した VLAN 単位のネットワークにサーバーが接続される
  - Neco（Kubernetes ベース）向けのネットワーク
    - 各サーバーが BGP を話す
    - ソフトウェアで定義されたネットワークが BGP でサーバーから広報される
- **これらを繋ぐ DCI（Data Center Interconnect）ネットワーク**
- **上記 ネットワークはすべて NW&DC チームが管理・運用している**

cybozu.com のネットワークアーキテクチャ



データセンター間ネットワーク

- 東日本と西日本のDCをリング状に専用線で接続
- VXLAN/EVPN でDC間に仮想のP2P L2ネットワークを構築
- DC間のトラフィックはネットワークレイヤーで暗号化



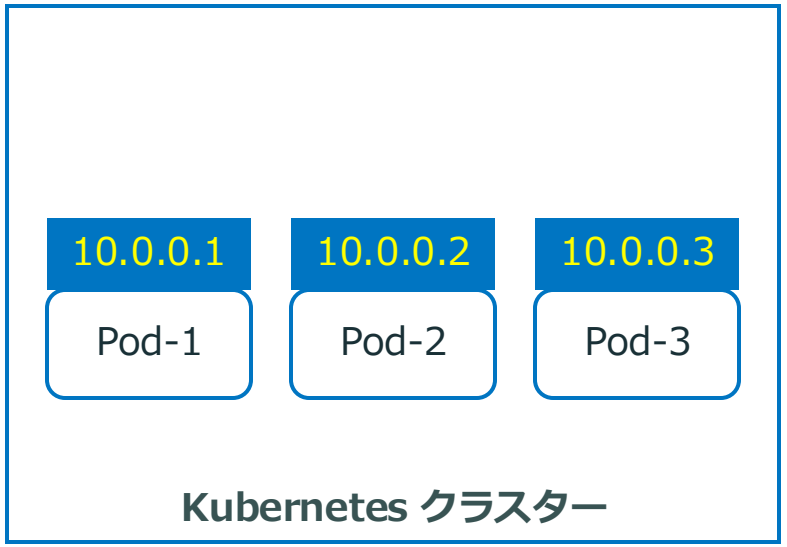
Neco の ネットワーク

Kubernetes における IP の整理

Service: 複数の Pod への通信を1つのエンドポイントに束ねる Kubernetes の L4 ロードバランサー

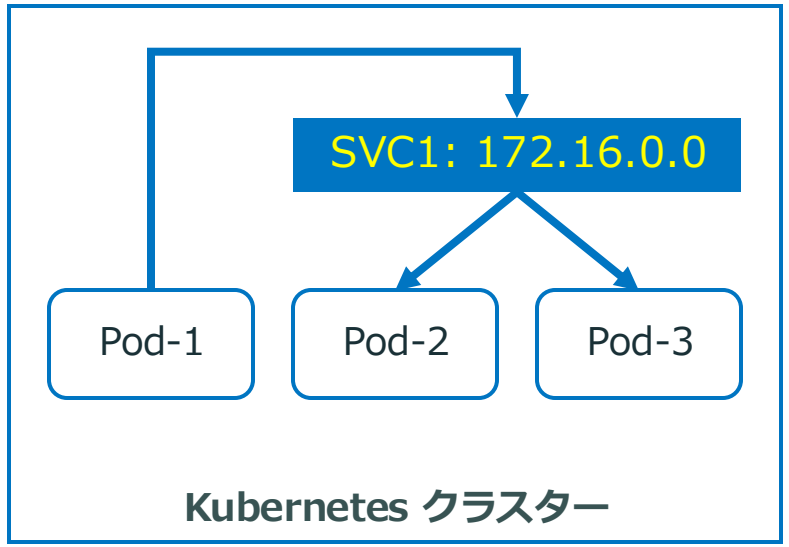
Pod IP

- Pod に割り当てられる IP



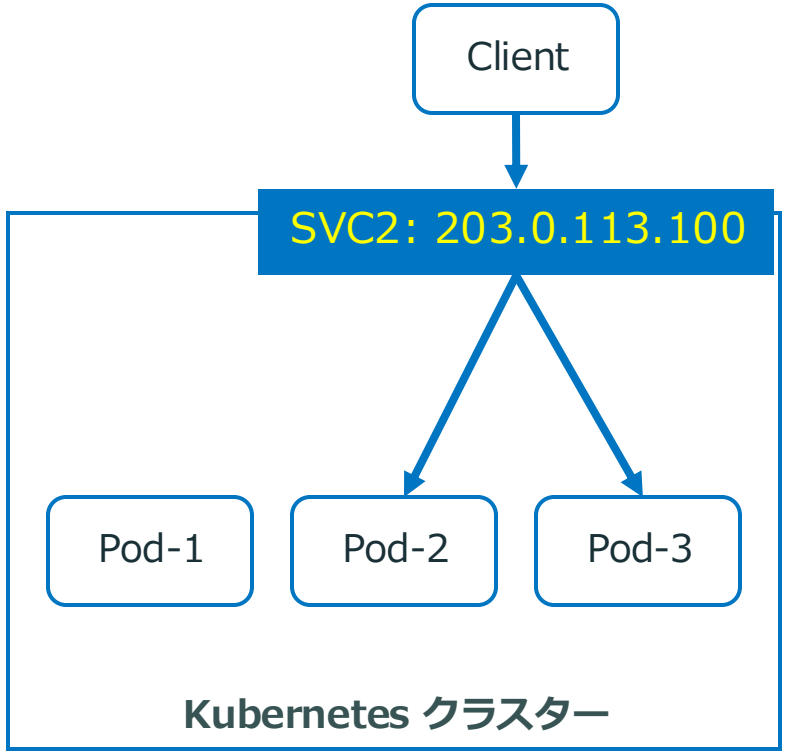
ClusterIP

- クラスターの内部から Service にアクセスするための IP
- この IP を持つ実体は存在しない (送信時に Pod IP へ DNAT)



LoadBalancer IP

- クラスターの外部から Service にアクセスするための IP



Kubernetes ネットワークと CNI プラグイン

- Kubernetes にはデフォルトのネットワーク実装は存在せず、ネットワーク要件のみが定められている
  - Pod はクラスタ全体でユニークな IP アドレスを持つ
  - クラスタ内の Pod は NAT やプロキシを介さず直接通信できる
  - ノードとそのノード上の Pod は NAT なしに通信できる

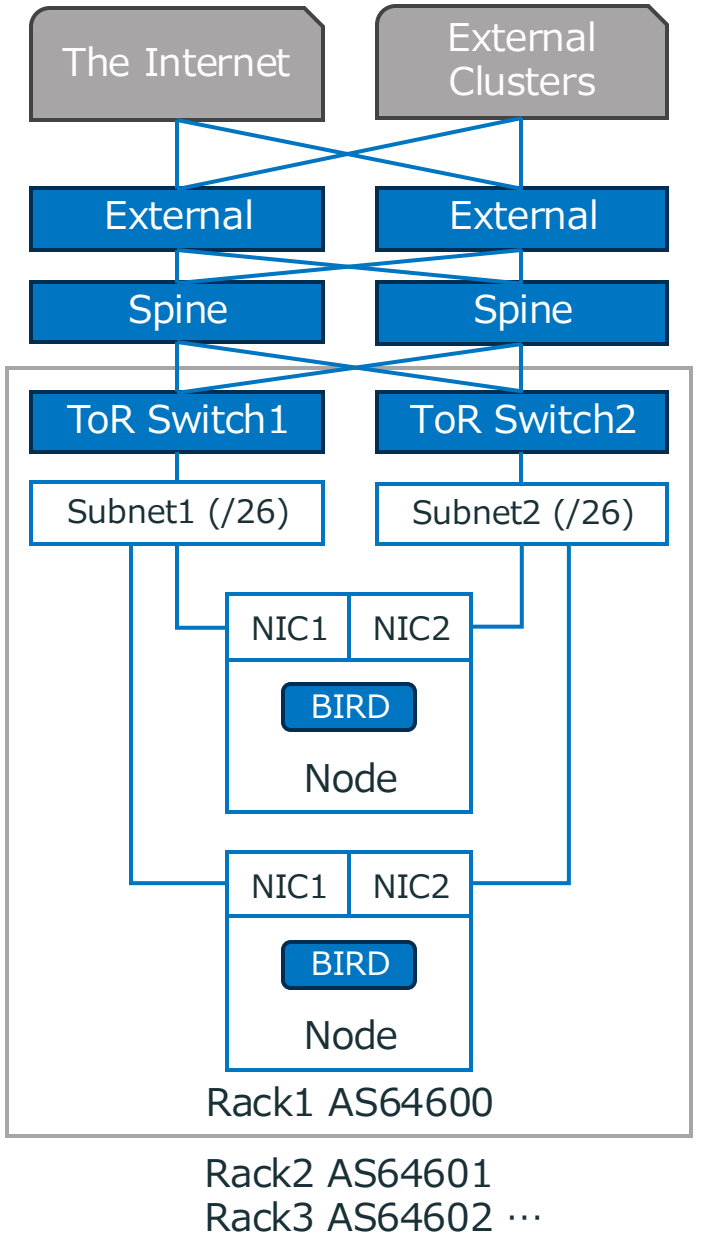
要件さえ満たせば実装方式は自由

- CNI プラグインは、上記の要件を満たすネットワークを Kubernetes に提供する
  - Neco 基盤が採用している CNI プラグイン
    - **Coil – サイボウズが開発する CNI プラグイン（OSS）**
    - **Cilium – eBPF を活用した高パフォーマンスな CNI プラグイン**
- kube-proxy
  - 実体が存在しない ClusterIP への通信を提供する
  - Kubernetes の標準コンポーネントであり、デフォルト実装が提供されている
  - **kube-proxy Replacement – Cilium が提供する、kube-proxy を代替する eBPF ベースの機能**

Neco のネットワークアーキテクチャ：ラックデザイン

シンプルな IP Clos

- 物理ネットワーク機材は Leaf-Spine 構成
- Spine と Leaf（ToR スイッチ）間は BGP Unnumbered で接続
- ラック単位に AS 番号を持つ（AS per Rack）
- 物理サーバー（Node）は NIC ごとに異なるサブネットに接続
- 物理サーバーの BIRD と ToR の間で BGP ピアを確立する
  - BIRD が Kubernetes のアドレスを広報する
- 外部接続は専用の L3 スイッチ（External）に集約

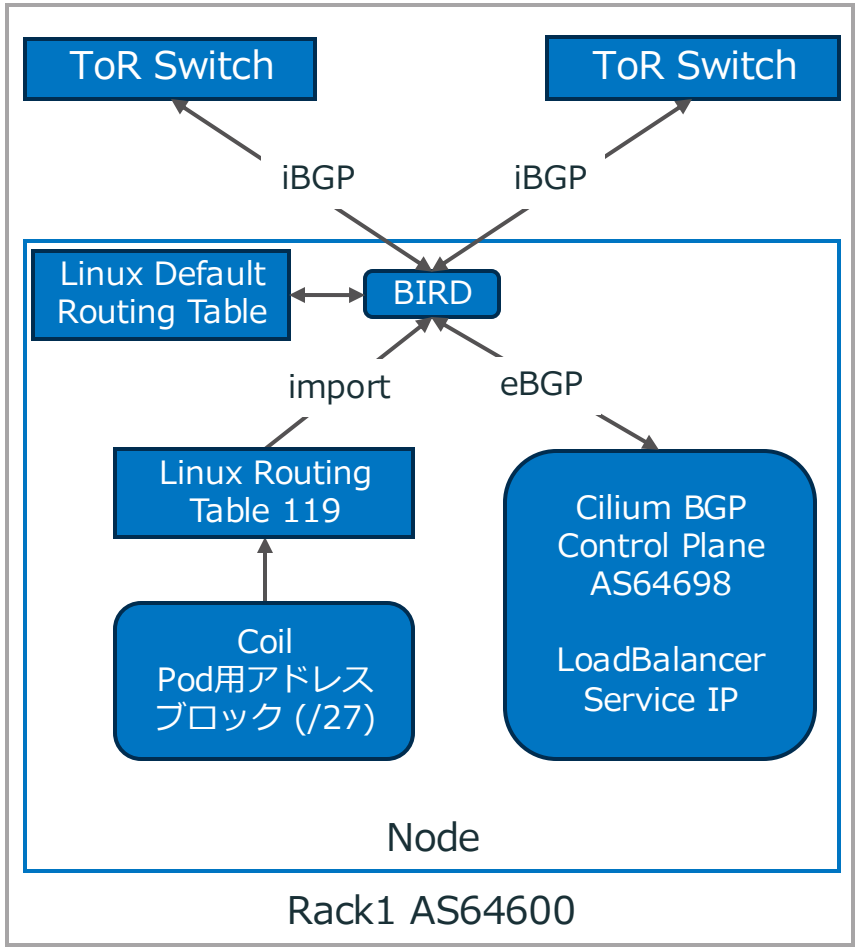


Neco のネットワークアーキテクチャ：ノードデザイン

- BIRD は ToR スイッチと BGP で経路交換をする
- BIRD からは以下の IP が広報される
  - Node Loopback IP
  - Pod IP
  - LoadBalancer Service IP

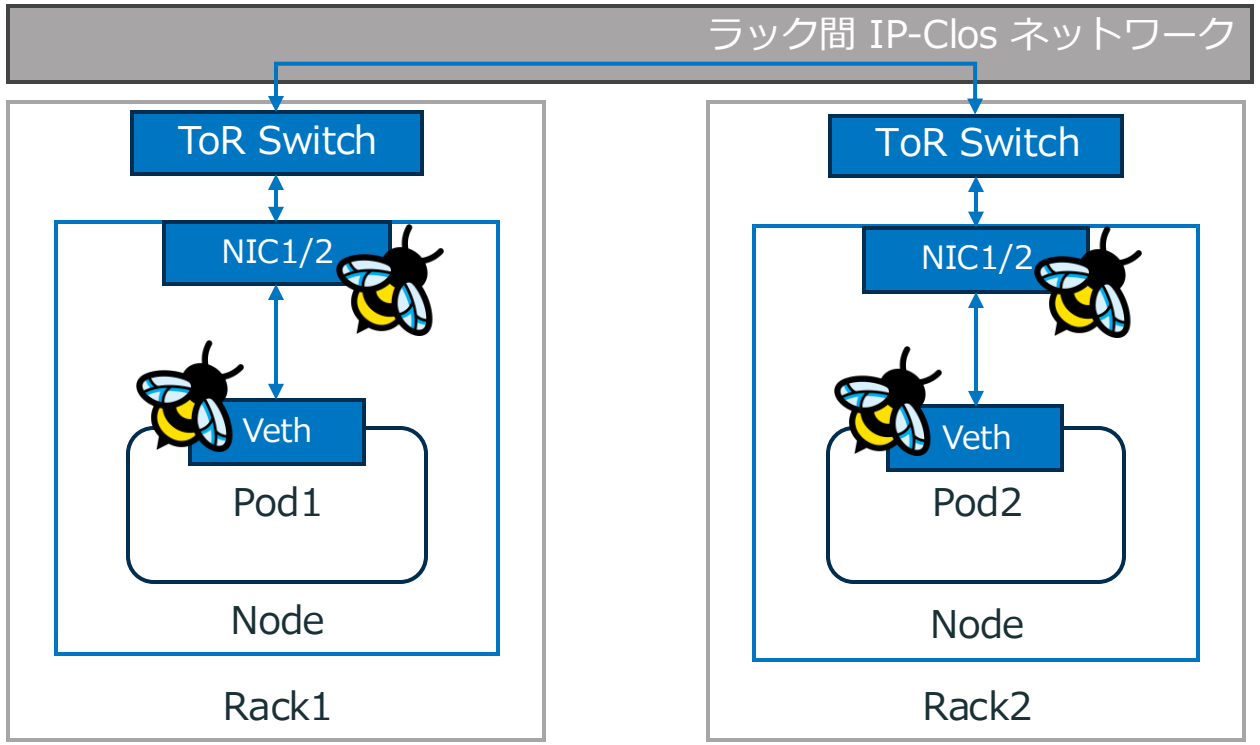
各 IP の由来

- Node Loopback IP
  - Linux Default Routing Table から取得される
- Pod IP
  - Coil の IPAM 機能によって割り当てられる
- LoadBalancer Service IP
  - Cilium の Load Balancer 機能によって割り当てられる



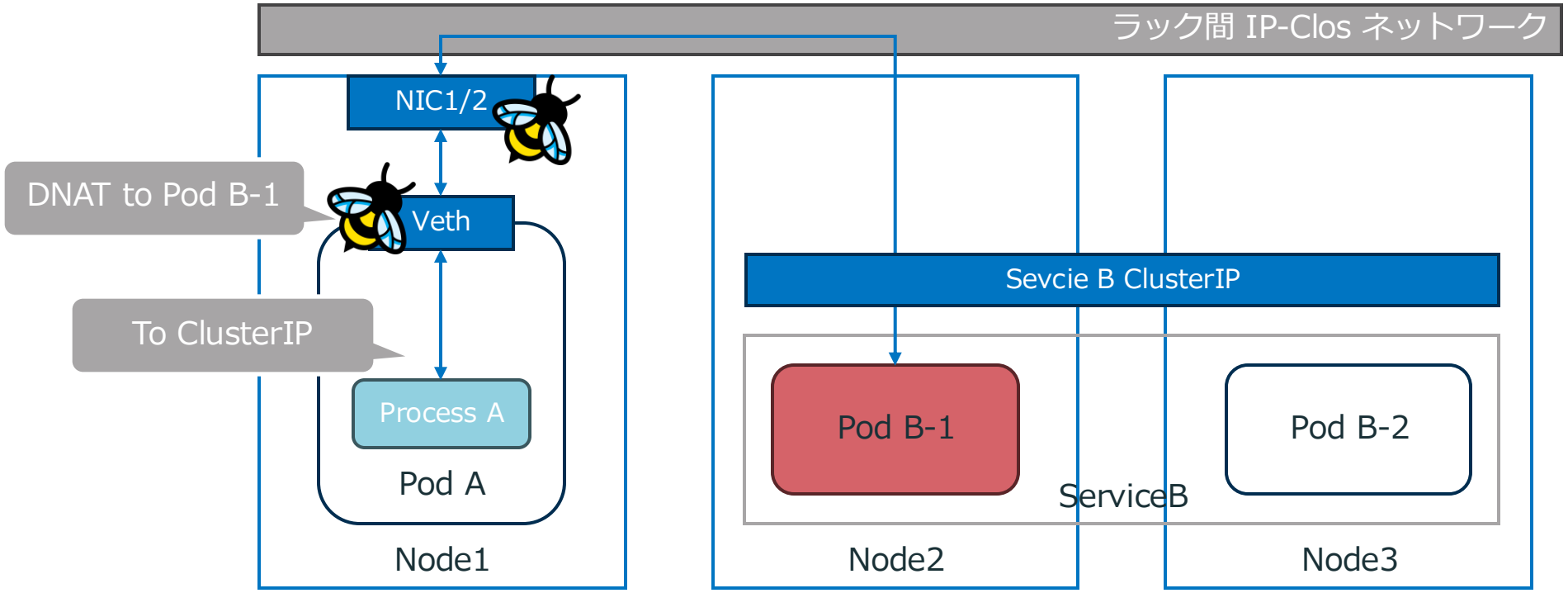
Neco のネットワーク(1) : Pod 間通信

- Coil IPAM
  - Pod に IP アドレスを割り当ててカーネルのテーブルに書き込む
- BIRD
  - Coil が書き込んだ経路をノードの外に広報
- Cilium
  - ノードの物理 NIC と Pod の Veth に eBPF プログラムをアタッチ
  - パケット転送
  - コネクション管理
  - ネットワークポリシー
  - パケットトレース



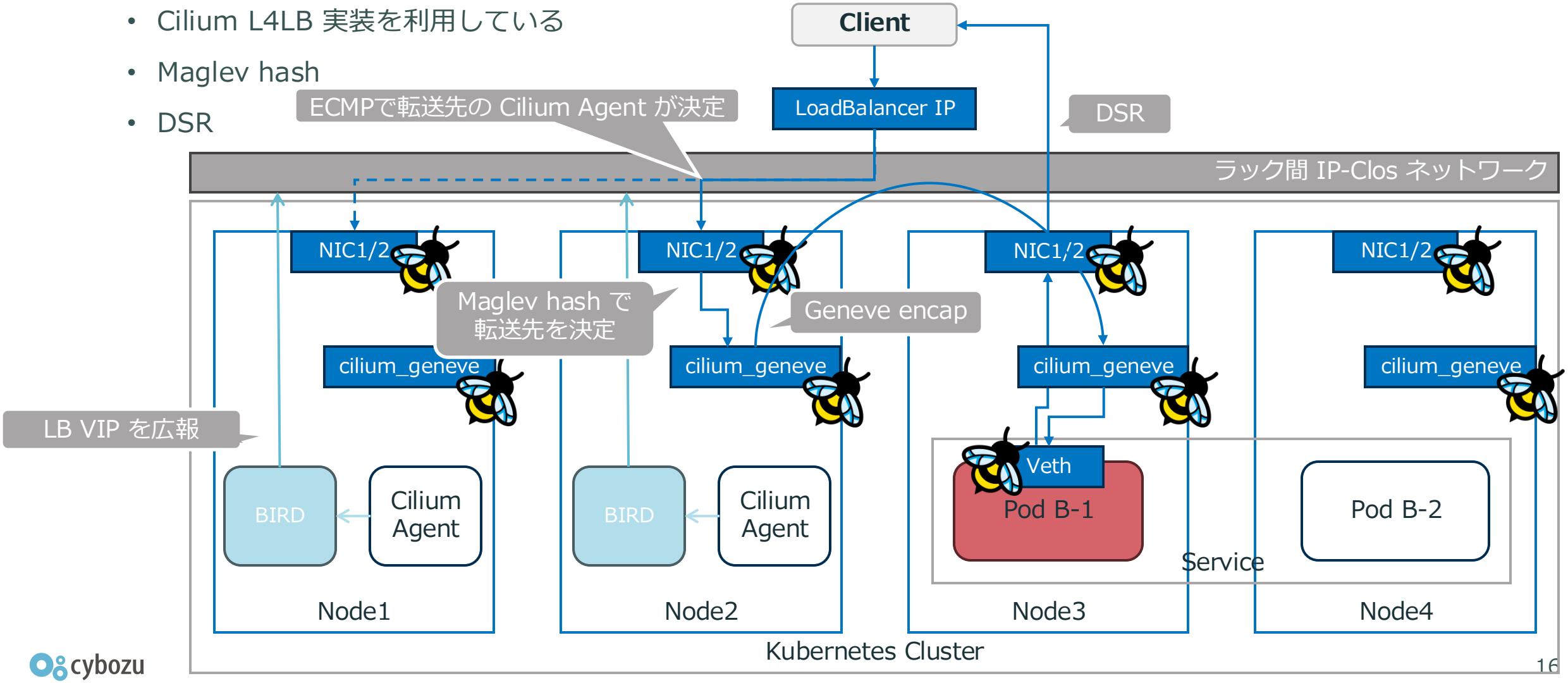
Neco のネットワーク(2) : ClusterIP への通信

- クラスタ内通信を抽象化する Service リソースの機能
- Cilium の kube-proxy Replacement で提供される
  - Pod にアタッチされた eBPF プログラム内でランダムに宛先 Pod を解決して DNAT する
  - 経路上のネットワークスイッチから見ると通常の Pod 間通信



Neco のネットワーク(3) : L4 Load Balancer への通信

- クラスタ外からの Ingress 通信を抽象化する Service リソースの機能
  - Cilium L4LB 実装を利用している
  - Maglev hash
  - DSR



互いのシステムに対する理解を深める

安定してシステムを運用する上で、挙動について認識しておきたいポイントがある

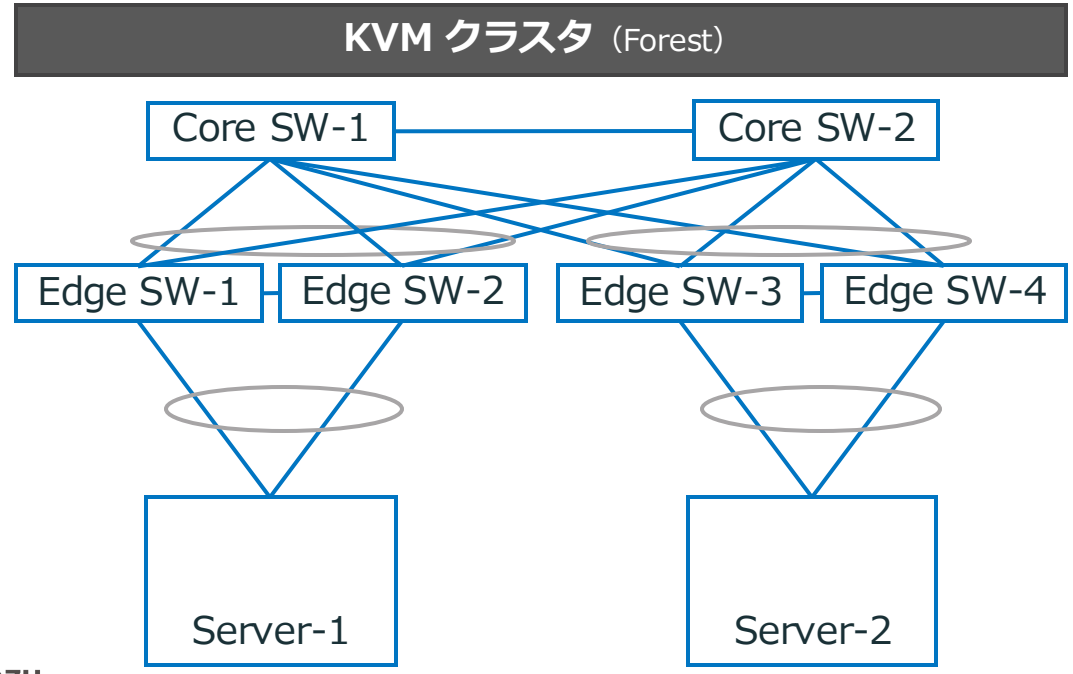
- NW&DC チームから見て、知っておきたい Kubernetes ネットワークの特徴
- Kubernetes ネットワークチームから見て、知っておきたいサイボウズネットワーク（あるいは一般の NW）の特徴

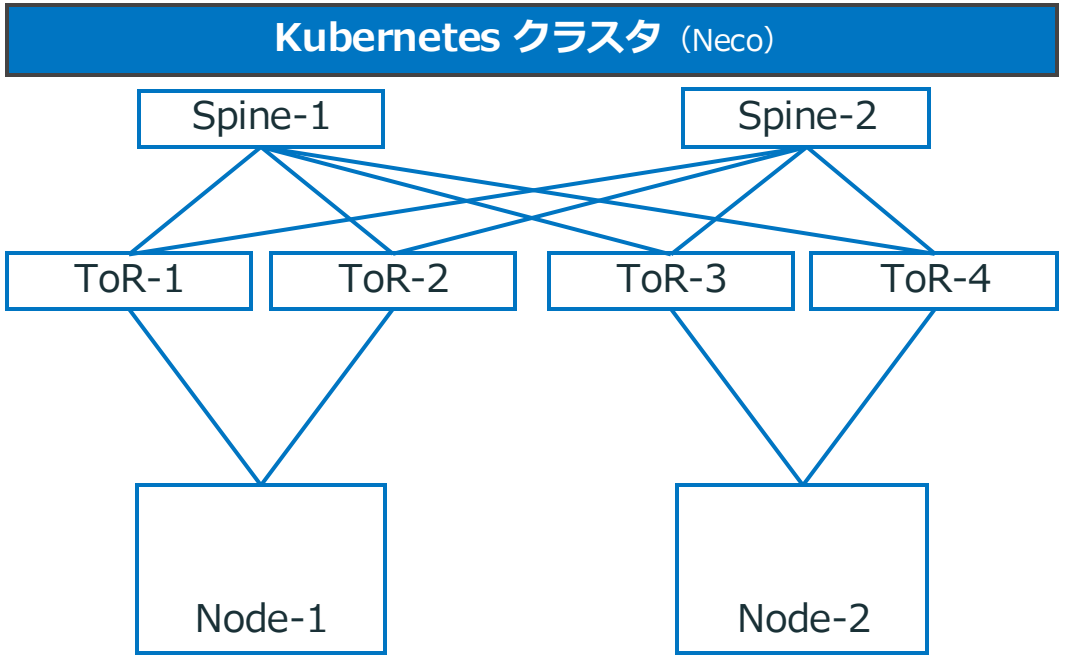
※ 一般論というより、我々の事例です

NW&DC チーム から見た Kubernetes ネットワークの特徴

Kubernetes ピュアL3ネットワーク採用による運用面のメリット

- クラスタ内のトラフィックエンジニアリングが可能になった
- 旧基盤ではサーバーから Core SW まで MC-LAG で冗長性を担保しており、リンクダウンが発生するとサービストラフィックのパケットロスは不可避
  そのためメンテナンスウィンドウの中でしか作業ができなかった
- Kubernetes クラスタではリンクの不調が疑われる場合、迂回処理をして作業が可能に





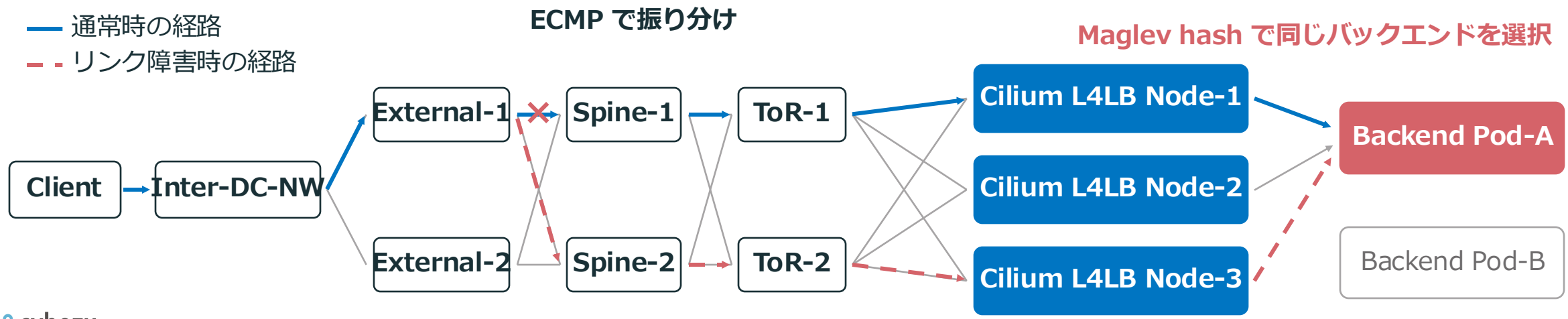
Cilium L4LB の仕様【Maglev hash】

トラフィックの迂回処理に際して出た懸念

- Kubernetes で稼働している L4LB に対して ECMP でパケットが振り分けられる
- メンテナンスに伴い ECMP のハッシュが変更になると着信する LB Node が変わる
- 着信する LB Node が変更になってしまうが、TCP セッションは維持されるのか？

結論

- L4LB は Maglev hash アルゴリズムにより、どの LB Node からも同一のバックエンドの Pod を選択するため、セッションは切れない



NW&DC チームが認識しておきたいこと

IPよりも上位のレイヤーでサービスの冗長性が担保されていることがある

スイッチのルーティングテーブルに載る IP
= ネットワーク機器がルーティングに責任をもつ経路

- Node IP
- Pod IP
- Service IP（type: LoadBalancer）

ルーティングテーブルに載らない IP
= ネットワーク機器は以下のIPが宛先となるパケットを受け取らない

- Service IP（type: ClusterIP）
  - Service IP 宛のパケットの宛先IPはバックエンドの Pod の IP に書き換えられる
    - Cilium の eBPF による

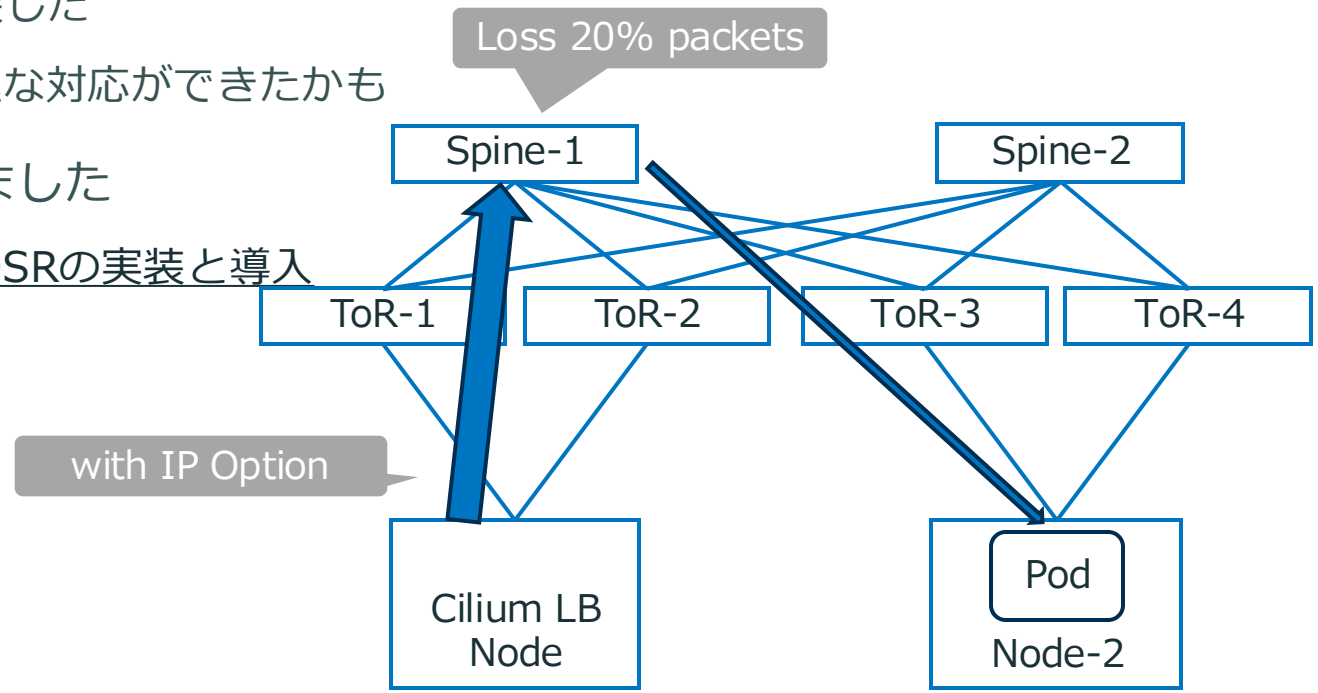
対話のためには採用している CNI の外部仕様を理解することが重要。

Kubernetes ネットワークチーム から見た サイボウズネットワークの特徴

ネットワーク機器による制約

IP Option と Layer 2 slow path

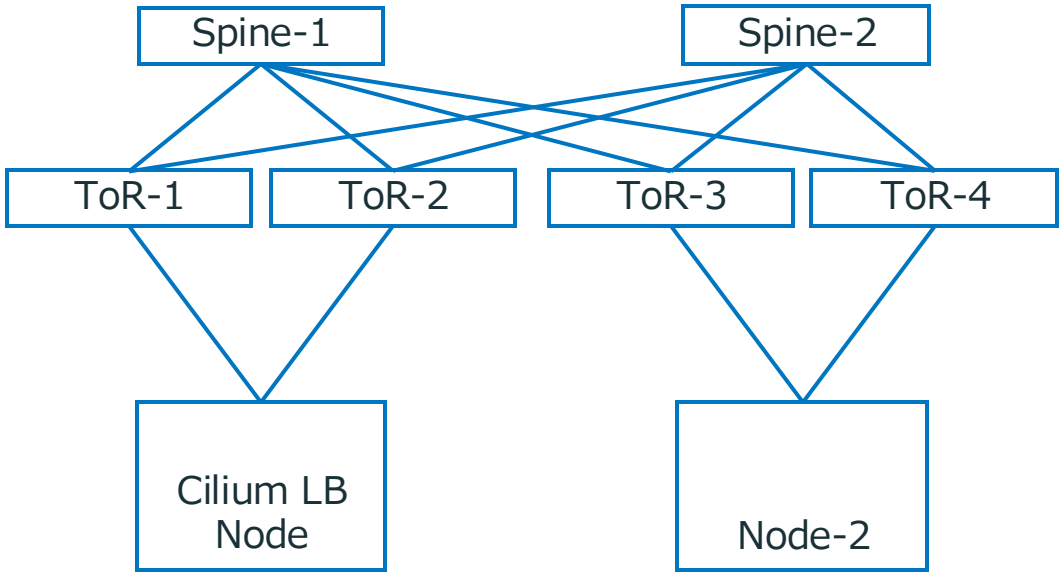
- Cilium L4LB IP Option mode を利用していた際の問題
  - LB Node から Pod がいる Node へ転送する際にカスタム IP Option を付与していた
  - 転送パケットが Spine の L2 slow path に入って閾値を超えた結果約 20% のパケロスが発生した
- ソフトウェアエンジニアで調査して知識 0 から原因究明・対策を実装した
  - IP Option の代わりに Geneve encap で実装した
  - 早い段階で NW チームと連携していたら迅速な対応ができたかも
- 詳細は Cloud Native Days 2023 で発表しました
  - CiliumにおけるGeneveプロトコルを用いたDSRの実装と導入



ネットワーク機器による制約

Cilium L4LB と Spine の ECMP 経路数上限

- Cilium L4LB は大量の汎用サーバを利用して可用性・耐障害性を実現している
  - L4LB 全体のキャパシティは **LB Node 数 x 単一ノードのスループット**
  - 理論的には LB Node を増やせば無限にスケールするはず…
- 実際はスイッチの ECMP 経路保持数の上限が L4LB のキャパシティ限界になってしまう
  - Neco だと 1 ラックあたり 2 台の ToR が接続する
  - → **ラック内の機材数 x (Spine の経路保持数 / 2) が L4LB のキャパシティ上限**



Kubernetes ネットワークチームが認識しておきたいこと

IPを操作しているのは、物理機材も同等であり、その性質を認識しておく必要がある

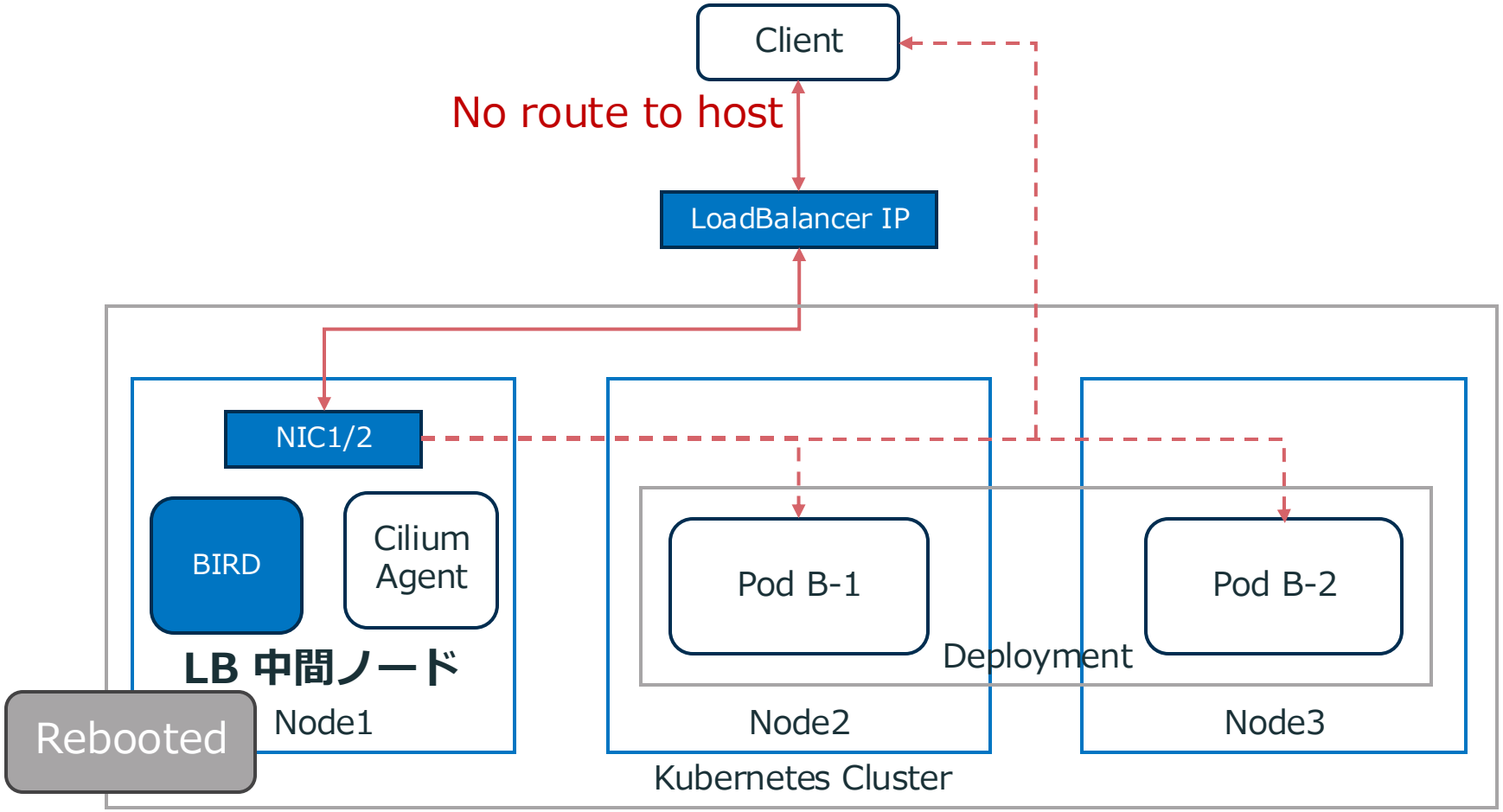
- 物理機器の限界がソフトウェアの理論的性能をキャップすることがある
- ソフトウェアの性能がネットワーク機器の性能を最大限引き出せないことがある
- 予期しないところで物理的制約を受けることがある

対話のためには採用している物理ネットワークの特徴を理解することが重要。

障害事例と対話による解決

協力して解決した問題の実例

- Cilium L4LB を経由してクラスタ内に接続するクライアントが “No route to host” を一時的に受け取る
  - クラスタ内の全ノードの順次再起動（全台再起動）を実施していた
  - 全台再起動に伴って度々観測されており、アプリケーション側のコネクション断として顕在化した



ソフトウェアエンジニア視点からの原因調査

全台再起動によるノードの再起動が関連すると推測

↓

当該時刻の再起動ノードを特定し、BIRD や Cilium Agent 等の関連ソフトウェアの終了、起動の時系列を整理

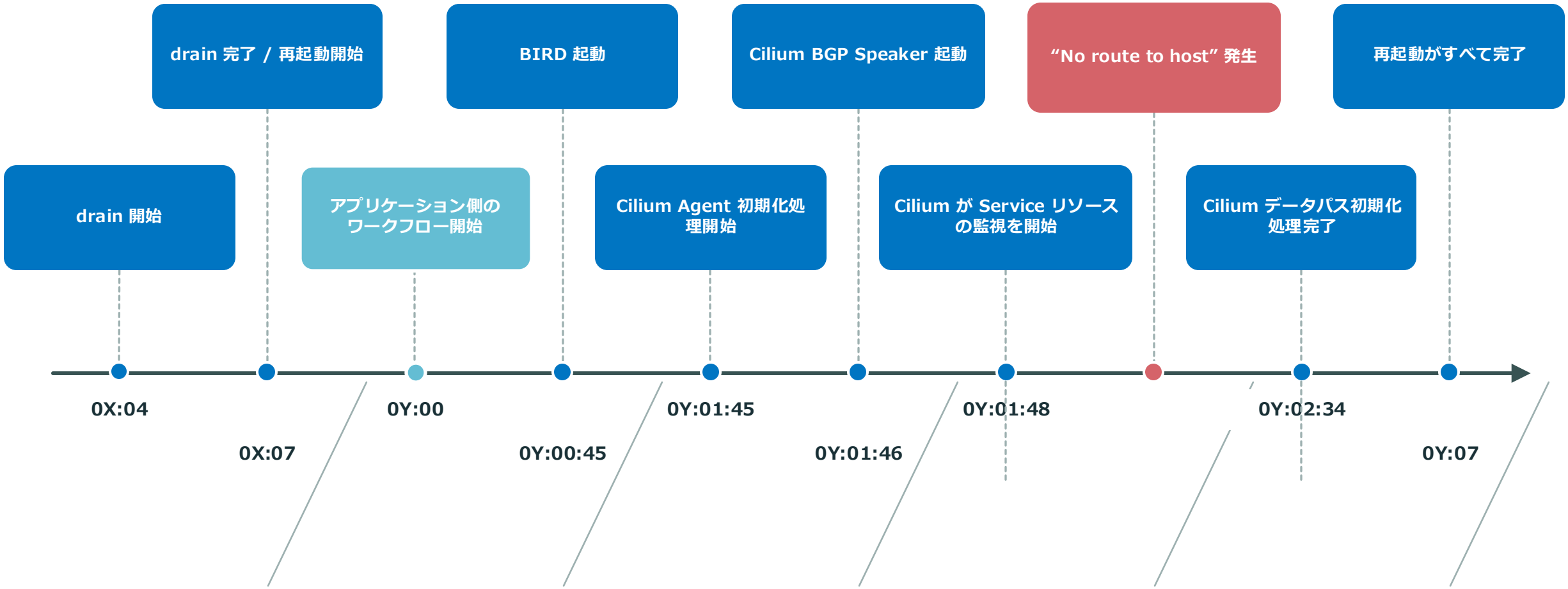
↓

Cilium Agent の起動ログやメトリクスを精査

↓ ↑

気になるログ付近のソースコードを参照しつつ現象と突き合わせて仮説を立てる

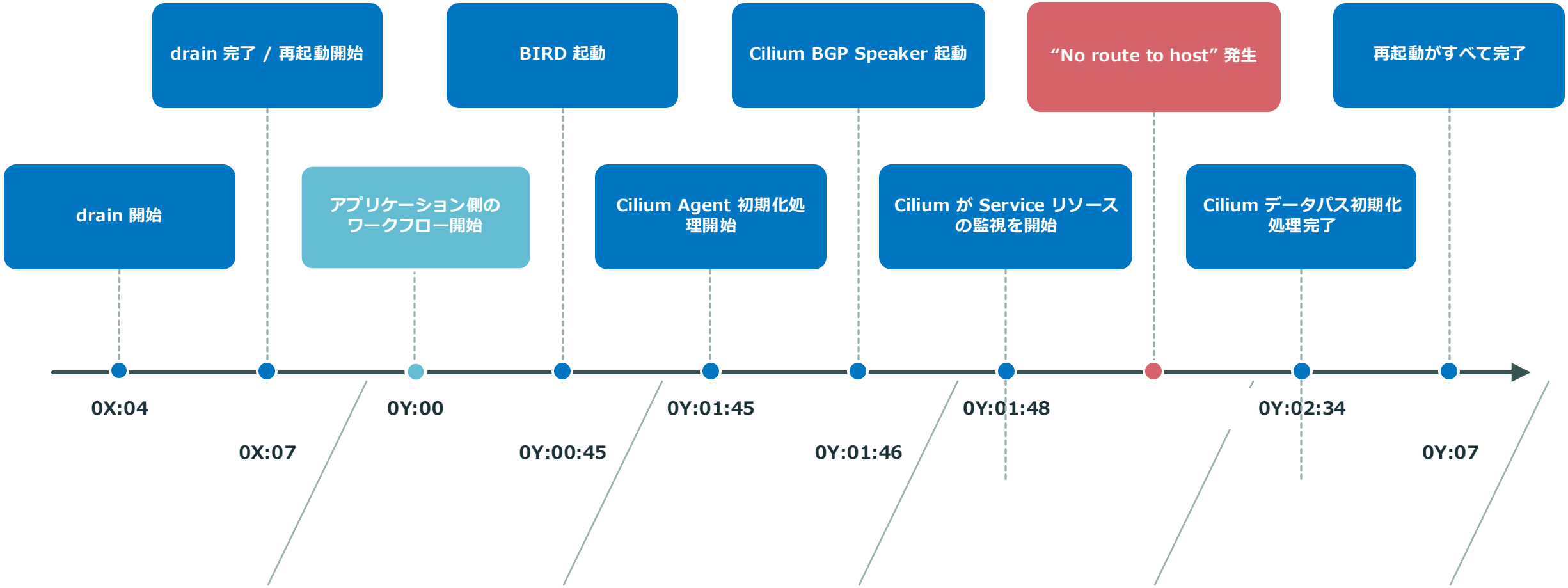
障害のタイムライン


drain 開始
drain 完了 / 再起動開始
アプリケーション側のワークフロー開始
BIRD 起動
Cilium Agent 初期化処理開始
Cilium BGP Speaker 起動
Cilium が Service リソースの監視を開始
“No route to host” 発生
Cilium データパス初期化処理完了
再起動がすべて完了
0X:04
0X:07
0Y:00
0Y:00:45
0Y:01:45
0Y:01:46
0Y:01:48
0Y:02:34
0Y:07

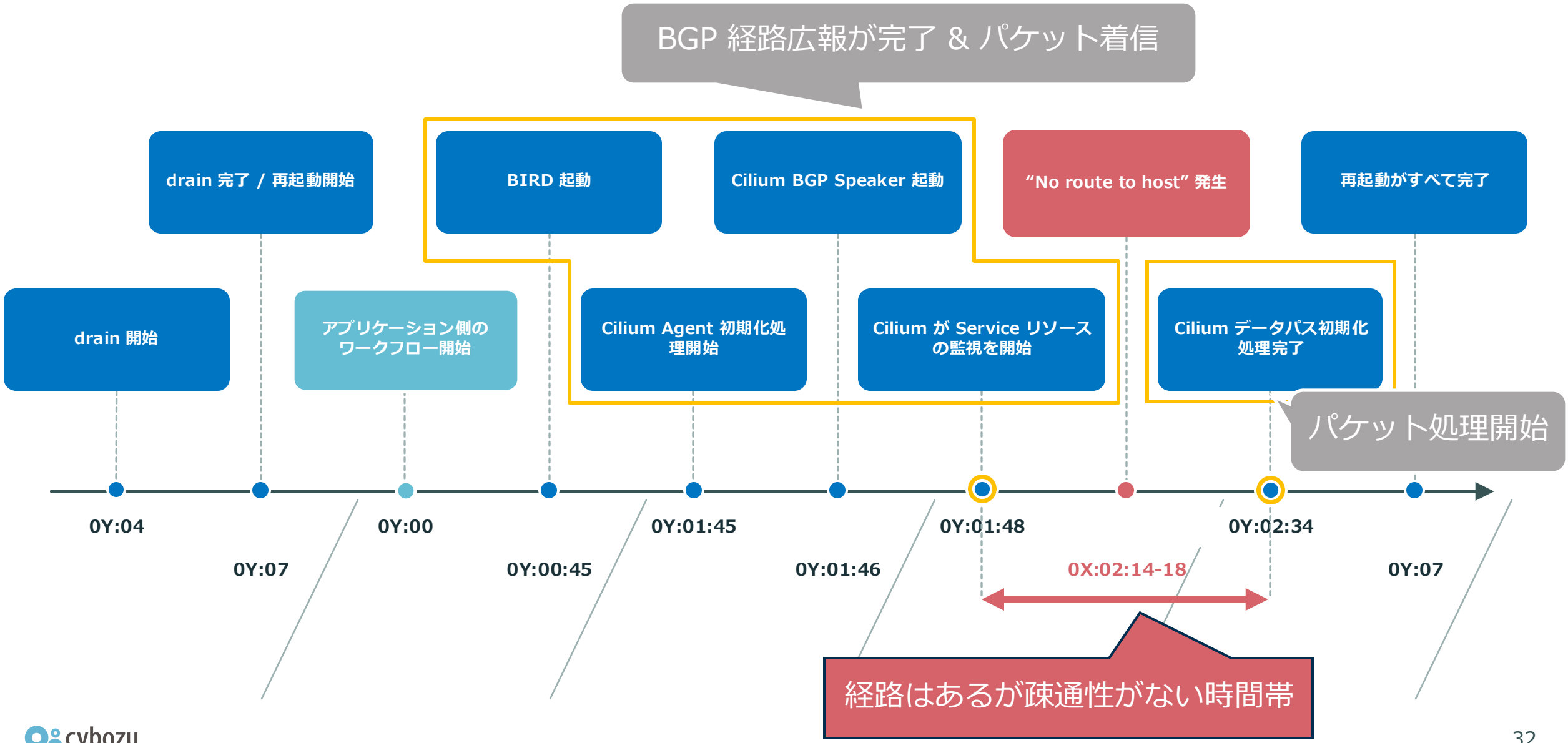
障害のタイムライン

尾崎さん

タイミング的にcilium がパケット転送 ready になる前に、BGP で経路が配布されてしまったことが原因のように見えますね

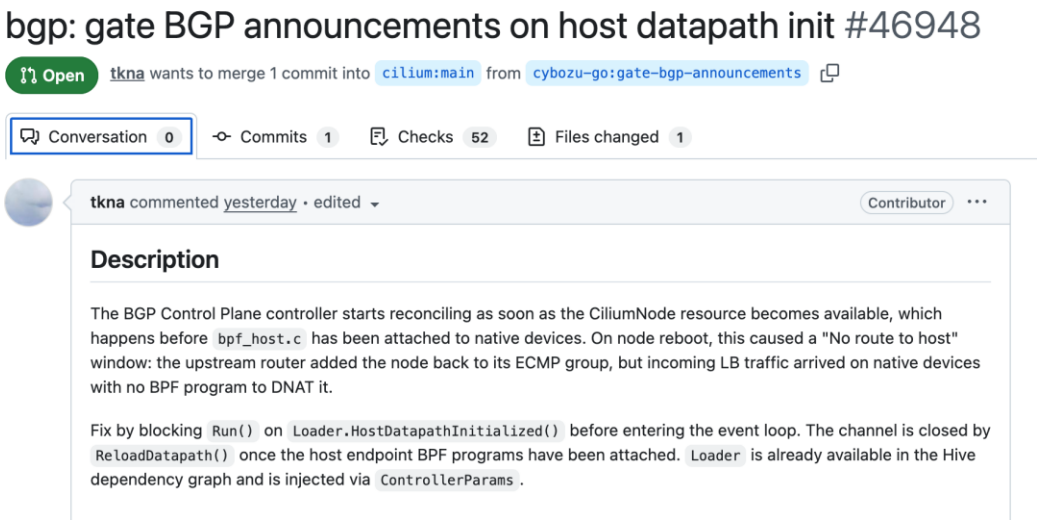


障害のタイムライン


BGP 経路広報が完了 & パケット着信
drain 開始
drain 完了 / 再起動開始
アプリケーション側のワークフロー開始
BIRD 起動
Cilium Agent 初期化処理開始
Cilium BGP Speaker 起動
Cilium が Service リソースの監視を開始
“No route to host” 発生
Cilium データパス初期化処理完了
再起動がすべて完了
パケット処理開始
0Y:04
0Y:07
0Y:00
0Y:00:45
0Y:01:45
0Y:01:46
0Y:01:48
0X:02:14-18
0Y:02:34
0Y:07
経路はあるが疎通性がない時間帯

解決

- Cilium Agent の初期化時の非同期性が問題を引き起こしている疑いが強いと分かった
- 再現実験
  - 検証用の仮想データセンタ環境（dctest）や Containerlab + Kind の環境を使って BGP の設定と Cilium のコードを変更して実験
  - ここでも DC ネットワークの知識とソフトウェアの知識を組み合わせて手順を用意した
  - 再現に成功し、問題の原因を特定
- あとはソフトウェアエンジニアの領域
  - bgp: gate BGP announcements on host datapath init #46948



bgp: gate BGP announcements on host datapath init #46948

Open tkna wants to merge 1 commit into cilium:main from cybozu-go:gate-bgp-announcements

Conversation 0 Commits 1 Checks 52 Files changed 1

tkna commented yesterday • edited Contributor

Description

The BGP Control Plane controller starts reconciling as soon as the CiliumNode resource becomes available, which happens before `bpf_host.c` has been attached to native devices. On node reboot, this caused a "No route to host" window: the upstream router added the node back to its ECMP group, but incoming LB traffic arrived on native devices with no BPF program to DNAT it.

Fix by blocking `Run()` on `Loader.HostDatapathInitialized()` before entering the event loop. The channel is closed by `ReloadDatapath()` once the host endpoint BPF programs have been attached. `Loader` is already available in the Hive dependency graph and is injected via `ControllerParams`.

```
nyamber@take:~$ while true;do sudo ip netns exec external nc -vz 172.19.0.16 80; sleep 1;done

Connection to 172.19.0.16 80 port [tcp/http] succeeded!
Connection to 172.19.0.16 80 port [tcp/http] succeeded!
Connection to 172.19.0.16 80 port [tcp/http] succeeded!
nc: connect to 172.19.0.16 port 80 (tcp) failed: No route to host
nc: connect to 172.19.0.16 port 80 (tcp) failed: No route to host
(snip: 上記含め179行)
Connection to 172.19.0.16 80 port [tcp/http] succeeded!
Connection to 172.19.0.16 80 port [tcp/http] succeeded!
Connection to 172.19.0.16 80 port [tcp/http] succeeded!
```

これで中間ノードがいなくなるとNo route to hostになることは確認できた。

何が解決に向けた肝だったか？

- フラットに議論できる土台ができていた
  - オープンなチャットで互いに作業を覗いて、気になった時にコメントできる
- それぞれの専門を背景として、違うアプローチで同じ問題に取り組める
  - 問題解決のためには普段と異なる視点からのアイデアが有効なときがある

- 互いの専門領域の暗黙知が共有されにくい
  - 意外と専門外の情報は何も知らない
- 一つの専門チームだけで解決できる問題は少ない

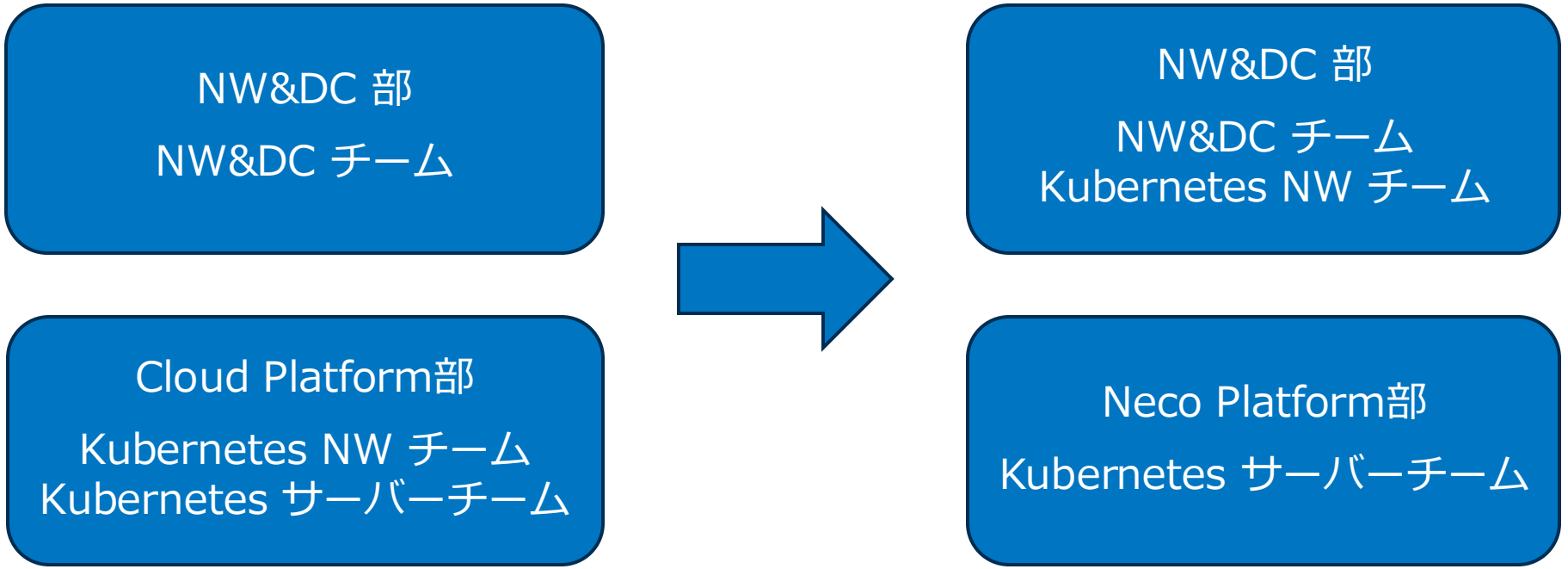
我々の悩み

プラットフォーム運用には広範な知識が求められる

- 弊社で遭遇した不具合と、その中で必要だった「対話」について紹介しました。
- テクニカルな面白さは数多くありますが、今回はもう少し一般的なことについて議論をしたいです。
- プラットフォーム運用は、決して NW の運用と切り離すことが出来ない。
- 専門家がいる分野を横断するような、幅広い知見が要求される中で、プラットフォームの知見と NW の知見をどうやって統合して運用していくかはとても悩ましい問題。

「Necoという基盤を作る」フェーズから「対話による知識の統合」のフェーズへ

- Neco という基盤を立ち上げた際には、Neco というカットで組織が切られていた。
- 対話の重要性を認識し、人的/知識的交流を目的として組織を割り直した。
- 幅広い問題に対処できる、面白いチームが出来上がったと考えています。



プラットフォーム運用における知識交流の問題をもっと議論したい

- あれ？じゃあ Neco のサーバーチームとの交流はおそろかになって良いの？
  - 実務上の交流を続けて、なんとか知識交流を続けている
- 我々はチームを割り直し続けたいわけではない。
- 組織論に頼らない、新たな知識交流の方法が強く求められているのでは？
- ソフトウェア業界でもこの辺りが大きな問題になっているように見え、NW 業界からも知見を吸収したい
- 強力な AI の登場で、そのあたりへのアプローチが出来れば良いなと考えているものの…

- 皆様はどう思われますか？


cybozu