JANOG52
CNFって使っていますか？

F5ネットワークスジャパン合同会社

中嶋大輔

Who am I ?

名前：中嶋 大輔

所属：F5ネットワークスジャパン

タイトル：ソリューションアーキテクト

過去の発表

- JANOG50: 集まれSRv6の森
- JANOG48: NaaS – Cloud Native時代のネットワークのあり方
- JANOG47: エッジコンピューティング時代のサービス運用の課題
- JANOG41: NFV+SDN標準化/実装動向と運用の課題

発表内容

1. CNFとは？
    1. 既存のオペレーションとクラウドネイティブなオペレーション
    2. CNFのユースケース
2. CNF基盤としてのKubenretes
    1. Kubernetes ネットワークの基本
    2. ネットワークファンクションの利用
3. デモ
    1. FRRoutingのコンフィグと立ち上げ
    2. FRRoutingのアップグレード
4. まとめ

CNFとは？

CNF (Cloud-Native Network Function)はCNCFなど様々なところで定義されています。CNFとはクラウドネイティブアプリケーションと同様に、コンテナや宣言型APIなどを使用したネットワーク機能の事を指します。

例えばCNCFのドキュメントでは以下のような一文があります。

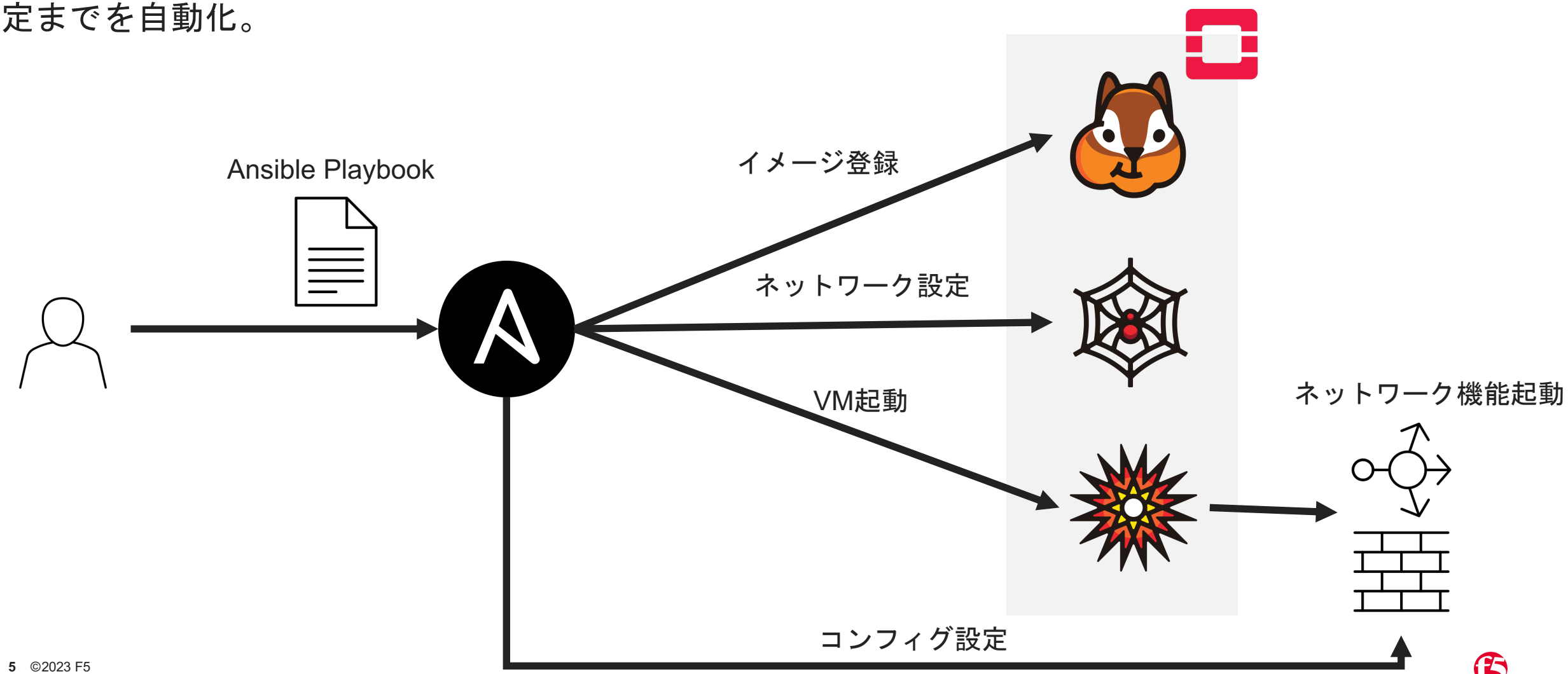
「クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実行するための能力を組織にもたらします。 このアプローチの代表例に、コンテナ、サービスメッシュ、マイクロサービス、イミュータブルインフラストラクチャ、および宣言型APIがあります。」

https://github.com/cncf/toc/blob/main/DEFINITION.md#%E6%97%A5%E6%9C%AC%E8%AA%9E%E7%89%88

今回のセッションでは、**Kubernetes基板上で動作する、コンテナ化されたネットワーク機能とその周辺技術**をCNFとして扱います。

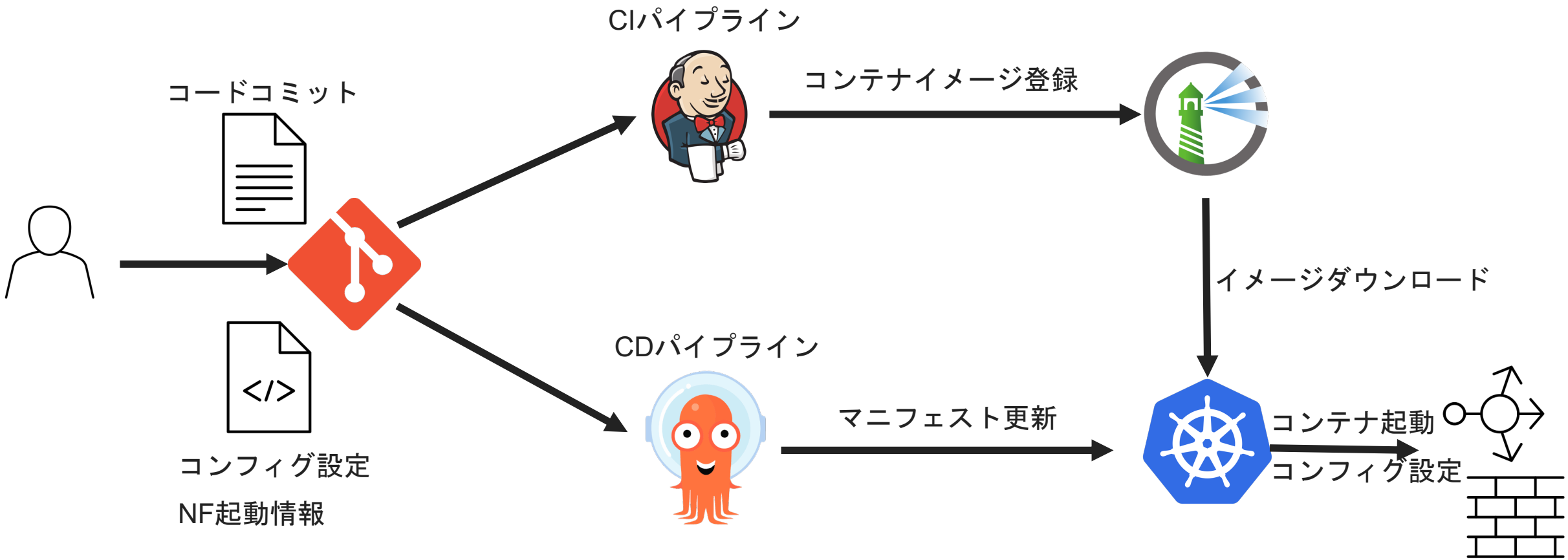
既存のVNFオペレーション

仮想基板上に仮想マシンとしてネットワーク機能を作成。Ansibleなどで立ち上げからコンフィグ設定までを自動化。



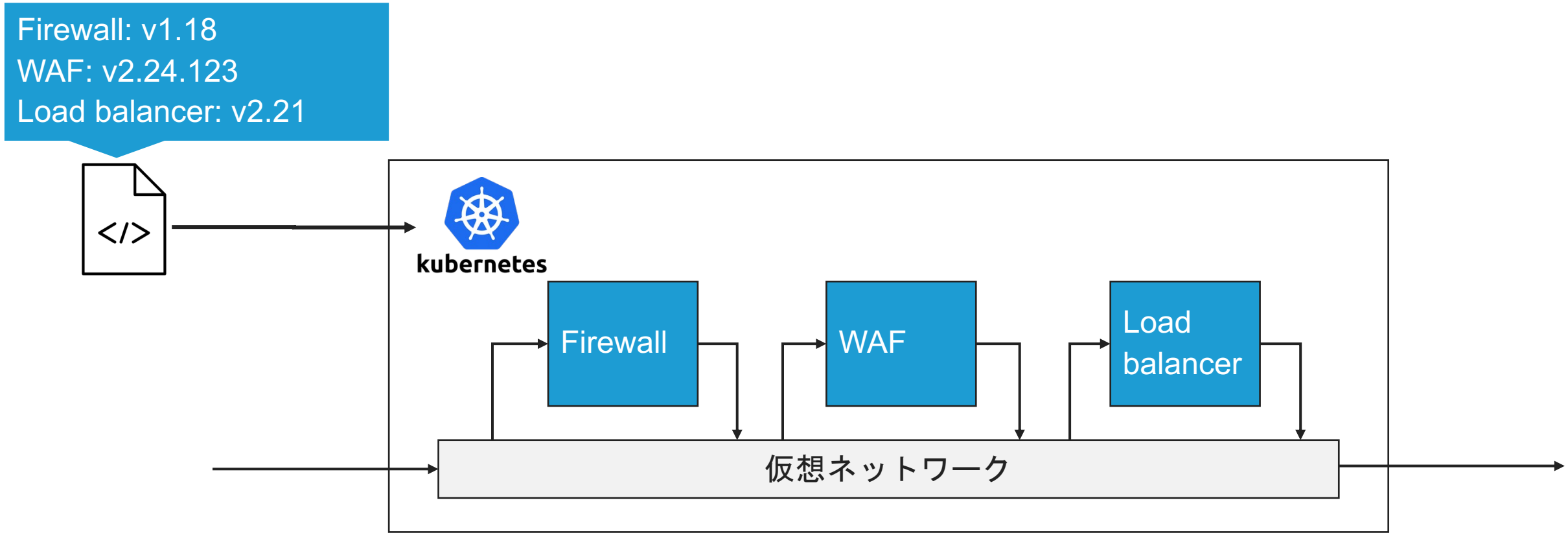
Cloud nativeなオペレーション

ネットワーク機能をコンテナ化し、Manifestというファイルでコンテナのコンフィグや状態を宣言。
KuberentesはManifestをもとにコンテナをPodとして起動し、その状態を維持しようとする。



CNFのユースケース

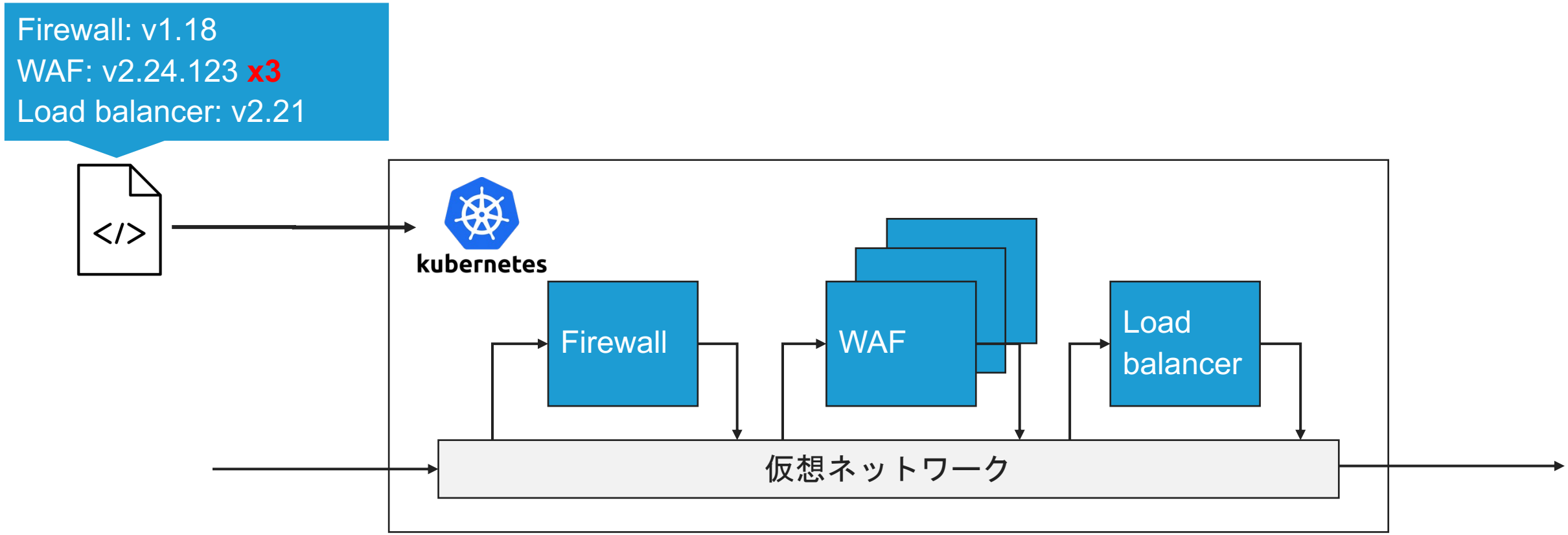
ネットワーク機能をマニフェストとして定義し、Kubernetes基板上に機能をデプロイ。
バージョンアップやスケールアウトはマニフェストを更新することで実現。
GithubやCI/CDツールと組み合わせることで、テストやデプロイの自動化も視野に。



CNFのユースケース

ユーザートラフィックの増加などで、特定の機能の負荷が増加した場合、コンテナ数を増やすことでスケールアウトが可能になるケースがあります。

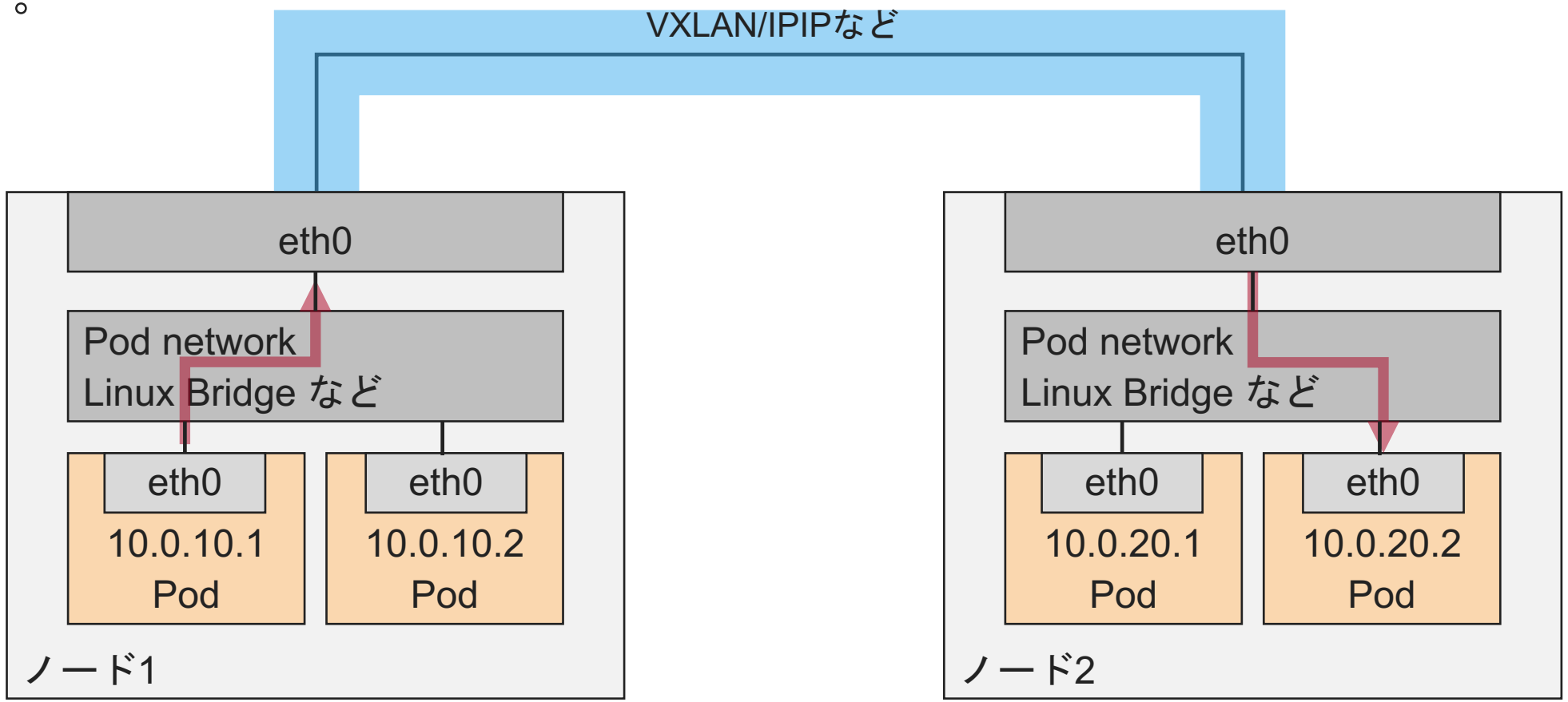
この場合、マニフェストを修正するだけでコンテナ数の増減が可能です。



Kubernetesネットワークの基本

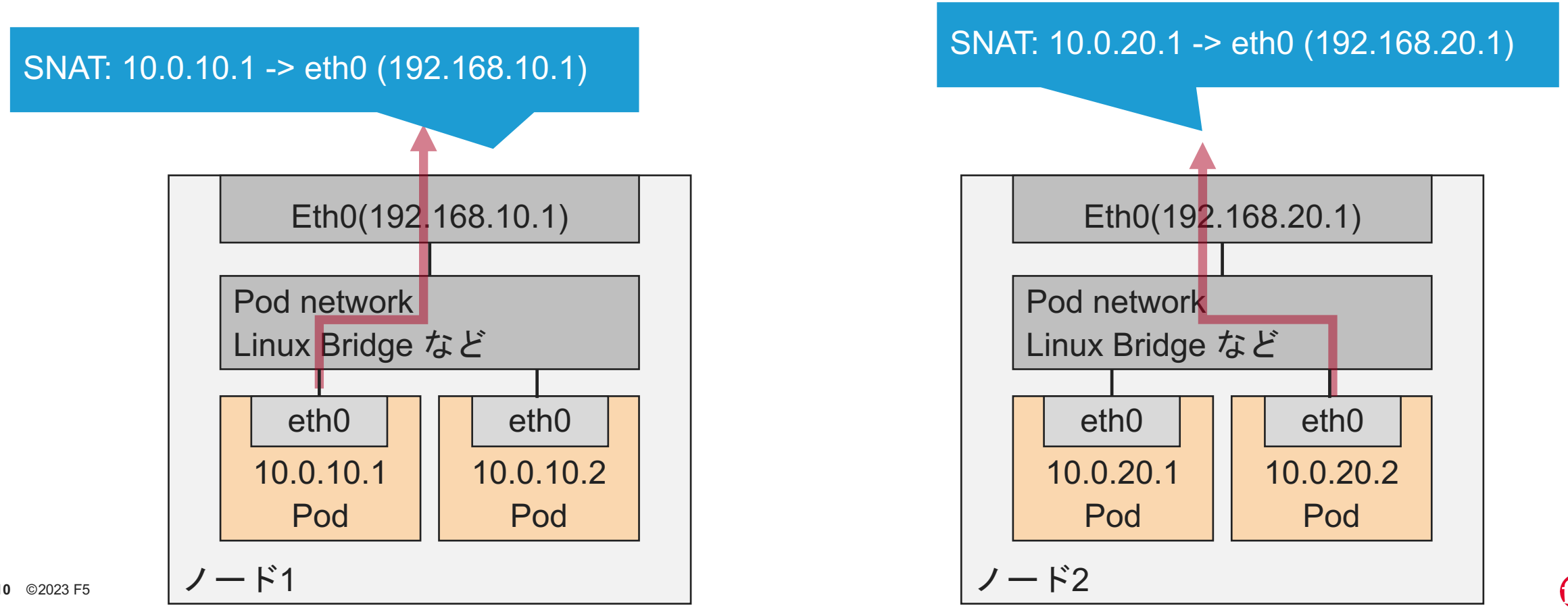
Pod (コンテナ）はPod Networkと呼ばれる内部ネットワークに接続します。Podは基本的に1ポートのみPod networkに接続します。

異なるノードのPod通信はVXLAN/IPIPなどでカプセル化し、通信する方式が多く採用されています。



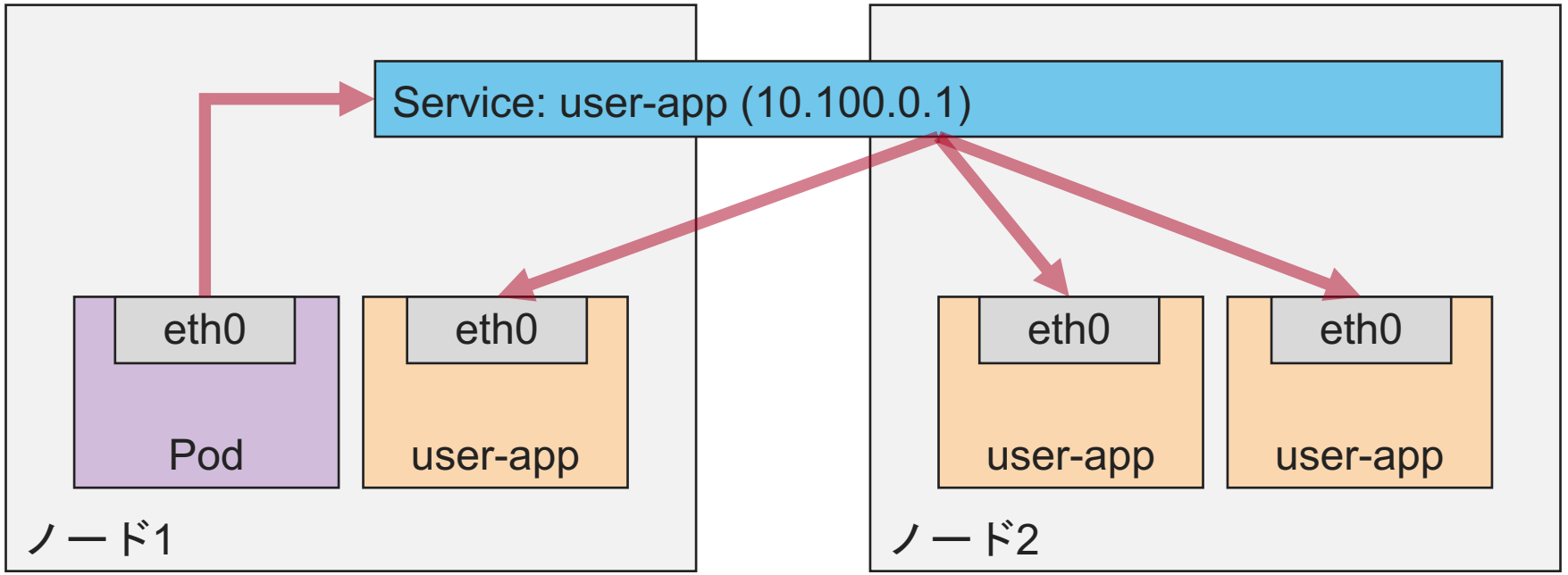
Kubernetesネットワークの基本

外部ネットワークと通信する場合、PodのIPアドレスはノードの物理インターフェイス(eth0)のアドレスにSNATされます（例外もあります）。



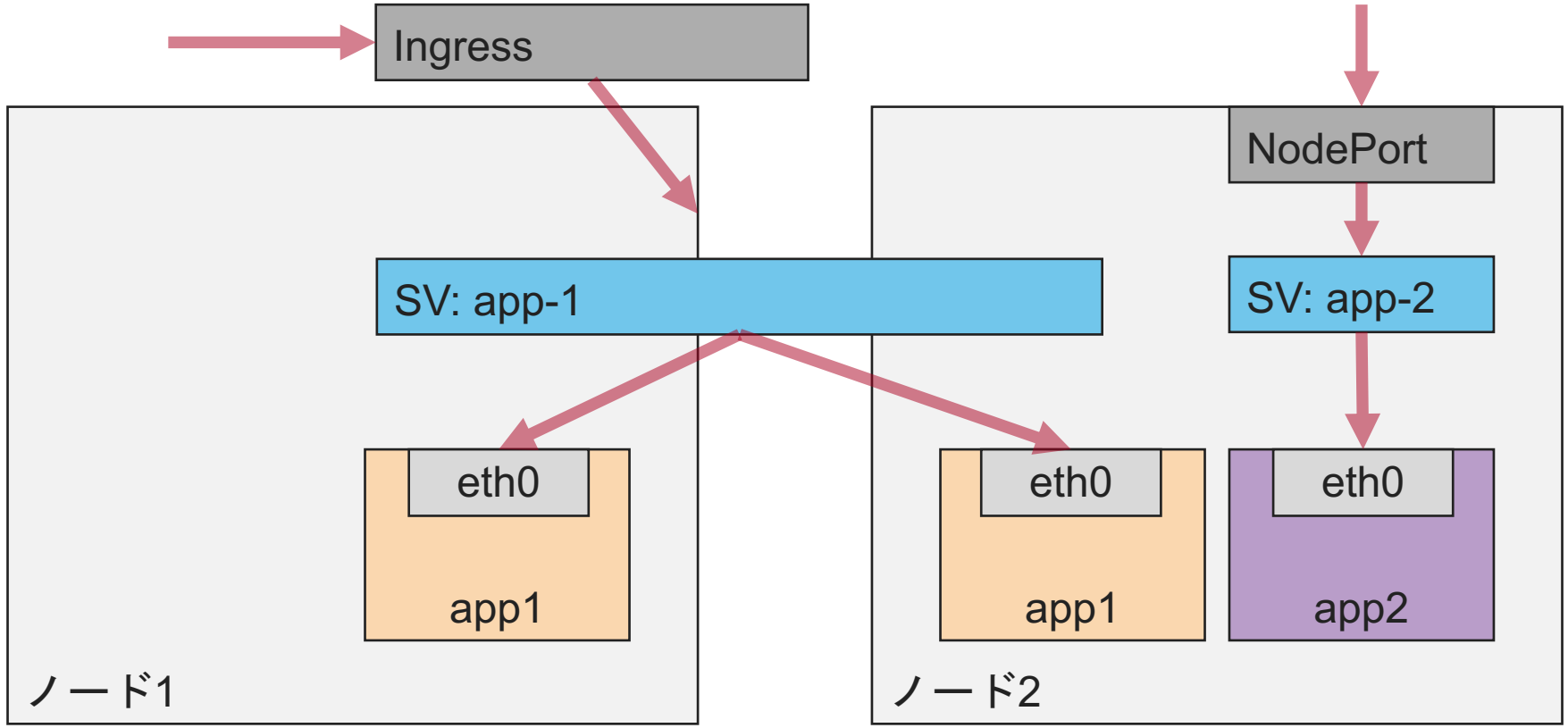
Kubernetesネットワークの基本

Podへのアクセスの負荷分散は「Service」と呼ばれる機能で解決します。Serviceにはドメイン名が設定され、PodはServiceのIPアドレスを名前解決します。Serviceは複数のPodにトラフィックを分散します。ServiceはiptablesやIP Virtual Serverなどで実装されることが多いです。



Kubernetesネットワークの基本

外部ネットワークからPodへの通信はIngressやNodePortといったオブジェクトを使うことで可能になります。Ingressは外部の別ノード、NodePortは予約した特定ポートを使用し、外部通信をKubernetes Service経由でPodに通信します。この時送信先IPはNATされます。



Kubernetesネットワークまとめ

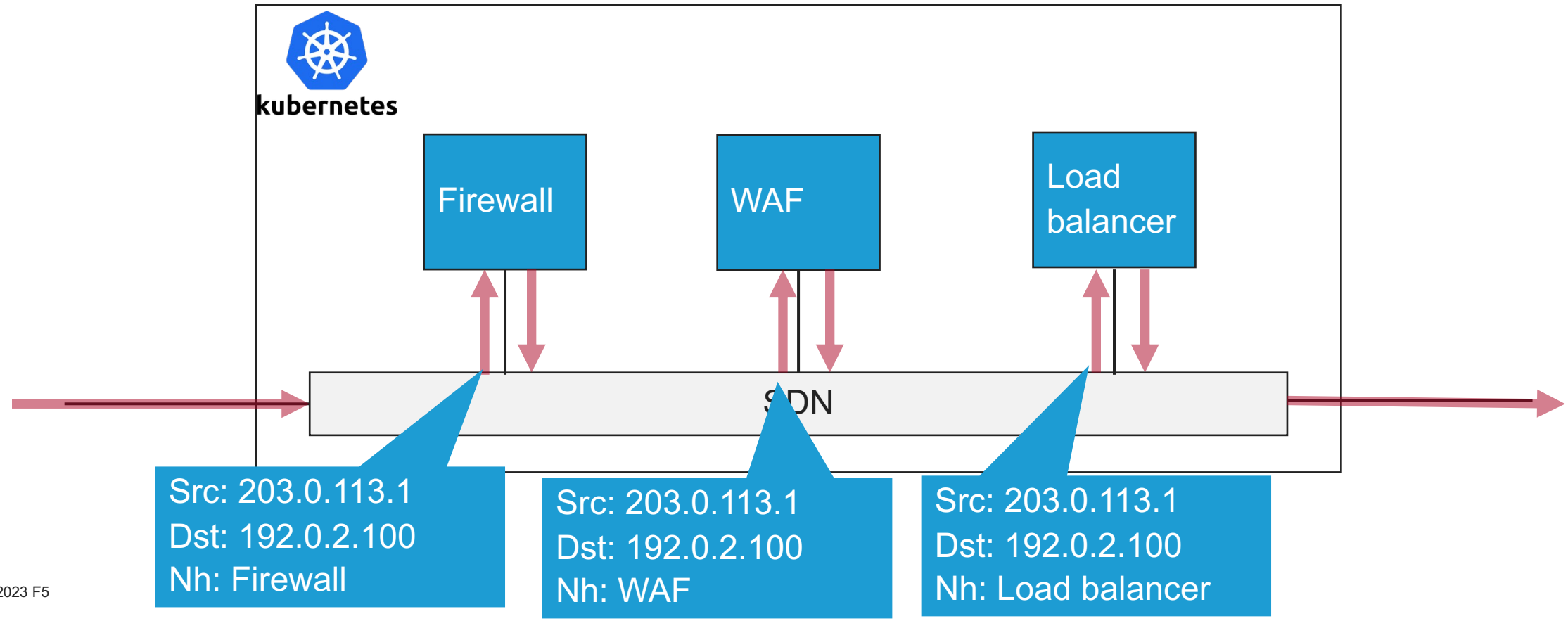
- Pod NetworkやKubernetes Serviceなどでネットワークが抽象化
- Podの仮想NICは1つのみ
- 外部ネットワークとの通信ではPodのIPアドレスはNATされる
  - EgressはSNAT
  - IngressはDNAT

どうやってルータやファイアウォールを乗っけよう？

SDNを使った方法

ルーティングに対応したSDNソフトウェアを使用し、NF間通信のサービスチェイニングで処理

例えば、インターネット公開サービスのセキュリティやロードバランサーを以下のようにSDNでNF(Network Function) をチェイニングして処理します。



機能をPod単位に細分化

SDNでユーザートラフィックを各Podにルーティングすることでサービスチェイニングを実現

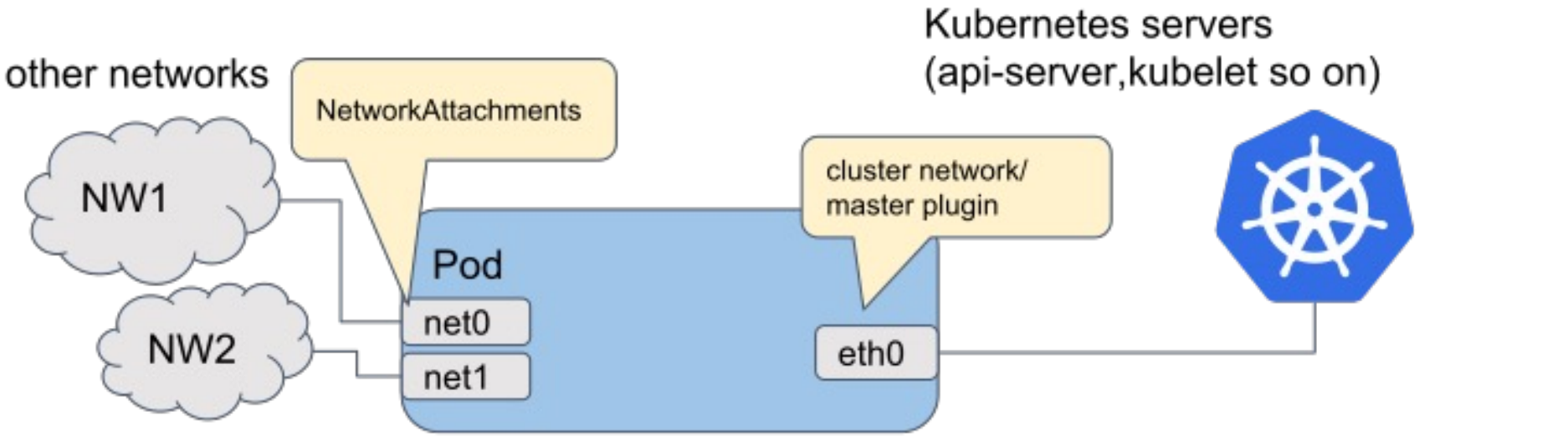
新機能はPodを追加し、SDN側のルールを変更することで対応

F5 Distributed Cloud ServicesのPod例

```
NAME                                READY   STATUS      RESTARTS          AGE
argo-kqdz6                          2/2     Running     0                 4h23m
aspen-lsr67                         2/2     Running     0                 4d
bdbewaf-9z5ft                       2/2     Running     0                 4d
dnsmasq-748d9f659-rtdfn             2/2     Running     10 (3d22h ago)    4d
envoy-2n75l                         2/2     Running     0                 4d
etcd-0                              2/2     Running     0                 4d
etcd-defrag-28106160-hgk2s          0/2     Completed   0                 3d2h
frr-gdl8l                           3/3     Running     0                 4d
ganges-9fccc986d-ztzdb              2/2     Running     3 (3d20h ago)     6d1h
gubernator-648b9f559-b8qzl          2/2     Running     4 (3d20h ago)     6d1h
ike-ddd8p                           2/2     Running     0                 4d
keepalived-n8vxh                    2/2     Running     0                 6d1h
obelix-cztlm                        2/2     Running     0                 4d
openvpn-j77q4                       3/3     Running     0                 4d
opera-x9tfw                         2/2     Running     8 (4h22m ago)     4d
phobos1-8bdccbcbb-bscpc             2/2     Running     1 (3d20h ago)     4d
piku-7b96df9b48-tlm56               2/2     Running     0                 4d
pmtud-8dfwx                         2/2     Running     0                 4d
ver-59f5b67dc7-db29n                2/2     Running     2 (3d23h ago)     4d
ver-6c8j8                           2/2     Running     3 (3d22h ago)     4d
webroot-rfm6g                       2/2     Running     0                 4d
```

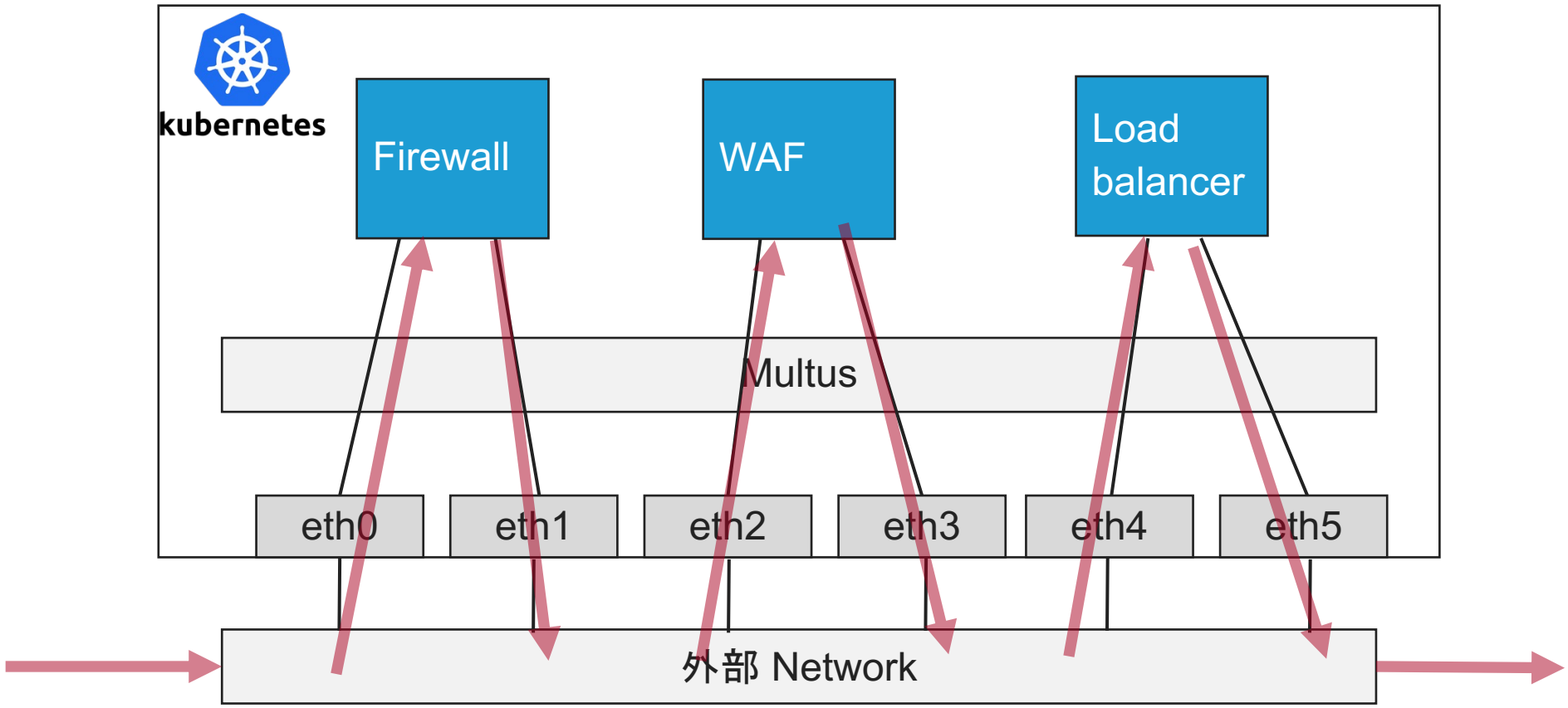
Multusを使った方法

Multus CNI (https://github.com/k8snetworkplumbingwg/multus-cni) を使用するとPodに複数仮想ポートを設定することができます。この追加ポートはKubenretes Serviceなどの処理はされず、仮想マシンのように外部ネットワークと通信が可能です。



Multusを使った方法

Multusを使った場合、Podは複数NICを接続できますが、Multusインターフェイスを使ったPod間通信は外部ネットワークに依存します。この構成の場合、仮想マシンと(大体)同じようにNFを扱えるようになります。



Multusを使った方法

ほぼVMと同じ用にインターフェイスが設定される。

インターフェイスごとに異なる種類(SR-IOVなど)の接続できる。

Eth0のPod networkは必ず接続

デフォルトでは、デフォルトルートがPod network を向いているので注意

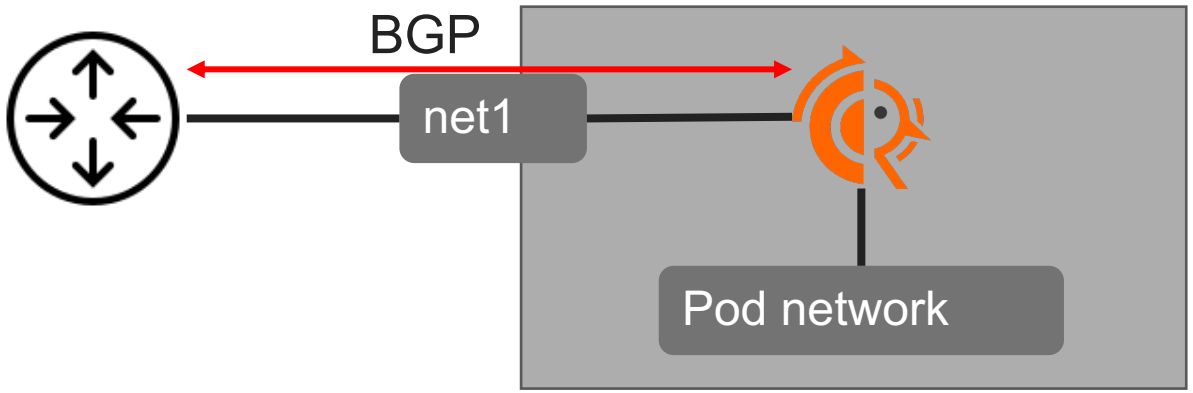
```
[root@janog-multus /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0@if68: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1436 qdisc noqueue state UP group default
    link/ether 1e:ac:3a:83:7f:ac brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 100.127.0.25/18 brd 100.127.63.255 scope global eth0
       valid_lft forever preferred_lft forever
3: net1@if69: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether c2:40:4d:0a:da:c2 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.201.224/24 brd 192.168.201.255 scope global net1
       valid_lft forever preferred_lft forever
4: net2@if70: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether ea:29:42:3a:3a:fd brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.202.224/24 brd 192.168.202.255 scope global net2
       valid_lft forever preferred_lft forever
```

```
[root@janog-multus /]# ip route
default via 100.127.0.1 dev eth0
100.127.0.0/18 dev eth0 proto kernel scope link src 100.127.0.25
192.168.201.0/24 dev net1 proto kernel scope link src 192.168.201.224
192.168.202.0/24 dev net2 proto kernel scope link src 192.168.202.224
```

	SDN	Multus
Pros ✓	スケールアウト 俊敏性⇧ 柔軟性⇧	既存VNFとの親和性 ハードウェアオフロード
Cons ✗	パフォーマンス SDN開発 監視/運用	俊敏性⇓ 柔軟性⇓

Demo

Demo



- FRRoutingコンテナをKubernetesで動作
- FRRoutingのコンフィグはConfigMapで管理
- FRRoutingコンテナの起動
- FRRoutingコンテナのバージョンアップ

まとめ

- Kubernetes Manifestを活用するとクラウドネイティブな管理ができそう
- ネットワークの仕様には注意
  - SDN対応のCNIの利用
  - MultusのようなCNIで複数ポートを利用
- Manifestの変更を適用するにはPodのリスタートが伴う場合がある
  - In Service Software Upgradeみたいなのは難しそう

議論のポイント

- 既存のオペレーションからCNFに移行するモチベーション
  - 使えそうなユースケース

- CNF使っている方上手く使えていますか？
  - 上手く使えるケース使えないケース
  - 合わなそうだなというケース

- CNFに期待すること
  - Container型アプライアンスなど