JANOG49 – Day3 SAIGOU 16:45-17:45

KDDI固定電話ネットワーク、NFV化の5年間の道のり

KDDI株式会社 次世代自動化開発本部　コアネットワーク部

2022/1/28

Tomorrow, Together

KDDI

自己紹介（辻）

次世代自動化開発本部
コアネットワーク部
辻　広志

VERSION="89.05.19”; RELEASE=”32"

仕事内容：アーキテクト的ななにか

- NFVなどの仮想化まわり
- 運用自動化

クラフトビールが燃料の関西人のNFV芸人です。

自己紹介（木場）

次世代自動化開発本部
コアネットワーク部

木場　仁美

仕事内容：

- VNFM/VNFDの開発
- 運用自動化

大学時代は騎射場でよく飲んでました！

Notices and Disclaimers

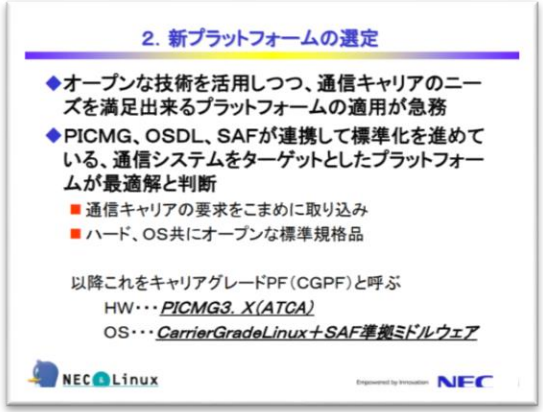
- 本発表はKDDI従業員個人の見解であり会社としての公式見解ではありません
- 特に断りが無い限り一人称は発表者個人、もしくは所属部署であるコアネットワーク部としての視点になります
- KDDI社内のネットワーク仮想化状況がすべて
  本発表の通りとなっているわけではありません
- 発表のために多少正確性に欠ける表現を用いている場合がございます

アブストラクト&イントロダクション

本発表のモチベーション

- 新しい技術が流行っている中では紹介話や導入談は多い
- その後実際どうなったの？っていう振返り話は少ない
  - aTCA / Carrier Grade Linux（1つ前のパラダイム）
  - いまは「クラウドネイティブ」がバズワード（次のパラダイム）
- 次に行く前に一度立ち止まって「振返り」を残したい

https://xtech.nikkei.com/it/article/COLUMN/20061205/255975/



http://lc.linux.or.jp/lc2003/slide/KN-01.pdf

本発表の骨子

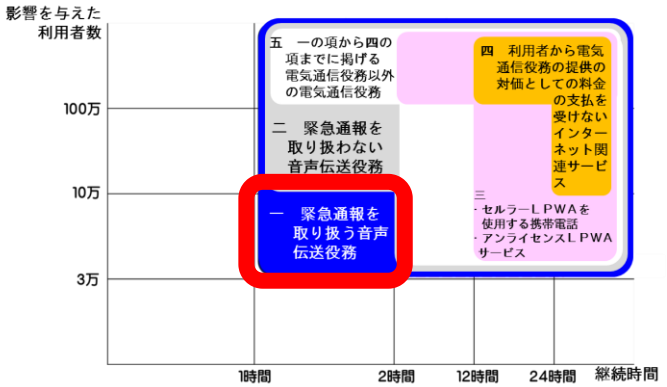
- KDDI固定電話ネットワーク更改プロジェクトを
  2016年の11月頃から2021年の10月頃まで実施
- 更改システムでは2016年当時のパラダイムであったNFV
  （ネットワーク仮想化）を採用
- うまくいったこと、うまくいかなかったことを
  振返り、「NFV」とは何だったのかを考える

KDDI固定電話ネットワーク更改プロジェクトとは

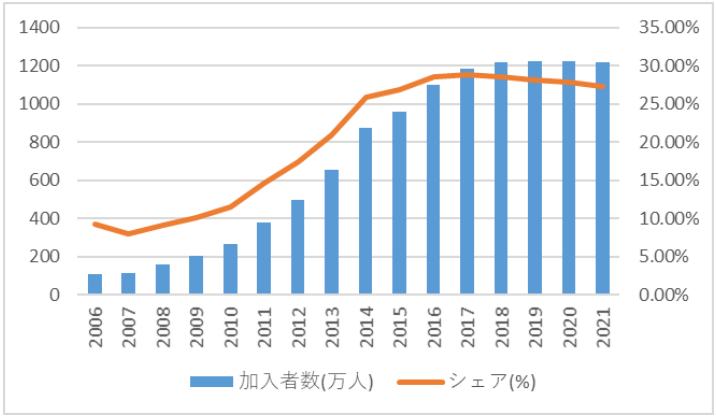
- **固定系IMS：0AB-J IP電話を提供する設備**
  - 加入者数: 1200万+
  - IP電話提供の「単独」事業者として国内最大
- **aTCA設備(専用HW)の保守限界(2021年)を契機に設備更改**
  - 2016年からNFVでの検討を開始し2021年10月完了
- **2004年のサービス開始から2度目の大規模設備更改**
  - 前回は 2011年にUNIX -> aTCA/CGLinuxかつIMS化

KDDIでも「ヤバい」設備の一つ

- 0AB-Jの電話は総務省が定める中でも最も重要な通信サービス(伝送役務)
- 緊急通報（110/118/119）を取り扱う



- KDDIの0AB-J IP電話の歴史
  - VoIP興隆期から取組み
    - 2003/10 ~
  - 加入者1200万程度(意外と多い)
  - J.D. パワー「2021年法人向けIP電話・直収電話サービス顧客満足度調査 (SM)」において9年連続第1位を受賞
    https://news.kddi.com/kddi/corporate/newsrelease/2021/09/03/5370.html



ネットワーク仮想化（NFV）の採用

NFV(Network Function Virtualization)

- ネットワーク仮想化ともいう
- 専用ハードウェアで動いていたネットワーク機器を
  仮想サーバ（基本的にamd64）で動かす取組み

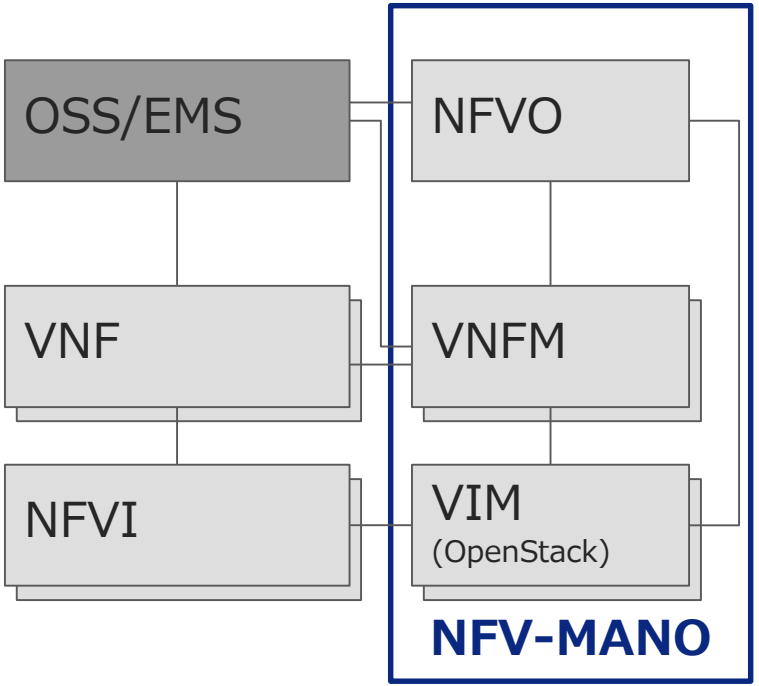
広い意味でのNFV　←LINEさん等の発表はこちら

- NW機能（RT,FWやLB、NAT等々）を仮想化しちゃうこと

狭い意味でのNFV　←今日の話はこちら

- キャリアのコアネットワークやRANなどを仮想化すること

NFV概説（用語の説明）



- **NFVO (NFV-Orchestrator)**
  VNFMを制御しVNFMで制御しきれない部分の面倒を見る
- **VNFM(VNF Manager)**
  IaaSにVMやNWを作り、出来上がったVMを使える状態まで持って行く機能
- **VIM(Virtual Infrastructure Manager)**
  IaaS、AWS EC2みたいなもの。
  OpenStackを利用することが多い
- **VNF(Virtual Network Function)**
  要するにアプリケーション、仮想化されたNW機器
- **NFVI(NFV Infrastructure)**
  要するにハイパーバイザやNWやストレージ

プロジェクトを振り返って（総括）

- 仮想化としては失敗（失敗ではない）
- 自動化（というか運用の変革）としては成功
- マイグレとしては（トラブルは多少あったものの）成功
- 組織としては大きく成長

本発表では失敗事例も織り交ぜつつ、上記4点を振返り NFVの実際を紹介します

仮想化としては失敗（失敗ではない）

なにが失敗だったのか

特殊要件の染み出し

- レガシなVNFに起因する特殊要件を基盤で吸収
- OpenStackの仮想基盤が特殊化

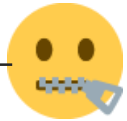
専用基盤化

- 結果としてシングルベンダの専用基盤になった
- 他のベンダのVNFの搭載が難しい

課題の先送り

- NFVの共通基盤化をするなら将来やることに違いはないため実質課題の先送りをした

※失敗とはいっても仮想化しない方が良かったという意味ではないので「KDDI仮想化に失敗」とだけは吹聴しないようにお願いします！



そもそもNFVでの更改を検討した時代背景

OpenStack/NFVは絶頂期。 hype cycleの頂点。

- ETSI-NFV標準化が形になり始めたのが2014年-2015年
- 2015年 OpenStack Summit Tokyo
- NTTドコモ　2014年 5月にvEPCを検証成功を発表
- NTTドコモ　2016年 3月にvEPCを商用導入

社内でも「NFV」や「MANO」、「オーケストレータ」という単語がバズっていた

いずれにしても我々はNFVがよくわからなかった

なにかできそう、でも実際NFVってどうなの？

- OpenStackでNFVって実際どんなものなのか？
- 性能は問題ないか？
- MANOってなに？なにができそうなのか？
- MANO＝自動化？

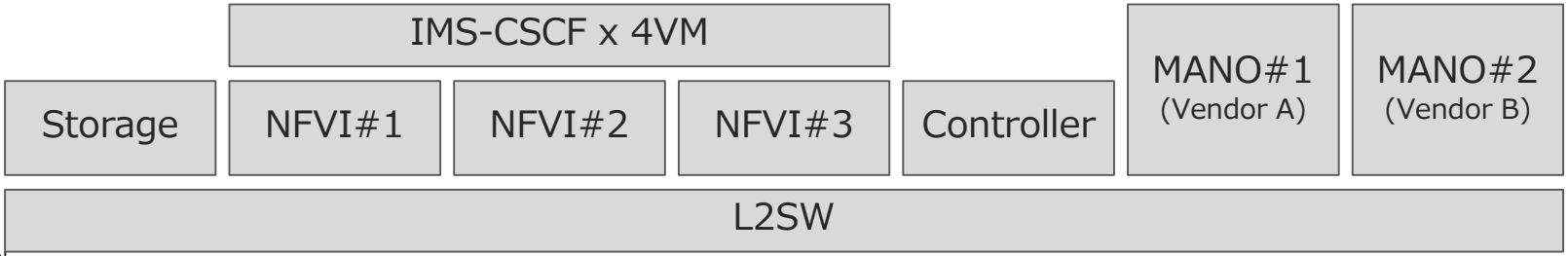
NFVの二大デプロイ方法

- マルチベンダNFV　→　大変そう
- シングルベンダNFV　→　こちらが良い？

2016年 PoCの実施

2016年11月~2017年3月

- お金を掛けた、ちゃんとした検証をした
- OpenStack Liberty(RHOSP 8)
- 3台のコンピュートサーバ（スモールだがマルチノードを触れたのは初）
- 単純ポーティングした既存IMSアプリ(CSCF系)の動作を確認
- 二つのNFV-MANOソリューションを比較
  - 一つはVNF同一ベンダ、もう一つは別ベンダ（マルチベンダ構成）



2016年のPoCでの確認内容と結果

- **既存のアプリをポーティングしてVM化**
  - VNFに合わせたため少々いびつなOpenStack構成/仕様
- **アプリケーションの機能試験・性能試験**
  - 懸念されていた性能低下もなく問題なし
- **NFV-MANO経由でできる自動化の検証**
  - NFV-MANOで、オートヒールぐらいしかできないことを認識
- **複数ベンダのインテグレーション**
  - 問題なし、MANOベンダごとに実装差異が結構あり
- **その他更改に向けた課題の抽出**

PoCから見えてきたNFVの実態

「NFVとはVNFのことである」 談：つじ

- VNFの出来の良し悪しがNFVの良し悪しを左右する
  - **「クラウドネイティブ」でないアプリケーション**
    - PoCを行ったようなレガシーなアプリケーション
  - **基盤と密結合なVNF**
    - サポートするNIC/Storage/サーバetcを制限してるアプリ

VNFの要件を基盤で吸収することとになる
1つのベンダのVNFであれば良いが複数ベンダになると…？

基盤に合わせてVNFの改修を行いたい

NFV化をする目的をどこに置くのか

我々がNFVに求めた価値

HWの抽象化

- サーバ/NW/ストレージ仮想化とはHWを抽象化すること
- 抽象化レイヤを間に入れる事により移行性が増す
- ハードウェアのライフサイクルと、アプリケーションのライフサイクルを分離して考えられる

集約化

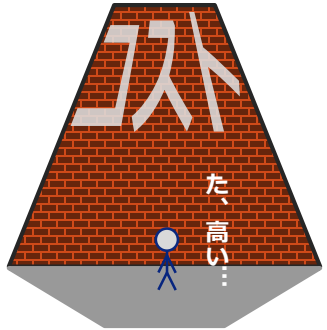
- 場所や電気の利用効率を上げられる
- 共通機能をサービス間で個別開発・構築しなくて済む

運用効率化

- コードによる自動化がしやすい
- NFV-MANOは標準化による相互接続やベストプラクティスの提供とOpenStackに足りない部分の機能補助

いざ始めようとしても…

立ちはだかるコストの壁



- マイグレーションであること
- 元がレガシーなアプリケーションであること
- 重要設備であること

「NFVとはVNFのことである」

VNFの要件がきついことに起因する課題

立ちはだかるコストの壁（1/2）

「専用ハードウェアから脱却し、大幅にコストを削減」

と言いたいところですが、やはりそう単純ではない

不都合な真実

- ハードウェアはそんなに高くない、ソフトウェアが高い
- コア系設備はもともと大部分が「ソフトウェア化」されている
- ソフトウェアの改修は高コスト、仮想化対応だけでもそれなりに
- 仮想化レイヤ(OpenStackやMANO)のオーバヘッドは大きい
- シングルベンダ構成は競争効果が働かない

立ちはだかるコストの壁（2/2）

我々がプロジェクトで重視したポイント

最重要

1. **確実性/安定性/安全性**
2. **コスト**
3. **自動化/運用効率化**

- 非常に重要なシステム
- サービス採算性は重視
- NFV化の目玉機能

2017年に我々がとった選択（バランスを調整して諦めたこと）

- 確実性重視でシングルベンダ構成
- 確実性とコスト低減のために大幅なアプリ改修を避ける
- 構築・復旧自動化、運用自動化にはしっかりコストを掛ける

結果として仮想化基盤は専用システム化してしまうが
運用機能の改善は必ずやり遂げるという意思でプロジェクトを始めた

当時の選択を今改めて振返って

ベストな選択だったと思うか？ → No

ベストな選択をとれたと思うか？ → No

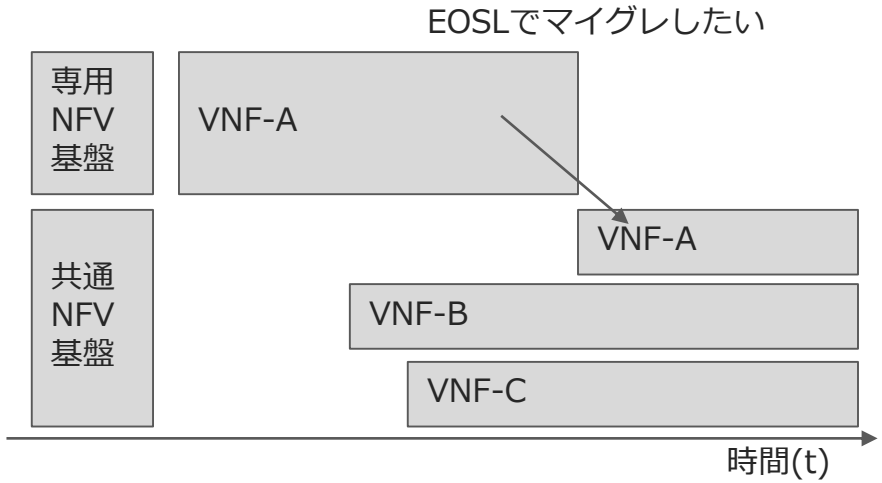
当時の状況ではコストを削減できるやり方の提案も
マルチベンダで行くための根拠を示せなかった

- 対象サービスがミッションクリティカル
- 事例不足
- 実力不足

結果的に残ってしまった課題（失敗ポイント振返り）

もともと描いていた構想

- 導入した基盤を拡張してNFV共通基盤に　→　×
- 後から構築する共通基盤にゆくゆくはマイグレ　→　？



- VNF要件を専用NFV基盤で吸収したVNFは基盤要件をVNFで吸収する共通NFV基盤に載せることは困難
- というか結局それは今回諦めたポイント

かといって専用基盤を残し続けるのもかなりの負担

実質問題を先送りしただけ

（将来の技術が解決してくれる側面は多少ありますが…）

自動化（というか運用の変革）としては成功

どう成功なのか

やりたかったことが実現できた

- 課題を明確にし、解決した
- 自分たちで作り上げた
- クラウドっぽいやり方にできた

拡張性

- 将来の課題に対して備える体制と技術

運用担当時代の閉塞感

もっといいやり方があると思いつつ変えられない

忙しい・・・

自動化をもっと進めたいけど難しい

もっとどうにかできるはずなのに…

TeraTermマクロを作ったら喜んでもらえるが
コレジャナイ感がどこかある

当時の運用における課題

- ハザーダス時間の長期化、深夜駆け付け対応
- 増設が頻繁にあってその準備や作業が大変
- アプリの設定情報の管理（ルーティング情報など）
- 統計情報の可視化や利用
- サービス監視
- 踏台…telnet…踏台…telnet…

※ハザーダス：片系運用中を表すKDDI用語

コロンブスの卵

NFV化で運用改善の検討を開発担当とやることに
(その後、運用と開発の部門統合)

設備側を変えればいい

運用からすると設備を変えて自動化や
効率化を進めるという発想が当時なかった

NFVでは自動化もセットで考えられてて
モチベーションをもって変えていける気がした

NFVでできる自動化とできない自動化

PoCでNFV-MANOの役割を確認

できること

- OpenStackの制御
  （仮想マシン作ったり消したり）
- VNFの初期設定を少しだけ
  （決められスクリプトを初期流せる）

できないこと(できてほしかったこと)

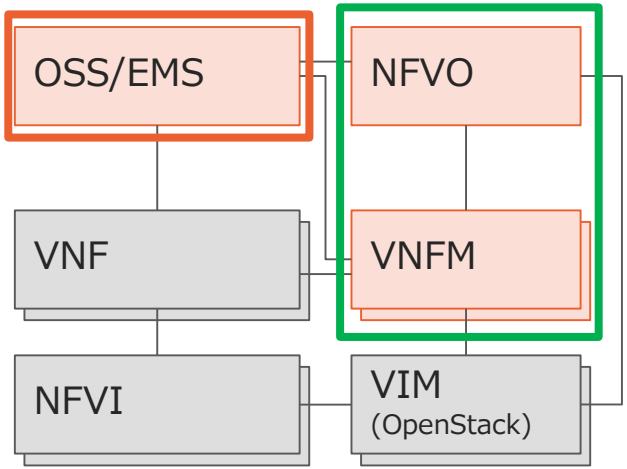
- VNFの設定変更
  （Config修正したり）
- ルールベースの自動化
  （やってやれないことはないが…）

NFV-MANOの範囲で実現できない自動化を認識

(NFV化で全部が解決できるわけではないっぽい・・・)

OSS/EMSでやることNFV-MANOでやること

課題をどこで解決できるのかを考えた



MANOで解決できるもの

- ハザーダス時間の長期化、深夜駆け付け対応
- 増設が頻繁にあってその準備や作業が大変

OSS/EMSで解決できるもの

- アプリの設定情報の管理（ルーティング情報など）
- 統計情報の可視化や利用
- サービス監視

VNFの作りで何とかするもの

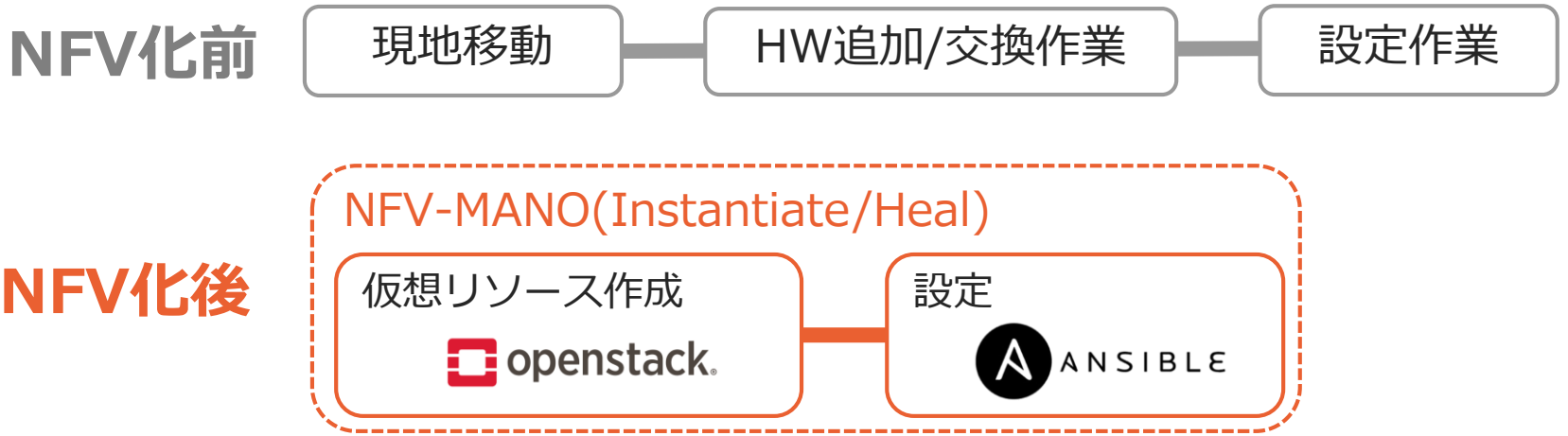
- 踏台…telnet…踏台…telnet…

MANOはVNF立ち上げとヒーリング、それ以外はOSS/EMS

※ハザーダス：片系運用中を表すKDDI用語

MANOで解決できるもの

- ハザーダス時間の長期化、深夜駆け付け対応
- 増設が頻繁にあってその準備や作業が大変

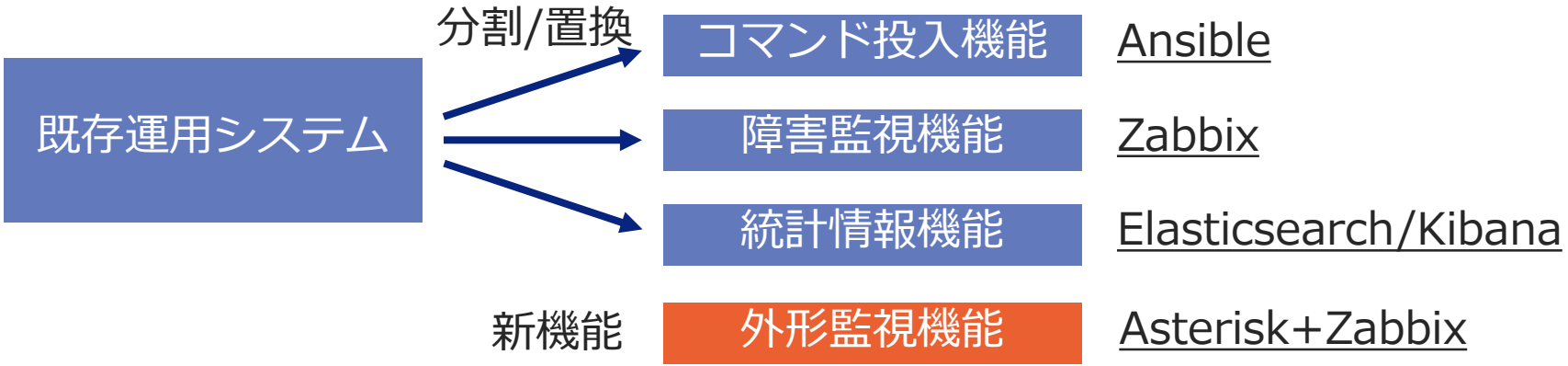


復旧はゼロタッチ化、構築はワンタッチ化
にこだわってベンダと開発した

※ハザーダス：片系運用中を表すKDDI用語

OSS/EMSで解決できるもの

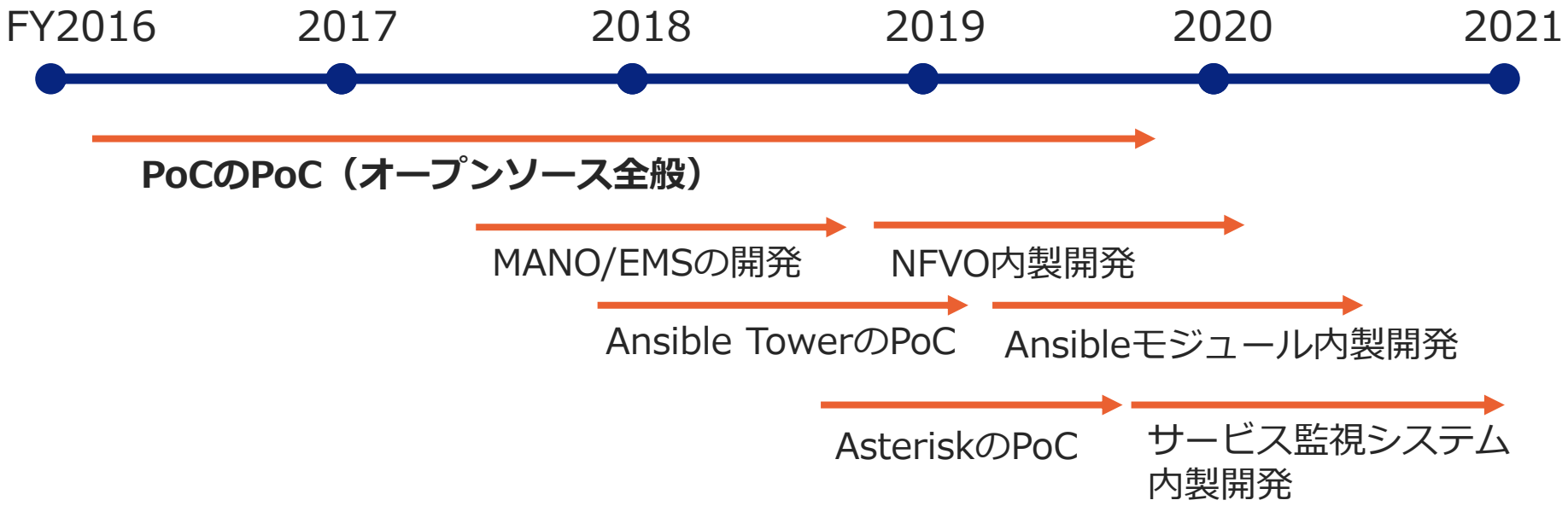
- アプリの設定情報の管理（ルーティング情報など）
- 統計情報の可視化や利用
- サービス監視



既存システムでできることベースに機能配置を見直し
オープンソースベースでの実装を検討した

時間を有効に使えた

NFV検討開始からマイグレ直前までPoCと開発を実施した



課題が本質的に改善できるかをしっかり見極めて導入できた

オープンソース利用の見極め

とにかく手元で試してから本格検討するようにした

PoCのPoC
- とりあえずできそうか？
- どんなもの？

↓

PoC
- 実運用に近い形でがっつりお試し

↓ お金はここから使う
- 商用ライセンス
- コンサル

↓

実開発
- 実際に作って導入

動くものを見せると正式な仕事となり、取り組みやすくなった

NFV化がもたらした”自動化”の変革

Infrastructure as Code 的な運用の浸透

- NFV化により、VNF情報がすべてテキストとなり、Gitで管理するようになった
- “Gitの情報が神様”であるという軸ができ、それを参照して他のツールも作るようになった

属人化による散乱した自動化からの脱却

- Ansible Towerの導入で、ツールを実行する場所/言語/保管場所などの最低限のルールが確立された
- ルールがあるからこそ個々のレベルに応じた自動化ができるようになった

まとめと所感

- 運用で困っている立場の人が時間をかけて取り組んだ
- 自動化は人や時間を正式にアサインしづらいが、動くものを見せることで変わった
- オープンソースを活用して拡張性とアジリティの両方を向上できた
- NFV化により自動化に用いるデータがテキストベースとなった

・・・ここから所感・・・

- 自分が運用担当の時と比べると、夢のようなものに変わったと思う
- 運用の課題は、運用しないとわからないから
  完璧を目指すより「改善しやすい仕組み」「管理しやすいルール」
  「運用者が改善にガッツリ取り組める雰囲気」が必要だと感じた

マイグレとしては
（トラブルは多少あったものの）成功

なにが成功だったのか

自動化の効果による作業の稼働低減

- 作業の内製化
- NFVの効果で低稼働でVNF構築を実現
- パラメータを作成、あとは待つだけ

作業完遂

- 総務省報告物の事故等もなく安全に切替が完了
- 仮想NW特有の問題も発生（本日ご紹介）

自動化による稼働低減

Ansible化＋IaC化で構築稼働を大幅短縮

- 6つのVNF種別
  （HSS/SLF/etc…）
- 合計約400VM
- MW、アプリの設定まで自動化

作業体制、社員3名程度での内製化

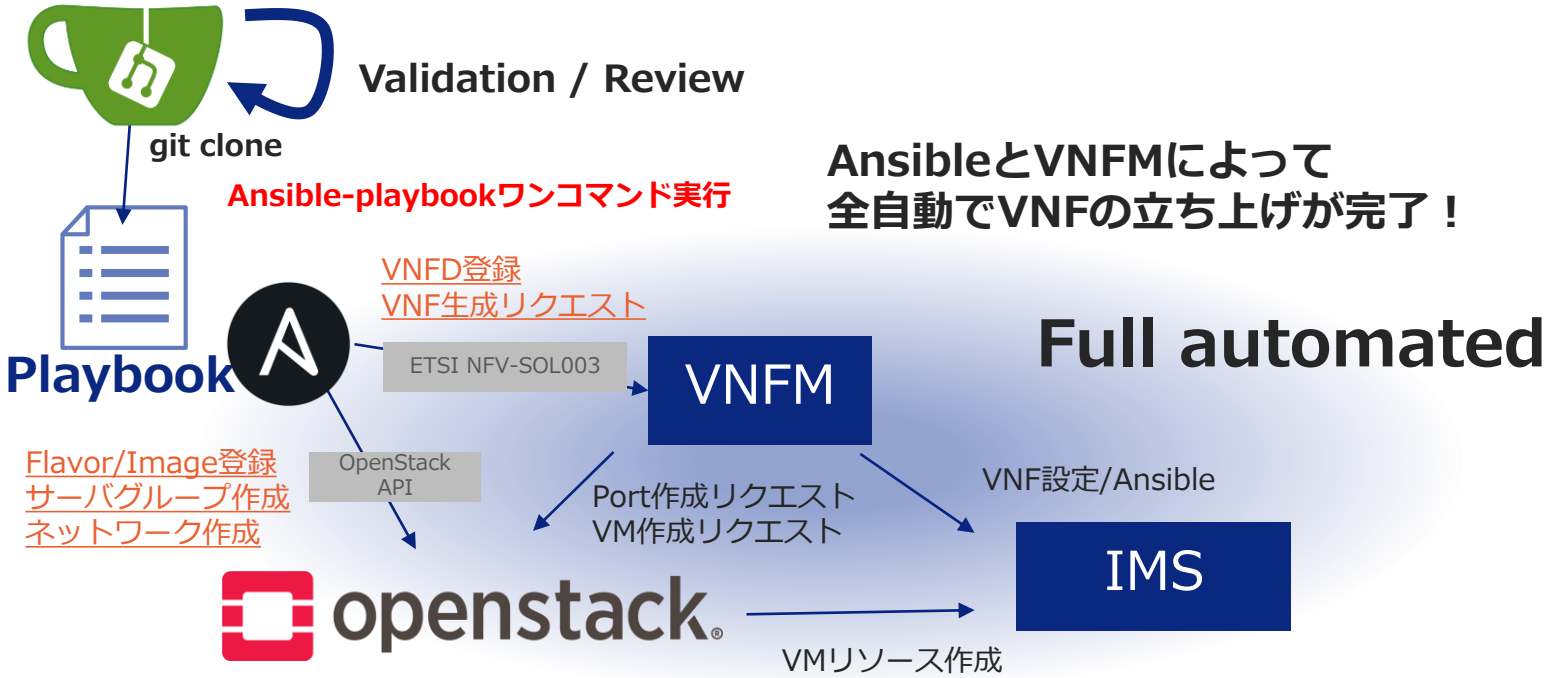
- Ansible Playbook化することで作業が単純化し体制も組みやすくなる

いわゆる「現調(現地調整作業)」まで一気に完了

今までハードウェア追加作業、ベンダ設計、作業等、
3カ月以上かかっていた作業が、一週間程度で完了

自動化による稼働低減

VNF（VM/Network）の構築作業の自動化・内製化



自動化による稼働低減

Playbook/Roleを含んだ
テンプレートリポジトリを作成

roles	複数のVNFMへのAPI
ns_update.yml	複数のVNFMへのAPI
onboarding.yml	Initial commit

MANOチーム

Fork

環境に合わせて
host_vars/group_vasrsを追加し
Playbookを実行

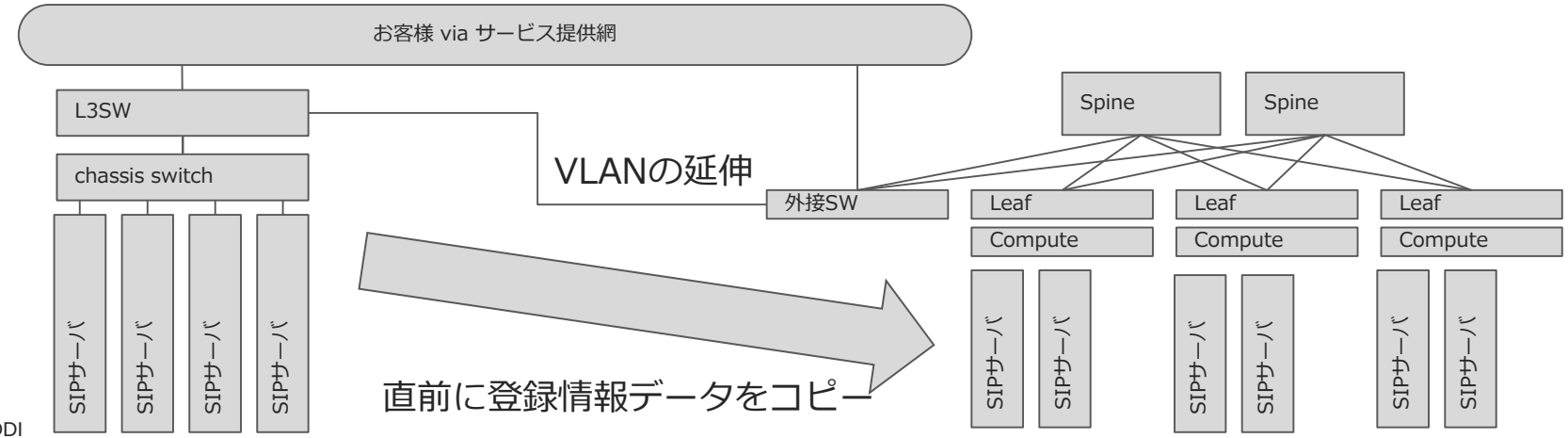
group_vars	商用パラメータ設定 (#8)
host_vars	20210301_VEGA_C5

VNFチーム

マイグレ作業：作業内容

しびれる同番IPアドレスでの移行

- どうしてもSIPサーバだけはIPアドレスの同番移行が必要
- 直前に登録情報（REGISTER信号情報）のコピー
- 緊急呼通信がないことを確認して旧サーバのポート閉塞
- 新サーバの閉塞解除、GARP送信



だいたいはうまくいきました…

NFVならではの障害事例の共有

- SDN/データセンターNWソリューション特有の問題発生
- 原因不明の切戻し

基本的には一般に公開していない「お客様にご迷惑をおかけした障害事例」となりますので次ページからしばらく オフレコ、おねがいします。

内容自体は大したことないためそっと心の中に‥。

事象1(多少脚色・単純化してます)

当日投影のみ

事象２(多少脚色・単純化してます)

当日投影のみ

原因

当日投影のみ

根本原因の調査

当日投影のみ

説明しよう！①

当日投影のみ

説明しよう！②

当日投影のみ

障害っていつもだいたいこういう感じ

当日投影のみ

教訓

- 「対策」が新たな事故を引き起こす可能性もあること
- 「確認系コマンド」だと思っていた traceroute で障害
  - 危険性を認識していなかった
  - 事前準備、ダブルチェック、ツール化できっちり準備を。
- ソースIPアドレス指定は気を付けよう
  - ネットワークコマンドは大なり小なり
    スイッチ側にサイドエフェクトを及ぼすと認識
- SDN系ソリューションは思わぬ落とし穴もある

やっと完遂

2021年 9月(2020/10~)
仮想化システムに移行完了！

大規模でかつ歴史ある設備の切替はやはり難しい

- 作業回数が増えると人がミスをする可能性も増える
- やっぱり「地雷」が埋まっている
- 周辺部署、チームとのミスコミュニケーションが起こるetc…

安全に作業をするために心がけたこと

- 小規模に行い、問題が出ても影響範囲を少なくすること
- サービス復旧優先
- 躊躇なく戻すこと（とはいえ難しい、うまくできたことできなかったことはあり）

組織としては大きく成長

組織として成長した

DevOps

- 運用と一緒になったことで効率化
- オーナシップが生まれる

スキルの向上

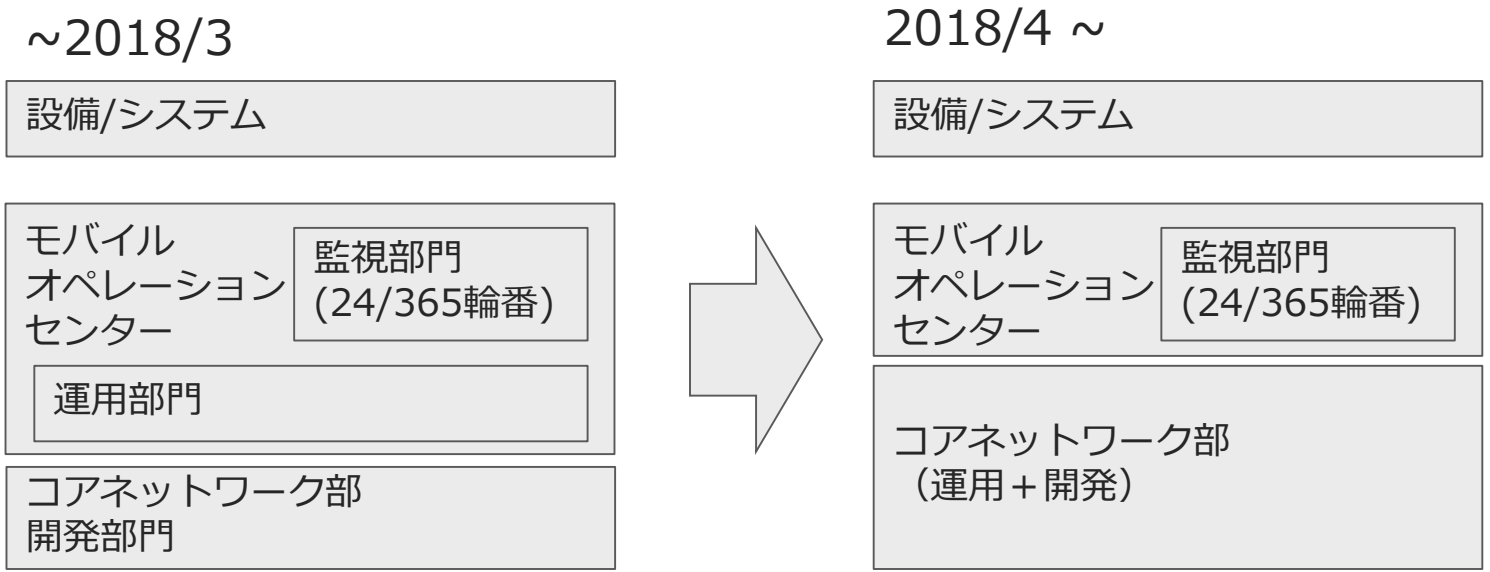
- NFVではOSSがコア技術となる
- 調べてわかる、ベンダと対等に話せる

マインドの変化

- 自分たちでやることが近道であると気づく

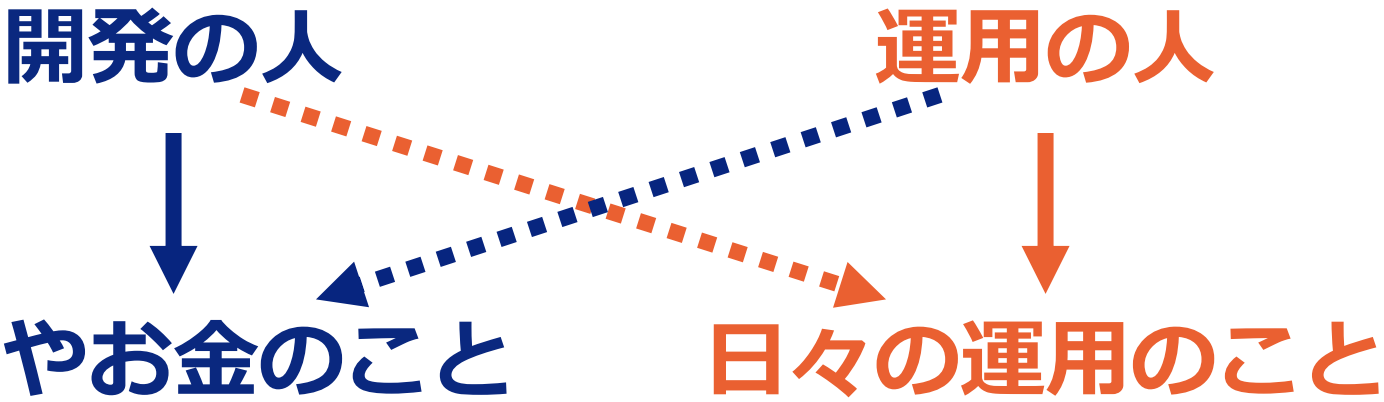
DevOps体制への移行

この5年の中で大きく体制が変わった



組織統合はうまくいった

システムやサービスに対するオーナシップが生まれる

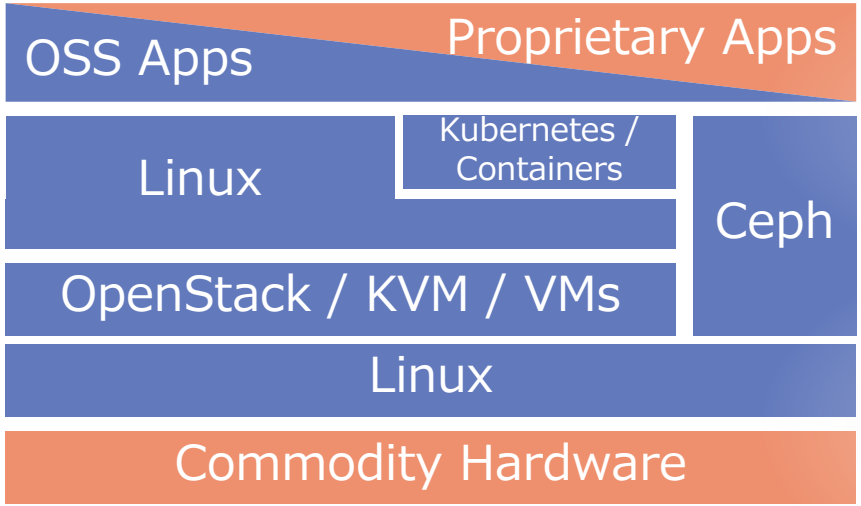


- 運用フェーズに入ってからも開発したナレッジがそのまま生かせる
- 開発の早い段階で運用を意識できる

▸ **我々がNFVで目指した運用効率化の加速材料**

スキルの向上

NFVではOpen source softwareが技術の中心になる



基盤技術の中心にいるのはLinuxであり、
またOpenStackをはじめとするOSS。
OSSは平等に情報にアクセスができる。

システムをインテグレートする
SIerやNEP固有の技術は薄くなり
オペレータ側と技術的知識が
大差が無くなってくる

スキル向上が加速し自分たちでできる範囲が広がる

スキルの向上はマインドの変化ももたらす

SIer/NEP 経由でうまくいかないことが出てくる

技術力の問題

- 旧来のSIモデルが急速な技術変化に追従できていない
- OSSへの理解/仮想化の幅広い領域を求められる高度な専門性

モチベーションの問題 (こっちらの問題の方が大きい)

- 運用を知っているのはオペレータである自分たち
- 仮想化/自動化は変化に対応するための技術
- 仮想化/自動化のモチベーションは圧倒的にオペレータの視点
- SIerに変化への対応や運用効率を上げるモチベーションはない

マインドの変化

結果生じること

- KDDIでコミュニティのナレッジからや独自調査で解決、誤りの指摘が増加
- OSSベンダのサポートへKDDIから直接問合わせで解決するケースが増加
- 自分たちで修正できるのにSIer等を間に挟まないといけないフラストレーション

- **自分たちでやるのが何よりも近道ってことに気づく**
- **スキルが向上したことにより見える範囲が広がった**

エピローグ

プロジェクトを振り返って（総括/再掲）

- 仮想化としては失敗（失敗ではない）
- 自動化（というか運用の変革）としては成功
- マイグレとしては（トラブルは多少あったものの）成功
- 組織としては大きく成長

最後に失敗だったと語った仮想化に対して、
その失敗をどう**乗り越え**ようとしているのかを紹介します。

臥薪嘗胆

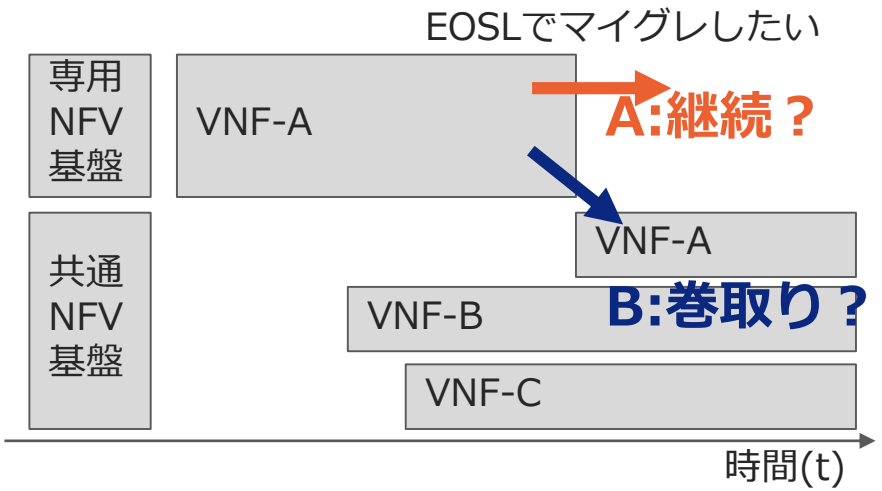
- **今回作った基盤で将来にわたる拡張性を見込むことが厳しい**
  - 早々に諦めて次を考え始める
- **2018年から次期基盤の検討を開始（並行開発）**
  - とあるVNFの構築に合わせて実施
  - より一般的なOpenStackの機能を使用
    - Cinder / Swift / NeutronのL3機能etc..
- **反省を生かして共通NFV基盤として企画**
  - 共通NFV基盤には明確にポリシーを定める
  - 要件が合わない部分はVNF側が合わせる

先行してしまった基盤とVNFの将来

いわゆる技術的負債。どう返済していくかを考える。

A）専用基盤を継続してメンテしていく

B）専用基盤を積極的に共通基盤に巻き取る



Aの場合

- 二重管理の直接的・間接的コスト

Bの場合

- 特殊要件をどうするか？
- 共通基盤のポリシーに反する部分をどう解決するか？

積極的に巻取りをかけていく（B）で検討と一部マイグレを開始

共通基盤の難しさも当然ある

自動化やNFV関係の設計がうまくいかない

- アプリチームはNFVや自動化回りのスキルが弱め
- シングルベンダ構成の時はチーム間で面倒を見れていたが水平分業に近くなってくると見え辛い
- 結果的に自動化やNFV的な出来はシングル構成の方がうまくいっている側面がある

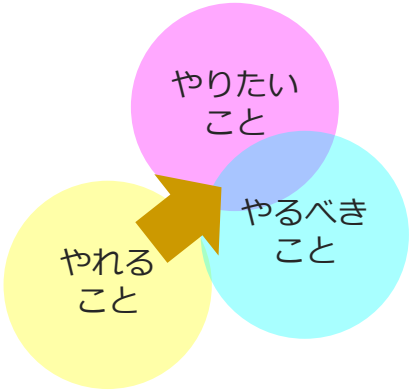
体制の問題もあるためやり方を模索中

- 社内SE的な立ち位置でVNFの検討体制でアドバイスする体制を作ってみたり・・・

NFVとは何だったのか

NFVをうまく活用するためには

- **HWの抽象化**
- **集約化**
- **運用効率化**



NEP / VNFベンダに依存するキャリアでは
「やれること」を変えるのが一番難しい

- 成功するには3つのファクタが交わる領域をいかに最大化できるか
  - 会社の事業課題を解決できること（やるべきこと）
  - やりたいことを明確に描けること（やりたいこと）
  - ベンダ/NEPを動かせること（やれること）
- 構図としては仮想化でもコンテナでも変わらないと思える

アプリケーションの進化がカギ

オペレータのやりたいこと

基盤技術の進化

Kubernetes
OpenStack
CGlinux/aTCA

NEPベンダのやりたいこと

アプリケーションの進化

3GPP etc

基盤技術側だけを考えてもだめ、アプリとどう歩調を合わせるか

- 前NFV時代は基盤技術もアプリの構成要素の一部でありNEP側が考えた
- NFVにより基盤技術はオペレータがコントロールしたい領域へと変化した
- 基盤とアプリの両輪での進化を目指す必要がある

NFVとは。

NFVとはオペレータの変革の道

- **NFVとはオペレータの課題を解決するもの**
  - HW抽象化、集約化、運用効率化
- **「NFV=VNF」。成否はアプリケーションの「クラウド化」**
  - 仮想化すること　≠　クラウド化
  - コンテナ化すること　≠　クラウド化
- **すでにNFVは冷静の中にあり**
  - 終わった技術ではなく、これからが正念場。

議論したいポイント等々…

- **ビジネスや実力を無視した、技術ドリブン過ぎる環境への対応の仕方**（vRAN / 5GCのCNF）
- **DCNW/SDN関係でのハマり話**（数日前にもLeafスイッチの故障で中途半端に疎通ができないことが…）
- **SIer/ベンダ/メーカ目線でのNFVてなにか変わりましたか**
- **既存システムでの自動化でうまくいった例とそのコツ。**（中々弊社ではやはりみられない。）
- **ソフトウェアエンジニアの育て方,人材育成**
- **水平分業のむつかしさ、イイアイデアありますか？**
- **OSSとの付き合い方**
  - 弊社はOSSセンタ的なものがないので、そういうところをお持ちの方からみた組織論なども気になります
- **SIer/ベンダ/メーカ目線からみてオペレータの自動化ってどうですか**
  - ビジネスチャンス？運用に直結していて入り込むのが難しい？


Tomorrow, Together
KDDI