NFV基盤で利用しているCephクラスタのバージョンアップを内製対応した話

KDDI株式会社 松本 良輔、齋藤 裕子

Tomorrow, Together
KDDI
おもしろいほうの未来へ。
au

本発表でお話しすること

- KDDIではOpenStackとCephを用いて、音声通信設備のためのNFV仮想化基盤を構築(※)しています。
- この仮想化基盤においてCephのバージョンアップを内製で実施しました。
- 本発表では、直面した課題をどう解決したか、効率的な手順検証とメンバーのスキルアップを両立させた点について紹介します。

【※過去発表】
CODT2020:そういやNFVってどうなったん？最近あまり聞かなかくなったNFVの話（KDDI編）
(https://cloudopsdays.com/archive/2020/program/)
CODT2021:NFVでクラウドネイティブに変わる電話会社の運用（KDDI編）
(https://cloudopsdays.com/archive/2021/sessions/)
JANOG49 :KDDI固定電話ネットワーク、NFV化の5年間の道のり
(https://www.janog.gr.jp/meeting/janog49/kddinfv/)
CODT2022: OpenStack NFV基盤のバージョンアップと運用改善を内製対応した話
(https://cloudopsdays.com/archive/2022/ondemand/)

発表者紹介

ネットワーク開発本部　キャリアクラウド開発部

松本　良輔

前半

ネットワーク開発本部　キャリアクラウド開発部

齋藤　裕子

後半

アジェンダ

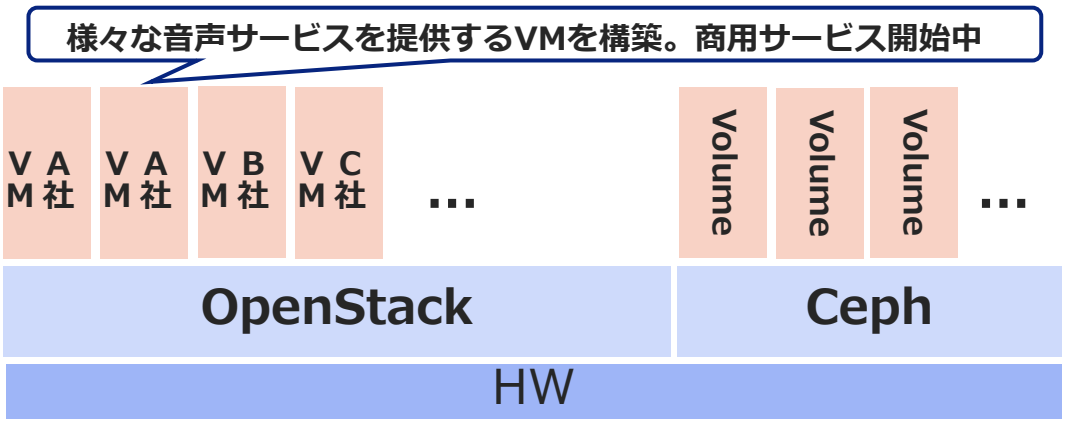
1. オンラインセッション部分のおさらい
2. 作業時間短縮のために実施した工夫
3. OA

1. NFV基盤について

NFV基盤について

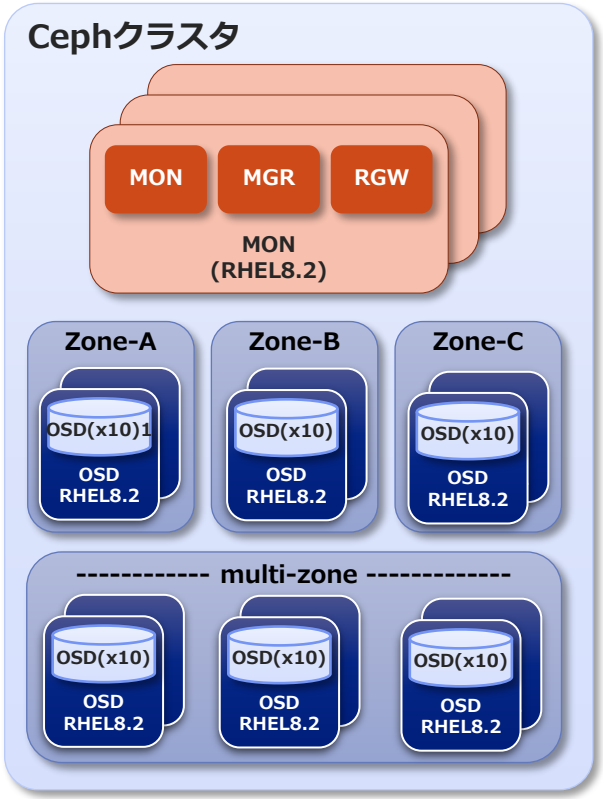
OSSを活用したベンダニュートラルな仮想化基盤を構築

OpenStack＋Cephで仮想化基盤を構築



2. Cephのバージョンアップ概要と直面した課題

当初思い描いていた手順　Cephバージョンアップ



各ノードのOSとコンテナを以下に

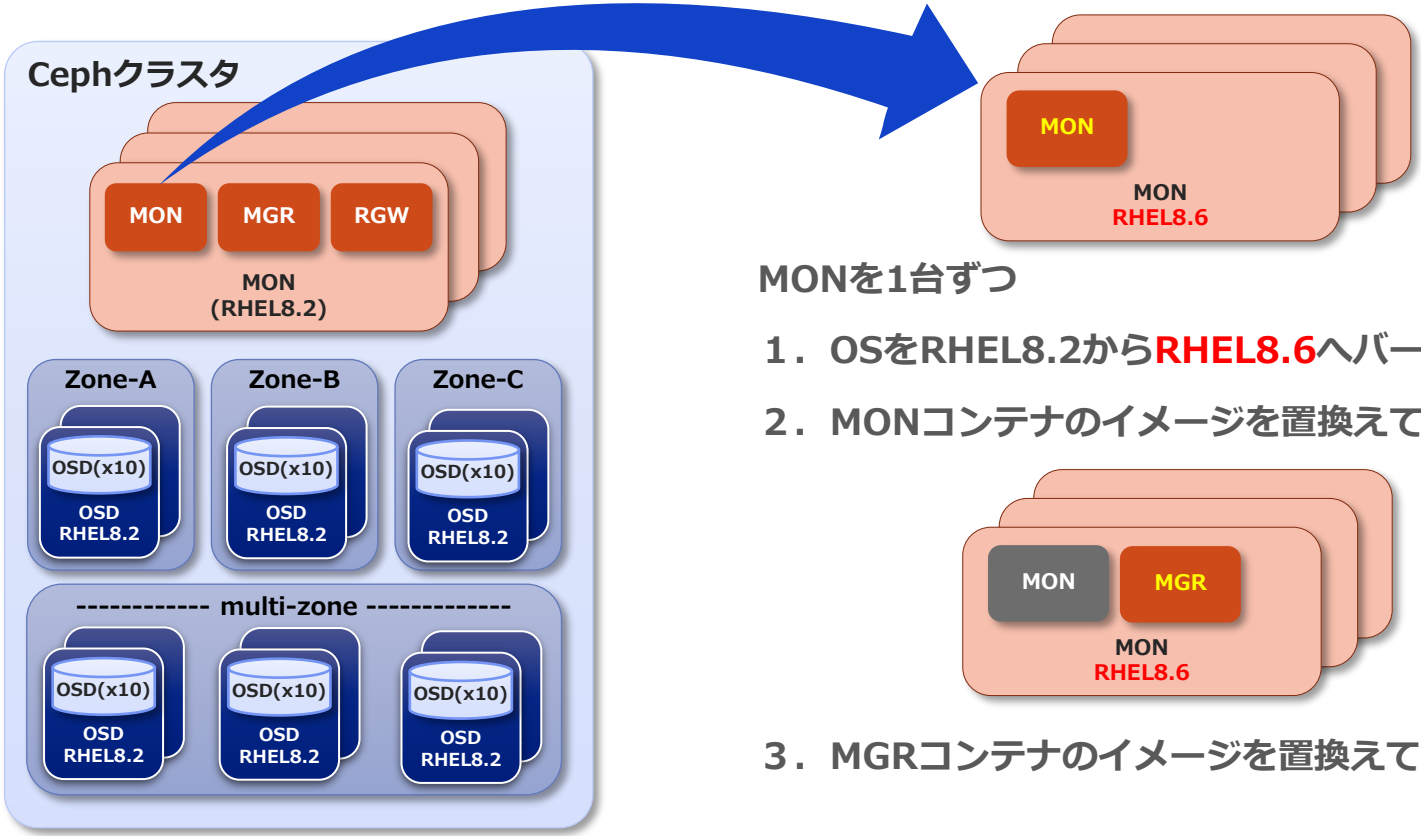
RHEL8.2 ⇒ RHEL8.6
Ceph(Nautilus)マイナーバージョンアップ

１台ずつクラスタから切り離しながら・・・

- OSはansible-playbookを使って1台ずつバージョンアップ
- コンテナはceph-ansible-playbookのrolling_updateを使い、Limitオプションを付けて１コンテナずつバージョンアップ

(スモールスタート)

当初思い描いていた手順　Cephバージョンアップ



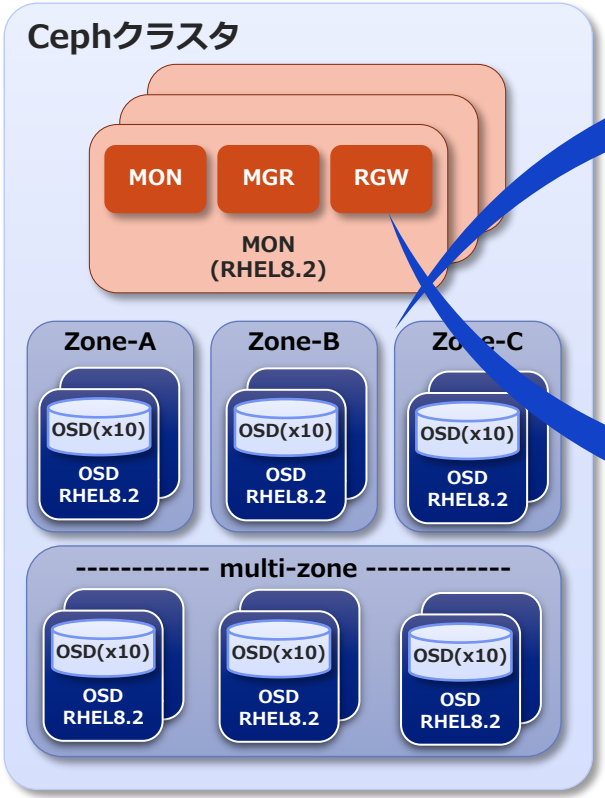
MONを1台ずつ

1．OSをRHEL8.2から**RHEL8.6**へバージョンアップ

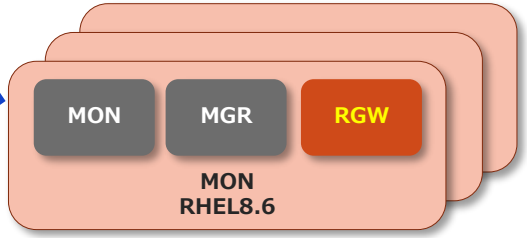
2．MONコンテナのイメージを置換えてバージョンアップ

3．MGRコンテナのイメージを置換えてバージョンアップ

当初思い描いていた手順　Cephバージョンアップ



OSDを1台ずつ
4．クラスタから切り離し
5．OSをRHEL8.2からRHEL8.6へバージョンアップ
6．OSDコンテナのイメージを置換えてバージョンアップ



6．RGWコンテナのイメージを置換えてバージョンアップ

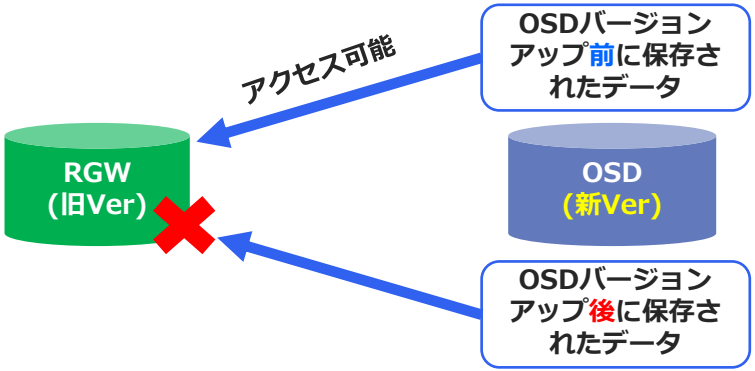
問題発生

前述の手順でやりたかったけれど・・・
とある問題のせいでできない。。。

バージョンアップする際の問題

今回対象バージョン間のアップグレードに際し一部機能(RGW)においてバージョンアップ最中の互換性が失われているような変更が入っていた

OSDを先にバージョンアップした場合

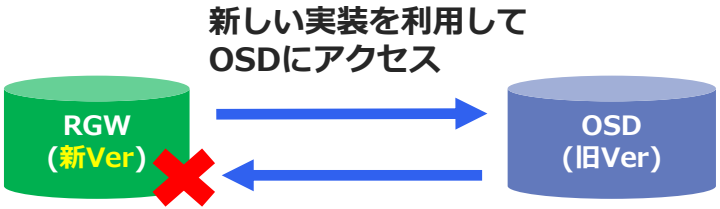


RGWが新しいコードを理解できず、バージョンアップ後に保存されたデータがクラッシュする可能性があった

バージョンアップする際の問題

では、RGWを先にバージョンアップしたら？

RGWを先にバージョンアップした場合



OSDはRGW側の新しい実装によるメッセージを理解できず
誤ったレスポンスを返す

バージョンアップする際の問題

どちらも選べない・・・

公式手順は前述の問題回避として、OSD/RGW間のバージョン差異期間を出来るだけ短くするため、**一気にバージョンアップする**ものとなっていた

しかし、当基盤上には総務省報告対象となるサービスが動いていたため、社内承認上、上記のやり方は選択できなかった（原則スモールスタート）

Step by stepの手順を確立する必要があった

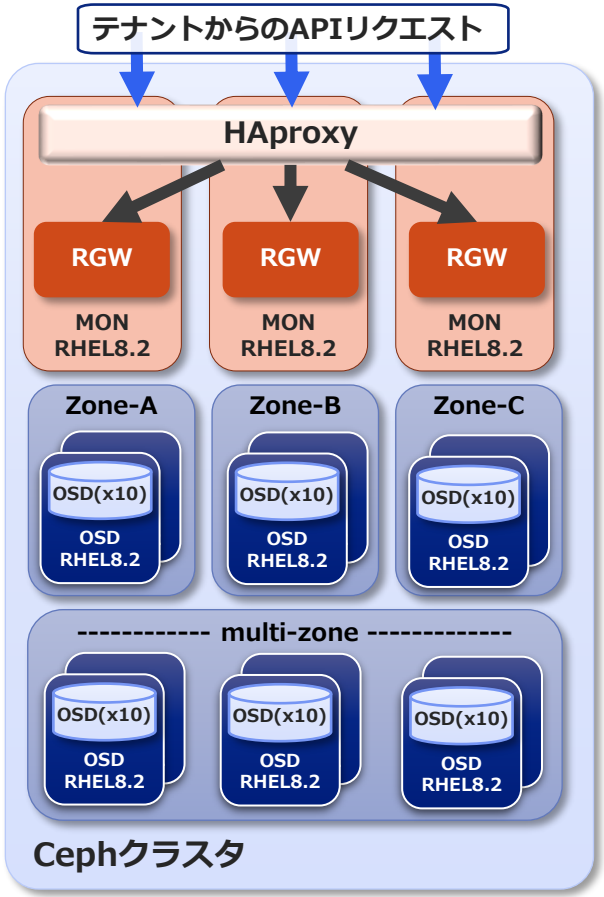
3. どのように課題を解決したか

課題解決に向けて

バージョンアップ前にRGWの機能を

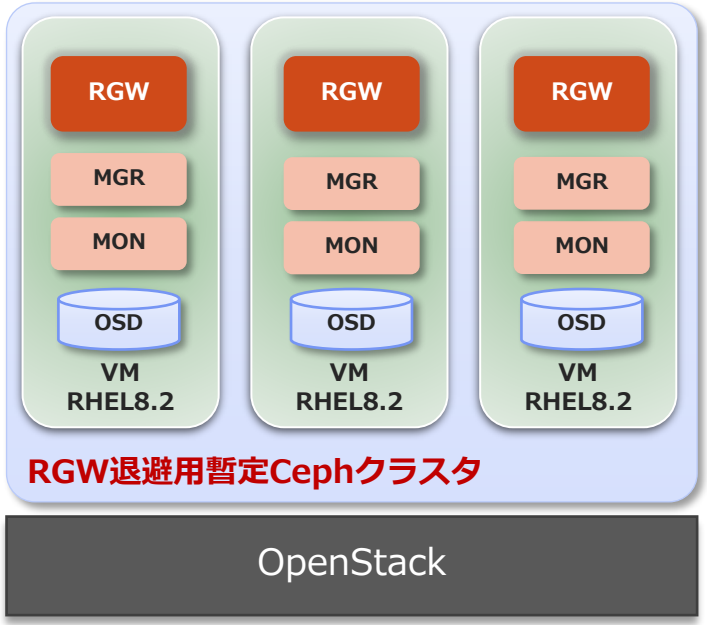
既存クラスタから切り離せばいいのでは？

課題の対策(1/3)

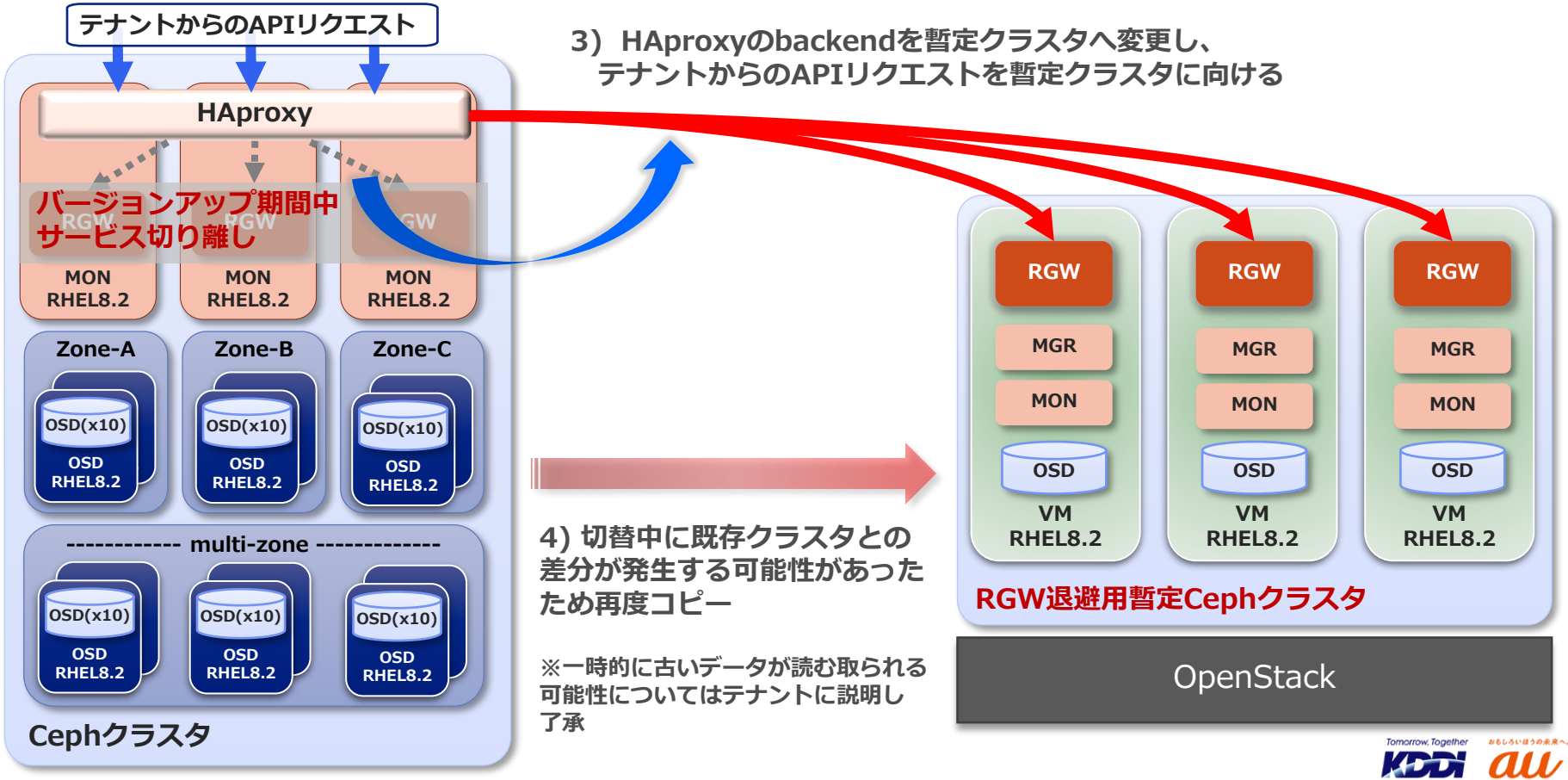


2) rclone(コピーツール)を使い、すべてのオブジェクトデータを現行クラスタから退避用クラスタへコピー

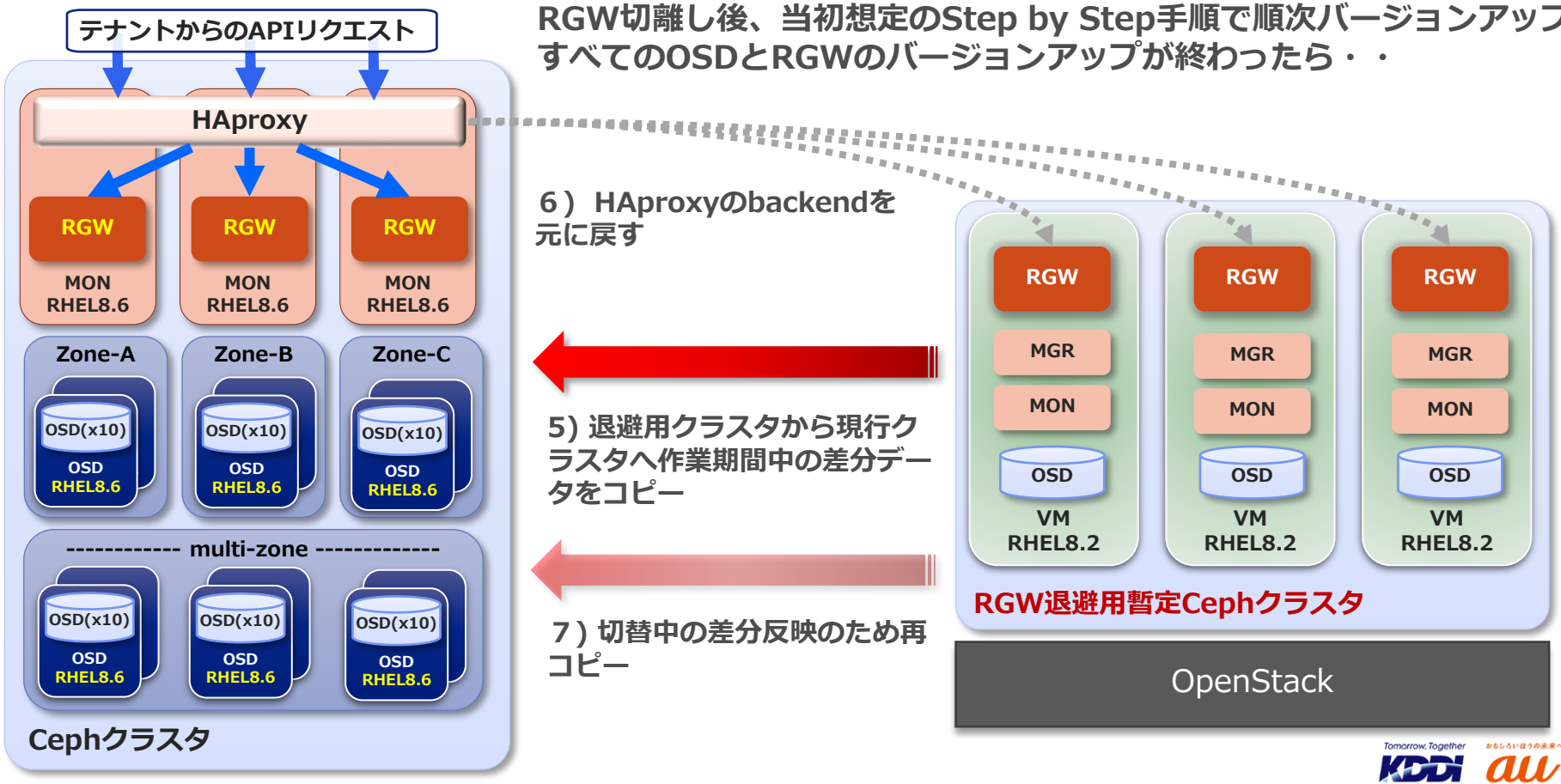
1) 暫定Cephクラスタを既存と同バージョンのVMで構築



課題の対策(2/3)


テナントからのAPIリクエスト
HAproxy
バージョンアップ期間中
サービス切り離し
RGW
RGW
GW
MON
RHEL8.2
MON
RHEL8.2
MON
RHEL8.2
Zone-A
Zone-B
Zone-C
OSD(x10)
OSD
RHEL8.2
OSD(x10)
OSD
RHEL8.2
OSD(x10)
OSD
RHEL8.2
multi-zone
OSD(x10)
OSD
RHEL8.2
OSD(x10)
OSD
RHEL8.2
OSD(x10)
OSD
RHEL8.2
Cephクラスタ
3) HAproxyのbackendを暫定クラスタへ変更し、
テナントからのAPIリクエストを暫定クラスタに向ける
4) 切替中に既存クラスタとの
差分が発生する可能性があった
ため再度コピー
※一時的に古いデータが読む取られる
可能性についてはテナントに説明し
了承
RGW
MGR
MON
OSD
VM
RHEL8.2
RGW
MGR
MON
OSD
VM
RHEL8.2
RGW
MGR
MON
OSD
VM
RHEL8.2
RGW退避用暫定Cephクラスタ
OpenStack

課題の対策(3/3)


テナントからのAPIリクエスト
HAproxy
RGW
RGW
RGW
MON
RHEL8.6
MON
RHEL8.6
MON
RHEL8.6
Zone-A
Zone-B
Zone-C
OSD(x10)
OSD
RHEL8.6
OSD(x10)
OSD
RHEL8.6
OSD(x10)
OSD
RHEL8.6
---------- multi-zone ----------
OSD(x10)
OSD
RHEL8.6
OSD(x10)
OSD
RHEL8.6
OSD(x10)
OSD
RHEL8.6
Cephクラスタ
RGW切離し後、当初想定のStep by Step手順で順次バージョンアップ
すべてのOSDとRGWのバージョンアップが終わったら・・
6）HAproxyのbackendを
元に戻す
5) 退避用クラスタから現行ク
ラスタへ作業期間中の差分デー
タをコピー
7）切替中の差分反映のため再
コピー
RGW
RGW
RGW
MGR
MGR
MGR
MON
MON
MON
OSD
OSD
OSD
VM
RHEL8.2
VM
RHEL8.2
VM
RHEL8.2
RGW退避用暫定Cephクラスタ
OpenStack

次の課題発生

ここで次の課題が・・・

- これ本当に出来るの？
- 検証環境は2環境しかないが、仕様上、一度バージョンアップすると元のバージョンには戻せない・・
- この手順のフィージビリティをどう確認するか
- 検証環境でいきなりは試せない・・

どうするか・・・？

デジタルツイン環境を構築しよう！

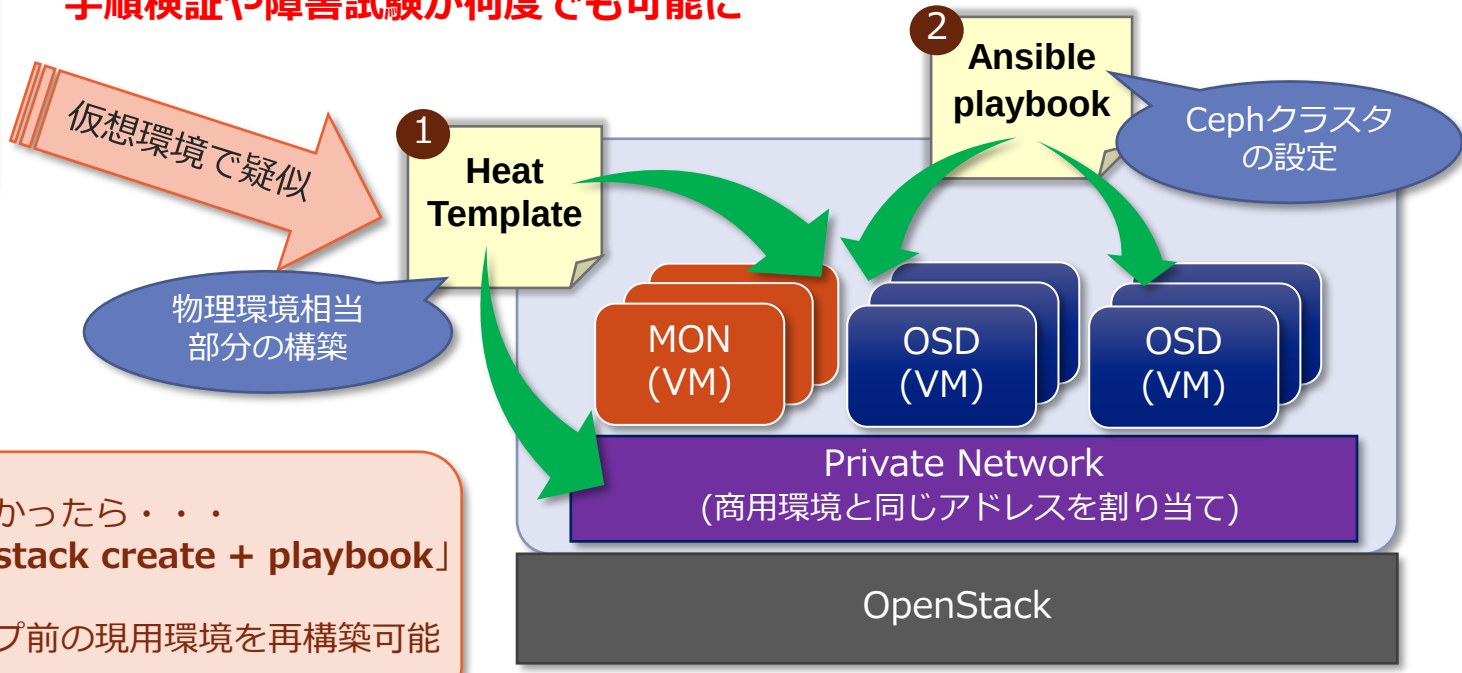
課題の対策

とある商用環境

MON(MGR/RGW)×3台
OSDx21台

OpenStack上にVMで本番環境の疑似環境を構築することで手順検証や障害試験が何度でも可能に

商用環境と同じアドレス、ホスト名、構成台数で構築(リソースはスモール)することで、構築用playbookは本番用のものを流用



手順検証がうまくいかなかったら・・・
「**stack delete**」 ⇒ 「**stack create + playbook**」
(1時間ほどで)
何度でもバージョンアップ前の現用環境を再構築可能

デジタルツイン環境構築にあたって

■ 環境構築で苦労したこと

- **Heat Template(HOT)はイチから作成**
  - それなりの台数のVMや周辺リソースをまとめて作成するため、効率的な書き方を考える必要があった

■ HOTを作る上で工夫した事

- **HOTを多段構成で作成**
  - 親HOTから子HOTに差分となる個々の情報(アドレスやホスト名など)を渡す形にして、NWやVMのresourceは1つだけ書く形に集約
- **「ResourceGroup」や「for_each」を使って複数ボリュームをまとめて記述**
  - OSDは1ホストあたり10volumeのアタッチが必要だったが、上記の記述によって10個並べて書かずに済んだ

デジタルツイン環境構築にあたって

■ 作成したHOTの一部サンプル(1/2)

親HOT

```
. . .
   osd_zone_a:
    type: ceph_vup_child_az.yaml
    properties:
      osd_image: {get_param: osd_image}
      zone: "a"
      az: "az-1"
      int_net_cidr: 10.1.16.160/27
      int_net_gw: 10.1.16.190
      int_net_dests:
        - 10.1.16.0/27
        - 10.1.16.32/27
        - 10.1.16.160/27
      osd_names:
        - east-ceosd-a-0001p
        - east-ceosd-a-0002p
        - east-ceosd-a-0003p
      port1_addrs:
        - 10.1.16.161
        - 10.1.16.162
        - 10.1.16.163
. . .
```

ホスト名、アドレスなど
子HOTに渡すパラメータ

子HOT　ceph_vup_child_az.yaml

```
  int_subnet:
    type: OS::Neutron::Subnet
    properties:
      name:
        str_replace:
          template: ceph_int_osd_$zone_subnet
          params:
            "$zone": {get_param: zone}
      network_id: {get_resource: int_net}
      cidr: {get_param: int_net_cidr}
      gateway_ip: {get_param: int_net_gw}
      host_routes:
        repeat:
          for_each:
            <%dest%>: {get_param: int_net_dests}
          template:
            destination: <%dest%>
            nexthop: {get_param: int_net_gw}
```

親から受け取ったパラメータによってzoneごとのNWを作成

```
osd_server:
   type: OS::Heat::ResourceGroup
   properties:
     count: 7
     resource_def:
       type: ceph_vup_child_osd.yaml
       properties:
         osd_image: {get_param: osd_image}
         flavor_osd: {get_param: flavor_osd}
         int_net: {get_resource: int_net}
         int_subnet: {get_resource: int_subnet}
         az: {get_param: az}
         zone: {get_param: zone}
         osd_name: {get_param: osd_names}
         port1_addr: {get_param: port1_addrs}
         port2_addr: {get_param: port2_addrs}
         osd_index: "%index%"
```

ResourceGroupを
使ってホスト数分定義

次頁に

デジタルツイン環境構築にあたって

■ 作成したHOTの一部サンプル(2/2)

孫HOT `ceph_vup_child_osd.yaml`

```
# *********************************
#   ceph-osd
# *********************************
  osd_server:
    type: OS::Nova::Server
    properties:
      name: {get_param: [osd_name, {get_param: osd_index}]}
      availability_zone: {get_param: az}
      image: {get_param: osd_image}
      flavor: {get_param: flavor_osd}
      networks:
        - port: {get_resource: osd_port_1}
        - port: {get_resource: osd_port_2}
      block_device_mapping_v2:
        repeat:
          permutations: False
          for_each:
            <%volid%>: {get_attr: [volume, refs]}
            <%dev%>: {get_param: device_names}
            <%bootindex%>:
              filter:
                - ['0']
                - get_attr: [val_grp, value]
          template:
            volume_id: <%volid%>
            device_name: <%dev%>
```

右で定義した10volumeを1ホストにまとめてアタッチ

```
# *********************************
#   volume
# *********************************
  volume:
    type: OS::Heat::ResourceGroup
    properties:
      count: 10
      resource_def:
        type: OS::Cinder::Volume
        properties:
          name:
            str_replace:
              template: ceph_vup_volume_$zone$num_%index%
              params:
                "$zone": {get_param: zone}
                "$num": {get_param: osd_index}
          availability_zone: {get_param: az}
          size: '20'
          volume_type: 'iops_rw_no_limit'
```

10volume分をまとめて定義

デジタルツイン環境構築にあたって

■ 良かったこと

- 商用環境と**同じアドレス、ホスト名、構成台数**で仮想インフラ構築したことで、設定用playbookは**本番用のものを使いまわし**出来てスムーズに構築できた

- 仮想化基盤の利用テナントに影響しない環境のため、普段できないような障害パターンの試験やその復旧手順の検証を行うこともできた
  - 自信をもって作業に望めた

■ 結果として・・・

- 作業は無事完了
- 作業時の問題発生はゼロ

デジタルツイン環境構築にあたって

■ 育成環境としても良かった

- 環境構築や試験をまだチームに参加して日が浅いメンバーに対応してもらうことで、Cephのデプロイ方法や障害時の動作についてTry&Errorを繰り返しながら学ぶことができた
- 今後新規に入ってくるメンバーに対するハンズオン環境が整い、オンボーディングの効率化ができようになった

■ 結果として・・・

- チーム全体のスキルアップにつながった
- 少ないメンバーで作業を回す事ができた(当時3人)

作業時間短縮のための工夫

一応作業手順としては確立できたが

まだまだ課題が・・・

- この作業をストレートにやるとかなり時間かかる
- チーム人数が少ないためそんなに時間をかけられない
- 他の案件もある、、、

課題

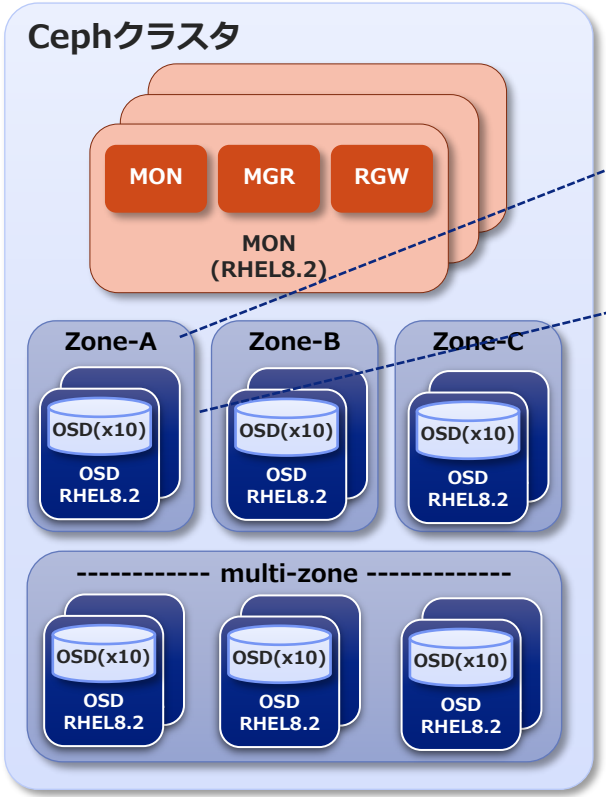
① 作業期間の長期化

- **状況**
  - 当初想定は4ケ月
- **問題点**
  - 公式手順では期間を空けずにバージョンアップすることが推奨のため、作業期間中のリスク増
- **要因**
  - データ退避をしながらバージョンアップするため、退避と戻しに時間がかかる

② メンバー編成

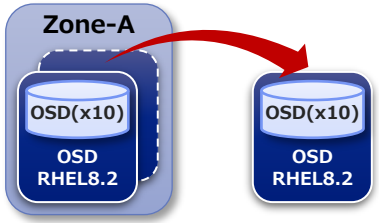
- **状況**
  - 商用作業は2人体制が必須
  - 当時のチームメンバーは３名
  - 他案件（新規基盤構築、コンテナ基盤のバージョンアップ等）も並行進捗中
- **問題点**
  - 他案件と並行するには作業人員が不足

課題①の工夫点(1)

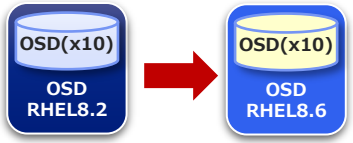


当初の想定

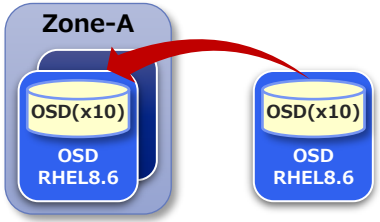
①データを退避してクラスタから切り離す(1～2h)



②切り離したノードをバージョンアップする(～1h)



③バージョンアップしたノードをクラスタに組み込む(1～2h)

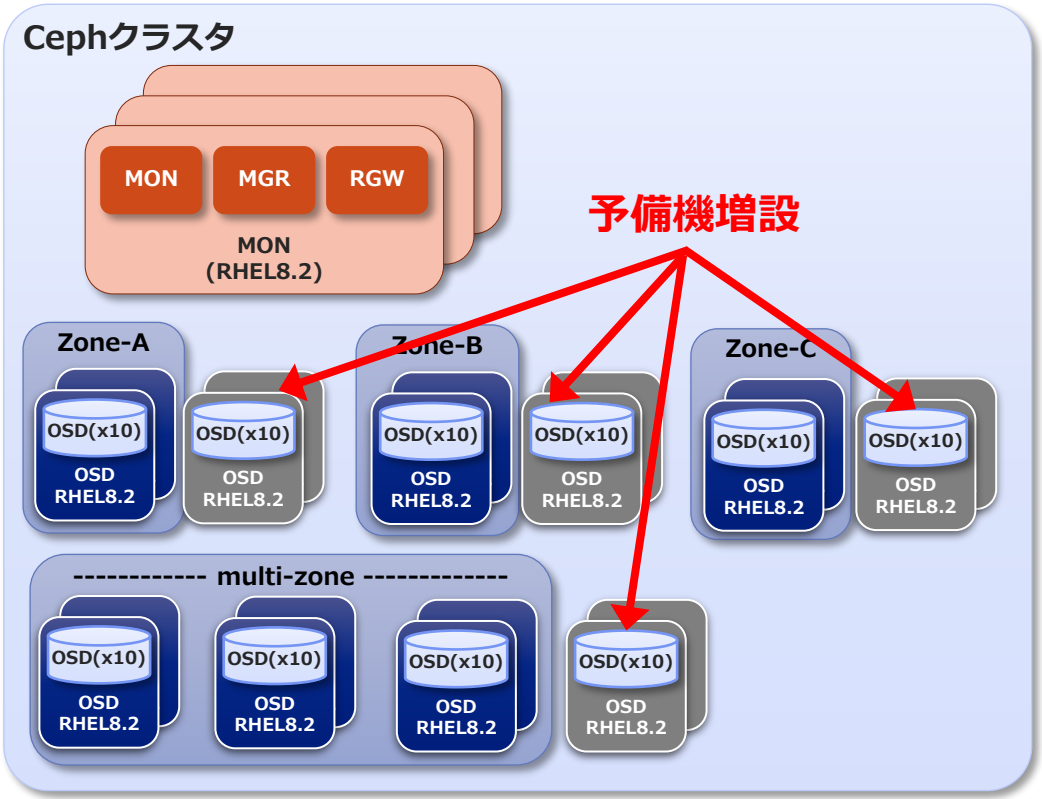


①～③を台数分繰り返す・・・

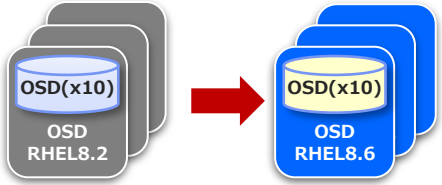
（作業時間は上記合計ｘOSDホスト台数）

課題①の工夫点(1)

増設用予備ノードの活用

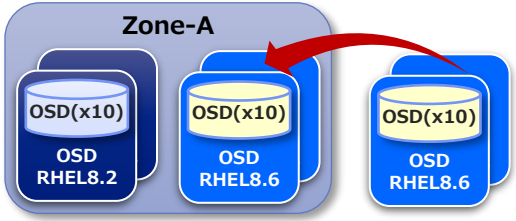


① 予備機をまとめてバージョンアップ(3h)



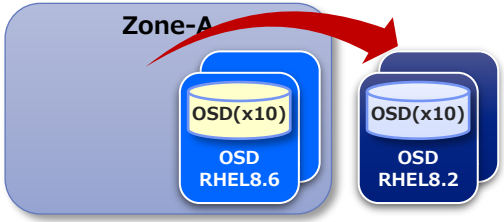
一括バージョンアップで時間短縮

② Zoneごとに組み込み(1h～5h)



zoneによっては、まとめて組み込みし、時間短縮

③ 既存ノードの切り離し



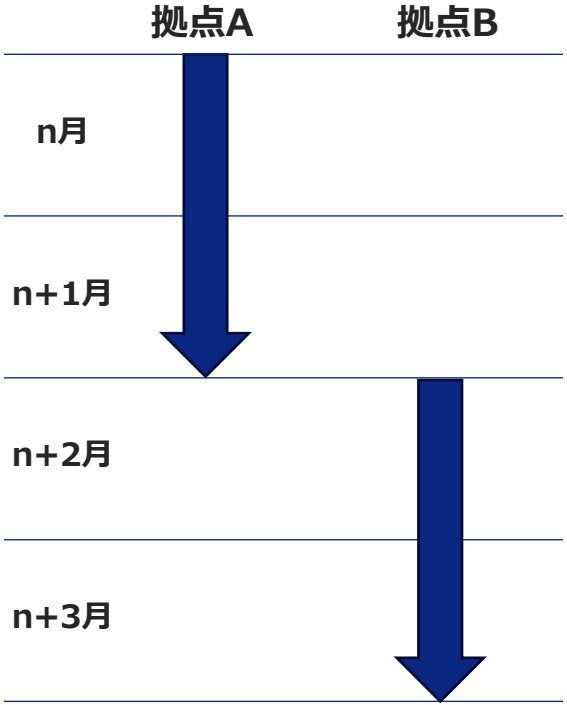
切離し後、まとめてバージョンアップして、予備機に

②、③をzoneごとに繰り返す

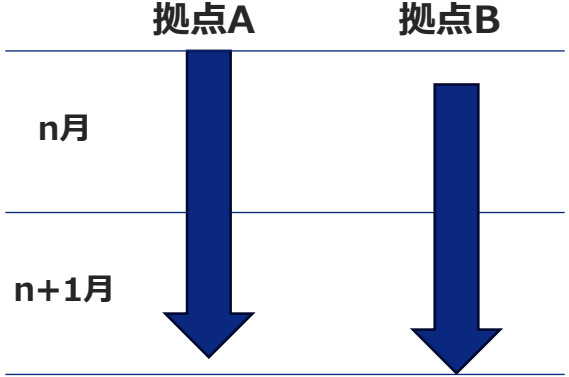
課題①の工夫点(2)

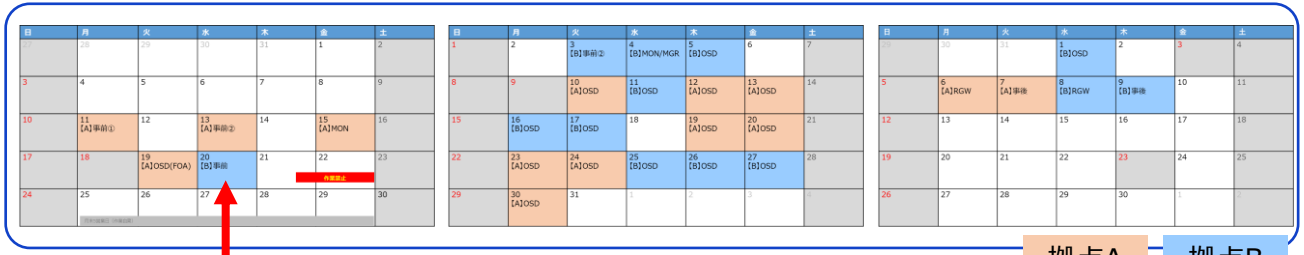
2拠点並行実施

- 当初の想定



- 作業並行化





拊点Aのバージョンアップ安定確認期間中に
拊点Bのバージョンアップを実施

拊点A　拊点B

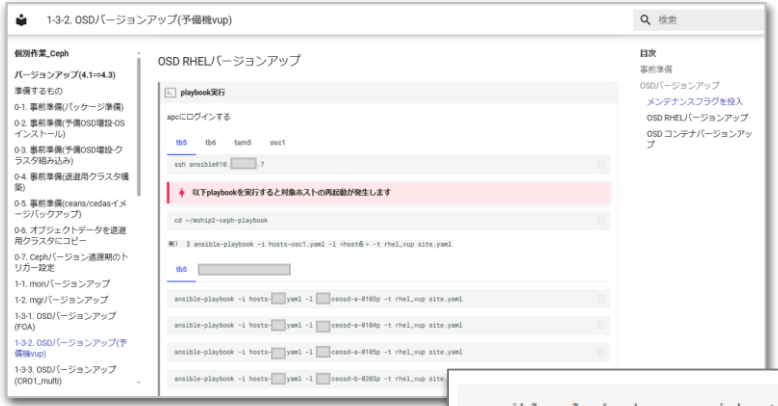
課題②の工夫点(1)

- 現地保守担当者の協力
  - 人員不足を補うため、他部署に協力を依頼
  - 作業のシンプル化、手順書の品質を上げることでCephに触れたことのない人でも開発担当とペアで作業可能に
  - 開発担当１名＋協力者１名体制で当作業を進めることで他案件も並行対応可能に
  - 現地担当者のスキルアップの機会にもなってWin-Win

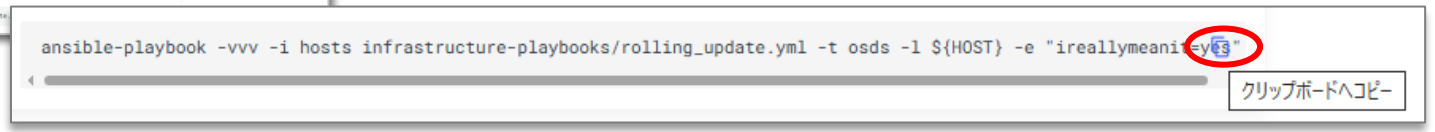
課題②の工夫点(2)

- **作業手順の品質向上**
  - **手順書をMkDocsで作成し、gitで管理することで改版管理、レビューを容易に**
  - **コピペ機能を活用し作業ミスを低減**

MkDocsで作成した手順書をブラウザで表示して使用



長いコマンドもワンクリックでコピー



QA

Tomorrow, Together
KDDI
おもしろいほうの未来へ。
au

「つなぐチカラ」を進化させ、
誰もが思いを実現できる社会をつくる。

KDDI VISION 2030