OpenStackでも重要な役割を果たす Pacemakerを知ろう！

～クラウドの雲の下では何が起こっているのか～

2016年11月5日

Linux-HA Japan プロジェクト
http://linux-ha.osdn.jp/
森 啓介

自己紹介

- 名前: 森 啓介 (Keisuke MORI)
  - twitter: @ksk_ha
- Linux-HA Japanプロジェクト関連の活動
  - Pacemakerリポジトリパッケージのリリース
  - http://linux-ha.osdn.jp/
- ClusterLabs プロジェクトのコミッタ
  - Pacemaker、resource-agents などHAクラスタ関連の開発コミュニティ
  - https://github.com/ClusterLabs/
- 本業
  - 普段の業務: NTT OSSセンタ
    - NTTグループ内におけるPacemaker/Heartbeatの導入支援・サポート
    - バグ報告・パッチ作成などによるNTTから開発コミュニティへのフィードバック・貢献

もくじ

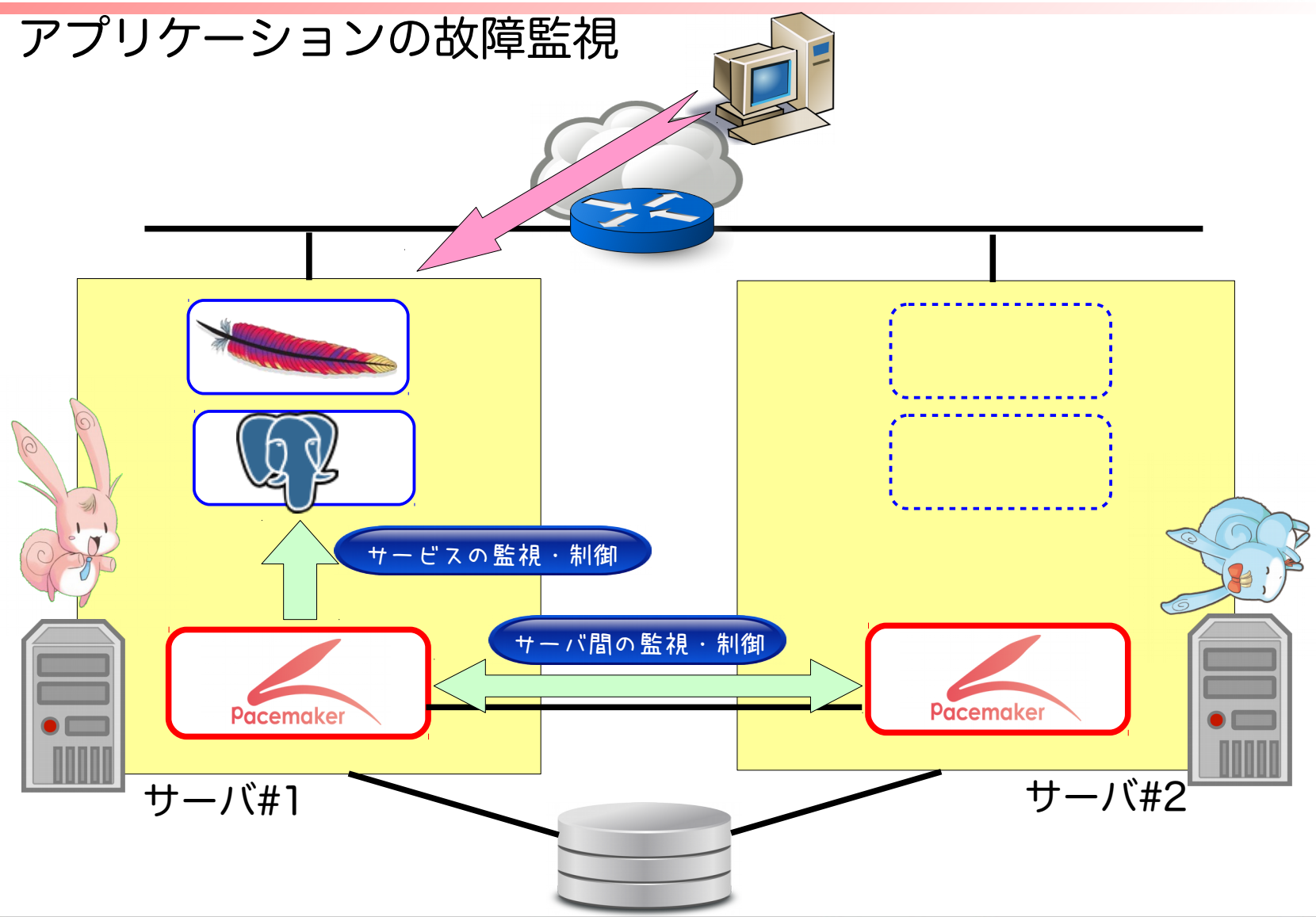
- Pacemakerとは
- OpenStack におけるHAの必要性
- Red Hat OpenStack Platform でのHA構成例
- OpenStack HA の今後の動向

もくじ

- Pacemakerとは
- OpenStack におけるHAの必要性
- Red Hat OpenStack Platform でのHA構成例
- OpenStack HA の今後の動向

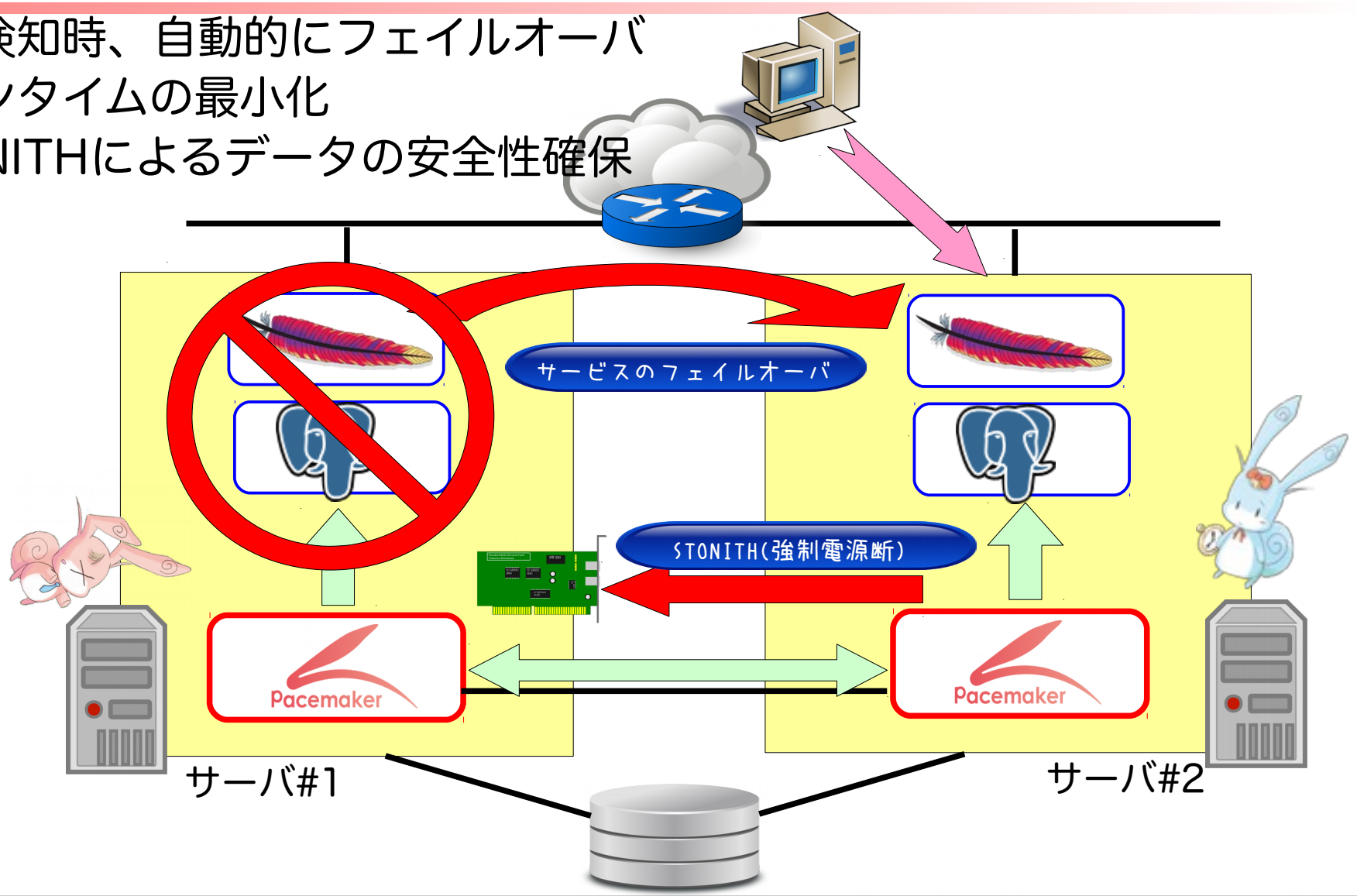
Pacemakerの概要

- サーバ・アプリケーションの故障監視



Pacemakerの概要

- 故障検知時、自動的にフェイルオーバ
- ダウンタイムの最小化
- STONITHによるデータの安全性確保



Pacemakerを詳しく知りたかったら…

Pacemakerとは…　高可用(HA)クラスタソフトウェアです。

Linux-HA Japan プロジェクト 2F 205教室にてデモ展示中！

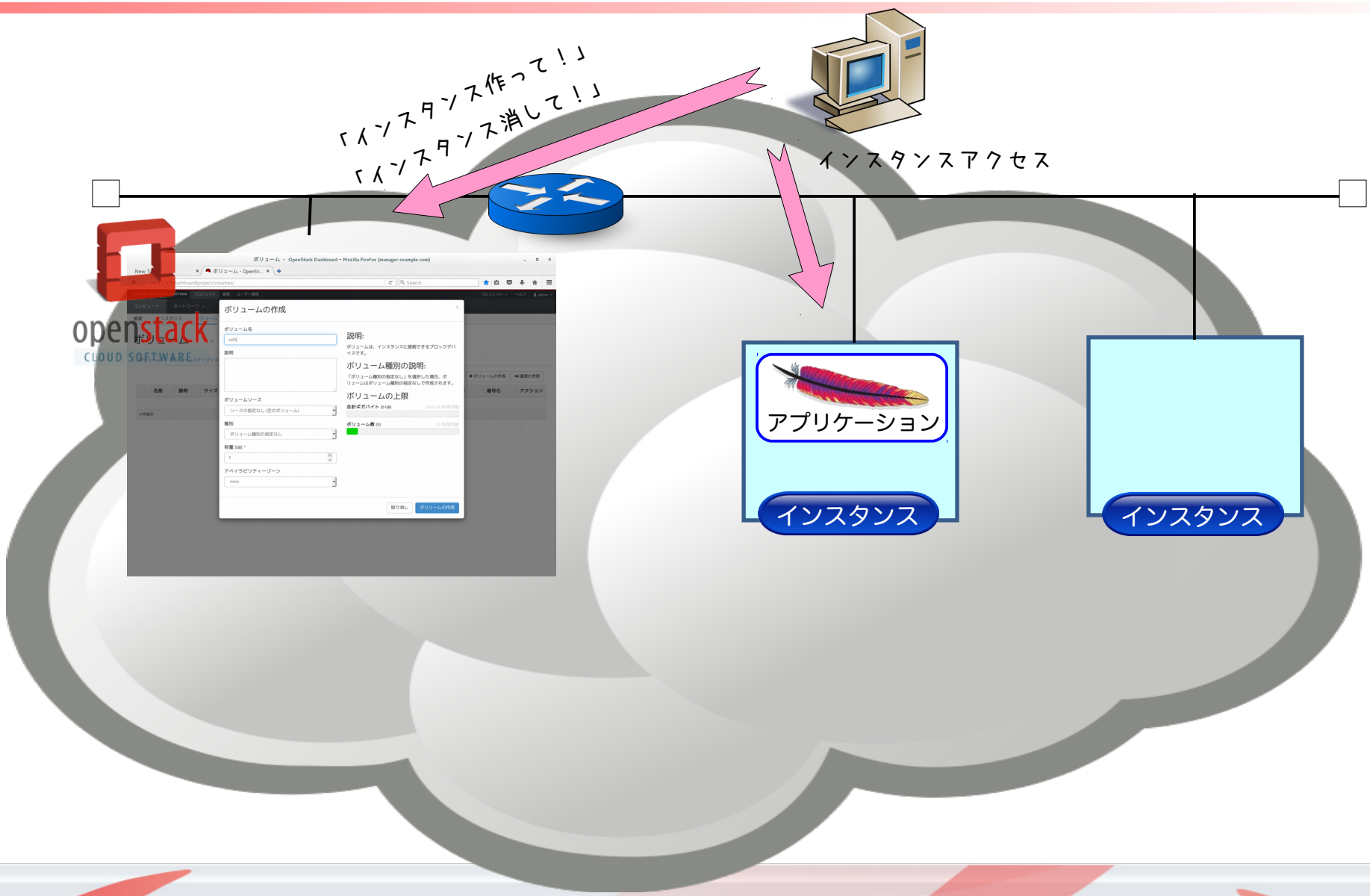
もくじ

- Pacemakerとは
- OpenStack におけるHAの必要性
- Red Hat OpenStack Platform でのHA構成例
- OpenStack HA の今後の動向

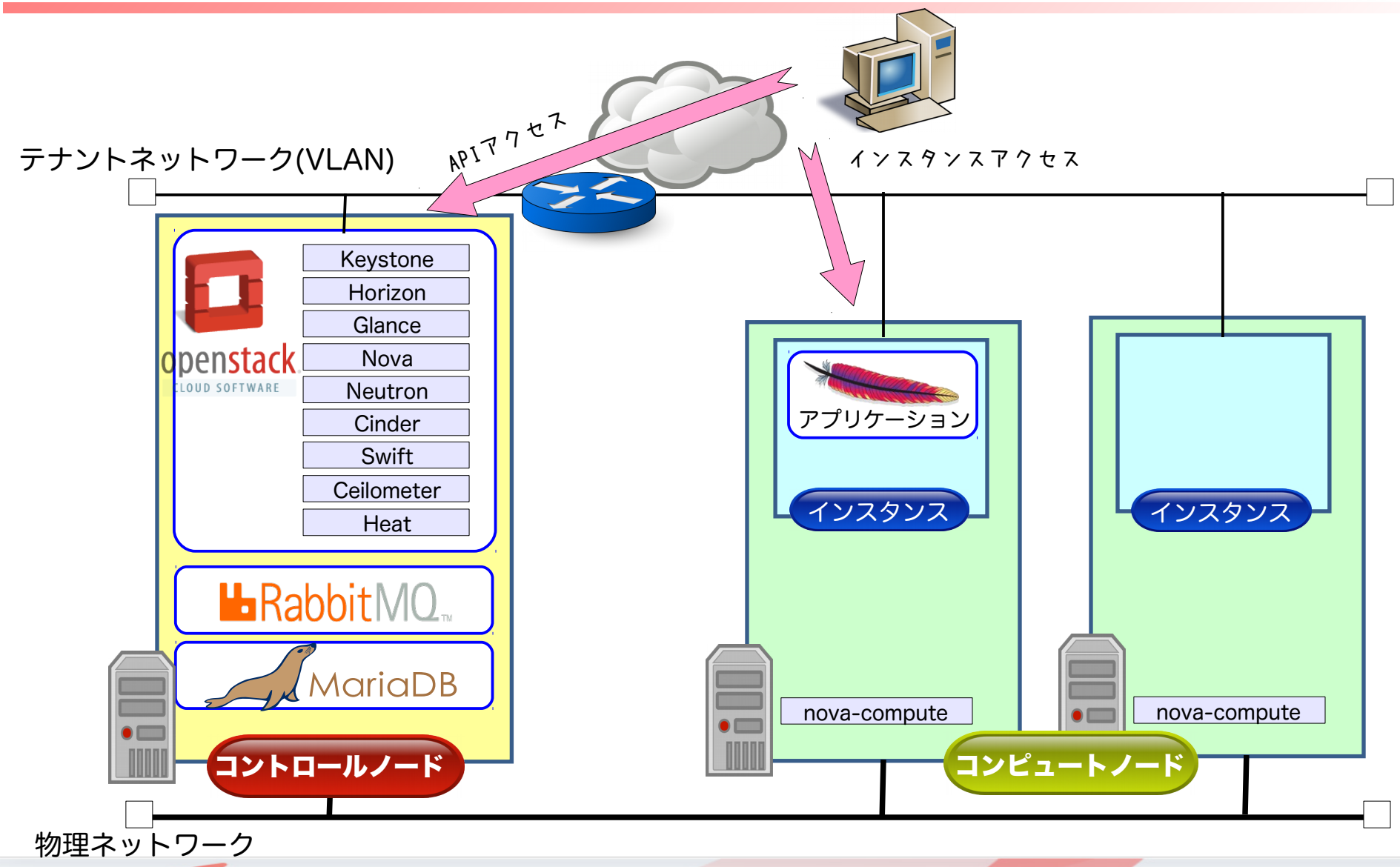
OpenStackにおけるHAの必要性

クラウド時代にHAクラスタなんているのぉ？？

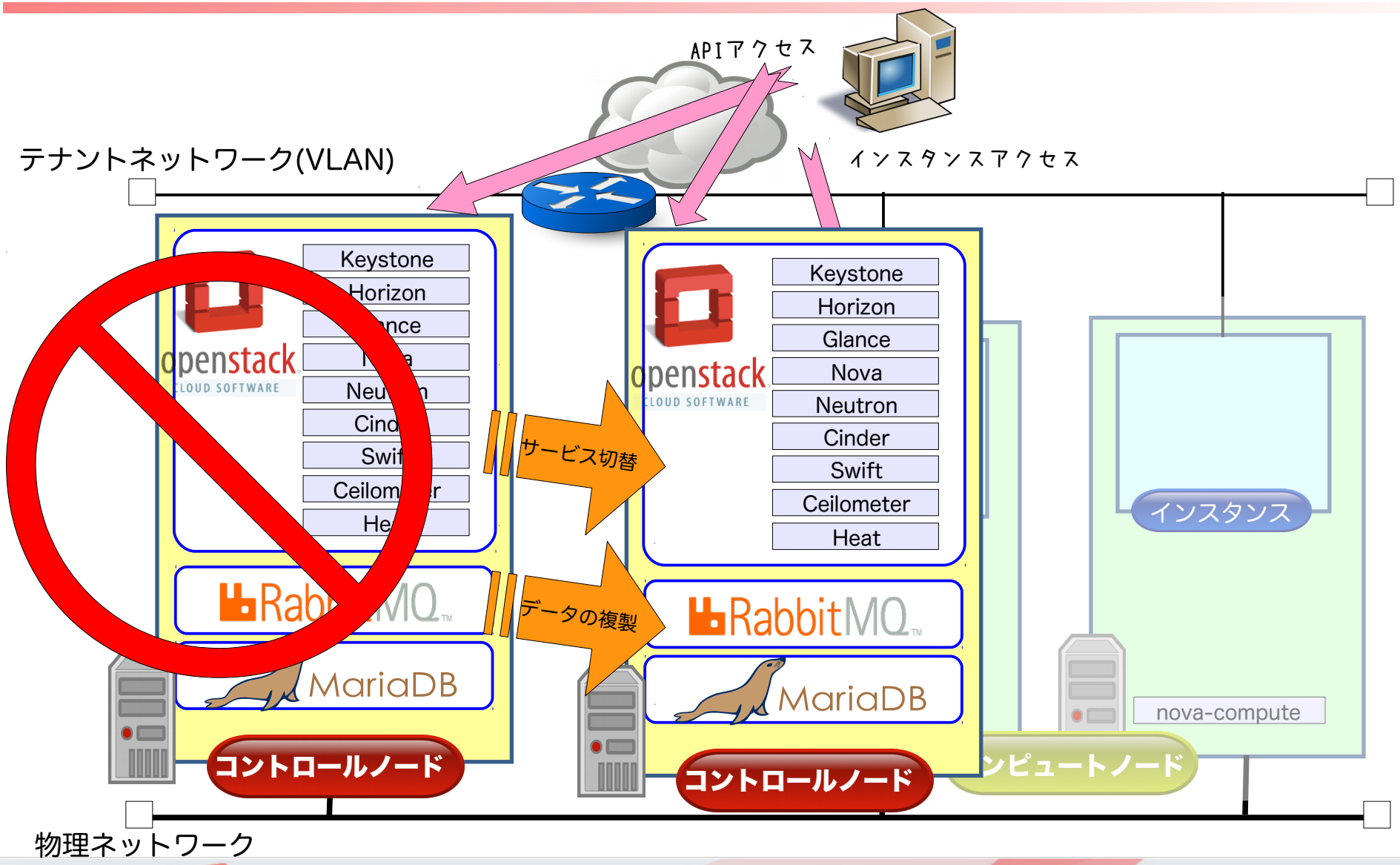
クラウドユーザ(テナント)からみた世界



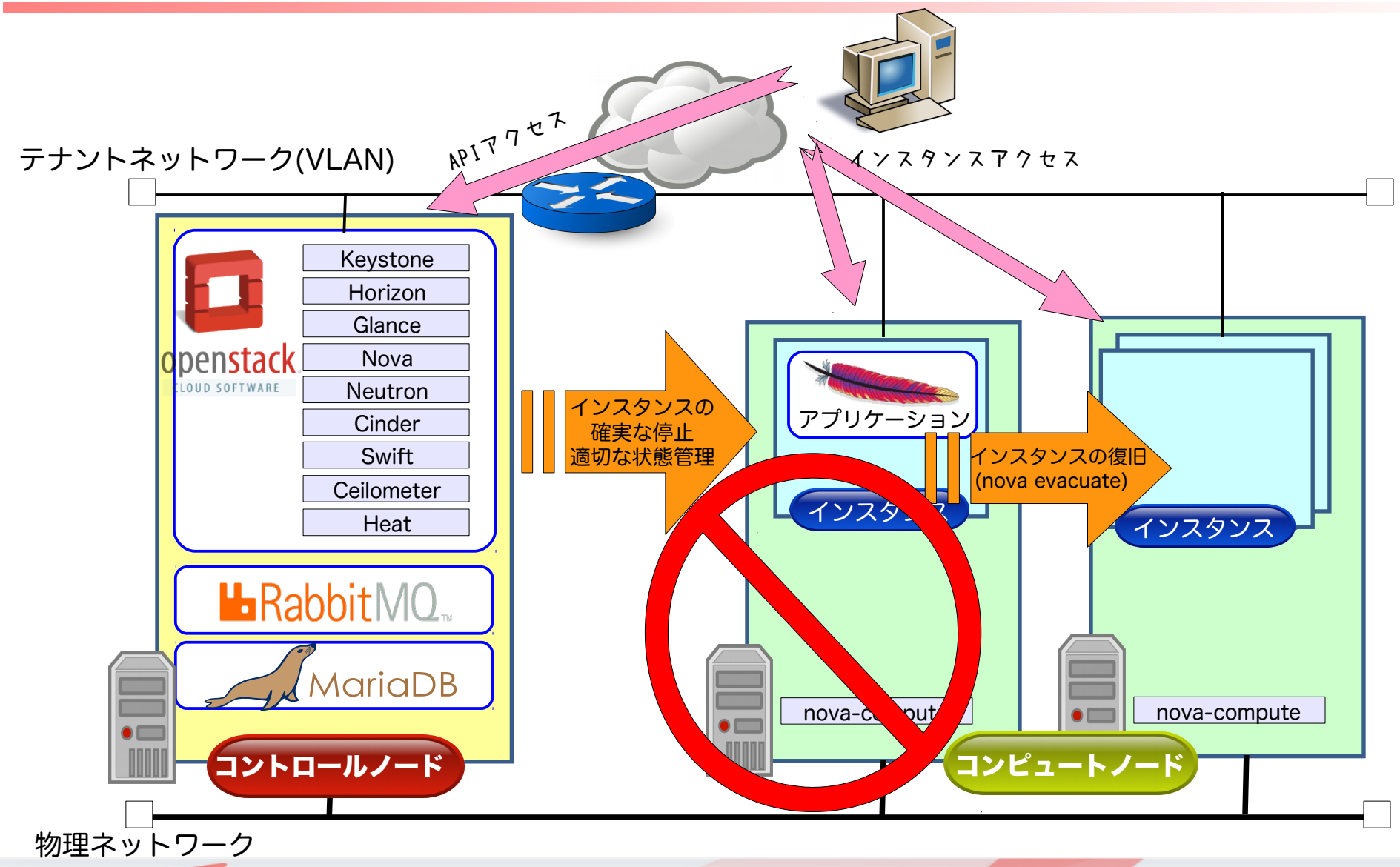
雲の下では…


テナントネットワーク(VLAN)
APIアクセス
インスタンスアクセス
Keystone
Horizon
Glance
Nova
Neutron
Cinder
Swift
Ceilometer
Heat
openstack
CLOUD SOFTWARE
RabbitMQ
MariaDB
コントロールノード
アプリケーション
インスタンス
インスタンス
nova-compute
nova-compute
コンピュートノード
物理ネットワーク

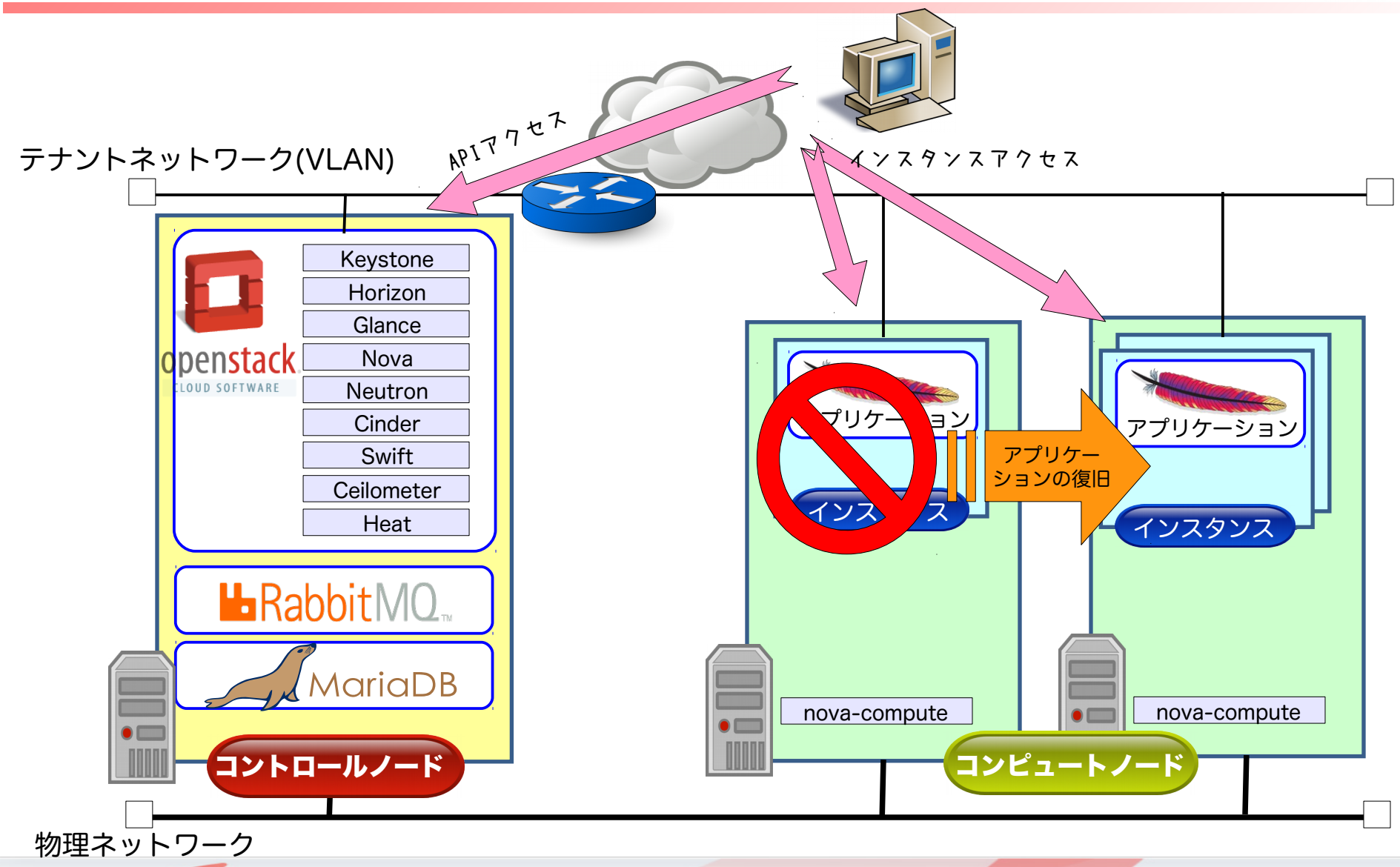
コントロールノードの故障



コンピュートノードの故障



アプリケーションの故障



OpenStackにおけるHAの必要性

クラウド時代にHAクラスタなんているのぉ？？

当然必要よ！

OpenStackにおけるHAの観点

- コントロールノードHA
  - コントロールノードの物理故障・通信断
  - 各OpenStackサービスプロセス・データベースサーバ・メッセージキューのソフトウェア故障
- コンピュートノードHA
  - コンピュートノードの物理故障・通信断
  - インスタンスのソフトウェア故障

クラウド提供者
(基盤管理者)
による管理

- アプリケーションHA
  - インスタンスのソフトウェア故障
  - テナントユーザのサービス(アプリケーション)故障

クラウドユーザ
(テナント)
による管理

OpenStackにおけるHAの必要性

- クラウド提供者(基盤管理者)にとっては…
  - クラウドユーザに対するOpenStackサービスの可用性確保
    - 従来通りの物理環境での HAクラスタが必要
  - インスタンスの可用性確保
    - 従来の HA クラスタのさらなる応用が必要

- クラウドユーザ(テナント)にとっては…
  - クラウド提供者(基盤管理者)が可用性を確保しているからこそ安心して利用が可能
  - 自分のサービス（アプリケーション)の可用性は自分で確保する必要あり
    - クラウド上でのHAクラスタ構成
    - クラウド特有の方式 (Ceilometer / Heat / Senlin 等)

主なディストリビューションのHA方式

ディストリビューション	構築方式	HAクラスタソフトウェア	コントロールノードHA	コンピュートノードHA	データベース冗長化方式	補足
Openstack.org (upstream)	手動	Pacemaker + Corosync	△	ー	Galera	• 公式ドキュメントでは要件・概念と手順の断片が示されているのみ。 • 具体的な設定・構築手順等は独自に設計する必要あり。 • 2016年10月時点で日々更新中
Red Hat OpenStack Platform 8 (RH OSP)	director (Triple O)	Pacemaker + Corosync	○	○	Galera / MariaDB	• コントロールノード構成は3ノード以上 • 全てのOpenStackサービスが同居する構成前提 • Neutron はL3HA設定によるActive/Active構成
SUSE OpenStack Cloud 6	Crowbar + barclamp	Pacemaker + Corosync	○	○	PostgreSQL + DRBD	• コントロールノード構成はDBサーバのみ2ノード、他のOpenStackサービスは3ノード以上 • Neutron はスクリプト方式(ha-tools)によるHA方式
Ubuntu	MAAS + juju	Pacemaker + Corosync	○	ー	Percona XtraDB Cluster	
Mirantis	Fuel	Pacemaker + Corosync	○	ー	Galera / MySQL	

今日のトピックはココ!

※主に Liberty ベース時点での情報

OpenStackにおけるHAの必要性

クラウド時代にHAクラスタなんているのぉ？？

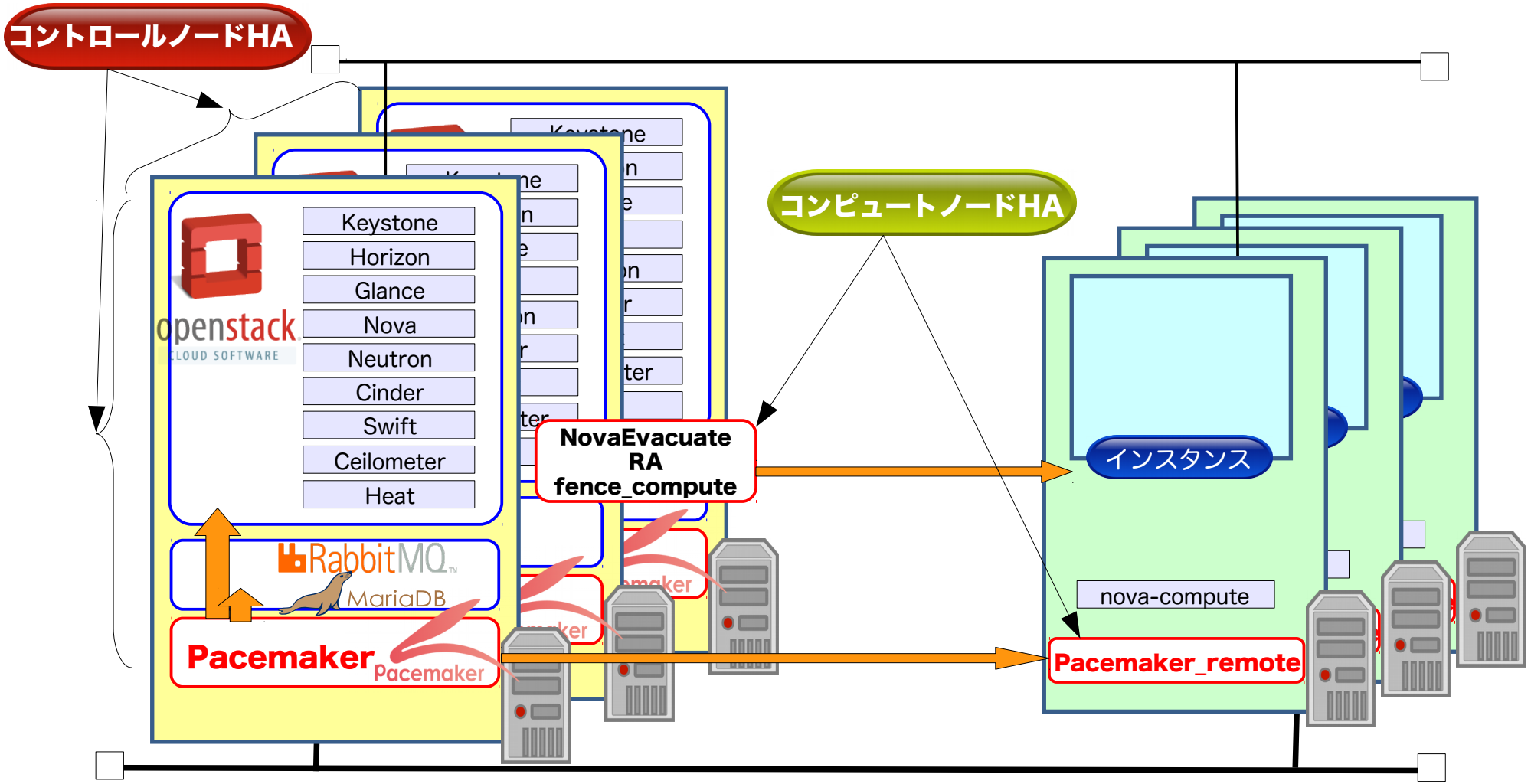
当然必要よ！

OpenStackクラウド基盤の高可用化にもPacemakerは使われてるの!

もくじ

- Pacemakerとは
- OpenStack におけるHAの必要性
- Red Hat OpenStack Platform でのHA構成例
- OpenStack HA の今後の動向

Red Hat OpenStack Platform 8 のHA構成概要

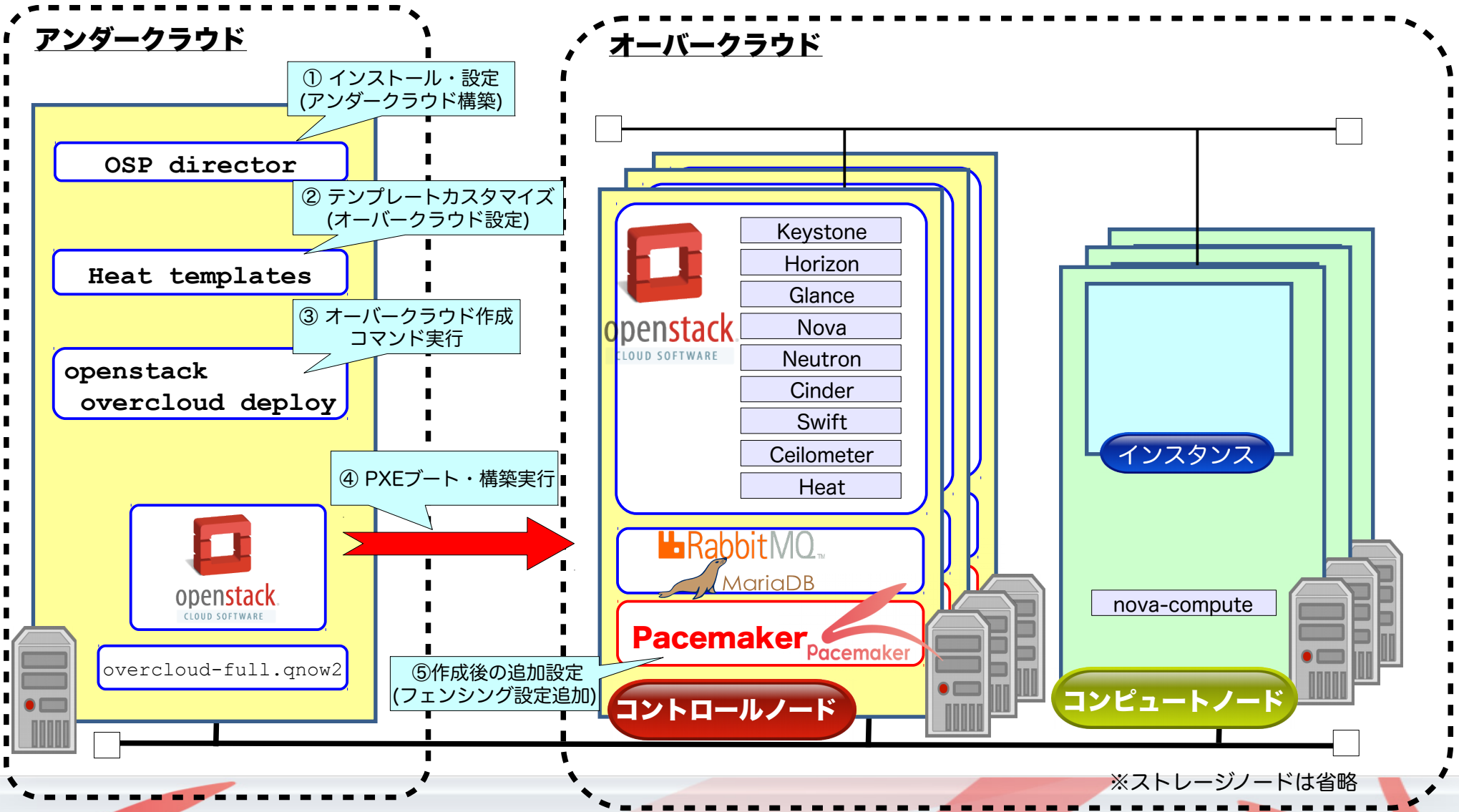


※ストレージノードは省略

Red Hat OpenStack Platform 8 HA構成の特徴

- コントロールノードHA
  - ノード構成: 3ノード以上
    - クォーラム制御(多数決制御)のため
  - すべてのOpenStackサービス・データベースサーバ等が同居する構成が前提
    - ストレージノードのみ分離可能
    - 本資料では説明の単純化のためNFSサーバを使用している
- コンピュートノードHA
  - Pacemaker_remote機能, NovaEvacuate RA による実装方式
- OSP director (Triple O) によるHA構成の自動構築
  - 実際にサービスを提供するOpenStack環境(オーバークラウド)を、構築用の別のマシン(アンダークラウド)から自動構築する
  - コントロールノードHAは自動構築が可能
  - コンピュートノードHAについては自動構築対象外
    - ドキュメントに手動構築手順あり

RH OSP 8 のHA環境構築の流れ



※ストレージノードは省略

RH OSP 8 のHA環境構築

Pacemaker

オーバークラウド作成コマンド オプション例

```
[stack@bl460g9n6 ~]$ openstack overcloud deploy
  --templates
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-management.yaml
  -e ~/templates/network-environment.yaml
  -e ~/templates/storage-environment.yaml
 --control-scale 3
 --compute-scale 1
 --control-flavor control
 --compute-flavor compute
 --ntp-server 192.168.28.80
 --neutron-network-type vxlan
 --neutron-tunnel-types vxlan
```

ポイント

- コントロールノードのノード数を指定する
- HA構成のためには 3ノード以上、奇数ノード数が必要
  - --control-scale 1：冗長化なし
  - --control-scale 2：冗長化なし(1ノード故障時にサービス停止)
    - 偶数ノード数では半数のノードが故障した時点でサービス停止となる（クォーラム制御)
  - --control-scale 3：冗長化あり

RH OSP 8 のHA環境構築

Pacemaker

オーバークラウド作成後の追加設定

コントローラノードのフェンシング設定追加

```
$ sudo pcs stonith create my-ipmilan-for-controller01 fence_ipmilan pcmk_host_list=overcloud-
controller-0 ipaddr=192.168.28.43 login=USERID passwd=PASSW0RD lanplus=1 op monitor interval=60s

$ sudo pcs constraint location my-ipmilan-for-controller01 avoids overcloud-controller-0

$ sudo pcs stonith create my-ipmilan-for-controller02 fence_ipmilan pcmk_host_list=overcloud-
controller-1 ipaddr=192.168.28.42 login=USERID passwd=PASSW0RD lanplus=1 op monitor interval=60s

$ sudo pcs constraint location my-ipmilan-for-controller02 avoids overcloud-controller-1

$ sudo pcs stonith create my-ipmilan-for-controller03 fence_ipmilan pcmk_host_list=overcloud-
controller-2 ipaddr=192.168.28.41 login=USERID passwd=PASSW0RD lanplus=1 op monitor interval=60s

$ sudo pcs constraint location my-ipmilan-for-controller03 avoids overcloud-controller-2

$ sudo pcs property set stonith-enabled=true
```

ポイント

- フェンシング機能(STONITH機能)設定はハードウェアに依存するため、環境に合わせ手動で設定追加を行う。
  - 一般的には IPMI 経由による強制電源断
- フェンシング機能はデータベース・メッセージキューのデータ整合性を保障するために非常に重要(スプリットブレインの防止)

何ができあがったの?

- crm_mon コマンドで見てみよう
  - Pacemaker のリソース稼働状況を確認するコマンド
  - 表示例: よくあるWeb・DBサーバの冗長構成の場合は…

```
2 Nodes configured
16 Resources configured

Online: [ pm01 pm02 ]

Full list of resources:

 Resource Group: master-group
     filesystem        (ocf::heartbeat:Filesystem):       Started pm01
     apache (ocf::heartbeat:apache):          Started pm01
     vip-master        (ocf::heartbeat:IPaddr2):          Started pm01
     vip-rep           (ocf::heartbeat:IPaddr2):          Started pm01
 Resource Group: grpStonith1
     prmStonith1-1     (stonith:external/stonith-helper):Started pm02
     prmStonith1-2     (stonith:external/ipmi):           Started pm02
 Resource Group: grpStonith2
     prmStonith2-1     (stonith:external/stonith-helper):Started pm01
     prmStonith2-2     (stonith:external/ipmi):           Started pm01
 Master/Slave Set: msPostgresql [pgsql]
     Masters: [ pm01 ]
     Slaves: [ pm02 ]
 Master/Slave Set: msDrbd [drbd]
     Masters: [ pm01 ]
     Slaves: [ pm02 ]
 Clone Set: clnPing [prmPing]
     Started: [ pm01 pm02 ]
 Clone Set: clnDiskd1 [prmDiskd1]
     Started: [ pm01 pm02 ]
```

- ノード数: 2
- 稼働リソース数: 16

RH OSP 8 でのcrm_mon出力

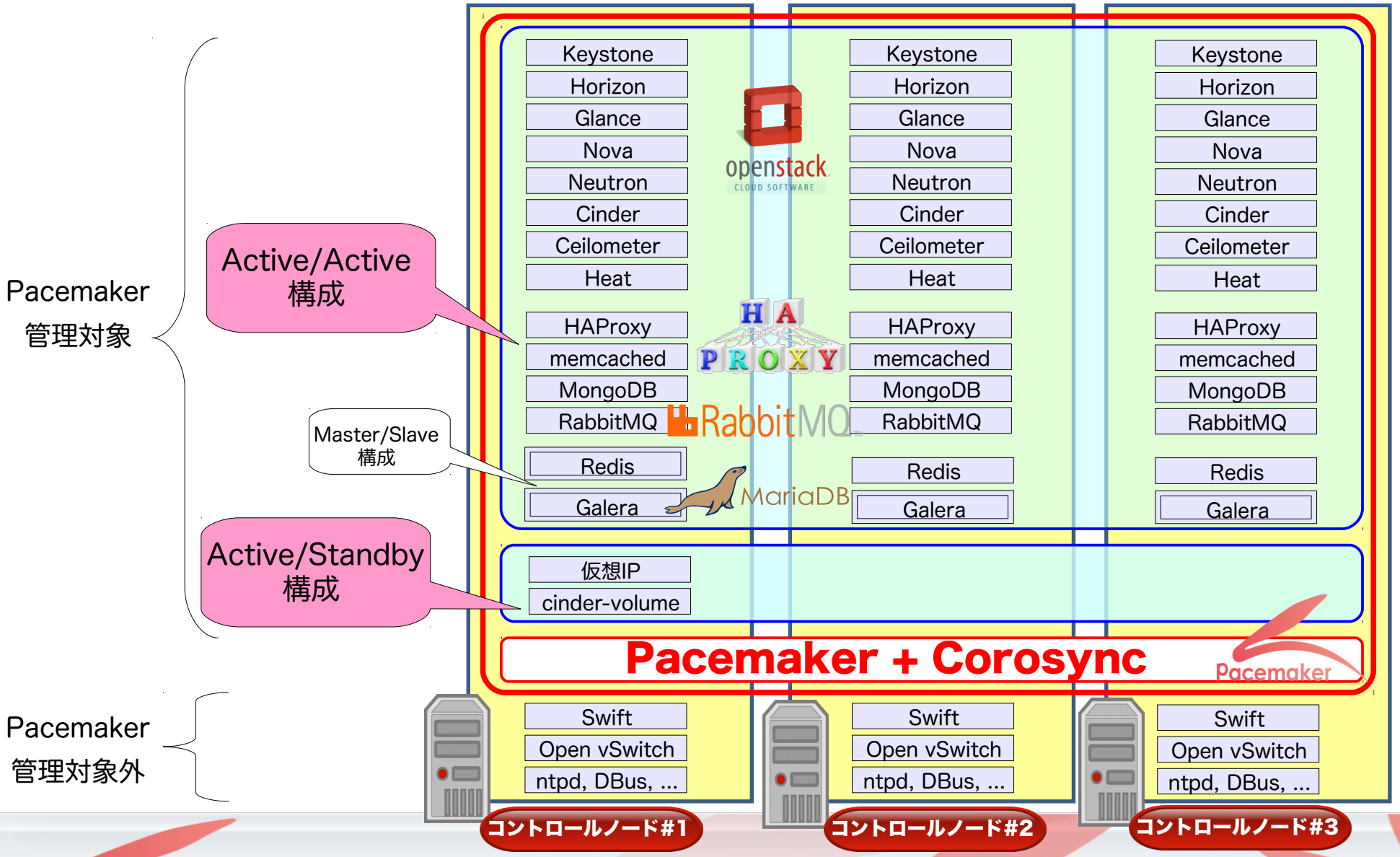
```
3 nodes and 118 resources configured

Online: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]

Full list of resources:

 Clone Set: haproxy-clone [haproxy]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 ip-172.18.0.10      (ocf::heartbeat:IPaddr2):               Started overcloud-controller-0
 ip-172.16.0.11      (ocf::heartbeat:IPaddr2):               Started overcloud-controller-1
 ip-172.16.0.10      (ocf::heartbeat:IPaddr2):               Started overcloud-controller-2
 ip-172.19.0.10      (ocf::heartbeat:IPaddr2):               Started overcloud-controller-0
 ip-192.0.2.26       (ocf::heartbeat:IPaddr2):               Started overcloud-controller-1
 ip-10.1.1.10        (ocf::heartbeat:IPaddr2):               Started overcloud-controller-2
 Master/Slave Set: redis-master [redis]
     Masters: [ overcloud-controller-1 ]
     Slaves: [ overcloud-controller-0 overcloud-controller-2 ]
 Master/Slave Set: galera-master [galera]
     Masters: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: mongod-clone [mongod]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: rabbitmq-clone [rabbitmq]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-contro
 Clone Set: memcached-clone [memcached]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-
 Clone Set: fs-varlibglanceimages-clone [fs-varlibglanceimages]
     Started: [ overcloud-controller-0 overcloud-controller-1 overc
 Clone Set: openstack-nova-scheduler-clone [openstack-nova-scheduler]
     Started: [ overcloud-controller-0 overcloud-controller-1 overclo
 Clone Set: neutron-l3-agent-clone [neutron-l3-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-alarm-notifier-clone [openstack-ceilometer-alarm-
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controlle
 Clone Set: openstack-heat-engine-clone [openstack-heat-engine]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-api-clone [openstack-ceilometer-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-metadata-agent-clone [neutron-metadata-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-ovs-cleanup-clone [neutron-ovs-cleanup]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-netns-cleanup-clone [neutron-netns-cleanup]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-heat-api-clone [openstack-heat-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-cinder-scheduler-clone [openstack-cinder-scheduler]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
```

- ノード数: 3
- 稼働リソース数: 118

なるほど、わからん…

```
Clone Set: openstack-nova-api-clone [openstack-nova-api]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-heat-api-cloudwatch-clone [openstack-heat-api-cloudwatch]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-ceilometer-collector-clone [openstack-ceilometer-collector]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-keystone-clone [openstack-keystone]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-nova-consoleauth-clone [openstack-nova-consoleauth]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-glance-registry-clone [openstack-glance-registry]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-ceilometer-notification-clone [openstack-ceilometer-notification]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clo                inder-api-clone [openstack-cinder-api]
                   ontroller-0 overcloud-controller-1 overcloud-controller-2 ]
                   lone [neutron-dhcp-agent]
                   ller-0 overcloud-controller-1 overcloud-controller-2 ]
                   lone [openstack-glance-api]
                   ler-0 overcloud-controller-1 overcloud-controller-2 ]
                   ch-agent-clone [neutron-openvswitch-agent]
                   d-controller-0 overcloud-controller-1 overcloud-controller-2 ]
                   openstack-nova-novncproxy-clone [openstack-nova-novncproxy]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: delay-clone [delay]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: neutron-server-clone [neutron-server]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: httpd-clone [httpd]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-ceilometer-central-clone [openstack-ceilometer-central]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-ceilometer-alarm-evaluator-clone [openstack-ceilometer-alarm-evaluator]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
Clone Set: openstack-heat-api-cfn-clone [openstack-heat-api-cfn]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
openstack-cinder-volume                 (systemd:openstack-cinder-volume):      Started overcloud-controller-0
Clone Set: openstack-nova-conductor-clone [openstack-nova-conductor]
    Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
my-ipmilan-for-controller01             (stonith:fence_ipmilan):                Started overcloud-controller-1
my-ipmilan-for-controller02             (stonith:fence_ipmilan):                Started overcloud-controller-2
my-ipmilan-for-controller03             (stonith:fence_ipmilan):                Started overcloud-controller-0
```

RH OSP 8 におけるPacemakerリソース構成



Pacemaker管理対象リソース

- OpenStack関連サービス: 29リソース
  - ほぼ全てActive/Active構成
    - 全ノードで起動し、HAProxyにより負荷分散
    - systemd経由による故障監視。プロセス故障のみ検知可能
  - cinder-volume のみ Active/Standby 構成
    - cinder-volume の制約による(現時点では stateful サービスであり Active/Active構成不可)
  - データベース、RabbitMQサービス起動完了後に起動するよう順序依存関係を制御

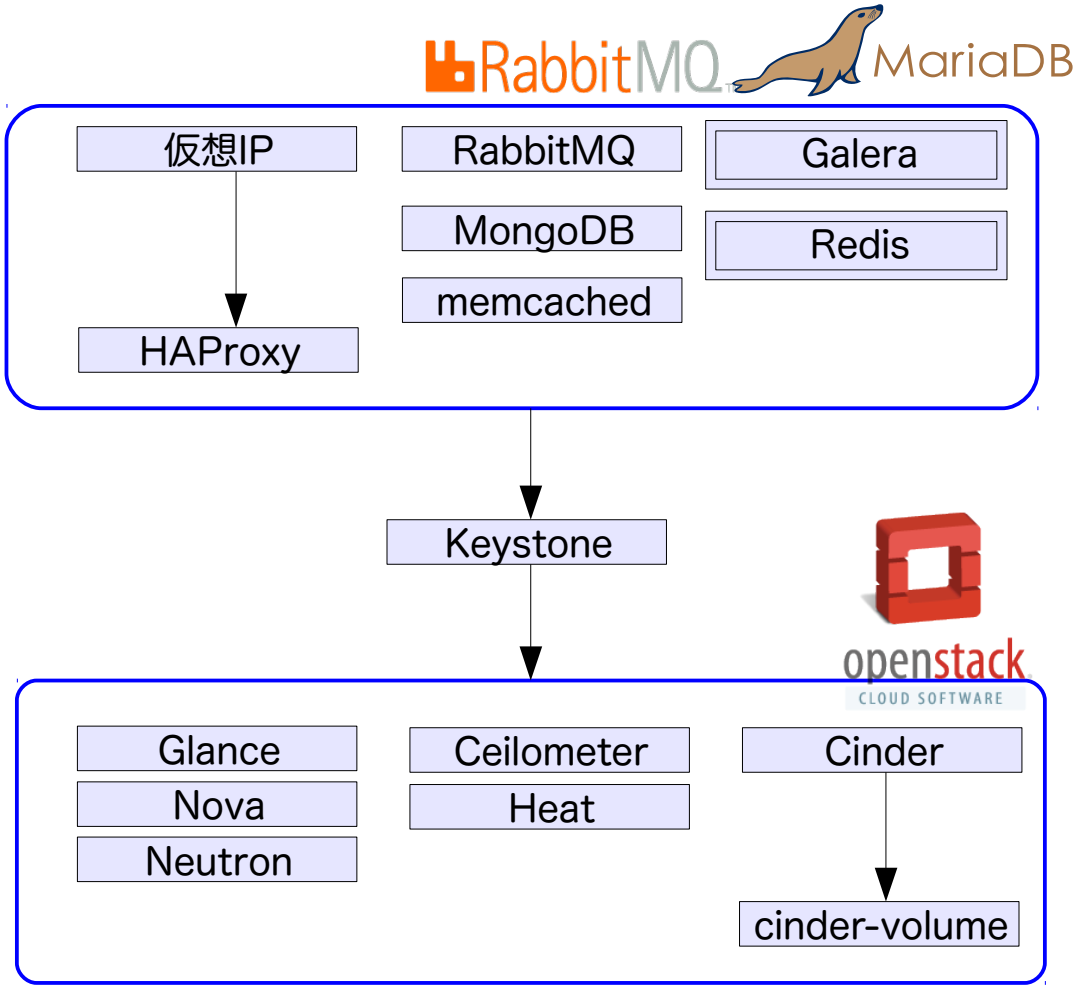
- データベース・RabbitMQサービス: 5リソース
  - Galera, RabbitMQ 独自の実装によりミラーリング対応・Active/Active構成が可能
    - ただし実際に読み書きを行うノードはHAproxy設定により１ノードのみに制限
  - クラスタとしての起動手順をリソースエージェント(OCF RA)により制御

- 仮想IP、HAProxy、他: 12リソース
  - ノード故障時の切替、負荷分散
  - STONITHエージェント等(図には未記載)

※Active/Activeリソースは3ノード全てで起動するため、crm_monで表示されるリソース数の合計は118リソースとなる。詳細内訳は省略

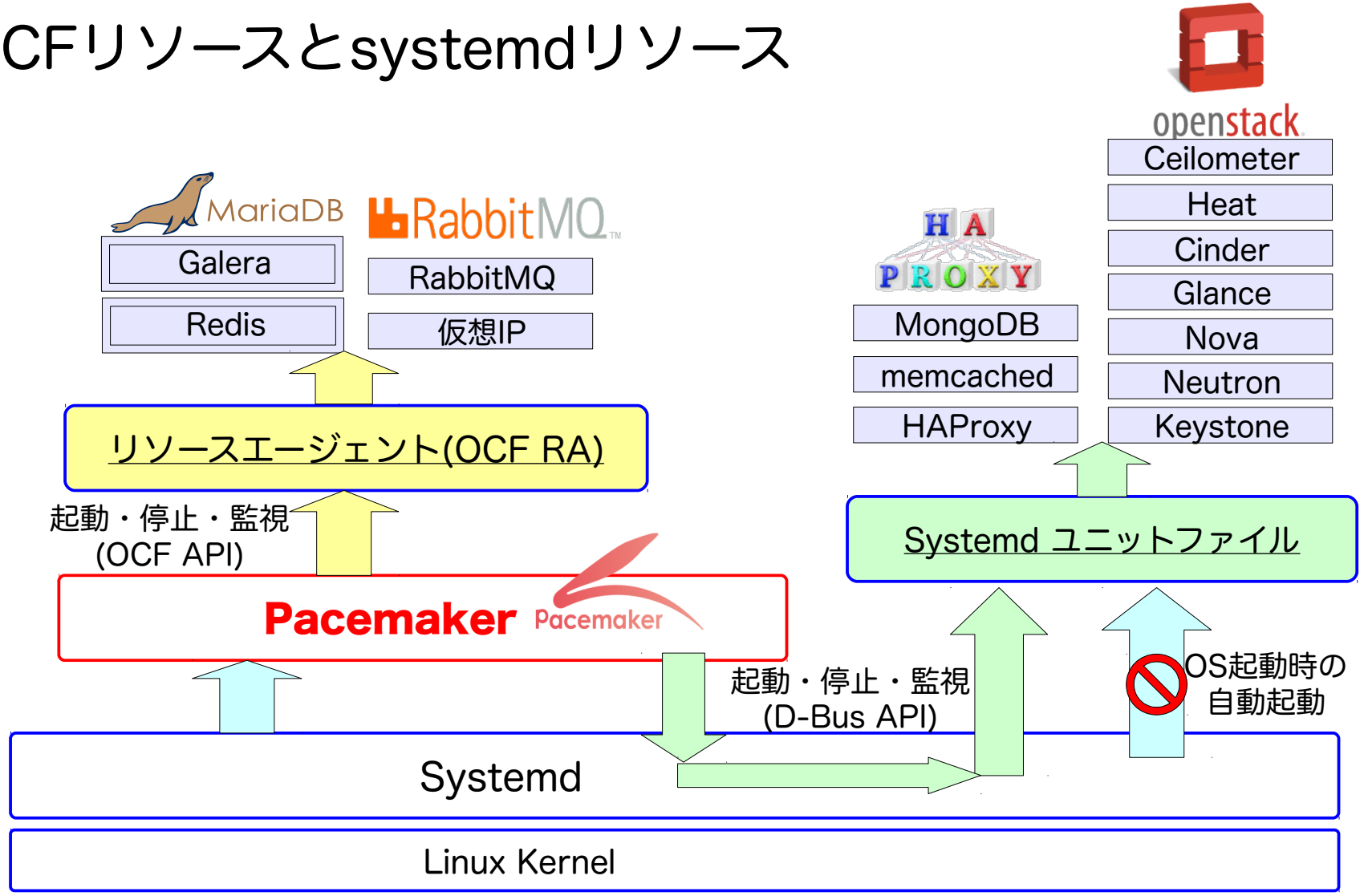
Pacemakerリソースの起動順序制御

- Pacemakerの制御により、以下の順序で起動される
  - (1)仮想IP、データベース、メッセージキューの起動
  - (2) 仮想IP起動完了後 HAProxy起動
  - (3) (1)(2)全て起動完了後、Keystone起動
  - (4) Keystone起動完了後、各OpenStackサービス起動



Pacemakerのリソース制御方法

- OCFリソースとsystemdリソース



Pacemaker管理対象外のサービス

- OpenStack関連サービス
  - swift
    - OS起動と同時に起動、systemd による監視・再起動
    - RH OSP における設計指針は不明だが、swift は他のサービスに比較して独立性が高く systemd による再起動のみで十分と判断したと推測
- Open vSwitch
  - 現状 systemd では故障検知不可能
    - systemctl status はプロセス故障時にも OK を返却する
    - したがってPacemakerのリソースに追加しても故障検知はできない
    - 意図した仕様なのか不具合なのかは不明
  - 監視が必要な場合は個別に監視処理を追加する必要あり
    - 独自にリソースエージェントの作成、運用監視システム等による監視など
- 各種OSサービス
  - ntpd, D-Bus, 他
    - 必要に応じて運用監視システム等により監視
    - 物理環境のHAクラスタにおいても通常は Pacemaker では管理していない

Pacemaker導入によるメリット

- 故障検知と自動フェイルオーバによるダウンタイム短縮
  - 切替時間目安:
    - 約70～80秒程度(負荷なし、ノード電源断～dashboard画面アクセス再開まで)
      - 内訳: Pacemaker内部処理 30～40秒程度 + OpenStackサービス再開処理 30～40秒程度
    - 故障箇所・故障タイミング・負荷等条件により大きく変わるのであくまで目安で
  - フェンシング(STONITH)によるデータ整合性の担保も含む
    - Galera, RabbitMQ 単体だけで担保できるのか?
    - 全ての故障、全てのネットワーク構成、全ての故障タイミング、ネットワーク一時分断・復活 etc...

- 起動手順・依存関係の自動化による運用手順簡易化
  - Q. そんなん systemd があるやん？
  - A. systemdだけではできないこともあるんやよ
    - Galera, RabbitMQ のクラスタ構成に必要なノード別の起動手順
    - 起動「完了」まで待ってからの次の起動
    - ノードをまたぐ順序関係

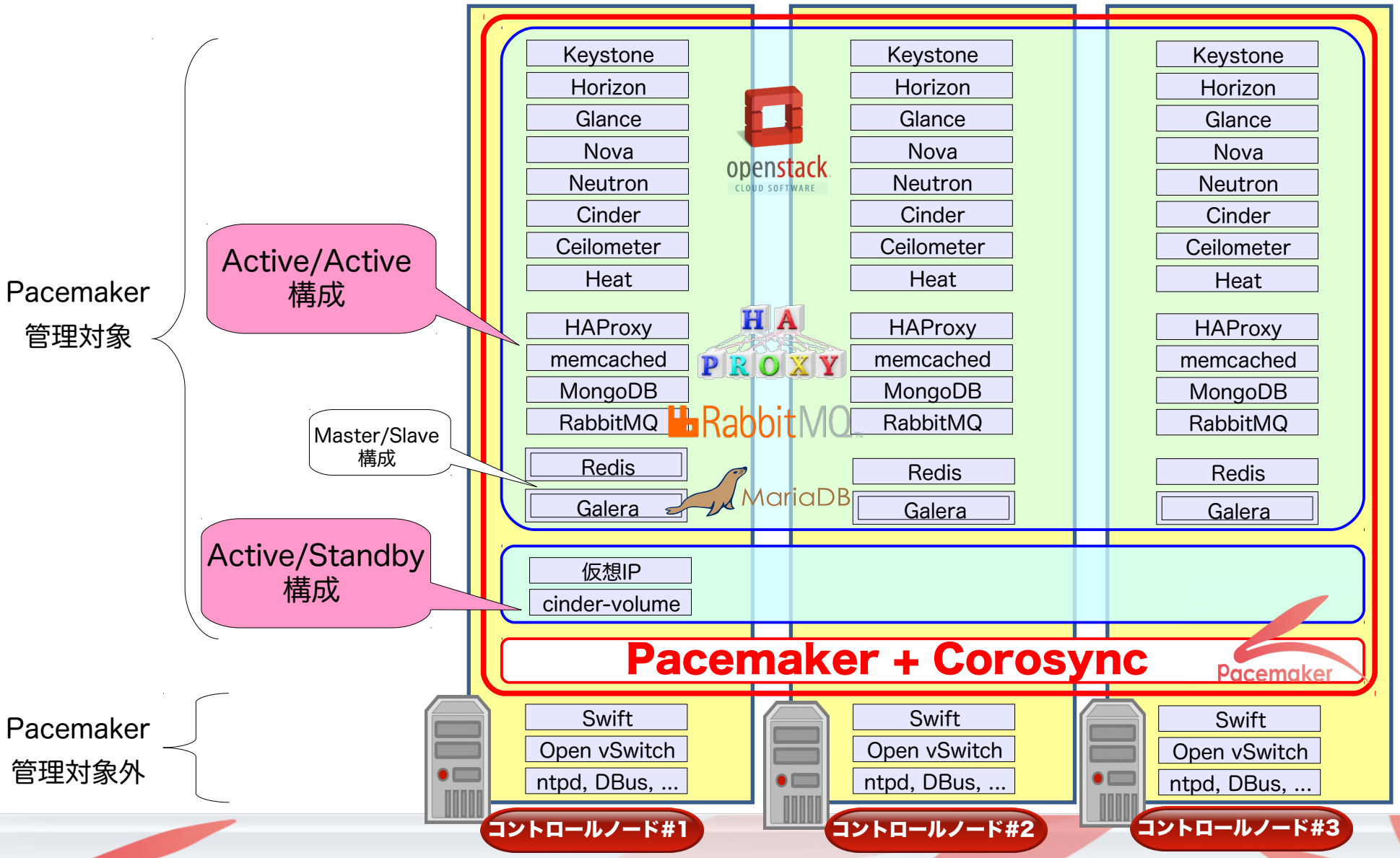
もくじ

- Pacemakerとは
- OpenStack におけるHAの必要性
- Red Hat OpenStack Platform でのHA構成例
- OpenStack HA の今後の動向

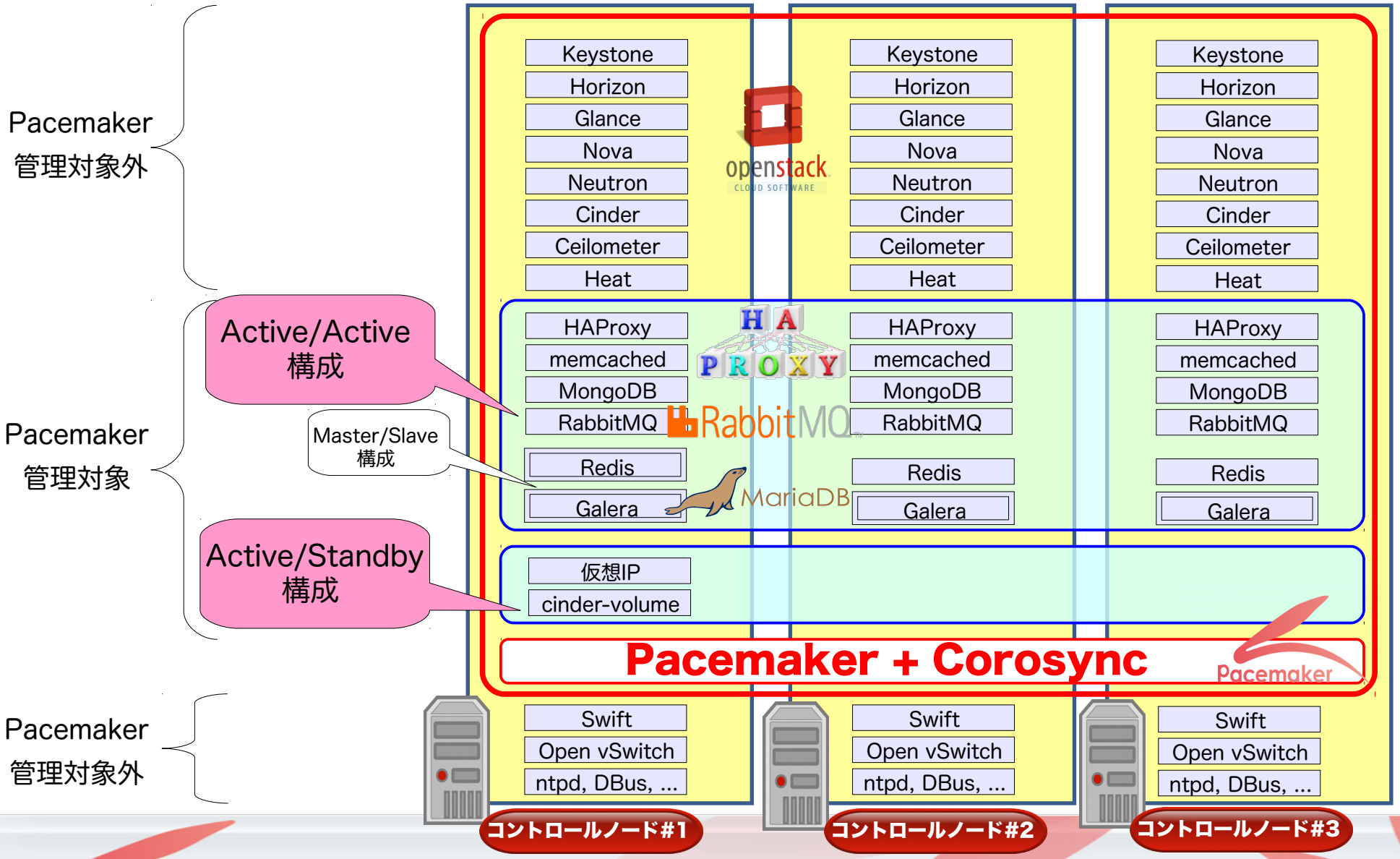
コントロールノードHAの今後

- OpenStackサービスはPacemaker管理外で良いんじゃね?
  - という方向で今開発コミュニティは動いてます

これが…


Pacemaker
管理対象
Active/Active
構成
Master/Slave
構成
Active/Standby
構成
Pacemaker
管理対象外
Keystone
Horizon
Glance
Nova
Neutron
Cinder
Ceilometer
Heat
openstack
CLOUD SOFTWARE
HAProxy
memcached
MongoDB
RabbitMQ
HA PROXY
RabbitMQ
Redis
Galera
MariaDB
仮想IP
cinder-volume
Pacemaker + Corosync
Pacemaker
Swift
Open vSwitch
ntpd, DBus, ...
コントロールノード#1
コントロールノード#2
コントロールノード#3

こうなる! (?)


Pacemaker
管理対象外
Pacemaker
管理対象
Pacemaker
管理対象外
Active/Active
構成
Master/Slave
構成
Active/Standby
構成
Keystone
Horizon
Glance
Nova
Neutron
Cinder
Ceilometer
Heat
openstack
CLOUD SOFTWARE
HAProxy
memcached
MongoDB
RabbitMQ
Redis
Galera
HA PROXY
RabbitMQ
MariaDB
仮想IP
cinder-volume
Pacemaker + Corosync
Pacemaker
Swift
Open vSwitch
ntpd, DBus, ...
コントロールノード#1
コントロールノード#2
コントロールノード#3

コントロールノードHAの今後

- OpenStackサービスはPacemaker管理外とする方向
  - ACT/SBYリソースのみ管理対象とする
    - cinder-volume、他にもあれば (neutron-lbaas-agent ?)
    - cinder-volume のACT/ACT対応は現在議論中、対応完了すれば管理外とする
  - ただし全てのOpenStackサービスは起動順序に依存しないことが前提
    - データベースやメッセージキュー等の依存サービスがダウンしても、フェイルオーバ後に自動的に復旧できること
    - 開発コミュニティにて検証中
  - RH OSPでは…
    - Mitakaベース(RH OSP 9)では keystone が管理外となっている
    - おそらく Newtonベースの RH OSP 以降で大きく変更されるのでは

コントロールノードHAの今後

- OpenStackサービスはPacemaker管理外とする方向

- 開発コミュニティでは異論もあり、議論継続中

  - 管理すべきだよ派
    - systemdのプロセス監視だけでは不十分、OCF RA等でサービス監視もやりたい
    - OpenStackサービスはSTONITH無くてホントに大丈夫なん?

  - 管理しなくてもいいよ派
    - サービス監視は運用監視ツール(Nagiosとか)でやればいいんじゃねーの
    - 起動順序の依存関係はOpenStackサービス側で解決すればいいよね
    - それよりもHeatテンプレートの維持管理とかバージョンアップを楽にしようよ

コンピュートノードHAの今後

- 現状以下の３方式があり、これから統合されていく予定
  - NovaEvacuate RA (Red Hat, SUSE)
    - 商用サポート提供済み、ただし故障対応範囲に制約あり
  - Masakari (NTT)
    - 故障対応範囲が充実、ただしPacemakerとの連携に向上の余地あり
  - Mistral (Intel)
    - OpenStack全体のアーキテクチャとして望ましい、ただし現状試験実装レベル
- 現時点での見込み
  - Mistral のワークフローベースにMasakariの機能を統合していく方針

	NovaEvacuate RA	Masakari	Mistral
HA対象インスタンスのタグ管理	○	○	○
故障時アクションのカスタマイズ	−	−	予定あり
自己監視	○	○	実装中
インスタンス故障監視	−	○	予定あり
force_down API対応	○	−	予定あり
nova-compute故障対応	−	○	予定あり
並行実行	−	−	○

出典: http://aspiers.github.io/openstack-summit-2016-austin-compute-ha/

OpenStack HA コミュニティの活動

- IRCミーティング
  - 毎週月曜
  - https://wiki.openstack.org/wiki/Meetings/HATeamMeeting
- ドキュメント改善
  - 公式HAドキュメントのメンテナがAndrew Beekhof 氏(Pacemaker開発者)に
    - http://docs.openstack.org/ha-guide/
    - 現在 (まさに今!) 随時更新中
  - プロジェクト間活動に向けたユーザストーリー・仕様書

おわり

- Pacemakerをこれからもよろしくお願いします!