Pacemakerでお手軽Dockerクラスタリング！

2018年 8月 4日
OSC2018 Kyoto

Linux-HA Japan
竹下　雄大

本日の内容

- Pacemakerってなに？
- Pacemakerでお手軽Dockerクラスタリング！

Pacemakerってなに？

Pacemakerってなに？

Pacemakerはオープンソースの
HAクラスタソフトです

Pacemakerってなに？

High Availability = 高可用性

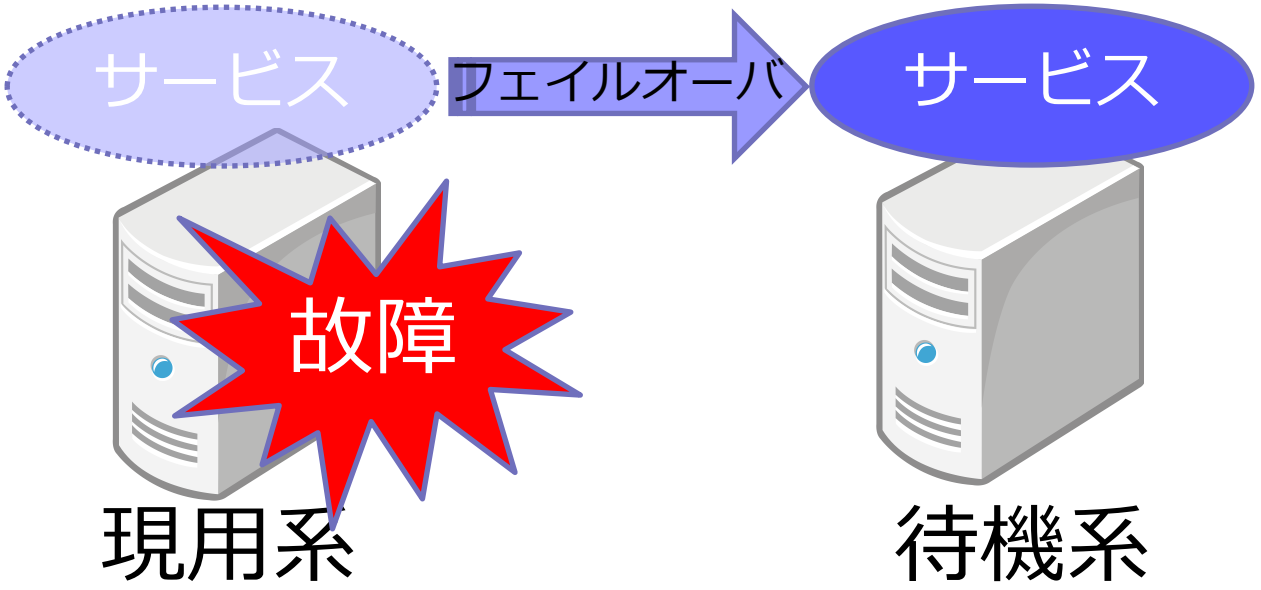
つまり　　サービス継続性

一台のコンピュータでは得られない高い
信頼性を得るために、
複数のコンピュータを結合(クラスタ化)し、
ひとまとまりとする…

ためのソフトウェアです

Pacemakerってなに？

HAクラスタを導入すると、
故障で現用系でサービスが運用できなくなったときに、
自動で待機系でサービスを起動させます
　→このことを**「フェイルオーバ」**と言います

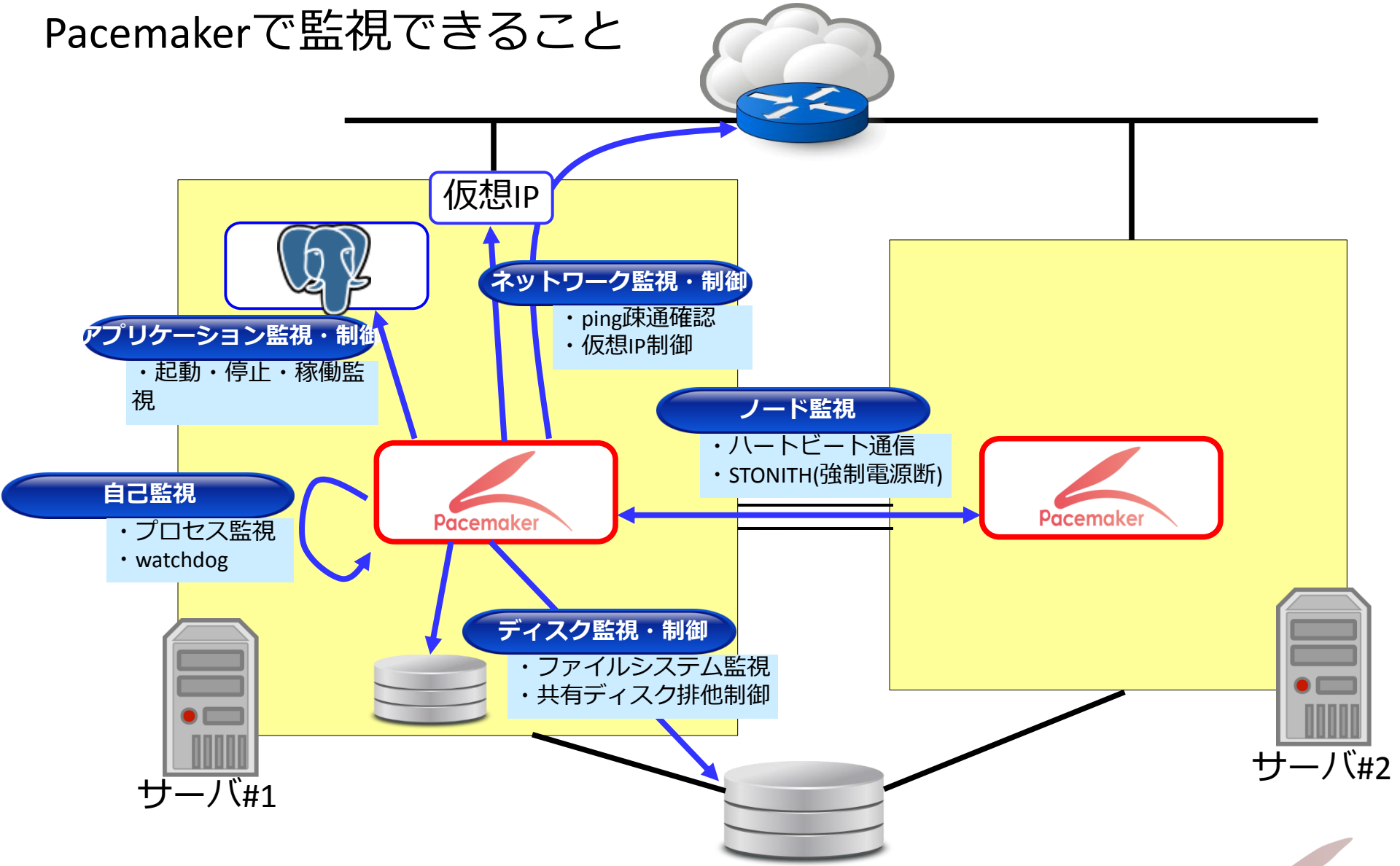


Pacemakerってなに？

は
このHAクラスタソフトとして
実績のある「**Heartbeat**」と
呼ばれていたソフトの後継です

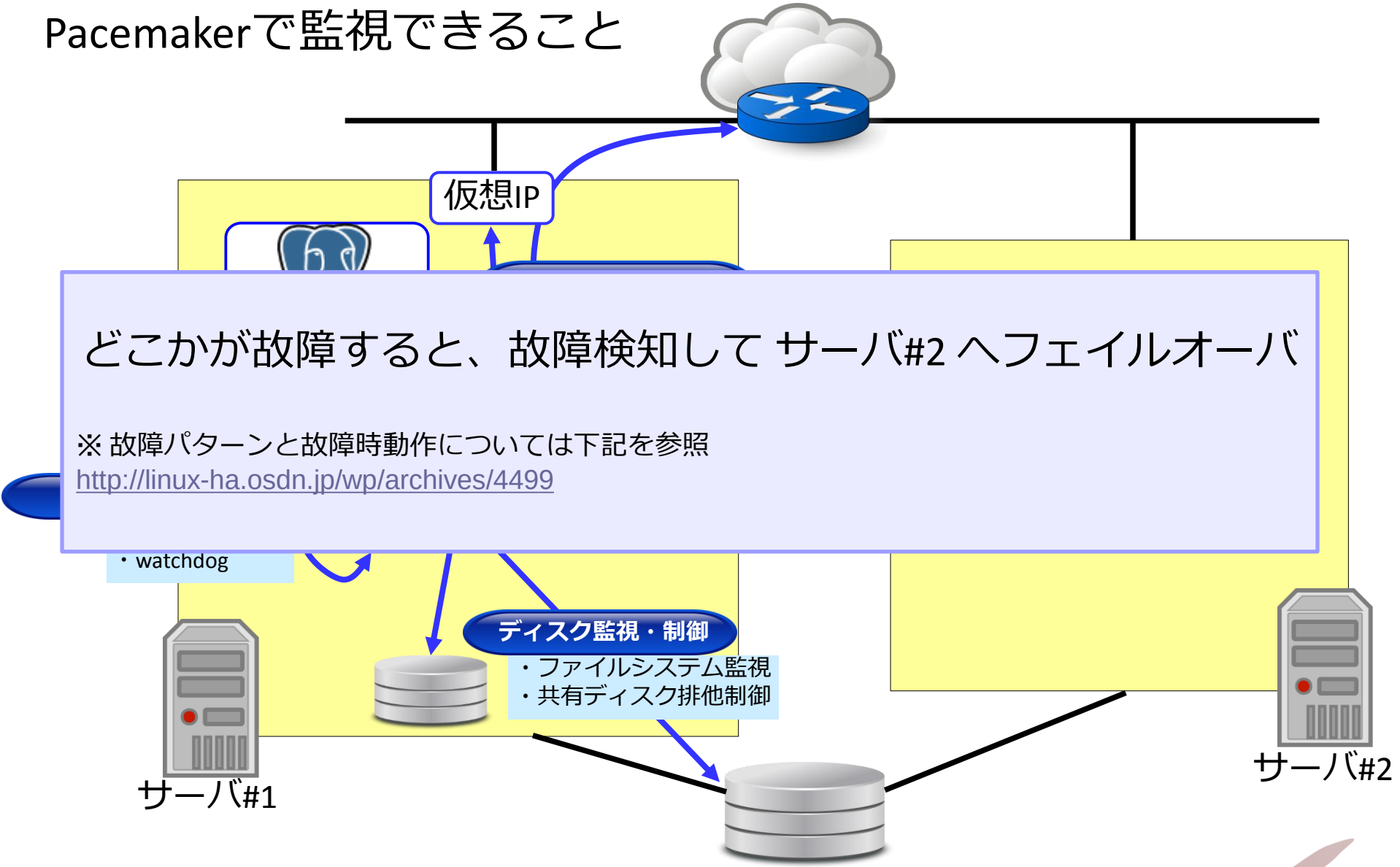
Pacemakerってなに？

Pacemakerで監視できること



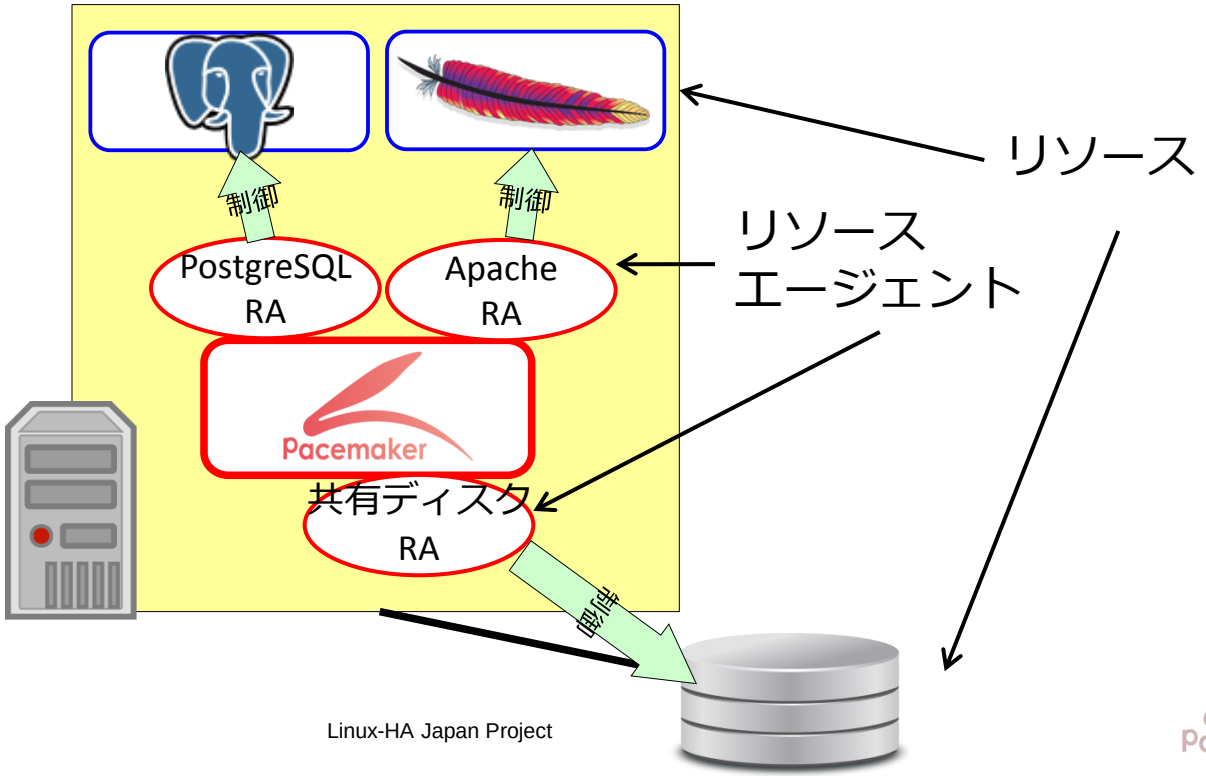
Pacemakerってなに？

Pacemakerで監視できること



Pacemakerってなに？

- Pacemakerが起動/停止/監視を制御する対象を**リソース**と呼ぶ
  - 例：Apache、PostgreSQL、共有ディスク、仮想IPアドレス…
- リソースの制御は**リソースエージェント(RA)**を介して行う
  - RAが各リソースの操作方法の違いをラップし、Pacemakerで制御できるようにしている
  - 多くはシェルスクリプト
  - 自作のアプリも RA を作成する事で制御可能！
    - OCFリソースエージェント開発者ガイド：http://linux-ha.osdn.jp/wp/archives/4328



Pacemakerの主な戦場

- ベアメタル環境
- 仮想環境
  - KVM、VMwareなどの非クラウド

昨今のトレンド

- クラウド
- コンテナ (Docker)

昨今のトレンド

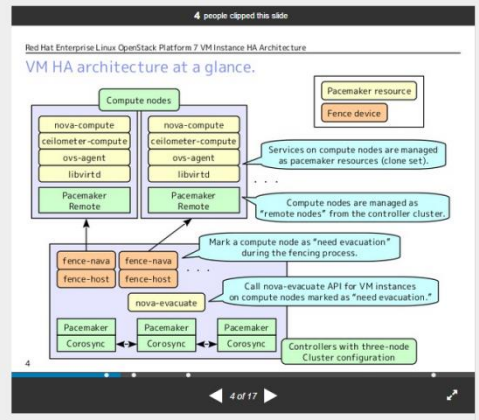
- クラウド
- コンテナ (Docker)

クラウドやコンテナでは
Pacemakerの存在感は薄い？

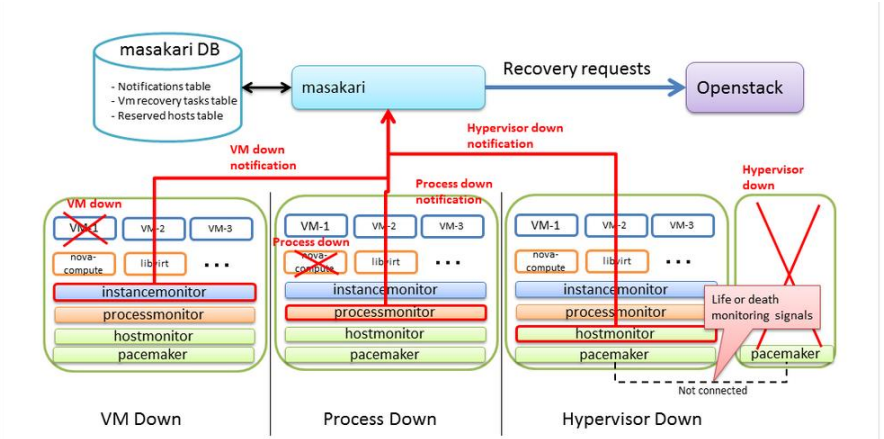
クラウドのHA

- OpenStack
  - ControllerノードのHAはPacemaker
  - ComputeノードのHAは、コミュニティで方式検討中
    - the mistral-based POC (Intel)
    - Masakari (NTT SIC)
    - OCF RAs (Red Hat, SUSE)

Pacemakerを利用



出展：https://www.slideshare.net/enakai/red-hat-enterprise-linux-openstack-platform-7-vm-instance-ha-architecture



出展：https://github.com/ntt-sic/masakari

- パブリッククラウド
  - インスタンスの冗長性はクラウド事業社が担保
    - インスタンス内で稼働するミドルウェア・アプリケーションはユーザがHA化
  - PacemakerでもインスタンスのHA可

コンテナ(Docker)のHA

- サービスを商用環境で運用する際にはHAが非常に重要
  - Dockerコンテナでも同様

- DockerコンテナのHAはオーケストレーションツールの利用が主流
  - Kubernetes
    - Docker Swarmなどもあるが、デファクトスタンダードの地位を確立

  - OpenShift Origin / OpenShift Container Platform (商用)
    - KubernatesをベースにCI/CD環境を付け加えたもの
      - Dockerイメージ管理
      - Docker コンテナ + アプリケーションの自動ビルド

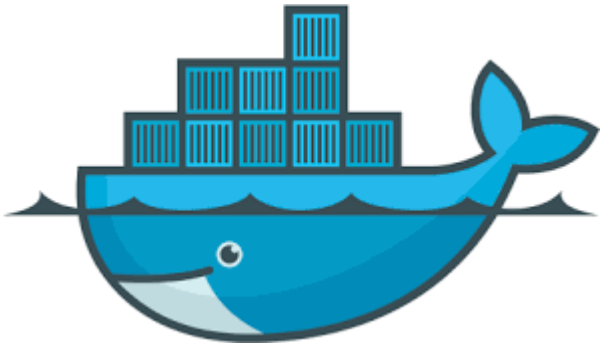
オーケストレーションツールの簡便さは開発者・運用者のもの

インフラエンジニアには極めて高度なスキルが求められる

- コンテナだけでなく、KVS、Ansible、CNIなどの関連技術への理解が必要
- 開発が早く、学んだ技術がすぐに陳腐化
- 複雑なアーキテクチャ、困難なトラブルシューティング
- Kubernetes on OpenStack…(›´ω`‹)

(※)　個人的な所感です

Pacemakerでお手軽
Dockerクラスタリング！



Pacemakerでお手軽Dockerクラスタリング！

- PacemakerでDockerコンテナ クラスタリングできます！
  - Docker RA
    - OSC2017 Tokyo/Springで紹介
    - http://linux-ha.osdn.jp/wp/archives/4601
  - **Bundle**
    - 今日のテーマ
- PacemakerによるDockerコンテナ HAの特徴
  - NativeなDockerの機能だけでクラスタ化可能
  - (オーケストレーションツールよりは) 簡単なアーキテクチャ
  - まだほとんど誰もやっていないので、今ならパイオニアになれる！

bundleとは

- 一般用語での意味
  - 英辞郎 on the WEB(https://eow.alc.co.jp/)より

bundle
【自動】
〔足早に・急いで・さっさと〕立ち去る、出て行く
〔急いで・さっさと・素早く〕荷物をまとめる
【他動】
～を束にする、束ねる、包む、くくる、〔荷物を〕まとめる
〔商品を〕バンドリングする
〔複数の商品を〕セット売りする、1セットにして販売する
【名】
束、束状構造
塊、一団、一括、一つにまとめた物
包み、小包
〈俗〉札束、大金
《植物》維管束
〈米俗〉かわい子ちゃん、いかす女

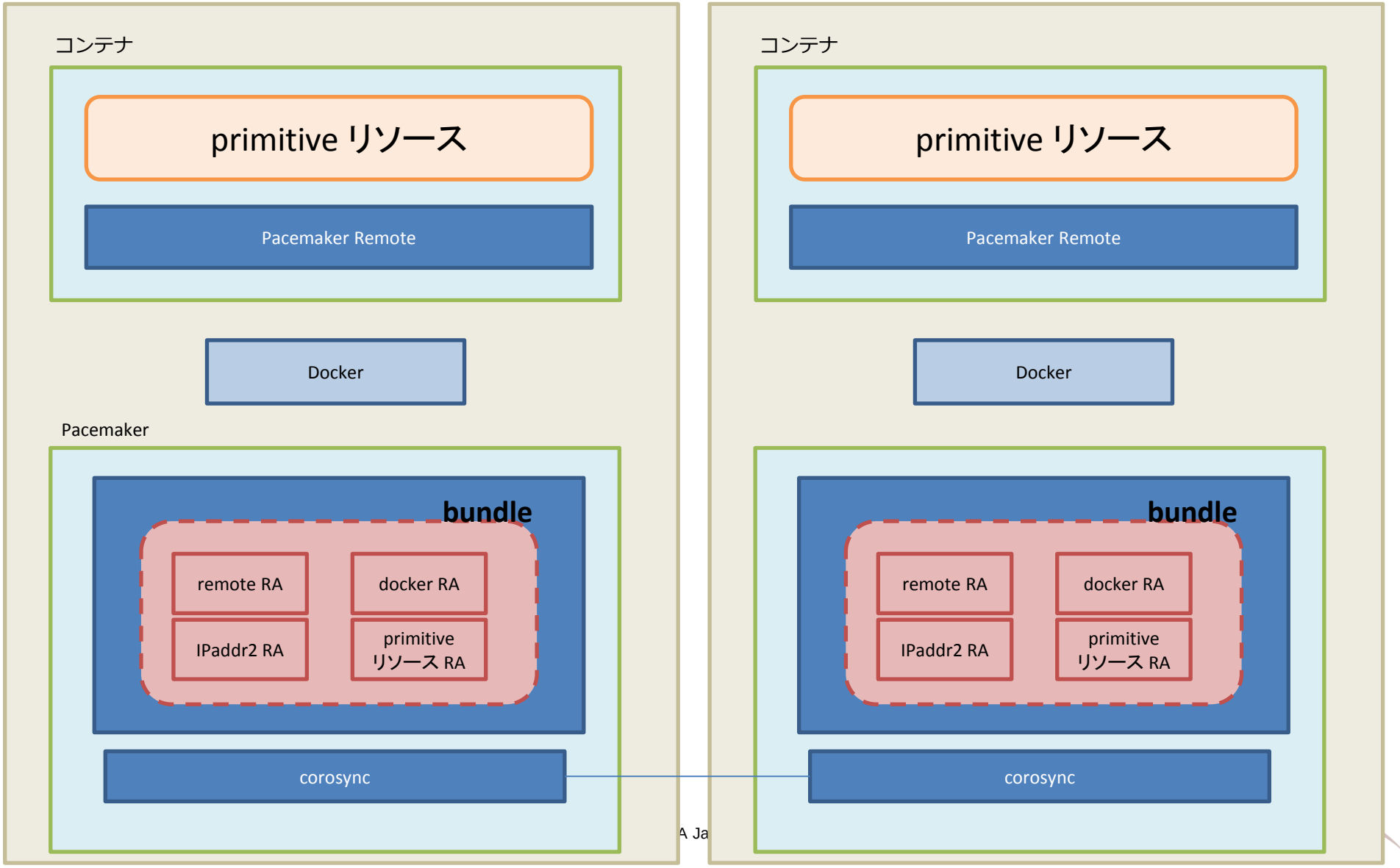
bundleとは

- Pacemakerでの意味
  - 隔離された環境とインフラをまとめる特別なsyntax
    - 隔離された環境：Dockerコンテナ (Pacemaker 1.1.17 現在)
    - インフラ：ミドルウェア、ネットワーク
      - http://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-resource-bundle.html
  - 具体的には以下をまとめたもの
    - docker RA : Dockerコンテナを管理するRA
    - IPaddr2 RA : 仮想IPを管理するRA
    - remote RA : Dockerコンテナをリモートノードとして管理するRA
    - <primitiveリソース> RA : コンテナ内で動作するアプリケーションのRA

Pacemake-1.1.17以降で利用可能な、Dockerコンテナを管理するための特別な機能

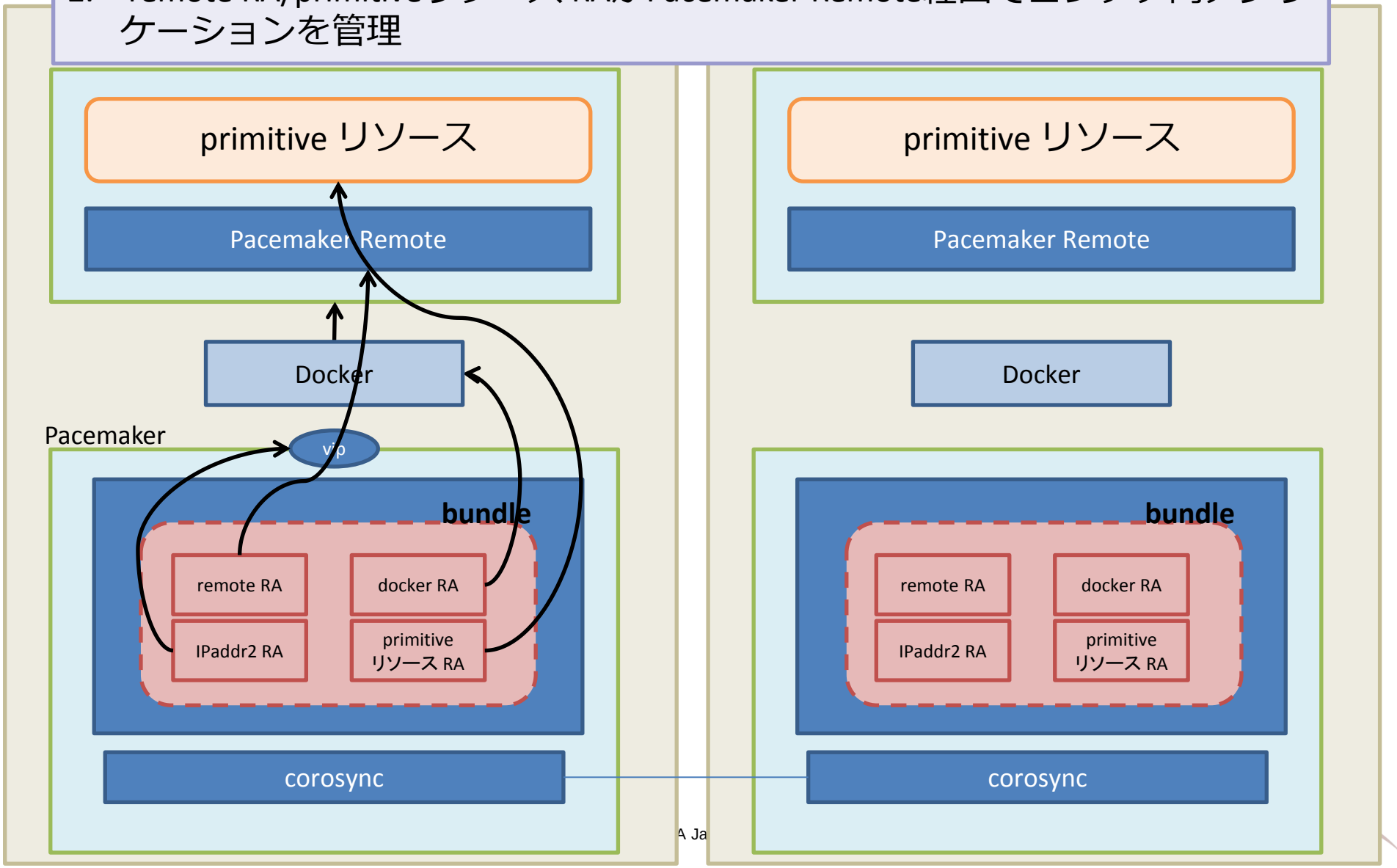
Pacemaker 1.1.18以降ではrktも管理可能

bundleの構成(イメージ)

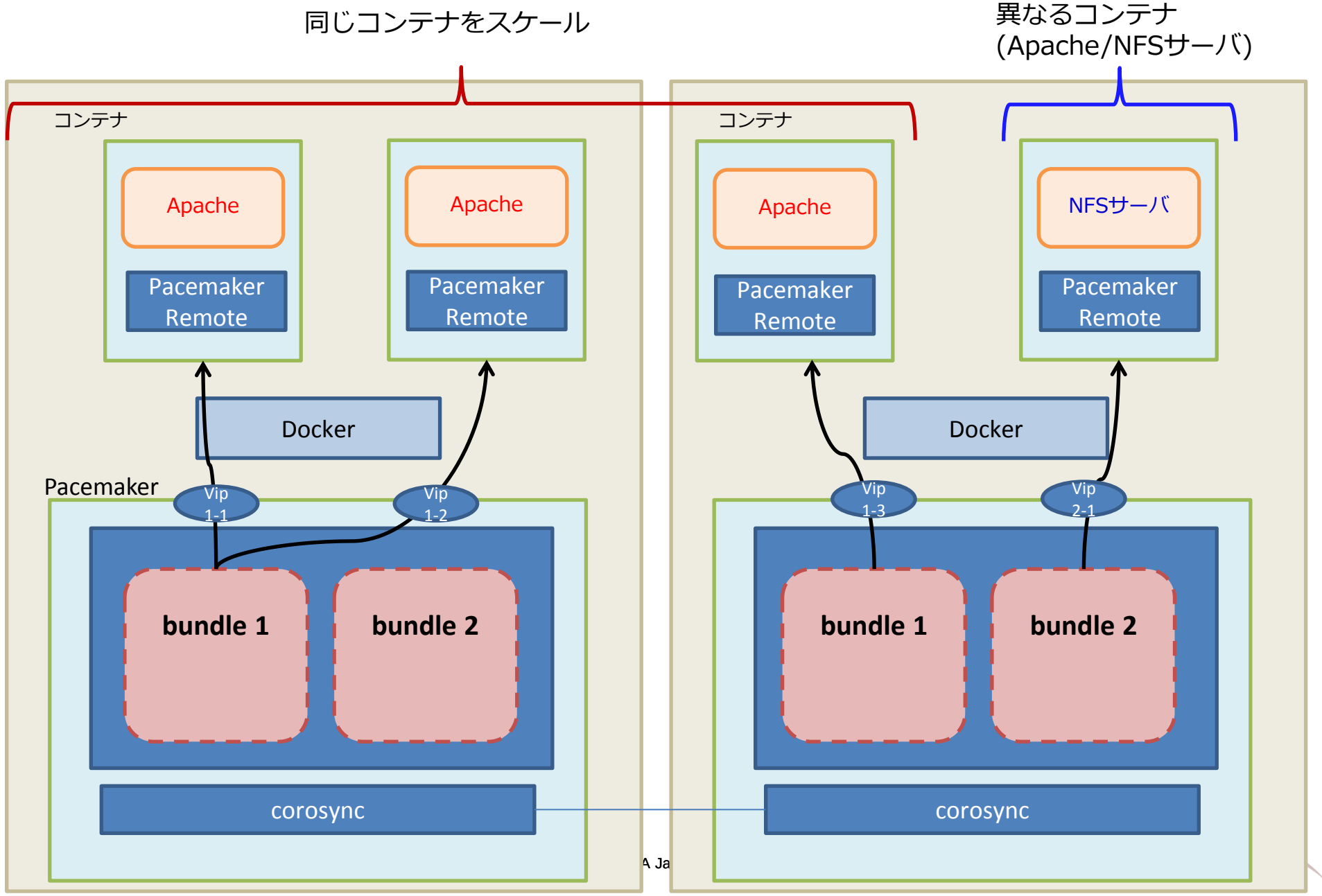


bundleの構成(イメージ)

1. Docker RAがコンテナを作成
2. remote RA/primitiveリソース RAがPacemaker Remote経由でコンテナ内アプリケーションを管理



bundleの構成(イメージ)



bundleの構成(xml)

以下の要素から構成される
- bundle : 必須
- docker : 必須
- network : 必須
- storage : オプション
- primitive : オプション

≠ RAのパラメータ

- 一つのbundleリソース定義
- 複数の異なるbundleを定義することも可能

```
<bundle id="httpd-bundle">
  <docker image="pcmktest:http" replicas="3" replicas-per-host= "2" options="--log-driver=journald"/>
  <network ip-range-start="192.168.0.200" host-interface="ens3" host-netmask="24">
    <port-mapping id="httpd-port" port="80"/>
  </network>
  <storage>
    <storage-mapping id="httpd-root"
      source-dir-root="/var/local/containers"
      target-dir="/var/www/html"
      options="rw"/>
    <storage-mapping id="httpd-logs"
      source-dir-root="/var/log/pacemaker/bundles"
      target-dir="/etc/httpd/logs"
      options="rw"/>
  </storage>
  <primitive class="ocf" id="httpd" provider="heartbeat" type="apache">
      <operations>
        <op name="start" interval="0s" timeout="60s" on-fail="restart" id="httpd-start-0s"/>
        <op name="monitor" interval="10s" timeout="60s" on-fail="restart" id="httpd-monitor-10s"/>
        <op name="stop" interval="0s" timeout="60s" on-fail="block" id="httpd-stop-0s"/>
      </operations>
  </primitive>
</bundle>
```

ip-range-startを起点に、コンテナ毎にVIPが付与される

storage-mapping要素は複数設定可能

Primitiveリソースはひとつのみ

bundleの構成(xml)

以下の要素から構成される
- bundle : 必須
- docker : 必須
- network : 必須
- storage : オプション
- primitive : オプション

≠ RAのパラメータ

```
<docker image="pcmktest:http" replicas= “3" replicas-per-host= “2" options="--log-driver=journald"/>
```

- コンテナの概要を記述
  - コンテナの数
  - 1ノード上で起動するコンテナの最大数
  - docker image名
  - docker runのオプション、コマンド

bundleの構成(xml)

以下の要素から構成される
- bundle : 必須
- docker : 必須
- network : 必須
- storage : オプション
- primitive : オプション

≠ RAのパラメータ

```
<network ip-range-start="192.168.0.200" host-interface="ens3" host-netmask="24">
 <port-mapping id="httpd-port" port="80"/>
</network>
```

- コンテナとホストのネットワーク経路を指定
  - コンテナと紐付くVIP
    - 各コンテナと1 : 1に対応
    - ip-range-startから順に付与
  - publish port

bundleの構成(xml)

以下の要素から構成される
- bundle : 必須
- docker : 必須
- network : 必須
- storage : オプション
- primitive : オプション

≠ RAのパラメータ

- コンテナにマウントするvolumeを指定
  - 複数のvolumeが存在する場合は、各volume毎にstorage-mapping要素を記述

```
<storage>
 <storage-mapping id="httpd-root"
  source-dir-root="/var/local/containers"
  target-dir="/var/www/html"
  options="rw"/>
 <storage-mapping id="httpd-logs"
  source-dir-root="/var/log/pacemaker/bundles"
  target-dir="/etc/httpd/logs"
  options="rw"/>
</storage>
```

bundleの構成(xml)

以下の要素から構成される
- bundle : 必須
- docker : 必須
- network : 必須
- storage : オプション
- primitive : オプション

≠ RAのパラメータ

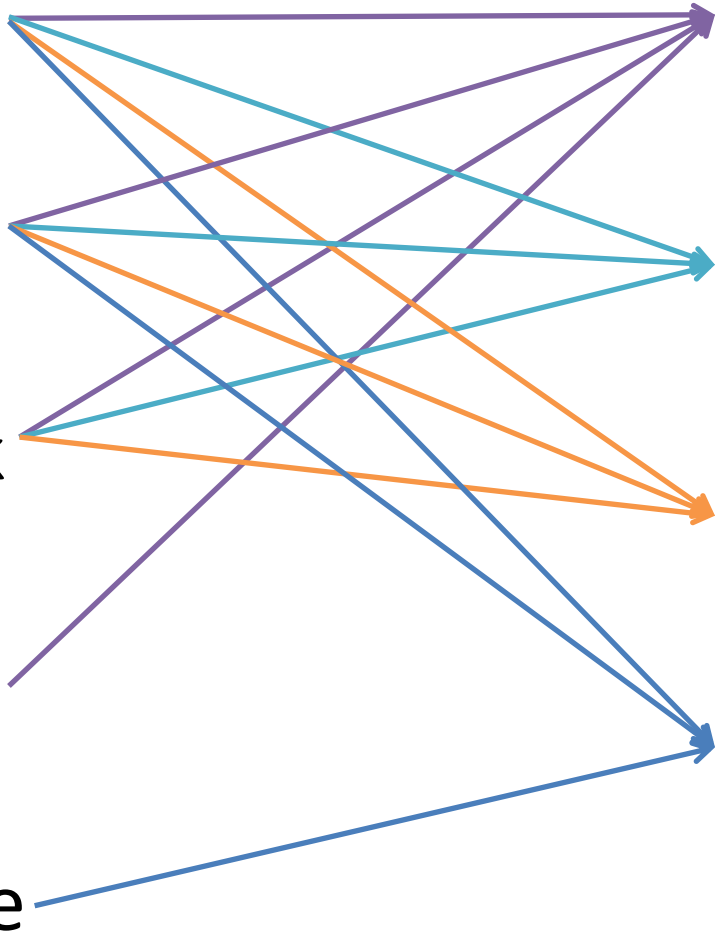
- コンテナ内で起動するprimitiveリソースを指定
  - 1つのbundleには、1つのprimitiveリソースのみ
  - 通常のprimitiveリソース定義と同じ内容

```
<primitive class="ocf" id="httpd" provider="heartbeat" type="apache">
    <operations>
      <op name="start" interval="0s" timeout="60s" on-fail="restart" id="httpd-start-0s"/>
      <op name="monitor" interval="10s" timeout="60s" on-fail="restart" id="httpd-monitor-10s"/>
      <op name="stop" interval="0s" timeout="60s" on-fail="block" id="httpd-stop-0s"/>
    </operations>
 </primitive>
```

bundle要素とRAの関係

各要素からRAのパラメータが生成される

- bundle
- docker
- network
- storage
- primitive

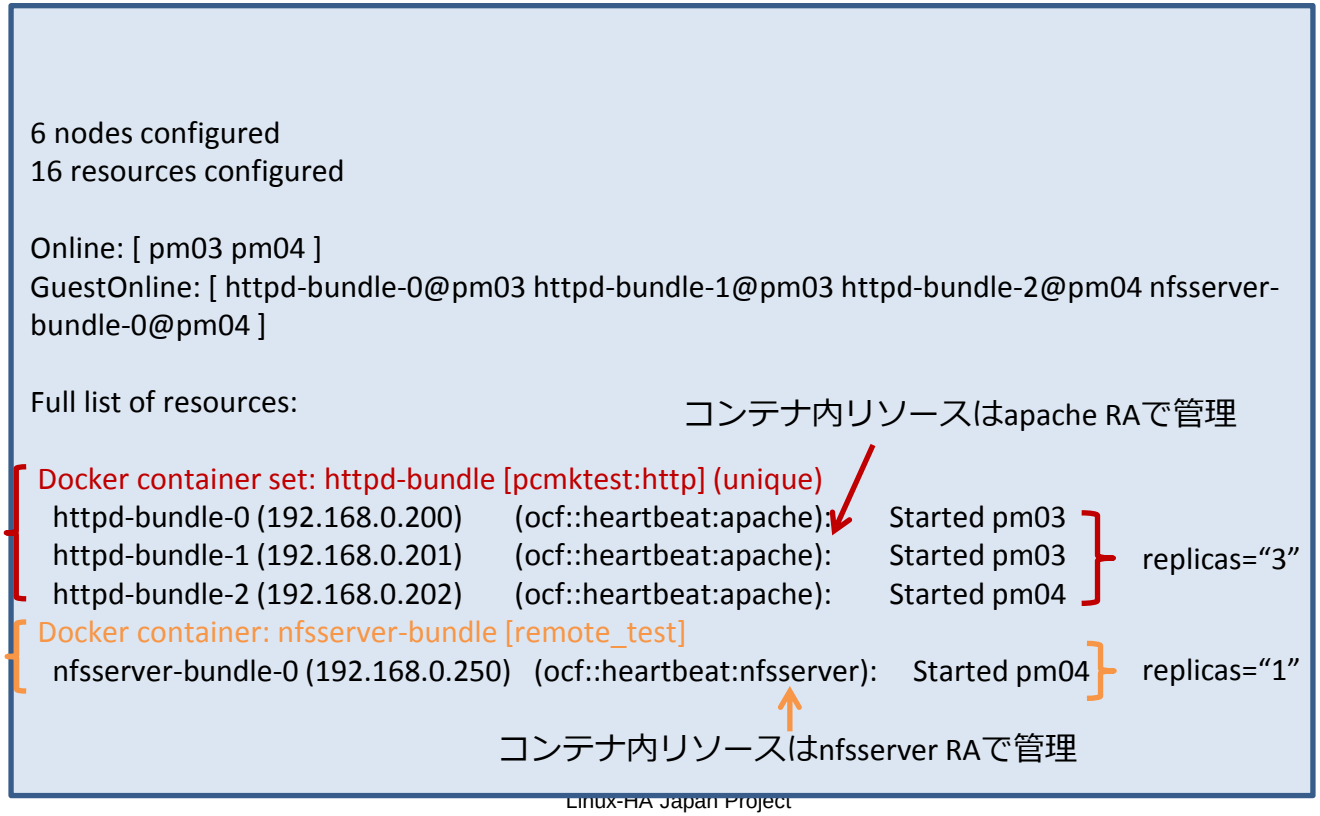


- docker RA
- IPaddr2 RA
- remote RA
- *<primitive>* RA

bundleの起動

- cibadminまたはpcsによるリソース管理
  - Pacemaker-1.1.17-1.1ではcrmshによるリソース管理不可
  - 具体的な設定例、作成したXMLの反映方法などは下記を参照
    - https://wiki.clusterlabs.org/wiki/Bundle_Walk-Through
    - https://www.clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-resource-bundle.html

ApacheとNFSサーバを bundle で起動した例



作成されるDockerコンテナ

- docker ps
  - 詳細はdocker inspect <CONTAINER ID>

```
# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                                    NAMES
f0a521f0477a   pcmktest:http      "/usr/sbin/pacemak..."   27 seconds ago   Up 22 seconds   192.168.0.201:80->80/tcp, 192.168.0.201:3121->3121/tcp   httpd-bundle-docker-1
867ecaae027f   pcmktest:http      "/usr/sbin/pacemak..."   33 seconds ago   Up 30 seconds   192.168.0.200:80->80/tcp, 192.168.0.200:3121->3121/tcp   httpd-bundle-docker-0
```

VIPは自動的に連番が振られる　　コンテナ名は”<bundle id>-docker-連番”

- bundleにより、コンテナと1 : 1に対応するVIPが割り当てられるため、クライアントはVIPを通して、任意のコンテナにアクセス可能
  - VIPとコンテナのIPはdockerdによってルーティングされる
- 一方、オーケストレータのように内部LBは持たないため、負荷分散用途では別途LBが必要
- bundleで起動するコンテナにはPacemaker Remoteが必要
  - 典型的には pacemaker_remoted + 管理するアプリケーション

作成されるDockerコンテナ

- コンテナ内のプロセス
  - COMMANDがpacemaker_remoted(デフォルト)の場合、PID 1はpcmk-init
    - Primitiveリソースの故障 ≠ コンテナの故障

```
# docker exec -it httpd-bundle-docker-0 ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  75504  2516 ?        Ss   00:46   0:00 pcmk-init
root         5  0.0  0.0  77792  3752 ?        Ss   00:46   0:00 /usr/sbin/pacemaker_remoted
root        44  0.0  0.0 221952  3484 ?        Ss   00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
apache      49  0.0  0.0 222088  3724 ?        S    00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
apache      50  0.0  0.0 222088  3724 ?        S    00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
apache      51  0.0  0.0 222088  3724 ?        S    00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
apache      52  0.0  0.0 222088  3724 ?        S    00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
apache      53  0.0  0.0 222088  3724 ?        S    00:46   0:00 /sbin/httpd -DSTATUS -f /etc/httpd/conf/httpd.conf -c PidFile /var/run//httpd.pid
root       576  0.0  0.0  47452  1676 pts/0    Rs+  00:53   0:00 ps aux
```

- オーケストレーションツールではコンテナ内のアプリケーションが故障するとコンテナが停止(再起動)するが、bundleではコンテナは停止しない
  - コンテナ内のアプリケーションが再起動する

bundle vs Docker RA

	bundle	Docker RA(単体)
制御できるコンテナ	Pacemaker Remoteコンテナ	制限なし
コンテナで起動できるリソース	RAで管理できるもの (lsb/systemdを含む)	制限なし (コンテナに依存)
スケーラビリティ	大 1500まで起動できた報告有	小 Dockerリソースのclone化でコンテナのスケーラビリティはある程度確保できるが、IPやvolumeの動的な割り当てが困難
監視レベル	RAによるサービス監視	ワンライナー、またはHEALTHCHECKによる簡易チェック
M/Sリソース管理	可能	不可

- Docker RA： コンテナを管理
- bundle：コンテナ＋インフラ(ネットワーク、volume、リソース)を管理

bundleを利用するモチベーション

オーケストレーションツールではなく、
bundleを使うモチベーションは？

bundleを利用するモチベーション

- オーケストレーションツールの方が適合するケース
  - 以下のようなキーワードを目的にコンテナを運用する場合
    - DevOps, CI/CD
    - マイクロサービス
    - SoE

- bundleの方が適合するケース
  - 高集約率を目的に従来システム(SoR)をコンテナに移行する場合
    - 高い信頼性が求められる
    - リリース頻度が低い
      - Ex) 基幹系システム
  - 物理サーバ上のDBとコンテナを連携する場合
    - コンテナとDB(物理)をPacemakerだけで管理可能
  - インフラエンジニアの負担を軽減したい場合

※ ただし、Pacemaker 1.1.17-1.1 ではTechnology Preview 扱いです

bundleの具体的な利用シーン

- bundleは Red Hat社でもTechnology Previewであることから、具体的な利用シーンはまだ少ない (2018.8現在)

- Red Hat OpenStack Platform 12 以降で利用する場合に限り フルサポート
  - https://access.redhat.com/articles/3388681
  - https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/12/html/understanding_red_hat_openstack_platform_high_availability/pacemaker#pacemaker-services

  - Controller ノードで起動する各種ミドルウェアをコンテナ化し、bundleで管理
    - HAProxy
    - RabbitMQ
    - Galera
    - redis

さいごに

Linux-HA Japan URL

http://linux-ha.osdn.jp/

http://osdn.jp/projects/linux-ha/

Pacemaker関連の最新情報を日本語で発信

Pacemakerのダウンロードもこちらからどうぞ
(インストールが楽なリポジトリパッケージを公開しています)

さいごに

日本におけるHAクラスタについての活発な意見交換の場として「Linux-HA Japan日本語メーリングリスト」 も開設しています。

Linux-HA-Japan MLでは、 Pacemaker、 Heartbeat３、 Corosync DRBDなど、 HAクラスタに関連する話題は歓迎！

- ML登録用URL

  http://linux-ha.osdn.jp/
  の「メーリングリスト」をクリック

- MLアドレス

  linux-ha-japan@lists.osdn.me

※スパム防止のために、登録者以外の投稿は許可制です

ご清聴ありがとうございました。

Linux-HA Japan

検索