プロキューブITサービス基盤

CentOS+Docker+Consul
マルチベンダクラウド、広域冗長、低速フェールオーバ

株式会社プロキューブ
中川路　充

冗長化→サービスディスカバリ

CentOS のクラスタソフト pacemaker の問題点

- 実装概念の論理構造が複雑な上に、OCFスクリプトが詳細にログを出さないため、障害発生時にログを見ても pacemaker が何を考えていたのかわかりにくい
- スプリットブレインになりやすい
- AWSのネットワークなど、IP take over が使えない場合に対応できない
- プロキューブのITサービスでは、IP take over + ウォームスタンバイ級の可用性は不要である

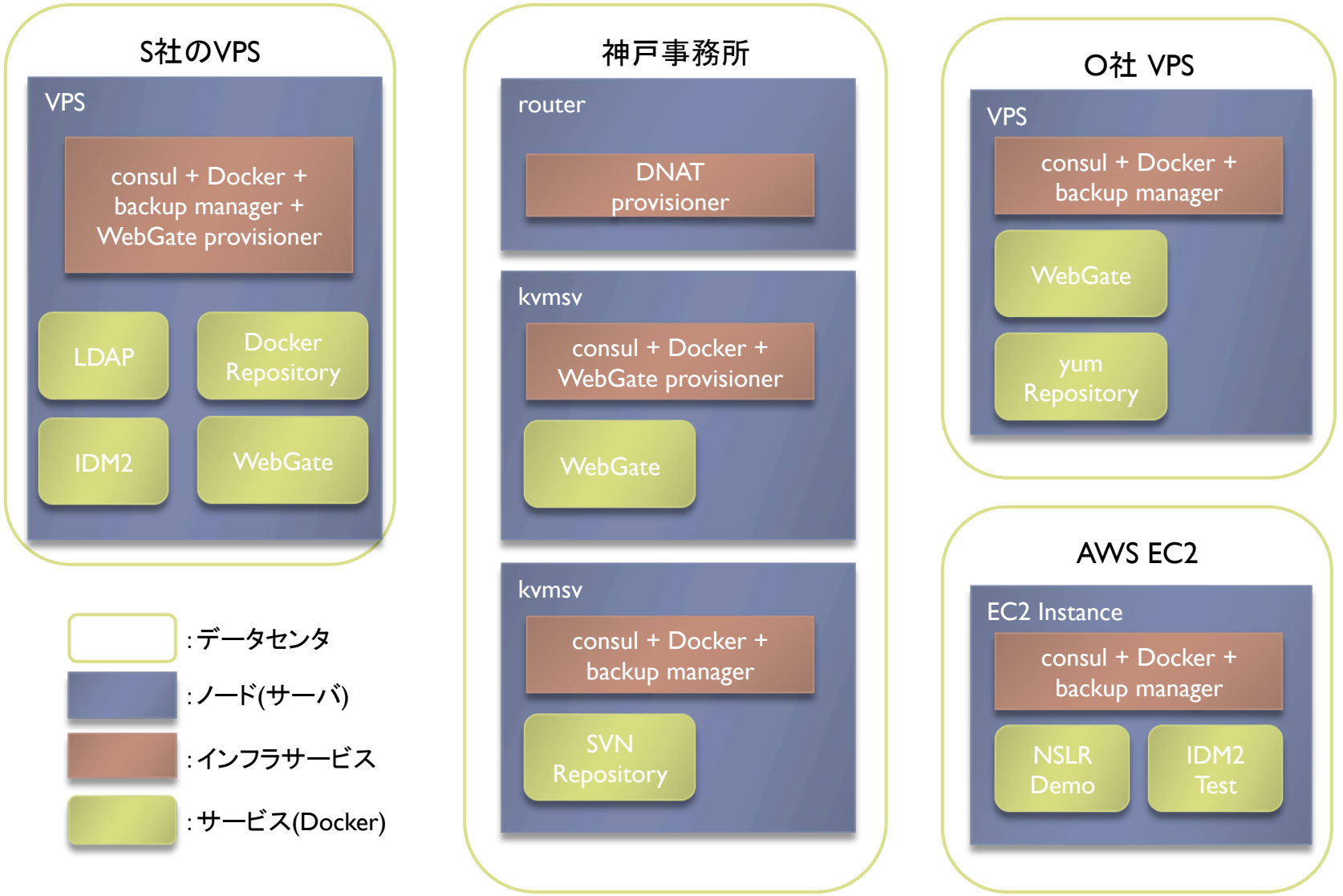
CentOS 7 + Docker + Consul を使用する

- サービスディスカバリ: Consul のデータベースにDNSで問い合わせることで、サービスがどこで実現されているかを検索できる→サービスを容易に移動できる（例：WebサーバがDBサーバを見つける）
- 投票制リーダー決定＋セマフォ排他制御で、スプリットブレインを防げる
- DNSで問い合わせることができるので、ブラウザやSSHターミナルなどのクライアントからもサービスディスカバリを利用可能
- Dockerの可搬性を利用してサービス移動を実装（データも前日のバックアップに戻る仕様であればそれほど難しくない）
- CoreOS+etcd でも同様のことが可能であるが、クラウドやハードウェアに依存してしまうので、メジャーなディストリビューションで動作できるConsul のほうが有利であると考えた
- AWS Route53などのクラウドサービスに依存せずに実装できる

マルチベンダクラウド


O社のほうがS社より少しだけ安いが、S社はすでに1台契約しているので、これはそのまま稼働させる
非常に短時間（月40時間くらい）の利用であれば、AWS オンデマンドインスタンスが安い
常時起動する場合は、S社やO社 のVPSが安い
S社、O社、AWSの3ベンダ体制とする
AWS EC2 は起動する毎にIPアドレスが変わるが、noip.com/free などを使用すれば、DNSに自動登録されるので、名前で参照が可能
→Elastic IPの購入不要
AWSの機能にあまり依存しないようにする

プロキューブITサービス基盤全体イメージ



Docker 製品の導入

Docker の問題点

- プロセスの管理がデーモン経由であるため、systemd や監視ソフトなどとの連携が複雑になる
- ネットワークをDocker デーモンが管理しているため、柔軟性にかける
  - --link は再起動するとアドレスが変わってしまうので使えない
  - マルチホスト対応には consul などの外部レジストリが必要
  - firewalld との相性が悪い
- 自前リポジトリには、一覧取得、削除、ゴミ掃除、ACLなどの機能がない
- リポジトリのメタデータ管理機能（タグ付け、版管理、ACL）が必要→開発サイドの検証環境と連動するような仕組みが必要
- イメージのレイヤの重ね方が自動で行うものしか使えない

将来的に systemd-nspawn に乗り換えることも考慮

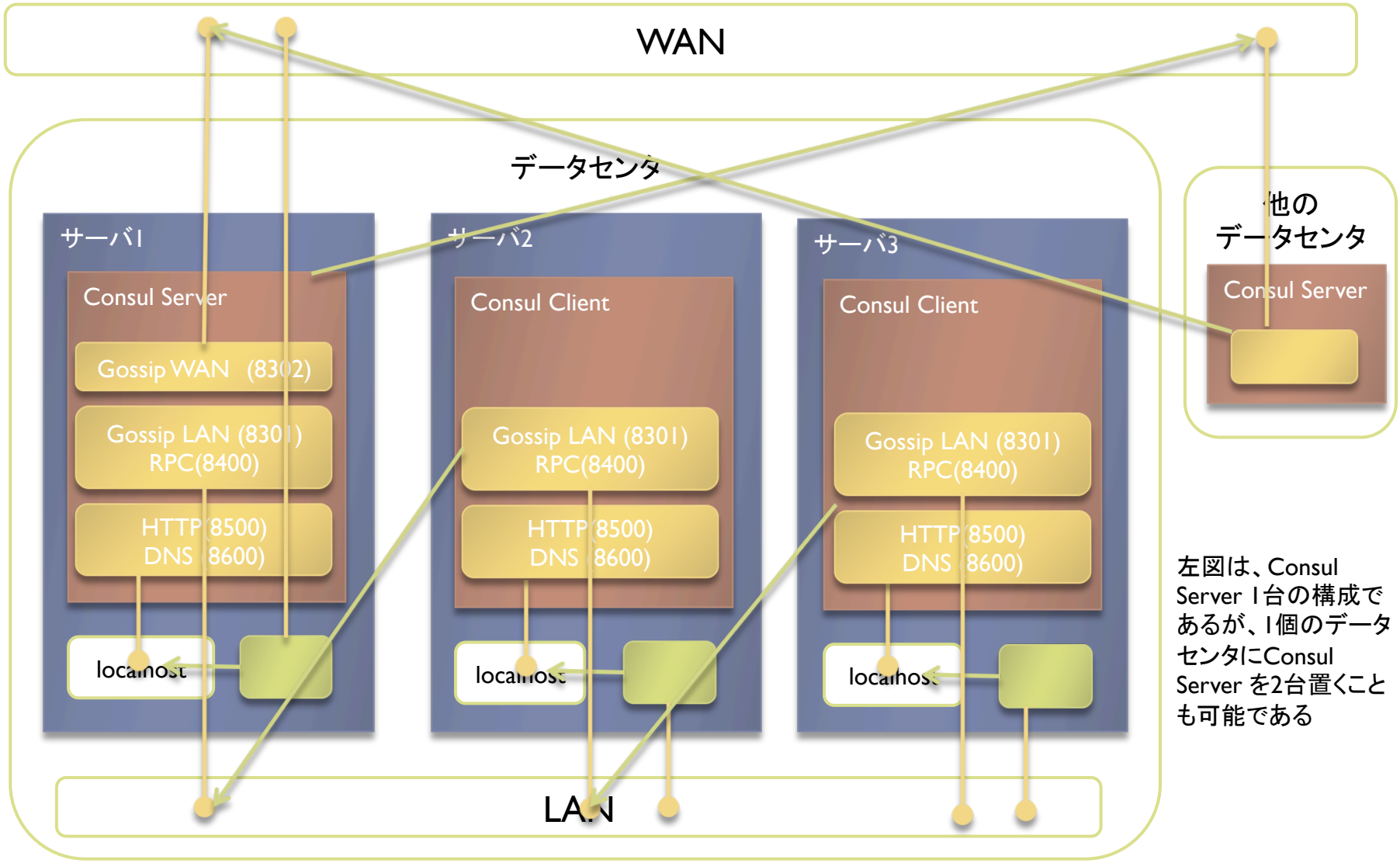
- クラスタ制御ソフトはDocker に依存しない構造とする（＝Docker はサービス実装の一手段とし、Swarm などを使用しない）
- ネットワークはDocker のデフォルトのものを使用するが、--link は用いずに consul でコンテナの間の名前解決を行う
- マルチホストにおけるマイグレーションは consul とクラスタ制御ソフトにより実装する
- systemd-nspawn の以下の課題が解決できたら、nspawn に乗り換える
  - CentOS7 の machinectl のイメージ管理機能が未実装
  - machinectl がアクセスするリポジトリを管理するソフトウェアが必要(webdav のようなもの）
  - イメージのレイヤ管理を行うソフトウェアが欲しい

consulの機能

- 分散 Key Value Store
  - 投票（Raftアルゴリズム）よるリーダー選出
  - ロック/セマフォによる排他制御
  - consul-template, watches によるイベントハンドリング
- 動的DNSによるサービスディスカバリ
- サービスのヘルスチェック
- HTTP API
- 管理用Web UI

クラスタ管理ソフトとしてのプリミティブは揃っているが、制御プログラムがない
↓
シェルスクリプトで作成することにした

物理構成パターン



左図は、Consul Server 1台の構成であるが、1個のデータセンタにConsul Server を2台置くことも可能である

設定ファイル例

Consul Server

```
{
  "datacenter": "public",
  "data_dir" : "/var/run/consul",
  "log_level": "INFO",
  "node_name" : "sakura-vps",
  "server": true,
  "domain" : "procube.jp",
  "encrypt": "aoxG1KHUWakCOGHB1jAqCg==",
  "start_join" : [“consul-kobe.procube.jp”,
      "consul-sakura.procube.jp","consul-onamae.procube.jp"],
  "bootstrap" : false,
 "advertise_addr" : ”192.168.33.2",
  "advertise_addr_wan" : "219.94.254.200",
"verify_incoming" :  true,
  "verify_outgoing" :  true,
  "cert_file" : "/etc/pki/CA/certs/server.public.procube.jp.crt",
  "key_file" : "/etc/pki/CA/private/server.public.procube.jp.key",
  "ca_file" : "/etc/pki/CA/cacert.pem"
}
```

Gossip通信の暗号化に使う共有鍵（ consul keygen コマンドで生成して、全ノードに同じ値を設定する）

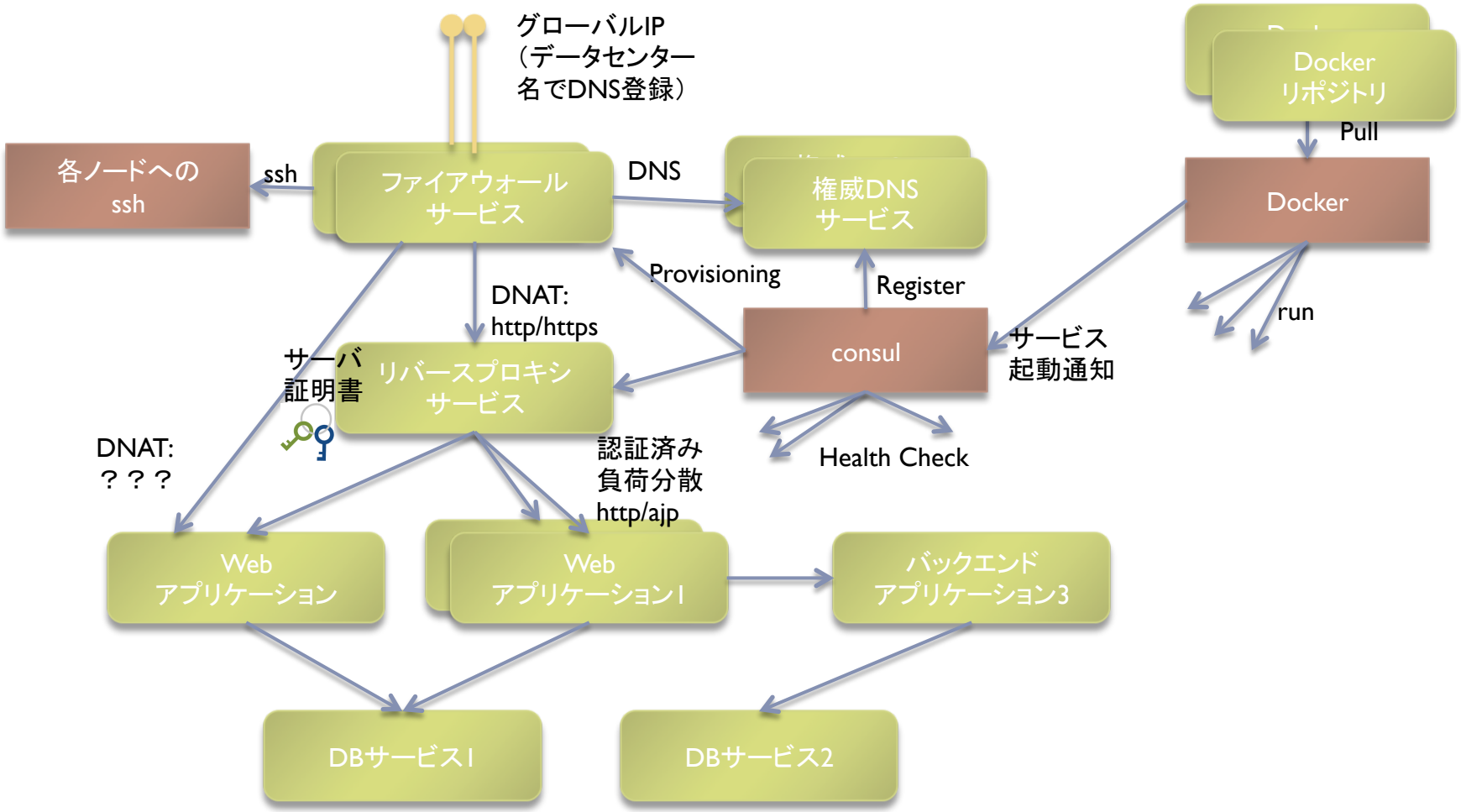
起動時にクラスタの他のサーバと連携する。これらサーバ間で投票が行われ、リーダが決定される

インターネットからこのノードにアクセスする場合のIPアドレス（サービスが権威DNSサーバに登録されるアドレス）

RPCの暗号化は非対称鍵で行う。独自CAを立てて証明書を作成したが、すべてのノードで同じ秘密鍵を持つ運用としている。

プライベートネットワークが複数刺さってるサーバの場合にエラーになる場合がある。その場合は、“bind_addr” : プライベートIPで、BINDするアドレスを指定する。

クラスタの論理構成


グローバルIP
（データセンター
名でDNS登録）
各ノードへの
ssh
ssh
ファイアウォール
サービス
DNS
権威DNS
サービス
Docker
リポジトリ
Pull
Docker
run
Provisioning
Register
DNAT:
http/https
サーバ
証明書
リバースプロキシ
サービス
consul
サービス
起動通知
Health Check
DNAT:
？？？
認証済み
負荷分散
http/ajp
Web
アプリケーション
Web
アプリケーション1
バックエンド
アプリケーション3
DBサービス1
DBサービス2

ファイアウォールサービス

- データセンターの入り口として、クライアントからのリクエストを受け付ける
  - グローバルIPを持つホストごとに稼働する（神戸事務所以外のデータセンタはすべてのサーバにグローバルIPがあるので、すべてのサーバで稼働する）
  - データセンター内に複数のグローバルIPがある場合は、同じ名前で複数登録され、DNSラウンドロビンで分散される
- consul の監視により、WANにポートを公開するサービス（サービス定義JSONの tag で判定）の起動をトリガとして、 自動的にDNATを追加し、サービスが停止すると削除する
- S社、O社、神戸事務所の場合
  - docker コンテナで実装することで、設定情報のバックアップ、可搬性、冗長化を実現する
  - CentOS のネットワーク管理機能でファイアウォール機能を提供する
    - Network Manager/Firewalldで管理
    - docker run –net host ととすることで、コンテナから直接ホストのネットワークインタフェースにアクセス
    - WANとLANの２個のインターフェースを持つ
    - S社のVPSの場合は、「スイッチ」を追加して LAN でVPS間を結ぶ
    - O社のVPSの場合は、仮想スイッチ(bridge)を作成し、これをLANとみなして構成する（O社のVPSはデータセンタに１台のみの構成しかできない）
- AWS EC2 の場合、AWS API でセキュリティグループを制御することで実現

リバースプロキシサービス

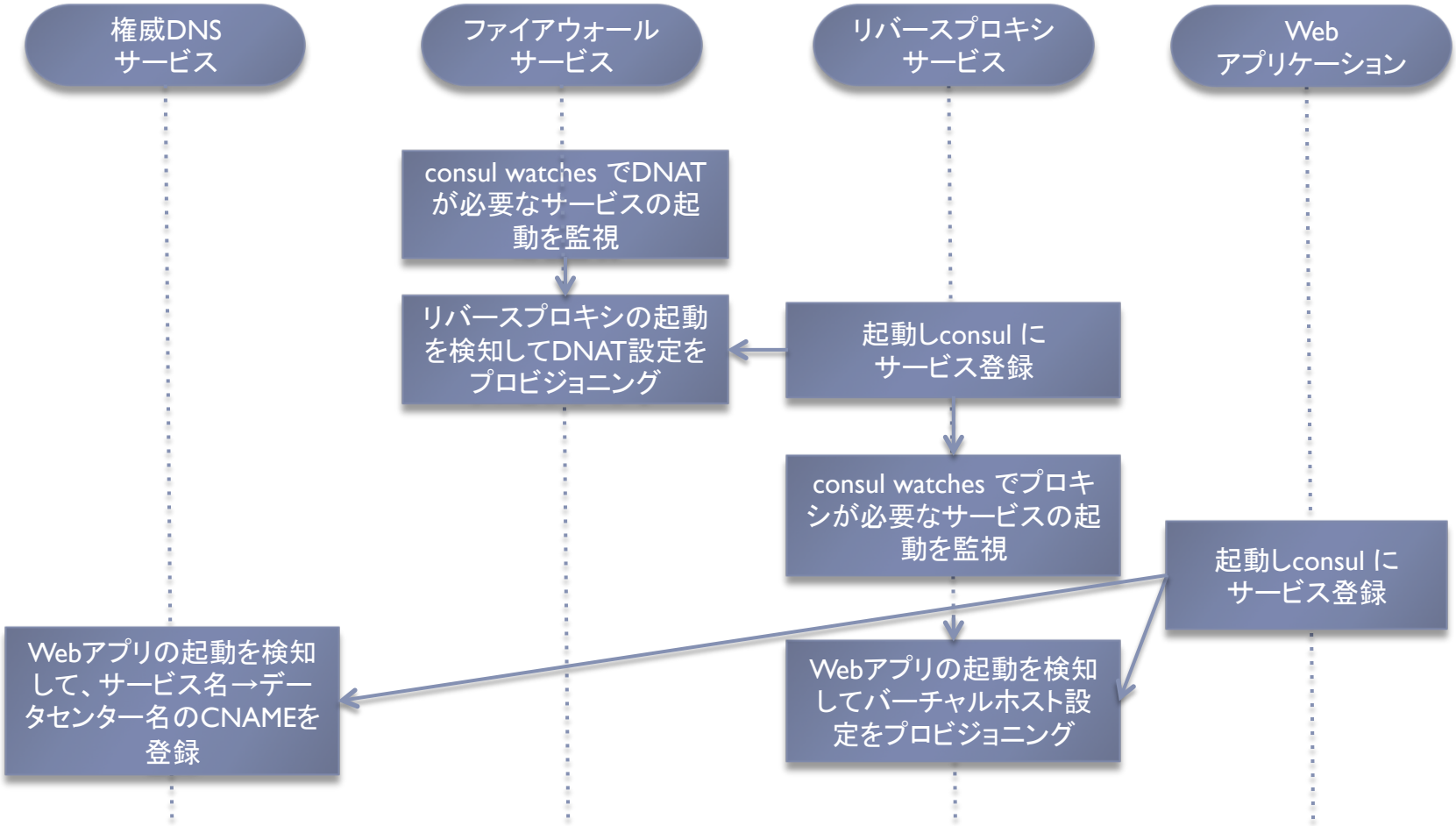
- Webアプリに対するリバースプロキシ機能を提供する
  - 代行ログイン機能を提供する（Basic認証、フォーム認証）
  - ロードバランサ機能を提供する
  - SSL接続機能を提供する（ワイルドカード証明書の秘密鍵を保持する）
- docker コンテナで実装することで、設定情報のバックアップ、可搬性、冗長化を実現する
- consul を監視し、Webでサービスを提供するサービスをトリガとして、自動的にリバースプロキシ設定を追加し、サービスが停止すると削除する
- ワイルドカードサーバ証明書を保持し、https でのサービス提供を可能とする
- APIを使用して、起動したサービスをデータセンターの名前に対するCNAME (ex. sdc.procube.jp → dc-onamae.proucbe.jp)としてDNSへ登録する
- LDAP フォーム認証、SAML SP認証などの認証機能を提供する
- 認証を使用した際は、LDAPやSAMLアサーションから取得したユーザの属性情報をCGI変数やヘッダー情報として転送する

権威DNSサービス

- クライアントに対するDNS機能を提供する
- docker コンテナで実装することで、設定情報のバックアップ、可搬性、冗長化を実現する
- consul を監視し、WANに対してサービスを提供するサービスが起動すると、起動したサービスをデータセンターの名前に対するCNAME (ex. sdc.procube.jp → dc-onamae.proucbe.jp)としてDNSへ登録し、停止時に削除する

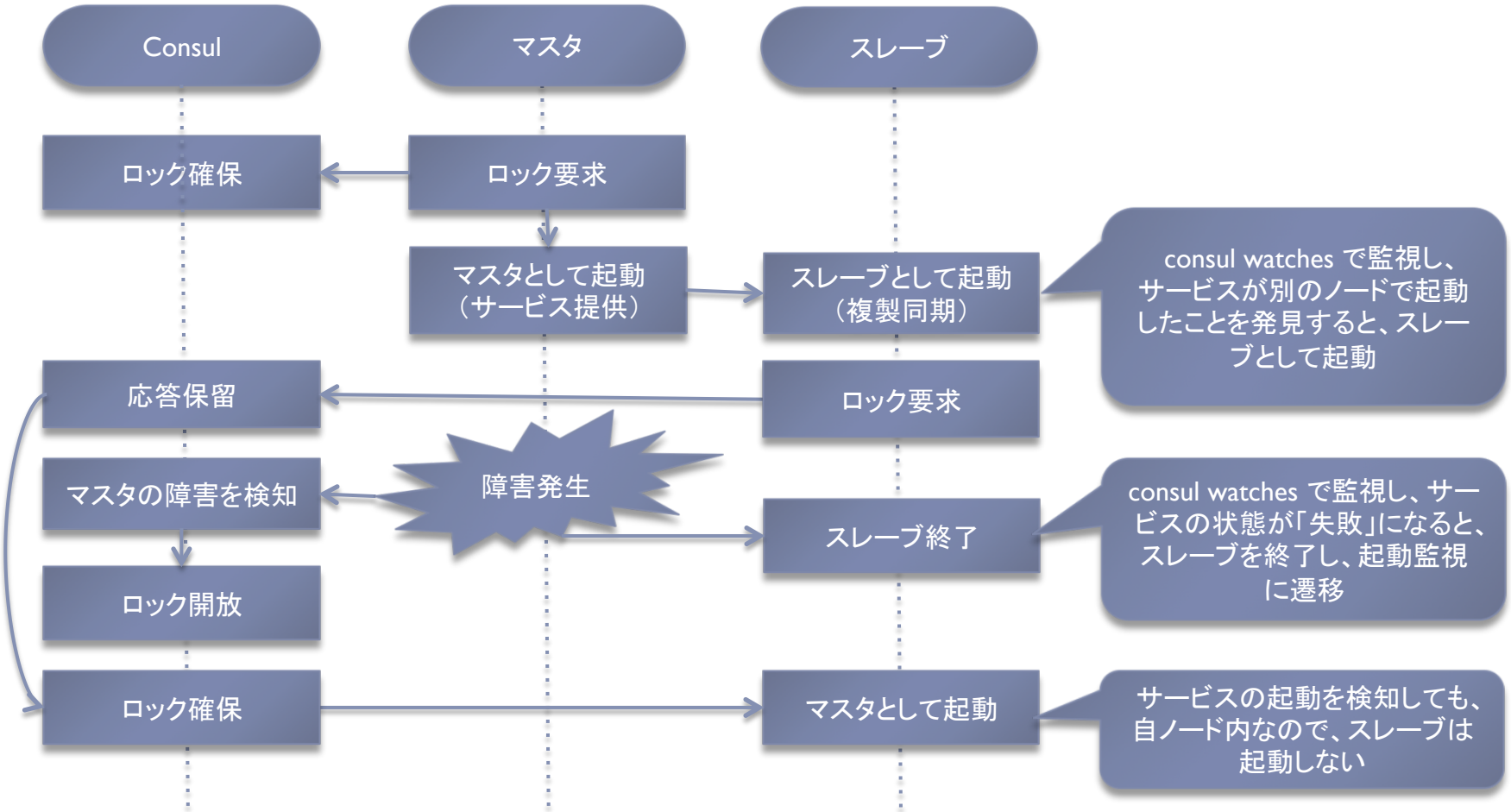
オンデマンドサービス登録

- サービス提供に必要な設定をオンデマンドプロビジョニング



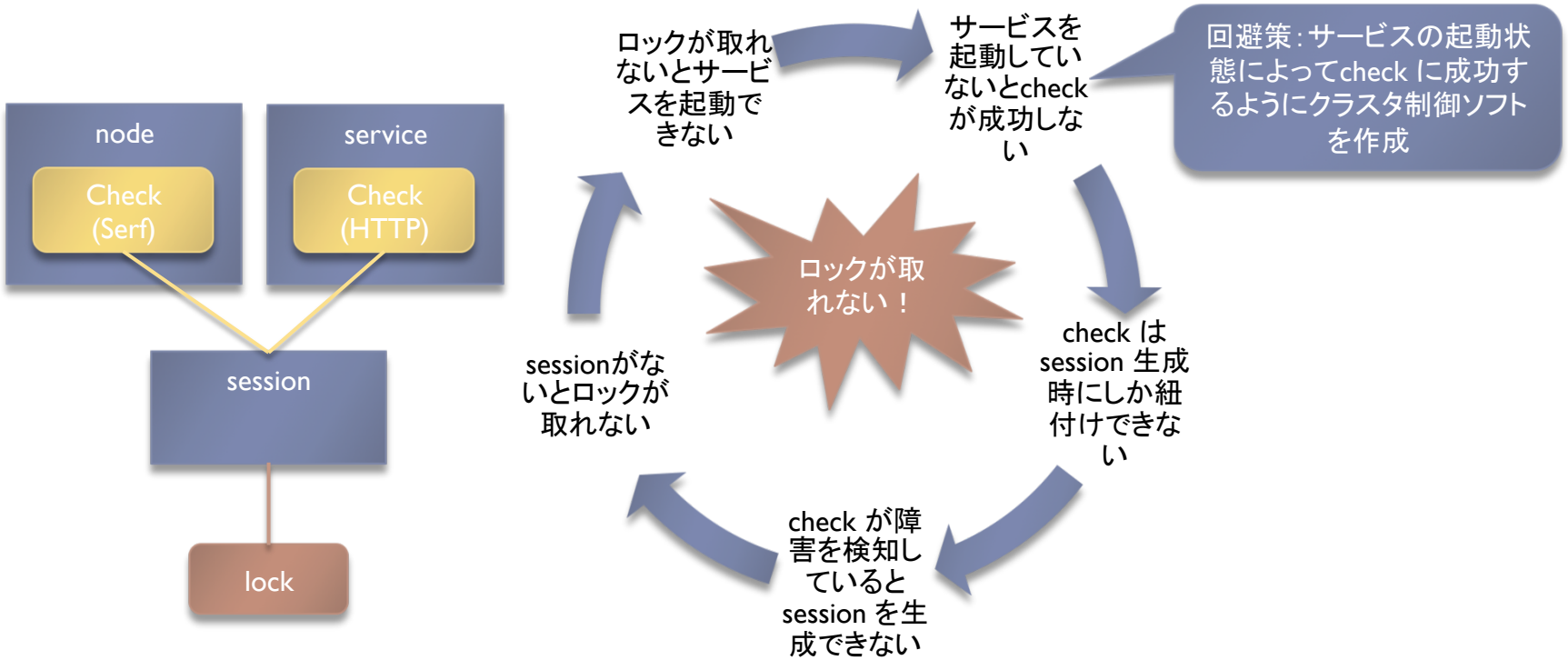
HA構成

- マスターースレーブ間でロックを取り合うことでHAを実現



consul のロックの問題点と回避策

- HA構成では、障害発生時に自動的にロックを開放する機能が鍵となる
  - service や node に対して check を設定することで障害検知を実装する
  - ロックは session に紐付けるられる
  - session に紐付けされた check で障害発生を検知すると、session が持っているロックが開放される



サービス定義ファイル例

Consul Server

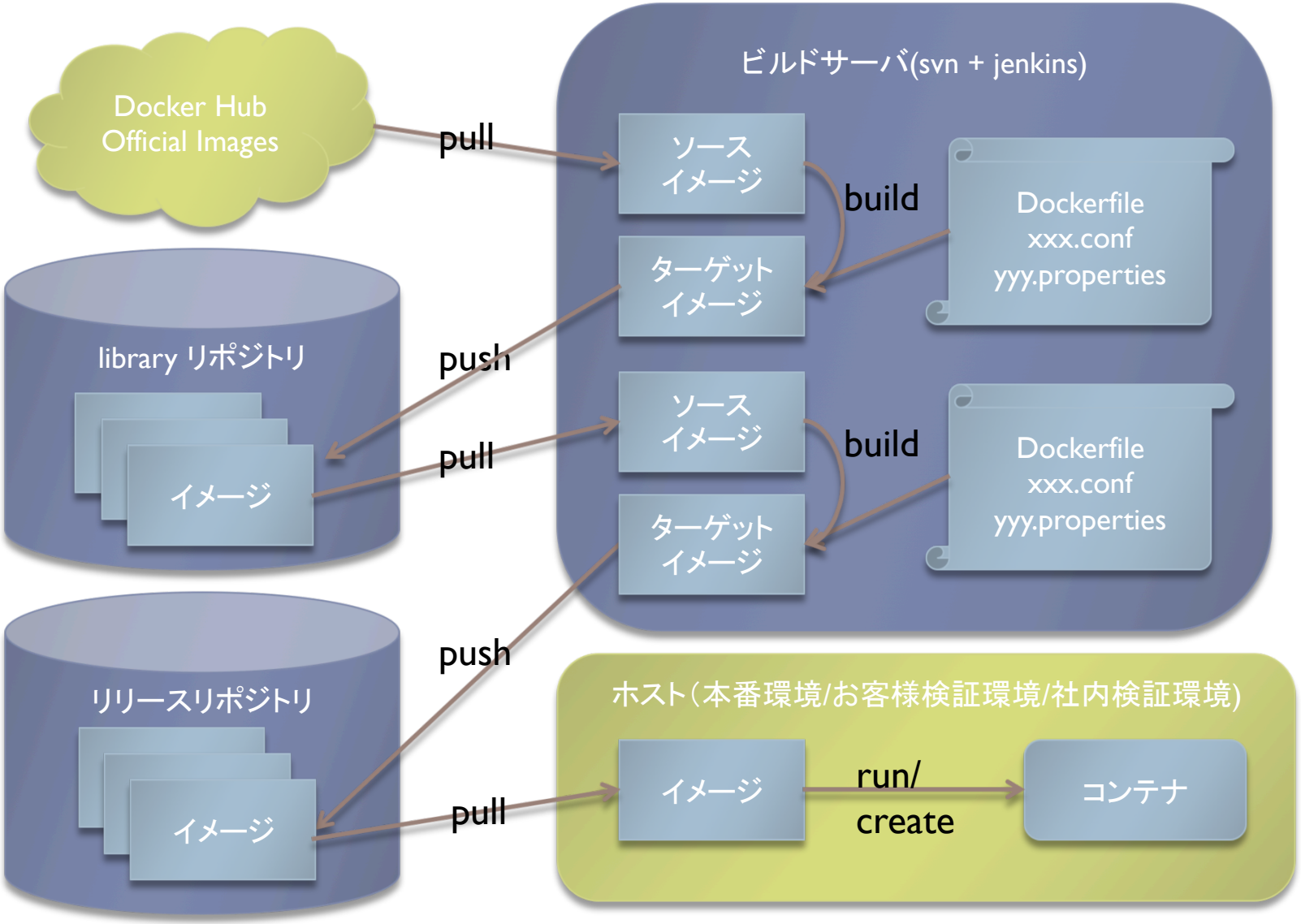
```
{
 "service": {
   "name": "nssdc",
   "tags": [“export”],
   "enableTagOverride" : true,
   ”address”: ”192.168.33.23”,
   "port": 443,
   "check": {
     "script":"/root/bin/consul_service nssdc check",
     "interval" : "10s",
     "timeout" : "4s"
   }
 }
}
```

export タグはDNAT で公開することを意味する。ファイアウォールサービスはこのtagを持つサービスが実行中状態になると、address へのDNATを自動的に追加するまた、ここに web が登録されるとリバースプロキシに転送パスが登録される

tag に “running”, “starting”, “stopped” 等の文字列を登録することでサービスの状態を管理する→check の障害検知スクリプトを状態がrunning でなければ成功を返すように実装

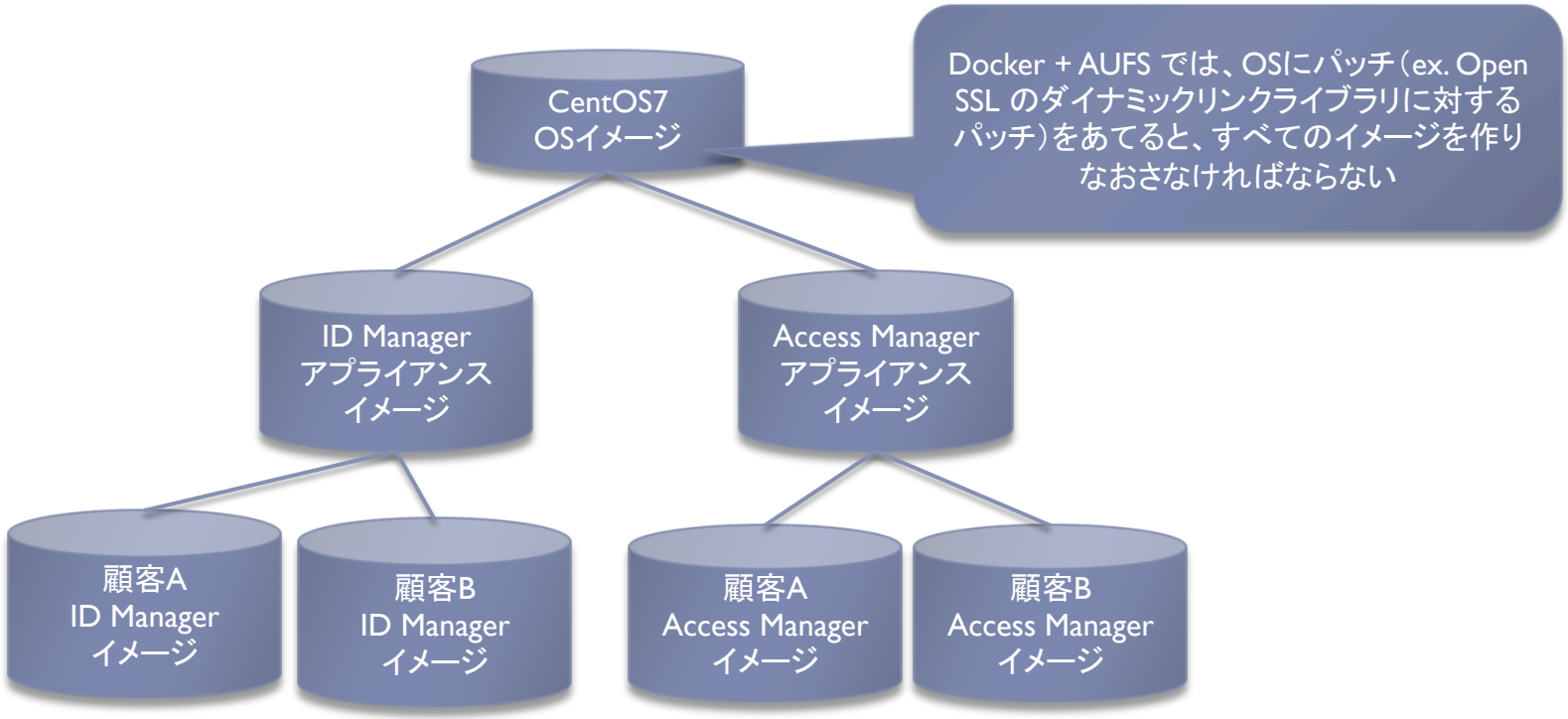
サービスの状態を監視するスクリプト、チェックの間隔、スクリプトがストールした場合のタイムアウトを指定できる

プロキューブのリポジトリ構成

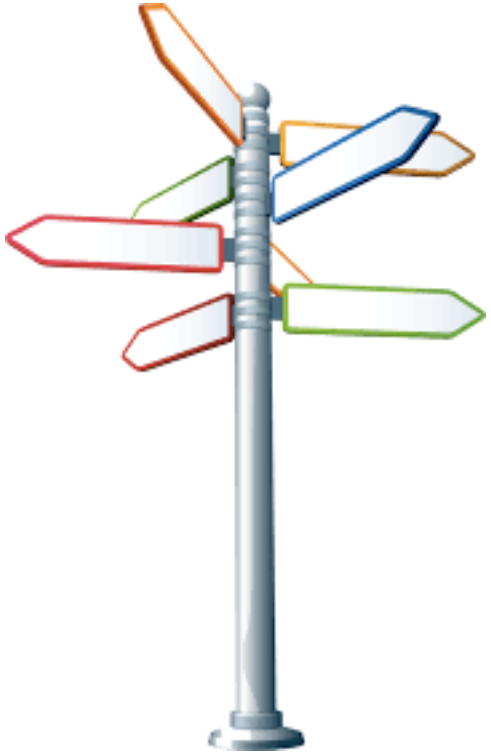


イメージレイヤ管理機能の問題点

- Docker の AUFS による差分イメージ管理では、上位レイヤにパッチをあてたときに下位レイヤも作り直しになってしまう
- 将来的にOverlayFS を自分で操作して、ファイル単位で管理したい→systemd-nspawn+overlayFS+自前リポジトリ管理ソフト



まとめ：どれが最適？ LXCクラスタ



CentOS + Docker + Swarm

シェア最強

Core OS + Rocket

クラスタ対応
完了

Cent OS + systemd-nspawn
+ consul

従来環境と
の親和性高

参考：

- AWS EC2 Linux インスタンスの動的な DNS のセットアップ (http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/dynamic-dns.html)
- Containers, systemd-nspawn and overlayfs(https://www.insecure.ws/linux/systemd_nspawn.html)