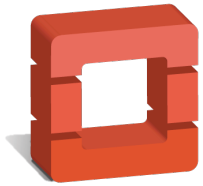
DataCenterとソフトウェア開発ワークショップ

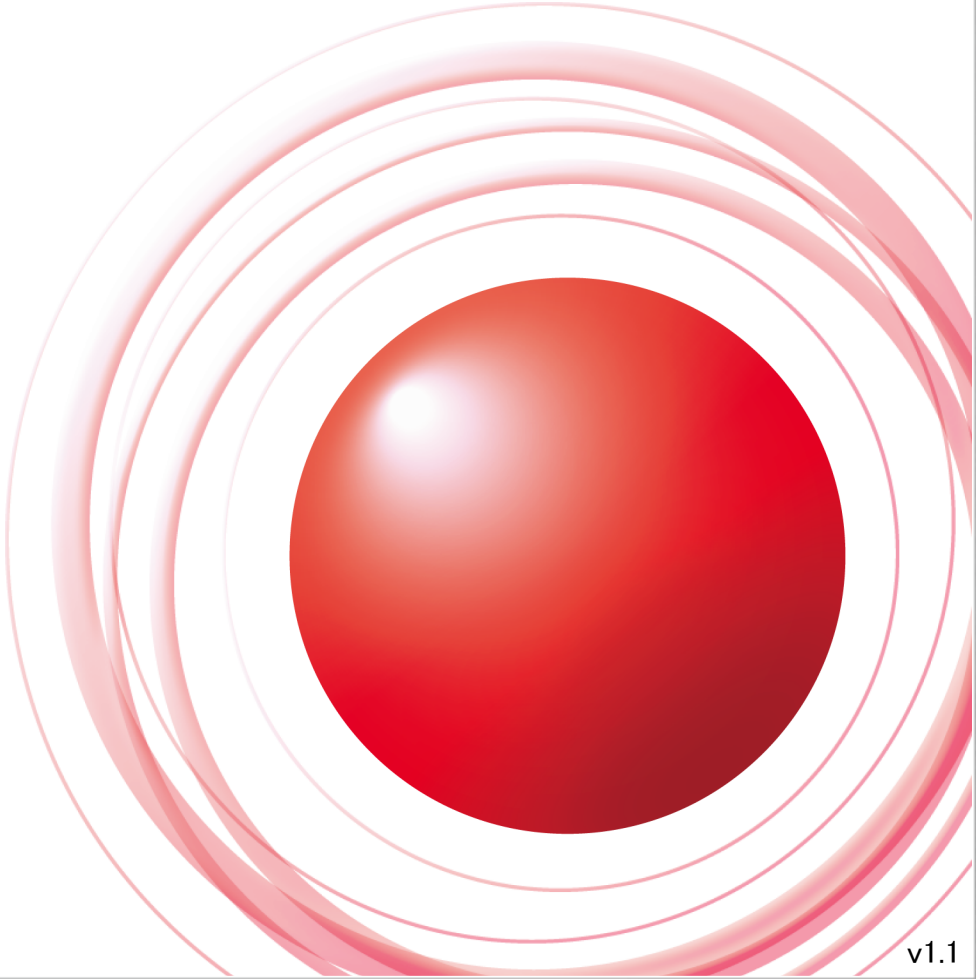
AnsibleとOpenStackによるクラウド基盤管理

IIJ
Internet Initiative Japan



openstack™
CLOUD SOFTWARE

ANSIBLE



株式会社インターネットイニシアティブ
齊藤秀喜

Ongoing Innovation

v1.1

はじめに

本セッションでは、ITシステム管理者を対象に、クラウドの普及によって寿命が短く変化の多い、仮想リソース上へ移行が進んだ現代のシステムの運用管理に焦点をあてます。

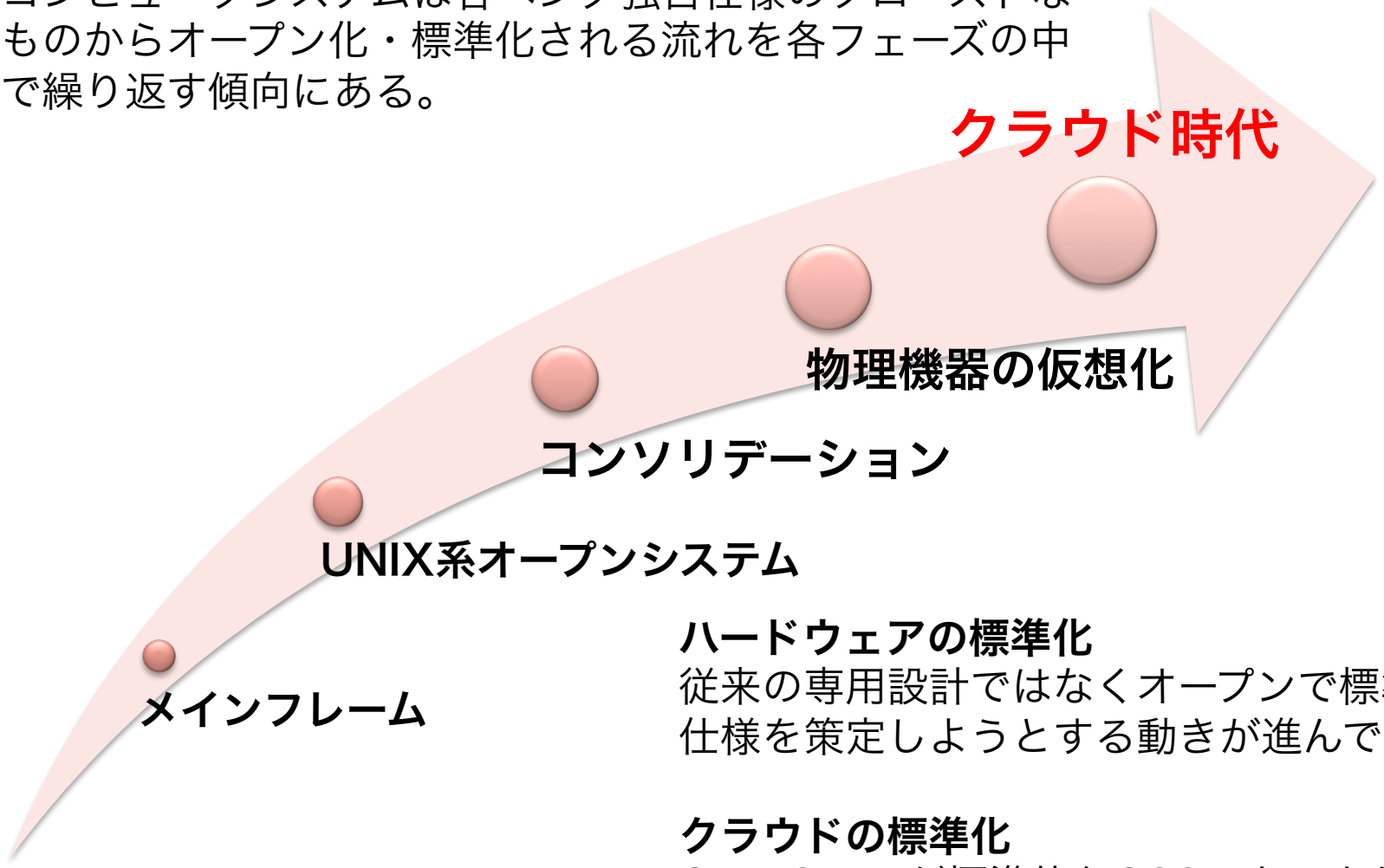
リファレンスモデルとして、OSSのクラウド管理基盤である**OpenStack**と、クラウド時代に対応したオーケストレータである**Ansible**の組み合わせを取り上げ、現代的なシステム管理手法や、それに必要となる技術を解説します。

もくじ

ITの変遷

ITインフラの変遷 ~クラウド時代~

コンピュータシステムは各ベンダ独自仕様のクローズドなものからオープン化・標準化される流れを各フェーズの中で繰り返す傾向にある。



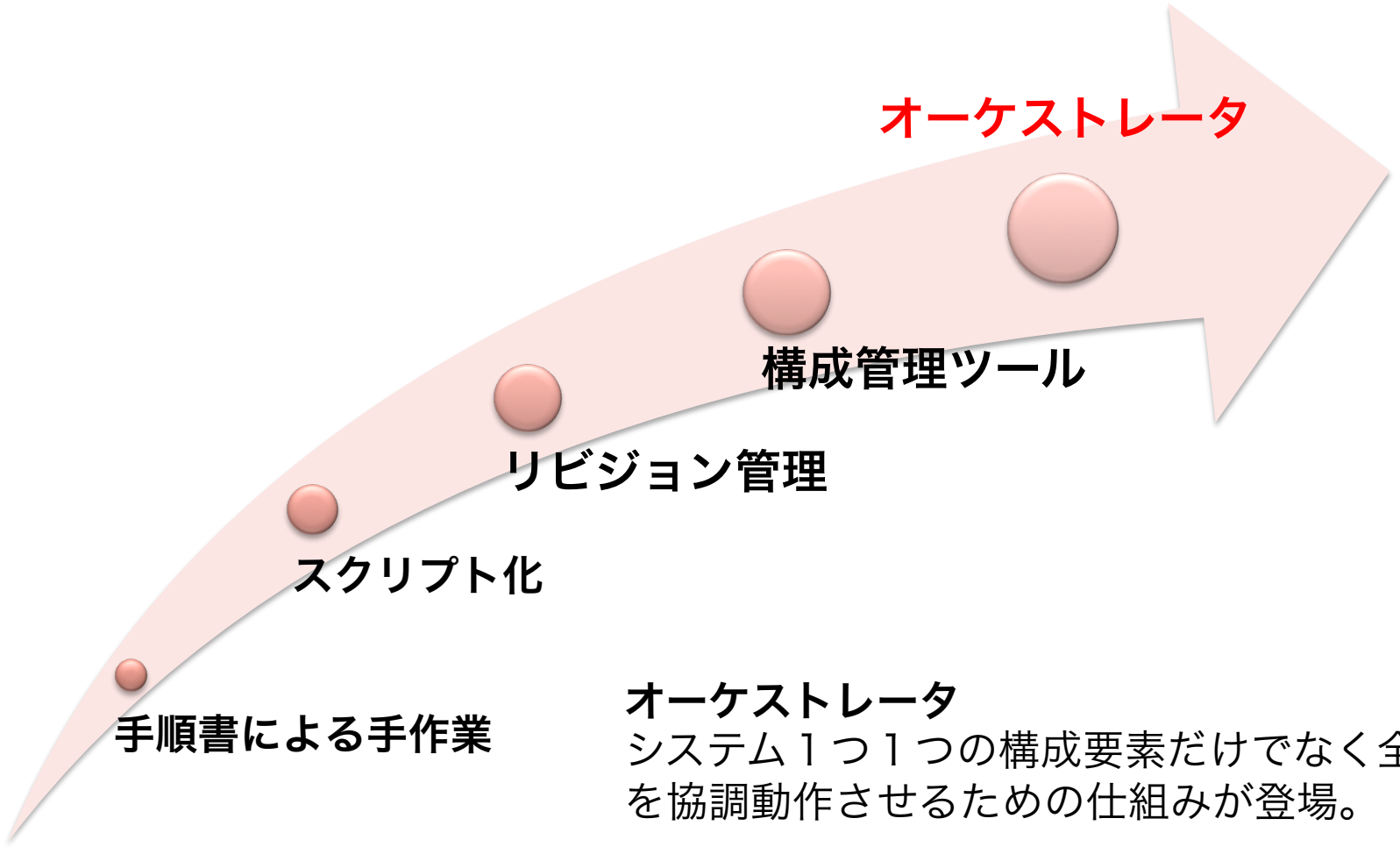
ハードウェアの標準化
従来の専用設計ではなくオープンで標準的な仕様を策定しようとする動きが進んでいる

クラウドの標準化
OpenStackが標準的なOSSのクラウド基盤として確固たる地位を築こうとしている

システムマネジメントの変遷 ~オーケストレータの登場~

Infrastructure As Code

手順書による手作業から徐々に作業のコード化を進めてきたが、従来手法では対応の難しかったインフラ初期構築作業もクラウドの登場でコード化が可能に。



オーケストレータ

システム１つ１つの構成要素だけでなく全体を協調動作させるための仕組みが登場。

変化するライフサイクル

進化する開発手法と追随するインフラ管理技術

現在のITビジネスの速度感にあわせた、新たなソフトウェア開発手法の登場と、それに追随するインフラ技術の進化が相乗効果となって、さまざまな変革が我々のエンジニアライフに訪れています。

新たなソフトウェア開発手法

1. Continuous Delivery & Continuous Integration
2. Blue Green Deployment
3. Immutable Infrastructure

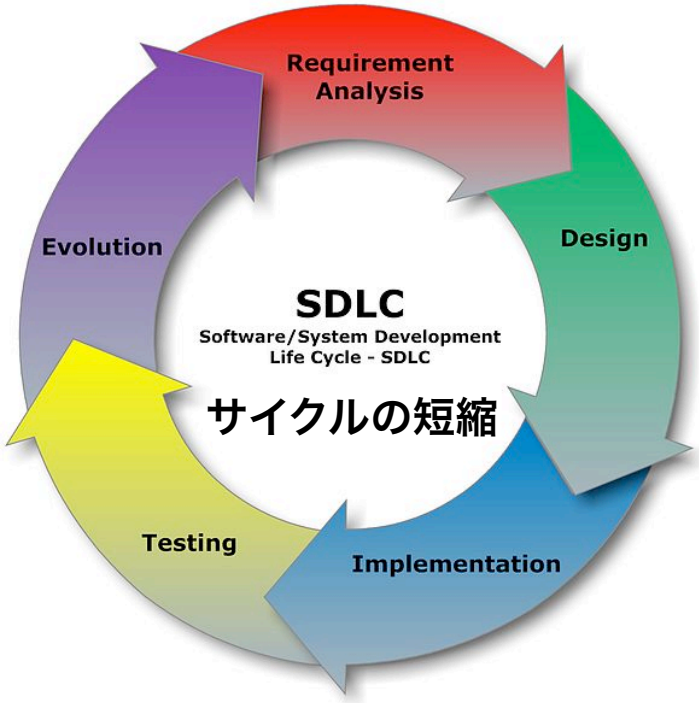
インフラ技術の進化

1. Software Defined Infrastructure
   Server / Network / Storage / Cloud / Datacenter
2. System Orchestration

新たな開発手法に対応して進化するインフラ技術

インフラの進化と、それに対応した新たな開発手法の提案という相乗効果が生まれている。その結果、システムのライフサイクルは劇的に短縮されました。

1) 物理リソースを直接利用していたシステム: **数年単位のサイクル**
2) 仮想化の普及:**数ヶ月のサイクル。実際は1)と変らない場合がほとんど**
3) クラウドの登場: **数日・数時間のサイクル**



相乗効果

ANSIBLE

openstack™
CLOUD SOFTWARE

システムの管理手法の進化

システムのオーケストレーション

システムのオーケストレーションという言葉から連想されるもの...

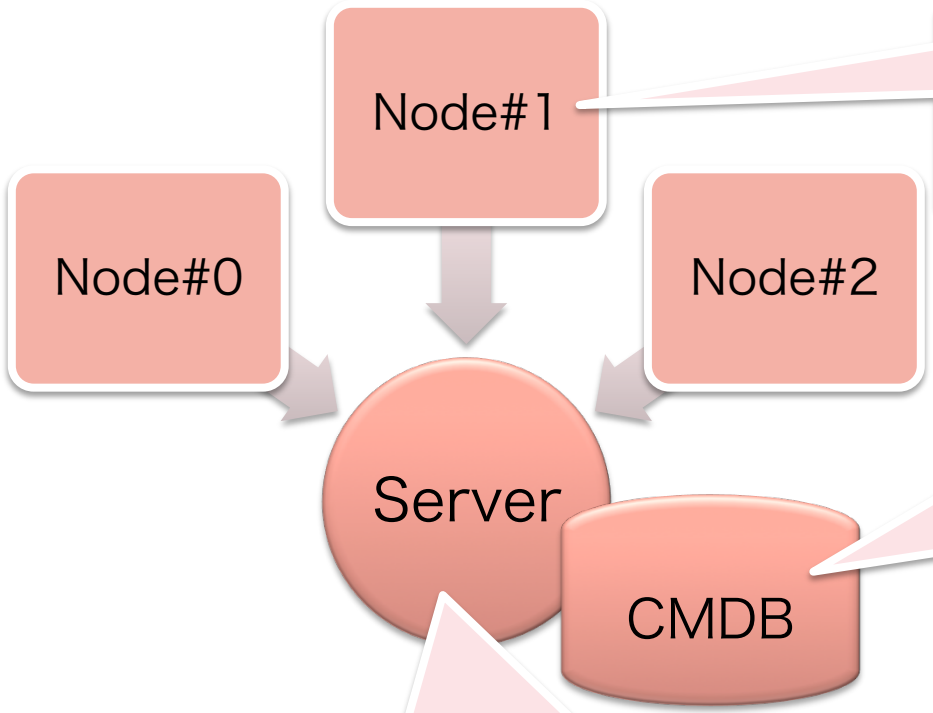
- システム間のフェデレーション
- システム構成要素の自律・協調動作
- **同一システム内の統一された管理手法(Ansibleの得意分野)**

オーケストレータは、このような夢の**一部分**を現実にしてくれるツールです。
OSSプロダクトとしては、**Ansible, Kubernetes, OpenStack-Heat, Serf, Terraform** などがこれにあたります。
活用方法しだいで、従来の構成管理ツールでは対応することが難しかったシステムの「変化」に柔軟に対応することができます。

Orchestra – 出展wikipedia

従来の構成管理ツールが抱える課題 ~変化への追随~

これまでの構成管理ツールのオーソドックスなスタイルはClient-Server型。
ノードの「あるべき姿」と「今の状態」をサーバ上の構成管理データベースで集中管理する**中央集権型**。
専用クライアントを管理対象ノードで起動することにより状態管理を行う。



あらかじめ専用クライアントを配置して起動させる必要がある。

インフラの構成情報をスプレッドシートなどで管理していた時代では非常に有効な手段だった。
クラウドでは構成管理情報を確実に管理する。そこでデータの２重管理 が問題となる。

複雑な構造を持つClient-Server型はスケールアップ/スケールアウトさせにくい

静的な要素が多い
→変化に弱い

オーケストレータ ~Ansibleの5つの特徴~

従来の構成管理ツールが苦手としていた、管理対象システムの変化に柔軟に対処できるオーケストレーションツール。
特定リソースだけでなく、ITインフラを構成する様々な要素を強調動作させることを目的に開発されている。

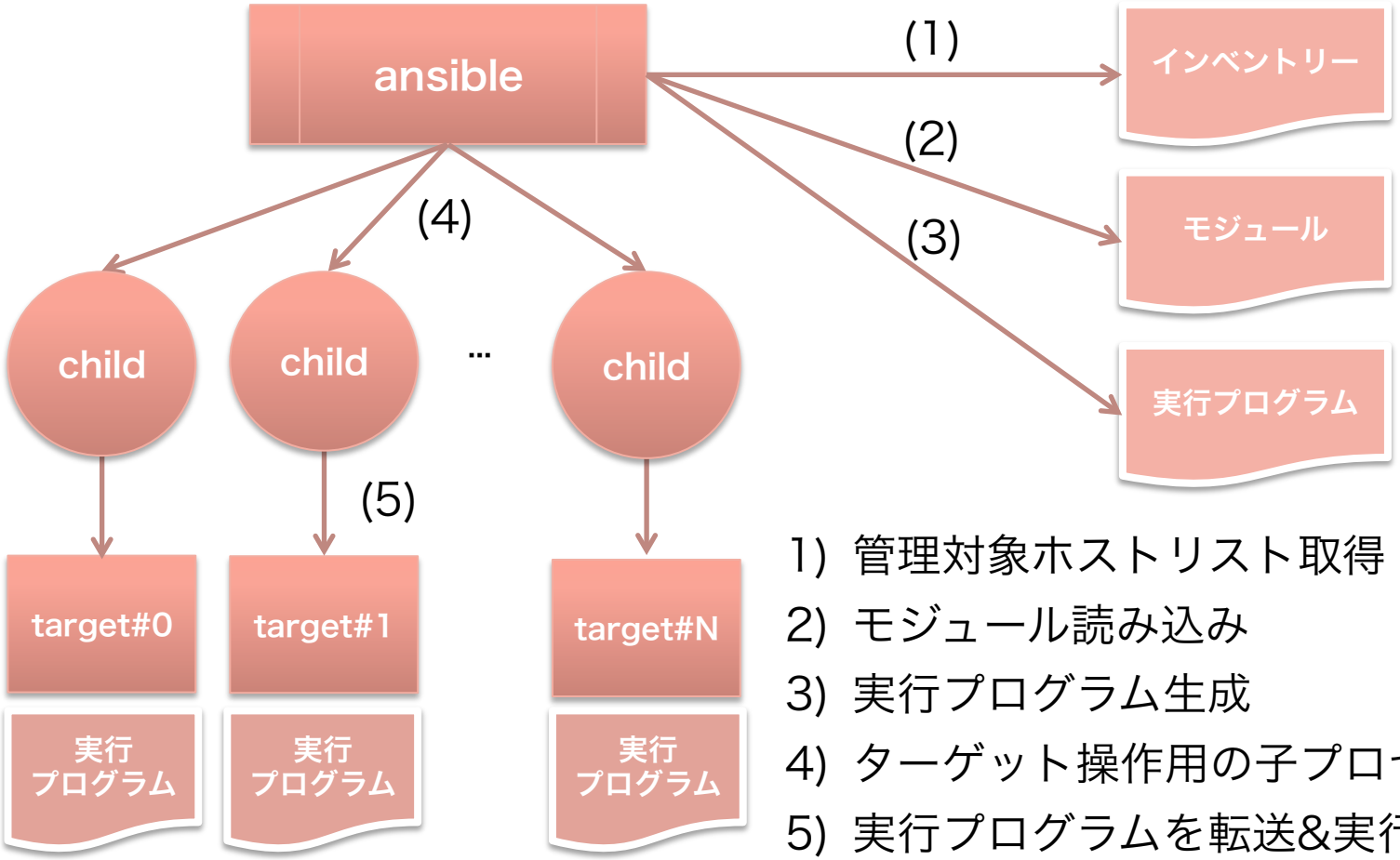
特徴

キーワード	概要
エージェントレス	管理対象ノードに専用エージェントを導入する必要がない ※Python2.4以降のランタイムが事実上必須
外部インベントリ	専用の構成管理データベースを持たず、必要に応じて外部システムの構成管理情報を参照する方式を採用している
すぐに利用可能	多数のモジュールが標準で提供されている。
シナリオ実行	多くの小さなタスクを１つにまとめることができる タスクの結果による条件分岐や繰り返し処理などの制御構造も記述可能
ドキュメントの充実	公式サイトのドキュメントが高品質で充実しており、ゼロからのスタートアップがしやすい

Ansibleのアーキテクチャ

Ansibleの*基本的な*コマンドライン

```
$ ansible <ホストパターン> -i <インベントリー> -m <モジュール>
```



1) 管理対象ホストリスト取得
2) モジュール読み込み
3) 実行プログラム生成
4) ターゲット操作用の子プロセス生成
5) 実行プログラムを転送&実行

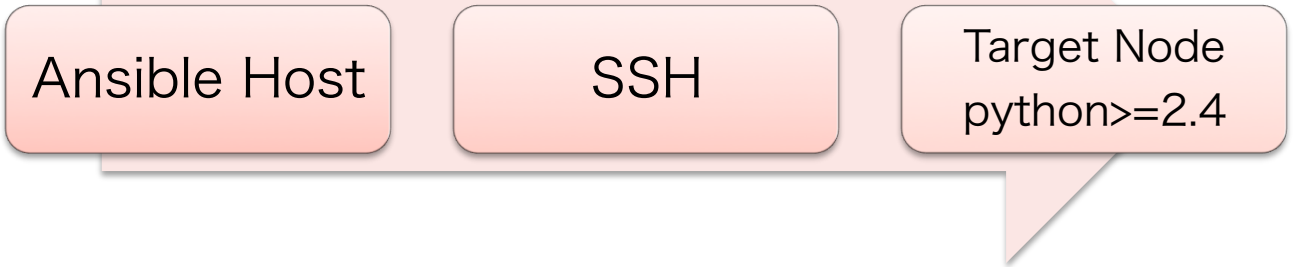
Ansibleを構成する5つの要素

構成は非常にシンプル。以下の5つの要素を利用するだけで十分な機能を利用できる。
OpenStackやAWSを操作するためもモジュールも標準で提供されており、インストール直後から利用できる。

構成要素	概要
設定ファイル	Ansibleの振る舞いを決める基本設定 デフォルトパス: /etc/ansible/ansible.cfg
インベントリー	管理対象ホストの一覧ファイル デフォルトパス: /etc/ansible/hosts
モジュール	管理対象ホストに操作を行うモジュール群
コマンド	ansibleやansible-playbookなど、モジュールやプレイブックを実行するコマンド
プレイブック	管理対象ホストに行う一連の操作をYAML形式で記述した手順書のようなもの

Ansibleによるシステム管理 ~基本3パターン~

例1. UNIXホストに対する操作



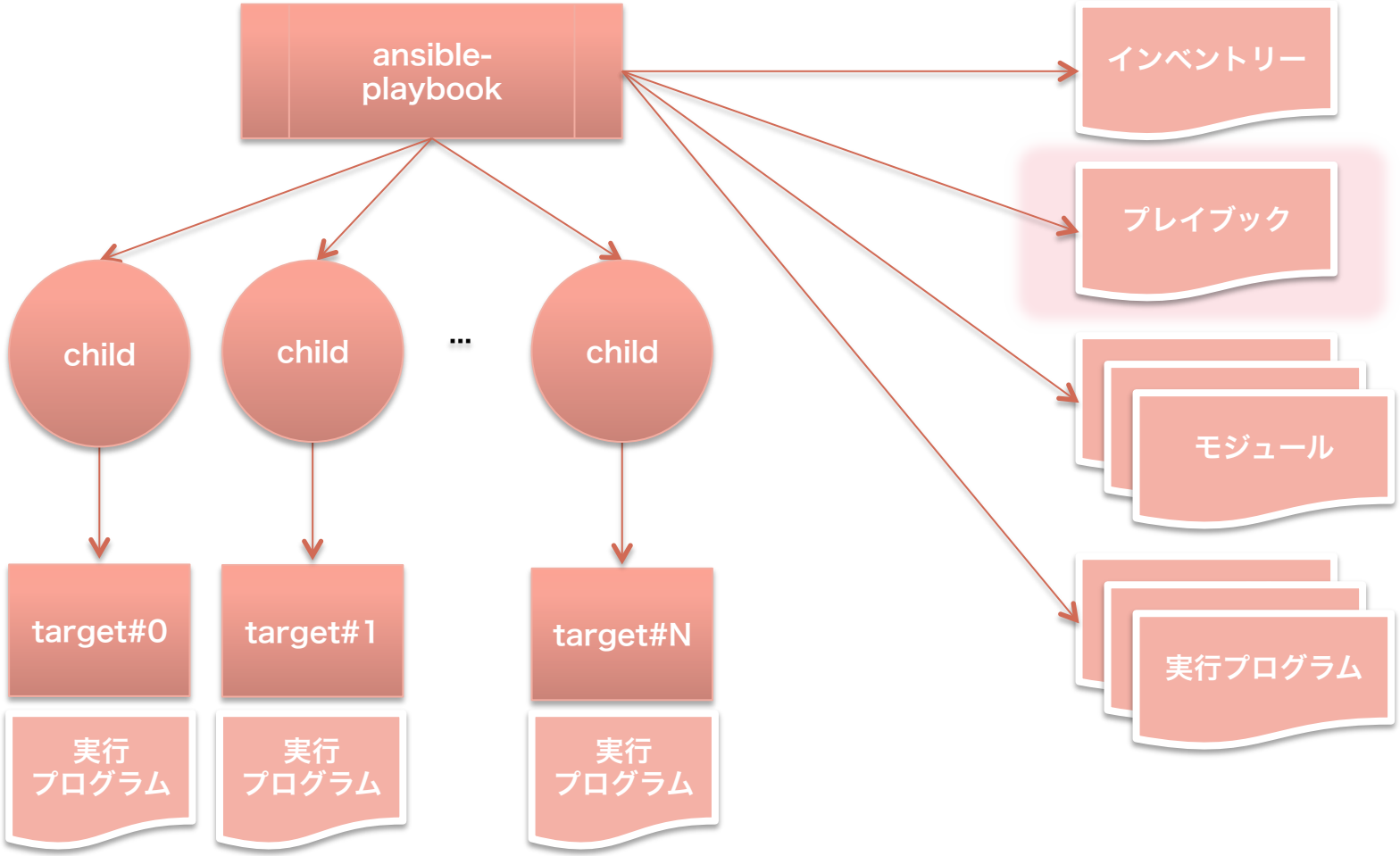
例2. Windowsホストに対する操作

例3. Netconfを利用したネットワークスイッチ操作

Ansibleによるシステム管理 ~シナリオ管理~

モジュールの適用順序をシナリオのように定義したPlaybookをターゲットに適用する。

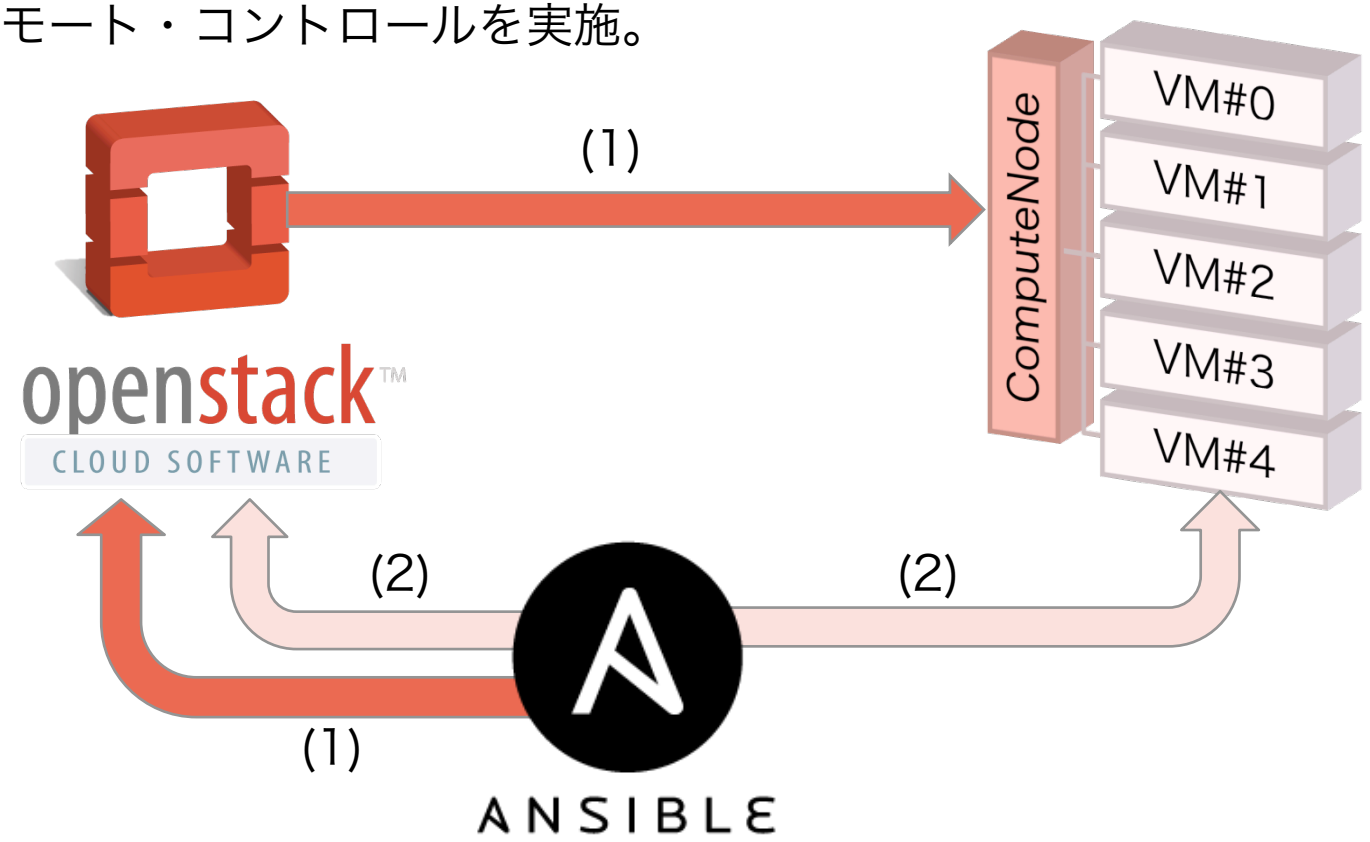
```
$ ansible-playbook -i <インベントリー> <Playbook>
```



Ansibleによるシステム管理 ~OpenStack連携(1)~

OpenStackは管理下にあるクラウド基盤を外部から制御するためのAPIを提供している。AnsibleはこのAPIを利用してオーケストレーションを実現する。

1) OpenStack API経由でリソースの作成・削除といった管理作業を実施。
2) 仮想マシンの構成情報をOpenStack APIから取得して操作対象を特定してリモート・コントロールを実施。



Ansibleによるシステム管理 ~OpenStack連携(2)~

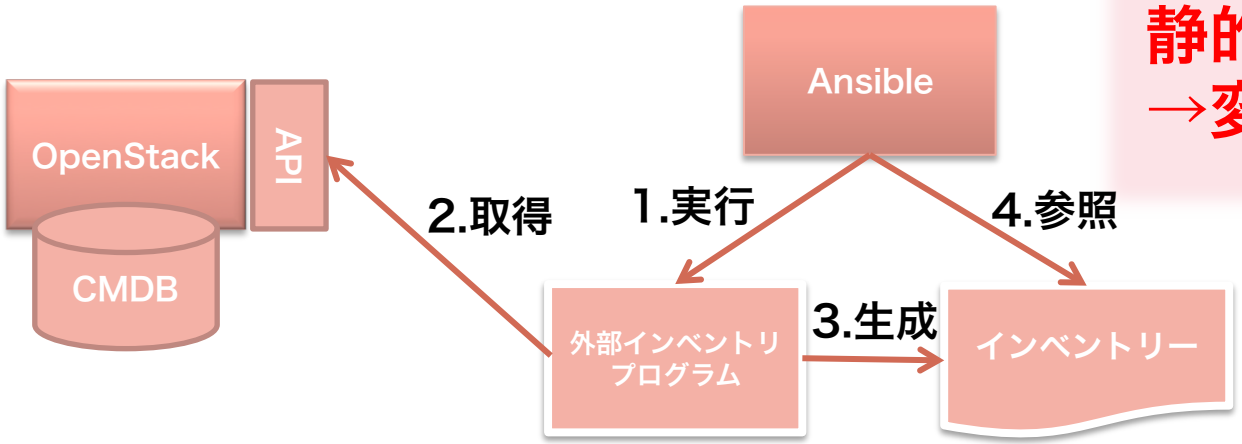
Ansibleは管理対象ノードのリスト(インベントリ)を1)または2)の方法で取得する。

1) 静的なインベントリ

UNIX系OSの/etc/hostsのようにホストのリストやパラメータが登録されたプレーンテキストファイルを利用する。

2) 動的なインベントリ(外部インベントリ)

JSON形式で構造化されたホストのリストとパラメータが標準出力される外部プログラムを実行した結果を利用する。



静的な要素が少ない
→変化に強い

Ansibleによるシステム管理 ~OpenStack連携(3)~

Ansible & OpenStack連携のデモンストレーション

AnsibleとOpenStack間の橋渡し

■ 直接制御 – AnsibleのモジュールからOpenStack APIを利用する

モジュール名	概要
glance_image	仮想マシンイメージの登録・削除
keystone_user	プロジェクト／ユーザ／ロールの作成・削除
nova_compute	仮想マシンインスタンスの作成・削除
nova_keypair	キーペアの登録・削除
quantum_floating_ip	フローティングIPの新規払い出しと仮想マシンへの割り当て
quantum_floating_ip_associate	フローティングIPの割り当てと割り当て解除
quantum_network	仮想ネットワークの作成・削除
quantum_router	仮想ルータの作成・削除
quantum_router_gateway	仮想ルータと外部ネットワークセグメントの接続・切断
quantum_router_interface	仮想ルータと仮想ネットワークセグメントの接続・切断
quantum_subnet	仮想ネットワーク内のサブネットの作成・削除

■ 間接利用 – 構成管理データベースとして間接利用する

仮想マシンの情報を管理する外部インベントリとして利用

組み合わせによる相乗効果をあげる5つのポイント

AnsibleとOpenStackをリファレンスモデルとしたオーケストレーションによる相乗効果をあげるための5つの重要な特徴

クラウド基盤

- **最新情報の管理と提供**
  構成管理データベースによるシステム情報の管理
- **コントローラブル**
  オーケストレータから制御可能なAPIを持つ

クラウドオーケストレータ

- **情報は源泉から**
  管理対象となる情報はクラウド基盤から引く
- **一貫性を保つ**
  自身では管理するのは必要最低限の情報のみ
- **相手を選ばない**
  エージェントレス型が理想。
  サーバだけでなくネットワーク機器やストレージも管理対象となる。

まとめ

オーケストレータを利用すべきたった1つの理由

アプリケーション開発の現場が求める実行環境の粒度は、物理サーバから仮想マシンへ、そしてさらに小さな環境へと変化していくことが予想されます。

実行環境が小さくなる ≒ システムライフサイクルの短縮

たとえば、物理サーバのシステムのライフサイクルは数年ですが 、仮想マシンやコンテナのような実行環境では、数分から数ヶ月程度となるでしょう。
このようなライフサイクルの短い実行環境で構成されたシステムは変化が激しく、従来の静的な情報で構成管理を行うツールでは追随が難しいのが実情です。
かといって、このような新しい概念のシステムを受け入れないわけにはいきません。ITシステムの管理者もAnsibleのようなオーケストレータを活用して、**変化に強い柔軟なシステム管理手法を積極的に提案**していきましょう。