OSSユーザーのための勉強会
< OSS X Users Meeting >
#17 Ansible と Chef

サーバー設定自動化は経営課題
--- CHEFの概要と特徴 ---

IBM Bluemix™

2017年2月23日

Presented by:
日本アイ･ビー･エム株式会社
クラウド･テクニカル･サービス
高良 真穂

IBM Cloud

自己紹介

https://www.facebook.com/maho.takara

- 高良　真穂　（たから　まほ）
- 会社　日本アイ･ビー･エム株式会社
  - 所属　クラウド･テクニカル･サービス
  - 入社16年目(中途入社)
- 仕事
  - アーキテクト / クラウド･エバンジェリスト
  - クラウド Bluemix Infrastructure（旧SoftLayer）、Bluemix Watson など

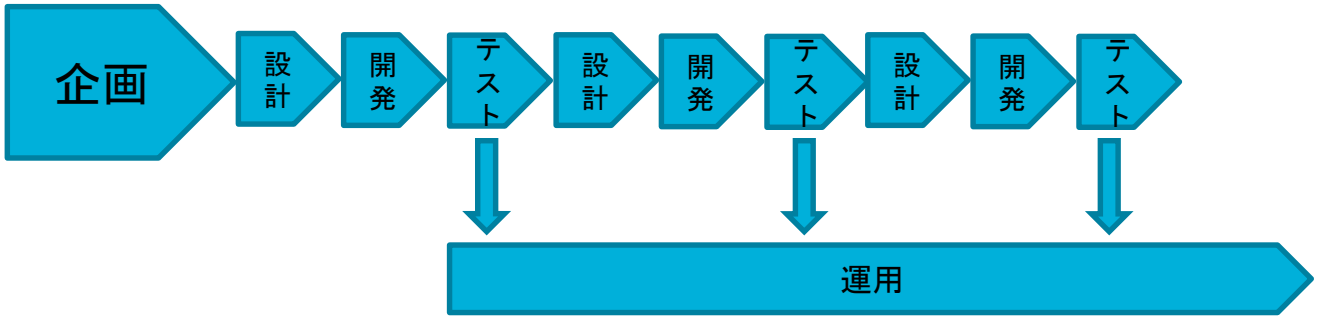
  - クラウド以前　企業向けシステムのIT基盤の設計＆構築を担当
    - 金融業、製造業、流通業、航空業界などのシステム構築に参画
    - 論文：サーバー間連携の稼働実態解析技術
    - 特許：
      - TECHNIQUE OF ANALYZING AN INFORMATION SYSTEM STATE（米国、日本）
      - METHOD AND APPARATUS FOR USING DATA（日本）
    - ウェブ記事
      - SoftLayer活用ガイド
        - https://www.change-makers.jp/docs/
      - IBM developerWorks ソフトレイヤー探検隊
        https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W2e55790226f1_47d0_a63b_84202b05783a/page/%E3%82%BD%E3%83%95%E3%83%88%E3%83%AC%E3%82%A4%E3%83%A4%E3%83%BC%E6%8E%A2%E6%A4%9C%E9%9A%8A
      - Qiita SoftLayer クッキングラボ
        - http://qiita.com/MahoTakara/items/464da29ccf932698b753

なぜサーバー自動設定ツール？ 1/2

– クラウドのプロビジョニング速度に対応
– インフラ･エンジニアの不足対策

クラウド以前　サーバーは将来の需要を見込んだ先行設備投資

クラウド後　サーバーは経費で必要な分だけ注文して即利用



短いサイクルで開発とリリースを繰り返すため
人材不足が深刻化

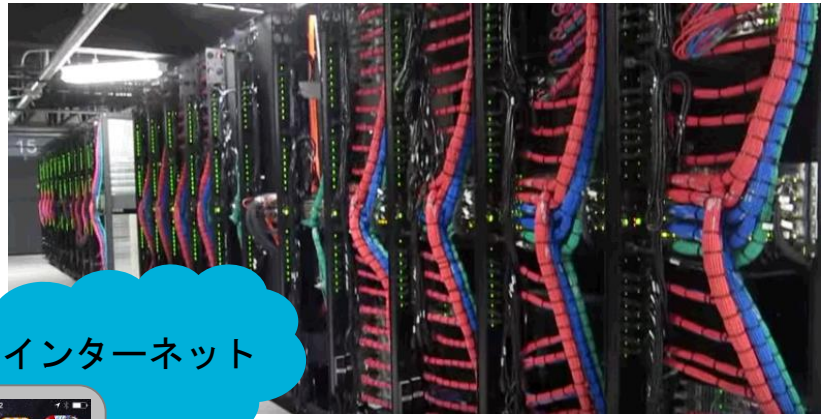
なぜサーバー自動設定ツール？　2/2

- ユーザー数が限定される企業の情報システムと異なり、スマホで何時でも何処でもアクセスできるアプリは、膨大な数のバックエンドのサーバーを必要
- アクセス数に比例して増強

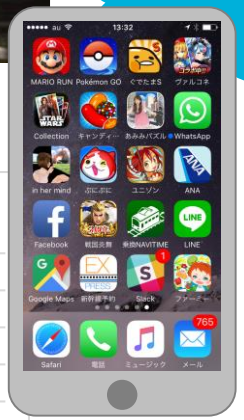
通勤の鉄道の中ではスマホ活用

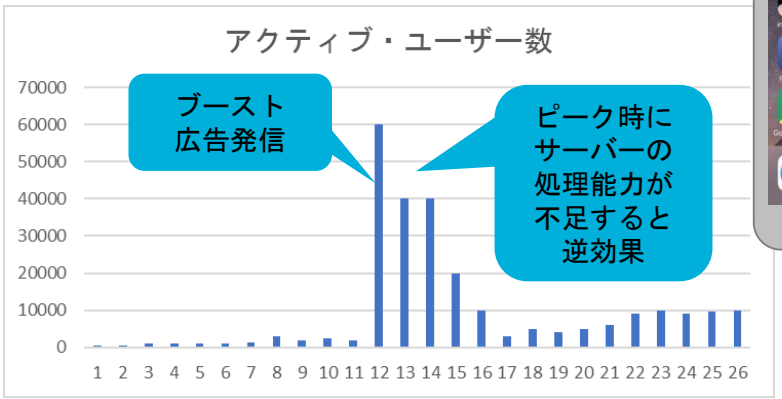
YouTube 東京メトロ 銀座線 Tokyo Metro 01,1000 系Series
https://www.youtube.com/watch?v=MUfEV8PvOvo

アプリの背後には膨大なサーバーが存在

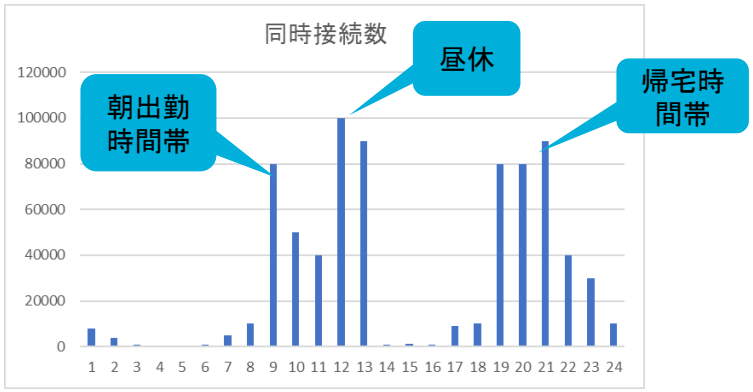


インターネット





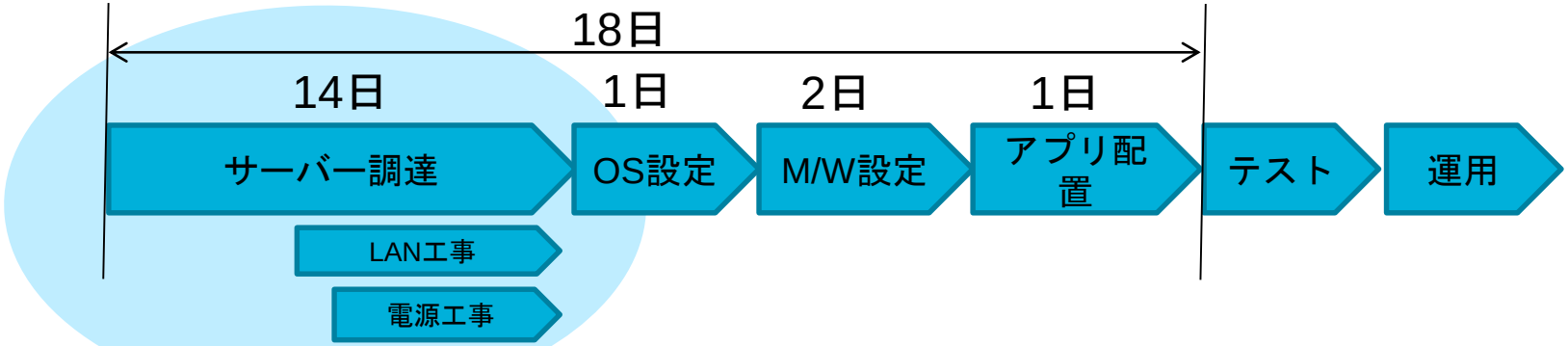
特定の時間帯などに極端に集中



ピークの伸びを考慮して増強

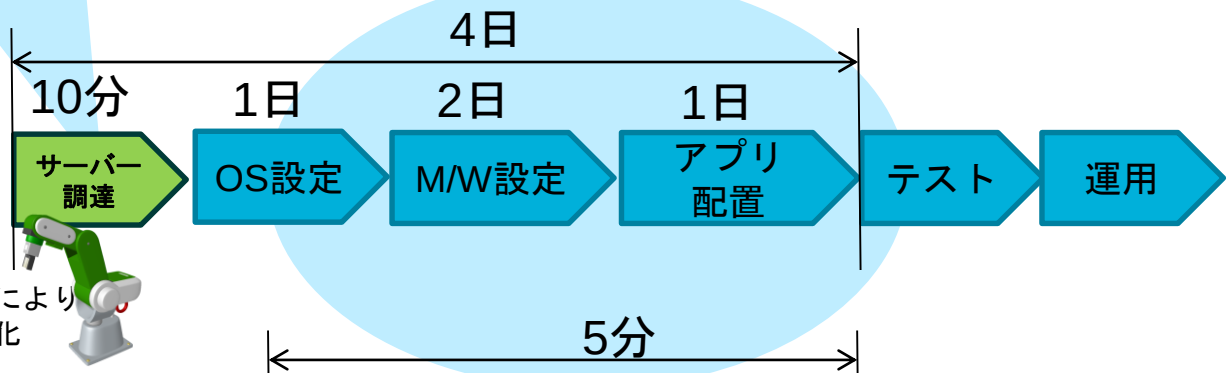
クラウドのスピードを活かすサーバー設定

オンプレ時代



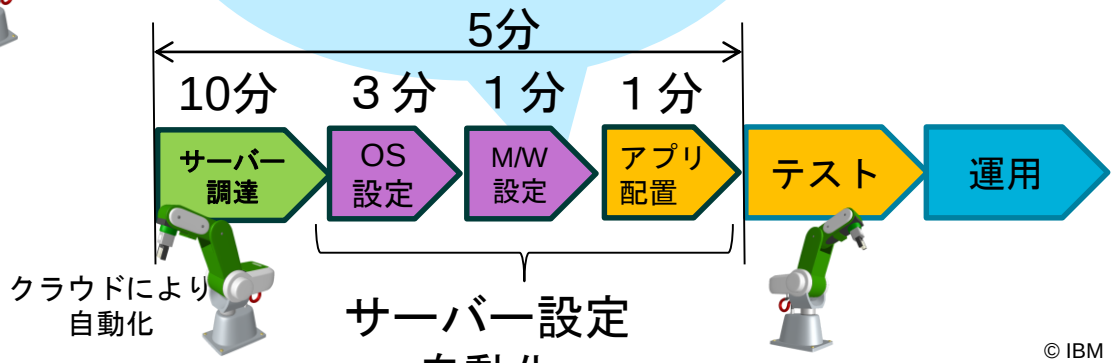
クラウド利用

サーバーの調達が
早くなっても
立上げに時間を要する



クラウド＋設定自動化

大量なサーバーの設定には
自動化したい



サーバーの自動設定ツールの役割

– 以下の作業を自動化するものです

– OS設定を自動化
  - OSのパッケージの最新化やアップグレード
  - 必要なOS配布パッケージの追加
  - ユーザー、グループの追加
  - セキュリティ設定
  - ネットワーク・ストレージとの接続設定

インフラ技術者に代って作業するロボット

- 省コスト
- 人的ミス排除
- スピードアップ

– ミドルウェアの導入と設定の自動化
  - データベースの導入と設定
    – MySQLのインストール、設定ファイル編集
    – 高可用性の設定（アクティブ・スタンバイ構成）
  - ウェブ・アプリケーション実行環境の導入と設定
    – Nginx の導入、設定、PHP-FPM (First CGI Process Manager)

– 監視エージェントの導入導入と設定自動化

ビジネスチャンスを逃さないために、サーバー構築の自動化は経営課題

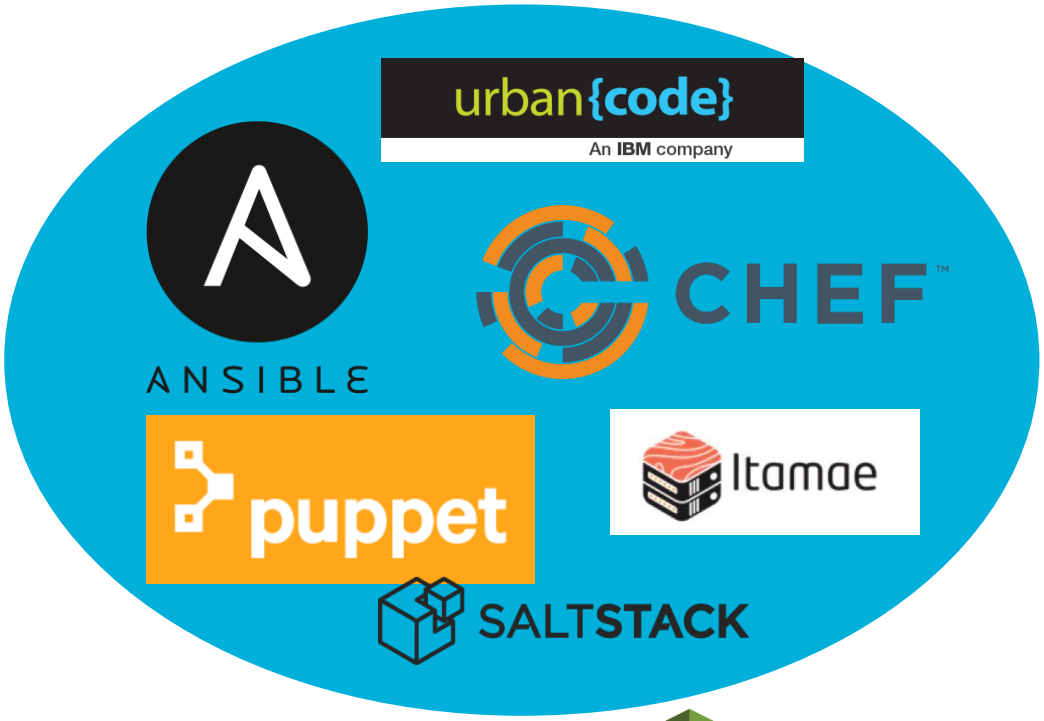
– 販売機会を逃さない事が、重要なのですが

ビジネスチャンスを逃さないために、サーバー構築の自動化は経営課題

競合他社に機動力で負けて、シェア争いに遅れを取るのは

自動化ツールは沢山あるけど今日はCHEFのはなし

- デプロイツール
  - Jenkins
  - Capistrano
- サーバー設定自動化ツール
  - ANSIBLE
  - CHEF
  - puppet
  - SALTSTACK
  - Itamae
  - Urban{code}
- AWS
  - Cloud Formation
  - OpsWorks



サーバー自動設定ツールで重要なキーワード

- 言葉はどうでも良いけど、特徴を理解してくださいね。

冪等性（べきとうせい）

サーバー自動設定ツールで重要なキーワード　冪等性

冪等性（べきとうせい、英: idempotence）は、

ある操作を

1回行っても

複数回行っても

結果が同じ

であることをいう概念

冪等性が、なんで重要な概念？

何回実行しても結果が同じという事は？

/etc/hosts に　IPアドレスとホスト名を追加するケースで考察

Shellで設定する場合

設定結果を示したシェル

```
#!/bin/bash

echo “192.168.10.11 mysql1” >>  /etc/hosts
```

shellで設定内容を反映

```
# config_hosts.sh
```

実行結果 (一回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

実行結果 (二回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
192.16.10.11 mysql1
```

Chefで設定する場合

設定結果を示したレシピ

```
hostsfile_entry ‘192.168.10.11' do
  hostname  ‘mysql1'
  action    :create_if_missing
end
```

chefで設定内容を反映

```
# chef-solo -o config_hosts
```

実行結果 (一回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

実行結果 (二回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

冪等性が、なんで重要な概念？

何回実行しても結果が同じという事は？

/etc/hosts に　IPアドレスとホスト名を追加するケースで考察

Shellで設定する場合

実行した回数だけ
追加されていく

実行結果 (一回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

実行結果 (二回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
192.16.10.11 mysql1
```

Chefで設定する場合

hostsファイル設定は、
何度実行しても
おかしく成らない

実行結果 (一回目）

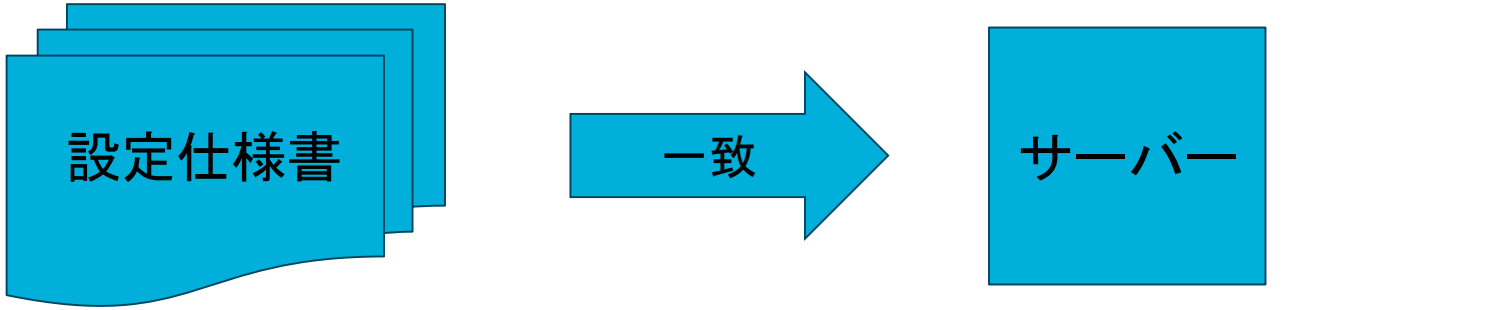
```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

実行結果 (二回目）

```
# cat /etc/hosts
< 中略 >
192.16.10.11 mysql1
```

自動設定ツールの冪等性とは

サーバーの設定仕様に基づいて、サーバーの設定状態一致させる
つまり、何回実行しても結果は同じ状態になる

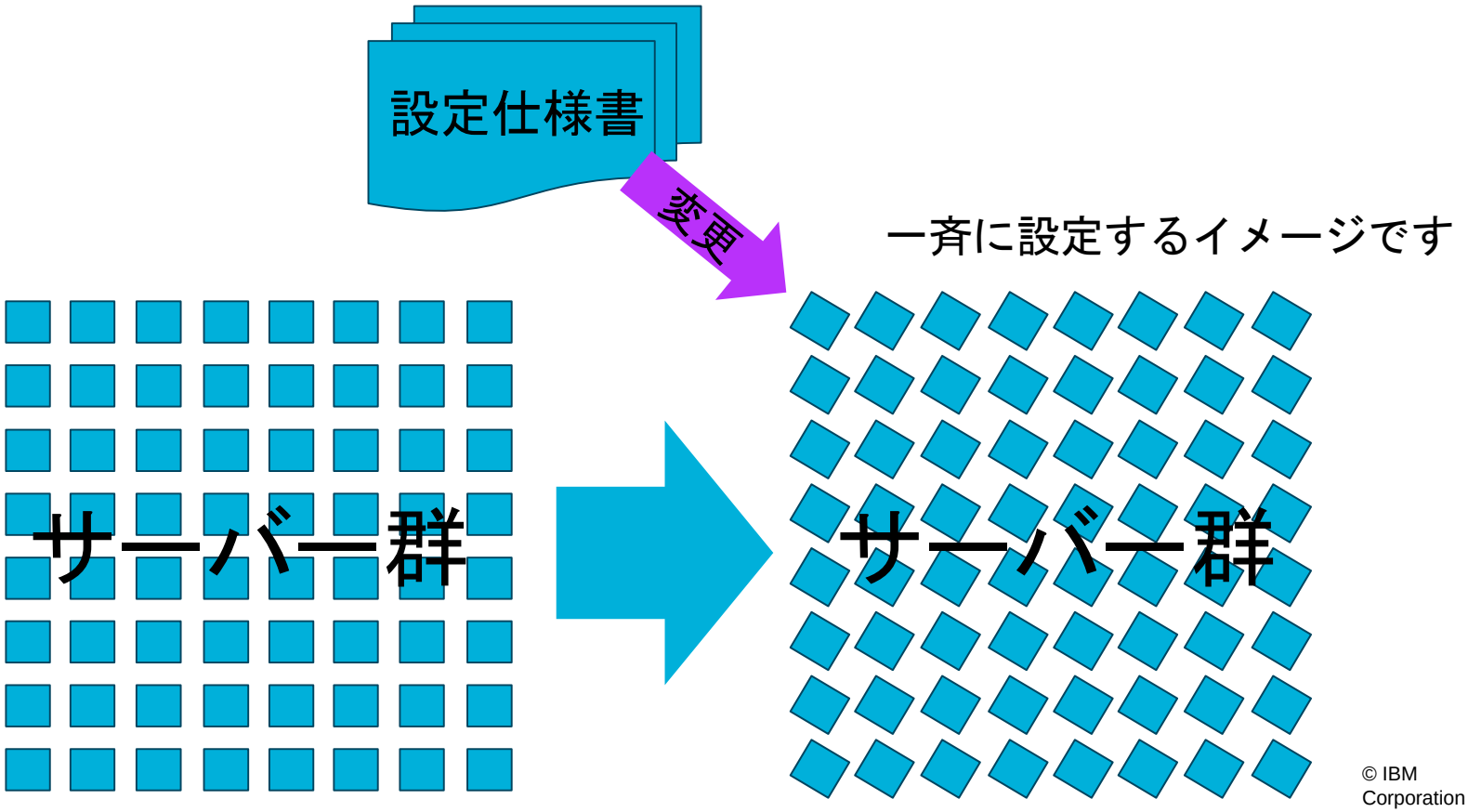


Chef では Cookbook
Ansibleでは Playbook

仮想サーバー
物理サーバー
クラウドのサーバー

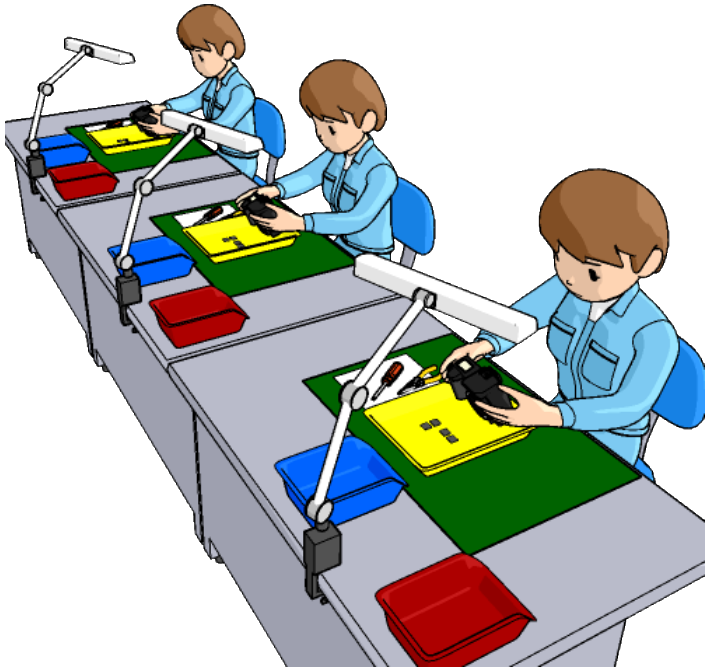
冪等性は設定変更に効果を発揮

サーバーの設定仕様に基づいて、サーバーの設定状態一致させる
つまり、何回実行しても結果は同じ状態なので、
大量のサーバー群の設定管理に威力を発揮する



サーバー設定自動ツールを使うという事は、手工業からロボットラインへ移行する様なこと

– ロボットをプログラムして、何度も同じ事をさせる行為に等しい



手作業の様なサーバー構築

人数が生産力に比例
優秀な技術者は簡単に育たない

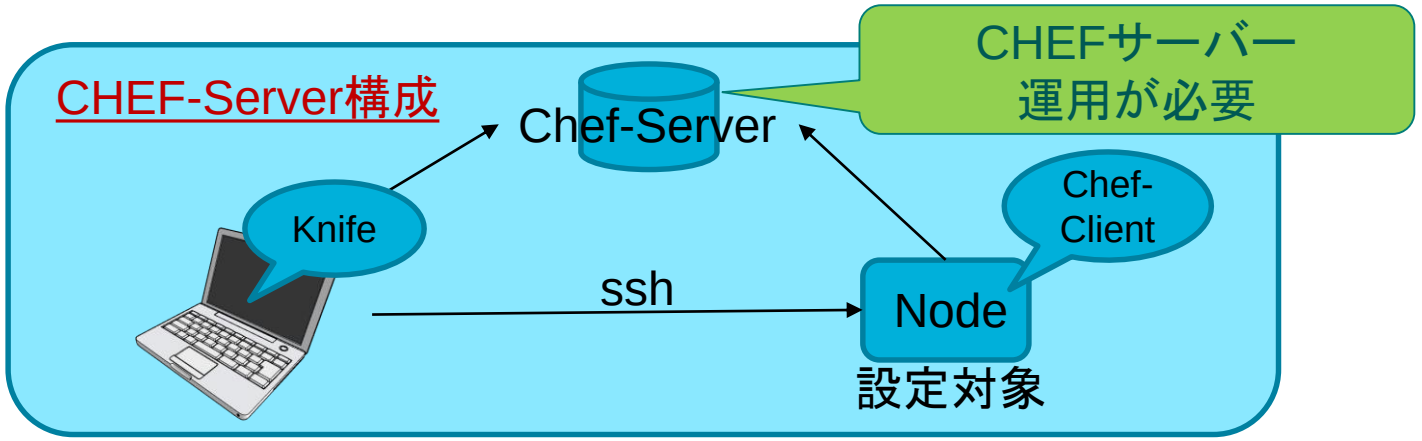
ロボットによるサーバー構築作業

ロボット数が生産力を決める
優秀な技術者のノウハウを真似

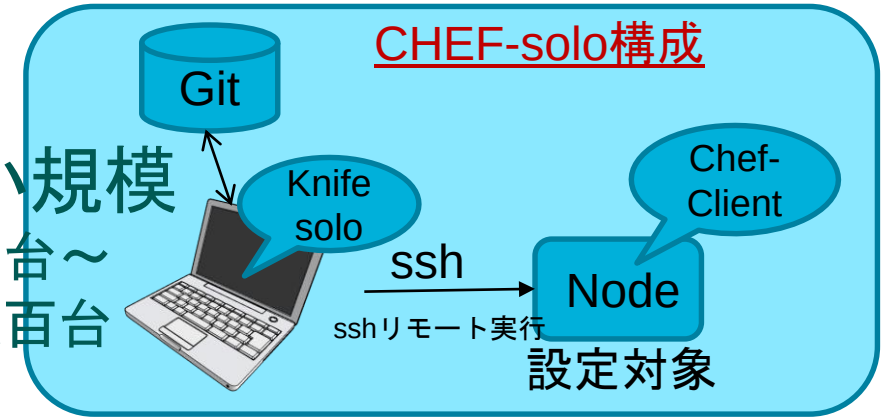
CHEFの適用パターン

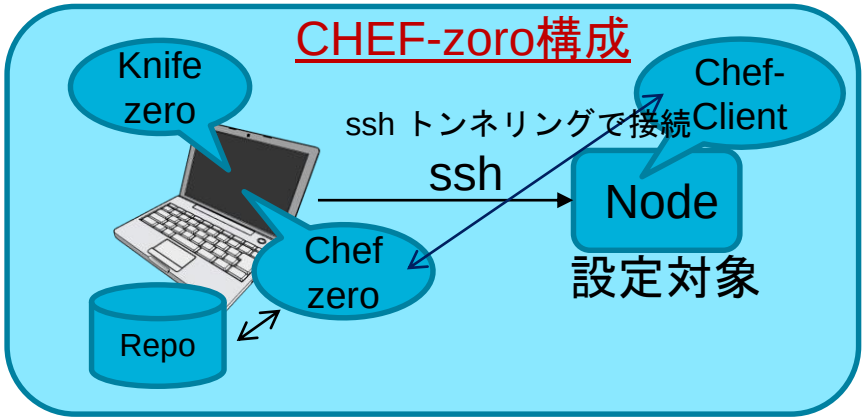
– 対象サーバーの台数規模に応じてパターンを選択

大規模
千台？～

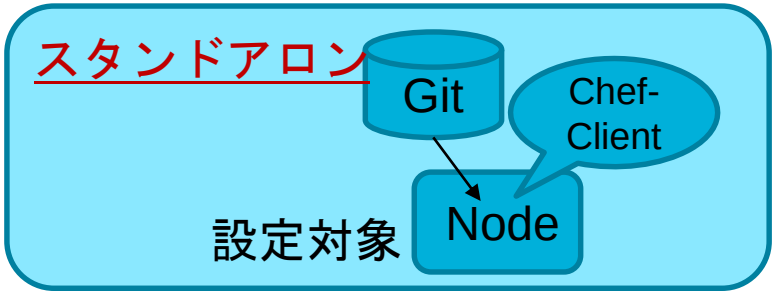


中小規模
数十台～
数百台





Cookbook
開発環境



ここで、少し話が変わりますが. . .

ChefやAnsibleの解説本には

必ずVagrant

が登場するのですが

これは何ですか？

CHEFの適用パターン

– 対象サーバーの台数規模に応じてパターンを選択

大規模
千台？～

CHEF-Server構成
Chef-Server
Knife
Chef-Client
Node
設定対象

CHEFサーバー
運用が必要

中小規模
数十台～
数百台

CHEF-solo構成
Git
Knife solo
Chef-Client
ssh
Node
sshリモート実行
設定対象

CHEF-zero構成
Knife zero
Chef-Client
sshトンネリングで接続
Chef zero
Repo
Node
設定対象

Chefの設定仕様書を開発するには、
OSフレッシュ・インストール状態や
OSカスタム・イメージの状態から
何度も実行してテストできる
自由に試せるサーバー環境が必須

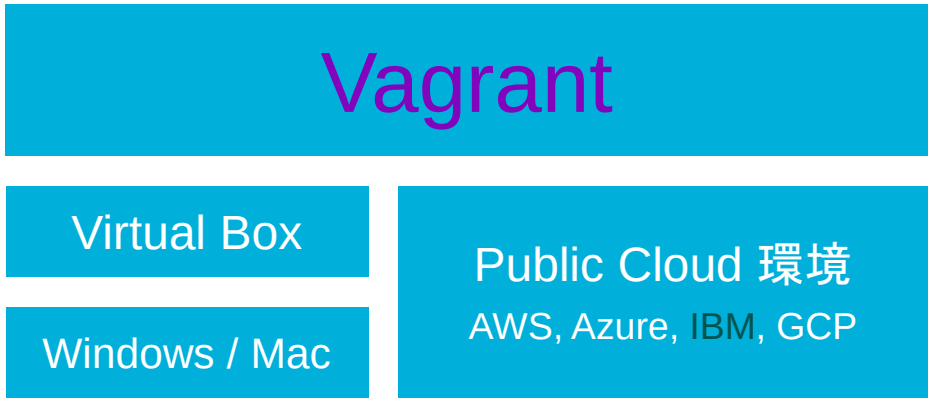
Cookbook
開発環境

スタンドアロン
Git
Chef-Client
Node
設定対象

Vagrantとは？

- 仮想サーバーを OS を指定して起動したり、削除したりできるOSSのツール
- 対象は、パソコンの仮想環境でも、パブリッククラウド環境でも良い
- 無料で利用できる Virtual Box を利用するのがお得
- もちろん、IBM Bluemix Infrastructure の仮想サーバーも利用可能（有料）

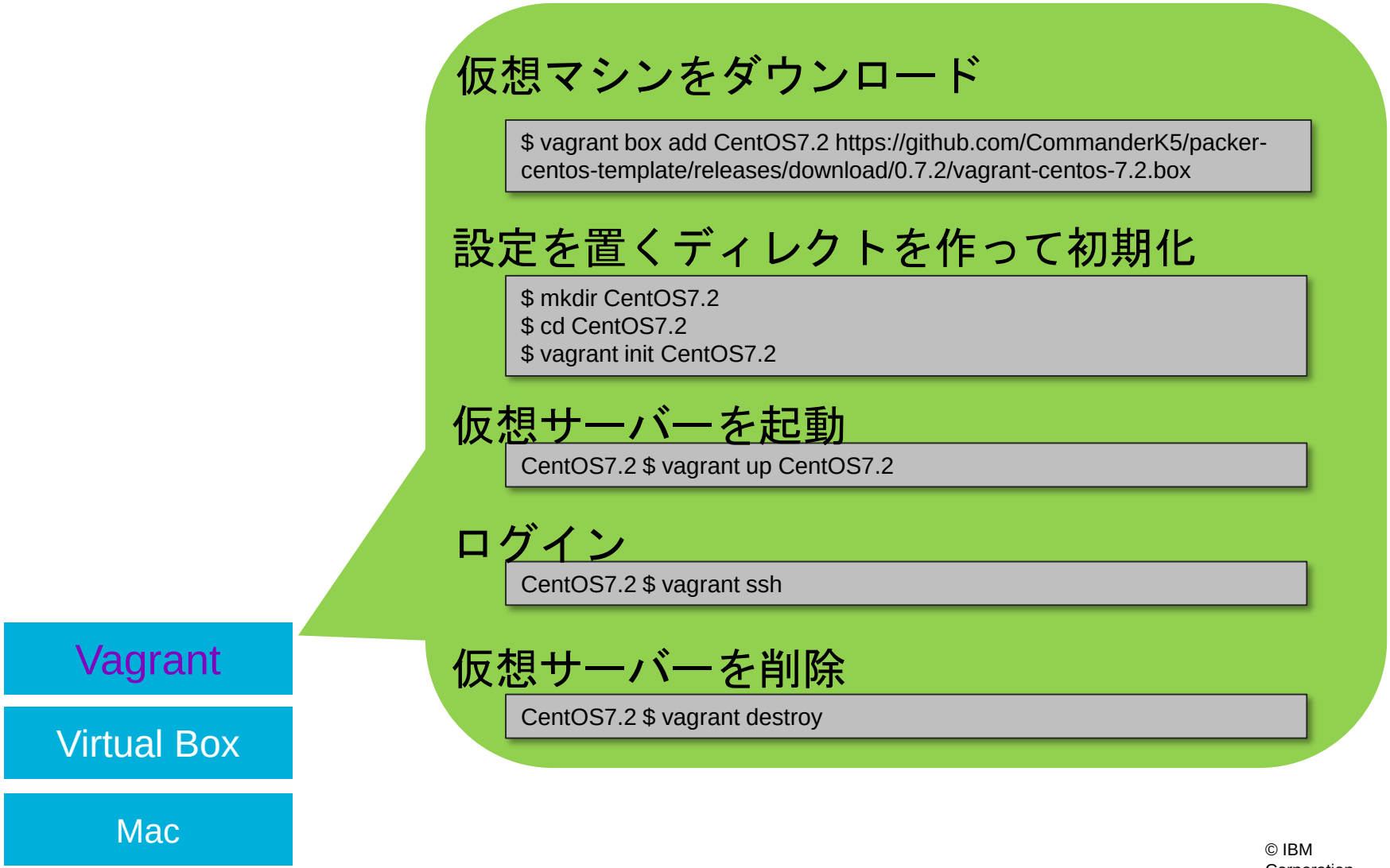
https://www.virtualbox.org/wiki/Downloads



https://www.vagrantup.com/

Vagrantとは？

CHEFのCookbook を開発するために最低限必要なVagrant コマンド



CHEFで設定できるサーバーOSは？

- メジャー Linuxディストリビューション　Debian/Red hat/CentOS/Ubuntu
- Microsoft サーバーOS　/ クライアントOS, Apple Mac OS X
- IBM ハイエンドサーバー／メインフレーム

Linux Intel Architecture

Debian 8
Debian 7
Debian 6
Red Hat Enterprise Linux 7 / CentOS7
Red Hat Enterprise Linux 6 / CentOS6
Red Hat Enterprise Linux 5 / CentOS5
Ubuntu 16.04
Ubuntu 14.04
Ubuntu 12.04

Microsoft

Windows Server OS

Windows 2008r2
Windows 2012r2

Windows OS

Windows 10
Windows 8.1
Windows 8
Windows 7

Apple

Mac OS X

macOS 10.11
macOS 10.10
macOS 10.9

Unix Intel Architecture

FreeBSD 10
FreeBSD 9

IBM

PowerPC Architecture

IBM AIX 7.1
IBM AIX 6.1

IBM Mainfame Architecture

Red Hat Enterprise Linux 7/6
SUSE Linux Enterprise Server 11/12

Oracle

SPARC/i386 Architecture

Solaris 11
Solaris 10

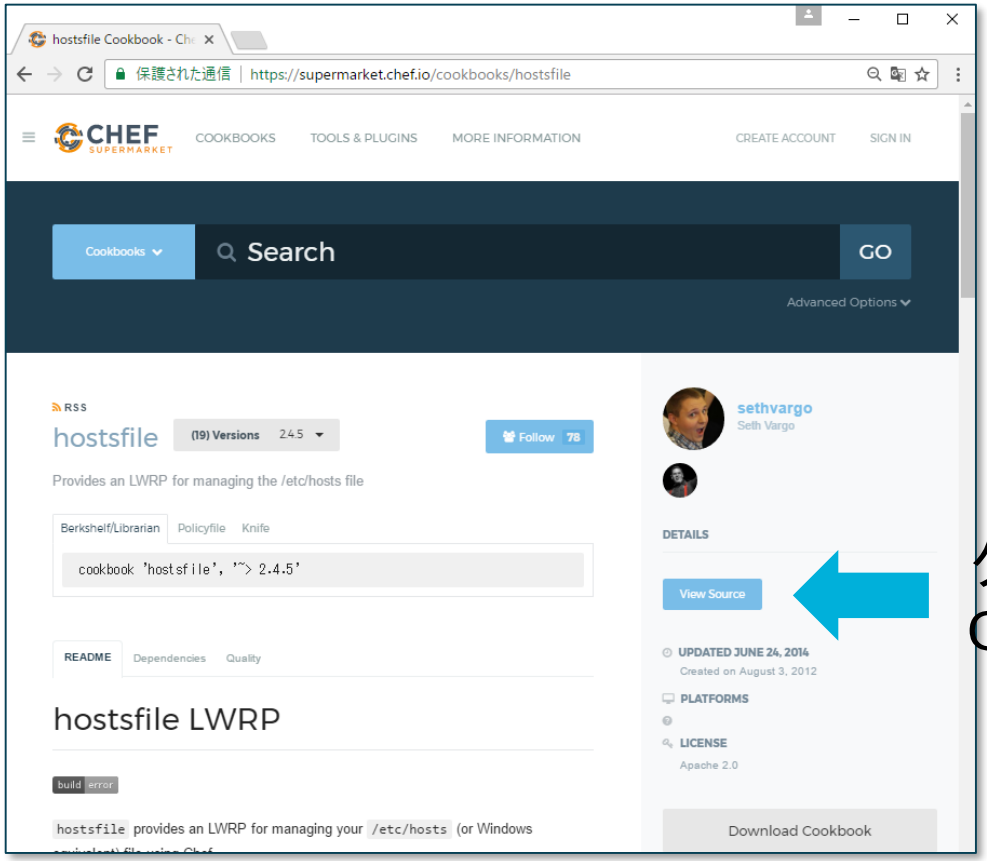
Chefの簡単なCookbookを開発してみましょう

- 開発物　hostsファイルを編集するCookbook
- 作業の手順
  - vagrant で CentOS7.2 の 仮想サーバーを立ち上げる
    - vagrant init CentOS7.2
    - vagrant up
  - CentOS にログインして、Chef Development Kit をインストールする
    - vagrant ssh
    - sudo –s
    - rpm -i https://packages.chef.io/files/stable/chefdk/1.2.22/el/7/chefdk-1.2.22-1.el7.x86_64.rpm
  - リポジトリを作成して、クックブックを作成する
    - mkdir -p /var/chef/cookbooks
    - cd /var/chef/cookbooks

Chefの簡単なCookbookを開発してみましょう

- 前頁からの続き
- 作業の手順
  - Chef の Super Market から使えそうなクックブックを探してくる　2017/2/22現在 3203件
  - 以下は /etc/hosts のファイルを編集するクックブック

https://supermarket.chef.io/



クリックして
GitHubを表示

Chefの簡単なCookbookを開発してみましょう

- 前頁からの続き
- 作業の手順
  - hostsfile の クックブックをリポジトリにコピーする
    - git clone https://github.com/customink-webops/hostsfile
  - このクックブックのウェブページに従ってレシピを追加する

hostsfile/recipes/default.rb

```
hostsfile_entry '192.168.33.11' do
  hostname  'server1'
  action    :create_if_missing
end
hostsfile_entry '192.168.33.12' do
  hostname  'server2'
  action    :create_if_missing
end
hostsfile_entry '192.168.33.13' do
  hostname  'server3'
  action    :create_if_missing
end
```

レシピの適用結果

```
127.0.0.1 localhost

192.168.3.11    zabbix_server
192.168.3.51    nfs_server1
192.168.3.52    nfs_server2
192.168.3.61    backup_server1


192.168.33.11   server1
192.168.33.12   server2
192.168.33.13   server3
```

既存へ影響を与えない

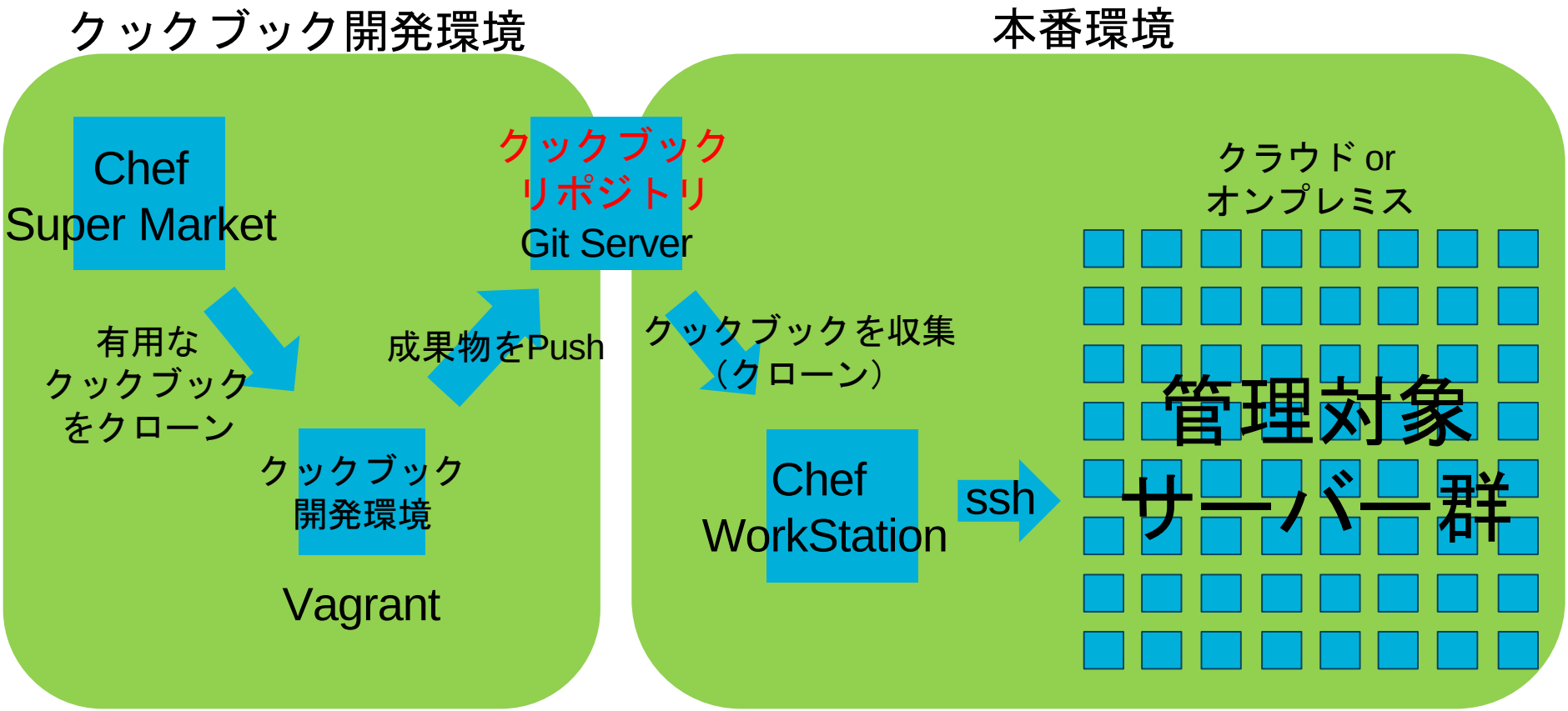
存在しなければエントリを追加

Chefの簡単なCookbookを開発してみましょう

- 前頁からの続き
- 作業の手順
  - ローカル環境でテスト、chef-solo で　クックブックを指定して実行
    - chef-solo -o hostsfile
  - Hostsファイルの内容を確認
    - cat /etc/hosts
  - 完成したクックブックをGitHubへ登録
    - GitHub に リポジトリ追加　hostsfile (GitHubにアカウントを持っている必要あり)
    - cd /var/chef/cookbooks/hostsfile
    - rm –fr .git　(プライベートのGitサーバーに登録するため)
    - git init
    - git add –all
    - git commit –m “first commit”
    - git remote add origin https://github.com/takara9/hostfile.git （プライベートのGitサーバー）
    - git push -u origin master
      - Username for…
      - Password for…

完成したクックブックで、他のサーバー群に設定してみましょう。

– イメージにすると、以下の様になります。

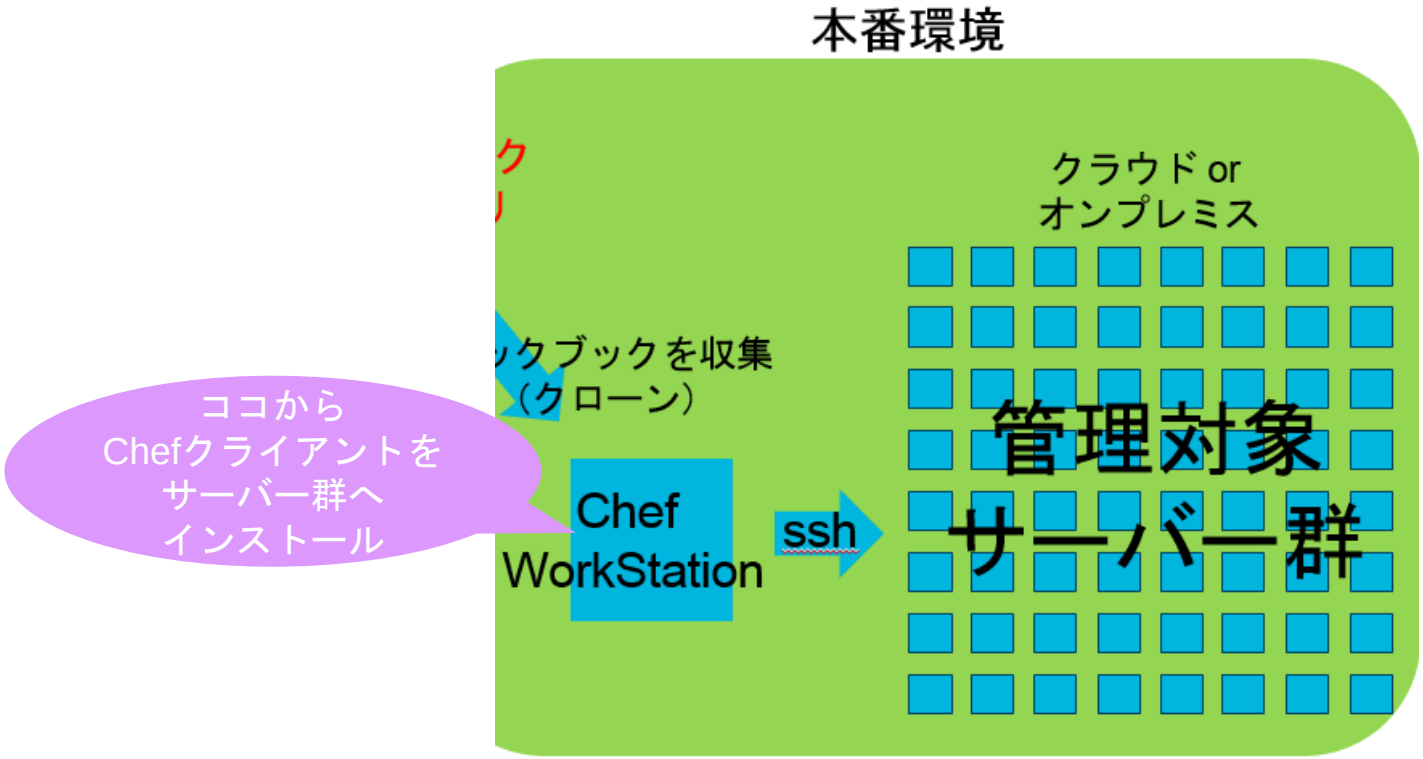


完成したクックブックで、他のサーバー群に設定してみましょう。

- 管理対象サーバー群は、Chef workstation から root ユーザで sshがリモート実行できる様になっていることが前提
- Chef workstation をセットアップします
  - Chef Developer tool kit をインストール
    - rpm -i https://packages.chef.io/files/stable/chefdk/1.2.22/el/7/chefdk-1.2.22-1.el7.x86_64.rpm
  - Knife solo をインストール
    - /opt/chefdk/embedded/bin/gem install knife-solo --no-ri --no-rdoc
  - 専用リポジトリを作成
    - knife solo init my-repo
  - クックブックのリポジトリからChef workstation にコピー
    - git clone https://github.com/takara9/hostsfile　などなど

完成したクックブックで、他のサーバー群に設定してみましょう。

- 前ページからの続き
- 管理対象サーバーをChef workstationへ登録とChef クライアントをインストール
  - 管理対象サーバー数だけ、IPアドレスを指定してroot で実行
    - knife solo prepare root@192.168.33.1 。。。



完成したクックブックで、他のサーバー群に設定してみましょう。

- 前ページからの続き
- サーバー（ノード）ごとの設定を編集
  - my-repo/nodes の中に生成されたJSON形式のファイル
  - 冪等性の特性を生かして、一度に完成ではなく、必要に応じて追加、何度でも実行

今回のサンプル

```
{
  “run_list”: [
    “hostsfile”
  ],
  “automatic”: {
    “ipaddress”: “192.168.33.12”
  }
}
```

累進的に追加も可

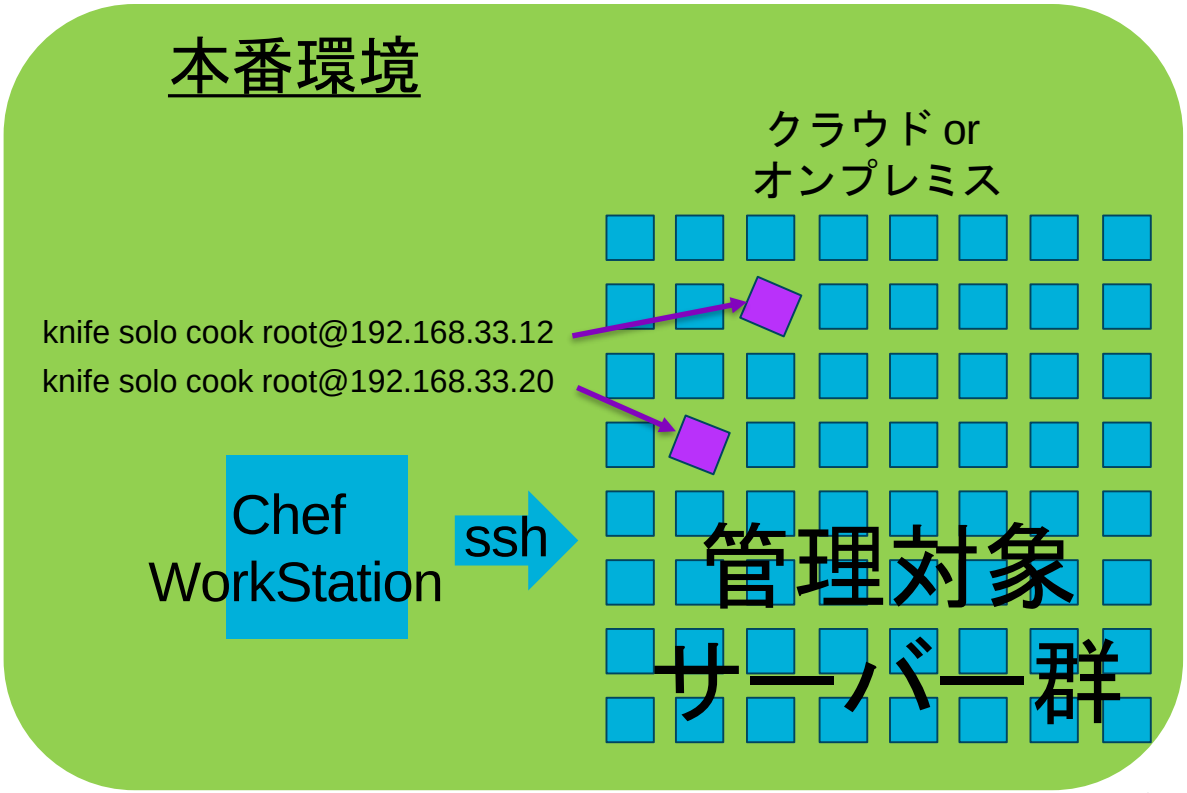
```
{
  “run_list”: [
    “os_config_common”,
    “hostsfile”,
    “zabbix_agent”,
    “nginx_fpm”
  ],
  “automatic”: {
    “ipaddress”: “192.168.33.12”
  }
}
```

設定に必要なCookbookを追加する

対象サーバーのIPアドレス

完成したクックブックで、他のサーバー群に設定してみましょう。

- 前ページからの続き
- 管理対象サーバー（ノード）に設定を実施
  - knife solo cook root@192.168.33.12



CHEFの概要と利用する価値について具体的にイメージできたでしょうか？

– 次はWordpressを使ったウェブサイトの構築例

インフラ技術者
に代って作業
するロボット

- 省コスト
- 人的ミス排除
- スピードアップ

具体的な実行例　WordPressのHP構築

– Cookbookのアセットがあれば、高い生産性を得られる例

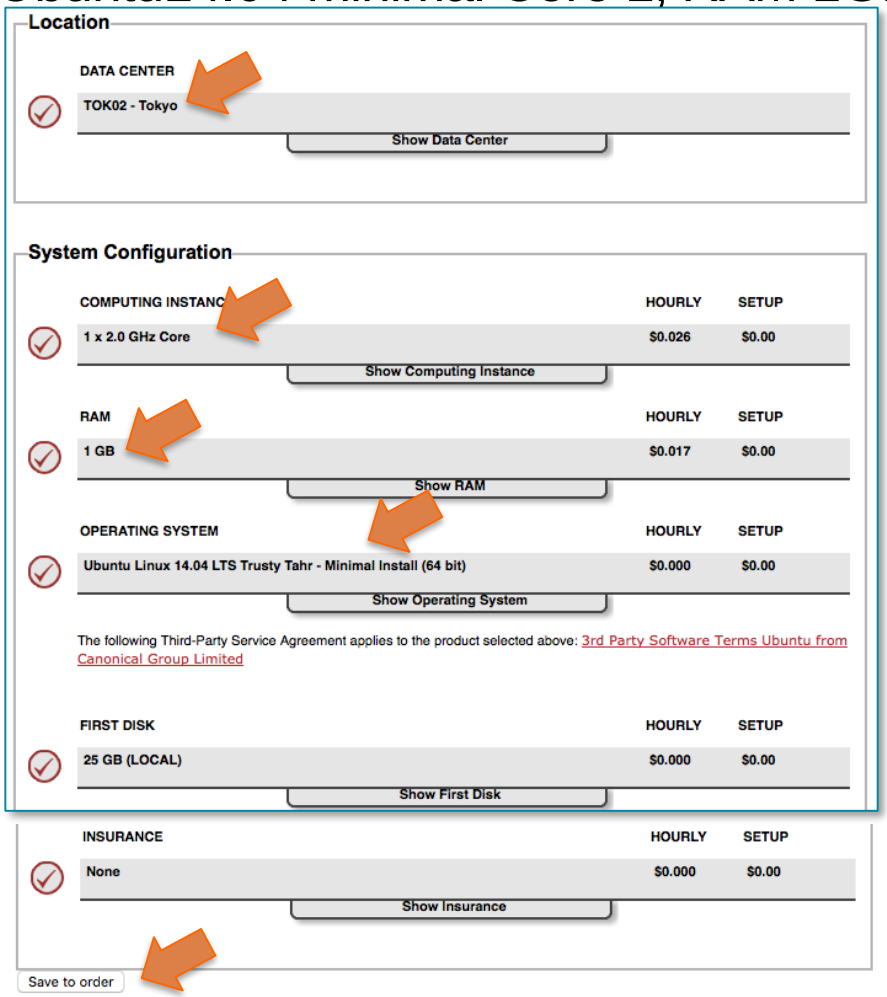
– CHEFを利用する場合の手順と時間
  - ポータルから仮想サーバーをオーダー (5分）
  - CHEFのコマンドを実行（2分）
    – クックブック (https://github.com/takara9/wordpress01)
  - WebページからWordPressの初期設定　(2分）

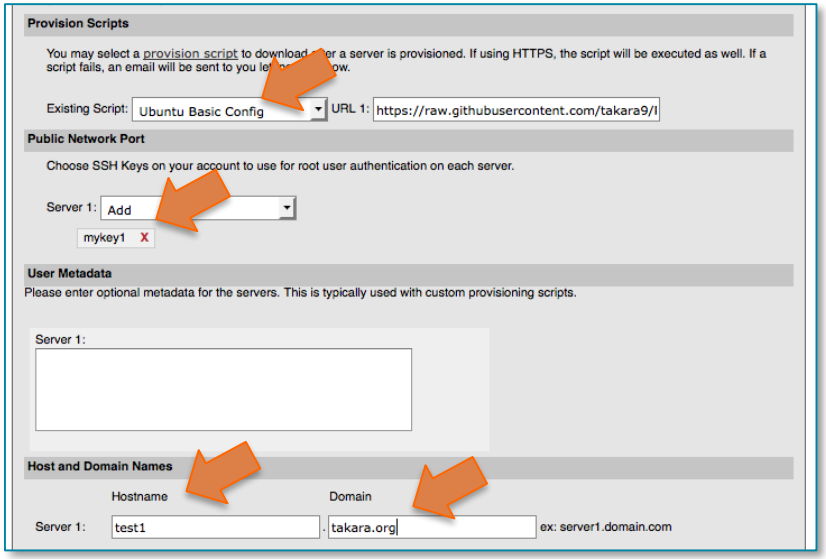
– CHEFのコマンドか実行する内容
  - Ubuntu リポジトリから最新状態に更新
  - nginx と php-fpm の導入と設定
  - MySQL の導入と設定（セキュアインストール）
  - WordPressの最新版の導入

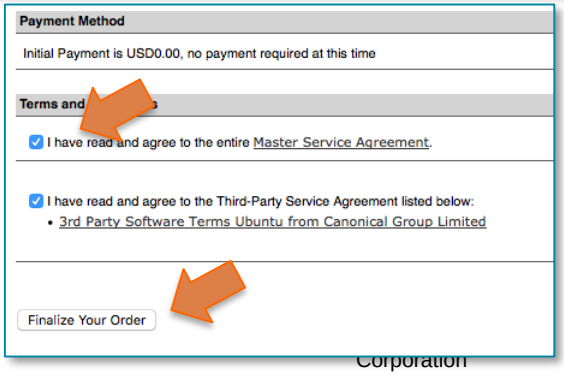
これだけの作業を
手作業で実施する
手間と時間を
考えてみてください

ポータルから仮想サーバーをオーダー

– Bluemix Infrastructure のポータルから
– Ubuntu14.04 minimal Core 1, RAM 1GB







仮想サーバーへログインしてCHEFを実行

– 構築コマンド実行手順

```
# curl -L https://www.opscode.com/chef/install.sh | bash
# knife cookbook create dummy -o /var/chef/cookbooks
# git -C /var/chef/cookbooks clone https://github.com/takara9/wordpress01
# chef-solo -o wordpress01
```

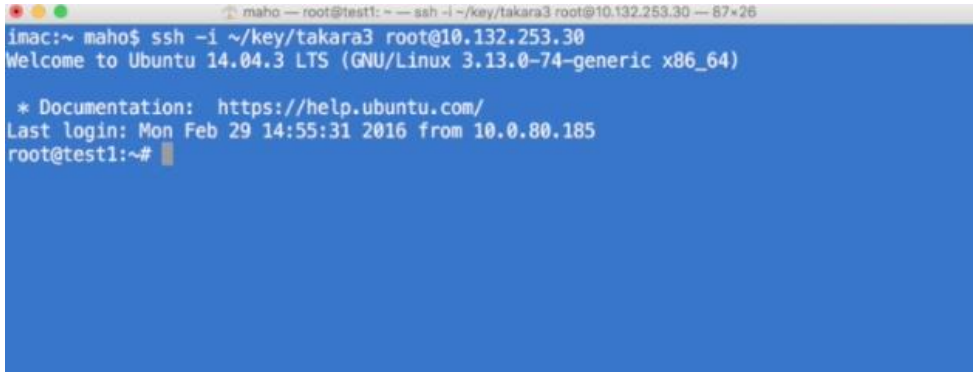
①CHEFインストール

②フォルダ作成

③クックブック取得

④クックブック適用

実行の様子をビデオでご覧下さい



ビデオは、Youtubeにあります。

YouTube JP https://www.youtube.com/watch?v=xxUc7vRjW5k

Cookbook適用後の作業

– インストール後の初期画面

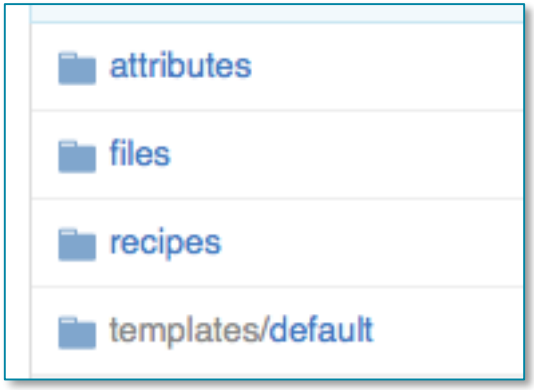
WP管理画面

完了！

CHEF クックブックの解説 1

– Cookbookのディレクトリ

/var/chef/cookbook/wordpress01



パラメータ　導入パス、ユーザーID、パスワードなど

そのまま置く設定ファイルやシェルなど

レシピ　設定の手順

固有パラメータなどを設定して配置するファイル

CHEF クックブックの解説 2

– Attribute のディレクトリ

/var/chef/cookbook/wordpress01

Branch: **master** ▾ **wordpress01** / attributes / **default.rb**

chef add function

0 contributors

6 lines (5 sloc) | 230 Bytes

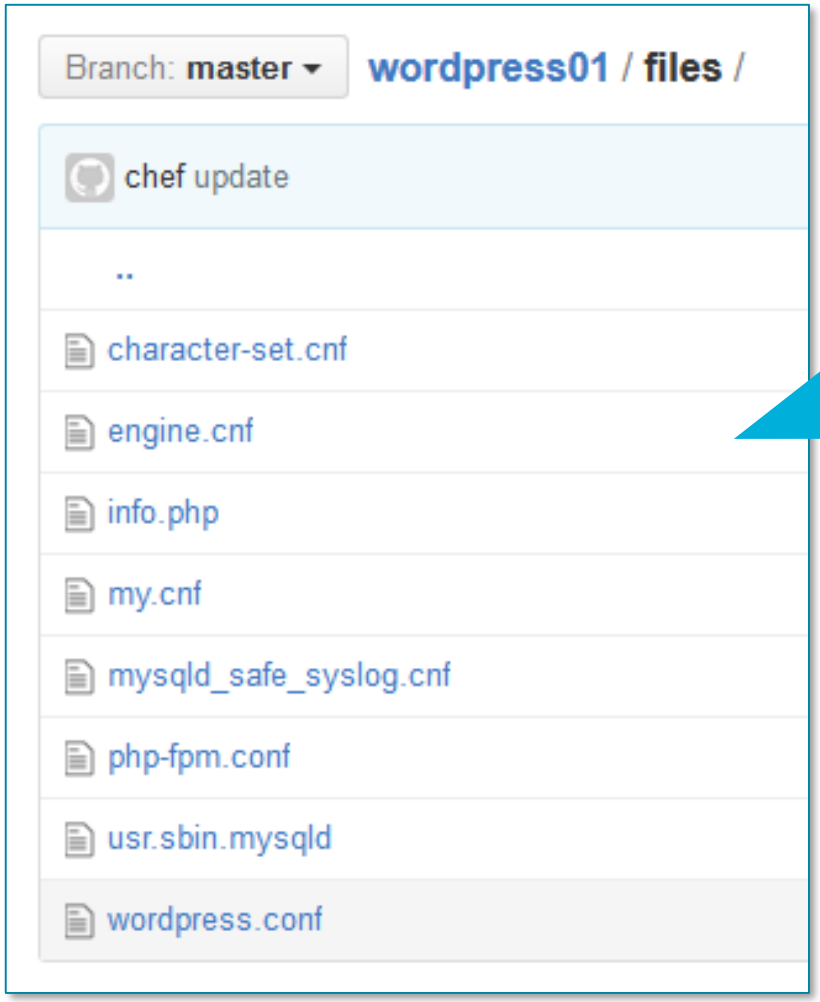
```
1  default["mysql"]["root_password"] = 'passw0rd'
2  default["mysql"]["db_name"] = 'wordpress'
3  default["mysql"]["user"]["name"] = 'wordpress'
4  default["mysql"]["user"]["password"] = 'wordpress'
5  default["mysql"]["hostname"] = 'localhost'
```

MySQLの固有項目を
アトリビュート
として設置

CHEF クックブックの解説 3

– files は　設定ファイルのひな型の置き場所

/var/chef/cookbook/wordpress01/files



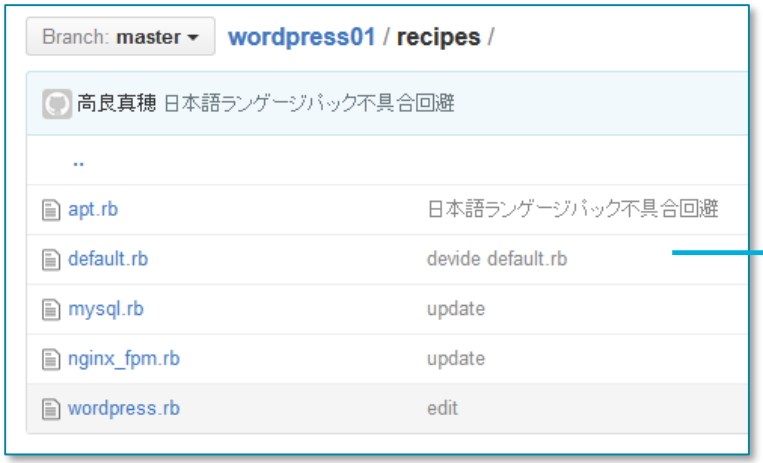
Nginx, MySQL,
PHP-FPMなどの
設定ファイル群

レシピから所定の場所に
コピーする

CHEF クックブックの解説 4

– Recipes は設定手順の本体です

/var/chef/cookbook/wordpress01/recipes



default.rb

```
9  include_recipe "wordpress01::apt"
10 include_recipe "wordpress01::nginx_fpm"
11 include_recipe "wordpress01::mysql"
12 include_recipe "wordpress01::wordpress"
```

クックブックの組み合わせ
だけでなく、
レシピの組み合わせも出来るよ

CHEF クックブックの解説 5

– Nginx_fpm の要所説明

/var/chef/cookbook/wordpress01/recipes/nginx_fpm.rb

パッケージを追加インストール

```
8  %w{
9    nginx
10   php5-fpm
11   php5-memcached
12   php5-mysqlnd-ms
13   mysql-client
14 }.each do |pkgname|
15   package "#{pkgname}" do
16     action :install
17   end
18 end
```

パラーメータ値をセットしてファイルを置く

```
50 # config mysql
51 mysql_hostname       = node["mysql"]["hostname"]
52 mysql_user_name      = node["mysql"]["user"]["name"]
53 mysql_user_password = node["mysql"]["user"]["password"]
54
55 template "/etc/php5/fpm/php.ini" do
56   source "php.ini.erb"
57   owner "root"
58   group "root"
59   mode 0644
60
61   variables({
62     :hostname => mysql_hostname,
63     :username => mysql_user_name,
64     :password => mysql_user_password,
65   })
66   notifies :restart, "service[php5-fpm]"
67 end
```

設定ファイルを置く

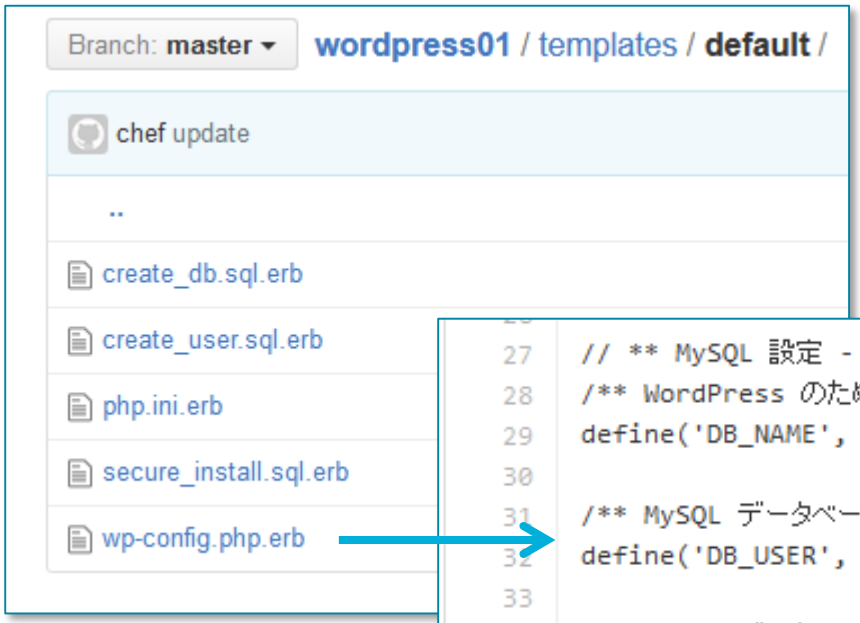
```
34 cookbook_file "/etc/nginx/sites-enabled/wordpress.conf" do
35   source "wordpress.conf"
36   owner "root"
37   group "root"
38   mode 0644
39   notifies :restart, "service[nginx]"
40 end
```

CHEF クックブックの解説 6

– template

/var/chef/cookbook/wordpress01/recipes/nginx_fpm.rb

テンプレートファイル

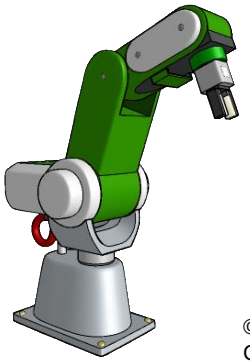


Attribute の値で置き換えられる

```
// ** MySQL 設定 - この情報はホスティング先から入手してください。 ** //
/** WordPress のためのデータベース名 */
define('DB_NAME', '<%= @db_name %>');

/** MySQL データベースのユーザー名 */
define('DB_USER', '<%= @username %>');

/** MySQL データベースのパスワード */
define('DB_PASSWORD', '<%= @password %>');

/** MySQL のホスト名 */
define('DB_HOST', 'localhost');
```

Chefを使って得た教訓

- Chefは、どんどん変わるので、半年前に作った手順が動かなくなる。
  - コードや機能の改善が活発なので注意
- プロジェクトで使う場合は、Chefのバージョンを決めて、自プロジェクトが管理するサーバーから、Chefのパッケージをダウンロードする
- RubyはChefパッケージに入っているのでインストールしない。
  - OSのRubyとGemインストールでChefを導入すると、バージョン問題で苦労するから回避
- クックブックの開発には、Vagrant が必須のツール
- Chefの運用には、Gitサーバーも必須の基盤
- CHEFで難しい事にチャレンジすると、コスパが悪くなるので、簡単なことからドンドン利用して、時間とコストを節約
- Chef Supermarket のアセットを活用して、時間を節約
  - 出来るだけ自分でクックブックを書かない



お話のまとめ

– スマホ・アプリが普及し、パブリック・クラウドが一般化した現在、サーバー設定の自動化は、ビジネスチャンスを掴むための経営課題

– スピードアップ、品質改善、生産性向上を目的として、適材適所に適用

– Chefの冪等性とは、
  - 繰り返し実行しても悪影響がない
  - 設定の変更、修正、追加などの都度、気軽に実行していける

– Chef 運用の3要素
  - クックブック開発環境
  - クックブックGitリポジトリ
  - Chefワークステーション　(Chefサーバーを持たなくても100台程度なOK)

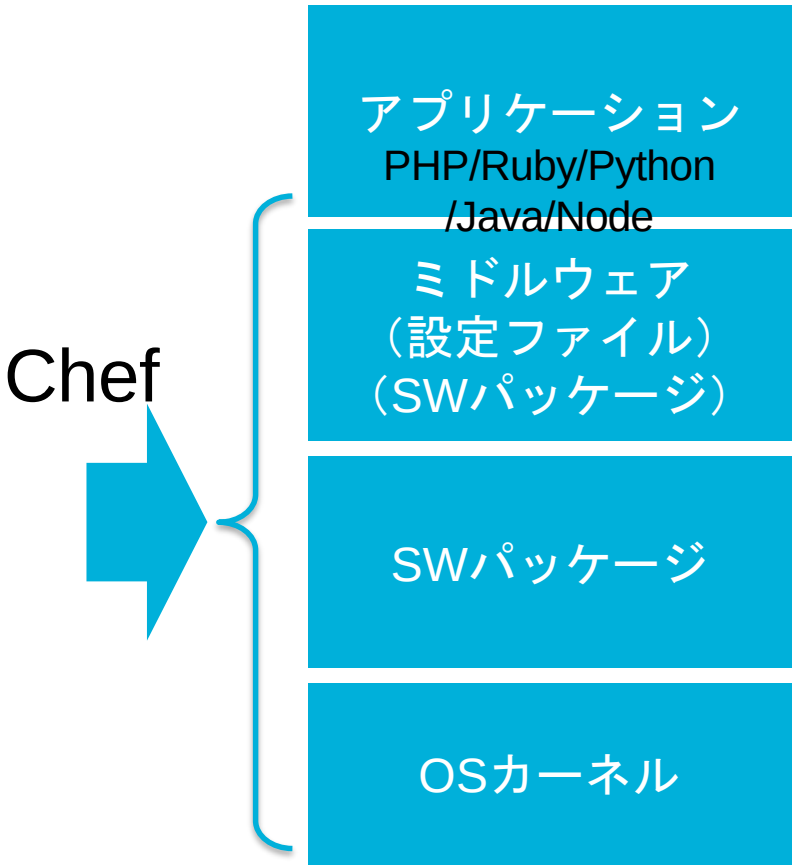
– 俗人的ノウハウ、複雑な設定手順をコード化してビジネスに貢献

ところで…
Dockerとはどんな関係？

たびたび頂く質問

ところで Docker との関係は？

– ChefはSWパッケージのバージョンを管理
– でも、これには重大な悩みがある？！



/bin/sh
config(nginx) = 1:1.10.2-1.el7
libc.so.6()(64bit)
libc.so.6(GLIBC_2.10)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.3.2)(64bit)
libc.so.6(GLIBC_2.3.4)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
libcrypt.so.1()(64bit)
libcrypt.so.1(GLIBC_2.2.5)(64bit)
libcrypto.so.10()(64bit)
libcrypto.so.10(OPENSSL_1.0.1)(64bit)
libcrypto.so.10(OPENSSL_1.0.1_EC)(64bit)
libcrypto.so.10(libcrypto.so.10)(64bit)
libdl.so.2()(64bit)
libdl.so.2(GLIBC_2.2.5)(64bit)
libpcre.so.1()(64bit)
libprofiler.so.0()(64bit)
libpthread.so.0()(64bit)
libpthread.so.0(GLIBC_2.2.5)(64bit)
libssl.so.10()(64bit)
libssl.so.10(libssl.so.10)(64bit)
libz.so.1()(64bit)
nginx-all-modules = 1:1.10.2-1.el7
nginx-filesystem
nginx-filesystem = 1:1.10.2-1.el7
openssl
pcre
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(FileDigests) <= 4.6.0-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rtld(GNU_HASH)
systemd

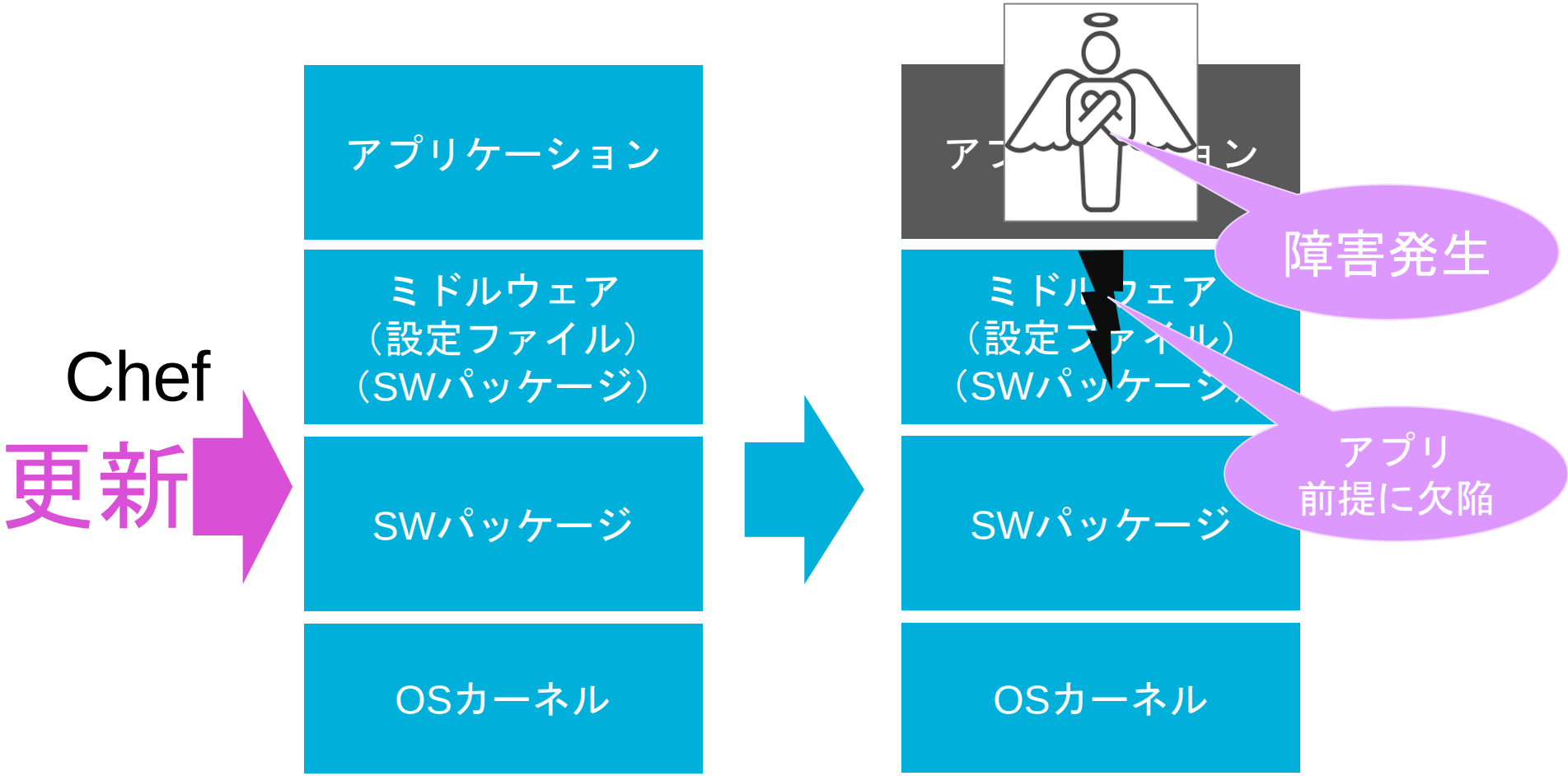
Nginxの依存パッケージ

httpd-mmn = 20120211x8664
php-common(x86-64) = 5.4.16-42.el7
php-cli(x86-64) = 5.4.16-42.el7
httpd
rpmlib(FileDigests) <= 4.6.0-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
libbz2.so.1()(64bit)
libcom_err.so.2()(64bit)
libcrypto.so.10()(64bit)
libcrypto.so.10(libcrypto.so.10)(64bit)
libcrypto.so.10(OPENSSL_1.0.1)(64bit)
libcrypt.so.1()(64bit)
libc.so.6()(64bit)
libc.so.6(GLIBC_2.11)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.15)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
libc.so.6(GLIBC_2.8)(64bit)
libdl.so.2()(64bit)
libdl.so.2(GLIBC_2.2.5)(64bit)
libgmp.so.10()(64bit)
libgssapi_krb5.so.2()(64bit)
libk5crypto.so.3()(64bit)
libkrb5.so.3()(64bit)
libm.so.6()(64bit)
libm.so.6(GLIBC_2.2.5)(64bit)
libnsl.so.1()(64bit)
libpcre.so.1()(64bit)
libresolv.so.2()(64bit)
libresolv.so.2(GLIBC_2.2.5)(64bit)
librt.so.1()(64bit)
libssl.so.10()(64bit)
libssl.so.10(libssl.so.10)(64bit)
libxml2.so.2()(64bit)
libxml2.so.2(LIBXML2_2.4.30)(64bit)
libxml2.so.2(LIBXML2_2.5.2)(64bit)
libxml2.so.2(LIBXML2_2.6.0)(64bit)
libxml2.so.2(LIBXML2_2.6.11)(64bit)
libxml2.so.2(LIBXML2_2.6.5)(64bit)
libxml2.so.2(LIBXML2_2.9.0)(64bit)
libz.so.1()(64bit)
rtld(GNU_HASH)
rpmlib(PayloadIsXz) <= 5.2-1

PHPの依存パッケージ

ところで Docker との関係は？

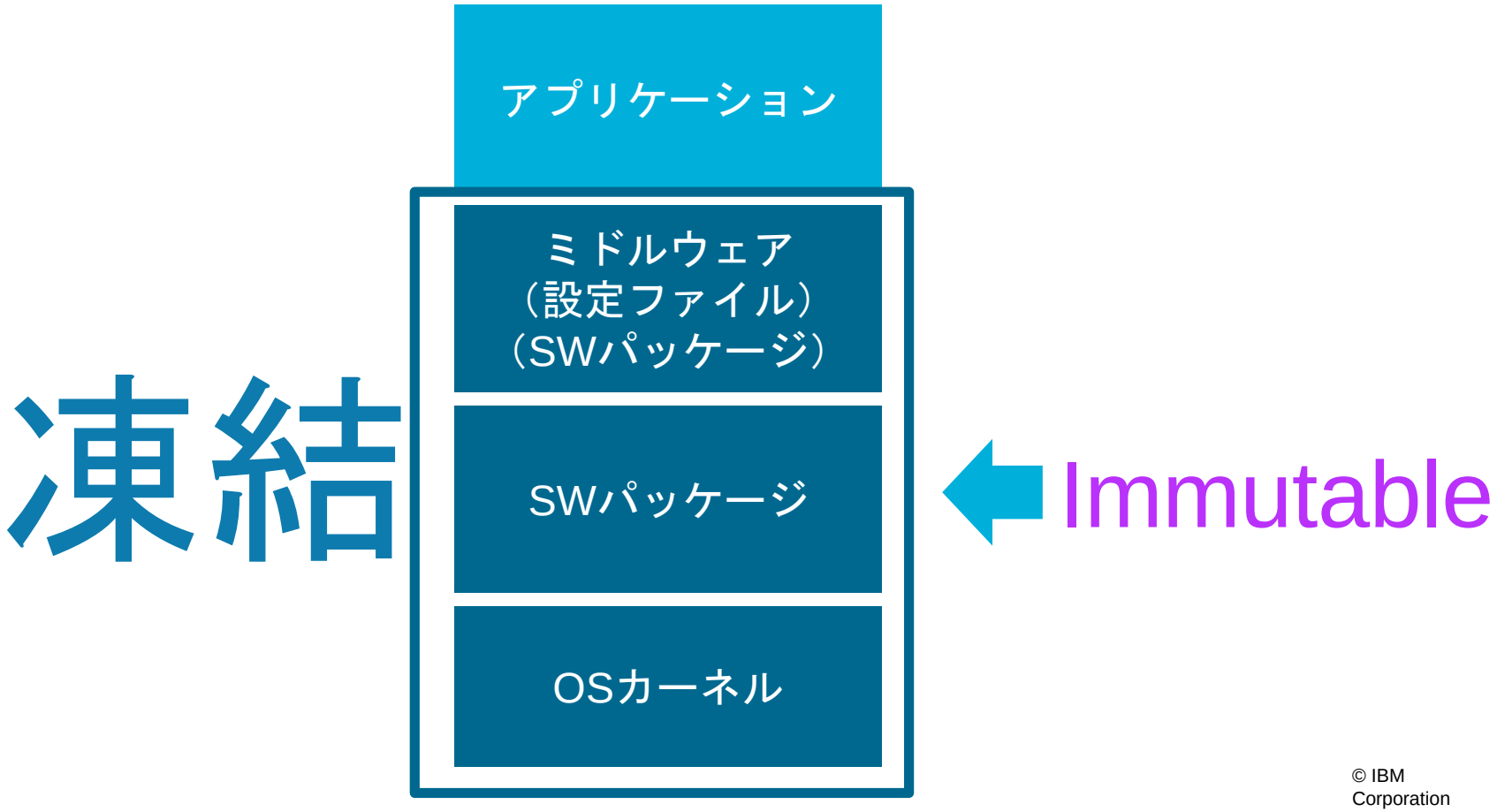
– 膨大な量のスタックの上で、アプリが稼動している事実
– スタックの更新で障害が出るか検証不可能、管理不可能、予知不可能



ところで Docker との関係は？

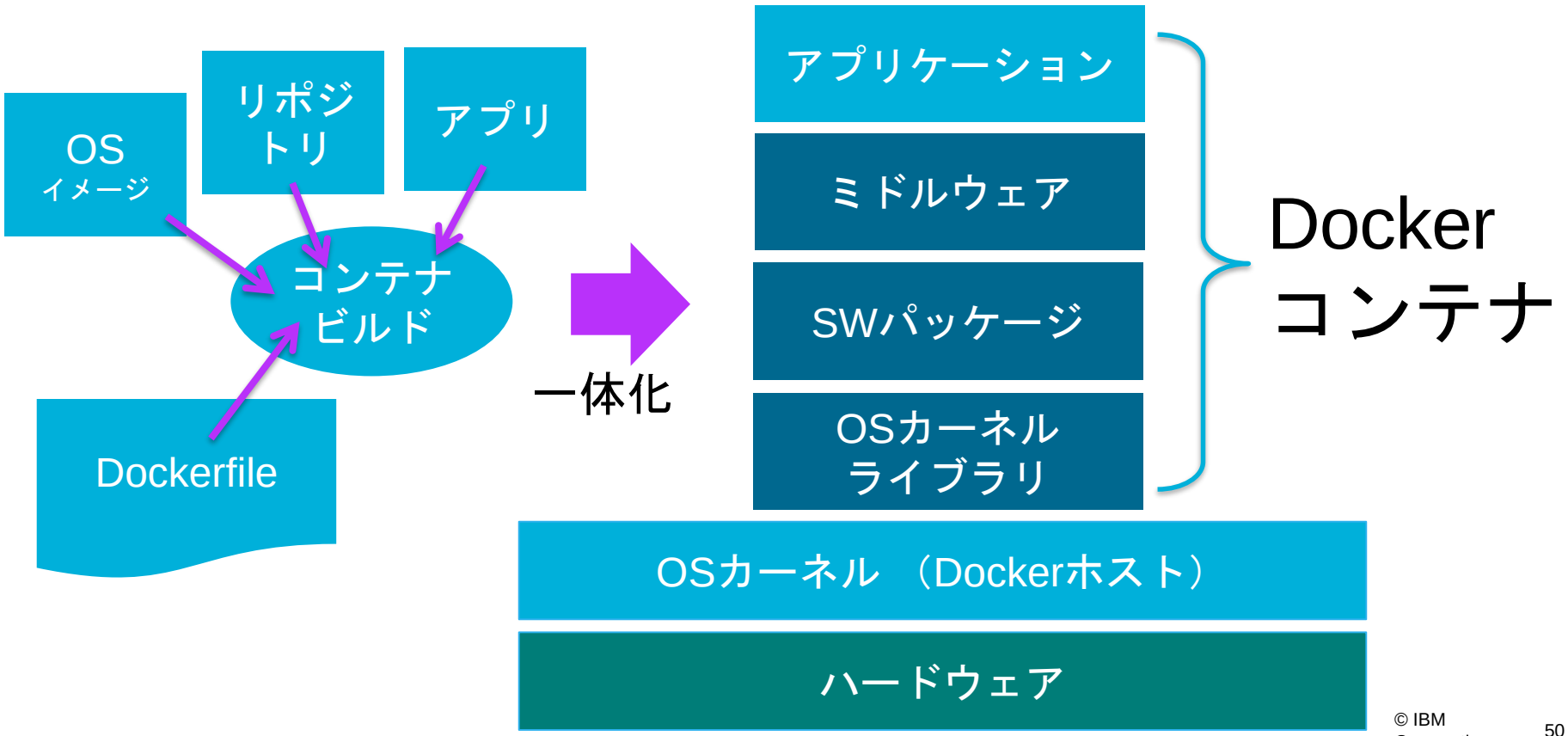
– 膨大な量のSWスタックを管理するのは無理なので、アプリのテストがパスしたら、その後はサーバーのソフトウェアに変更を加えない。

–Immutable Infrastructure（不変の基盤）



ところで Docker との関係は？

– Dockerfile (コンテナ仕様書）からアプリが含まれたコンテナを作成
– 変更が必要な場合は、コンテナごと再ビルド
– コンテナは、OSカーネルのSW更新の影響を受けない



大雑把に整理すると

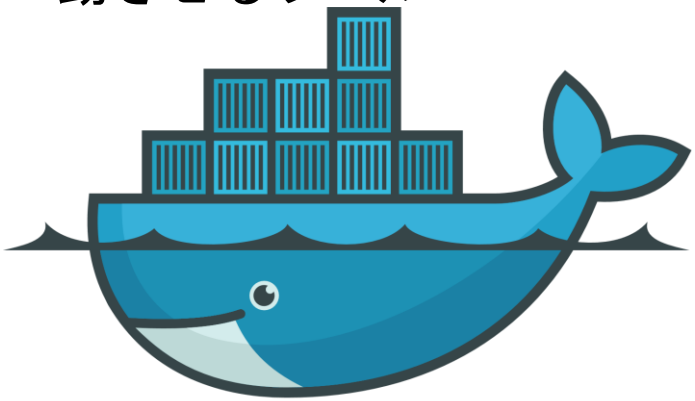
サーバー自動設定ツール CHEF とコンテナ･ランタイム環境 Dockerの整理

サーバー自動設定ツールは、 SWパッケージやソフトウェア設定を何度でも実行できるツール

CHEF™

冪等性 idempotence

Dockerはアプリの動作環境を変えないために、OSのSWスタックから隔絶（コンテナ化）して、稼動させるツール



docker

不変性 Immutable