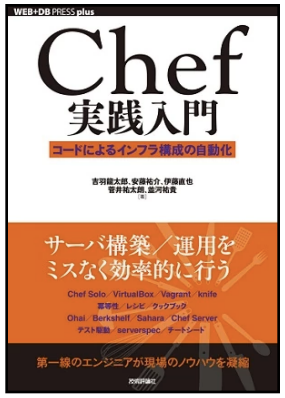
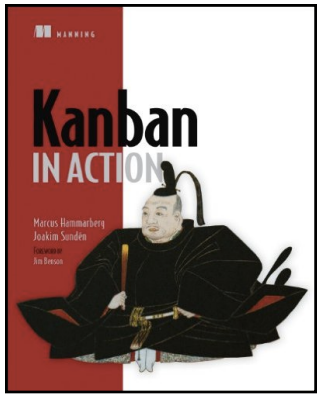
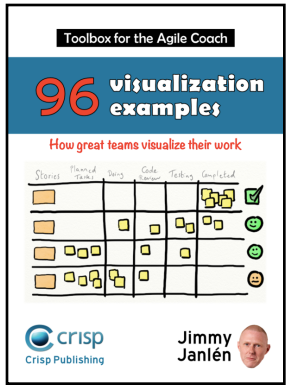

Chef Basic Training
(English version)
RYUZEE.COM

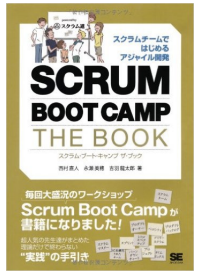
About RYUZEE.COM

- Consulting Service Provider
- Agile Development / DevOps / Cloud Computing
- http://www.ryuzee.com







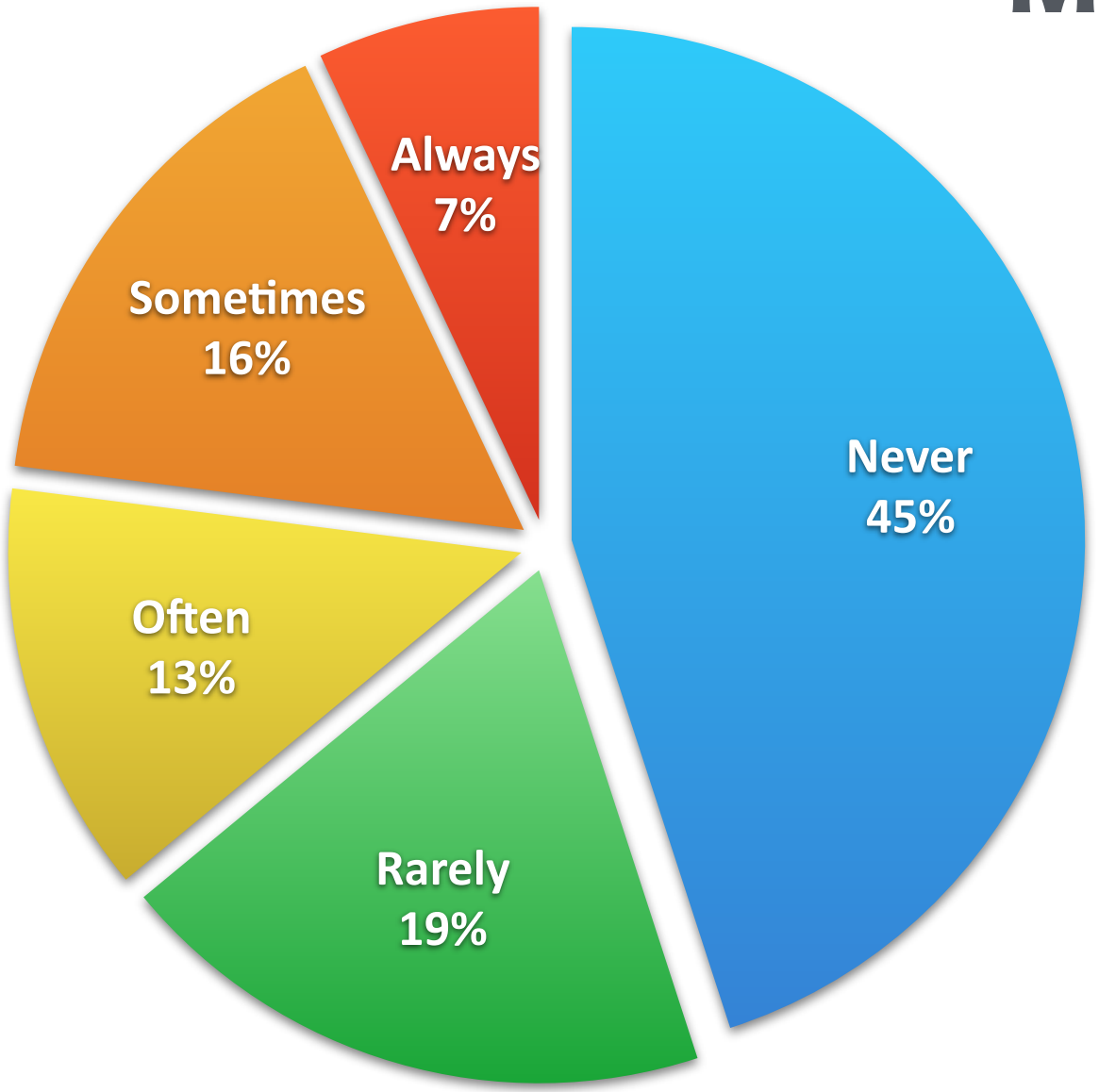


What is DevOps?

Current Challenges

- Business changes faster
- IT becomes a key element for business
- Delivering good software to customer is critical for business
- However, the speed of delivery is too slow and error prone (especially in traditional or large companies…)
- This causes the lost of chance and money in business
- Or you have your own issues
- IT often becomes Bottleneck

Muda



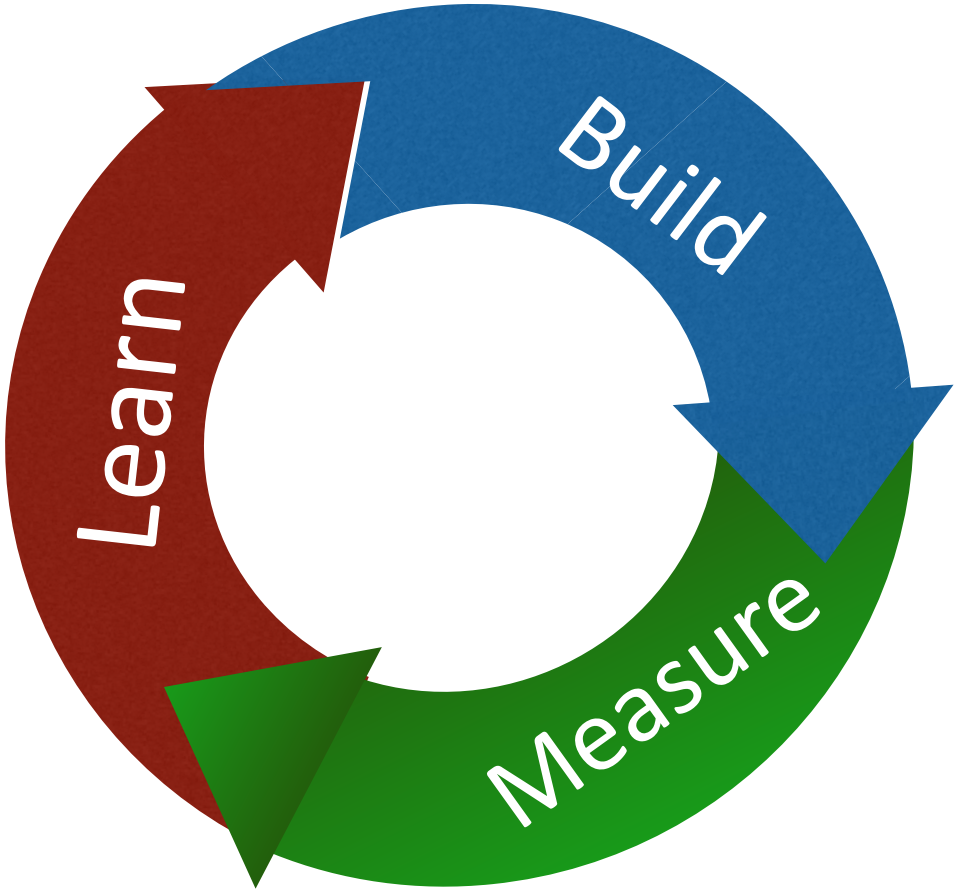
- **64** % of all features were rarely or never used (Excerpt from The Standish Group “Chaos” report 2002).
- You can imagine it by taking your computer’s builtin softwares’ usage into consideration…

Seven types of Waste

- Waste of over-production
- Waste of waiting
- Waste of transportation
- Waste of processing
- Waste of inventory
- Waste of motion
- Waste of making defects

“It’s difficult to know right things or future in advance.
It’s important to have a capability to catch up with
fast changes.”

Building fast feedback cycle



Conflict between Dev and Ops

- Differences of mission and responsibility (Who decides them…? Reasonable?)
- It's not my business (Says who?)
- Silo
- **It creates overhead and slows down your business result**

- Ops who think like devs.
- Devs who think like ops.

Werner Vogels, CTO, amazon.com

You build it, You run it

DevOps intends…

- DevOps intends to achieve business results, to enhance business agility and to avoid or reduce business risks **by leveraging culture and tool.**

Five Aspects

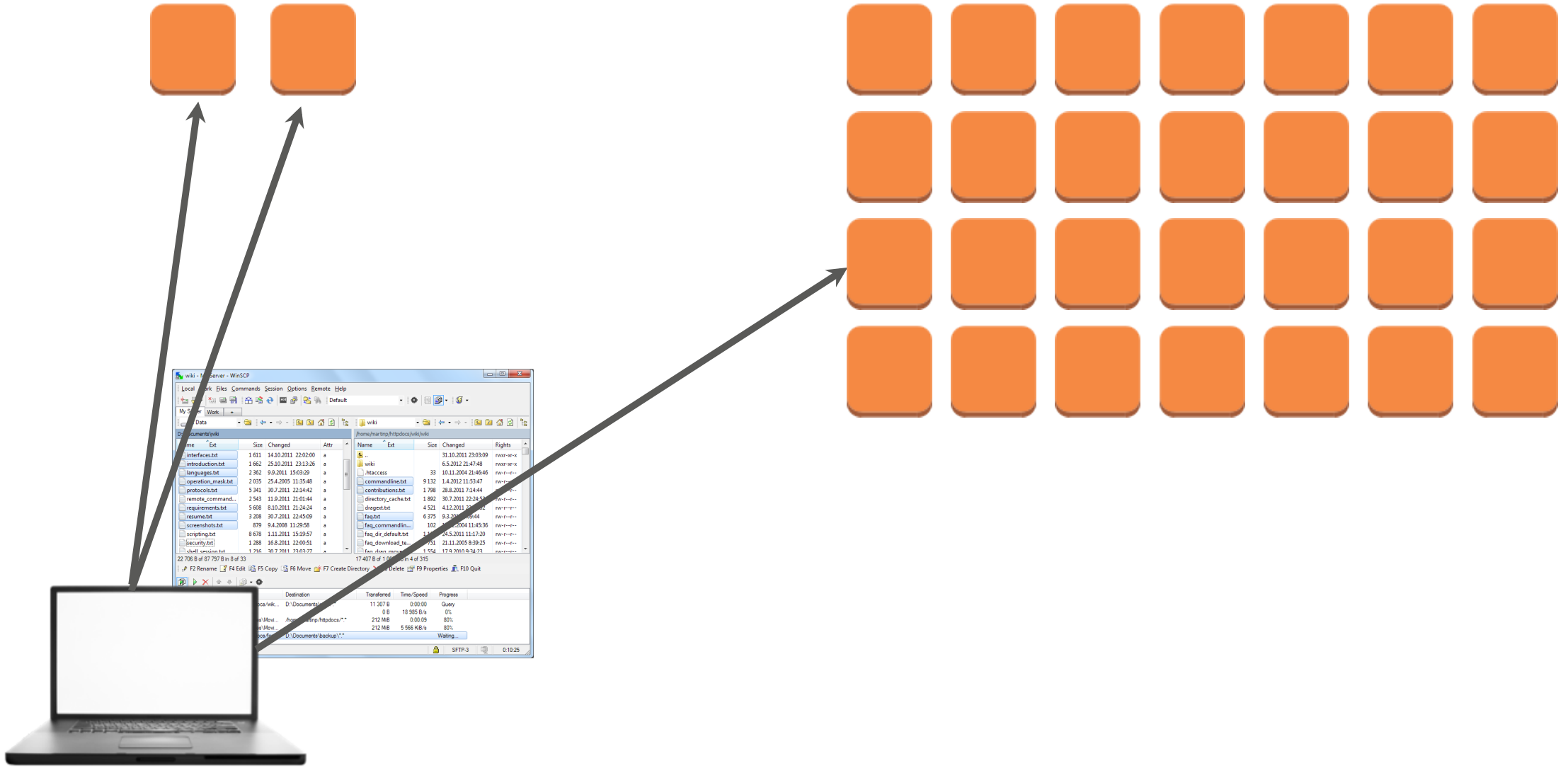
- Culture
- Lean
- **Automation**
- Measurement
- Sharing

Why do we need Infrastructure as Code?

Challenges in manual provisioning

- It takes longer time when the number of target servers increases
- Procedure documents or check lists is not maintained and kept updated
- Document driven manual operation causes mis-operation
- Documents can not be re-used across organisation

Imagine it…



Benefits of Infrastructure as Code

- The duration of provisioning is almost tranquil
- Code equals to “Procedure Document”. Only you need to do is to keep the code updated
- Code runs as it wrote. Same code bring us a same server
- Code can be tested continuously by using CI tools
- High reusability

How to automate infrastructure

There are several ways to automate

- Shell Script
- Capistrano or other deploy tools
- Provisioning tools such as Chef / Puppet / Ansible

Shell Script

```
#!/bin/sh
yum install -y httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server
/sbin/chkconfig --level 2345 httpd on
/sbin/chkconfig --level 2345 mysqld on
/etc/rc.d/init.d/mysqld start
/etc/rc.d/init.d/httpd start
```

- Simplest way
- However long script that includes conditional statements can not be maintained easily

Deploy Tool (Capistrano)

```
task :install_amp, roles => :web do
  run <<-CMD
    sudo yum install -y httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server &&
    sudo /sbin/chkconfig --level 2345 httpd on &&
    sudo /sbin/chkconfig --level 2345 mysqld on &&
    sudo /etc/rc.d/init.d/mysqld start &&
    sudo /etc/rc.d/init.d/httpd start
  CMD
end
```

- Capistrano or other deploy tools intend to be used for application deployment
- Of course, It's not impossible to automate infrastructure provisioning by deploy tools…

Provisioning Tool (Chef)

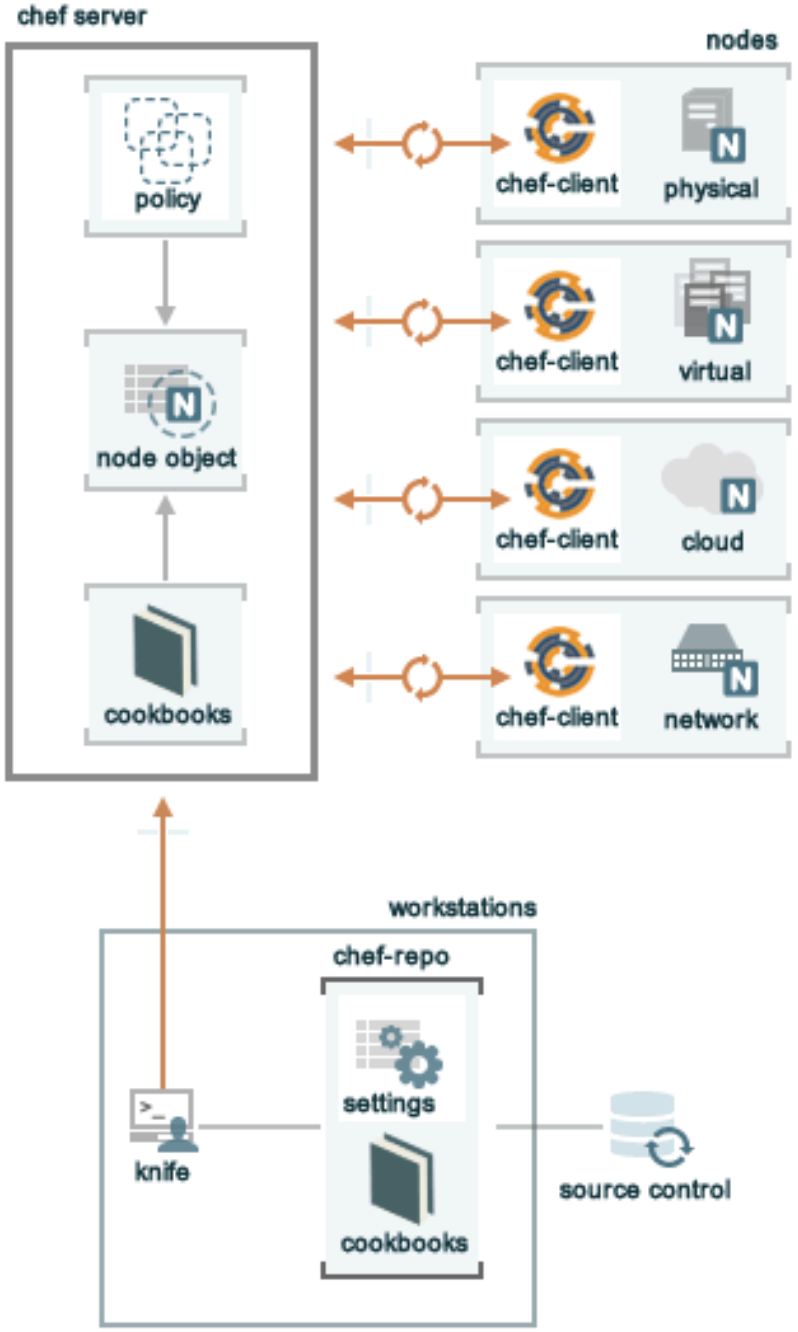
```
%w{httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server}.each do |p|
  package p do
    action :install
  end
end

service "httpd" do
  action [:enable, :restart]
  supports :status => true, :start => true, :stop => true, :restart => true
end

service "mysqld" do
  action [:enable, :restart]
  supports :status => true, :start => true, :stop => true, :restart => true
end
```

Several Provisioning Tools

Chef	Ansible	Puppet
DSL **(Ruby based)**	DSL	DSL
Client / Server (Agent)	Agentless	Client / Server (Agent)
Lots of related tools such as knife, berkshelf, foodcritic…	Few management tools	Already old…
DSL(Ruby based) must be learned	Simple. a few to learn	Unique DSL must be learned

- There are many server management tools that are written in Ruby. It's better to learn Ruby although you are an infrastructure engineer.

Chef's Architecture



Architecture

- Basically the architecture is **Client / Server** model
- Every information is stored in Chef Server and all nodes will access to Chef Server (From clients to Chef Server) to retrieve cookbooks, various attributes and so on

Basic Terminology #1

- **Chef Server** => The Chef Server acts as a hub for configuration data. The Chef Server stores cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by the Chef Client

- **Nodes** => A node is any machine—physical, virtual, cloud, network device, etc.—that is under management by Chef. Chef Client must be installed

- **Chef Client** => Tool to be installed into Nodes. It can be run as Service (daemon) or command line tool

Basic Terminology #2

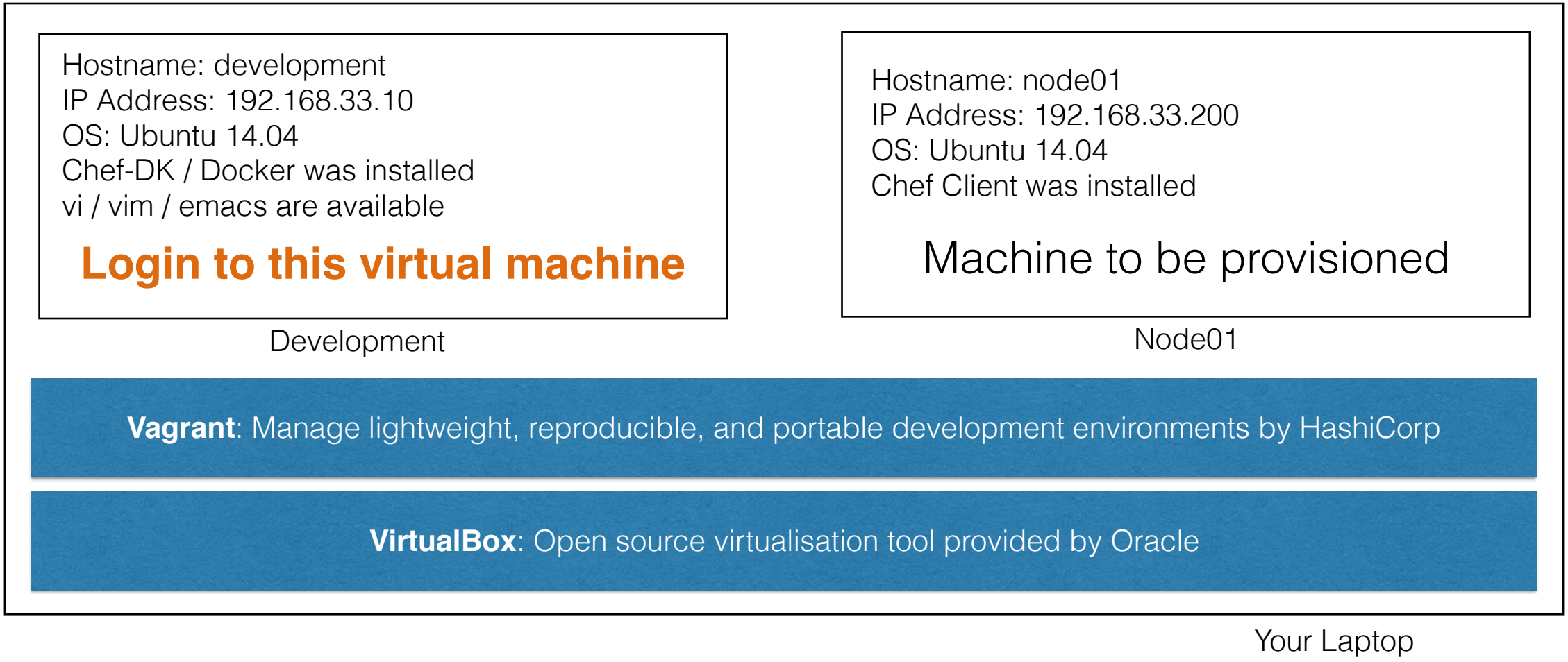
- **Cookbook** => A cookbook is the fundamental unit of configuration and policy distribution. A cookbook defines a scenario and contains everything that is required to support that scenario. Cookbook contains **Recipe, Attributes, Files, Templates and custom extensions**
- **Recipe** => DSL (Ruby based) code to install or configure target nodes. A cookbook can contain multiple recipes

NOTE: Chef Solo

- Chef also had a NON client / server mode named Chef Solo
- However, Chef Solo is now deprecated
- If you want to run Chef without Server, you are going to use **Chef Local Mode** (via knife-zero)
- Many web resources still pointed out Chef Solo. However, you need to remember the above.

Hands-on Environment

Environment Overview



VirtualBox

search...
Login Preferences

Welcome to VirtualBox.org!

- About
- Screenshots
- Downloads
- Documentation
  - End-user docs
  - Technical docs
- Contribute
- Community

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Download VirtualBox 5.0

News Flash

- **New March 4th, 2016**
  VirtualBox 5.0.16 released!
  Oracle today released a 5.0 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- **New January 19th, 2016**
  VirtualBox 4.3.36 released!
  Oracle today released maintenance releases which improve stability and fixes regressions. See the Changelog for details.
- **New July 9th, 2015**
  VirtualBox 5.0 released!
  Read the official press release for details.
- **Important February, 2015**
  We're hiring!
  Looking for a new challenge? We're looking for generic product developers (Russia).

More information...

Hot picks:

- Pre-built virtual machines for developers at Oracle Tech Network
- **Hyperbox** Open-source Virtual Infrastructure Manager project site
- **phpVirtualBox** AJAX web interface project site
- **IQEmu** automated Windows VM creation, application integration http://mirage335-site

ORACLE

Contact – Privacy policy – Terms

You can download VirtualBox from
https://www.virtualbox.org/

You can download Vagrant from
https://www.vagrantup.com/
If you already installed Vagrant,
please update it to 1.8+

Development environments made easy.

Create and configure lightweight, reproducible, and portable development environments.

DOWNLOAD GET STARTED

Vagrant Basics

- **Vagrant is one of the most popular open source tool to manage development environments provided by HashiCorp**
- Vagrant can run VirtualBox virtual machine, Docker machine, Azure virtual machine and so on. Vagrant wraps the deferences
- Vagrantfile is the definition of the environment. Same Vagrantfile produces same virtual environment. Thus your team members are able to obtain the same development environment. It could be a quite plus when you are developing something

Vagrantfile Example

```
Vagrant.configure(2) do |config|

  config.vm.define :development do |development|
    development.vm.box = 'ubuntu-14.04.4-chef-training-development-kit'
    development.vm.hostname = 'development'
    development.vm.network 'private_network', ip: '192.168.33.10'
  end

  config.vm.define :node01 do |node01|
    node01.vm.box = 'ubuntu-14.04.4-chef-training-node'
    node01.vm.hostname = 'node01'
    node01.vm.network 'private_network', ip: '192.168.33.200'
  end
end
```

You can download the script from
http://bit.ly/224TbdH

Vagrant basic commands (built-in)

boot virtual machines
vagrant up

boot specified machine
vagrant up development

login to the specific machine
vagrant ssh development

reboot machines
vagrant reload [machine name]

stop machines
vagrant halt [machine name]

dispose machines
vagrant destroy [machine name]

add box as a template
vagrant box add box_name box_url

install plugin
vagrant plugin install plugin_name

Add boxes

- Add boxes (from terminal or command prompt)

```
vagrant box add ubuntu-14.04.4-chef-training-development-kit http://bit.ly/1W4FWtV
vagrant box add ubuntu-14.04.4-chef-training-node http://bit.ly/1PQjMEl
```

All preparations finished? Then…

```
vagrant up
```

- Move to the directory that contains Vagrantfile, and then execute the command above (Terminal or Command Prompt)
- It will launch 2 virtual machines
- If it fails, please check logs or stderr

Writing first cookbook

Automate to install nginx

- nginx [engine x] is an HTTP and reverse proxy server. The performance is better than Apache HTTP Server
- Now we are going to automate to install nginx via Chef
- Please login to virtual machine named “development” by typing “**vagrant ssh development**” in terminal (OS X) or command prompt (Windows)

Only For Windows User

- Unfortunately Vagrant on Windows does not provide “vagrant ssh” functionality.
- Thus, if you want to login to virtual machine, open preferred ssh client such as Teraterm or Putty and then ssh access to **192.168.33.10 with username: vagrant and password vagrant**

Login to development environment

- You can look at the message as follows
- Now you are in the Linux(Ubuntu) virtual machine
- Username is vagrant and the current path is /home/vagrant

```
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Apr  1 20:54:29 2016 from 10.0.2.2
vagrant@development:~$
```

Git setup

- Before staring hands-on, setup your git account as follows

```
git config --global user.name “Sushi Taro”
git config --global user.email taro@example.com
```

Create Repository

- At first, create repository to store cookbooks, node information and so on

```
chef generate repo chef-repo
cd chef-repo
```

“chef” command shows…

```
Installing Cookbook Gems:
Compiling Cookbooks...
Recipe: code_generator::repo
  * directory[/home/vagrant/chef-repo] action create (up to date)
  * template[/home/vagrant/chef-repo/LICENSE] action create_if_missing (up to date)
  * cookbook_file[/home/vagrant/chef-repo/.chef-repo.txt] action create_if_missing (up to date)
  * cookbook_file[/home/vagrant/chef-repo/README.md] action create_if_missing (up to date)
(snip)
 Recipe: code_generator::repo
  * cookbook_file[/home/vagrant/chef-repo/cookbooks/README.md] action create_if_missing (up to date)
  * execute[initialize-git] action run
    - execute git init .
  * template[/home/vagrant/chef-repo/.gitignore] action create_if_missing
    - create new file /home/vagrant/chef-repo/.gitignore
    - update content in file /home/vagrant/chef-repo/.gitignore from none to 3523c4
    (diff output suppressed by config)
```

Directory Tree

- “tree -L 2” shows the directory structure and files.
- You can see “cookbooks” directory that stores chef cookbook.

```
vagrant@development:~/chef-repo$ tree -L 2
.
|-- chefignore
|-- cookbooks
|   |-- example
|   `-- README.md
|-- data_bags
|   |-- example
|   `-- README.md
|-- environments
|   |-- example.json
|   `-- README.md
|-- LICENSE
|-- README.md
`-- roles
    |-- example.json
    `-- README.md

6 directories, 9 files
```

Create nginx cookbook from template

- run “**knife cookbook create nginx -o ./cookbooks/**“ command in the current directory. It creates base files that consist of the cookbook
- Confirm the structure following to the right screenshot

```
vagrant@development:~/chef-repo$ tree -F 2 ./cookbooks/nginx/
2 [error opening dir]
./cookbooks/nginx/
|-- attributes/
|-- CHANGELOG.md
|-- definitions/
|-- files/
|   `-- default/
|-- libraries/
|-- metadata.rb
|-- providers/
|-- README.md
|-- recipes/
|   `-- default.rb
|-- resources/
`-- templates/
    `-- default/

10 directories, 4 files
```

Implement cookbook

- edit ./cookbooks/nginx/recipes/default.rb and input text indicated at the right
- This intends to **install nginx package, enable nginx service and run service**

```
package 'nginx' do
  action :install
end

service 'nginx' do
  action [ :enable, :start ]
end
```

Setup Chef Client

```
knife zero bootstrap 192.168.33.200 -x vagrant --sudo --ssh-password vagrant
```

- This command will install Chef Client in target node and create configuration file in source environment
- After this command, you will find some new directories such as “nodes” and “clients” in /home/vagrant/chef-repo/ directory

Confirm the target node is registered

```
knife node list -z
```

- “node01” will be shown
- If you are handling several environments, all nodes will be displayed

Set run_list

```
knife node run_list add node01 'recipe[nginx]' -z
```

- This command means that we are going to apply the default recipe in nginx cookbook to the target node named “node01”
- This command will update the configuration file “nodes/node01.json”. Please confirm bottom of that file.
- You can specify multiple recipes at the same time

Take a snapshot of Virtual Machine

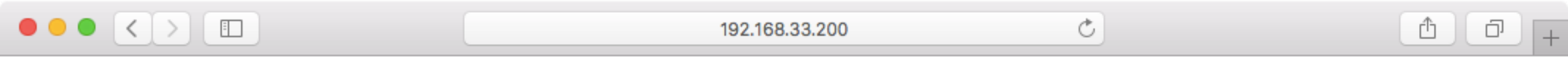
```
vagrant snapshot save node01 node01_001
```

- Run the command above in your host environment (NOT guest)
- “vagrant snapshot” is a built-in command to take snapshots and restore them
- To shorten the waiting duration, it’s better to leverage various kind of tools

Apply changes to target server

```
knife zero converge 'name:node01' -x vagrant --sudo -a knife_zero.host --ssh-password vagrant
```

- This command will apply changes to node01. In Chef World, we usually say that node will be converged into specific state
- Open web browser and access to http://192.168.33.200



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Are you OK?

- If it works well, add and commit files via git as follows

```
git add .
git commit -m “initial commit”
```

- If your cookbook does not work, check stdout and try again

Template #1

- Chef can generate and provision files by using template functionality
- Try to change index.html for nginx. Create a new file named “**index.html.erb**” in “**cookbooks/nginx/templates/default/**” with the content displayed in the right side

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
</head>
<body>
<h1>Welcome to Chef Training</h1>
<p><%= node.name %></p>
</body>
</html>
```

Template #2

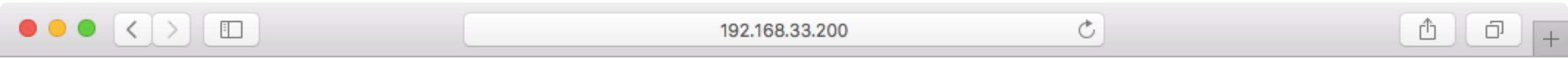
- Now, update the recipe
- Add highlighted section to cookbooks/nginx/recipes/default.rb
- Before converge, it's better to run "**vagrant snapshot restore node01 node01_001**" at the host machine to restore VM

```
package 'nginx' do
  action :install
end

template 'index.html' do
  path '/usr/share/nginx/html/index.html'
  owner 'root'
  group 'root'
  mode 0644
end

service 'nginx' do
  action [ :enable, :start ]
end
```

Apply changes to target server (Again)

```
knife zero converge 'name:node01' -x vagrant --sudo -a knife_zero.host --ssh-password vagrant
```

- This command will apply changes to node01. In Chef World, we usually say that node will be converged into specific state
- Open web browser and access to http://192.168.33.200



Welcome to Chef Training!!

node01

Resource

- At this point, you tried “package”, “service”, “template” in your first cookbook. These keywords are called as “**resource**”
- Chef has lots of built-in resources (See next page)
- You are going to write your own cookbook by leveraging various resources

Resources (Excerpt…)

package	user	powershell_script	ifconfig
template	group	ruby_block	http_request
service	remote_file	cron	link
file	execute	git	log
directory	script	mount	chai

Basic Terminology #3 & need to learn

- **Attribute** => Attribute(s) are variables that can be used when provisioning. For example, when you create php environment, you may want to set several variables in php.ini. You can set these values when provisions.
- **Role** => You can define Roles as you like. Typically, web server role, db server role, monitoring server role and so on. By using role, you can build specific infrastructure by only selecting role(s) for setting several recipes
- **Environment** => You can have several environment such as development, staging, production that have different environmental values(attributes)

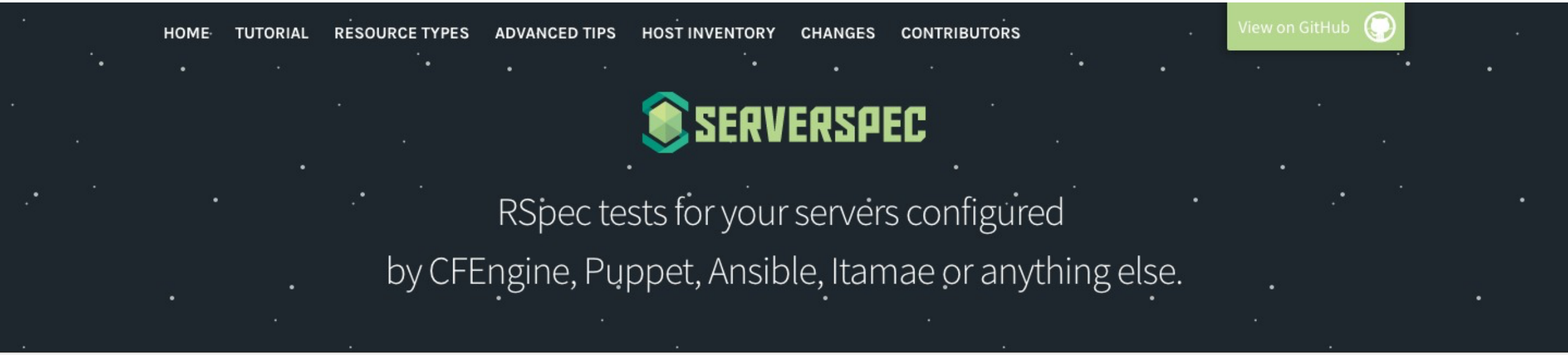
Test automation

Why automated tests matter?

- Reduce risks
- Reduce repetitive manual processes
- Generate deployable software at any time and at any place
- Enable better project visibility
- Establish greater confidence in the software product from the development team

Test Kitchen

- Test Kitchen is a test harness tool to execute your configured code on one or more platforms in isolation
- See more details at http://kitchen.ci
- Test Kitchen launches isolated environment (using Vagrant, Docker, AWS, Azure…), apply specified recipes in the cookbook, verify results
- It supports many testing frameworks including Bats, shUnit2, RSpec, **Serverspec**



About V2

Serverspec/Specinfra v2 has been just released. See the document about v2.

Visit http://serverspec.org/

About

With Serverspec, you can write RSpec tests for checking your servers are configured correctly.

Serverspec tests your servers' **actual state** by **executing command locally, via SSH, via WinRM, via Docker API and so on**. So you don't need to install any agent softwares on your servers and can use any configuration management tools, Puppet, Ansible, CFEngine, Itamae and so on.

But the true aim of Serverspec is to help refactoring infrastructure code.

Installation

Preparation

- Test Kitchen provides test generator. Run command as follows

```
cd /home/vagrant/chef-repo/cookbooks/nginx
kitchen init -D kitchen-docker
```

- .kitchen.yml, chefignore, test/integration/default must be created

```
mkdir -p test/integration/default/serverspec/localhost
```

Write Tests

```
require “spec_helper”

describe package("nginx") do
  it { should be_installed }
end

describe service("nginx") do
  it { should be_enabled }
  it { should be_running }
end

describe port(80) do
  it { should be_listening }
end

describe file(“/usr/share/nginx/html/index.html”) do
  it { should be_file }
end
```

test/integration/default/serverspec/
localhost/default_spec.rb

test/integration/default/serverspec/
spec_helper.rb

```
require “serverspec”

set :backend, :exec
```

What are testing for?

```
require “spec_helper”

describe package("nginx") do
  it { should be_installed }
end

describe service("nginx") do
  it { should be_enabled }
  it { should be_running }
end

describe port(80) do
  it { should be_listening }
end

describe file(“/usr/share/nginx/html/index.html”)
do
  it { should be_file }
end
```

- ✤ nginx package should be installed
- ✤ nginx service should be running
- ✤ nginx service should run after boot
- ✤ nginx should be listening on TCP/80
- ✤ index.html should exist

Edit configuration and start verification

```
---
driver:
  name: docker

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-14.04

suites:
  - name: default
    run_list:
      - recipe[nginx::default]
    attributes:
```

- Edit .kitchen.yml as noted in the left
- It means that test will run Ubuntu14 Docker machine
- Execute “**kitchen test**” and then verification starts!!
- Verification will take a few minutes

```
Fetching: rspec-mocks-3.4.1.gem (100%)
Fetching: rspec-3.4.0.gem (100%)
Fetching: multi_json-1.11.2.gem (100%)
Fetching: serverspec-2.31.1.gem (100%)
-----> serverspec installed (version 2.31.1)
       /opt/chef/embedded/bin/ruby -I/tmp/verifier/suites/serverspec -I/tmp/verifier/gems/gems/rspec-support-3.4.1/lib:/tmp/verifier/gems/g
pattern /tmp/verifier/suites/serverspec/\*\*/\*_spec.rb --color --format documentation --default-path /tmp/verifier/suites/serverspec

       Package "nginx"
         should be installed

       Service "nginx"
         should be enabled
         should be running

       Port "80"
         should be listening

       File "/usr/share/nginx/html/index.html"
         should be file

       Finished in 1.28 seconds (files took 0.81198 seconds to load)
       5 examples, 0 failures

       Finished verifying <default-ubuntu-1404> (0m13.20s).
-----> Destroying <default-ubuntu-1404>...
       UID                 PID                 PPID                C                   STIME               TTY                 TIME
       root                7220                728                 0                   03:48               ?                   00:00:00
asswordAuthentication=yes -o UsePrivilegeSeparation=no -o PidFile=/tmp/sshd.pid
       root                7240                7220                0                   03:48               ?                   00:00:00
       root                7914                7220                0                   03:48               ?                   00:00:00
       www-data            7915                7914                0                   03:48               ?                   00:00:00
       www-data            7916                7914                0                   03:48               ?                   00:00:00
       www-data            7917                7914                0                   03:48               ?                   00:00:00
       www-data            7918                7914                0                   03:48               ?                   00:00:00
       2c172be4204d611f13da8136bfa43b07862fcecc1b481f143f2ac7066d379e2b
```

You can see the verification result

Run Tests with Jenkins

- It's possible to run tests with Jenkins
- If you are interested in implement it, please try it later
    - Install JDK8 (NOT JDK7), Jenkins into development machine as follows

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk
wget http://pkg.jenkins-ci.org/debian-stable/binary/jenkins_1.642.4_all.deb
sudo dpkg -i jenkins_1.642.4_all.deb
```

How to write good cookbooks

Community Cookbooks

- Chef has a huge eco-system. Many cookbooks (see the table below) were released by chef community. Visit https://supermarket.chef.io/

mysql	nginx	apache2	postgresql
java	git	apt	yum
php	build-essential	nodejs	mongodb
ntp	jenkins	database	python
docker	tomcat	rabbitmq	elasticsearch

Berkshelf : Dependency manager

- Especially in community cookbook, there might be some dependencies on other cookbooks.
- To resolve this challenge, you can use Berkshelf. See http://berkshelf.com/
- It is similar to other language package manager such as composer(PHP), bundler(Ruby) and npm(Nodejs)

Keep the code Clean

- Cookbook is equal to the procedure document. So readability and maintainability is important (Also continuous integration matters)
- For example, Foodcritic verifies your cookbooks (static analysis) like rubocop
- Keep cookbook small that meet the Single Responsibility principle

Questions?