CHEF IN THE CLOUD AND ON THE GROUND

Michael T. Nygard

Relevance

michael.nygard@thinkrelevance.com
@mtnygard
Infrastructure As Code
Infrastructure As Code

Chef
Infrastructure As Code
Chef
Development Models
Infrastructure As Code
Chef
Development Models
Deployment Models
Infrastructure As Code
Problems beget solutions. Solutions become the next problems.
mandag den 10. oktober 11
mandag den 10. oktober 11
mandag den 10. oktober 11
mandag den 10. oktober 11
mandag den 10. oktober 11
What if we could version control our servers and network, the way we do with code?
Infrastructure as Code
Infrastructure as Code
Infrastructure as Code

- Desired state
Infrastructure as Code

• Desired state

• Transformations to achieve it
Infrastructure as Code

• Desired state
• Transformations to achieve it
• As text files
Infrastructure as Code

- Desired state
- Transformations to achieve it
- As text files
- In version control
Infrastructure as Code

- Desired state
- Transformations to achieve it
- As text files
- In version control
- Testable and reproducible
Chef Basics
# Chef Basics

<table>
<thead>
<tr>
<th>Cookbook</th>
<th>Container for semantically related recipes and their support files.</th>
</tr>
</thead>
</table>
# Chef Basics

<table>
<thead>
<tr>
<th>Cookbook</th>
<th>Container for semantically related recipes and their support files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe</td>
<td>Resource definitions, written as Ruby code</td>
</tr>
</tbody>
</table>

mandag den 10. oktober 11
# Chef Basics

<table>
<thead>
<tr>
<th>Cookbook</th>
<th>Container for semantically related recipes and their support files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe</td>
<td>Resource definitions, written as Ruby code</td>
</tr>
<tr>
<td>Attribute</td>
<td>Parameter used in recipes, can be supplied by cookbook defaults, the node, or the role</td>
</tr>
</tbody>
</table>
# Chef Basics

<table>
<thead>
<tr>
<th>Cookbook</th>
<th>Container for semantically related recipes and their support files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe</td>
<td>Resource definitions, written as Ruby code</td>
</tr>
<tr>
<td>Attribute</td>
<td>Parameter used in recipes, can be supplied by cookbook defaults, the node, or the role</td>
</tr>
<tr>
<td>Template</td>
<td>ERB file that gets expanded onto the target host, used by resources in recipes</td>
</tr>
</tbody>
</table>
## Chef Basics

<table>
<thead>
<tr>
<th>Cookbook</th>
<th>Container for semantically related recipes and their support files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe</td>
<td>Resource definitions, written as Ruby code</td>
</tr>
<tr>
<td>Attribute</td>
<td>Parameter used in recipes, can be supplied by cookbook defaults, the node, or the role</td>
</tr>
<tr>
<td>Template</td>
<td>ERB file that gets expanded onto the target host, used by resources in recipes</td>
</tr>
<tr>
<td>Library</td>
<td>Extra Ruby code, can create classes, resources, implement actions, etc.</td>
</tr>
</tbody>
</table>
# Chef Basics

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookbook</td>
<td>Container for semantically related recipes and their support files.</td>
</tr>
<tr>
<td>Recipe</td>
<td>Resource definitions, written as Ruby code</td>
</tr>
<tr>
<td>Attribute</td>
<td>Parameter used in recipes, can be supplied by cookbook defaults, the node, or the role</td>
</tr>
<tr>
<td>Template</td>
<td>ERB file that gets expanded onto the target host, used by resources in recipes</td>
</tr>
<tr>
<td>Library</td>
<td>Extra Ruby code, can create classes, resources, implement actions, etc.</td>
</tr>
<tr>
<td>Resource Provider</td>
<td>Lightweight Ruby DSL for creating new resources.</td>
</tr>
<tr>
<td>maintainer</td>
<td>&quot;Opscode, Inc.&quot;</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>maintainer_email</td>
<td>&quot;<a href="mailto:cookbooks@opscode.com">cookbooks@opscode.com</a>&quot;</td>
</tr>
<tr>
<td>license</td>
<td>&quot;Apache 2.0&quot;</td>
</tr>
<tr>
<td>description</td>
<td>&quot;Installs git and/or set...&quot;</td>
</tr>
</tbody>
</table>
| long_description    | `IO.read(File.join(  
|                     | File.dirname(__FILE__),  
|                     | 'README.rdoc'))` |
| version             | "0.9.0"              |
| recipe              | "git", "Installs git" |
| recipe              | "git::server",  
|                     | "Sets up a runit_service for git daemon" |
%w{ ubuntu debian arch}.each do |os|
  supports os
end

%w{ runit }.each do |cb|
  depends cb
end
recipes/default.rb

case node[:platform]
  when "debian", "ubuntu"
    package "git-core"
  else
    package "git"
end
include_recipe "git"

directory "/srv/git" do
  owner "root"
  group "root"
  mode 0755
end

cont...
case  node[:platform]
when  "debian",  "ubuntu"
  include_recipe  "runit"
  runit_service  "git-daemon"
else
  log  "Platform requires setting up ..."
  log  "Hint: /usr/bin/git daemon --export..."
end
Cookbook: MySQL

- attributes
  - server.rb
- libraries
  - database.rb
  - helpers.rb
- metadata.json
- metadata.rb
- providers
  - database.rb
- README.md
- recipes
  - client.rb
- templates
  - default
    - debian.cnf.erb
    - grants.sql.erb
    - my.cnf.erb
    - mysql-server.seed.erb
    - port_mysql.erb
- resources
  - database.rb
[client]
host     = localhost
user     = debian-sys-maint
password = <%= node['mysql']['server_debian_password'] %>
socket   = <%= node['mysql']['socket'] %>
basedir  = /usr

[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = <%= node['mysql']['server_debian_password'] %>
socket   = <%= node['mysql']['socket'] %>
base_dir = /usr
actions :flush_tables_with_read_lock, :unflush_tables, :create_db, :query

attribute :host, :kind_of => String
attribute :username, :kind_of => String
attribute :password, :kind_of => String
attribute :database, :kind_of => String
attribute :sql, :kind_of => String
attribute :exists, :default => false
Using the Custom Resource

```ruby
# In cookbooks/stratego/recipes/default.rb
include_recipe "mysql::client"

mysql_database "reporting database" do
  host "db01prod"
  username "root"
  password node[:mysql][:server_root_password]
  database "reporting_production"
  action :create_db
end
```
include Opscode::Mysql::Database

action :create_db do
  unless exists?
    begin
      Chef::Log.info "mysql_database: Creating database..."
      db.query("create database #{new_resource.database}")
      new_resource.updated_by_last_action(true)
    ensure
      db.close
    end
  end
end
How do cookbooks get applied to servers?
Runlist Executes On Node

- Runlist
- Recipe or Role
  - Recipe
  - Role
  - Resource
  - Attribute
  - Attribute

Include: Runlist -> Recipe or Role
Include: Recipe or Role -> Role
Include: Role -> Attribute
Include: Attribute -> Resource
Declares: Recipe
Declares: Role
Overrides: Role
Uses: Resource
Uses: Attribute
Example Runlist (from a role file)

name "dev_env_bamboo"
description "Continuous Integration server"

run_list "recipe[aws]", "recipe[xfs]",
  "recipe[bamboo]",
  "recipe[bamboo::crowd_authentication]",
  "recipe[bamboo::ebs_volume]",
  "recipe[build_dependencies]",
  "recipe[maven]", "recipe[maven3]",
  "recipe[bamboo::post_maven_bamboo]",
  "recipe[bamboo::start]",
  "recipe[iptables::ci_instance]"
Bamboo has a one time, setup wizard in the admin GUI. How do you automate that?
Recreate the results, not the process.
Recreate the results, not the process.
Recreate the results, not the process.

1. Install once by hand, stop before setup wizard.
Recreate the results, not the process.

1. Install once by hand, stop before setup wizard.

2. Make a tarball.
Recreate the results, not the process.

1. Install once by hand, stop before setup wizard.

2. Make a tarball.

3. Go through setup wizard.
Recreate the results, not the process.

1. Install once by hand, stop before setup wizard.
2. Make a tarball.
3. Go through setup wizard.
4. Compare resulting files with snapshot.

mandag den 10. oktober 11
Recreate the results, not the process.

1. Install once by hand, stop before setup wizard.

2. Make a tarball.

3. Go through setup wizard.

4. Compare resulting files with snapshot.

5. Build a template to replicate the outcome.
Where do you keep installation binaries?
Control, integrity, efficiency
Control, integrity, efficiency
Control, integrity, efficiency

1. Your VCS may hate binaries.
Control, integrity, efficiency

1. Your VCS may hate binaries.

2. S3 worked for us. Private maven repo also works well.
Control, integrity, efficiency

1. Your VCS may hate binaries.
2. S3 worked for us. Private maven repo also works well.
3. Vendor download might also work, but beware slipstreaming.
remote_file "crowd" do
  path "/tmp/crowd.tar.gz"
  source "http://downloads.atlassian.com/.../
atlassian-crowd-2.3.3.tar.gz"
  checksum "dcc486625e96925..."
end
Have a sandbox!
Where do I find cookbooks?
Many available on github

cookbooks.opscode.com
Download at the Shell

$ knife cookbook site download postgresql
Downloading postgresql from the cookbooks site ...
Cookbook saved: /Users/mtnygard/cookbooks/postgresql...
$ tar xvzf postgresql-0.11.1.tar.gz
x postgresql/
x postgresql/attributes/
x postgresql/metadata.json
x postgresql/metadata.rb
x postgresql/README.rdoc
x postgresql/recipes/
x postgresql/templates/
x postgresql/templates/default/
x postgresql/templates/default/debian.pg_hba.conf.erb
...

mandag den 10. oktober 11
One other cool knife trick

$ gem install knife-ec2
Fetching: excon-0.6.6.gem (100%)
...
Successfully installed knife-ec2-0.5.10
6 gems installed
$ knife ec2 server create 'role[webserver]' \ 
   -I ami-7000f019 -f m1.small
Instance ID: i-9e7ef1fe
Flavor: m1.small
Image: ami-7000f019
Region: us-east-1
Availability Zone: us-east-1c
Security Groups: default
SSH Key: relevance_aws

Waiting for server...
Throughout Development Cycle

Local Development
Throughout Development Cycle

Local Development

Single App (solo)
Throughout Development Cycle

Local Development

Single App (solo)

Server Based Rollout
Local Development
Vagrant
Vagrant

Builds virtual machines from a text file and some Chef cookbooks.
Vagrant Project

- rstudio-spike
  - Vagrantfile
  - README.md
  - Gemfile
- cookbooks
  - apt
  - rstudio
Vagrantfile

Vagrant::Config.run do |config|
  config.vm.box = "lucid64"
  config.vm.box_url = "http://files.vagrantup.com/lucid64.box"
  config.vm.forward_port "http", 8787, 8787
  config.vm.provision :chef_solo do |chef|
    chef.cookbooks_path = "cookbooks"
    chef.add_recipe "rstudio"
    chef.add_recipe "mysql::server"
    chef.json.merge!(
      {:mysql => {:
        :server_root_password => "foo"
      } }
    )
  end
end
Benefits
Benefits

Version control dev environment
Benefits

Version control dev environment

Share configs across team
Benefits

Version control dev environment

Share configs across team

Recreate project or client setup
Benefits

Version control dev environment
Share configs across team
Recreate project or client setup
Keep host environment clean
“Base Box”
“Base Box”

OS Image

+ Vagrant configs
Download Base Boxes

Vagrantbox.es

Vagrant is an amazing tool for managing virtual machines via a simple to use command line interface. With a simple vagrant up you can be working in a clean environment based on a standard template.

These standard templates are called base boxes, and this website is simply a list of boxes people have been nice enough to make publicly available.

Suggest a Box

Do you know of another high quality base box? Let us know about it and, once we've checked it out, we'll add it to the list below.

Available Boxes

<table>
<thead>
<tr>
<th>Box Type</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu 11.04 server</td>
<td></td>
<td><a href="http://dl.dropbox.com/u/7490647/talifun-ubuntu-11.04-server-amd64.box">http://dl.dropbox.com/u/7490647/talifun-ubuntu-11.04-server-amd64.box</a></td>
</tr>
</tbody>
</table>
Make Base Boxes

Easing the building of vagrant boxes — Read more

Merge pull request #115 from dcarley/vbox_sataportcount_1

jedi4ever authored 2 days ago
Some Recommendations
Know your image; build your own base boxes.
Avoid multiple Chef and Vagrant copies.

Use RVM with .rvmrc in your Vagrant directory.
Reuse cookbooks between vagrant boxes and real deployments.
Share the project directory with the VM.
Solo Deployment
Chef Solo
Chef Solo

Runs cookbooks, with attributes, on a node.
chef-solo

$ sudo chef-solo -c solo.rb -j node.json
... INFO: *** Chef 0.10.4 ***
... INFO: Setting the run_list to ["recipe[rstudio]"] from JSON
... INFO: Run List is [recipe[rstudio]]
... INFO: Run List expands to [rstudio]
... INFO: Starting Chef Run for lucid64.hsd1.ca.comcast.net.

... INFO: Processing package[r-base] action install
(rstudio::default line 18)
  -- many lines of output --
... INFO: Processing dpkg_package[rstudio-server] action install
(rstudio::default line 46)
... INFO: dpkg_package[rstudio-server] installed version 0.94.84
... INFO: Chef Run complete in 238.033898 seconds
... INFO: Running report handlers
... INFO: Report handlers complete
file_cache_path  "/var/chef-solo"
cookbook_path  ["/var/chef/cookbooks",
               "/var/chef/site-cookbooks"]
```json
{
  "mysql": {
    "server_root_password": "foo"
  },
  "run_list": ["recipe[rstudio]"
}
```
Benefits
Benefits

Self-contained bundle
Benefits

Self-contained bundle

Can run from tarball at URL
Benefits

Self-contained bundle

Can run from tarball at URL

Easy setup; no extra infrastructure
Some Recommendations
Rubyists: deliver cookbooks with capistrano, use chef-solo to execute.
Everyone: start with chef-solo before you deal with chef server.
Make a shared repository for cookbooks.
Use attributes to factor out environment and tuning.
Use roles to set parameters for an environment.
Factoring Roles

**Compose Role**
Runlist for role includes other roles
Use for applications

**Mix in Role**
Runlist for role includes other roles
Use for environments

mandag den 10. oktober 11
Example: Factoring Roles

```json
{
  "run_list": [
    "role[rstudio-server]",
    "role[qa]"
  ]
}
```
Separate off-the-shelf cookbooks from locally maintained cookbooks
Example: Isolating Local Cookbooks

<table>
<thead>
<tr>
<th>project-root</th>
</tr>
</thead>
<tbody>
<tr>
<td>README.md</td>
</tr>
<tr>
<td>cookbooks</td>
</tr>
<tr>
<td>site-cookbooks</td>
</tr>
<tr>
<td>vagrant</td>
</tr>
<tr>
<td>Vagrantfile</td>
</tr>
</tbody>
</table>
Example: Local Cookbooks With Overrides

- cookbooks
  - apache2
    - templates
      - default
  - site-cookbooks
    - apache2
      - templates
        - default
          - default-site.erb
Use overrides sparingly.
It’s implementation inheritance.
Server-Based Deployment
Chef Server
Chef Server

- Client-server polling
- Searchable via API
- Authenticated and encrypted
- Centralized management
Server Based Search

API access to Lucene search
Nodes, roles, or data bags
Can be used from recipes on clients
Example: Automatic Nagios Config

1. Search for nodes in the environment

2. Generate Nagios config files with information about nodes.
Step 1: Chef Server Assigns Recipe “nagios::server”
Step 1: Chef Server Assigns Recipe “nagios::server”
Step 1: Chef Server Assigns Recipe “nagios::server”
Step 2: Client Downloads Cookbooks
Step 2: Client DownloadsCookbooks

Chef Server

node 1

get cookbooks

mandag den 10. oktober 11
Step 2: Client Downloads Cookbooks

Chef Server

node

get cookbooks

cookbooks

mandag den 10. oktober 11
Step 3: Client Executes Recipe

```ruby
sysadmins = search(:users, 'groups:sysadmin')
nodes = search(:node, "hostname:[* TO *] AND role:#{node[:app_environment]}")

members = Array.new
sysadmins.each do |s|
  members << s['id']
end
```
Step 3a: Client Searches Nodes

Chef

Server

node 1
Step 3a: Client Searches Nodes

Chef Server

search

node 1

mandag den 10. oktober 11
Step 3a: Client Searches Nodes

Chef

Server

node1

search

lucene

search
Step 3a: Client Searches Nodes
Step 3b: Client Applies Templates

```erb
nagios_conf "hosts" do
  variables :nodes => nodes
end
```
Step 3b: Client Applies Templates

<% @nodes.each do |n| -%>
define host {
  use server
  address <%= n['ipaddress'] %>
  host_name <%= n['hostname'] %>
  hostgroups <%= n.run_list.roles.to_a.join(",") %>
}
<% end -%>
Data Bags

JSON data

Held on server

Access via API
begin
  bamboo_app = Chef::DataBagItem.load(:apps, :bamboo)
  Chef::Log.info("Loaded... apps[#{bamboo_app['id']}]")
rescue
  Chef::Log.fatal("Could not find #{:bamboo}... ")
raise
end
Search & data bags require Chef Server.

Breaks Vagrant and Chef Solo.
Encrypted Data Bags

https://jtimberman.posterous.com/64227128
Chef Server in EC2
Challenges

No long-lived identity
Must use EBS for server
Authentication of new nodes
Example: EC2 + ELB + AS

- Elastic load balancer
- app1
- app2
- extra instances
- Autoscale Group
- db
- chef-server
- ami with chef client
Recent News: CloudFormation + Chef
Recent News: CloudFormation + Chef

1. create virtual machine
Recent News: CloudFormation + Chef

EC2 Host

1. create virtual machine
2. yum install gcc, ruby, rubygems
gem install chef
Recent News: CloudFormation + Chef

1. create virtual machine
2. yum install gcc, ruby, rubygems
gem install chef
3. deposit solo.rb & node.json files
Recent News: CloudFormation + Chef

1. create virtual machine
2. yum install gcc, ruby, rubygems
   gem install chef
3. deposit solo.rb & node.json files
4. run chef-solo with a URL target
Recent News: CloudFormation + Chef

1. create virtual machine
2. yum install gcc, ruby, rubygems gem install chef
3. deposit solo.rb & node.json files
4. run chef-solo with a URL target
5. chef-solo fetches cookbooks & executes recipes
Chef in Private Clouds
Chef + OpenStack: Automation 2 Ways

1. Build the platform itself
2. Build nodes in the platform
Other Options

Hosted Chef: Chef Server as a Service

www.opscode.com
Other Options

Hosted Chef: Chef Server as a Service

Private Chef: Like Hosted Chef, as an appliance behind your firewall

www.opscode.com
Resources

Chef
http://wiki.opscode.com

Vagrant
http://www.vagrantup.com/

CloudFormation
http://bit.ly/mPFFJG
CHEF IN THE CLOUD AND ON THE GROUND

Michael T. Nygard
Relevance, Inc.

michael.nygard@thinkrelevance.com @mtnygard