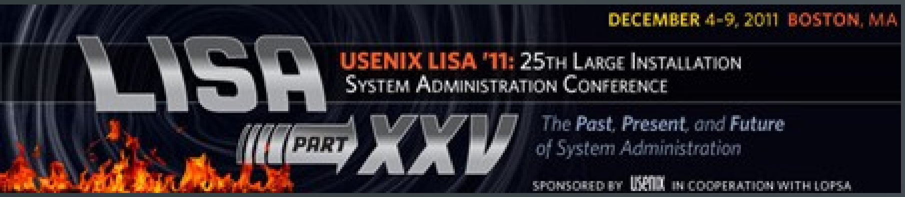
Converting the Ad-Hoc Configuration of a Heterogeneous Environment to a CFM

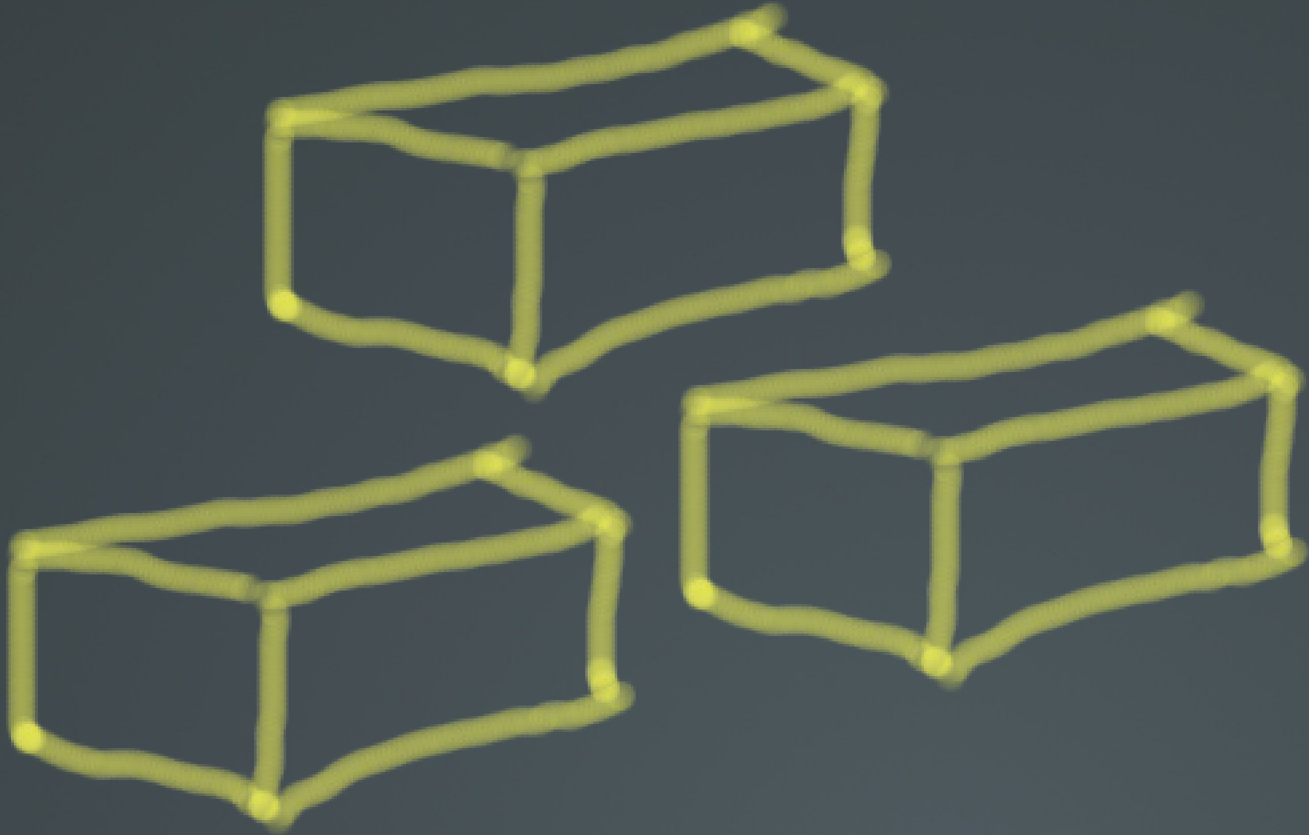
How I learned to stop worrying and love the Chef

Dimitri Aivaliotis

Every Ware Ltd


DECEMBER 4–9, 2011 BOSTON, MA
LISA
PART XXV
USENIX LISA '11: 25th Large Installation System Administration Conference
The Past, Present, and Future of System Administration
SPONSORED BY USENIX IN COOPERATION WITH LOPSA

In the beginning...



As time goes on...

Chef

Where to begin?

MOTD

- your CFM server is setup correctly
- your client machines are configured to talk to the CFM server
- communication works between your CFM server and client machines

Engineer
a solution!

MOTD Cookbook

cookbooks/motd/recipes/default.rb:

```
file "/etc/motd" do
  content "
This system is managed by #{node[:company][:name]}.
All activity may be monitored and reported.
Authorized use only.
"
  mode 0644
end
```

MOTD Cookbook

cookbooks/motd/recipes/default.rb:

```
template "/etc/motd" do
  source "motd.erb"
  owner 0
  group 0
  mode 0644
end
```

MOTD Cookbook

Ohai line:

```
<%= node[:kernel][:os] %> <%= node[:kernel][:release] %> (<%= node[:kernel][:ident] %>) #<%= node[:kernel][:version].split('#')[1].split(' ').first %>
```

What next?

Engineer a solution!

2DGOGGLES.COM

System Cookbook

"Installs/Configures key/value system files"

"system::boot", "Sets boot parameters"

"system::logrotate", "Configures log rotation"

"system::make", "Provides parameters for building packages"

"system::periodic", "Makes periodic job configuration"

"system::sysctl", "Tunes kernel parameters"

"system::syslog", "Sets-up system logging"

System Cookbook

cookbooks/system/templates/default/sysctl.conf.erb:

```
<% unless node[:system][:sysctl].class === nil -%>
<% node[:system][:sysctl].each do |k,v| -%>
<%= k %>=<%= v %>
<% end -%>
<% end -%>
```

System Cookbook

cookbooks/system/templates/freebsd/loader.conf.erb:

```
<% unless node[:system][:boot].class === nil -%>
<% node[:system][:boot].each do |k,v| -%>
<%= k %>="<%= v %>"
<% end -%>
<% end -%>
```

MOTD Cookbook

TMTOWTDI

cookbooks/motd/templates/default/motd.erb:

<% if node[:platform] == "**freebsd**" -%>

cookbooks/motd/templates/**freebsd**/motd.erb

So far...

- working CFM
- MOTD
- ”system” cookbook

Engineer

a solution!

2DGOGGLES.COM

Roles

- FreeBSD
- Solaris
- Linux
- Base
- DC1 / DC2

Base Role

```
name "base"
description "This is the base role."

override_attributes(
  "motd" => {
    "managed_by" => "EveryWare AG"
  },
```

Base Role

```
  "postfix" => {
    "mail_type" => "client-base"
  }
)
```

Base Role

```
run_list(
  "recipe[chef-client::delete_validation]",
  "recipe[chef-client]",
  "recipe[motd]",
  "recipe[resolver]",
  "recipe[ntp]",
  "recipe[postfix]"
)
```

Engineer a solution!

lib/chef/provider/package/freebsd.rb

```
def load_current_resource
  …
     begin
       @candidate_version = ports_candidate_version
     rescue
       @candidate_version = file_candidate_version
     end
```

lib/chef/provider/package/freebsd.rb

```
def file_candidate_version
  file_candidate_version_path.split(/-/).last.split(/.tbz/).first
end
```

lib/chef/provider/package/freebsd.rb

```
def file_candidate_version_path
    Dir["#{@new_resource.source}/
    #{@current_resource.package_name}*"][0].to_s
end
```

lib/chef/provider/package/freebsd.rb

```
def install_package(name, version)
  …
    when /^\//
      shell_out!("pkg_add
        #{file_candidate_version_path}", :env =>
        { "PKG_PATH" => @new_resource.source ,
  'LC_ALL'=>nil}).status
```

lib/chef/provider/package/freebsd.rb

```
    Chef::Log.debug("Current version is
#{@current_resource.version}") if
@current_resource.version

    Chef::Log.debug("Ports candidate version is
#{@candidate_version}") if @candidate_version

    Chef::Log.info("Installed package
#{@new_resource.name} from:
#{@new_resource.source}")
```

Postfix Cookbook

```
if node[:postfix][:mail_type] == "client-base"
  package "postfix-base" do
    source "/usr/ports/packages/All"
    action :install
  end
end
```

Base Role

```
run_list(
...
  "recipe[postfix]"
)
```

DC1 Role

```
override_attributes(

  "resolver" => {
    "nameservers" => ["x", "y"],
    "search" => "DC1"
  },
```

DC1 Role

```
"ntp" => {
  "servers" => ["ntp1","ntp2"]
},
```

DC1 Role

```
"postfix" => {
  "relayhost" => "relay1"
},
```

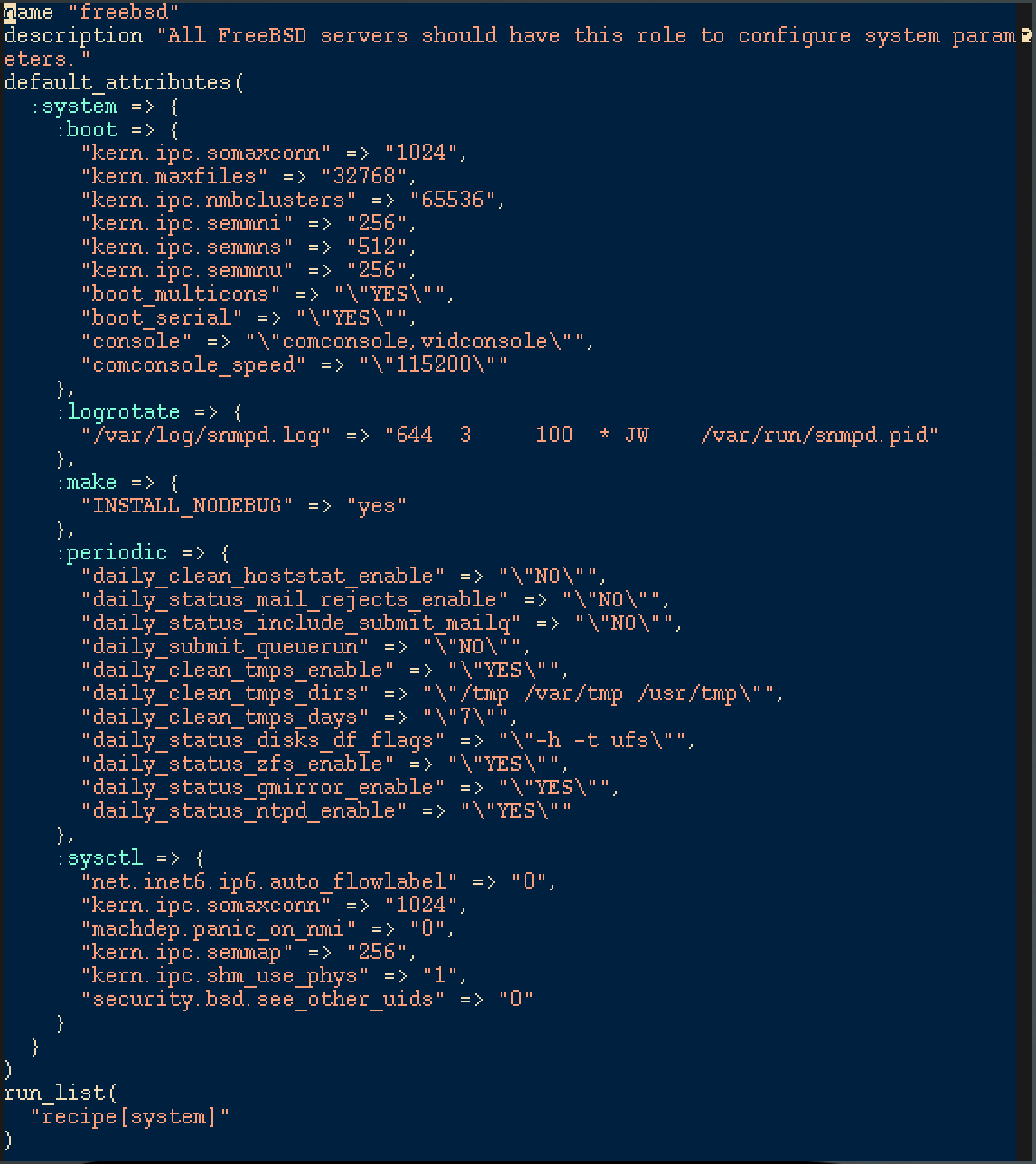
DC1 Role

```
  "system" => {
    "syslog" => { "*.*" => "@syslog1" }
  }
)
```

DC1 Role

```
run_list(
  "role[base]"
)
```

FreeBSD Role



```
name "freebsd"
description "All FreeBSD servers should have this role to configure system param
eters."
default_attributes(
  :system => {
    :boot => {
      "kern.ipc.somaxconn" => "1024",
      "kern.maxfiles" => "32768",
      "kern.ipc.nmbclusters" => "65536",
      "kern.ipc.semmni" => "256",
      "kern.ipc.semmns" => "512",
      "kern.ipc.semmnu" => "256",
      "boot_multicons" => "\"YES\"",
      "boot_serial" => "\"YES\"",
      "console" => "\"comconsole,vidconsole\"",
      "comconsole_speed" => "\"115200\""
    },
    :logrotate => {
      "/var/log/snmpd.log" => "644  3     100  * JW    /var/run/snmpd.pid"
    },
    :make => {
      "INSTALL_NODEBUG" => "yes"
    },
    :periodic => {
      "daily_clean_hoststat_enable" => "\"NO\"",
      "daily_status_mail_rejects_enable" => "\"NO\"",
      "daily_status_include_submit_mailq" => "\"NO\"",
      "daily_submit_queuerun" => "\"NO\"",
      "daily_clean_tmps_enable" => "\"YES\"",
      "daily_clean_tmps_dirs" => "\"/tmp /var/tmp /usr/tmp\"",
      "daily_clean_tmps_days" => "\"7\"",
      "daily_status_disks_df_flags" => "\"-h -t ufs\"",
      "daily_status_zfs_enable" => "\"YES\"",
      "daily_status_gmirror_enable" => "\"YES\"",
      "daily_status_ntpd_enable" => "\"YES\""
    },
    :sysctl => {
      "net.inet6.ip6.auto_flowlabel" => "0",
      "kern.ipc.somaxconn" => "1024",
      "machdep.panic_on_nmi" => "0",
      "kern.ipc.semmap" => "256",
      "kern.ipc.shm_use_phys" => "1",
      "security.bsd.see_other_uids" => "0"
    }
  }
)
run_list(
  "recipe[system]"
)
```

FreeBSD Role

```
name "freebsd"
description "All FreeBSD servers should have this
   role to configure system parameters."
default_attributes(
  :system => {
```

FreeBSD Role

```
:boot => {
"kern.ipc.somaxconn" => "1024",
"kern.maxfiles" => "32768",
"kern.ipc.nmbclusters" => "65536",
"kern.ipc.semmni" => "256",
"kern.ipc.semmns" => "512",
"kern.ipc.semmnu" => "256",
"boot_multicons" => "YES",
"boot_serial" => "YES",
"console" => "comconsole,vidconsole",
"comconsole_speed" => "115200"
},
```

FreeBSD Role

```
 :logrotate => {
   "/var/log/snmpd.log" => "644  3     100  * JW /var/run/snmpd.pid"
 },
 :make => {
   "INSTALL_NODEBUG" => "yes"
 },
```

FreeBSD Role

```
:periodic => {
"daily_clean_hoststat_enable" => "NO",
"daily_status_mail_rejects_enable" => "NO",
"daily_status_include_submit_mailq" => "NO",
"daily_submit_queuerun" => "NO",
"daily_clean_tmps_enable" => "YES",
"daily_clean_tmps_dirs" => "/tmp /var/tmp /usr/tmp",
"daily_clean_tmps_days" => "7",
"daily_status_disks_df_flags" => "-h -t ufs",
"daily_status_zfs_enable" => "YES",
"daily_status_gmirror_enable" => "YES",
"daily_status_ntpd_enable" => "YES"
},
```

FreeBSD Role

```
  :sysctl => {
   "net.inet6.ip6.auto_flowlabel" => "0",
   "kern.ipc.somaxconn" => "1024",
   "machdep.panic_on_nmi" => "0",
   "kern.ipc.semmap" => "256",
   "kern.ipc.shm_use_phys" => "1",
   "security.bsd.see_other_uids" => "0"
  }
 }
)
```

FreeBSD Role

```
run_list(
  "recipe[system]"
)
```

Roles & Cookbooks

http://www.flickr.com/photos/library_of_congress/6442021039/

Workflow Integration

- New servers get chef-client installed at bootstrap
- Roles and recipes configured per node
- Server update => tie into Chef
- Configuration saved in Revision Control

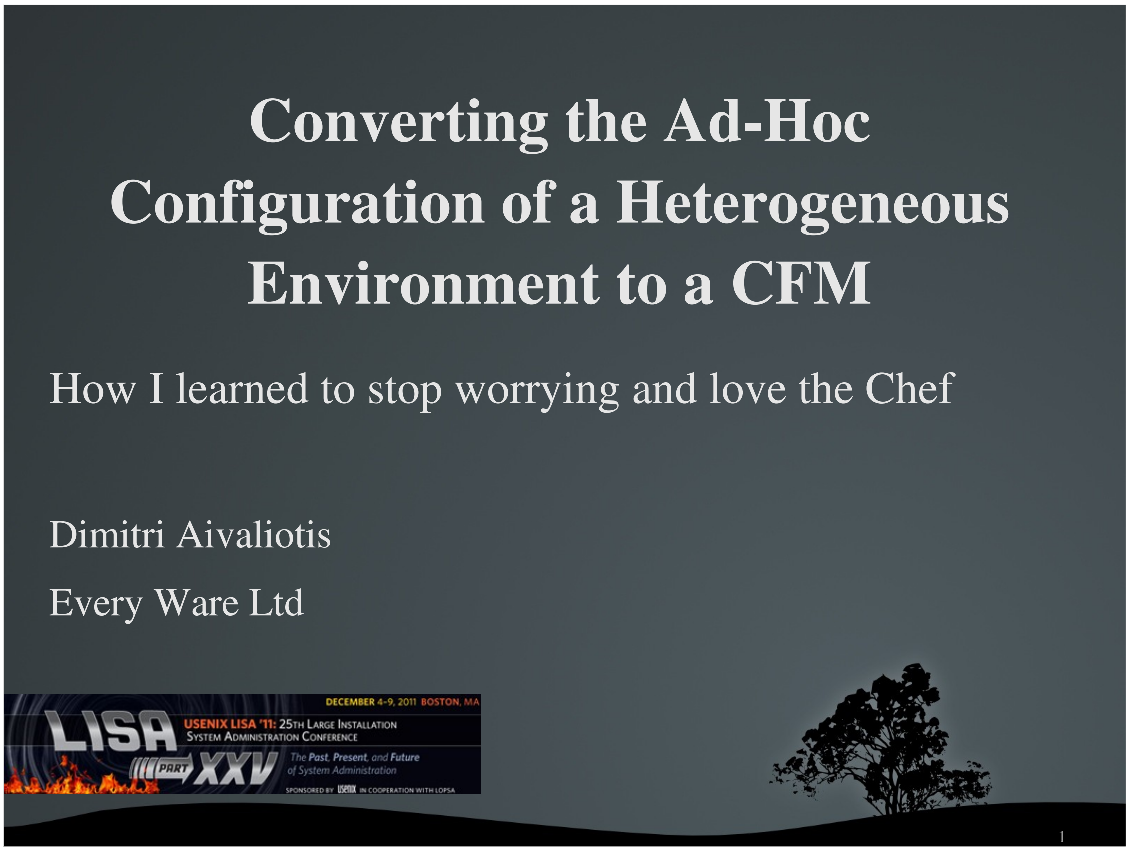
Recap...

Any Questions?

d.n.a@acm.org

Thanks to Opscode for Chef and to Sydney Padua

for the Brunel images (http://2dgoggles.com)



Converting the Ad-Hoc Configuration of a Heterogeneous Environment to a CFM

How I learned to stop worrying and love the Chef

Dimitri Aivaliotis

Every Ware Ltd

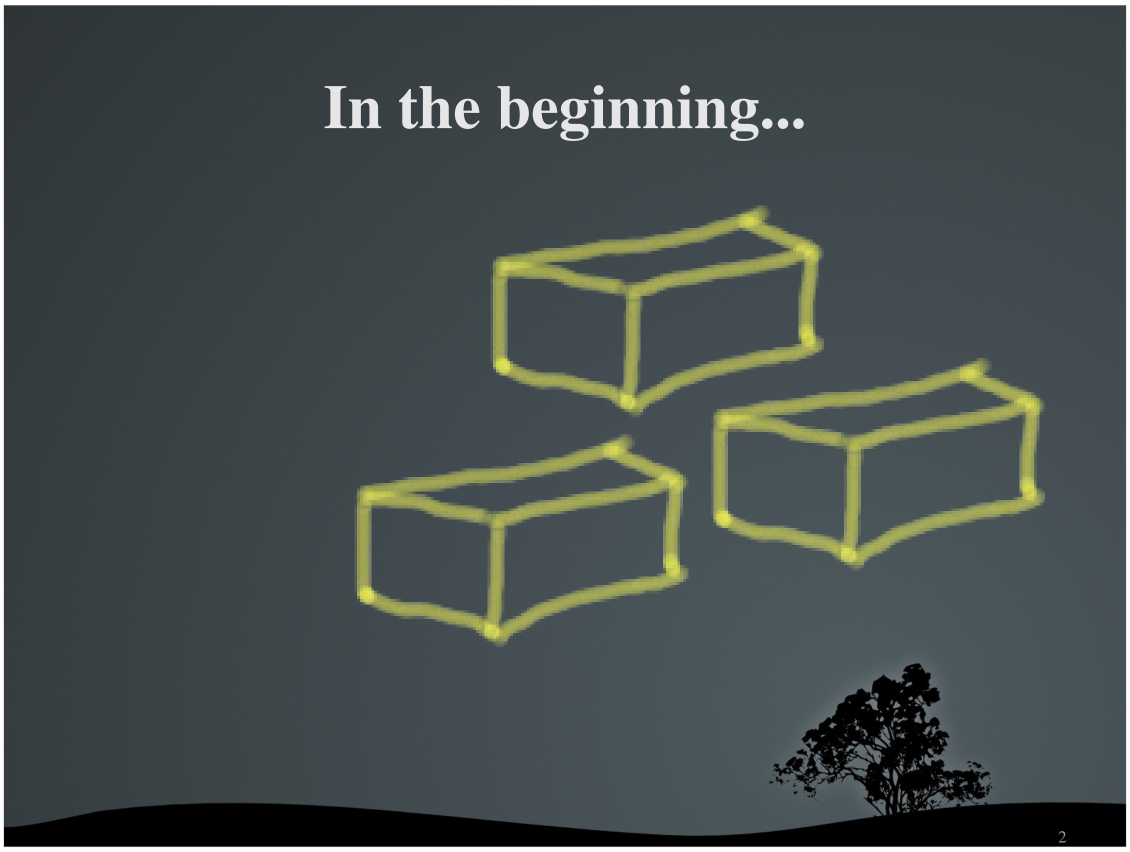
DECEMBER 4–9, 2011 BOSTON, MA

LISA PART XXV

USENIX LISA '11: 25th Large Installation System Administration Conference

The Past, Present, and Future of System Administration

SPONSORED BY USENIX IN COOPERATION WITH LOPSA



In the beginning...

It all started many years ago, back when there were only a handful of servers to manage.

As a lone admin, it was easy to develop a manual system of configuring each server, changing it as I learned more and our customers' needs changed.

Changes are propagated by doing the same thing across that handful of servers.

(Does this describe anybody's current configuration management system?)

As time goes on...

Eventually though, that number grows to the point where you can't even hold it in two hands.

And multiple admins get added to the mix.

Then you've reached the point where you know that things can't go on like this; that something has to change.

Chef

4

Enter Chef, the configuration management system.
As a CFM, Chef can help you codify the manual system that you developed and grew years ago.
But, it is a tool. A tool that can help you perform certain tasks better and easier.
It will not solve all your problems. It will not fit exactly into how you do things now.
But, Chef is Open Source. You can make it your own.

This is the story of how I used Chef to automate the configuration of the diverse systems under my care.

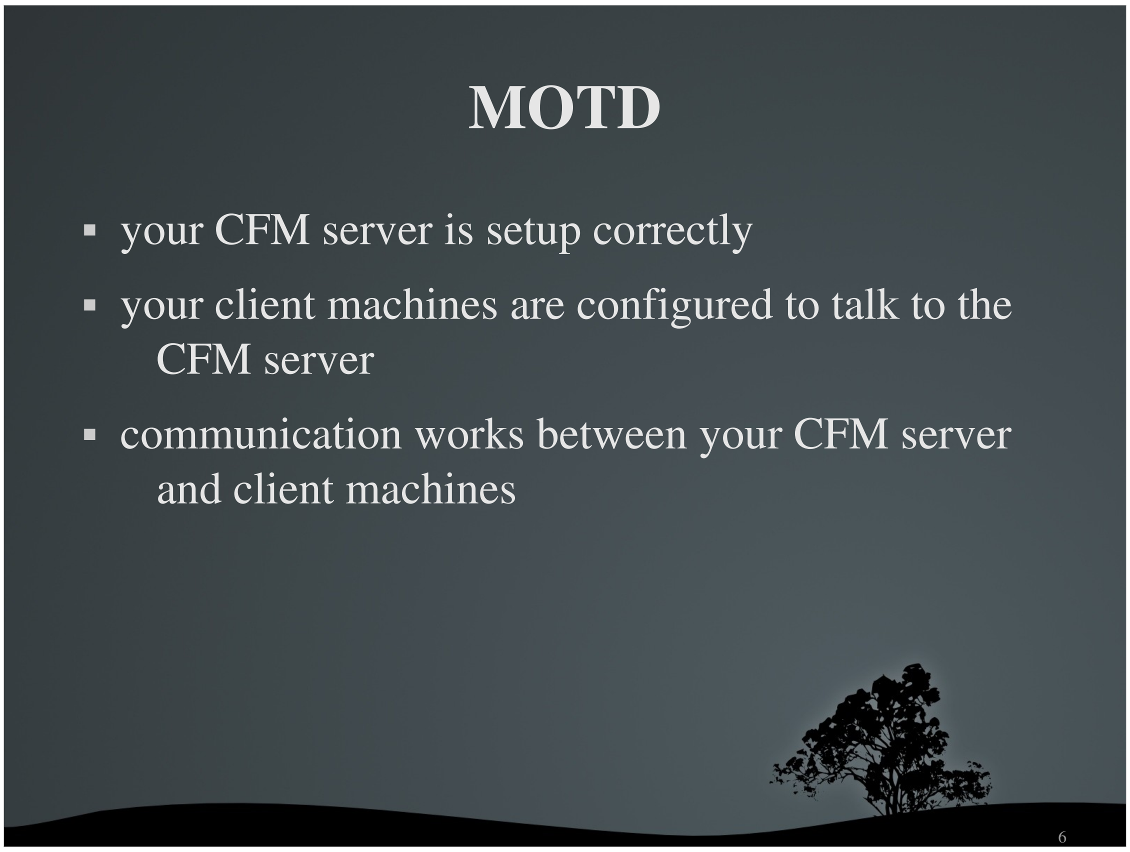
Where to begin?

Configuration management is such a huge topic and there are so many solutions to this problem, that you just have to dive in and start using it.

Back at LISA '09, I attended the Configuration Management Workshop.

(How many of you attended it this year?)

One of the organizers, Cory, gave us some practical advice. He said to start with the Message of the Day.



MOTD

- your CFM server is setup correctly
- your client machines are configured to talk to the CFM server
- communication works between your CFM server and client machines

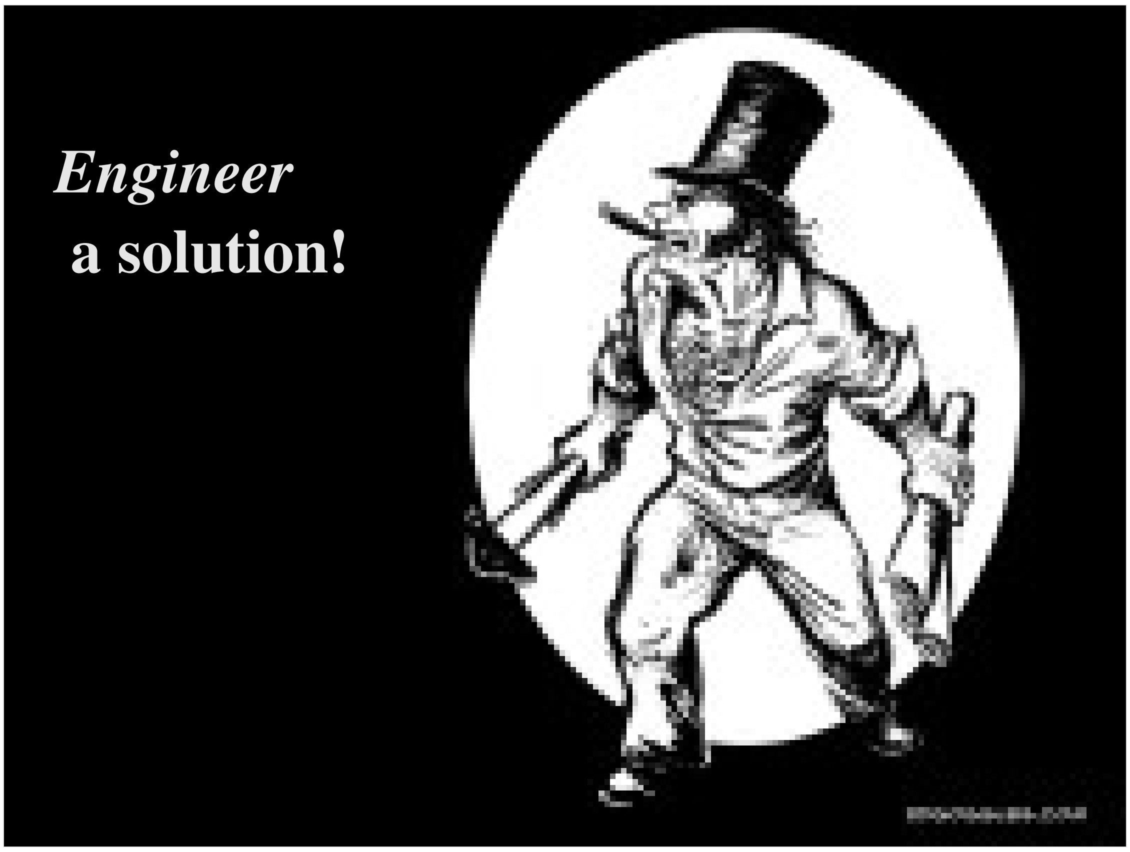
6

If you've got the MOTD file under control of a CFM, then you know that you have some basic requirements fulfilled:

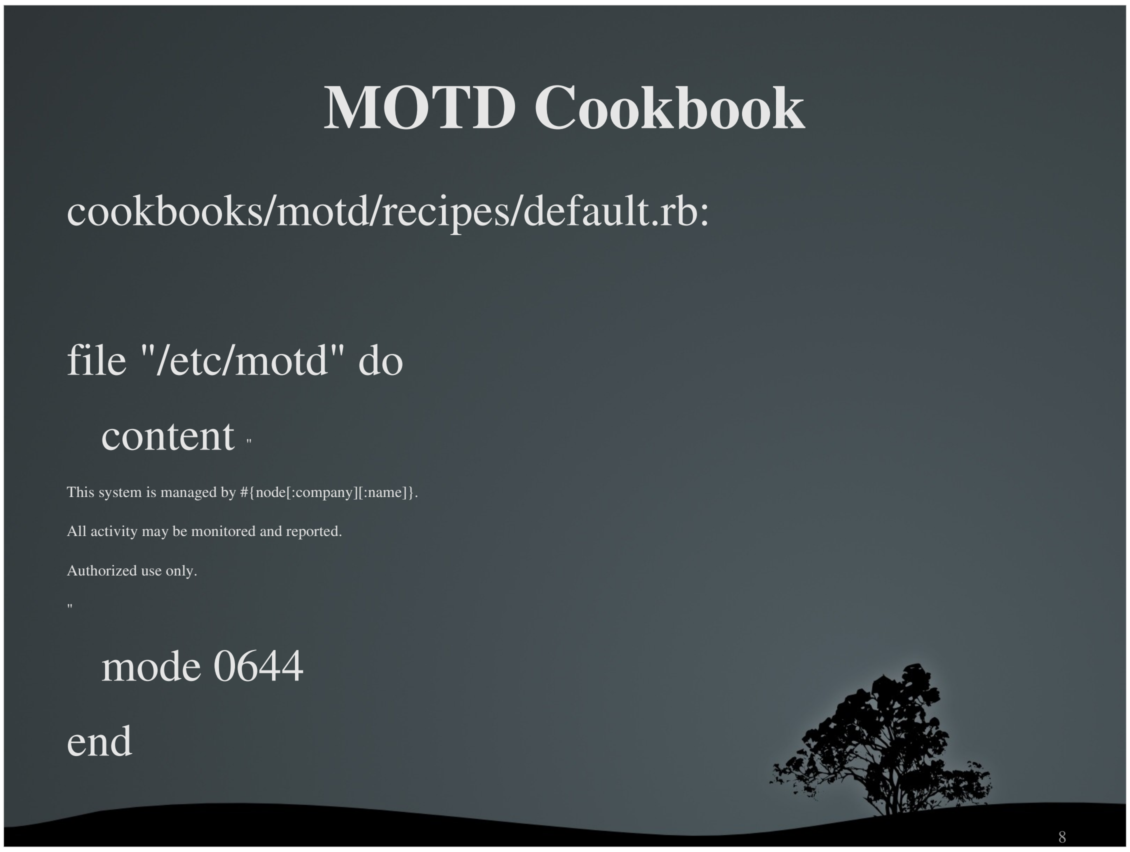
you know that...

So, I thought ”great, now I just need to get the motd cookbook and configure it”. Except, there was no motd cookbook...

What do you do when faced with a situation like this? You want to use something that doesn't exist.



make it your own



MOTD Cookbook

cookbooks/motd/recipes/default.rb:

```
file "/etc/motd" do
   content "
This system is managed by #{node[:company][:name]}.
All activity may be monitored and reported.
Authorized use only.
"
   mode 0644
end
```

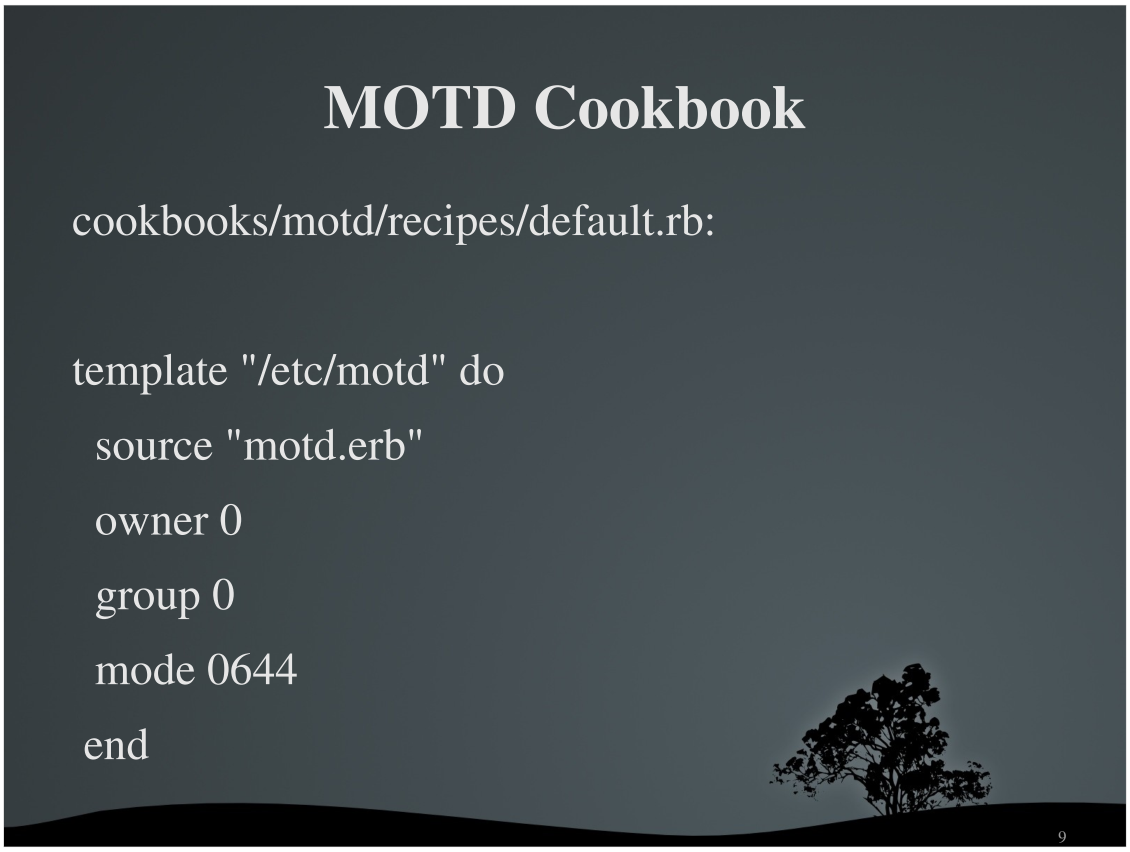
8

Of course, I first implemented the MOTD cookbook for FreeBSD systems, as most of the systems I manage are FreeBSD.

This presented some challenges because the default MOTD under FreeBSD has a line showing the running kernel, which is changed at boot, if needed.

This originally led to my /etc/motd being overwritten at boot, then overwritten again by Chef; to be repeated at each boot.

So, I needed a way to keep the file from constantly being changed.

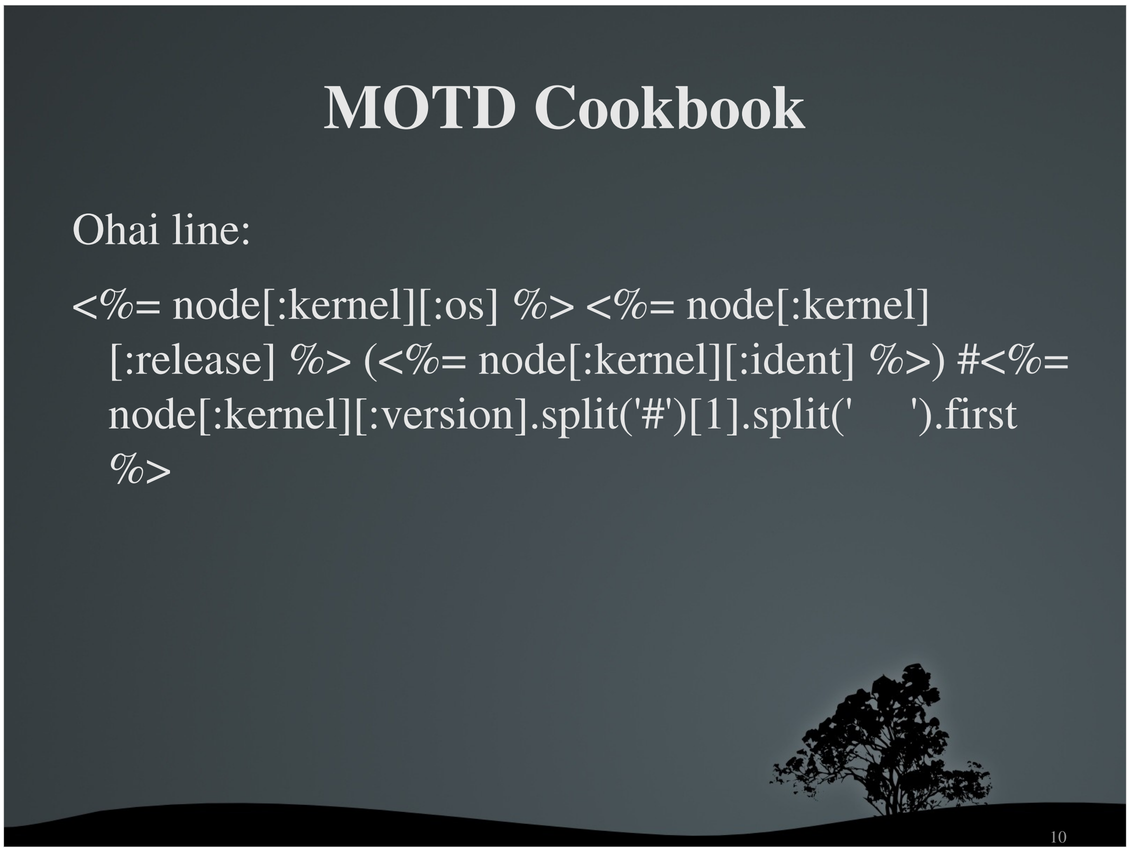


MOTD Cookbook

cookbooks/motd/recipes/default.rb:

```
template "/etc/motd" do
  source "motd.erb"
  owner 0
  group 0
  mode 0644
 end
```

I exchanged the ”file” resource for a ”template” resource, which made the recipe much cleaner.
But also enabled me to satisfy FreeBSD's default behavior.



MOTD Cookbook

Ohai line:

```
<%= node[:kernel][:os] %> <%= node[:kernel]
[:release] %> (<%= node[:kernel][:ident] %>) #<%=
node[:kernel][:version].split('#')[1].split('     ').first
%>
```

10

I added a line to the template file, which referenced the requisite kernel information from Ohai.
This satisfied both the boot process and my cookbook, so I walked away happy.
My first cookbook, and it works!

What next?

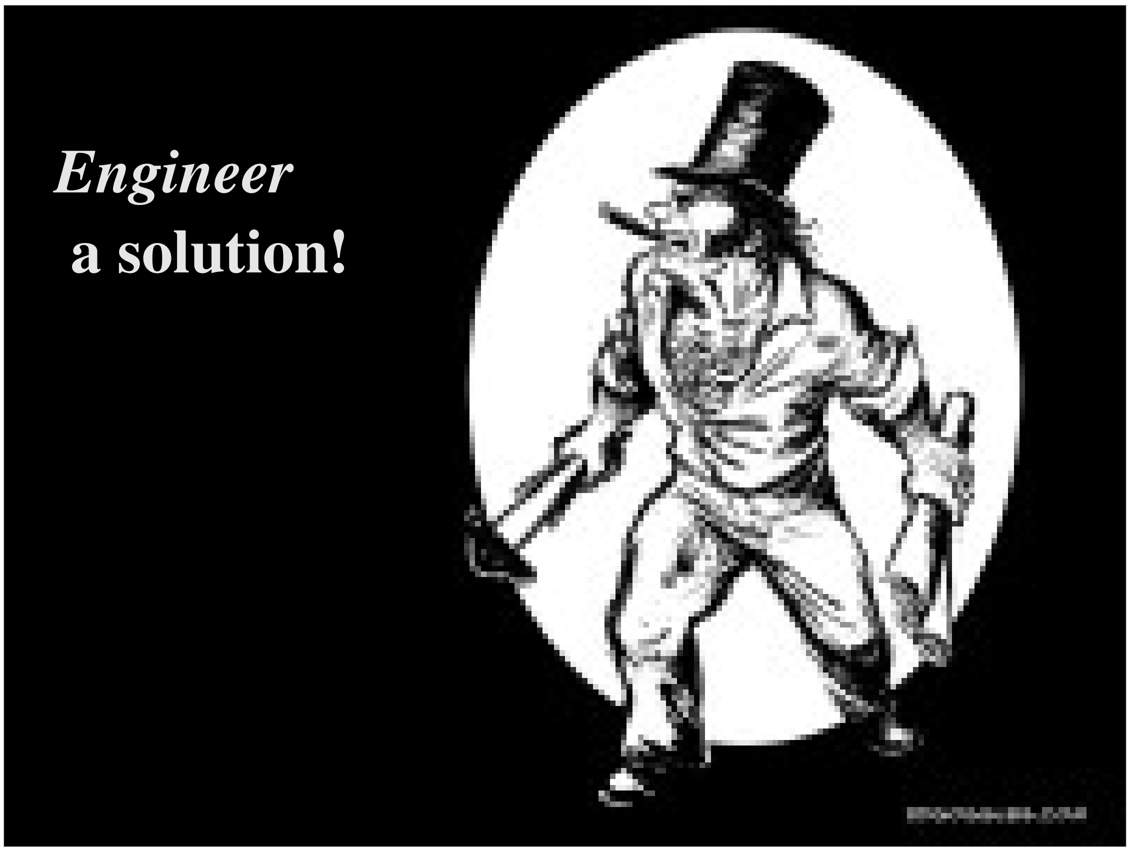
11

Well, we've got a working CFM system and a system file managed under it. What other things could I use it for?

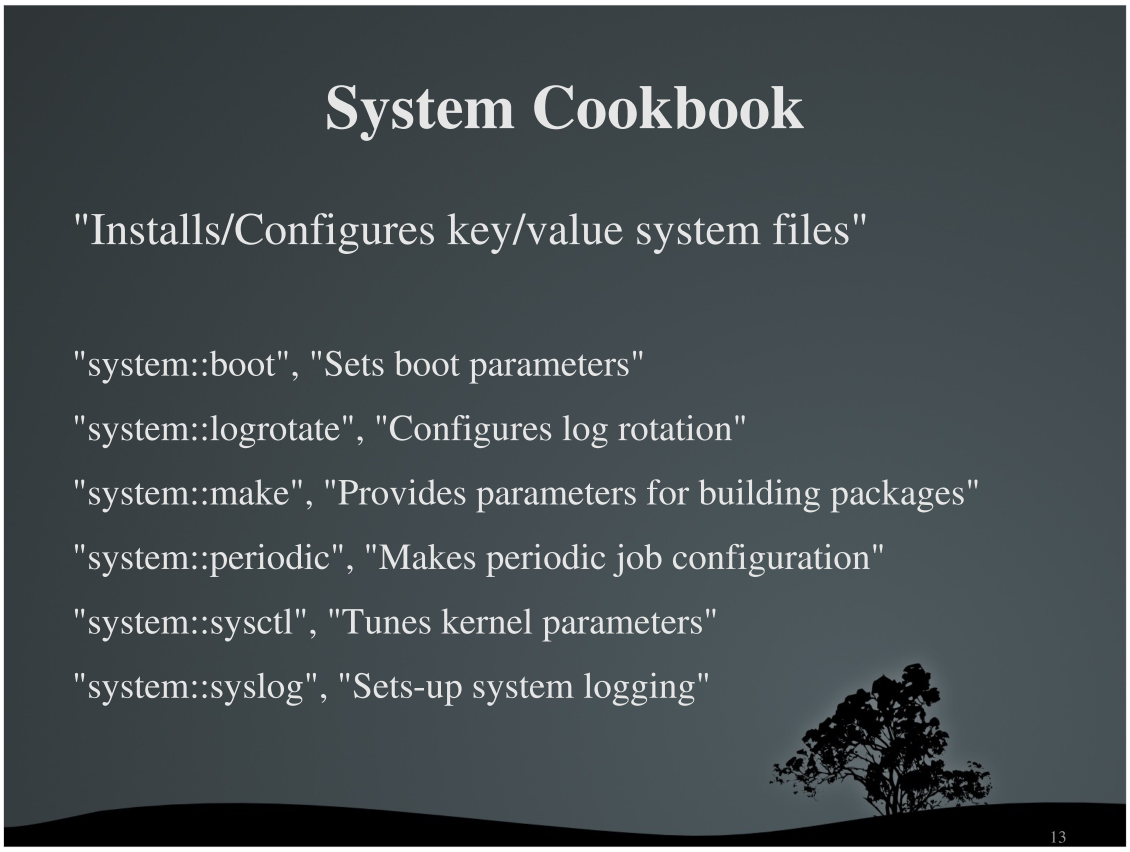
There are a few files with multiple non-mutually-exclusive parameters that can be set in them.

Under FreeBSD, you've got /boot/loader.conf, /etc/syslog.conf, /etc/newsyslog.conf, /etc/make.conf, and /etc/periodic.conf, and additionally you've got /etc/sysctl.conf under Linux as well.

There was no cookbook to manage all these files together, so what do we do?



make it your own



System Cookbook

"Installs/Configures key/value system files"

"system::boot", "Sets boot parameters"

"system::logrotate", "Configures log rotation"

"system::make", "Provides parameters for building packages"

"system::periodic", "Makes periodic job configuration"

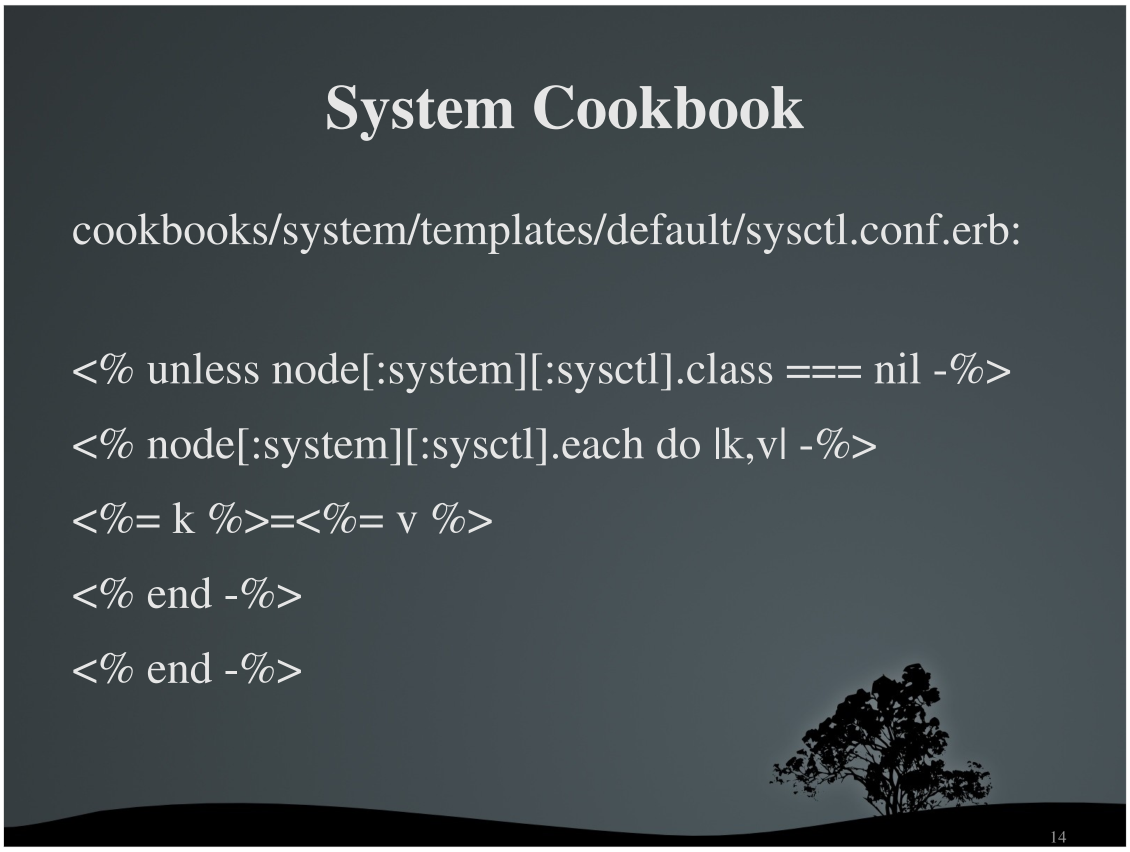
"system::sysctl", "Tunes kernel parameters"

"system::syslog", "Sets-up system logging"

13

So, I wrote another cookbook to manage all these system files.

The aptly-named ”system” cookbook has at its heart an assortment of key/value templates. This is due to FreeBSD's adherence to the true UNIX spirit of configuration files.

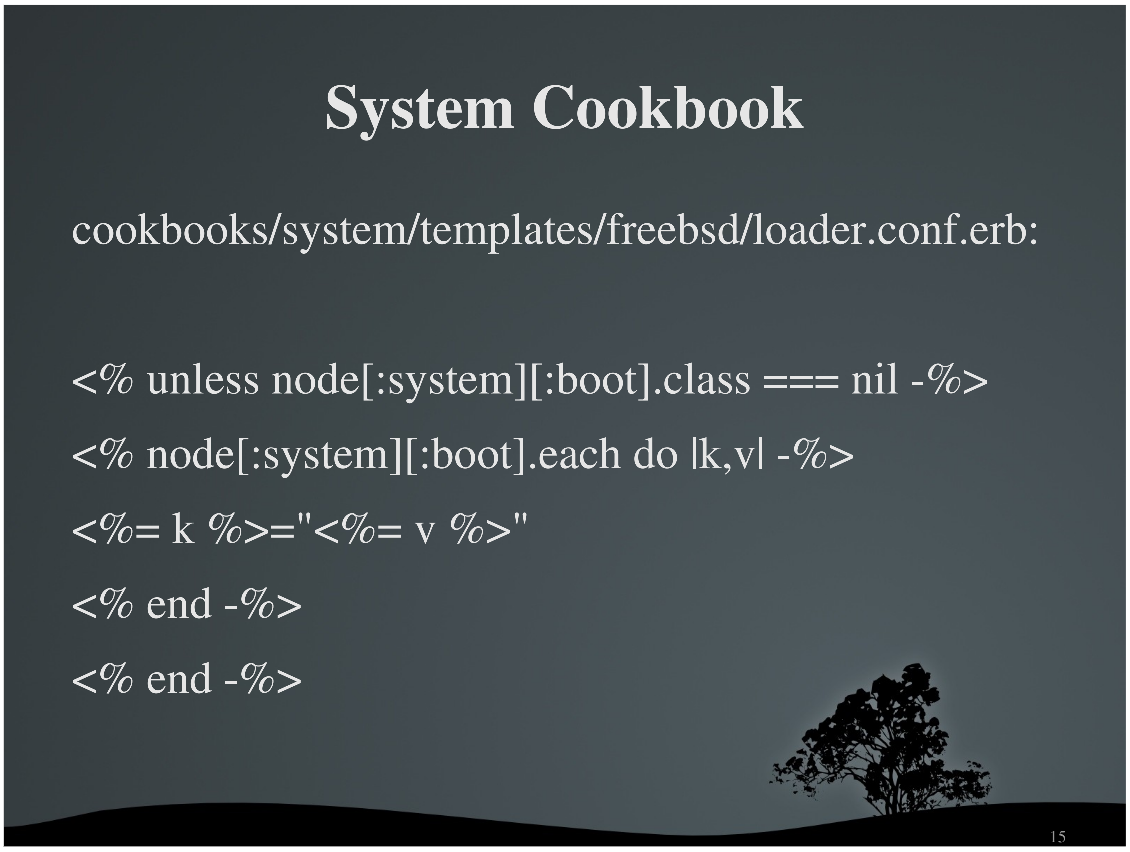


System Cookbook

cookbooks/system/templates/default/sysctl.conf.erb:

```
<% unless node[:system][:sysctl].class === nil -%>
<% node[:system][:sysctl].each do |k,v| -%>
<%= k %>=<%= v %>
<% end -%>
<% end -%>
```

This works for Linux, too. So, we can put the /etc/sysctl.conf template under the ”default” directory.

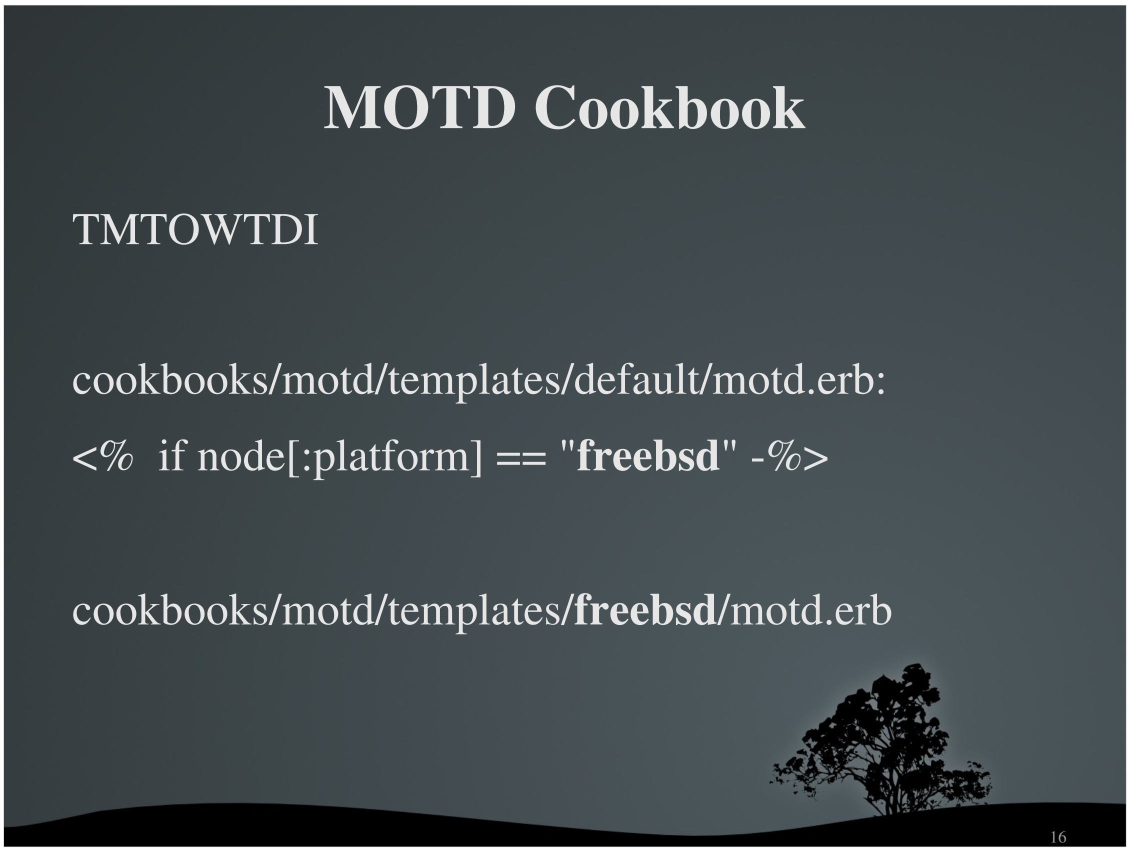


System Cookbook

cookbooks/system/templates/freebsd/loader.conf.erb:

```
<% unless node[:system][:boot].class === nil -%>
<% node[:system][:boot].each do |k,v| -%>
<%= k %>="<%= v %>"
<% end -%>
<% end -%>
```

Can you spot the difference?
This is why we used a separate template file for /boot/loader.conf.
We placed this template under the ”freebsd” directory because of what Chef refers to as ”file specificity”.
Now, what does this mean?
Those of you familiar with cfengine will know this as ”Single Copy Nirvana”.
This means that the more specific a change to a template file needs to be, it gets placed into a directory matching that level of specificity.
(Illustrate difference between these slides again.)



MOTD Cookbook

TMTOWTDI

cookbooks/motd/templates/default/motd.erb:

<% if node[:platform] == "**freebsd**" -%>

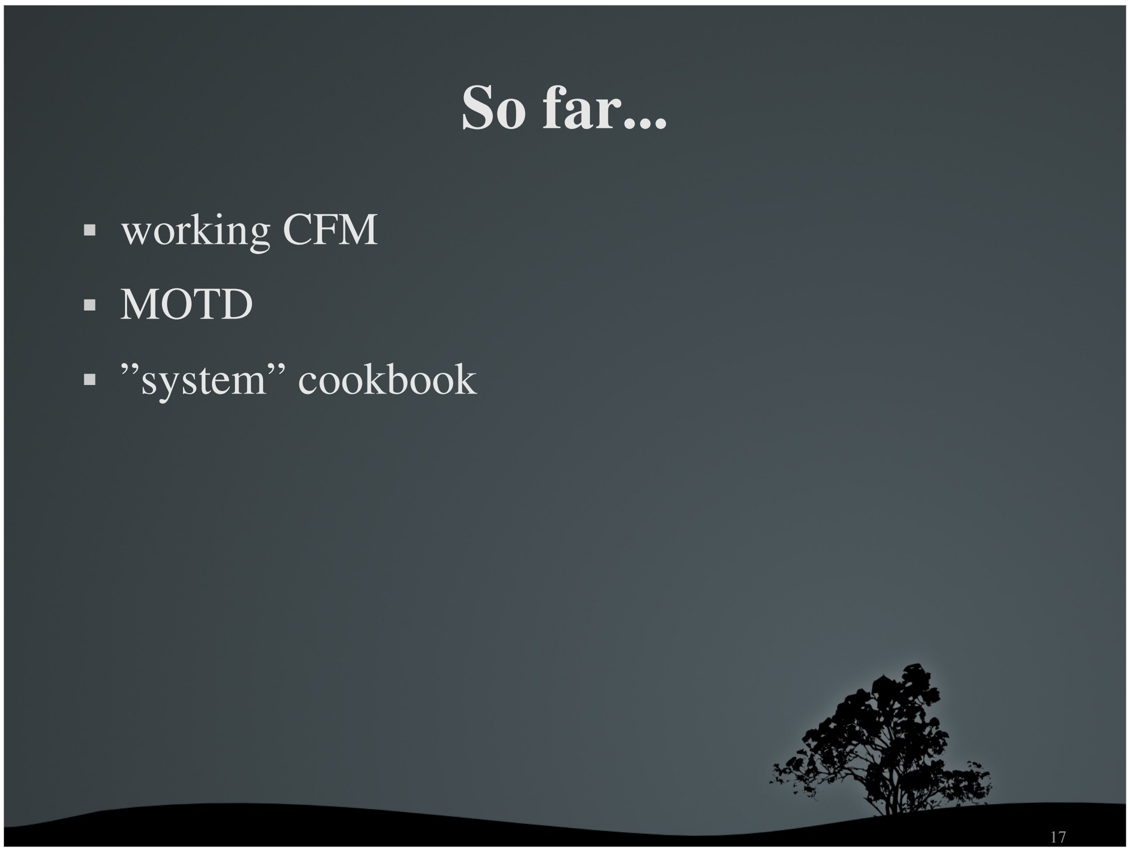
cookbooks/motd/templates/**freebsd**/motd.erb

16

Remember our MOTD cookbook?

Now, we can go back and make the implementation more elegant.

Instead of an ”if” clause, we can use file specificity.



So far...

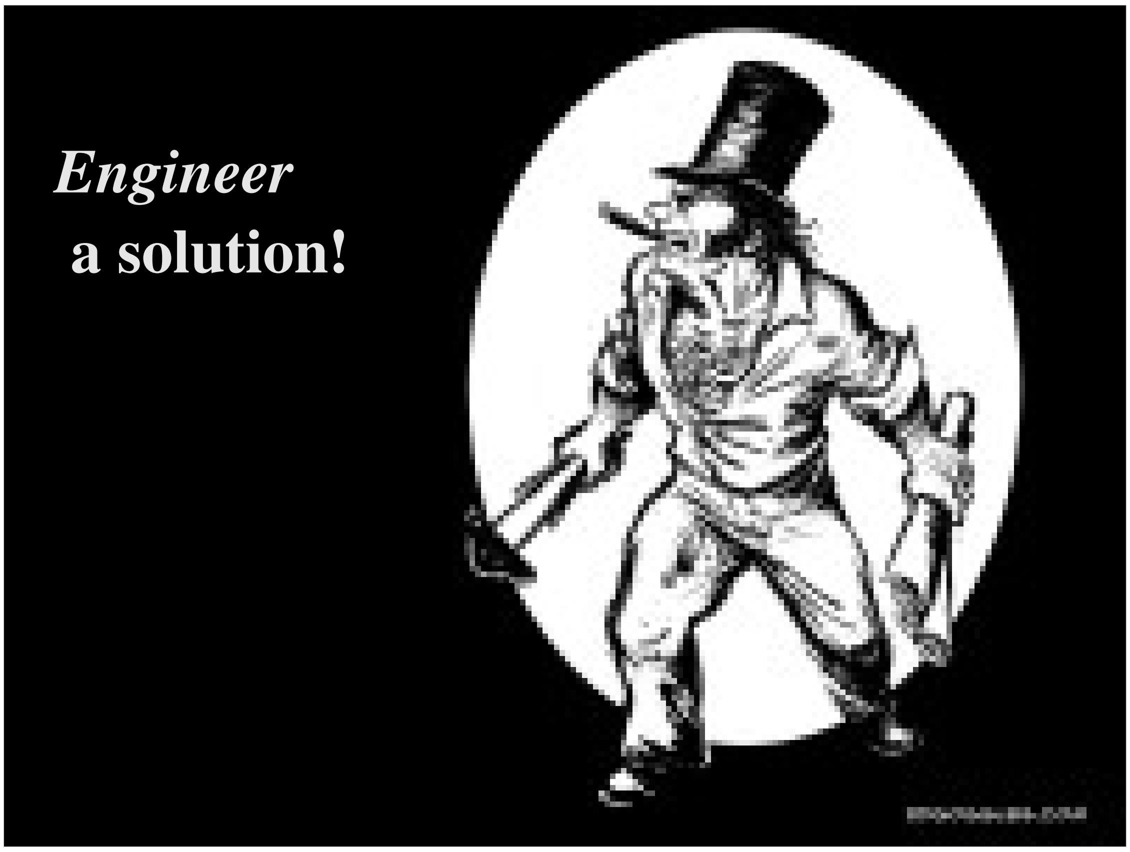
- working CFM
- MOTD
- ”system” cookbook

17

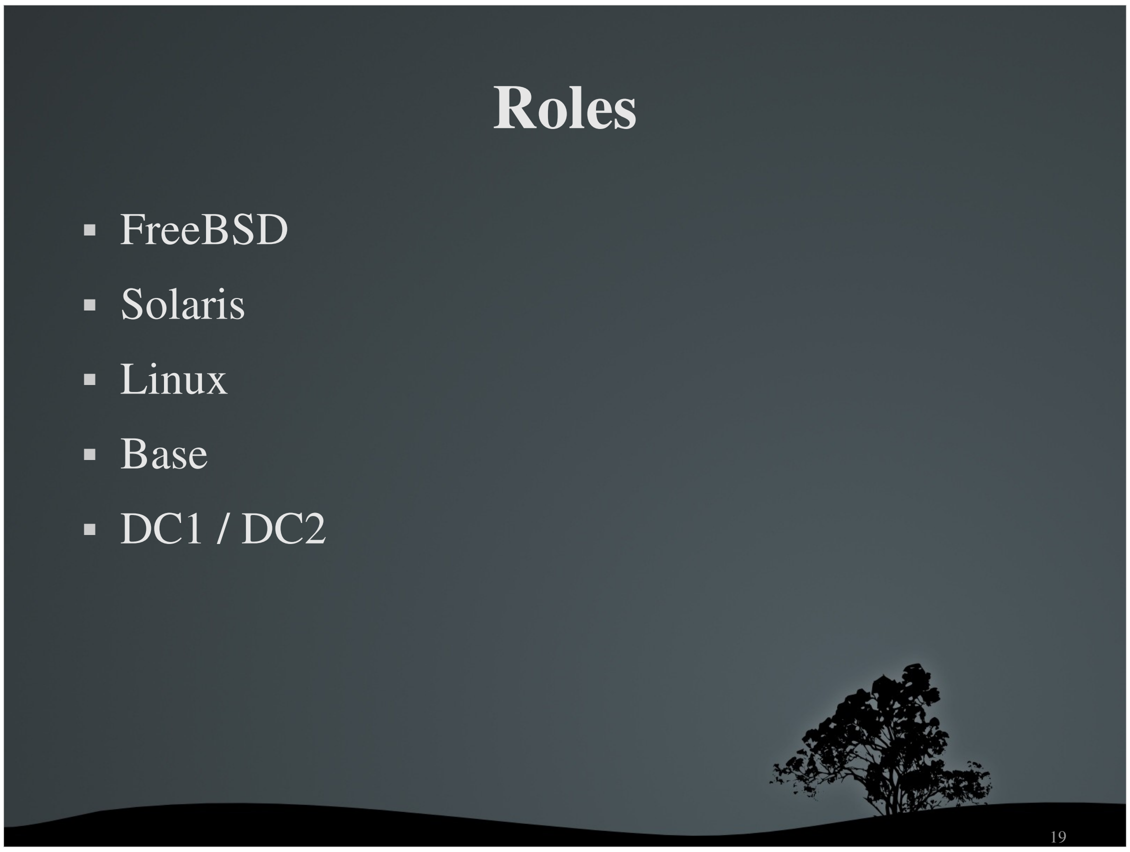
OK, what have we got so far?
We have a...

We've seen the template files for these, but how do the key/value pairs actually get substituted in?

We...



make it your own



Roles

- FreeBSD
- Solaris
- Linux
- Base
- DC1 / DC2

We're going to do that by using ”roles”.
Now, roles can be used for many things. At its most basic, a role is nothing more than a label:
 - these systems are FreeBSD
 - these are Solaris
 - and these over here are running Linux
 - this is a base set of values that all systems should start out with
 - and these are located in Datacenter 1, those in Datacenter 2

Let's look at these roles more in-depth.

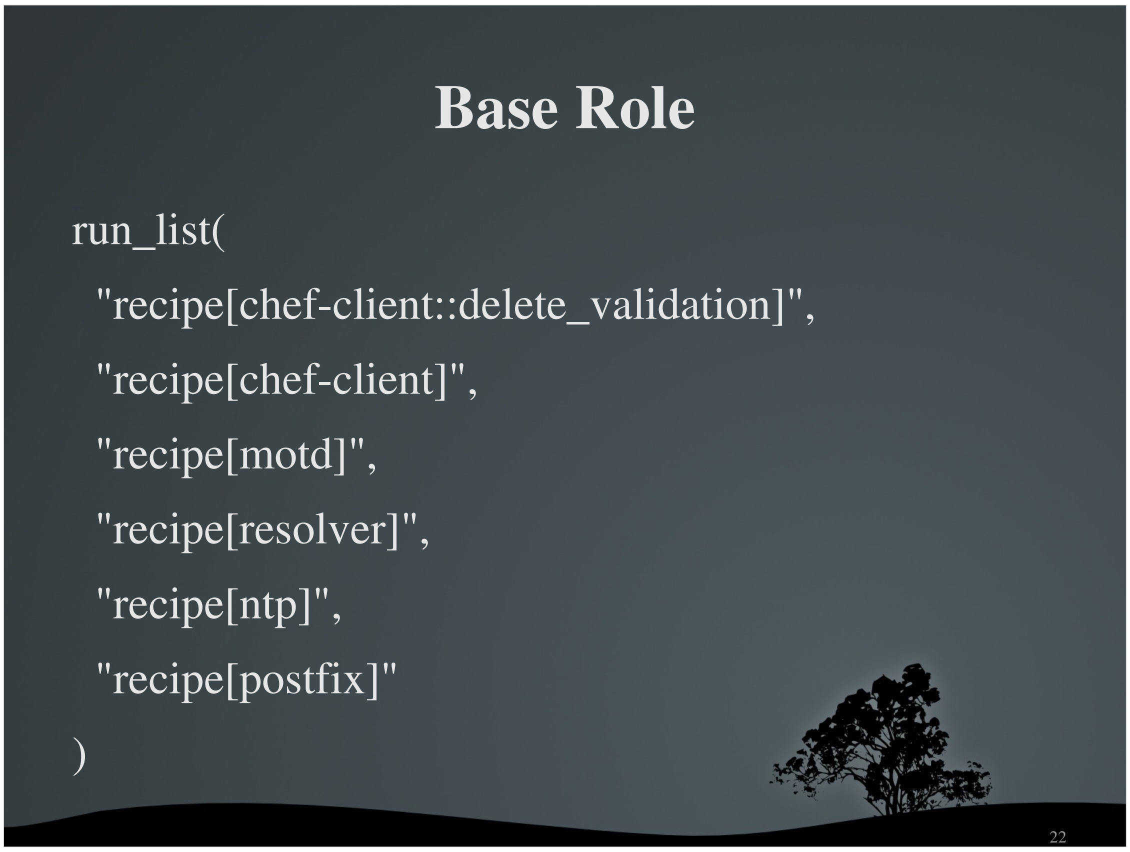
Base Role

```
name "base"
description "This is the base role."

override_attributes(
  "motd" => {
    "managed_by" => "EveryWare AG"
  },
```

Base Role

```
  "postfix" => {
    "mail_type" => "client-base"
  }
)
```



Base Role

```
run_list(
  "recipe[chef-client::delete_validation]",
  "recipe[chef-client]",
  "recipe[motd]",
  "recipe[resolver]",
  "recipe[ntp]",
  "recipe[postfix]"
)
```

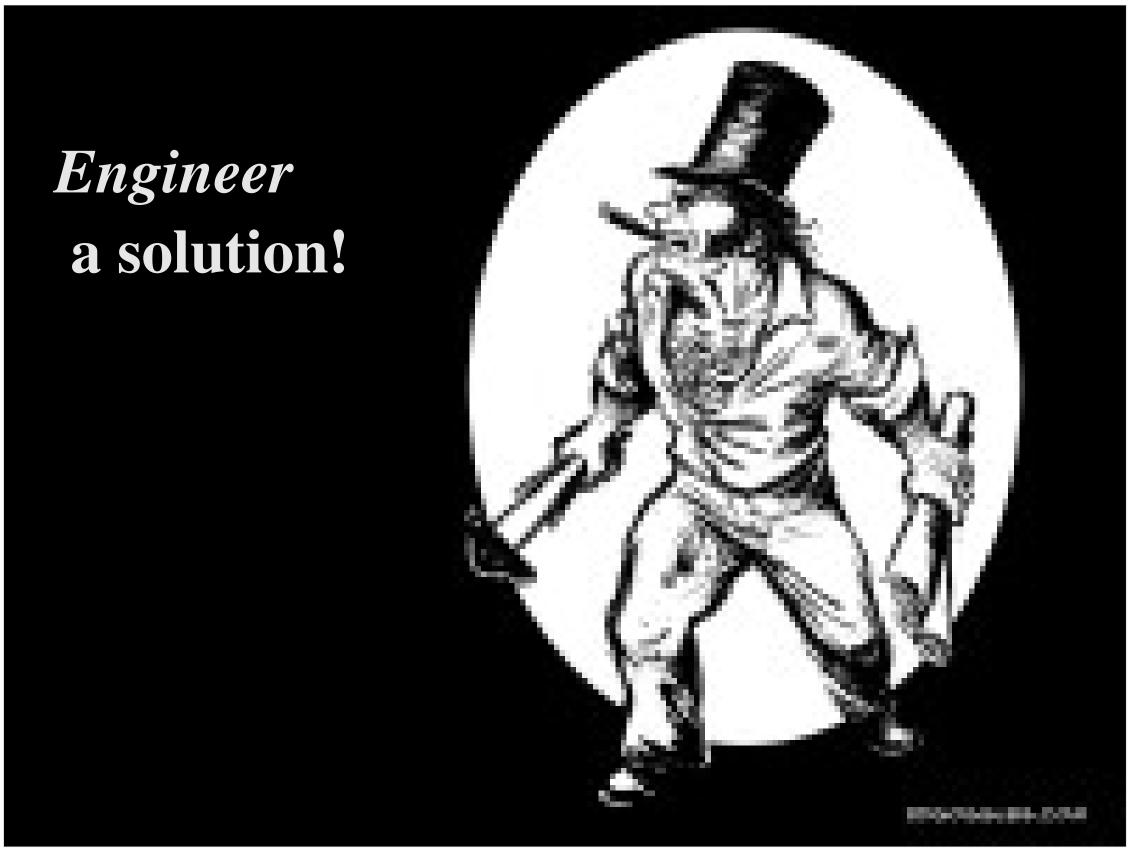
22

Oh, but wait...

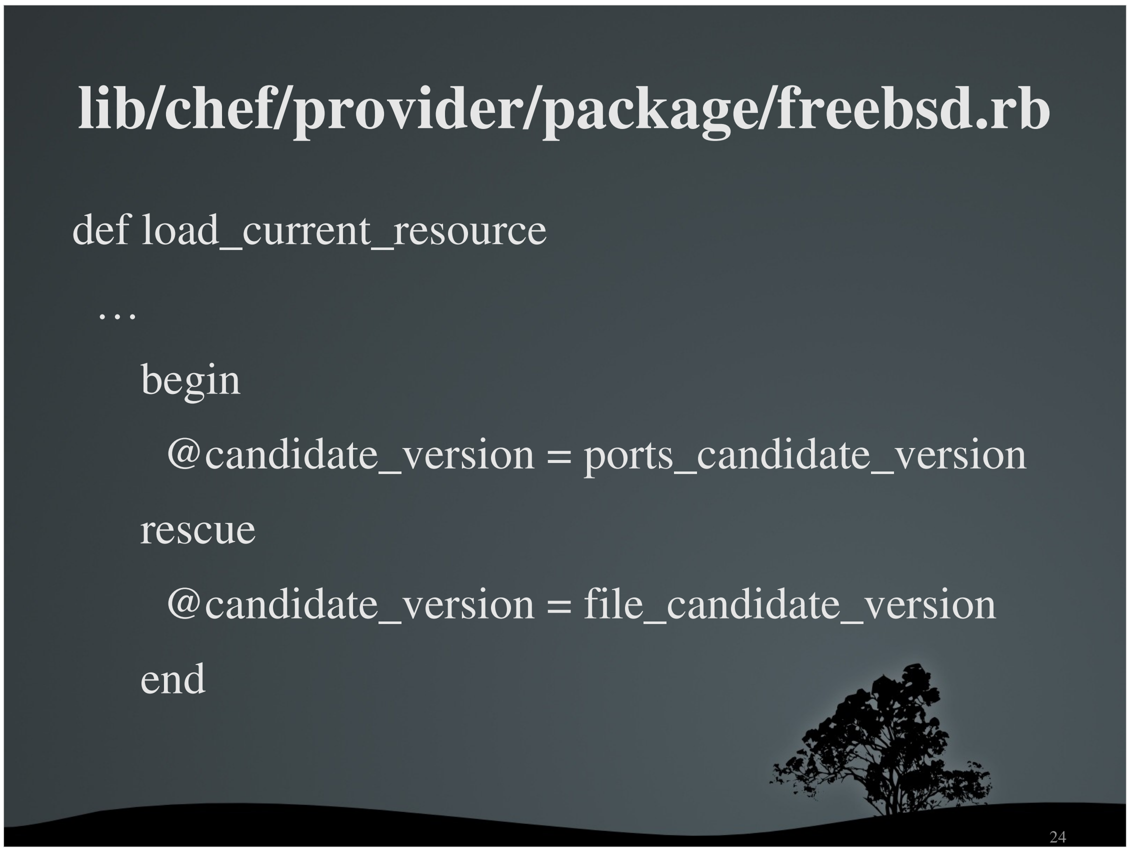
Our base role calls for the postfix default recipe to be run. But, that involves installing postfix.

Chef tries to install it, but can't find the ports directory for postfix because our new installs only install the minimal distribution set.

What to do, what to do?



make it your own



lib/chef/provider/package/freebsd.rb

```
def load_current_resource
  …
      begin
        @candidate_version = ports_candidate_version
      rescue
        @candidate_version = file_candidate_version
      end
```

That's right. Chef's OpenSource, so we can do more than just write our own cookbooks. We can make changes to the core itself.

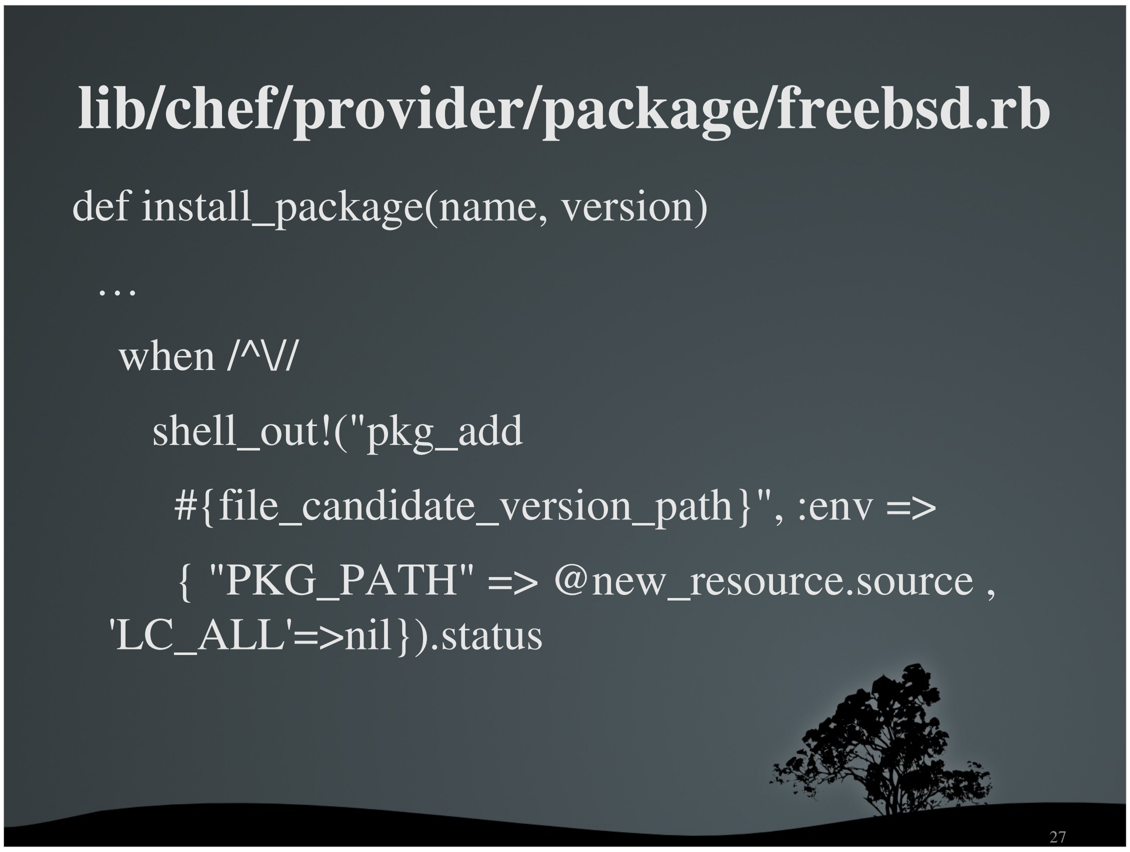
In this case, we offer an alternative to installing a package via ports.

lib/chef/provider/package/freebsd.rb

```
def file_candidate_version
    file_candidate_version_path.split(/-/).last.split(/.tbz/).first
end
```

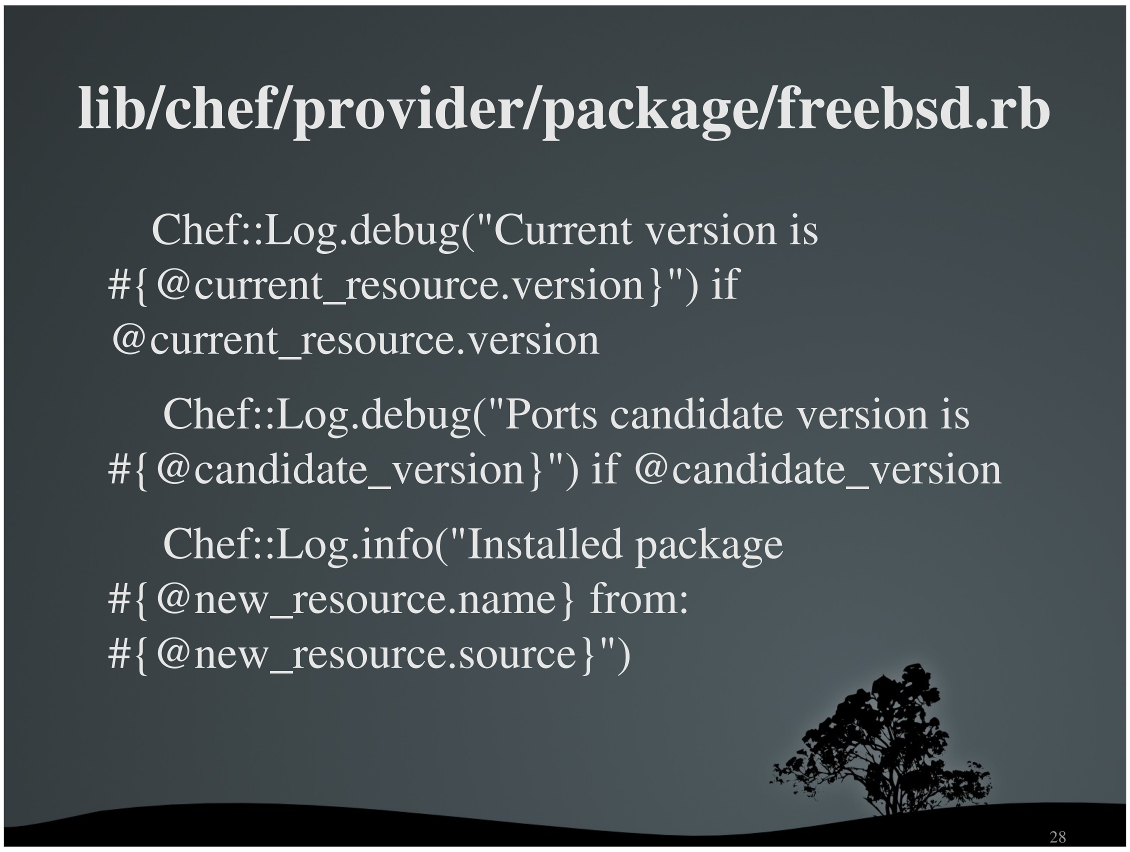
lib/chef/provider/package/freebsd.rb

```
def file_candidate_version_path
    Dir["#{@new_resource.source}/
    #{@current_resource.package_name}*"][0].to_s
end
```



lib/chef/provider/package/freebsd.rb

```
def install_package(name, version)
  …
    when /^\//
      shell_out!("pkg_add
        #{file_candidate_version_path}", :env =>
        { "PKG_PATH" => @new_resource.source ,
  'LC_ALL'=>nil}).status
```

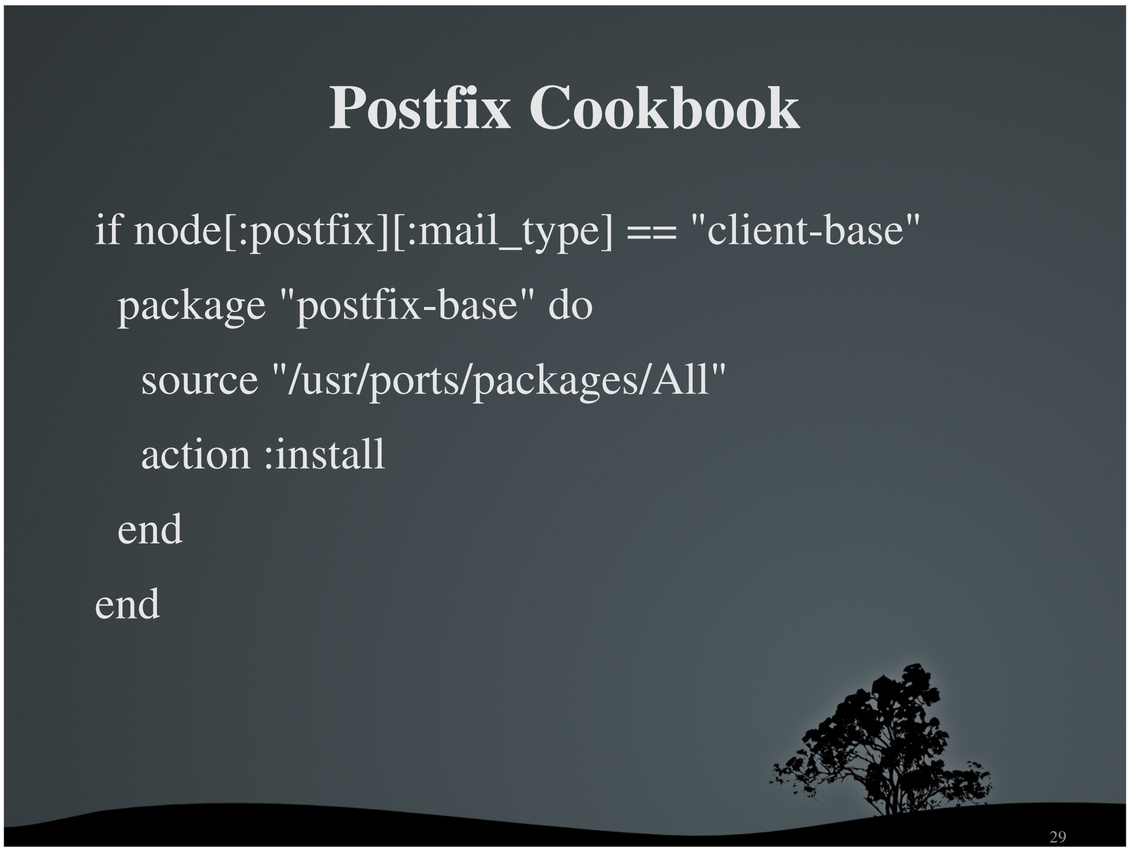


lib/chef/provider/package/freebsd.rb

```
    Chef::Log.debug("Current version is
#{@current_resource.version}") if
@current_resource.version
    Chef::Log.debug("Ports candidate version is
#{@candidate_version}") if @candidate_version
    Chef::Log.info("Installed package
#{@new_resource.name} from:
#{@new_resource.source}")
```

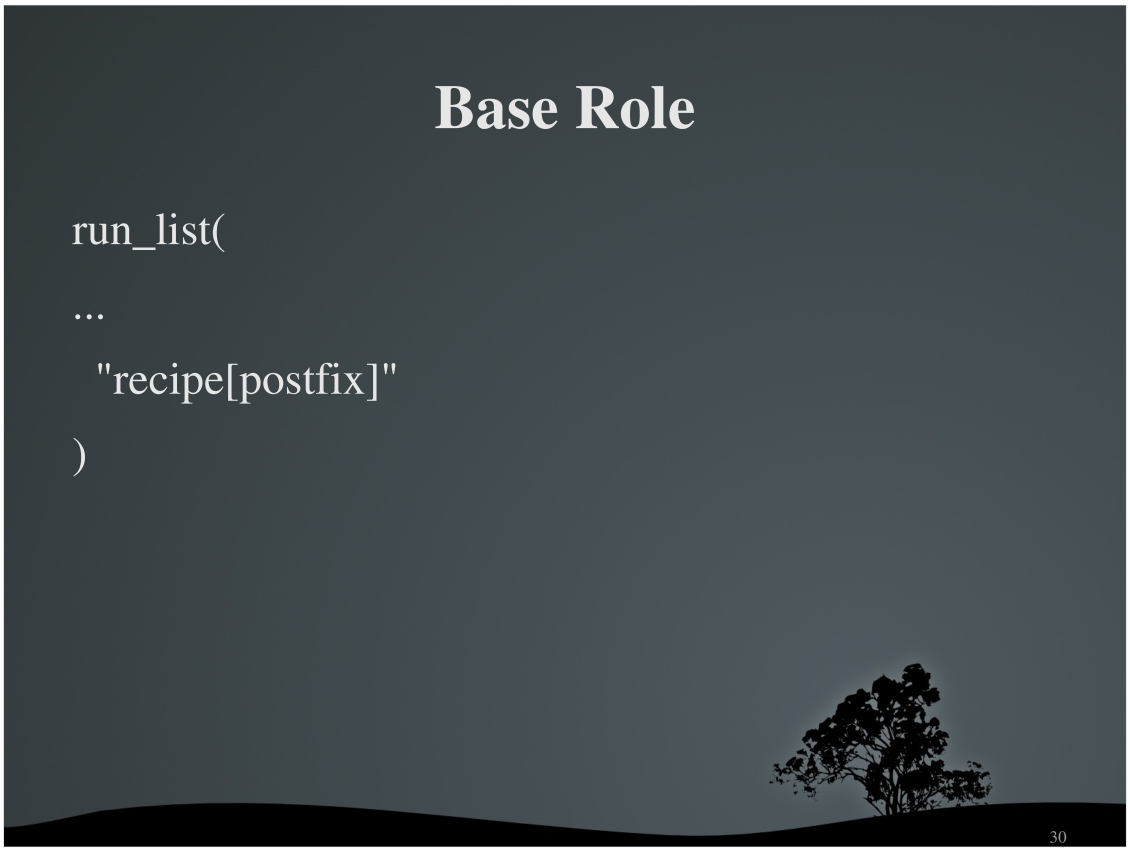
28

and, of course, logging all over the place



Postfix Cookbook

```
if node[:postfix][:mail_type] == "client-base"
  package "postfix-base" do
    source "/usr/ports/packages/All"
    action :install
  end
end
```



Base Role

```
run_list(
...
  "recipe[postfix]"
)
```

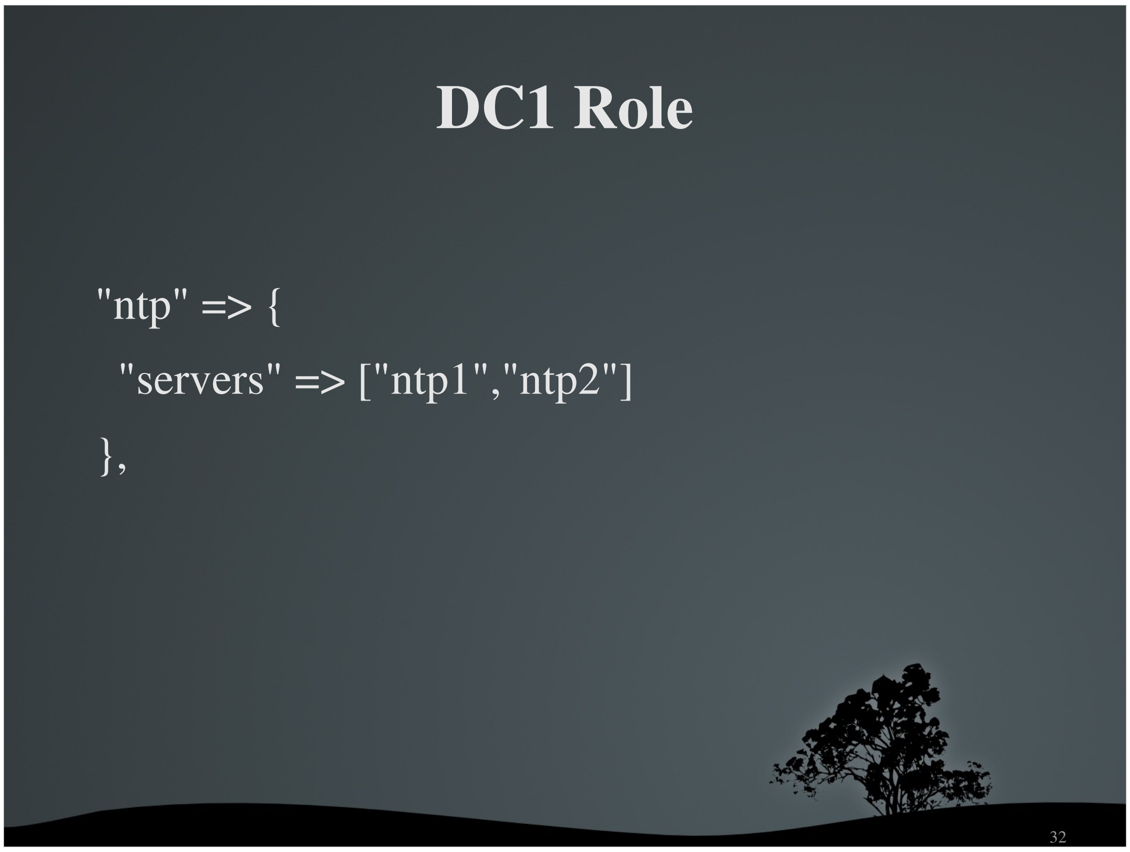
30

Now, that we have postfix installed, we can go back to our other roles and see what they do.

DC1 Role

```
override_attributes(

  "resolver" => {
    "nameservers" => ["x", "y"],
    "search" => "DC1"
  },
```



DC1 Role

```
"ntp" => {
  "servers" => ["ntp1","ntp2"]
},
```

DC1 Role

```
"postfix" => {
  "relayhost" => "relay1"
},
```

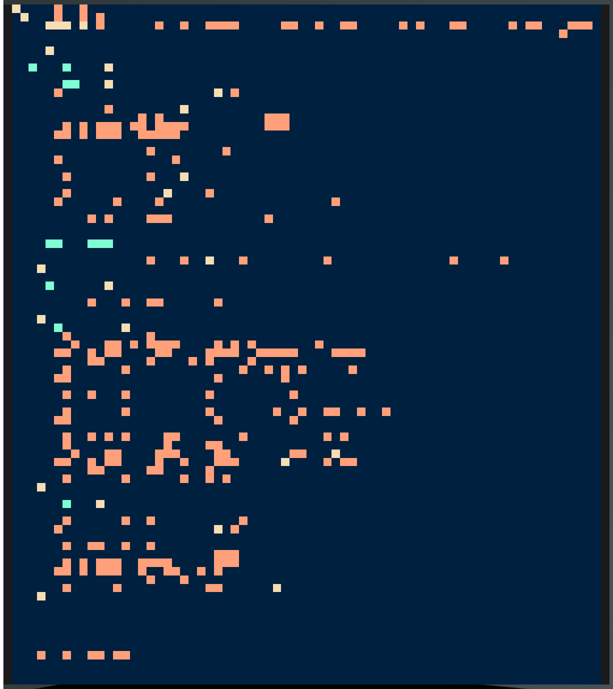
DC1 Role

```
  "system" => {
    "syslog" => { "*.*" => "@syslog1" }
  }
)
```

DC1 Role

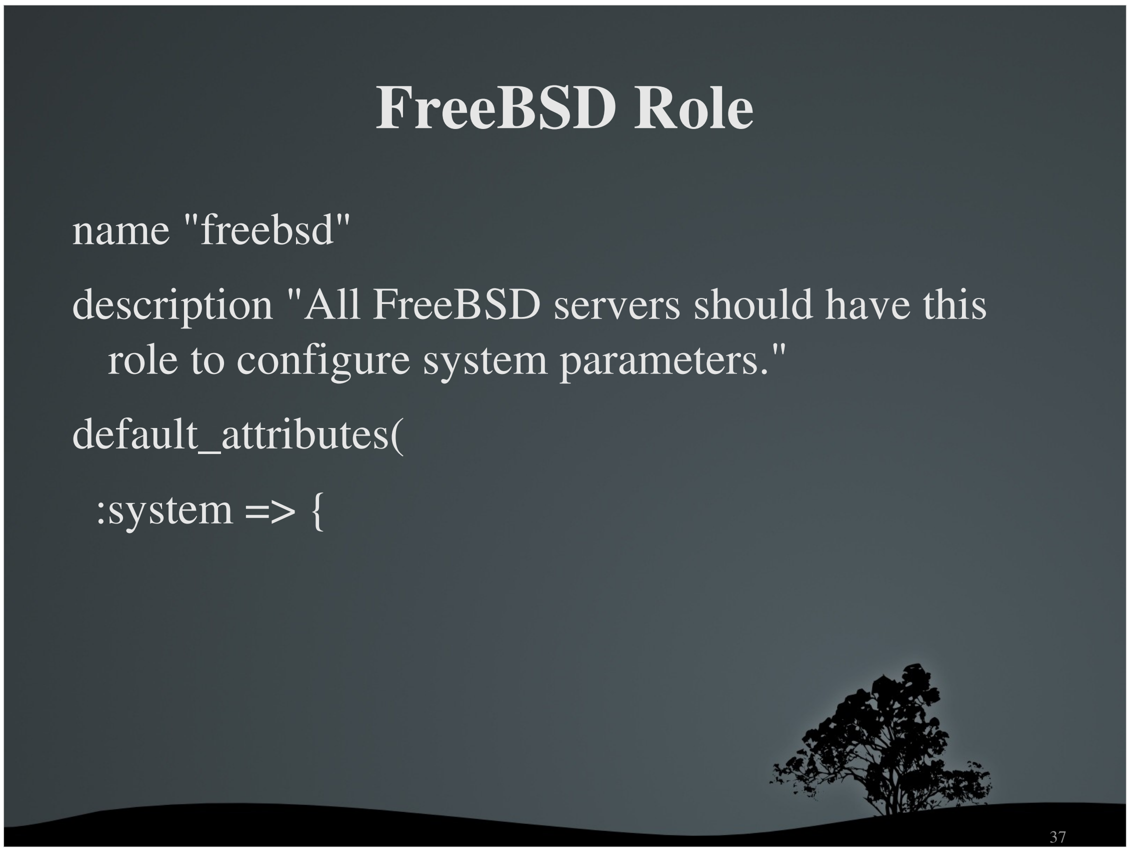
```
run_list(
  "role[base]"
)
```

FreeBSD Role



What is that?
It's pretty, but you can't read it.
I just like the way my editor highlights it. :)

Sometimes you need to have some sort of alterior motive for getting things done. Me, I like nicely-formatted code. It just looks nice, and that motivates me.



FreeBSD Role

```
name "freebsd"
description "All FreeBSD servers should have this
   role to configure system parameters."
default_attributes(
  :system => {
```

FreeBSD Role

```
:boot => {
"kern.ipc.somaxconn" => "1024",
"kern.maxfiles" => "32768",
"kern.ipc.nmbclusters" => "65536",
"kern.ipc.semmni" => "256",
"kern.ipc.semmns" => "512",
"kern.ipc.semmnu" => "256",
"boot_multicons" => "YES",
"boot_serial" => "YES",
"console" => "comconsole,vidconsole",
"comconsole_speed" => "115200"
},
```

FreeBSD Role

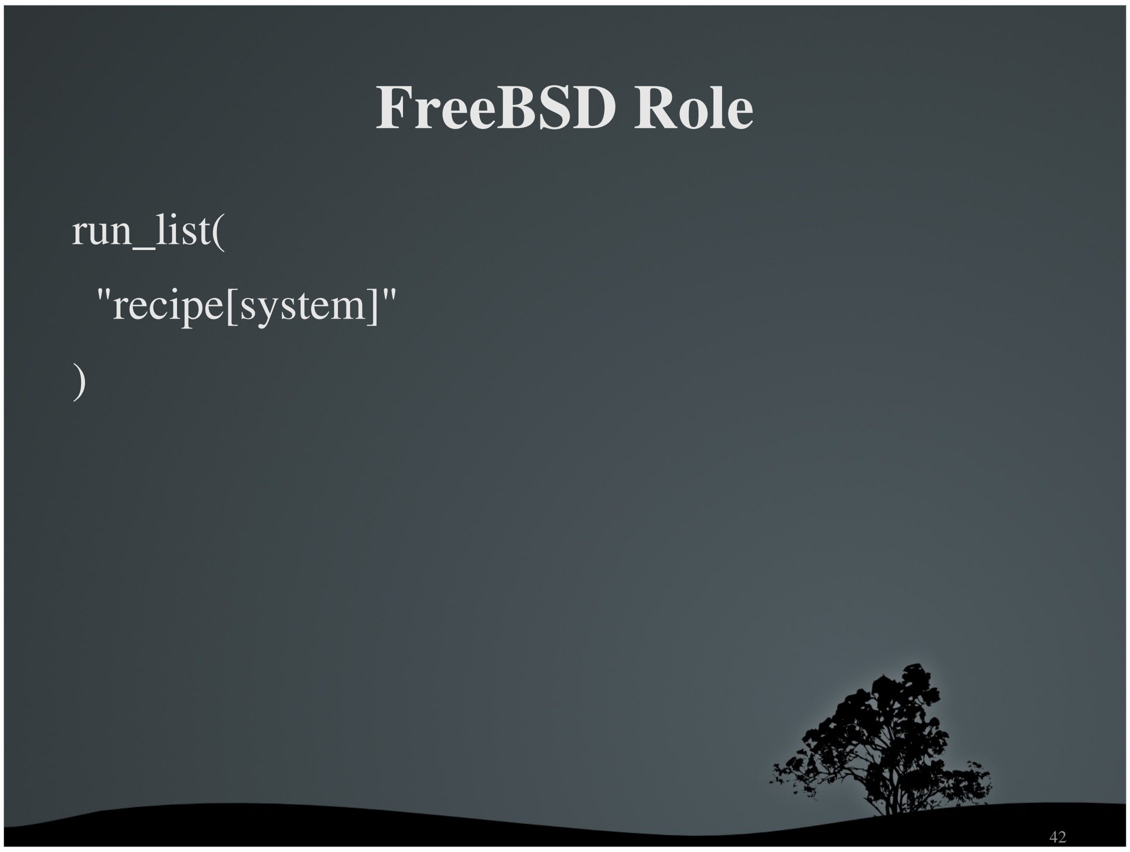
```
:logrotate => {
  "/var/log/snmpd.log" => "644  3     100  * JW
/var/run/snmpd.pid"
},
:make => {
  "INSTALL_NODEBUG" => "yes"
},
```

FreeBSD Role

```
:periodic => {
"daily_clean_hoststat_enable" => "NO",
"daily_status_mail_rejects_enable" => "NO",
"daily_status_include_submit_mailq" => "NO",
"daily_submit_queuerun" => "NO",
"daily_clean_tmps_enable" => "YES",
"daily_clean_tmps_dirs" => "/tmp /var/tmp /usr/tmp",
"daily_clean_tmps_days" => "7",
"daily_status_disks_df_flags" => "-h -t ufs",
"daily_status_zfs_enable" => "YES",
"daily_status_gmirror_enable" => "YES",
"daily_status_ntpd_enable" => "YES"
},
```

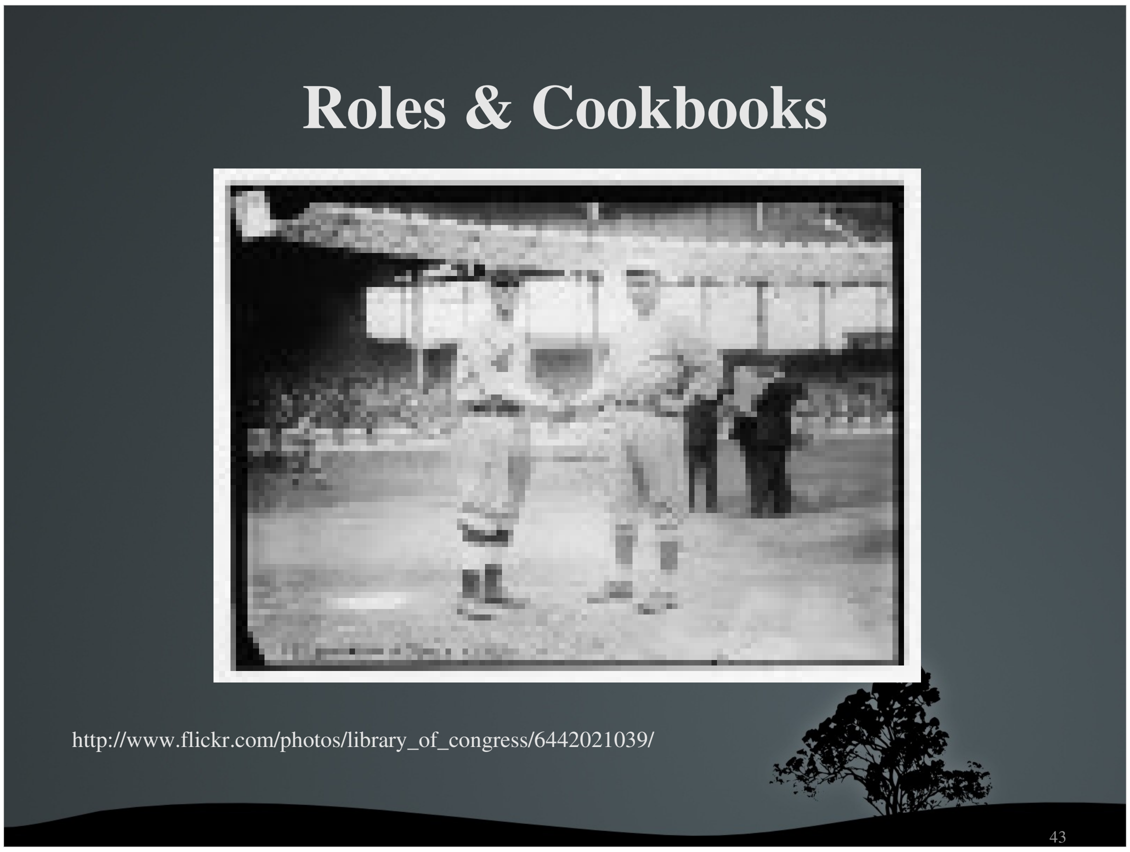
FreeBSD Role

```
:sysctl => {
 "net.inet6.ip6.auto_flowlabel" => "0",
 "kern.ipc.somaxconn" => "1024",
 "machdep.panic_on_nmi" => "0",
 "kern.ipc.semmap" => "256",
 "kern.ipc.shm_use_phys" => "1",
 "security.bsd.see_other_uids" => "0"
 }
}
)
```



FreeBSD Role

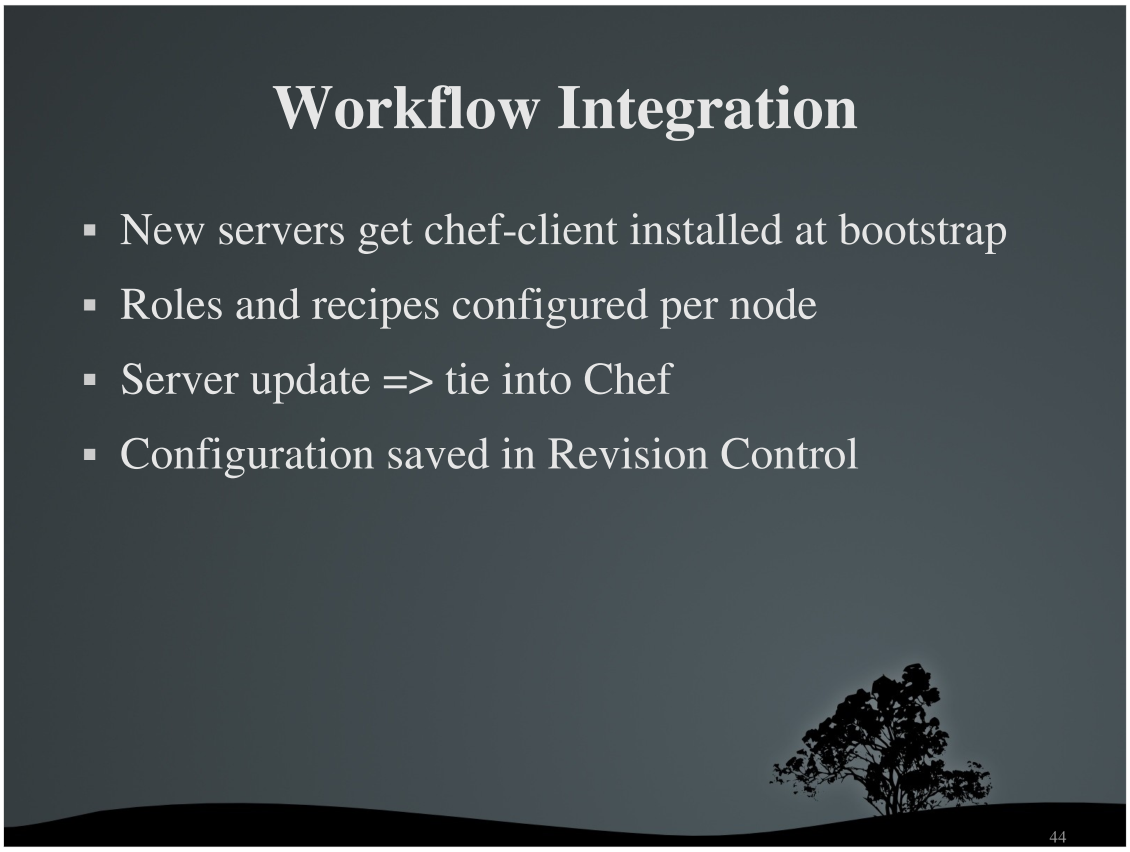
```
run_list(
  "recipe[system]"
)
```



Roles & Cookbooks

http://www.flickr.com/photos/library_of_congress/6442021039/

So remember why we created our roles?
It was so that we could place the values required by our cookbooks in a central location.
They go hand-in-hand to describe the state we'd like our systems to be in.



Workflow Integration

- New servers get chef-client installed at bootstrap
- Roles and recipes configured per node
- Server update => tie into Chef
- Configuration saved in Revision Control

44

So, now that we have a working system, we need to integrate it into our current workflow.

So...

Recap...

45

We started with a whole bunch of different systems, managed by hand by multiple admins. (Great Scott!)

Then we were able to use Chef, cooked up a few recipes, codified our sysadmin practises, and adapted where needed.

Such that we could stop worrying and love the Chef.

Any Questions?

d.n.a@acm.org

Thanks to Opscode for Chef and to Sydney Padua

for the Brunel images (http://2dgoggles.com)