WEBINAR

BUILD A CUSTOM OHAI PLUGIN

FRANKLIN WEBBER
TRAINING AND TECHNICAL CONTENT LEAD

CHEF™

Franklin Webber

Trainer by Day, Gnome by Night

CHEF™

DAY 1 // MAY 22

- ★ Workshops & Chef Training
- ★ DevOps Leadership Summit
- ★ Community Summit
- ★ Partner Summit
- ★ Welcome Reception
- ★ Customer Dinner
- ★ Analyst Day

DAY 2 // MAY 23

- ★ Keynotes
- ★ Technical Sessions
- ★ Happy Hour
- ★ Game Night
- ★ Executive Dinner

DAY 3 // MAY 24

- ★ Keynotes
- ★ Technical Sessions
- ★ Awesome Chef Awards
- ★ Community Celebration

DAY 4 // MAY 25

- ★ Hackday

• Exhibit Hall Open & Sales suites available • chefconf.chef.io •

Who I Think You Are

Chef Cookbook Author

- Build and maintain Chef Cookbooks
- Build and execute ChefSpec Tests
- Interest in learning how to create and test an Ohai plugin

chef.io/webinars/?commid=191263

chef.io/webinars/?commid=194483

OR

Training

training.chef.io

- Chef Essentials
- Test-Driven Cookbook Development

CHEF

What You Will Learn

The Focus

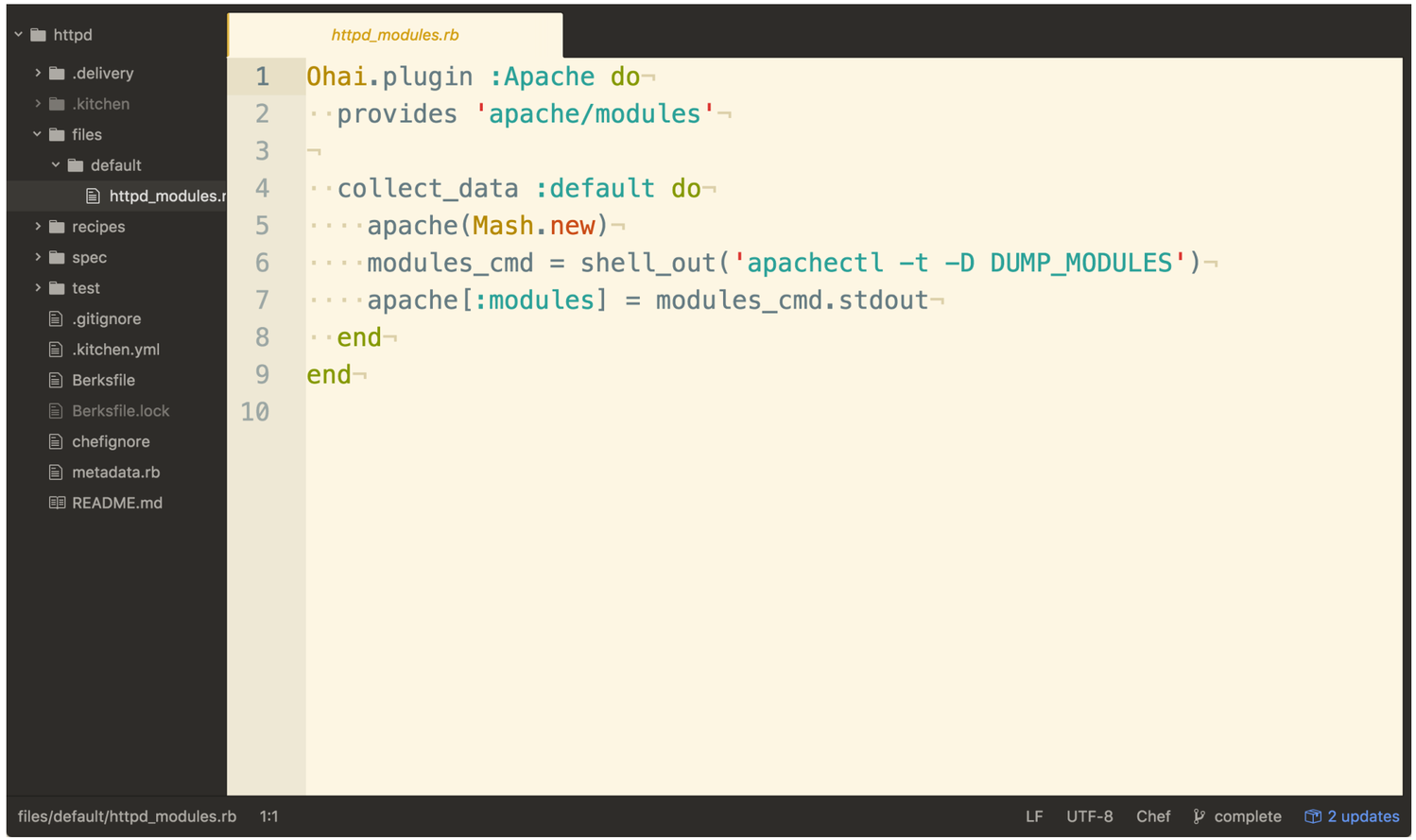
- **Define a test to capture our plugin requirements**
- Build the ohai plugin
- Build a recipe to deliver the plugin

CHEF

What You Will Learn

The Focus

- Define a test to capture our plugin requirements
- **Build the ohai plugin**
- Build a recipe to deliver the plugin



CHEF

What You Will Learn

The Focus

- Define a test to capture our plugin requirements
- Build the ohai plugin
- **Build a recipe to deliver the plugin**

CHEF

Schedule

- Introduction (5 minutes)
- Build and Test Ohai Plugin (40 minutes)
- Questions (12 minutes)
- Resources (3 minutes)

Hack Time

- Join the Chef Community Slack #webinar-ohai

CHEF

Example Code Repository

```
> git clone https://github.com/chef-training/webinar-ohai-repo.git
```

```
Cloning into 'webinar-ohai-repo'...
remote: Counting objects: 67, done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 67 (delta 12), reused 67 (delta 12), pack-reused 0
Unpacking objects: 100% (67/67), done.
```

Let Us Have a Discussion

"Ask a Question" and I will answer it.

"Rate this" presentation to leave your feedback and help me do my work better.

To share, click on the appropriate professional/social network in **"Details"**

CHEF

WEBINAR

BUILD A CUSTOM OHAI PLUGIN

FRANKLIN WEBBER
TRAINING AND TECHNICAL CONTENT LEAD

CHEF™

Agenda

CONCEPTS
- ❑ *Oh hai* Ohai!
- ❑ Meet and Greet with Ohai's Plugins

TECHNICAL
- ❑ Build the Ohai plugin
- ❑ Build a Recipe to deliver the plugin

TIPS
- ❑ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF

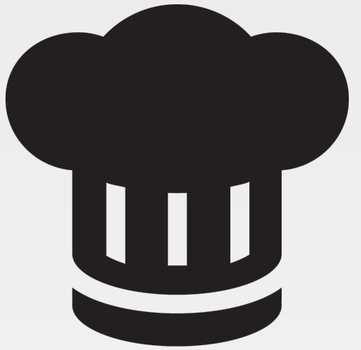
CONCEPT

Ohai

Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run. The types of attributes Ohai collects include (but are not limited to):

- Platform details
- Network usage
- Memory usage
- CPU data

EXERCISE



Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ❑ Execute Ohai to retrieve details about the node
- ❑ View Ohai's execution within a chef-client run
- ❑ Describe attributes precedence

CHEF™

CONCEPT

All About The System

Ohai queries the operating system with a number of commands.

The data is presented in JSON (JavaScript Object Notation).

CHEF™

Running Ohai to Show All Attributes

```
> ohai
```


```
{
  "kernel": {
    "name": "Linux",
    "release": "2.6.32-431.1.2.0.1.el6.x86_64",
    "version": "#1 SMP Fri Dec 13 13:06:13 UTC 2013",
    "machine": "x86_64",
    "os": "GNU/Linux",
    "modules": {
      "veth": {
        "size": "5040",
        "refcount": "0"
      },
      "ipt_addrtype": {
```


CHEF

Running Ohai to Show the IP Address

```
> ohai ipaddress
```

```
[
  "172.31.57.153"
]
```

Running Ohai to Show the Hostname

```
> ohai hostname
```

```
[
  "ip-172-31-57-153"
]
```

Running Ohai to Show the Memory

```
> ohai memory
```

```
{
  "swap": {
    "cached": "0kB",
    "total": "0kB",
    "free": "0kB"
  },
  "total": "604308kB",
  "free": "297940kB",
  "buffers": "24824kB",
  "cached": "198264kB",
```


CHEF

Running Ohai to Show the Total Memory

```
> ohai memory/total
```

```
[
  "604308kB"
]
```

Running Ohai to Show the CPU

```
> ohai cpu
```

```
{
  "0": {
    "vendor_id": "GenuineIntel",
    "family": "6",
    "model": "45",
    "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",
    "stepping": "7",
    "mhz": "1795.673",
    "cache_size": "20480 KB",
    "physical_id": "34
```

Running Ohai to Show the First CPU

```
> ohai cpu/0
```

```
{
  "vendor_id": "GenuineIntel",
  "family": "6",
  "model": "45",
  "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",
  "stepping": "7",
  "mhz": "1795.673",
  "cache_size": "20480 KB",
  "physical_id": "34",
  "core_id": "0",
  "cores": "1"
```


CHEF

Running Ohai to Show the First CPU Mhz

```
> ohai cpu/0/mhz
```

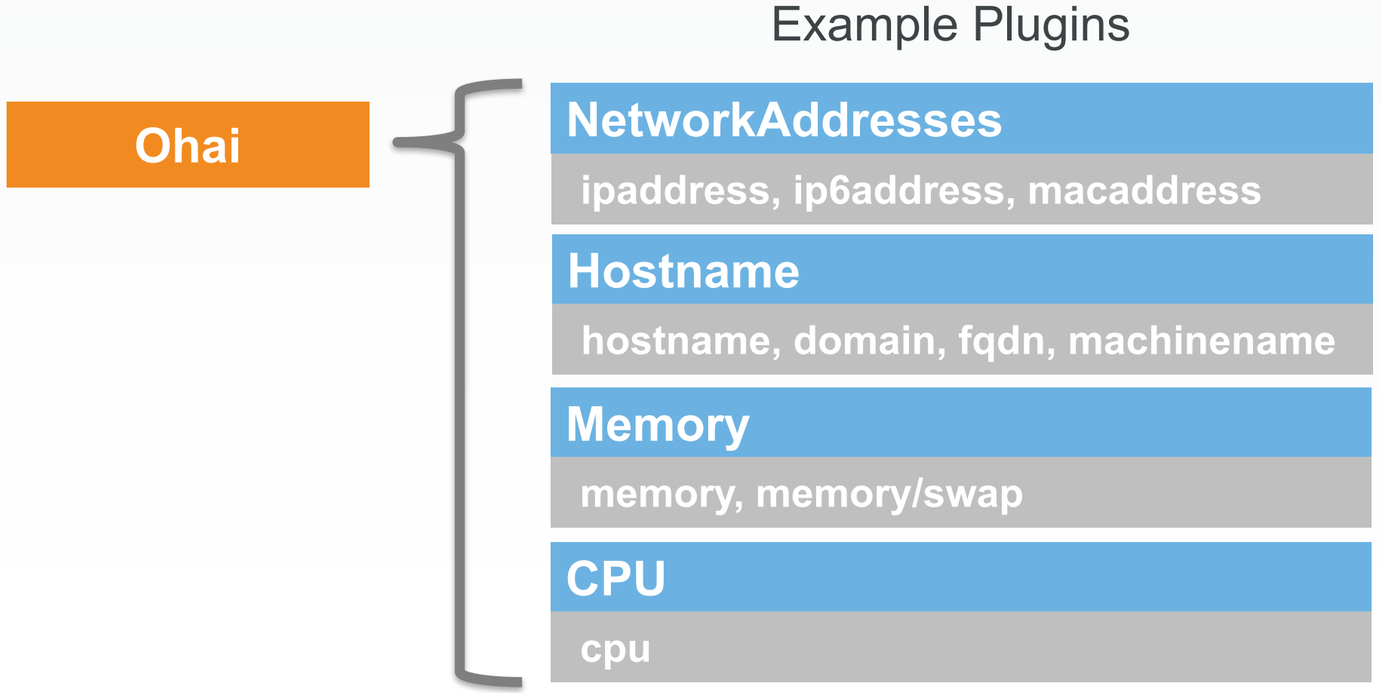
```
[
  "1795.673"
]
```

CONCEPT

Ohai is Composed of Plugins

Ohai is packaged with a core set of plugins that are automatically loaded when executing Ohai.

These plugins provide the attributes we see in the JSON output (e.g. ipaddress, hostname, memory, cpu).



CHEF

CONCEPT

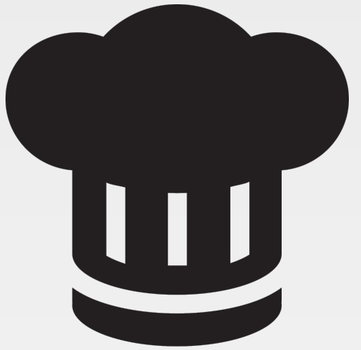
Custom Ohai Plugins

It is possible to define your own plugins and have Ohai load those plugins.

```
> ohai -d PATH_TO_CUSTOM_PLUGINS
```

We will explore creating an Ohai plugin and loading it with Ohai from the command-line in the Build an Ohai Plugin' module.

EXERCISE



Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ✓ Execute Ohai to retrieve details about the node
- ❑ View Ohai's execution within a chef-client run
- ❑ Describe attributes precedence

CONCEPT

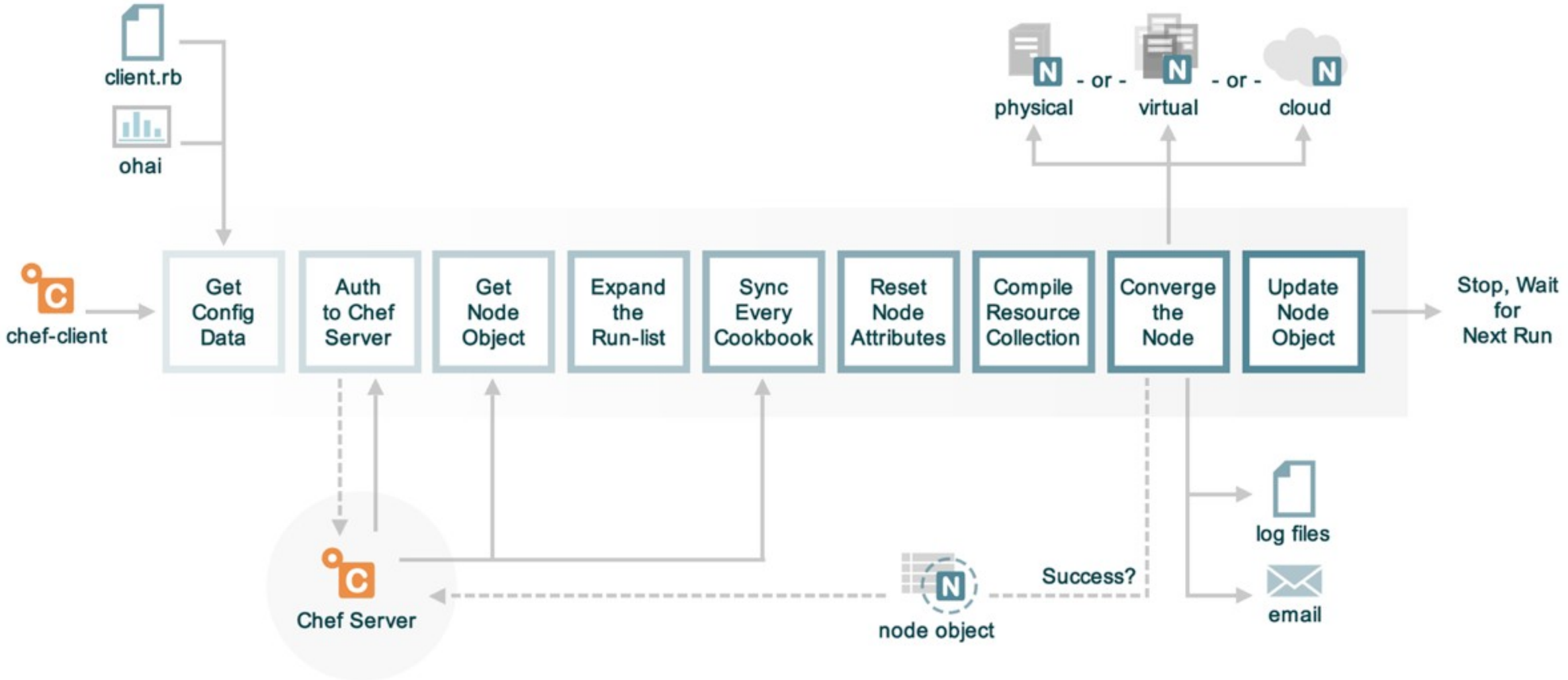
chef-client

chef-client automatically executes ohai and stores the data about the node in an object we can use within the recipes. When the chef-client run completes successfully the details about the node are sent to the Chef Server.

http://docs.chef.io/ohai.html

CHEF

The Anatomy of a chef-client Run



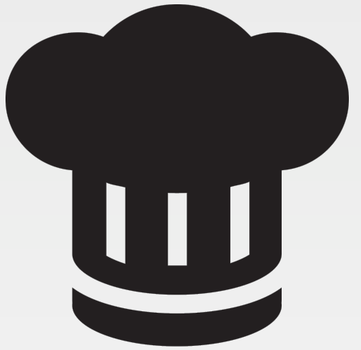
CHEF

Running Ohai in Code

```
> chef exec pry
```

```
[1] pry(main)> require 'ohai'
=> true
[2] pry(main)> ohai = Ohai::System.new
=> #<Ohai::System:0x007fc62fadc490 @plugin_pat...@safe_run=true>>
[3] pry(main)> ohai.all_plugins('ipaddress')
[4] pry(main)> ohai.all_plugins('hostname')
[5] pry(main)> ohai.all_plugins('memory')
[6] pry(main)> ohai.all_plugins('cpu')
[7] pry(main)> ohai.all_plugins
[8] pry(main)> exit
```

EXERCISE



Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ✓ Execute Ohai to retrieve details about the node
- ✓ View Ohai's execution within a chef-client run
- ❑ Describe attributes precedence

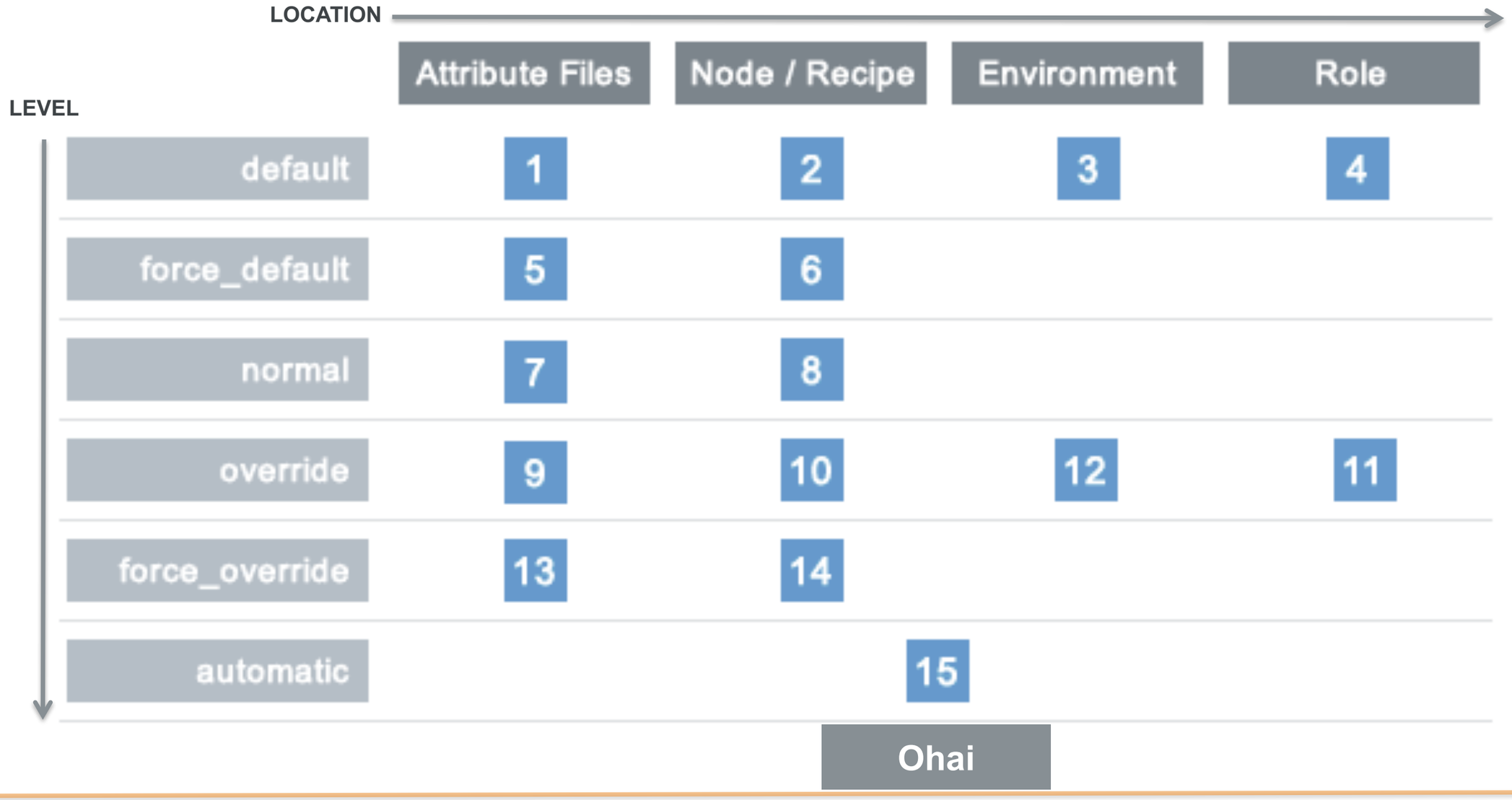
CONCEPT

Node Attributes

A node object maintains the attributes collected from Ohai, from the previous node object (as returned by the Chef Server), from the environments, roles, and the cookbooks defined in the run list.

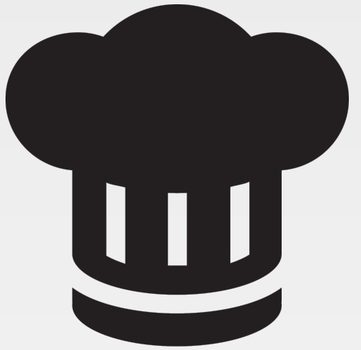
CHEF

Viewing Attribute Precedence as a Table



CHEF

EXERCISE



Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ✓ Execute Ohai to retrieve details about the node
- ✓ View Ohai's execution within a chef-client run
- ✓ Describe attributes precedence

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ❑ Meet and Greet with Ohai's Plugins

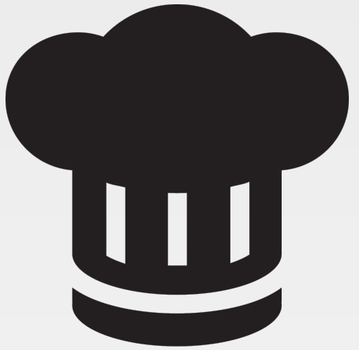
TECHNICAL
- ❑ Build the Ohai plugin
- ❑ Build a Recipe to deliver the plugin

TIPS
- ❑ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF

EXERCISE



Reviewing the Ohai Gem

Ohai is a Rubygem. First we need to learn about how a gem is structured.

Objective:

- ❑ Review the basic structure of the Ohai gem
- ❑ Review the 'language' plugin
- ❑ Review the 'python' plugin

CONCEPT

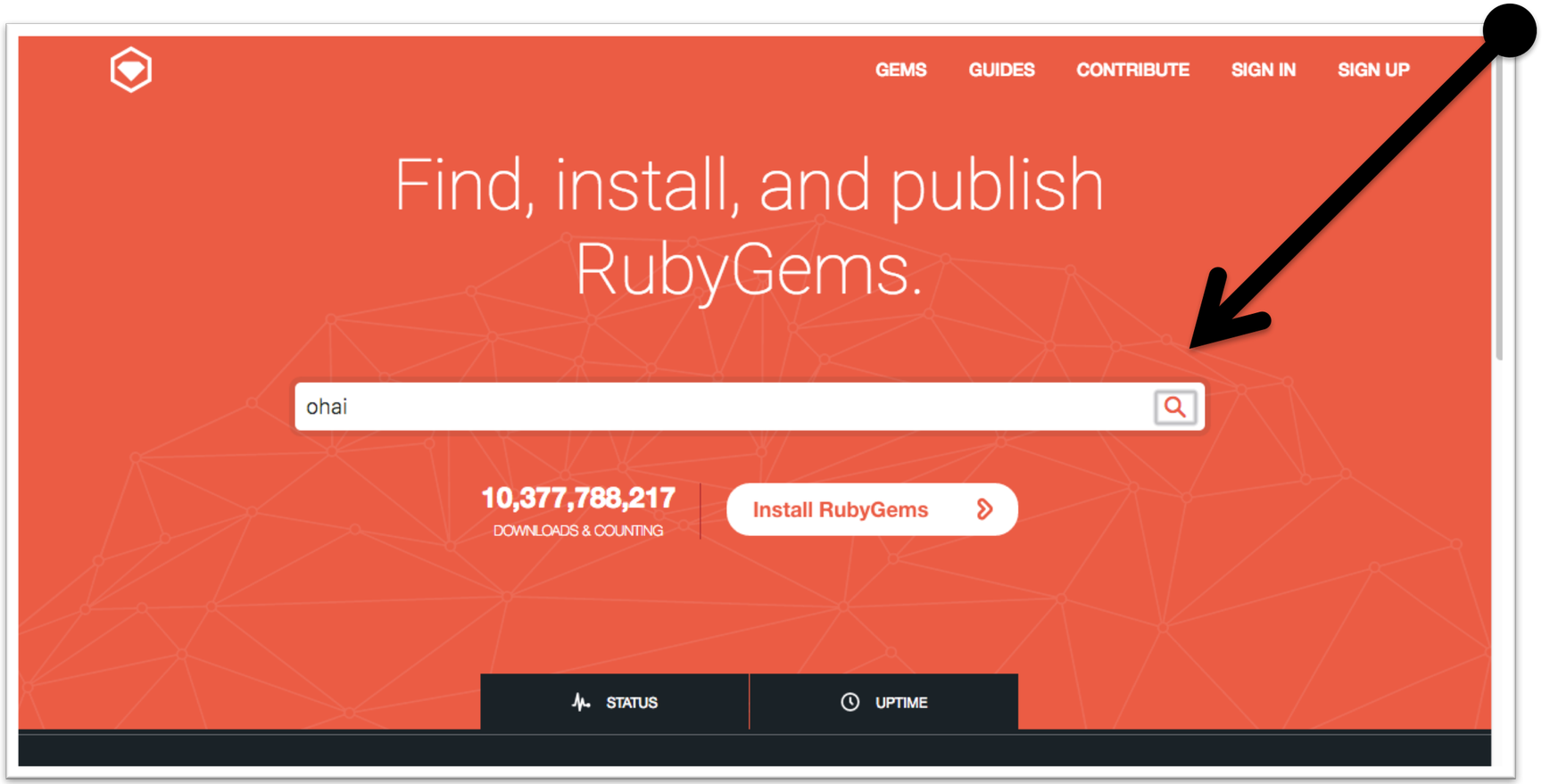
Ohai is Ruby Gem

Ruby gems are the ways in which Ruby developers share the code that they develop with others. A Ruby gem is really a packaging structure similar to that of a Chef cookbook.

Searching for a Gem on Rubygems

Steps

1. Visit https://rubygems.org
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.

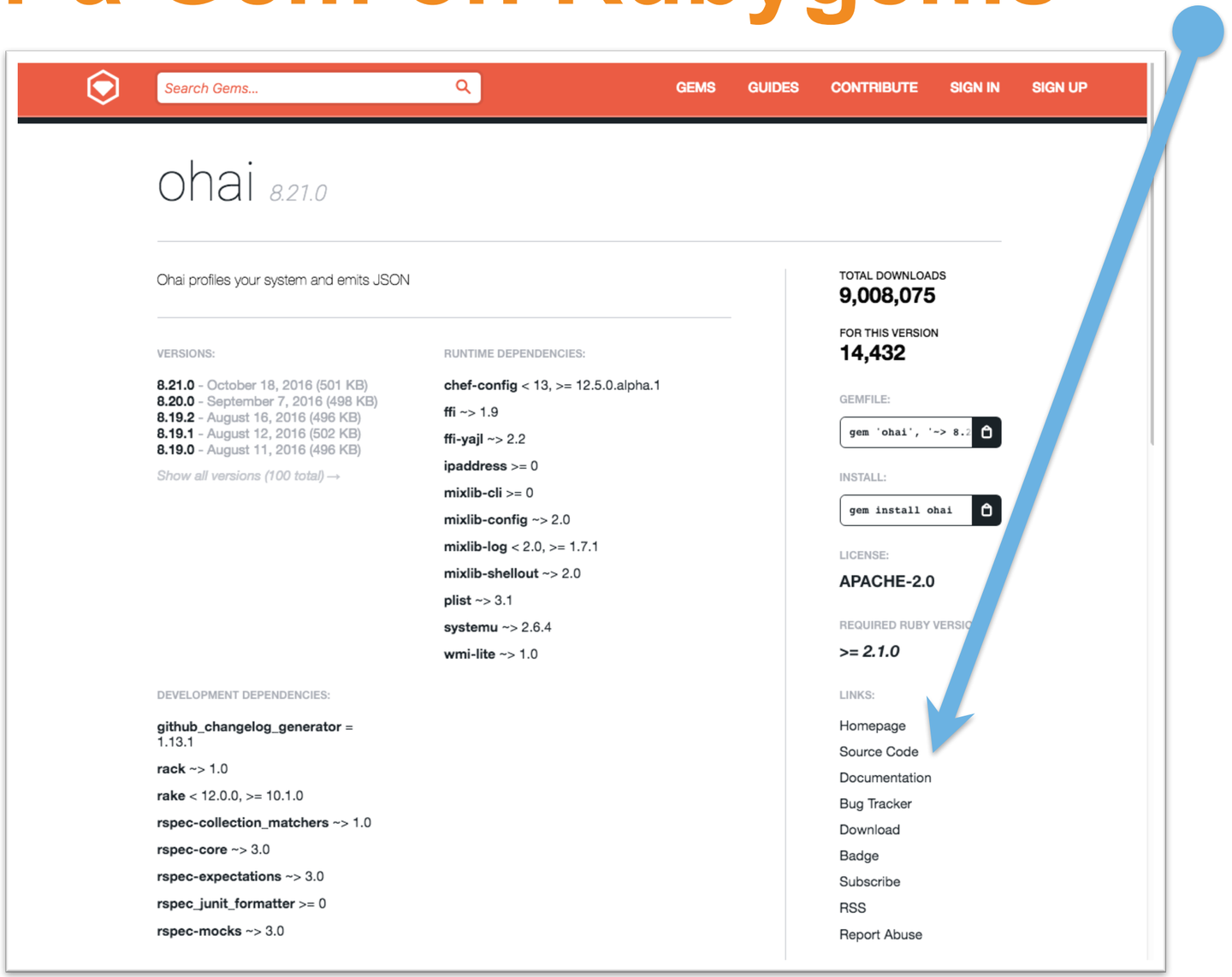


CHEF

Searching for a Gem on Rubygems

Steps

1. Visit https://rubygems.org
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.

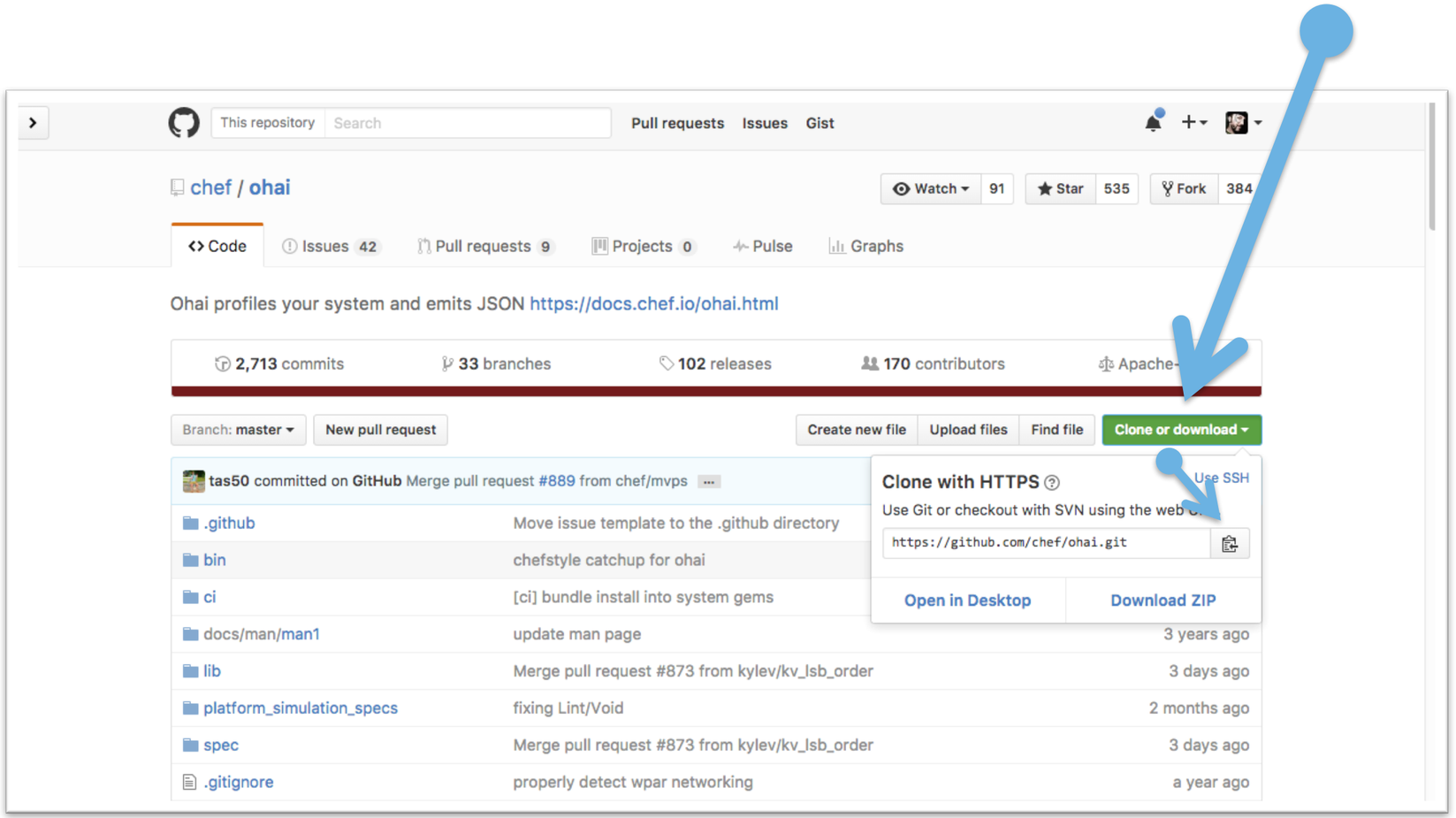


CHEF

Searching for a Gem on Rubygems

Steps

1. Visit https://rubygems.org
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.



CHEF

Returning to the Home Directory

```
> cd ~
```

Cloning the Ohai Library

```
> git clone https://github.com/chef/ohai.git
```

```
Cloning into 'ohai'...
remote: Counting objects: 20571, done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 20571 (delta 7), reused 0 (delta 0), pack-reused 20540
Receiving objects: 100% (20571/20571), 4.46 MiB | 2.13 MiB/s, done.
Resolving deltas: 100% (13408/13408), done.
Checking connectivity... done.
```

Viewing the Contents of the Project

```
> tree ohai
```

```
ohai
├── CHANGELOG.md
├── DOC_CHANGES.md
├── Gemfile
├── LICENSE
├── NOTICE
├── OHAI_MVPS.md
├── README.md
├── RELEASE_NOTES.md
├── Rakefile
├── appveyor.yml
├── bin
```

Viewing the README

~/ohai/README.md

```
# ohai

... PROJECT BADGES ...

## Description

Ohai detects data about your operating system. It can be used standalone, but its primary
purpose is to provide node data to Chef.

Ohai will print out a JSON data blob for all the known data about your system. When used with
Chef, that data is reported back via node attributes.

Chef distributes ohai as a RubyGem. This README is for developers who want to modify the Ohai
source code. For users who want to write plugins for Ohai, see the docs:

- General documentation: <https://docs.chef.io/ohai.html>
- Custom plugin documentation: <https://docs.chef.io/ohai_custom.html>
```

Viewing the Gem Specification

~/ohai/ohai.gemspec

```
$:.unshift File.expand_path("../lib", __FILE__)
require "ohai/version"

Gem::Specification.new do |s|
  s.name = "ohai"
  s.version = Ohai::VERSION
  s.platform = Gem::Platform::RUBY
  s.summary = "Ohai profiles your system and emits JSON"
  s.description = s.summary
  s.license = "Apache-2.0"
  s.author = "Adam Jacob"
  s.email = "adam@chef.io"
  s.homepage = "https://docs.chef.io/ohai.html"

  s.required_ruby_version = ">= 2.1.0"

  s.add_dependency "systemu", "~> 2.6.4"
```

Viewing the lib Directory

```
> tree ohai/lib
```

```
ohai/lib
├── ohai
│   ├── application.rb
│   ├── common
│   │   └── dmi.rb
│   ├── config.rb
...
│   └── version.rb
└── ohai.rb
19 directories, 161 files
```


CHEF

Viewing the ohai.rb file in the lib Directory

~/ohai/lib/ohai.rb

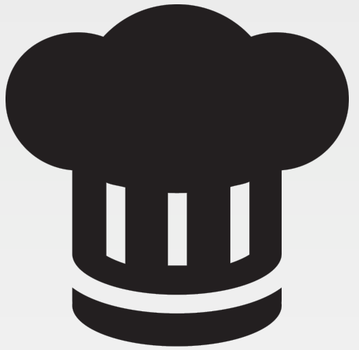
```
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

require "ohai/version"
require "ohai/config"
require "ohai/system"
require "ohai/exception"
```

Viewing the plugins directory

```
> tree ohai/lib/ohai/plugins
```

```
ohai/lib/ohai/plugins
├── aix
│   ├── cpu.rb
│   ├── filesystem.rb
│   ├── kernel.rb
│   ├── memory.rb
│   ├── network.rb
│   ├── os.rb
│   ├── platform.rb
│   ├── uptime.rb
│   └── virtualization.rb
├── azure.rb
├── bsd
│   ├── filesystem.rb
```

EXERCISE



Reviewing the Ohai Gem

Let's take a look at the structure of a plugin.

Objective:

- ✓ Review the basic structure of the Ohai gem
- ❑ Review the 'language' plugin
- ❑ Review the 'python' plugin

CONCEPT

Recent Major Ohai Releases

Ohai 6

Released: April 13, 2011

Chef Version: 0.10.0

docs.chef.io/release/ohai-6

Ohai 7

Released: April 8, 2014

Chef Version: 11.12.0

docs.chef.io/release/ohai-7

Ohai 8

Released: Dec. 4, 2014

Chef Version: 11.18.0

docs.chef.io/release/ohai-8

CHEF

CONCEPT

Focus on Ohai 7

Ohai 7 refined the Domain Specific Language (DSL) created in the previous version of Ohai. Ohai 8 continues to use the same language.

Ohai 6 Ohai 7 Ohai 8

Viewing the Languages Plugin

~/ohai/lib/ohai/plugins/languages.rb

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

Viewing the Languages Plugin

~/ohai/lib/ohai/plugins/languages.rb

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

Plugin Name
- Ruby Symbol
- First Letter Capitalized

Node attributes provided by the plugin

Code executed on all platforms and stored in the provided attribute(s).

CHEF

Viewing the Language Plugin

~/ohai/lib/ohai/plugins/languages.rb

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

The Languages plugin provides the node attribute 'languages' which is populated, on all platforms, with a Mash.

CONCEPT

Hash

Using a String as a key

```
[1] pry(main)> content = Hash.new
=> {}
[2] pry(main)> content['name'] =
'Chef'
=> "Chef"
[3] pry(main)> content['name']
=> "Chef"
[4] pry(main)> content[:name]
=> nil
```

Using a Symbol as a key

```
[1] pry(main)> content = {}
=> {}
[2] pry(main)> content[:name] =
'Chef'
=> "Chef"
[3] pry(main)> content[:name]
=> "Chef"
[4] pry(main)> content['name']
=> nil
```


CHEF

CONCEPT

Mash

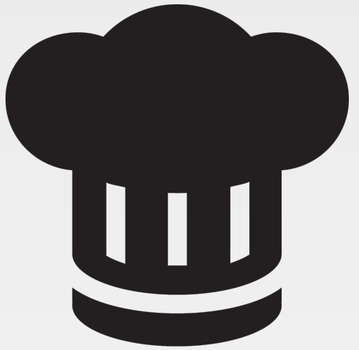
Using a String as a key

```
[1] pry(main)> require 'chef'
=> true
[2] pry(main)> content = Mash.new
=> {}
[3] pry(main)> content['name'] = 'Chef'
=> "Chef"
[4] pry(main)> content['name']
=> "Chef"
[5] pry(main)> content[:name]
=> "Chef"
```

Using a Symbol as a key

```
[1] pry(main)> require 'chef'
=> true
[2] pry(main)> content = Mash.new
=> {}
[3] pry(main)> content[:name] = 'Chef'
=> "Chef"
[4] pry(main)> content[:name]
=> "Chef"
[5] pry(main)> content['name']
=> "Chef"
```

EXERCISE



Reviewing the Ohai Gem

Now it is time to look at a more complex plugin.

Objective:

- ✓ Review the basic structure of the Ohai gem
- ✓ Review the 'language' plugin
- ❑ Review the 'python' plugin

Viewing the Python plugin

~/ohai/lib/ohai/plugins/python.rb

```
Ohai.plugin(:Python) do
  provides "languages/python"

  depends "languages"

  collect_data do
    begin
      so = shell_out("python -c \"import sys; print (sys.version)\"")
      # Sample output:
      # 2.7.11 (default, Dec 26 2015, 17:47:53)
      # [GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)]
      if so.exitstatus == 0
        python = Mash.new
        output = so.stdout.split
        python[:version] = output[0]
        if output.length >= 6
```

Node attributes provided by the plugin

This plugin depends on the attributes in the Languages plugin to be defined

Viewing the Python plugin

~/ohai/lib/ohai/plugins/python.rb

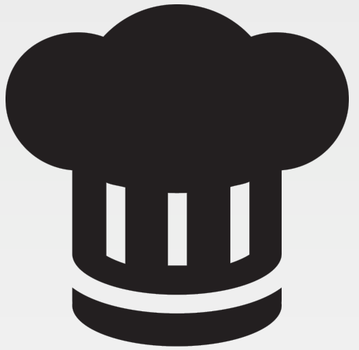
```
Ohai.plugin(:Python) do
  provides "languages/python"

  depends "languages"

  collect_data do
    begin
      so = shell_out("python -c \"import sys; print (sys.version)\"")
      # Sample output:
      # 2.7.11 (default, Dec 26 2015, 17:47:53)
      # [GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)]
      if so.exitstatus == 0
        python = Mash.new
        output = so.stdout.split
        python[:version] = output[0]
        if output.length >= 6
```

CONCEPT

collect_data for a specific Platform

Plugins can collect data in different ways across different platforms. When defining a collect_data block if you do not provide any arguments it is assumed the default and all platforms unless you define a collect_data block specific for a platform.

EXERCISE



Reviewing the Ohai Gem

We know where the plugins are located and what they look like. Now it's time to make one.

Objective:

- ✓ Review the basic structure of the Ohai gem
- ✓ Review the 'language' plugin
- ✓ Review the 'python' plugin

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

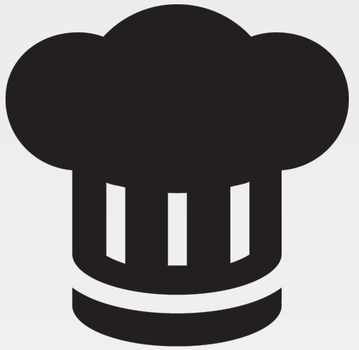
TECHNICAL
- ❑ Build the Ohai plugin
- ❑ Build a Recipe to deliver the plugin

TIPS
- ❑ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ❑ Define expectations for the Ohai plugin
- ❑ Create the Ohai plugin

Installing the chefspec-ohai gem

```
> chef gem install chefspec-ohai
```

```
Successfully installed chefspec-ohai-0.1.0
1 gem installed
```

Adding the Gem to the Spec Helper

~/cookbooks/httpd/spec/spec_helper.rb

```
require 'chefspec'
require 'chefspec/berkshelf'
require 'chefspec/ohai'
```

Creating a Directory for Plugins Tests

```
> mkdir cookbooks/httpd/spec/unit/plugins
```


CHEF

Defining the First Expectation for Plugin

~/cookbooks/httpd/spec/unit/plugins/httpd_modules_spec.rb

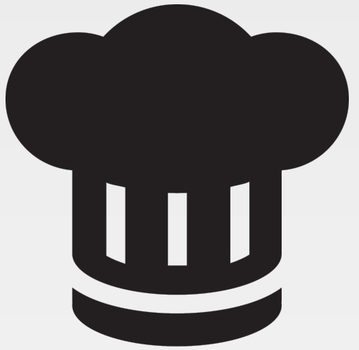
```
require 'spec_helper'

describe_ohai_plugin :Apache do
  let(:plugin_file) { 'files/default/httpd_modules.rb' }

  it 'provides apache/modules' do
    expect(plugin).to provides_attribute('apache/modules')
  end
end
```

Executing the Tests to See Failure

```
> chef exec rspec
```

```
F.........

Failures:

 1) Apache provides 'apache/modules'
     Failure/Error: let(:plugin_source) { File.read(plugin_file) }

     Errno::ENOENT:
       No such file or directory @ rb_sysopen - files/default/httpd_modules.rb
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ❑ Create the Ohai plugin

Creating the Plugin as a Cookbook File

```
> chef generate file httpd_modules.rb
```

```
Recipe: code_generator::cookbook_file
  * directory[/Users/franklinwebber/training/source/extending_cookbooks-
repo/cookbooks/httpd/files/default] action create (up to date)
  * template[/Users/franklinwebber/training/source/extending_cookbooks-
repo/cookbooks/httpd/files/default/httpd_modules.rb] action create
    - create new file
/Users/franklinwebber/training/source/extending_cookbooks-
repo/cookbooks/httpd/files/default/httpd_modules.rb
    - update content in file
/Users/franklinwebber/training/source/extending_cookbooks-
repo/cookbooks/httpd/files/default/
```

Defining the Plugin that Provides Values

~/cookbooks/httpd/files/default/httpd_modules.rb

```
Ohai.plugin :Apache do
  provides 'apache/modules'
end
```

Executing the Tests to See Success

```
> chef exec rspec
```

```
..........

Finished in 5.85 seconds (files took 2.74 seconds to load)
10 examples, 0 failures
```

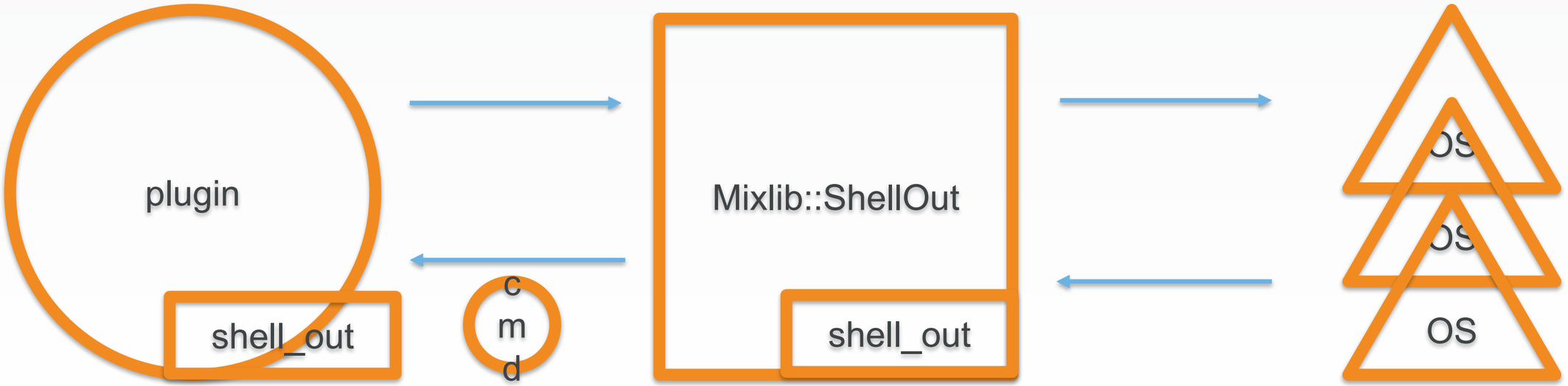
Defining an Expectation for Attribute Value

~/cookbooks/httpd/spec/unit/plugins/httpd_modules_spec.rb

```
require 'spec_helper'

describe_ohai_plugin :Apache do
  let(:plugin_file) { 'files/default/httpd_modules.rb' }

  it 'provides apache/modules' do
    expect(plugin).to provides_attribute('apache/modules')
  end

  it 'correctly captures output' do
    allow(plugin).to receive(:shell_out).with('apachectl -t -D
DUMP_MODULES').and_return(double(stdout: 'OUTPUT'))
    expect(plugin_attribute('apache/modules')).to eq('OUTPUT')
  end
end
```

CONCEPT

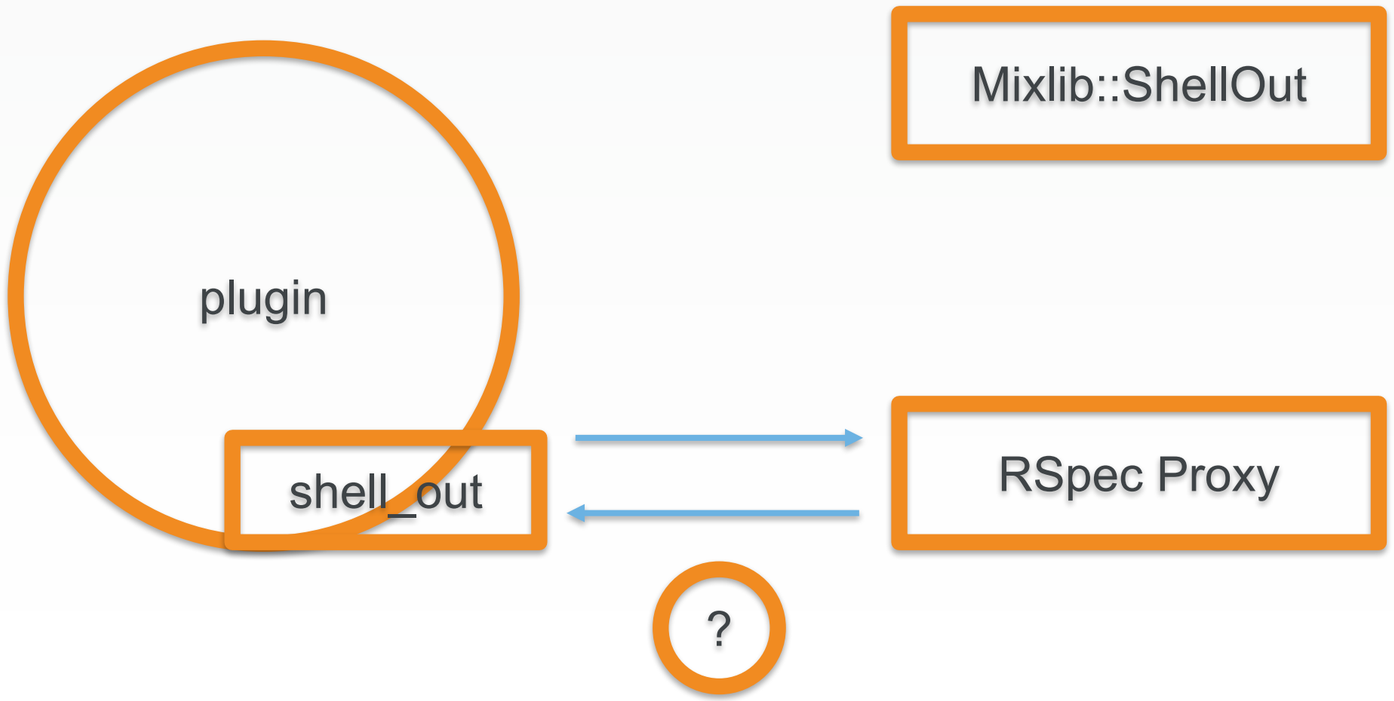
Shelling Out to the Operating System!



CHEF

CONCEPT

Stubbing Out the `shell_out`!



CHEF

CONCEPT

Doubles

Optionally give me a name

code: double("command")

output: failed with a name

CONCEPT

Doubles with a Name

Define methods using an unadorned hash of parameters. Key, values is the method name and the result of the method. You can even respond back with other variables or a block of code. But the goal of doubles is not to be fancy or have to worry about dependencies. The goal of doubles is to build the perfect replacement for that very moment. It has to do nothing else but act in a particular way for a particular moment and then dissapear. This lets you more easily create the various scenarios necessary to simulate all these various states.

code: double("command", name: 'jamie', title: 'Marvelous Human',)
code: show obj.name obj.title

Executing the Tests to See Failure

```
> chef exec rspec
```

```
.F.........

Failures:

  1) Apache correctly captures output
     Failure/Error: expect(plugin_attribute('apache/modules')).to
eq("OUTPUT")

     NoMethodError:
       undefined method `[]' for nil:NilClass
```

Capturing the Apache Modules Content

~/cookbooks/httpd/files/default/httpd_modules.rb

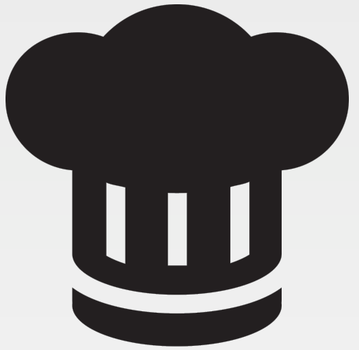
```
Ohai.plugin :Apache do
  provides 'apache/modules'

  collect_data :default do
    apache(Mash.new)
    modules_cmd = shell_out('apachectl -t -D DUMP_MODULES')
    apache[:modules] = modules_cmd.stdout
  end
end
```

Executing the Tests to See Success

```
> chef exec rspec
```

```
.[2016-10-05T17:00:43-05:00] WARN: Plugin Definition Error:
<files/default>: collect_data already defined on platform default

..........

Finished in 5.84 seconds (files took 3.01 seconds to load)
11 examples, 0 failures
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ✓ Create the Ohai plugin

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

TECHNICAL
- ✓ Build the Ohai plugin
- ❑ Build a Recipe to deliver the plugin

TIPS
- ❑ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF

CONCEPT

Plugin is Here in the Cookbook

1. Deliver plugin to a particular directory on our nodes
2. Inform ohai to load the plugin

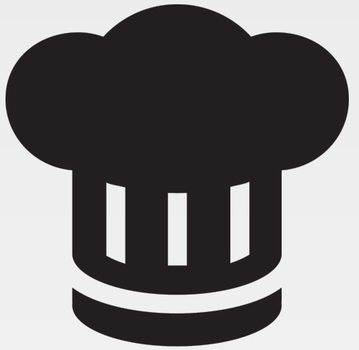
CHEF

CONCEPT

ohai_plugin resource

The ohai community cookbook contains a custom resource that will deliver the plugin from the cookbook to a default directory on the node (/etc/chef/ohai_plugins or c:\chef\ohai_plugins) and it will also tell ohai to reload ohai to evaluate the plugin.

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ❑ Define expectations for the recipe to deliver the Ohai plugin
- ❑ Create the recipe that delivers the Ohai plugin
- ❑ Define an integration test

Adding a Dependency on the Ohai Cookbook

~/cookbooks/httpd/metadata.rb

```
name 'httpd'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'all_rights'
description 'Installs/Configures httpd'
long_description 'Installs/Configures httpd'
version '0.1.0'

# If you upload to Supermarket you should set this so your cookbook
# gets a `View Issues` link
# issues_url 'https://github.com/<insert_org_here>/httpd/issues' if respond_to?(:issues_url)

# If you upload to Supermarket you should set this so your cookbook
# gets a `View Source` link
# source_url 'https://github.com/<insert_org_here>/httpd' if respond_to?(:source_url)
depends 'ohai'
```

Creating the Recipe to Add the Plugin

```
> chef generate recipe ohai_httpd_modules
```

```
Recipe: code_generator::recipe
  *
directory[/Users/franklinwebber/training/source/extending_cookbook
s-repo/cookbooks/httpd/spec/unit/recipes] action create (up to
date)
  *
cookbook_file[/Users/franklinwebber/training/source/extending_cook
books-repo/cookbooks/httpd/spec/spec_helper.rb] action
create_if_missing (up to date)
  *
template[/Users/franklinwebber/training/source/extending_cookbooks
-
```

Defining the Expectation Plugin Deployed

~/cookbooks/httpd/spec/unit/recipes/ohai_httpd_modules_spec.rb

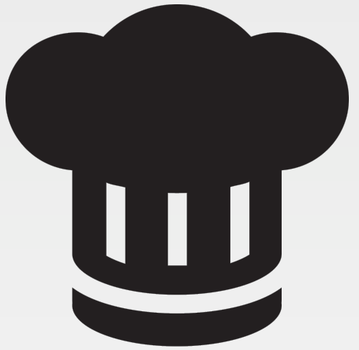
```
# ... UPPER PORTION OF THE SPECIFICATION ...
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end

    it 'installs the modules plugin' do
      expect(chef_run).to create_ohai_plugin('httpd_modules')
    end
  end
end
```

Execute the Tests to See Failure

```
> chef exec rspec
```

```
.........F

Failures:

  1) httpd::ohai_httpd_modules When all attributes are default, on
an unspecified platform installs the modules plugin
     Failure/Error: expect(chef_run).to
create_ohai_plugin('httpd_modules')

       expected "ohai_plugin[httpd_modules]" with action :create
to be in Chef run. Other ohai_plugin resources:
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the recipe to deliver the Ohai plugin
- ❑ Create the recipe that delivers the Ohai plugin
- ❑ Define an integration test

Defining the Ohai Plugin in the Recipe

~/cookbooks/httpd/recipes/ohai_httpd_modules.rb

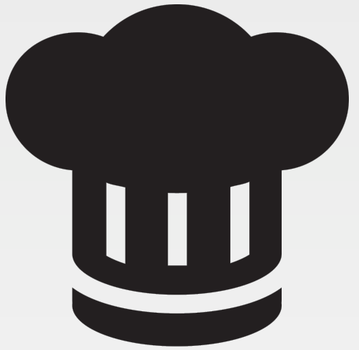
```
#
# Cookbook Name:: httpd
# Recipe:: ohai_httpd_modules
#
# Copyright (c) 2016 The Authors, All Rights Reserved.
include_recipe 'httpd::default'
ohai_plugin 'httpd_modules'
```

Executing the Tests to See Success

```
> chef exec rspec
```

```
...........

Finished in 5.77 seconds (files took 2.62 seconds to load)
12 examples, 0 failures
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the recipe to deliver the Ohai plugin
- ✓ Create the recipe that delivers the Ohai plugin
- ❑ Define an integration test

Appending the New Recipe to the Suite

~/httpd/.kitchen.yml

```
# ... TOP OF KITCHEN CONFIGURATION ...
suites:
  - name: default
    run_list:
      - recipe[httpd::default]
      - recipe[httpd::ohai_httpd_modules]
    verifier:
      inspec_tests:
        - test/recipes
    attributes:
```

Converging the Updated Default Suite

```
> kitchen converge
```

```
-----> Starting Kitchen (v1.11.1)
-----> Converging <default-centos-67>...
       resolving cookbooks for run list: ["httpd::default",
"httpd::ohai_httpd_modules"]
       Synchronizing Cookbooks:
         - ohai (4.2.2)
         - httpd (0.1.0)
         - compat_resource (12.14.7)
       Installing Cookbook Gems:
       Compiling Cookbooks...
       Recipe: httpd::ohai_httpd_modules
```

Defining an Expectation for the Plugin

~/httpd/test/recipes/ohai_httpd_modules.rb

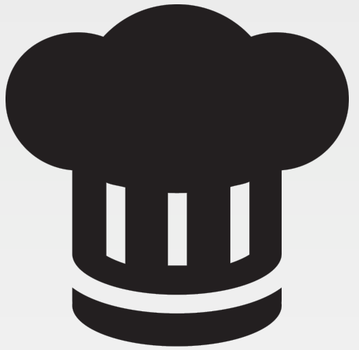
```
plugin_directory = '/tmp/kitchen/ohai/plugins'

describe command("ohai -d #{plugin_directory} apache") do
  its(:stdout) { should match(/core_module/) }
end
```

Verifying the New Expectation

```
> kitchen verify
```

```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <ohai-plugin-centos-67>...
       Use `/home/chef/httpd/test/recipes/default` for testing

Target:  ssh://kitchen@localhost:32768

  ✔  Command ohai -d /tmp/kitchen/ohai/plugins apache stdout
should match /core_module \(static\)/

Summary: 1 successful, 0 failures, 0 skipped
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the recipe to deliver the Ohai plugin
- ✓ Create the recipe that delivers the Ohai plugin
- ✓ Define an integration test

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

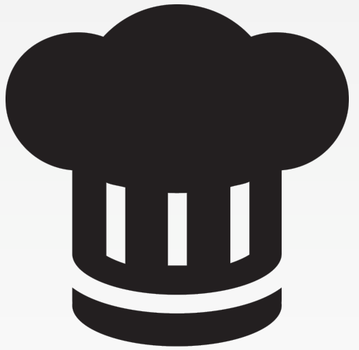
TECHNICAL
- ✓ Build the Ohai plugin
- ✓ Build a Recipe to deliver the plugin

TIPS
- ❑ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF™

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ❑ Configure the node to automatically load ohai plugins
- ❑ Enable ohai hints
- ❑ Remove plugins that you do not want executed
- ❑ Choose only the plugins you want executed

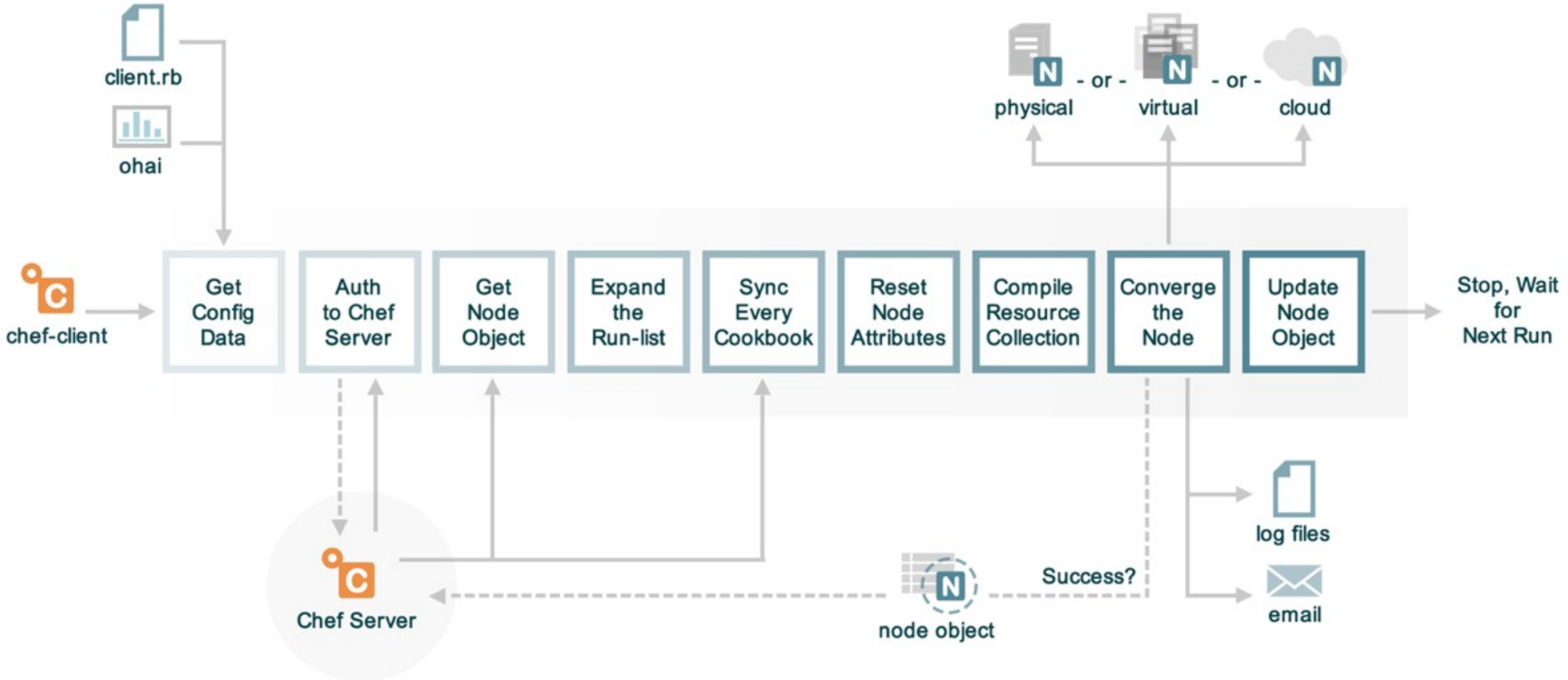
CONCEPT

Node Configuration File

All nodes have a configuration file that contain details about node, overrides, and other data.

CHEF

The Anatomy of a chef-client Run



CHEF

CONCEPT

chef-client cookbook

The chef-client cookbook contains a number of useful recipes:

- service (default)
  - Run chef-client as a service, converging at a periodic interval
- delete_validation
  - Remove the organization validation key [SECURITY ISSUE]
- config
  - Define node configuration

https://supermarket.chef.io/cookbooks/chef-client

Viewing the chef-client config Recipe

~/cookbooks/chef-client/recipes/config.rb

```
# ... OTHER RESOURCES ...
template "#{node["chef_client"]["conf_dir"]}/client.rb" do
  source 'client.rb.erb'
  owner d_owner
  group d_group
  mode 00644
  variables(
    :chef_config => node['chef_client']['config'],
    :chef_requires => chef_requires,
    :ohai_disabled_plugins  => node['ohai']['disabled_plugins'],
    :start_handlers => node['chef_client']['config']['start_handlers'],
    :report_handlers => node['chef_client']['config']['report_handlers'],
    :exception_handlers => node['chef_client']['config']['exception_handlers']
  )
  notifies :create, 'ruby_block[reload_client_config]', :immediately
end
```


CHEF

~/cookbooks/chef-client/templates/default/client.rb.erb

```
<% if node.attribute?("ohai") && node["ohai"].attribute?("plugin_path") -%>

Ohai::Config[:plugin_path] << "<%= node["ohai"]["plugin_path"] %>"
<% end -%>
<% unless @ohai_disabled_plugins.empty? -%>
Ohai::Config[:disabled_plugins] = [<%= @ohai_disabled_plugins.map { |k| k...

<% end -%>
```

PROBLEM

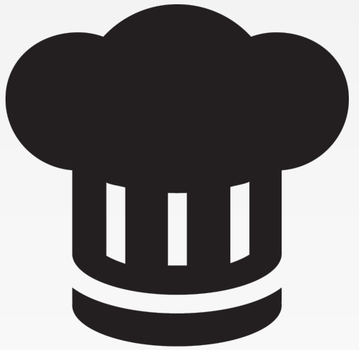
The Work

- Add the chef-client cookbook to your cookbook collection
- Provide attributes in a wrapper cookbook, role, or environment for:

```
node['ohai']['plugin_path']
```

- Add chef-client cookbook to every node's run list

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ❑ Enable ohai hints
- ❑ Remove plugins that you do not want executed
- ❑ Choose only the plugins you want executed

CONCEPT

Not all Node Plugins are Executed

Some of the plugins will not automatically run against your node. This is particularly true of the cloud provider plugins. As the node does not often know the environment in which is being run.

CONCEPT

Ohai Hints

For those plugins that are not executed you can leave them a hint file that will ensure they are executed.

CHEF™

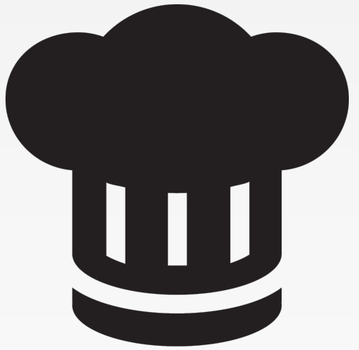
PROBLEM

The Work

- Add the ohai cookbook to your cookbook collection
- Define a recipe that uses the ohai cookbook's ohai_hint resource
- Add this recipe to every node's run list

CHEF

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- ❑ Remove plugins that you do not want executed
- ❑ Choose only the plugins you want executed

~/cookbooks/chef-client/templates/default/client.rb.erb

```
<% if node.attribute?("ohai") && node["ohai"].attribute?("plugin_path") -%>

Ohai::Config[:plugin_path] << "<%= node["ohai"]["plugin_path"] %>"
<% end -%>
<% unless @ohai_disabled_plugins.empty? -%>
Ohai::Config[:disabled_plugins] = [<%= @ohai_disabled_plugins.map { |k| k...

<% end -%>
```

Viewing the chef-client config Recipe

~/cookbooks/chef-client/recipes/config.rb

```
# ... OTHER RESOURCES ...
template "#{node["chef_client"]["conf_dir"]}/client.rb" do
  source 'client.rb.erb'
  owner d_owner
  group d_group
  mode 00644
  variables(
    :chef_config => node['chef_client']['config'],
    :chef_requires => chef_requires,
    :ohai_disabled_plugins  => node['ohai']['disabled_plugins'],
    :start_handlers => node['chef_client']['config']['start_handlers'],
    :report_handlers => node['chef_client']['config']['report_handlers'],
    :exception_handlers => node['chef_client']['config']['exception_handlers']
  )
  notifies :create, 'ruby_block[reload_client_config]', :immediately
end
```


CHEF

PROBLEM

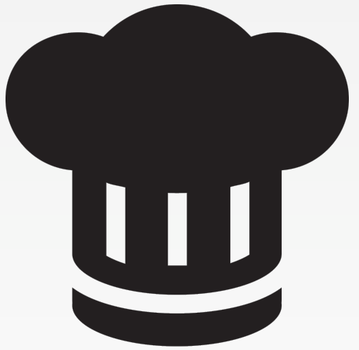
The Work

- Add the chef-client cookbook to your cookbook collection
- Select attributes you want to remove
- Find the name of the plugin that provides those attributes
- Provide attributes in a wrapper cookbook, role, or environment for:

```
node['ohai']['disabled_plugins']
```

- Add chef-client cookbook to every node's run list

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- ✓ Remove plugins that you do not want executed
- ❑ Choose only the plugins you want executed

CONCEPT

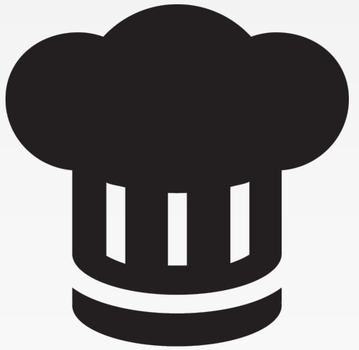
Whitelist Node Attributes

When it seems like you are disabling far too many plugins and you want to consider attacking it from the other side:

http://ckbk.it/whitelist-node-attrs

CHEF

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- ✓ Remove plugins that you do not want executed
- ✓ Choose only the plugins you want executed

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

TECHNICAL
- ✓ Build the Ohai plugin
- ✓ Build a Recipe to deliver the plugin

TIPS
- ✓ Tuning Ohai

DISCUSSION
- ❑ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF™

Let Us Have a Discussion

"Ask a Question" and I will answer it.

"Rate this" presentation to leave your feedback and help me do my work better.

To share, click on the appropriate professional/social network in **"Details"**

CHEF

DISCUSSION

Q&A

What questions can I answer for you?

CHEF

Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

TECHNICAL
- ✓ Build the Ohai plugin
- ✓ Build a Recipe to deliver the plugin

TIPS
- ✓ Tuning Ohai

DISCUSSION
- ✓ Questions
- ❑ Resources
- ❑ Hack Time!

CHEF™

DISCUSSION

Resources for Ohai

What resources have helped you understand Ohai better?

CHEF™

Check The Attachments

All of the resources I am going to provide you can be found under **"Attachments"**.

Presentation Resources

Slides

github.com/chef-training/webinars/raw/master/build_a_custom_ohai_plugin.pdf

Example Cookbook

github.com/chef-training/webinar-ohai-repo

chefspec-ohai gem

github.com/burtlo/chefspec-ohai

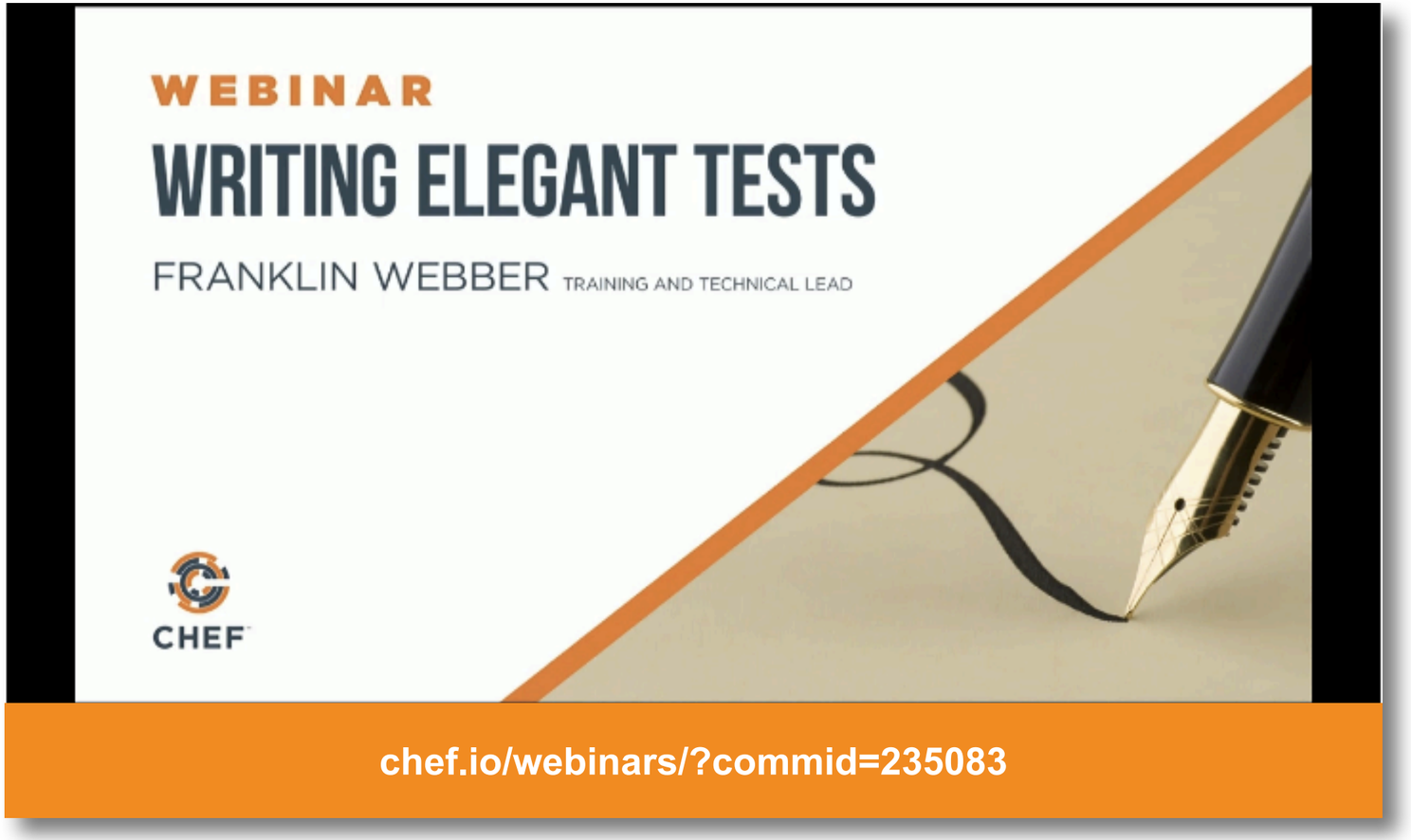
Learn Chef Tutorial

learn.chef.io/tutorials/build-an-ohai-plugin

Creating a Gem like chefspec-ohai

Learn more about the structure of the **chefspec-ohai** gem which defined a lot of those special test helpers.

Write more elegant tests!



Agenda

CONCEPTS
- ✓ *Oh hai* Ohai!
- ✓ Meet and Greet with Ohai's Plugins

TECHNICAL
- ✓ Build the Ohai plugin
- ✓ Build a Recipe to deliver the plugin

TIPS
- ✓ Tuning Ohai

DISCUSSION
- ✓ Questions
- ✓ Resources
- ❑ Hack Time!

CHEF™

CHEF™