Ohai

CHEF

Objectives

After completing this module, you should be able to:

- Execute the Ohai command-line tool to return an attribute
- Describe when Ohai is loaded in the chef-client run
- Describe when new attributes for the node are stored
- Describe precedence of attributes collected by Ohai

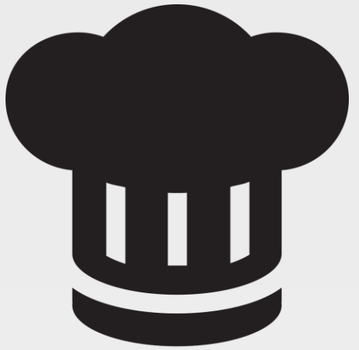
CONCEPT

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

CONCEPT

All About The System

Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

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

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"
]
```


CHEF

Running Ohai to Show the Hostname

```
> ohai hostname
```

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


CHEF

Running Ohai to Show the Memory

```
> ohai memory
```


```
{
  "swap": {
    "cached": "0kB",
    "total": "0kB",
    "free": "0kB"
  },
  "hugepages": {
    "total": "0",
    "free": "0",
    "reserved": "0",
    "surplus": "0"
  },
  "total": "1018184kB",
  "free": "280972kB",
  "buffers": "54340kB",
```


CHEF

Running Ohai to Show the Total Memory

```
> ohai memory/total
```

```
[
  "1018184kB"
]
```


CHEF

Running Ohai to Show the CPU

```
> ohai cpu
```

```
{
  "0": {
    "vendor_id": "GenuineIntel",
    "family": "6",
    "model": "63",
    "model_name": "Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz",
    "stepping": "2",
    "mhz": "2400.078",
    "cache_size": "30720 KB",
    "physical_id": "0",
```

Running Ohai to Show the First CPU

```
> ohai cpu/0
```

```
{
  "vendor_id": "GenuineIntel",
  "family": "6",
  "model": "63",
  "model_name": "Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz",
  "stepping": "2",
  "mhz": "2400.078",
  "cache_size": "30720 KB",
  "physical_id": "0",
  "core_id": "0",
```


CHEF

Running Ohai to Show the First CPU Mhz

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

```
[
  "2400.078"
]
```

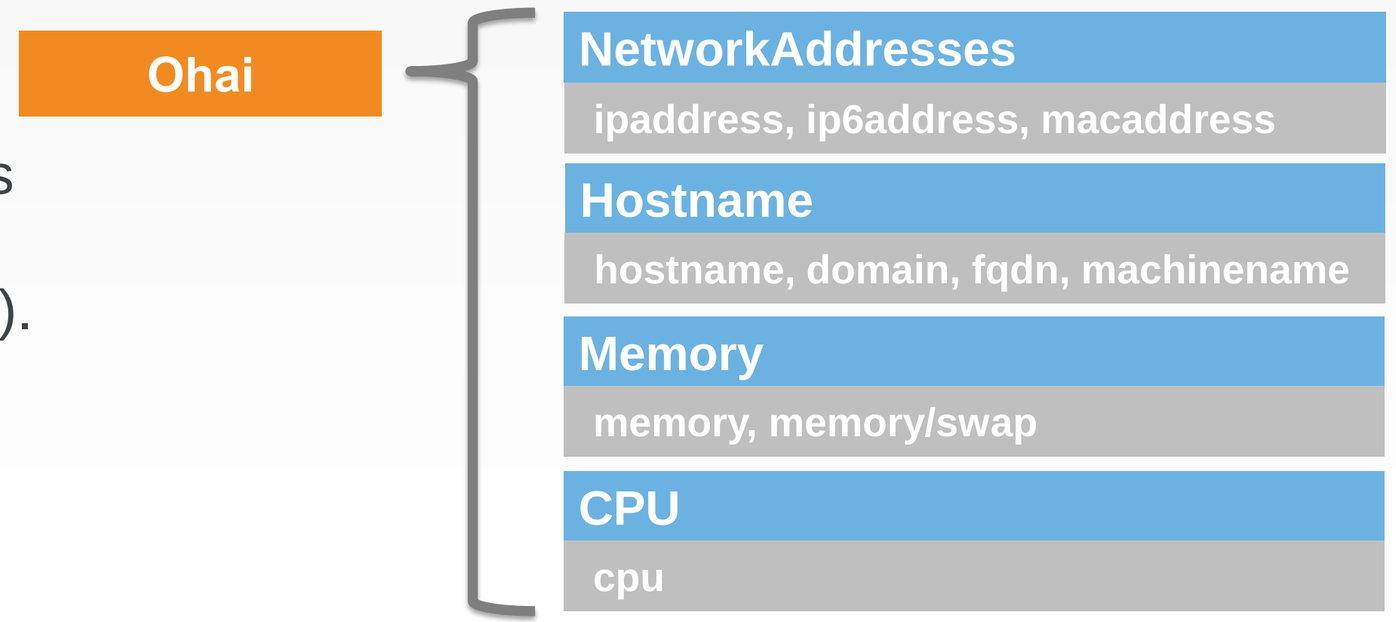

CHEF

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

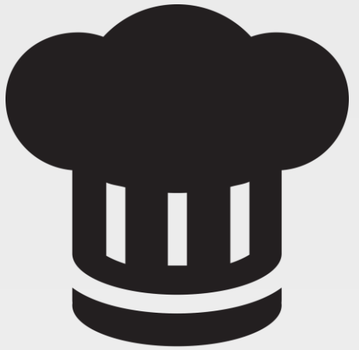
It is possible to define your own plugins and have Ohai load those plugins.

```
> ohai -d PATH_TO_CUSTOM_PLUGINS
```

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

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

CHEF

CONCEPT

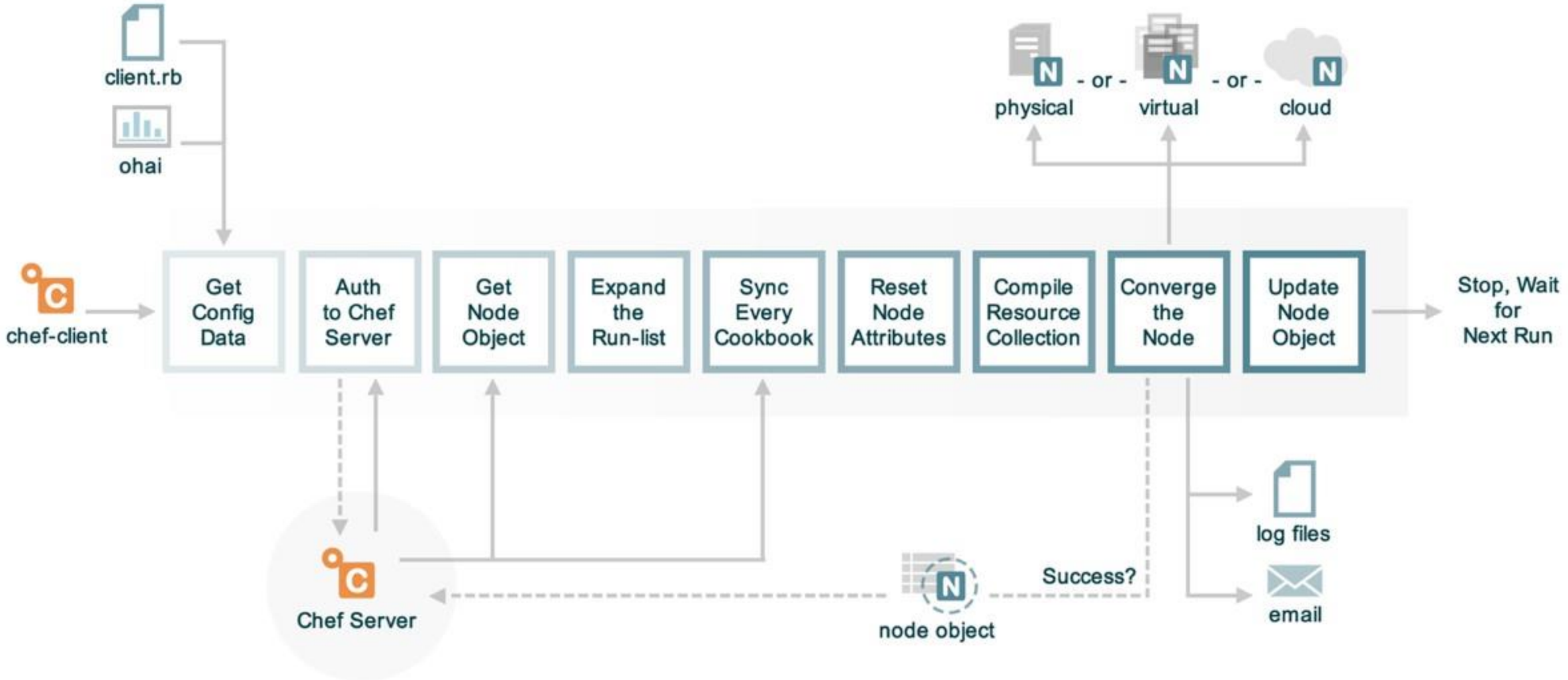
chef-client

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



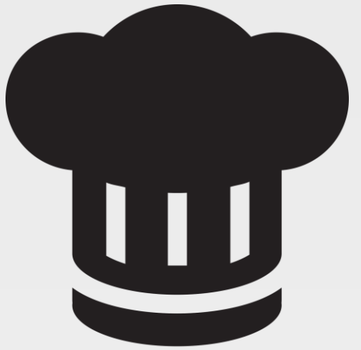
CHEF

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

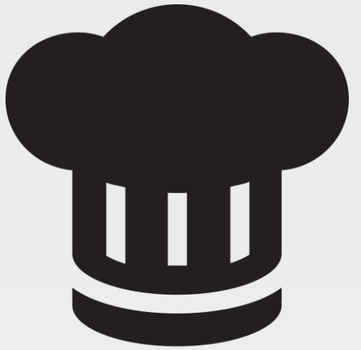
Node Attributes

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.

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

Discussion

When might you execute ohai from the comand-line to gather data about the system?

Why might it be important to collect details about the system, through Ohai, early in the chef-client run?

What kind of data should be collected and stored within Ohai? What kind of data should it not collect?

DISCUSSION

Q&A

What questions can we answer for you?

CHEF

CHEF™