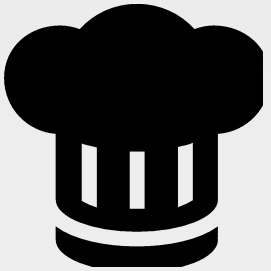
Managing a Large Number of Servers

Have you ever had to manage a large number of servers that were almost identical?

How about a large number of identical servers except that each one had to have host-specific information in a configuration file?



Details About the Node

Displaying system details in the MOTD definitely sounds useful.

Objective:

- Update the MOTD file contents, in the "workstation" cookbook, to include node details

Some Useful System Data

- ❑ IP Address
- ❑ hostname
- ❑ memory
- ❑ CPU - MHz

Discover the IP Address

```
$ hostname -I
```

```
172-31-57-153
```

Discover the Host Name

```
$ hostname
```

```
ip-172-31-57-153
```

Discovering the Memory

```
$ cat /proc/meminfo
```

```
MemTotal:         502272 kB
MemFree:          118384 kB
Buffers:          141156 kB
Cached:           165616 kB
SwapCached:            0 kB
Active:           303892 kB
Inactive:          25412 kB
Active(anon):      22548 kB
Inactive(anon):      136 kB
Active(file):     281344 kB
Inactive(file):    25276 kB
Unevictable:           0 kB
Mlocked:               0 kB
```

Discover the CPU - MHz

```
$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz
stepping  : 4
cpu MHz         : 2399.998
cache size      : 15360 KB
fpu       : yes
fpu_exception   : yes
cpuid level     : 13
wp        : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36
```


CHEF

Adding the CPU

~/cookbooks/workstation/recipes/setup.rb

```
file '/etc/motd' do
  content 'Property of ...

  IPADDRESS: 172-31-57-153
  HOSTNAME : ip-172-31-57-153
  MEMORY   : 502272 kB
  CPU      : 2399.998 MHz
'
  mode '0644'
  owner 'root'
  group 'root'
end
```

Return Home and Apply workstation Cookbook

```
$ cd ~
$ sudo chef-client --local-mode -r "recipe[workstation]"
```

```
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
Converging 6 resources
Recipe: workstation::setup
  * yum_package[nano] action install (up to date)
  * yum_package[vim] action install (up to date)
  * yum_package[emacs] action install (up to date)
  * yum_package[tree] action install (up to date)
  * yum_package[git] action install (up to date)
```

Verify that the /etc/motd Has Been Updated

```
$ cat /etc/motd
```

```
Property of ...

  IPADDRESS: 172-31-57-153
  HOSTNAME : ip-172-31-8-68
  MEMORY   : 605048 kB
  CPU      : 1795.672
```


CHEF

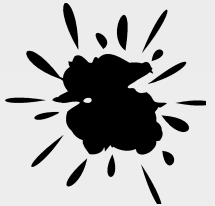
DISCUSSION

Capturing System Data

What are the limitations of the way we captured this data?

How accurate will our MOTD be when we deploy it on other systems?

Are these values we would want to capture in our tests?



Hard Coded Values

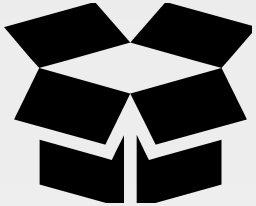
The values that we have derived at this moment may not be the correct values when we deploy this recipe again even on the same system!

DISCUSSION

Data In Real Time

How could we capture this data in real-time?

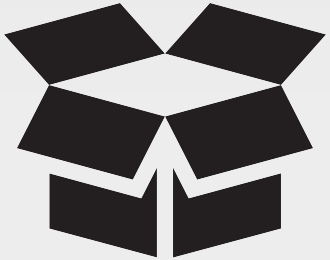
CONCEPT



Ohai!

Ohai is a tool that already captures all the data that we similarly demonstrated finding.

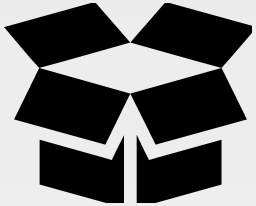
http://docs.chef.io/ohai.html



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. Ohai is required by the chef-client and must be present on a node. (Ohai is installed on a node as part of the chef-client install process.)

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



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).

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

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"
]
```

Running Ohai to Show the Memory

```
> ohai memory
```

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

Running Ohai to Show the Total Memory

```
> ohai memory/total
```

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


CHEF

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",
```

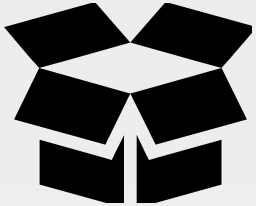
Running Ohai to Show the First CPU Mhz

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

```
[
  "1795.673"
]
```


CHEF

CONCEPT

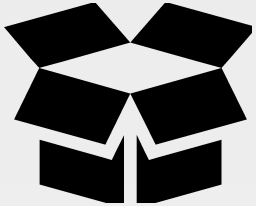


ohai + chef-client = <3

chef-client and chef-apply automatically executes ohai and stores the data about the node in an object we can use within the recipes named node.

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

CONCEPT

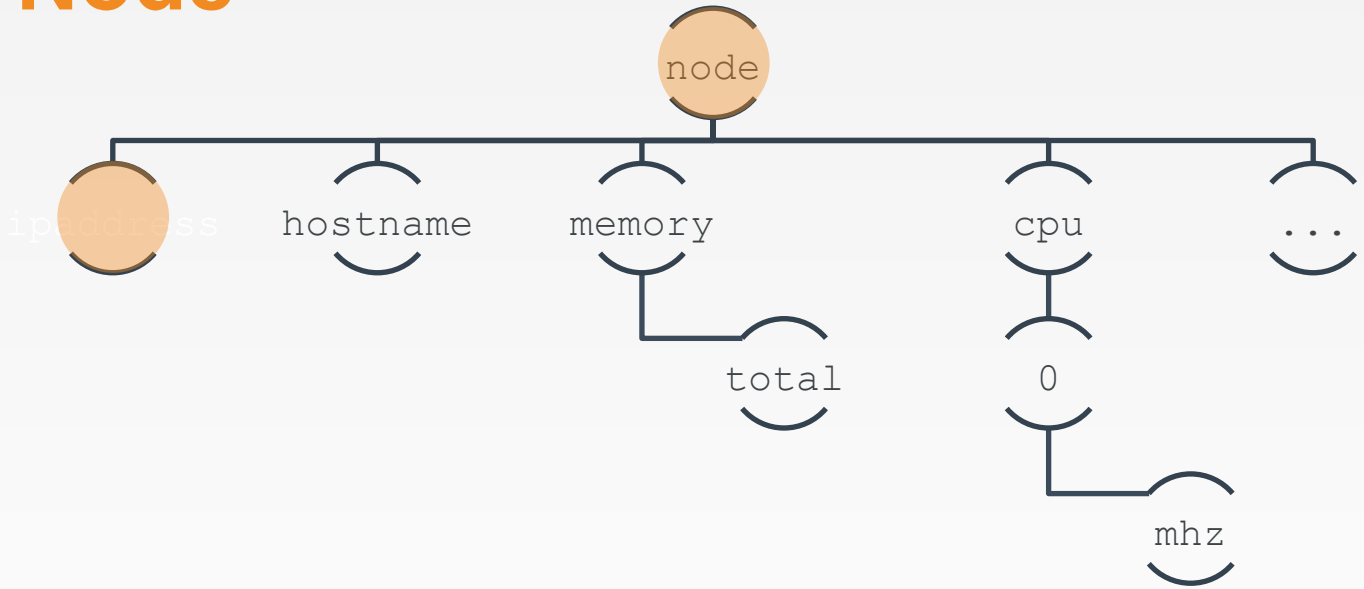


The Node Object

The node object is a representation of our system.
It stores all the attributes found about the system.

http://docs.chef.io/nodes.html#attributes

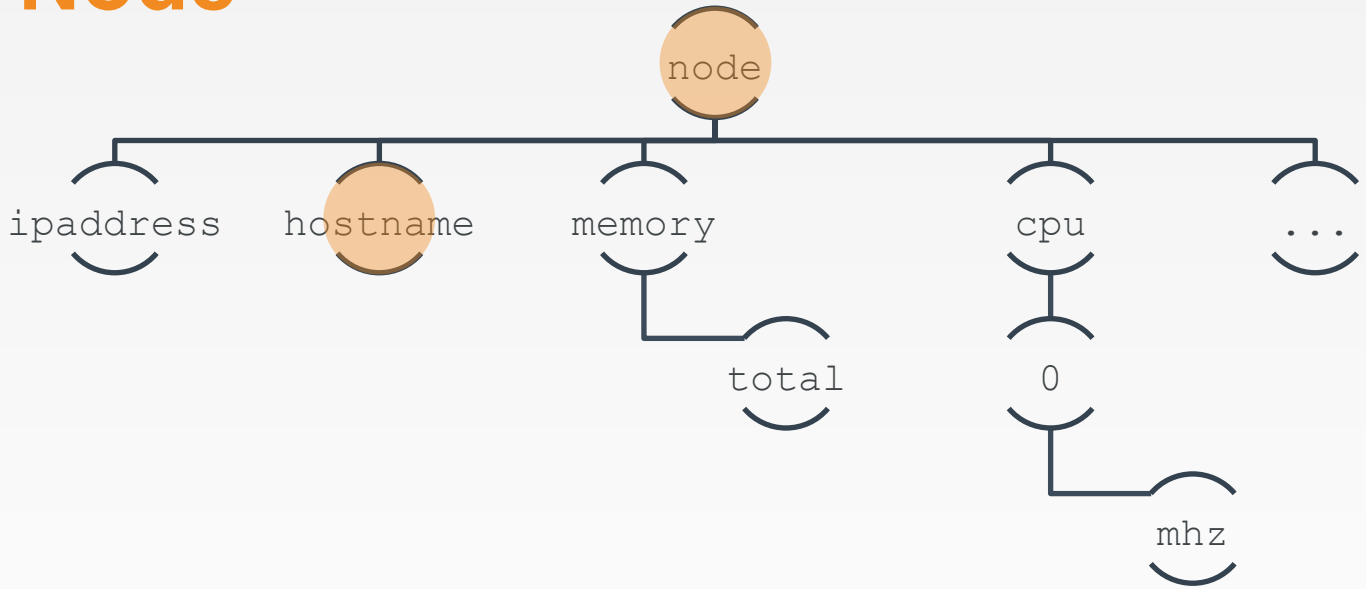
The Node




```
IPADDRESS: 172-31-57-153
```

```
node['ipaddress']
```

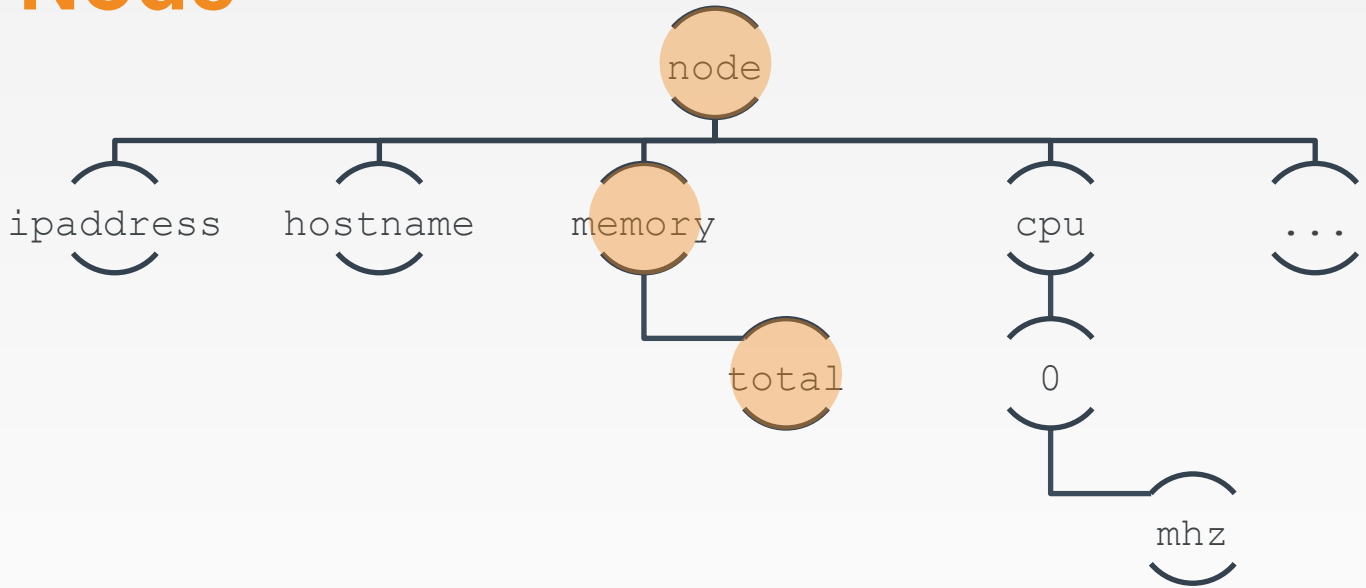
The Node




```
HOSTNAME: ip-172-31-57-153
```

```
node['hostname']
```

The Node

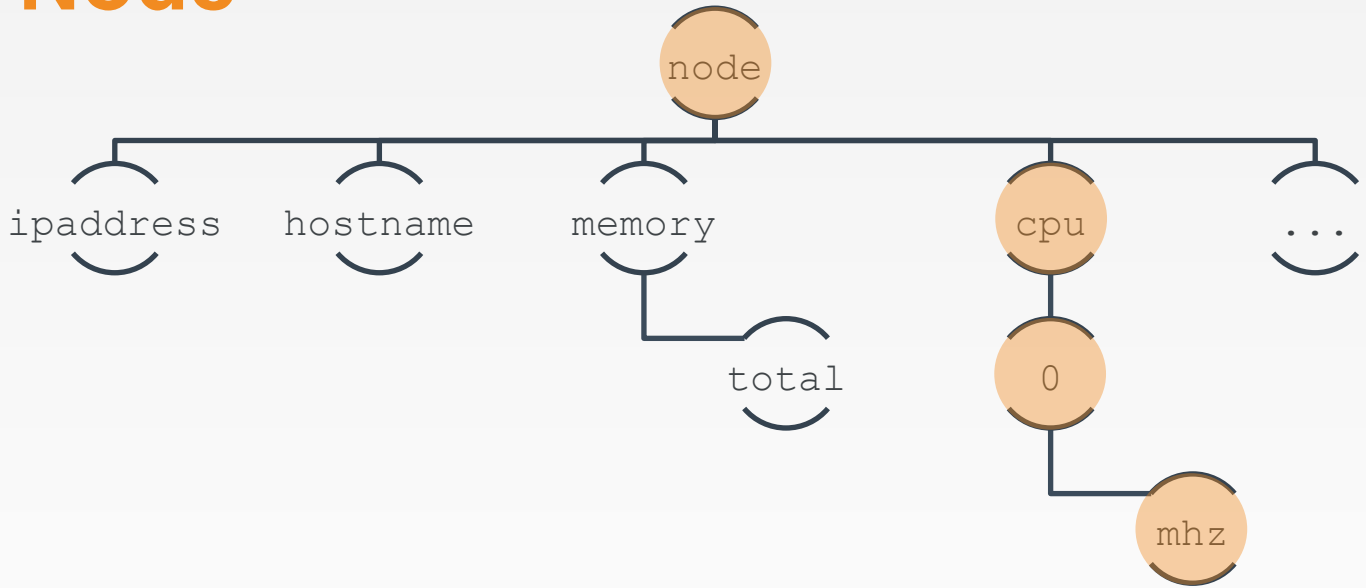



```
MEMORY: 502272kB
```

```
node['memory']['total']
```


CHEF

The Node




```
CPU: 2399.998MHz
```

```
node['cpu']['0']['mhz']
```