Details About the System

Finding and Displaying Information About Our System

CHEF™

Objectives

After completing this module, you should be able to

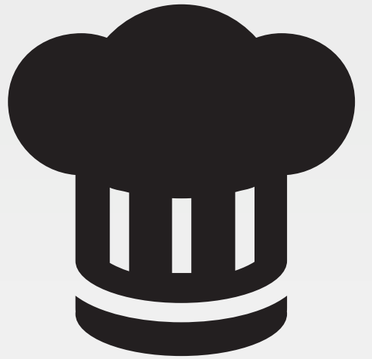
- Capture details about a system
- Use the node object within a recipe
- Use Ruby's string interpolation
- Update the version of a cookbook

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?

CHEF™



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

CHEF™

Some Useful System Data

- IP Address
- hostname
- memory
- CPU - MHz

GL: Discover the IP Address

```
$ hostname -I
```

```
172.31.8.68 172.17.42.1
```


CHEF

Demo: Finding Platform Info

```
> cat /etc/os-release
```

```
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"
…
…
REDHAT_SUPPORT_PRODUCT_VERSION="7"
```

CHEF

GL: 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
```


CHEF

GL: 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
```

GL: Adding the CPU

~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 104.236.192.102
  HOSTNAME : banana-stand
  MEMORY   : 502272 kB
  CPU      : 2399.998 MHz
'
  mode '0644'
  owner 'root'
  group 'root'
end
```


CHEF

GL: Introducing a Change

By creating a change we have introduced risk.

Lets run our cookbook tests before we apply the updated recipe.

GL: Change into Our Cookbook

```
$ cd ~/cookbooks/workstation
```

GL: Return Home and Apply workstation Cookbook

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

```
Starting Chef Client, version 12.13.37
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation (0.1.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - update content in file /etc/motd from d100eb to 63e97f
```

GL: Verify that the /etc/motd Has Been Updated

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

```
Property of ...

  IPADDRESS: 172.31.8.68
  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?

CHEF

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?

CHEF

CONCEPT

Ohai!

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

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

CHEF

Ohai!

```
$ 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

Ohai!

```
$ ohai memory
```


```
{
  "swap": {
    "cached": "0kB",
    "total": "0kB",
    "free": "0kB"
  },
  "hugepages": {
    "total": "0",
    "free": "0",
    "reserved": "0",
    "surplus": "0"
  },
  "total": "1014424kB",
 …
```


CHEF

Ohai!

```
$ ohai memory/swap
```

```
{
  "cached": "0kB",
  "total": "0kB",
  "free": "0kB"
}
```


CHEF

Ohai!

```
$ ohai hostname
```

```
[
  "ip-172-31-43-221"
]
```


CHEF

Ohai!

```
$ 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": "2400.092",
    "cache_size": "30720 KB",
    "physical_id": "0",
    "core_id": "0",
    "cores": "1",
    "flags": [
      "fpu", ……
```


CHEF

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).

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

CHEF

CONCEPT

The Node Object

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/nodes.html#attributes

CHEF

CONCEPT

String Interpolation

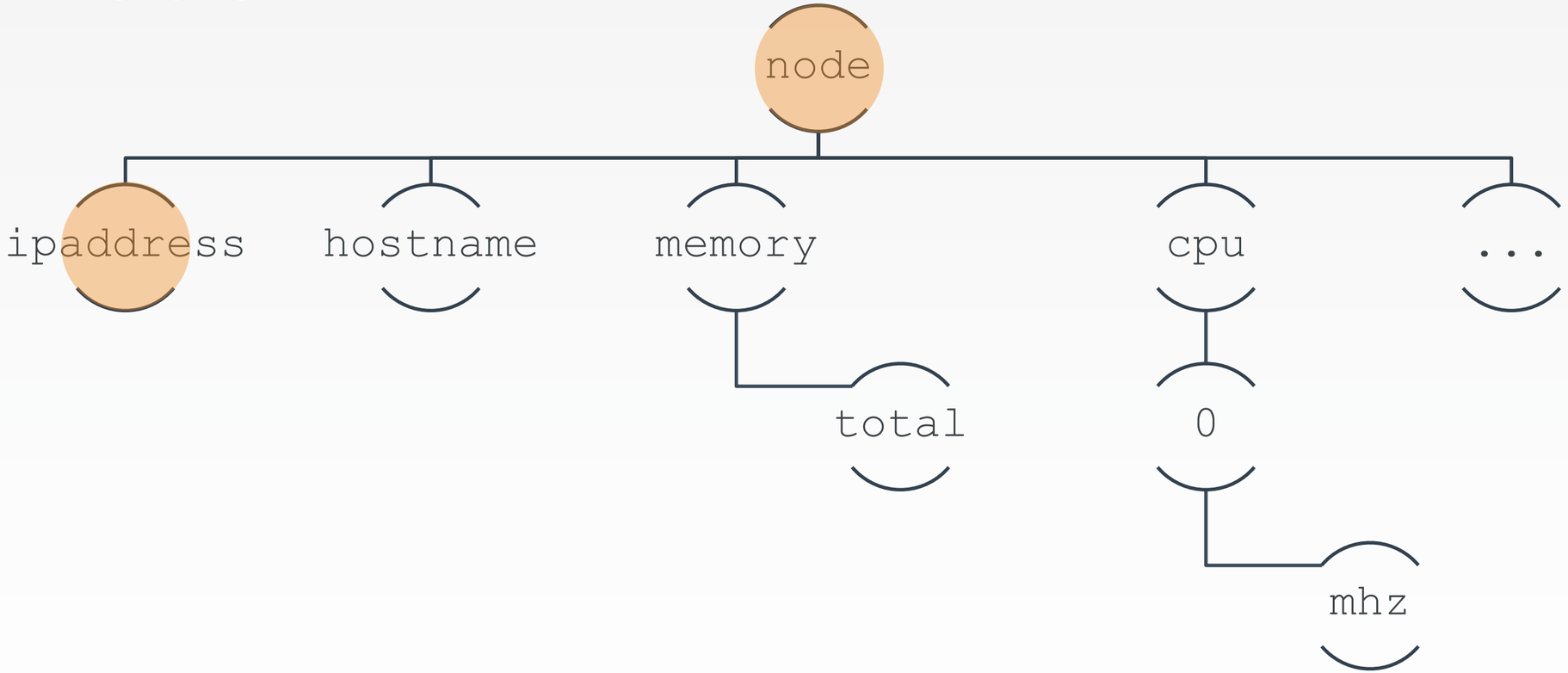
Use variables in Ruby

```
var = 4
puts "Value is: #{var}"
```

http://en.wikipedia.org/wiki/String_interpolation#Ruby

CHEF

The Node

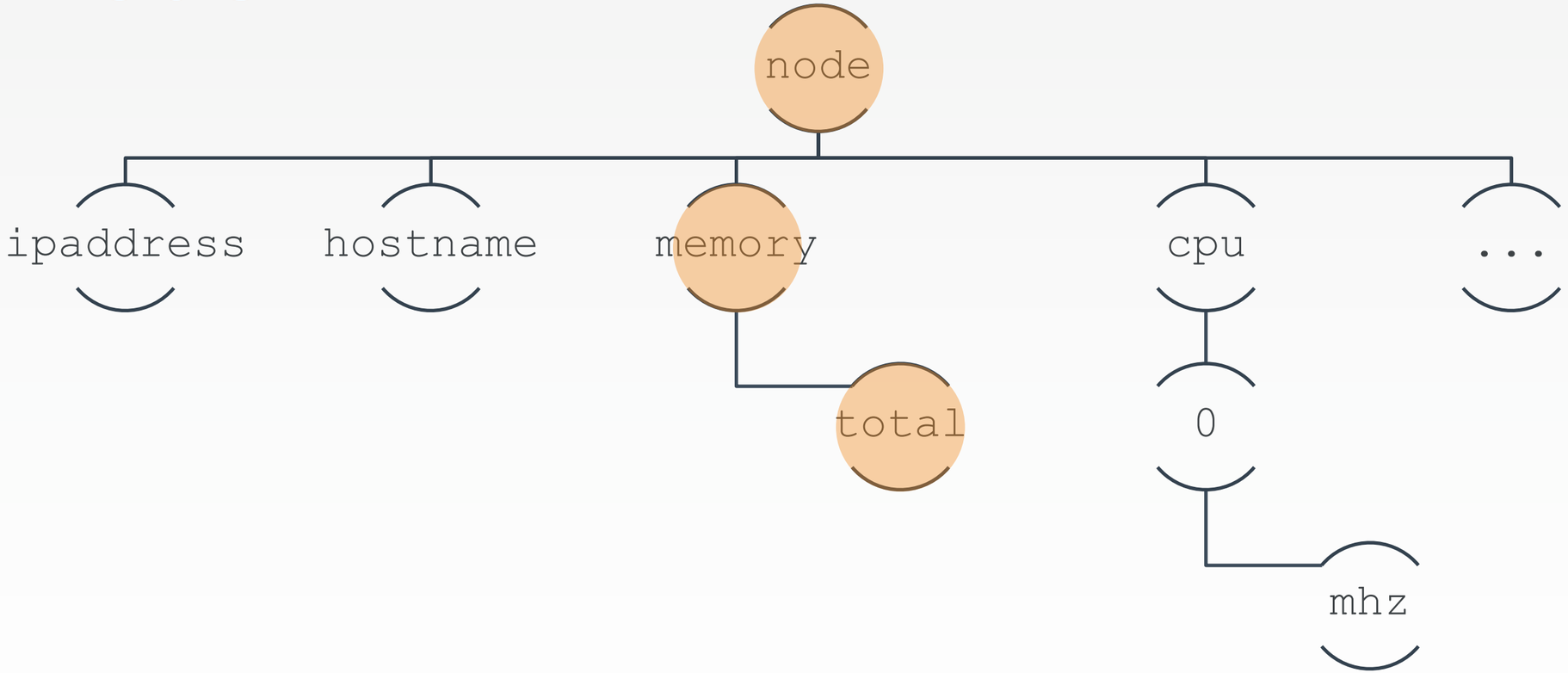



```
IPADDRESS: 104.236.192.102
```

```
"IPADDRESS: #{node['ipaddress']}"
```


CHEF

The Node

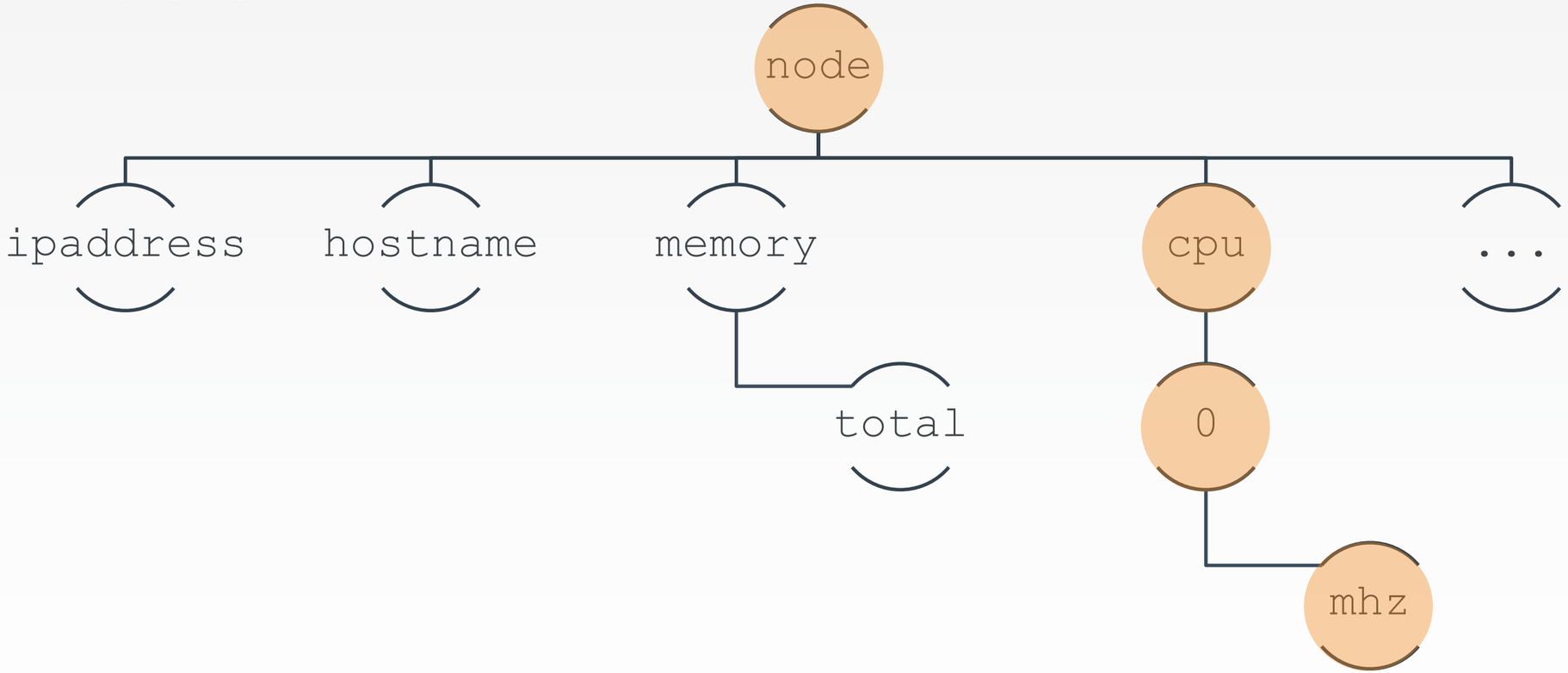



```
MEMORY: 502272kB
```

```
"Memory: #{node['memory']['total']}"
```


CHEF

The Node




```
CPU: 2399.998MHz
```

```
"CPU: #{node['cpu']['0']['mhz']}"
```


CHEF

GL: Using the Node's Attributes

~/cookbooks/workstation/recipes/setup.rb

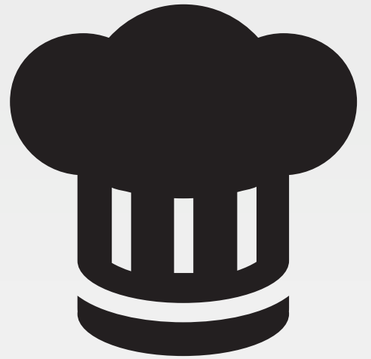
```
# ... PACKAGE RESOURCES ...
file '/etc/motd' do
  content "Property of ...


  IPADDRESS: #{node['ipaddress']}
  HOSTNAME : #{node['hostname']}
  MEMORY   : #{node['memory']['total']}
  CPU      : #{node['cpu']['0']['mhz']}
"
  mode '0644'
  owner 'root'
  group 'root'
end
```

Lab: Apply the Workstation's Default Recipe

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

```
Starting Chef Client, version 12.13.37
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation (0.1.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)
```



Changes Mean a New Version

Let's bump the version number and check in the code to source control.

Objective:

- ❑ Update the version of the "workstation" cookbook
- ❑ Commit the changes to the "workstation" cookbook to version control

CONCEPT

Cookbook Versions

A cookbook version represents a set of functionality that is different from the cookbook on which it is based.

https://docs.chef.io/cookbook_versions.html

CHEF

CONCEPT

Semantic Versions

Given a version number **MAJOR**.**MINOR**.**PATCH**, increment the:

- **MAJOR** version when you make incompatible API changes
- **MINOR** version when you add functionality in a backwards-compatible manner
- **PATCH** version when you make backwards-compatible bug fixes

http://semver.org

CHEF

Lab: Node Details in the Webserver

In this lab, the file resource named '/var/www/html/index.html' is created with the content that includes the node details:

- ipaddress
- hostname

- ❑ Run chef-client to locally apply the "apache" cookbook's default recipe.
- ❑ Update the version of the "apache" cookbook

Lab: Apache Recipe

~/cookbooks/apache/recipes/server.rb

```
...

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
"
end

...
```

Lab: Run chef-client to Apply the Apache Cookbook

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

```
Starting Chef Client, version 12.13.37
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache (0.1.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::server
  * yum_package[httpd] action install (up to date)
  * file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 17d291 to 158cae
```

Lab: Update the Cookbook Version

~/cookbooks/apache/metadata.rb

```
name              'apache'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures apache'
long_description  'Installs/Configures apache'
version           '0.2.0'
```

COMMIT

Lab: Commit Your Work

```
$ cd ~/cookbooks/apache
$ git add .
$ git status
$ git commit -m "Release version 0.2.0"
```

Lab:
60 minutes

https://github.com/shekhar2010us/chef-essentials-repo-15/blob/master/labs/chapter%206.md

DISCUSSION

Discussion

How are the details about the system available within a recipe?

How does the version number help convey information about the state of the cookbook?

DISCUSSION

Q&A

What questions can we help you answer?

- Ohai
- Node Object
- Node Attributes
- String Interpolation
- Semantic Versions

CHEF

CHEF™