Pre-Requisites

In order to maximize classroom time focused of using Chef, we are providing a pre-configured Chef workstation available via E2. The workstation is based on a bare CentOS 6 AMI in us-west-2 or us-east-1 (location depending). Instructions replicating work done to prepare this VM as a workstation for this class can be found below.

We do not recommend running your own Chef workstation for this class. We will make live edits to the workstation for object lessons. The examples will not work on non RHEL 6 derivative systems.

After class, you may access a version of this workstation by starting a new EC2 instance in **us-west-2** or **us-east-1**.

AMI:	**ami-69e6d659 (us-west-2) / ami-4d2fd426 (us-east-1)**
User:	**root**
Password:	**chef&aws2015**

Note: for this class, we will all work from EC2 instances provided in a shared account (to avoid special configurations and ensure consistency in the classroom). After class, if you launch your own EC2 instance, the SSH key you use will belong to the ec2-user account. You may access the instance this way, but all work in the classroom material will be done with the root account. We recommend you login (or sudo) as root.

Workstation Configuration

If you want to create your own workstation AMI to model the image we have provided follow these instructions. The public AMI above is based on a bare publicly available CentOS 6 x86_64 image in us-west-2 (ami-c00d80f0) or us-east-1 (ami-bf5021d6). The following steps were completed (exact commands captured in root's history on the AMI):

1) Enable RootLogin and PasswordAuthentication in /etc/ssh/sshd_config, /etc/cloud/cloud.cfg, and /root/.ssh/authorized_keys
2) Install the ChefDK -- https://docs.chef.io/install_dk.html
3) Update the chef-provisioning and chef-provisioning-aws RubyGems
4) Install a private RSA key necessary if working in the shared classroom AWS account, /root/.ssh/aws-popup-chef
5) Create a mock AWS config file, /root/.aws/config

Workstation Setup

Your workstation IP: ____

Login: **root**

Password: **chef&aws2015**

SSH to your workstation

```
$ ssh root@WORKSTATION_IP
```

Terminology

Node

Resource

Recipe

Cookbook

Run list

Role

Search

Resources

Lab 1

```
$ chef-apply -e "package 'vim'"
$ chef-apply -e "package 'emacs'"
$ chef-apply -e "package 'nano'"
```

Lab 2

Create file **hello.rb** in root’s home directory.

FILE: ~/hello.rb

```
file ”hello.txt” do
  action :create
  content ”Hello, World!”
  mode ”0644”
  owner ”root”
  group ”root”
end
```

Run single recipe from command line with chef-apply:

```
$ chef-apply hello.rb
```

Verify that file **hello.txt** was created as expected:

```
$ cat hello.txt
```

Lab 3

motd stands for **m**essage **o**f **t**he **d**ay. We want to create a recipe that will display **Property of COMPANY NAME** upon login. On CentOS systems, a "message of the day" is controlled by file /etc/motd.

Create file **motd.rb** in root's home directory.

FILE: ~/motd.rb

```
file "/etc/motd" do
  content "Property of COMPANY NAME\n"
  action :create
  mode "0644"
  owner "root"
  group "root"
end
```

Run single recipe from command line with chef-apply:

```
$ chef-apply motd.rb
```

Verify that file **/etc/motd** was updated as expected:

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

___________________________END OF MODULE___________________________

Describing Policies

Lab 4

Right now our motd file is created with the content of "Property of COMPANY NAME" which intermingles our data (content that might frequently change) and policy (how that file should be configured). In this lab we want to manage data and policy separately.

1. Create a repository to store our cookbooks.

```
$ chef generate repo chef-repo
```

2. Create a cookbook called motd.

```
$ chef generate cookbook cookbooks/motd
```

3. Copy our original **motd.rb** file into our new cookbook.

```
$ cat ~/motd.rb >> ~/chef-
repo/cookbooks/motd/recipes/default.rb
```

4. Update our default recipe.

FILE: ~/chef-repo/cookbooks/motd/recipes/default.rb

```
template "/etc/motd" do
  source "motd.erb"
  action :create
  mode "0644"
  owner "root"
  group "root"
end
```

5. Create a template called motd in the motd cookbook.

```
$ chef generate template cookbooks/motd motd -s /etc/motd
```

6. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r "recipe[motd]"
```

_________________________END OF MODULE_________________________

Creating a simple webserver

Lab 5

We want to install Apache on our node and display a custom home page.
Current directory should be **~/chef-repo**.

1. Create a cookbook called webserver.

```
$ chef generate cookbook cookbooks/webserver
```

2. Update our default recipe.

FILE: ~/chef-repo/cookbooks/webserver/recipes/default.rb

```
package "httpd"

service "httpd" do
 action [ :enable, :start ]
end

template "/var/www/html/index.html" do
  source "index.html.erb"
end

service "iptables" do
  action [ :disable, :stop ]
end
```

3. Create a template called index.html in the webserver cookbook.

```
$ chef generate template cookbooks/webserver index.html
```

Note: When creating templates, chef will automatically append the suffix .erb which stands for embedded ruby.

4. Update the contents of the template index.html.erb.

```
FILE: ~/chef-
repo/cookbooks/webserver/template/default/index.html.erb
<html>
 <body>
  <h1>Hello, World!</h1>
 </body>
</html>
```

5. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r "recipe[webserver]"
```

6. Verify that we have apache running with our custom index.html.

```
$ curl http://localhost
```

7. Apply our recipe to the node again using chef-client.

```
$ chef-client --local-mode -r "recipe[webserver]"
```

Lab 6

1. List nodes we know about.

```
$ knife node list --local-mode
```

2. Show node information about the node.

```
$ knife node show $(hostname -f) --local-mode
```

3. List existing node objects.

```
$ ls ~/chef-repo/nodes
```

4. Look inside the node object.

```
$ less ~/chef-repo/nodes/$(hostname -f).json
```

5. Run ohai on the node.

```
$ ohai | more
```

6. Show a specific attribute with knife.

```
$ knife node show $(hostname -f) --local-mode -a ipaddress
```

7. Show a specific attribute with knife.

```
$ knife node show $(hostname -f) --local-mode -a
cpu.0.model_name
```

8. Show a specific attribute with knife.

```
$ knife node show $(hostname -f) --local-mode -a name
```

9. Update index.html.erb template to include node name.

```
FILE: ~/chef-
repo/cookbooks/webserver/templates/default/index.html.erb
<html>
 <body>
 <h1>Hello from <%= node.name %></h1>
 </body>
</html>
```

10. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r "recipe[webserver]"
```

11. Verify that we have an updated index.html.

```
$ curl http://localhost
```

END OF MODULE

Applying policy to new EC2 instances

Documentation:
https://github.com/chef/chef-provisioning-aws
Check out the docs/examples section

Provisioning:
https://docs.chef.io/provisioning.html

Update your bash profile with AWS credentials.

~/.aws/config

```
aws_access_key_id = AKIAAABBCCDDEEFFGGHH
aws_secret_access_key = Abc0123dEf4GhIjk5lMn/OpQrSTUvXyz/A678bCD
```

NOTE: Fill in values of AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY as defined in link.

Pull real current values for IAM keys from http://bit.ly/popup-chef-creds

Lab 7

1. Create a recipe called provision in our webserver cookbook.

```
$ chef generate recipe cookbooks/webserver provision
```

2. Update the provision recipe with the following content.
(Reference the full recipe http://bit.ly/provision1)

FILE: ~/chef-repo/cookbooks/webserver/recipes/provision.rb

```
name = "YOURNAME"
```

CHANGE THIS VALUE TO YOUR NAME!

```
require "chef/provisioning/aws_driver"
with_driver "aws::us-east-1" do

  aws_security_group "#{name}-ssh" do
    inbound_rules '0.0.0.0/0' => 22
  end

  aws_security_group "#{name}-http" do
    inbound_rules '0.0.0.0/0' => 80
  end

  with_machine_options({
    :aws_tags => {"belongs_to" => name},
    :ssh_username => "root",
    :bootstrap_options => {
      :image_id => "ami-bf5021d6",
      :instance_type => "t1.micro",
      :key_name => "aws-popup-chef",
      :security_group_ids => ["#{name}-ssh","#{name}-http"]
    },
    :convergence_options => {
      :chef_version => "12.2.1",
    }
  })

  machine "#{name}-webserver-1" do
    recipe "webserver"
    tag "my-webserver"
    converge true
  end
end
```

3. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r 'recipe[webserver::provision]'
```

4. Find your new node's IP.

```
$ knife node show YOURNAME-webserver-1 --local-mode -a
ec2.public_ipv4
```

5. Verify your new node running Apache.

```
$ curl http://NEW_NODE_IP
```

Lab 8

1. Add the following to the bottom of your current provision recipe.

FILE: ~/chef-repo/cookbooks/webserver/recipes/provision.rb

MORE RECIPE ABOVE THIS

```
machine "#{name}-webserver-2" do
  recipe "webserver"
  tag "my-webserver"
  converge true
end

machine "#{name}-webserver-3" do
  recipe "webserver"
  tag "my-webserver"
  converge true
end
```

2. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r 'recipe[webserver::provision]'
```

3. Replace the 3 machine resources in our provision recipe with a loop.

FILE: ~/chef-repo/cookbooks/webserver/recipes/provision.rb

MORE RECIPE ABOVE THIS

```
1.upto(3) do |n|
  instance = "#{name}-webserver-#{n}"
  machine instance do
    recipe "webserver"
    tag "my-webserver"
    converge true
  end
end
```

4. Update the provision recipe to use *machine_batch* resource.
   (Reference the entire recipe: http://bit.ly/provision2)

FILE: ~/chef-repo/cookbooks/webserver/recipes/provision.rb

MORE RECIPE ABOVE THIS

```
machine_batch do
  1.upto(3) do |n|
    instance = "#{name}-webserver-#{n}"
    machine instance do
      recipe "webserver"
      tag "my-webserver"
      converge true
    end
  end
end
```

5. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r 'recipe[webserver::provision]'
```

________________________END OF MODULE________________________

Policy for other AWS Services

Lab 9

1. Update the provision recipe.

(Reference the full recipe http://bit.ly/provision3)

FILE: ~/chef-repo/cookbooks/webserver/recipes/provision.rb

MORE RECIPE ABOVE THIS

```
  # track all the instances we need to make
  webservers = 1.upto(3).map {|n| "#{name}-webserver-#{n}"}

  machine_batch do
    webservers.each do |instance|
      machine instance do
        recipe "webserver"
        tag "my-webserver"
        converge true
       end
     end
   end

  load_balancer "#{name}-webserver-lb" do
    machines webservers
  end
end
```

Lab 10

1. Update the default recipe in the webserver cookbook.

FILE: ~/chef-repo/cookbooks/webserver/recipes/default.rb

```
package "httpd"

service "httpd" do
 action [ :enable, :start ]
end

template "/var/www/html/index.html" do
  source "index.html.erb"
  variables({
    :webservers => search(:node, "tags:my-webserver AND NOT name:#{node.name}")
  })
end

service "iptables" do
  action [ :disable, :stop ]
end
```

2. Update the index.html template in the webserver cookbook.
   (Reference the full recipe http://bit.ly/lbmachines)

~/chef-repo/cookbooks/webserver/templates/default/index.html.erb

```
<html>
 <body>
 <h1>Hello from <%= node.name %></h1>
   <p>The other webservers on this load balancer are:
     <ul>
     <% @webservers.each do |server| %>
     <li><%= server.name %></li>
     <% end %>
     </ul>
   </p>
 </body>
</html>
```

3. Apply our recipe to the node using chef-client.

```
$ chef-client --local-mode -r 'recipe[webserver::provision]'
```

____________________________END OF MODULE____________________________

Healing your Infrastructure

Lab 11

1. Create a recipe called destroy in our webserver cookbook.

```
$ chef generate recipe cookbooks/webserver destroy
```

2. Update the destroy recipe

FILE: ~/chef-repo/cookbooks/webserver/recipes/destroy.rb

```
name = "YOURNAME"

require 'chef/provisioning/aws_driver'
with_driver 'aws::us-east-1'

machine_batch do
  action :destroy
  machines 2.upto(3).map { |n| "#{name}-webserver-#{n}" }
end
```

3. Destroy your web servers

```
$ chef-client --local-mode -r 'recipe[webserver::destroy]'
```

4. List all current nodes via knife

```
$ knife node list --local-mode
```

5. List existing node objects

```
$ ls ~/chef-repo/nodes
```

6. Repair your infrastructure

```
$ chef-client --local-mode -r 'recipe[webserver::provision]'
```

Lab 12

1. Modify your destroy recipe to destroy the 3 webservers, the load balancer, and the security groups

(Reference the full recipe http://bit.ly/machine_destroy2)

FILE: ~/chef-repo/cookbooks/webserver/recipes/destroy.rb

MORE RECIPE ABOVE THIS

```
machine_batch do
  action :destroy
  machines 1.upto(3).map { |n| "#{name}-webserver-#{n}" }
end

load_balancer "#{name}-webserver-lb" do
 action :destroy
end

aws_security_group "#{name}-ssh" do
 action :destroy
end

aws_security_group "#{name}-http" do
 action :destroy
end
```

2. Cleanup your work.

```
$ chef-client --local-mode -r 'recipe[webserver::destroy]'
```