Cooking on the Cloud

•••

An Immutable experience

Ohai!

(Provisioning) Infrastructure

Context is everything!

Moar Context!

What we knew about Hosting!

What we knew about Deploying!

What we knew about Systems!

Tons of Options out there!

Terraform

PROS:

- Fast development pace
- Ability to preview actions

CONS:

- Not Codeable
- Break hard on state changes

Cloudformation

PROS:

- Major AWS support

CONS:

- Not a DSL

Chef-Provisioning (metal was such a cooler name)

PROS:

- Chef integration
- Codeable (Ruby everywhere!)

CONS:

- Coverage of AWS resources
- Slow development pace (at times)

What we realised about state!

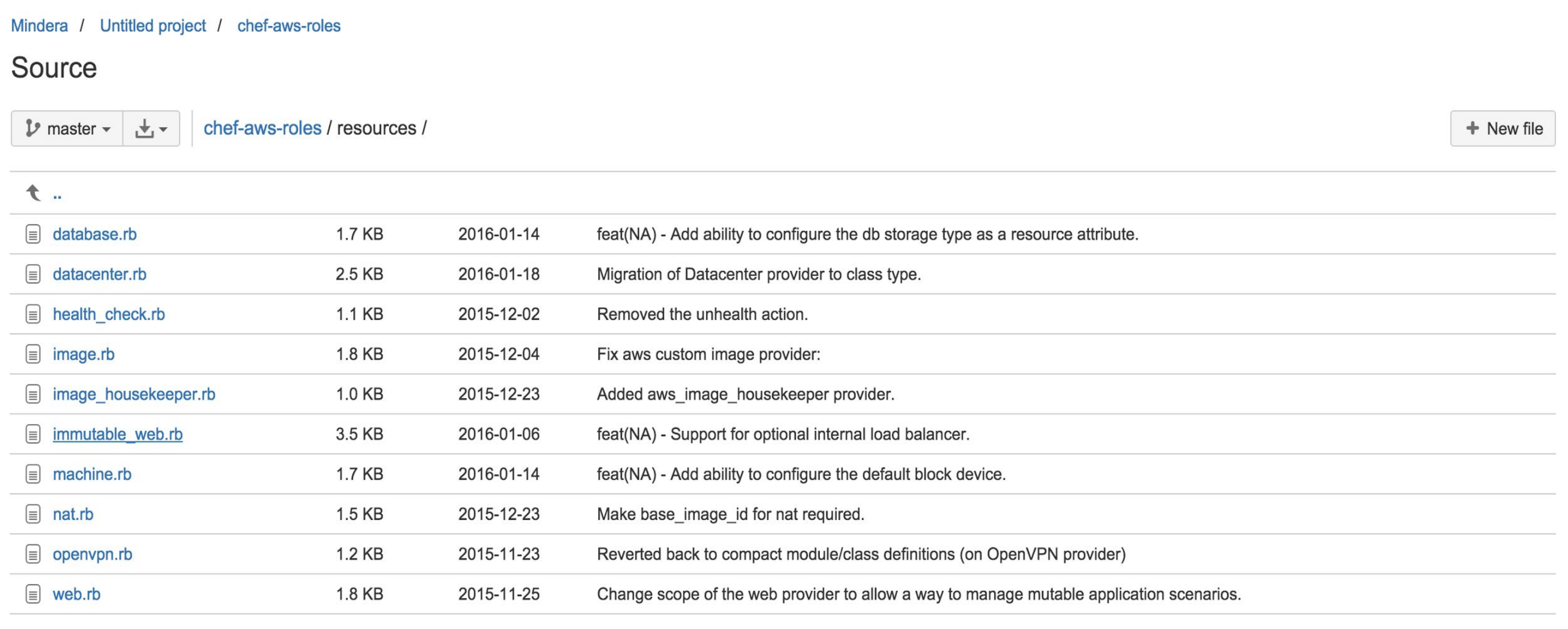
Bet on Chef!

(Provisioning)
With Chef

Concepts on the table!

A whole lot of things!

...Roles! (not the ones you're used to)



Mindera / Untitled project / chef-aws-roles

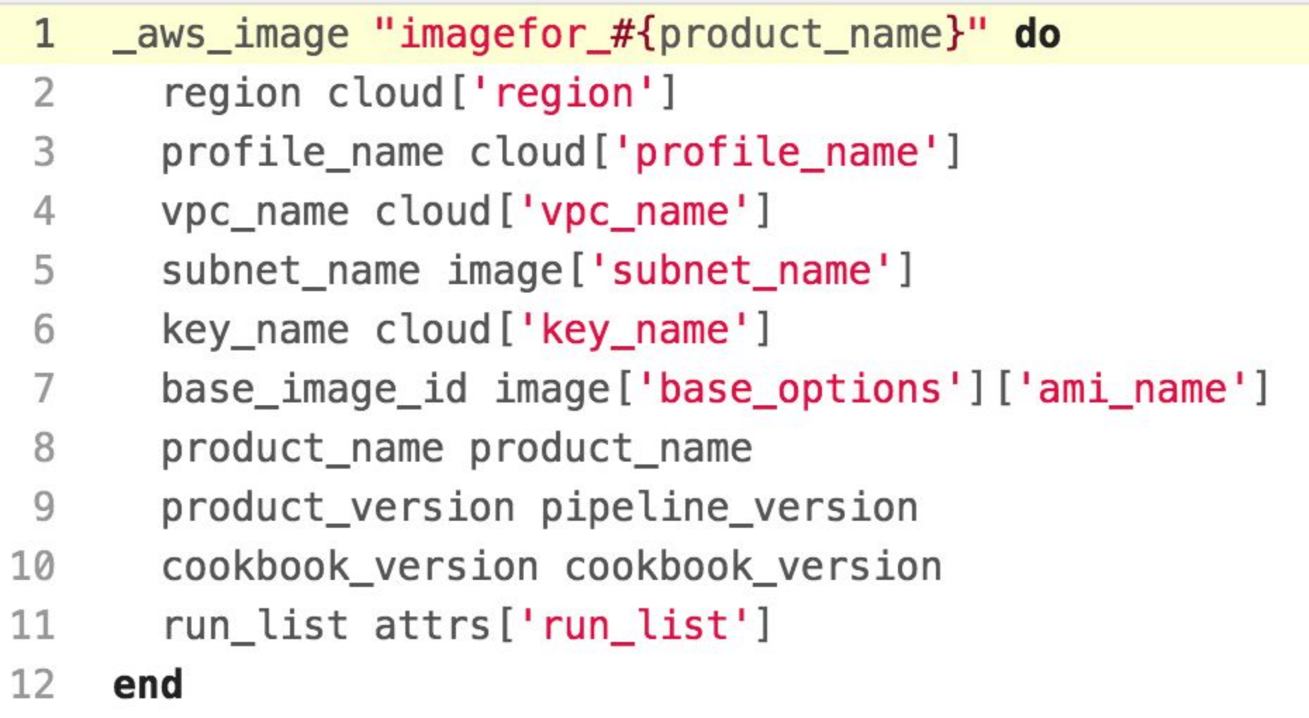
Source

chef-aws-roles / resources /

..

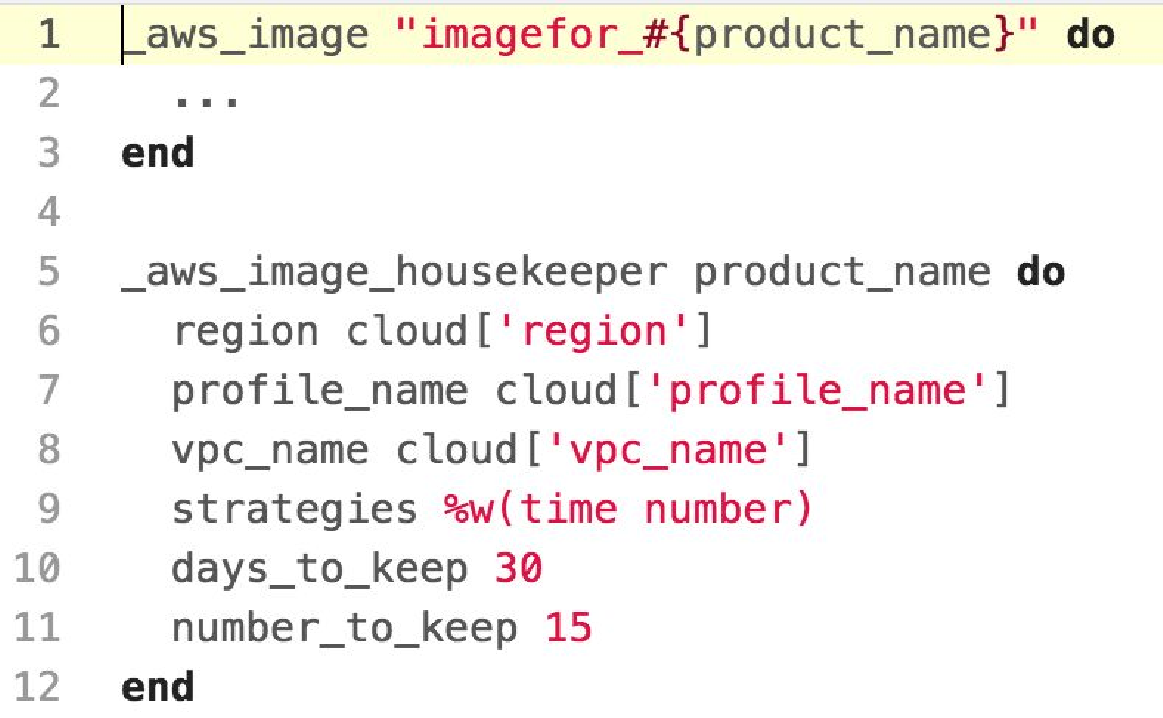
database.rb	1.7 KB	2016-01-14	feat(NA) - Add ability to configure the db storage type as a resource attribute.
datacenter.rb	2.5 KB	2016-01-18	Migration of Datacenter provider to class type.
health_check.rb	1.1 KB	2015-12-02	Removed the unhealth action.
image.rb	1.8 KB	2015-12-04	Fix aws custom image provider:
image_housekeeper.rb	1.0 KB	2015-12-23	Added aws_image_housekeeper provider.
immutable_web.rb	3.5 KB	2016-01-06	feat(NA) - Support for optional internal load balancer.
machine.rb	1.7 KB	2016-01-14	feat(NA) - Add ability to configure the default block device.
nat.rb	1.5 KB	2015-12-23	Make base_image_id for nat required.
openvpn.rb	1.2 KB	2015-11-23	Reverted back to compact module/class definitions (on OpenVPN provider)
web.rb	1.8 KB	2015-11-25	Change scope of the web provider to allow a way to manage mutable application scenarios.

Baking an AMI!



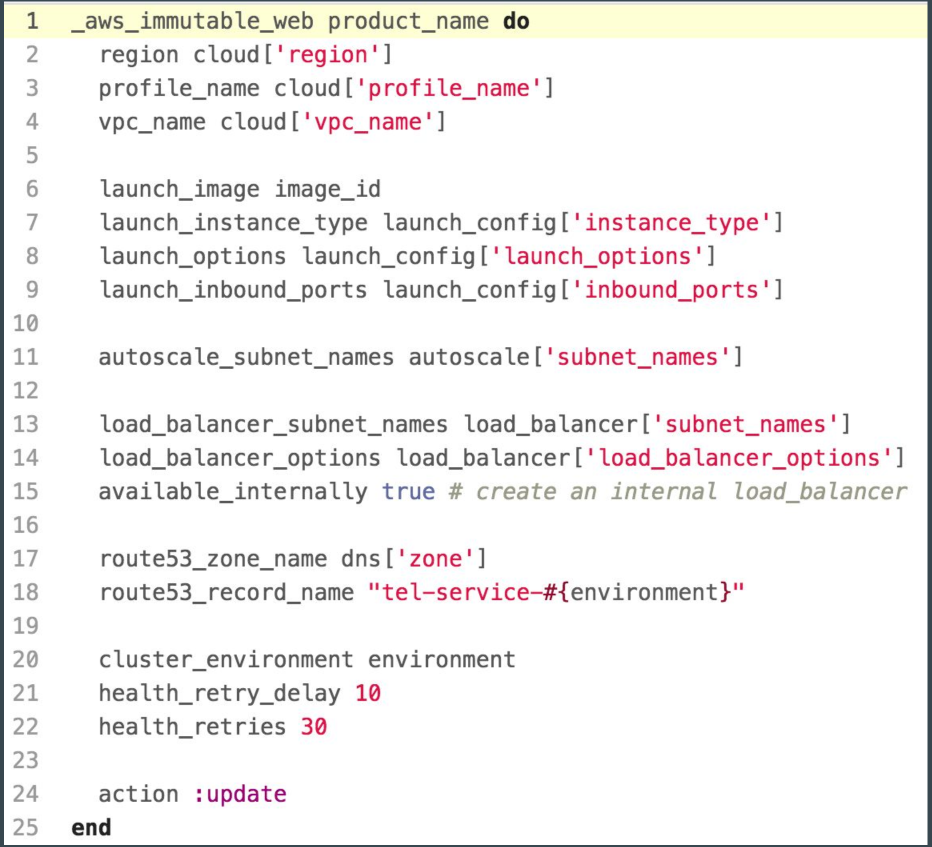
```
_aws_image "imagefor_#{product_name}" do
  region cloud['region']
  profile_name cloud['profile_name']
  vpc_name cloud['vpc_name']
  subnet_name image['subnet_name']
  key_name cloud['key_name']
  base_image_id image['base_options']['ami_name']
  product_name product_name
  product_version pipeline_version
  cookbook_version cookbook_version
  run_list attrs['run_list']
end
```

Cleaning up an AMI!



```
_aws_image "imagefor_#{product_name}" do
  ...
end

_aws_image_housekeeper product_name do
  region cloud['region']
  profile_name cloud['profile_name']
  vpc_name cloud['vpc_name']
  strategies %w(time number)
  days_to_keep 30
  number_to_keep 15
end
```

Spin up a Web Service!



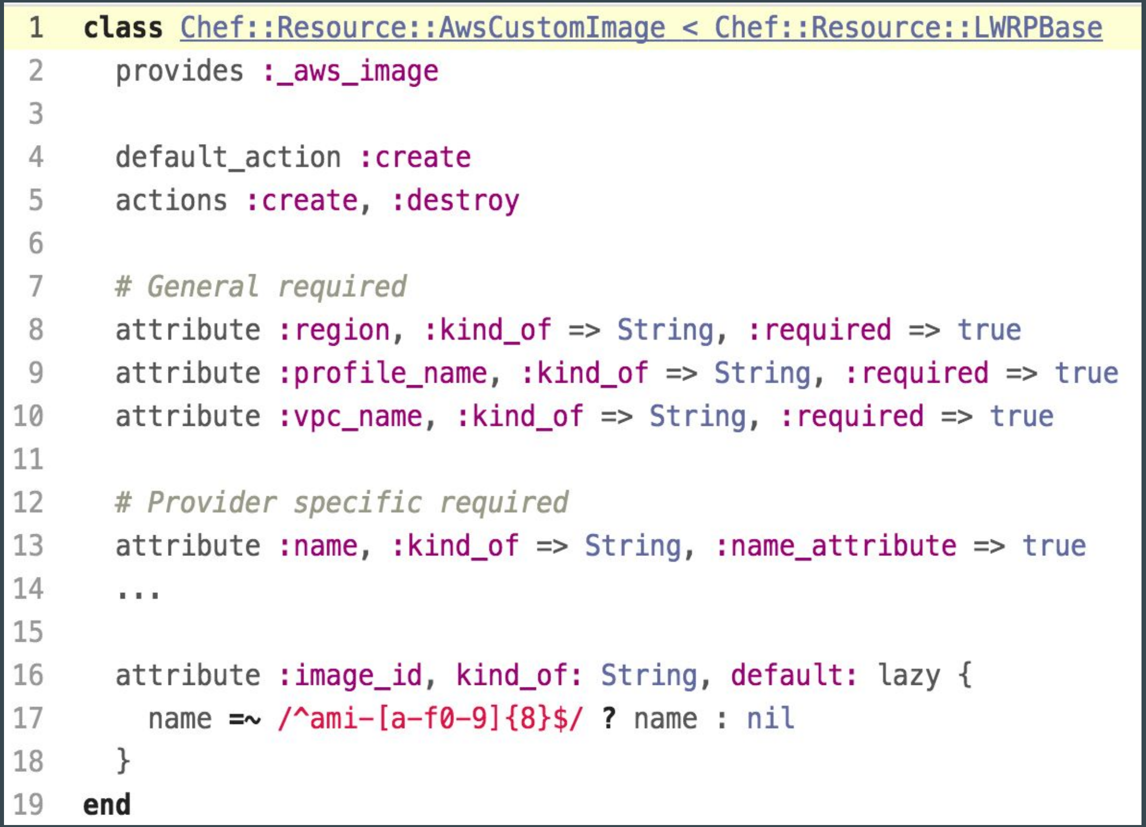
```
_aws_immutable_web product_name do
  region cloud['region']
  profile_name cloud['profile_name']
  vpc_name cloud['vpc_name']

  launch_image image_id
  launch_instance_type launch_config['instance_type']
  launch_options launch_config['launch_options']
  launch_inbound_ports launch_config['inbound_ports']

  autoscale_subnet_names autoscale['subnet_names']

  load_balancer_subnet_names load_balancer['subnet_names']
  load_balancer_options load_balancer['load_balancer_options']
  available_internally true # create an internal load_balancer

  route53_zone_name dns['zone']
  route53_record_name "tel-service-#{environment}"

  cluster_environment environment
  health_retry_delay 10
  health_retries 30

  action :update
end
```

(Provisioning) With Jenkins

Let’s kick one of our pipelines!

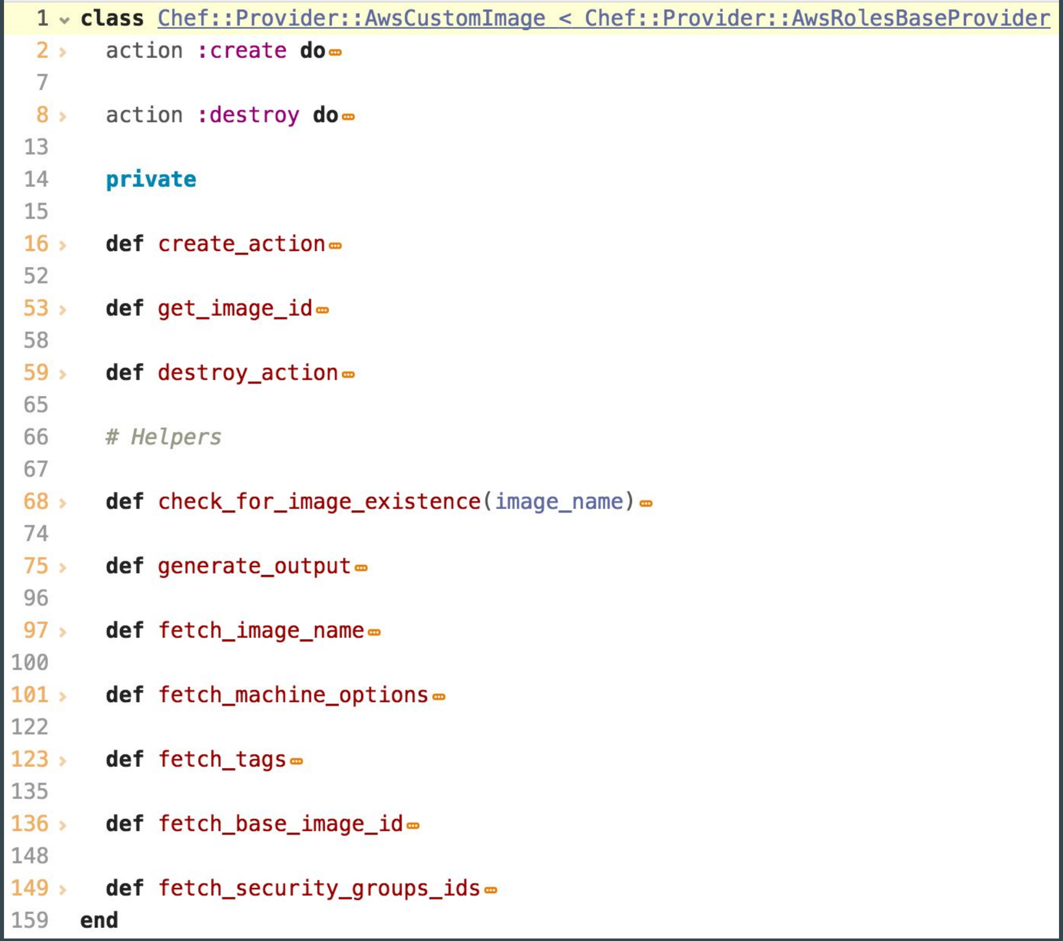
(Provisioning)
The code

The image resource!



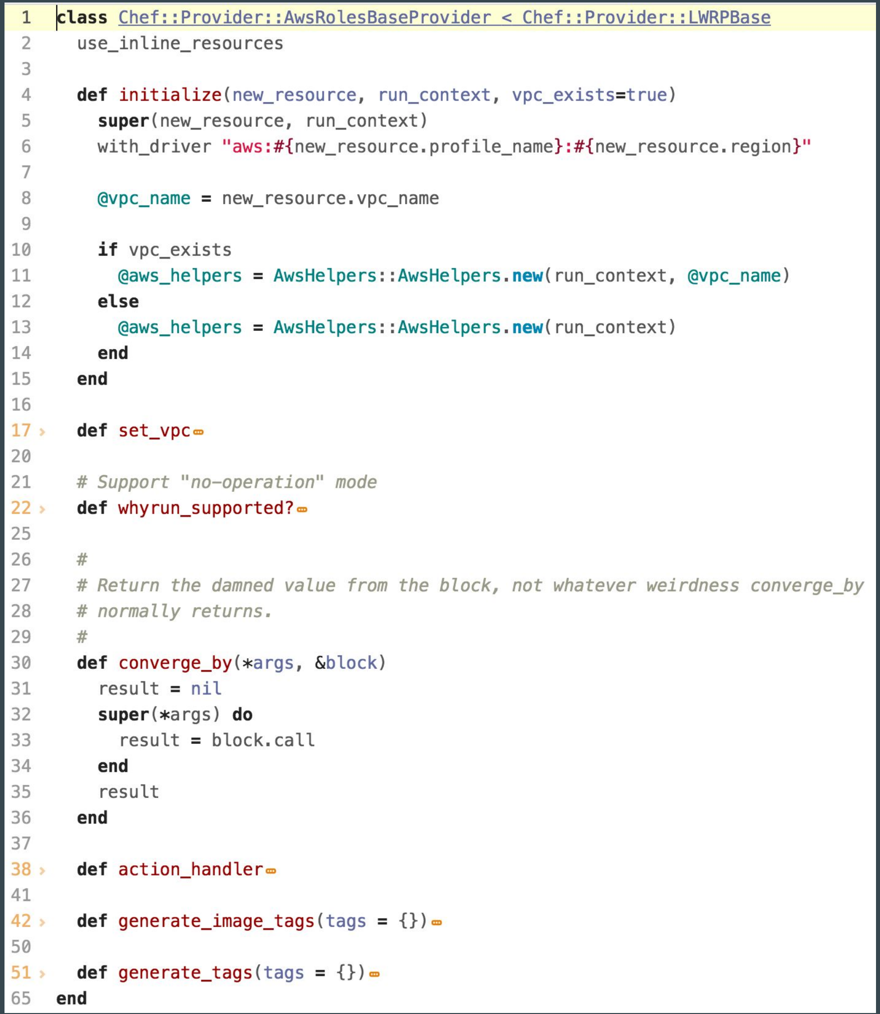
```ruby
class Chef::Resource::AwsCustomImage < Chef::Resource::LWRPBase
  provides :_aws_image

  default_action :create
  actions :create, :destroy

  # General required
  attribute :region, :kind_of => String, :required => true
  attribute :profile_name, :kind_of => String, :required => true
  attribute :vpc_name, :kind_of => String, :required => true

  # Provider specific required
  attribute :name, :kind_of => String, :name_attribute => true
  ...

  attribute :image_id, kind_of: String, default: lazy {
    name =~ /^ami-[a-f0-9]{8}$/ ? name : nil
  }
end
```

The image provider!



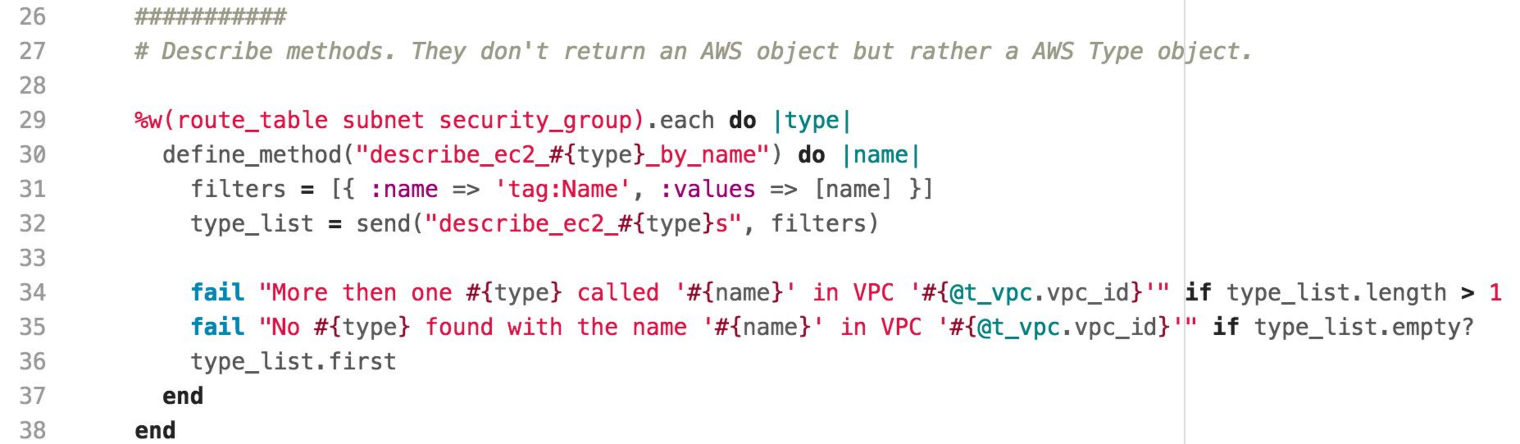
```ruby
class Chef::Provider::AwsCustomImage < Chef::Provider::AwsRolesBaseProvider
  action :create do

  action :destroy do

  private

  def create_action

  def get_image_id

  def destroy_action

  # Helpers

  def check_for_image_existence(image_name)

  def generate_output

  def fetch_image_name

  def fetch_machine_options

  def fetch_tags

  def fetch_base_image_id

  def fetch_security_groups_ids
end
```

AwsRolesBaseProvider!



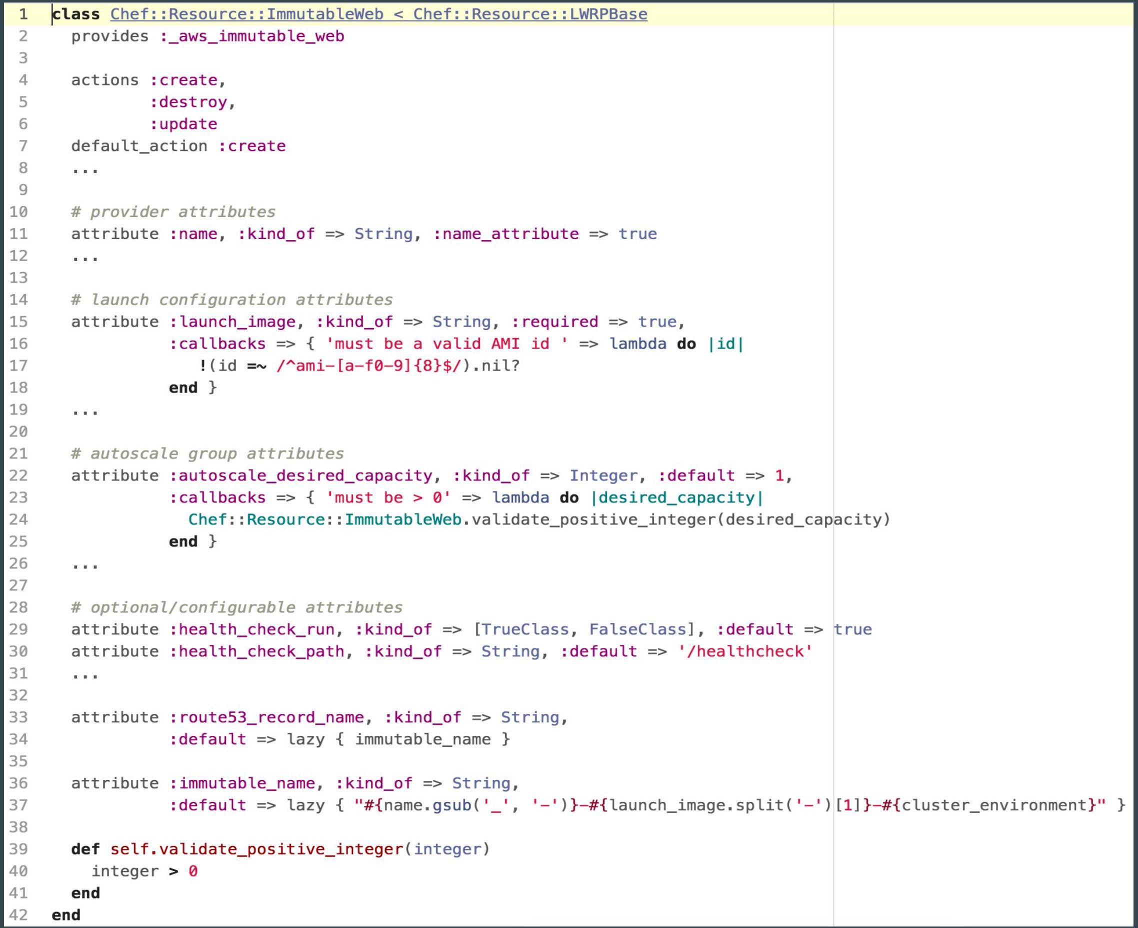
```ruby
class Chef::Provider::AwsRolesBaseProvider < Chef::Provider::LWRPBase
  use_inline_resources

  def initialize(new_resource, run_context, vpc_exists=true)
    super(new_resource, run_context)
    with_driver "aws:#{new_resource.profile_name}:#{new_resource.region}"

    @vpc_name = new_resource.vpc_name

    if vpc_exists
      @aws_helpers = AwsHelpers::AwsHelpers.new(run_context, @vpc_name)
    else
      @aws_helpers = AwsHelpers::AwsHelpers.new(run_context)
    end
  end

  def set_vpc

  # Support "no-operation" mode
  def whyrun_supported?

  #
  # Return the damned value from the block, not whatever weirdness converge_by
  # normally returns.
  #
  def converge_by(*args, &block)
    result = nil
    super(*args) do
      result = block.call
    end
    result
  end

  def action_handler

  def generate_image_tags(tags = {})

  def generate_tags(tags = {})
end
```

AwsHelpers!



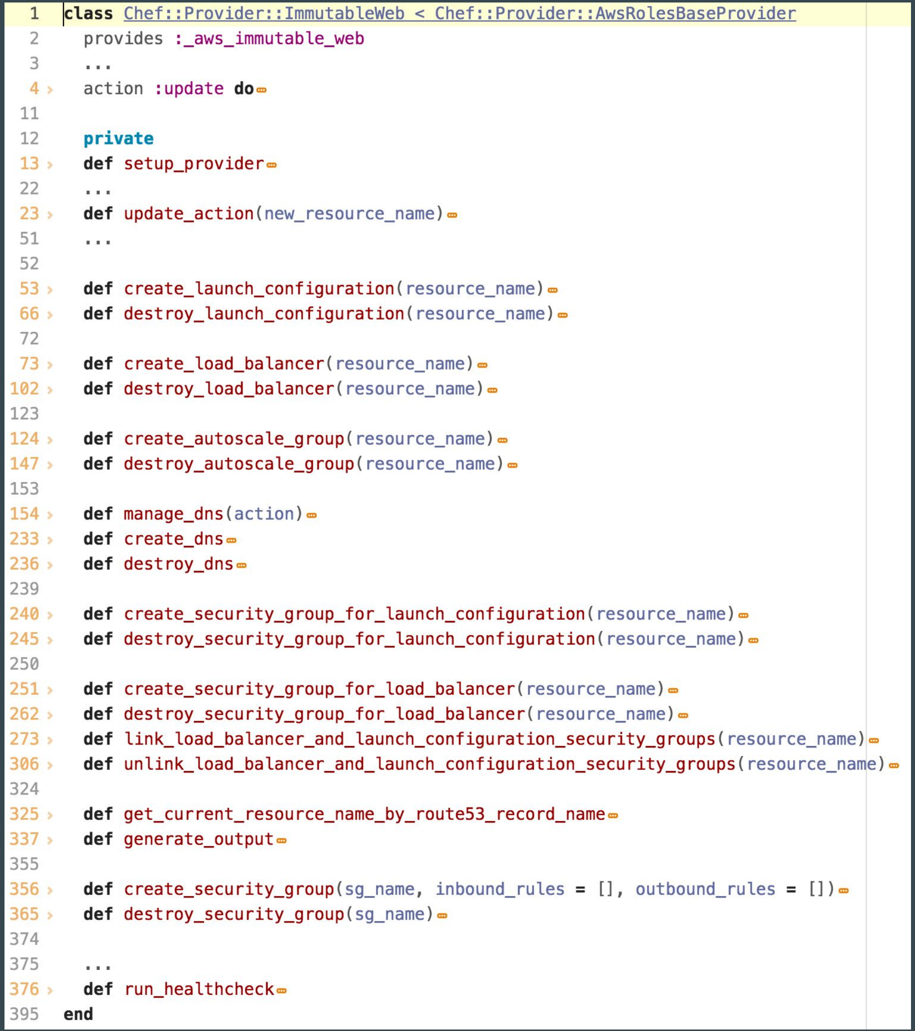
```
###########
# Describe methods. They don't return an AWS object but rather a AWS Type object.

%w(route_table subnet security_group).each do |type|
  define_method("describe_ec2_#{type}_by_name") do |name|
    filters = [{ :name => 'tag:Name', :values => [name] }]
    type_list = send("describe_ec2_#{type}s", filters)

    fail "More then one #{type} called '#{name}' in VPC '#{@t_vpc.vpc_id}'" if type_list.length > 1
    fail "No #{type} found with the name '#{name}' in VPC '#{@t_vpc.vpc_id}'" if type_list.empty?
    type_list.first
  end
end
```

The immutable_web resource!



```
class Chef::Resource::ImmutableWeb < Chef::Resource::LWRPBase
  provides :_aws_immutable_web

  actions :create,
          :destroy,
          :update
  default_action :create
  ...

  # provider attributes
  attribute :name, :kind_of => String, :name_attribute => true
  ...

  # launch configuration attributes
  attribute :launch_image, :kind_of => String, :required => true,
            :callbacks => { 'must be a valid AMI id ' => lambda do |id|
               !(id =~ /^ami-[a-f0-9]{8}$/).nil?
            end }
  ...

  # autoscale group attributes
  attribute :autoscale_desired_capacity, :kind_of => Integer, :default => 1,
            :callbacks => { 'must be > 0' => lambda do |desired_capacity|
              Chef::Resource::ImmutableWeb.validate_positive_integer(desired_capacity)
            end }
  ...

  # optional/configurable attributes
  attribute :health_check_run, :kind_of => [TrueClass, FalseClass], :default => true
  attribute :health_check_path, :kind_of => String, :default => '/healthcheck'
  ...

  attribute :route53_record_name, :kind_of => String,
            :default => lazy { immutable_name }

  attribute :immutable_name, :kind_of => String,
            :default => lazy { "#{name.gsub('_', '-')}-#{launch_image.split('-')[1]}-#{cluster_environment}" }

  def self.validate_positive_integer(integer)
    integer > 0
  end
end
```

The immutable_web provider!



```ruby
class Chef::Provider::ImmutableWeb < Chef::Provider::AwsRolesBaseProvider
  provides :_aws_immutable_web
  ...
  action :update do

  private
  def setup_provider
  ...
  def update_action(new_resource_name)
  ...

  def create_launch_configuration(resource_name)
  def destroy_launch_configuration(resource_name)

  def create_load_balancer(resource_name)
  def destroy_load_balancer(resource_name)

  def create_autoscale_group(resource_name)
  def destroy_autoscale_group(resource_name)

  def manage_dns(action)
  def create_dns
  def destroy_dns

  def create_security_group_for_launch_configuration(resource_name)
  def destroy_security_group_for_launch_configuration(resource_name)

  def create_security_group_for_load_balancer(resource_name)
  def destroy_security_group_for_load_balancer(resource_name)
  def link_load_balancer_and_launch_configuration_security_groups(resource_name)
  def unlink_load_balancer_and_launch_configuration_security_groups(resource_name)

  def get_current_resource_name_by_route53_record_name
  def generate_output

  def create_security_group(sg_name, inbound_rules = [], outbound_rules = [])
  def destroy_security_group(sg_name)


  ...
  def run_healthcheck
end
```

(Provisioning) Results

What we did learn over the past year!

(Provisioning)
Done With Jenkins?

@miguelcnf

(Provisioning) Questions?