Approaches to Extending Resources

Objectives

After completing this module, you should be able to:

➢ Describe the difference between:

- Custom Resources
- Definitions
- Heavy-Weight Resource-Providers
- Light-Weight Resource-Providers

CONCEPT

Approaches to Extending Resources

1. Pure Ruby (Heavy-Weight Resource-Providers / HWRP)
2. Definitions
3. Light-Weight Resource-Providers (LWRP)
4. Custom Resources

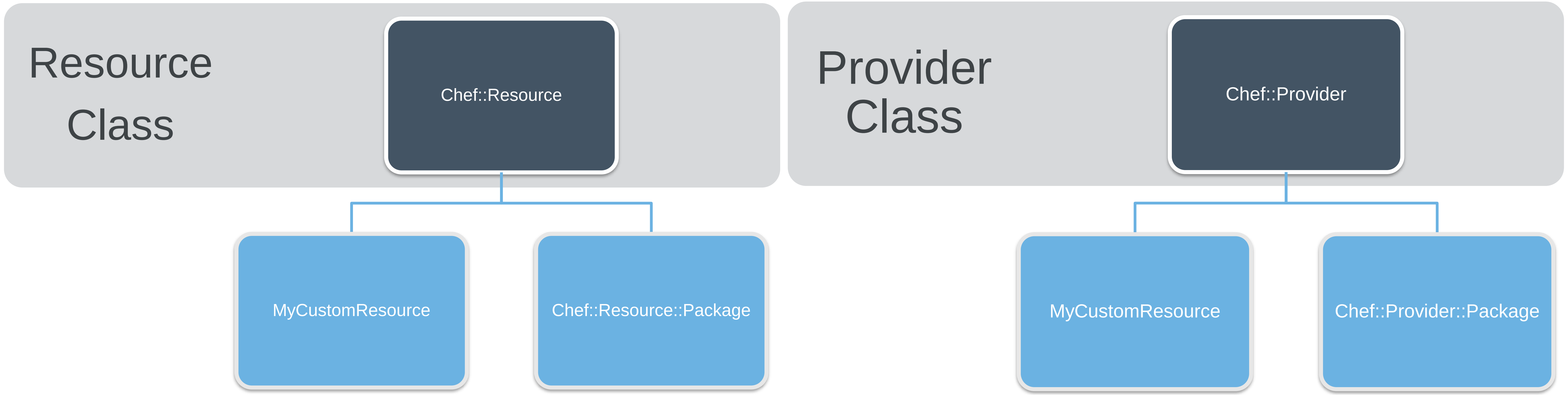
CONCEPT

1 Pure Ruby (Heavy-Weight Resource-Providers / HWRP)

- Description
- File and Folder Structure
- Implementation Language & Usage
- Benefits & Drawbacks

1 Pure Ruby (Heavy-Weight Resource-Providers)

DESCRIPTION



Describes how the resource appears within the recipe (e.g. resource name, properties, supported actions)

Describes how the resource behaves when it takes a supported action on each supported platform

1 Pure Ruby (Heavy-Weight Resource-Providers)

STRUCTURE

my_cookbook

libraries/

- [my_custom_resource]_resource.rb
- [my_custom_resource]_provider.rb

They are stored within the libraries folder in separate files for the resource and the provider. The file names are snake case representations of the class name stored within the file.

CHEF

1 Pure Ruby (Heavy-Weight Resource-Providers)

IMPLEMENTATION LANGUAGE - RESOURCE

`libraries/apache_vhost_resource.rb`

```
class Chef
  class Resource
    class ApacheVhost < Chef::Resource
      def initialize(name, run_context=nil)
        super
        @resource_name = :apache_vhost              # Defining the resource name
        @provider = Chef::Provider::ApacheVhost  # Specifying which Provider to use
        @action = :create                           # Setting the default action
        @allowed_actions = [:create, :remove]     # Setting the list of actions
        # ... SETUP ANY DEFAULT VALUES HERE ...
      end

      def site_name(arg=nil)
        set_or_return(:site_name, arg, :kind_of => String)
      end
    end
  end
end
```

CHEF

1 Pure Ruby (Heavy-Weight Resource-Providers)

IMPLEMENTATION LANGUAGE - PROVIDER

`libraries/apache_vhost_provider.rb`

```ruby
class Chef
  class Provider
    class ApacheVhost < Chef::Provider
      def load_current_resource
        @current_resource ||= Chef::Resource::ApacheVhost.new(new_resource.name)

        @current_resource.site_name(new_resource.site_name)
        # ... remaining properties defined in the resource
        @current_resource
      end

      def action_create
        # ... code that creates the resource on all supported platforms ...
      end
    end
  end
end
```

1 Pure Ruby (Heavy-Weight Resource-Providers)

USAGE

recipes/default.rb

```
apache_vhost 'welcome' do
  action :delete
end

apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

1 Pure Ruby (Heavy-Weight Resource-Providers)

BENEFITS & DRAWBACKS

- Available in some of the earliest versions of Chef
- Allows for extremely flexible and powerful resource implementations
- Requires knowledge of Ruby
- Requires knowledge of Object-Oriented Programming techniques

2 Definitions

DESCRIPTION

recipes/admins_site.rb

```
apache_vhost 'admins' do
  site_name 'admins'
end
```

recipes/users_site.rb

```
apache_vhost 'users' do
  site_name 'users'
end
```

recipes/dogs_site.rb

```
...
```

definitions/apache_vhost.rb

```
define :apache_vhost site_name: 'default' do

  directory ...

  template ...

  file ...

end
```

Definitions

STRUCTURE

my_cookbook

definitions/

- [my_definition_name].rb

They are stored within the definitions folder and often the name of the definition defines of the file.

Definitions

IMPLEMENTATION LANGUAGE

definitions/apache_vhost.rb

```
define :apache_vhost site_name: 'default', site_port: 80 do
  directory "/srv/apache/#{params[:site_name]}/html' do
    recursive true
    mode '0755'
  end

  templates "/srv/apache/#{params[:site_name]}/html" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{params[:site_name]}/html", port: params[:site_port]
    mode '0755'
    notifies :restart, 'service[httpd]'
  end

  # ... remaining resources ...
end
```

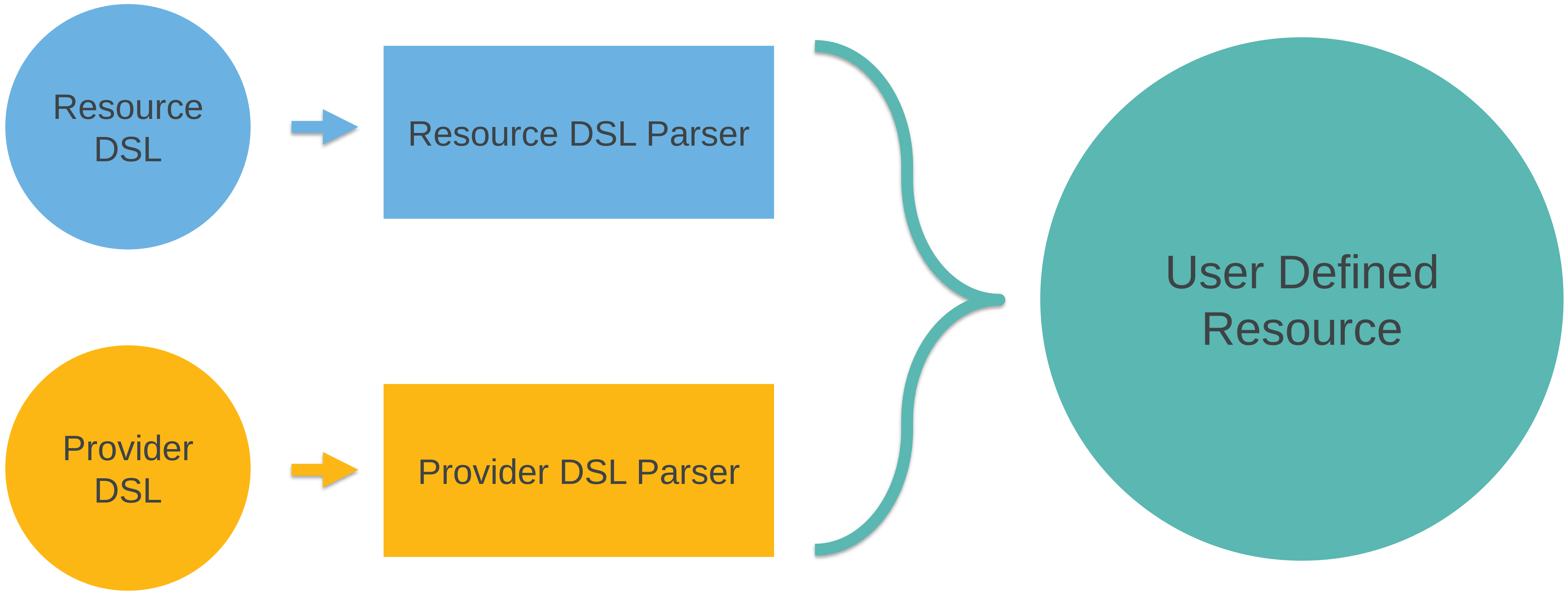
2 Definitions

BENEFITS & DRAWBACKS

- Available in some of the earliest versions of Chef
- Allows for code re-use within recipes
- Definition usage could be mistaken for a true resource
- Definitions do not support notifications (`subscribes` and `notifies`)

3 Light-Weight Resource-Providers (LWRP)

DESCRIPTION



3 Light-Weight Resource-Providers (LWRP)

STRUCTURE

my_cookbook

```
resources/
• [my_resource_name].rb
providers/
• [my_resource_name].rb
```

An LWRP is defined in two separate files that share the same name. The resource definition is defined in the resources directory of the cookbook; the provider definition in the providers directory.

The cookbook name is combined with the file name to create the name of the resource.

CHEF

3

Light-Weight Resource-Providers (LWRP)

IMPLEMENTATION LANGUAGE - RESOURCE

resources/vhost.rb

```
actions :create, :delete

default_action :create

attribute :site_name, String, name_attribute: true
attribute :site_port, Integer, default: 80
```

3 Light-Weight Resource-Providers (LWRP)

IMPLEMENTATION LANGUAGE - PROVIDER

providers/vhost.rb

```
action :create do
  directory "/srv/apache/#{new_resource.site_name}/html' do
    recursive true
    mode '0755'
  end

  templates "/srv/apache/#{new_resource.site_name}/html" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{new_resource.site_name}/html",
              port: new_resource.site_port]
    mode '0755'
    notifies :restart, 'service[httpd]'
  end

  # ... remaining resources ...
end
```

3

Light-Weight Resource-Providers (LWRP)

USAGE

`recipes/default.rb`

```
apache_vhost 'welcome' do
  action :delete
end

apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

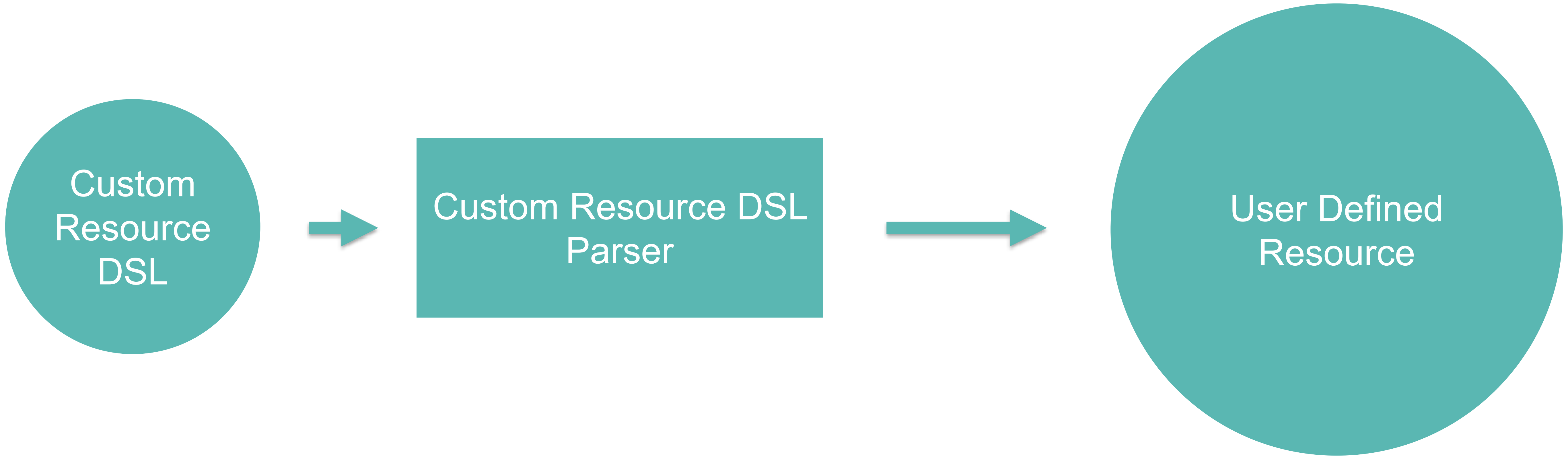
3 Light-Weight Resource-Providers (LWRP)

BENEFITS & DRAWBACKS

- Available in 0.7.12 version of Chef
- Allows for a real resource definition without understanding Ruby (vs. HWRP)
- Resource and provider implementation require learning a new DSL
- Complete resource definition is spread across two files

4 Custom Resources

DESCRIPTION



Custom Resources

my_cookbook

resources/

- [my_resource_name].rb

A custom resource is defined in a single file within the resources directory.

Custom Resources

IMPLEMENTATION LANGUAGE

resources/vhost.rb

```
resource_name :apache_vhost

property :site_name, String, name_attribute: true
property :site_port, Integer, default: 80

action :create do
  directory "/srv/apache/#{new_resource.site_name}/html' do
    recursive true
    mode '0755'
  end

  # ... remaining resources ...

end

# ... remaining actions ...
```

4 Custom Resources

USAGE

recipes/default.rb

```
apache_vhost 'welcome' do
  action :delete
end

apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

Custom Resources

BENEFITS & DRAWBACKS

- Available in 12.5.0 version of Chef
- Allows for a real resource definition without understanding Ruby (vs. HWRP)
- Complete resource definition is defined in a single file (vs. LWRP)
- Custom resource implementation require learning a new DSL

Approaches to Extending Resources

1. Pure Ruby (Heavy-Weight Resource-Providers / HWRP)
2. Definitions
3. Light-Weight Resource-Providers (LWRP)
4. Custom Resources

Discussion

Which approaches require you to define your solution in two separate files?

What are the limitations of choosing the Definitions approach?

What are some differences between LWRP and Custom Resources?

Given a Chef version prior to 12.5.0, which approach would you choose?

DISCUSSION

Q&A

What questions can we answer for you?

CHEF™