Progress

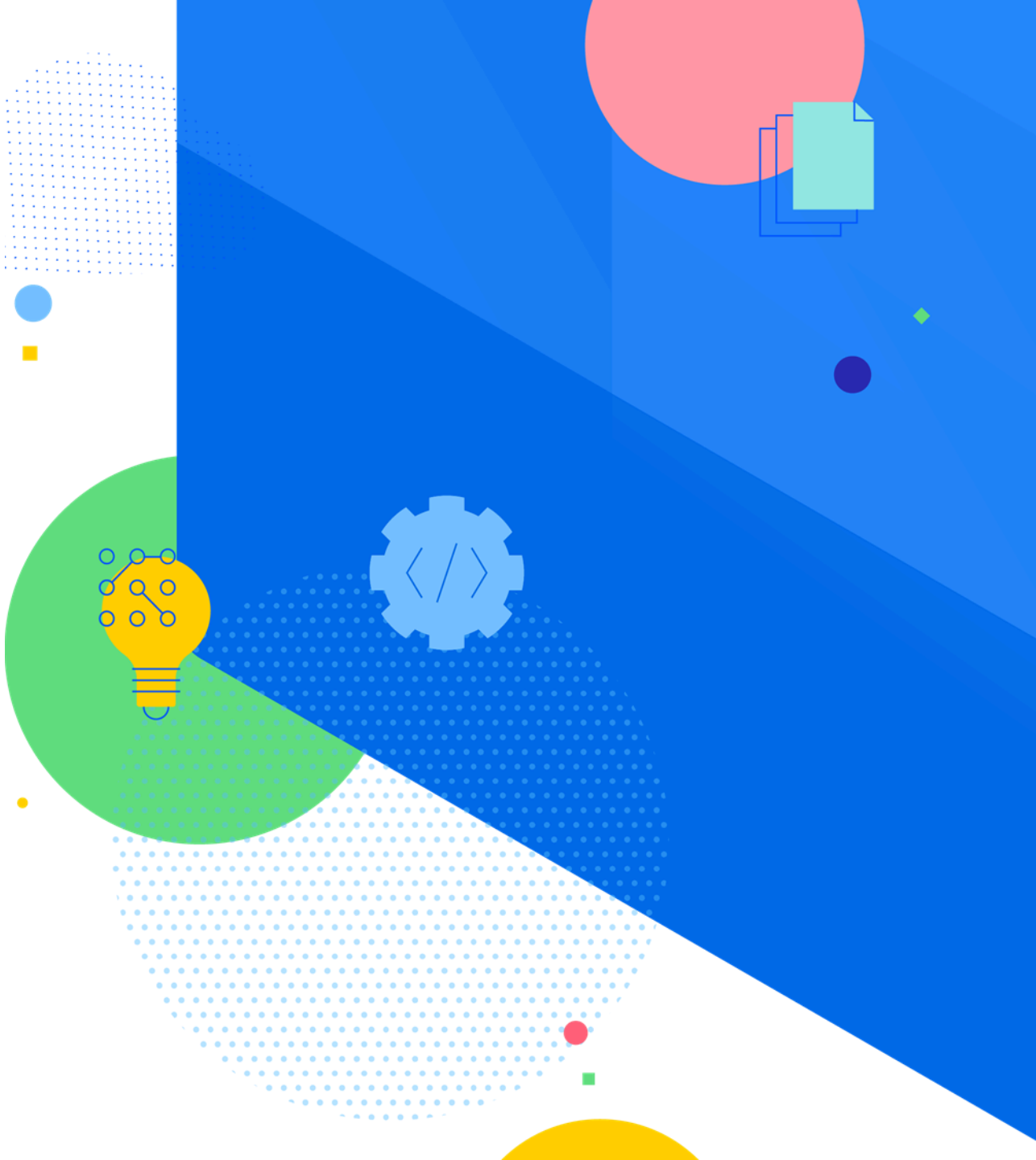
Custom Resources

Maintainable Cookbooks

Best Practices - level 100

Progress®

Objectives



Progress

Terminology

Resource

Progress

Example

```
file '/etc/motd' do
  content 'Welcome to my server!'
  owner 'root'
  group 'root'
  mode '0644'
  action :create
end
```

```
- type: file
  name: /etc/motd
  content: Welcome to my server!
  owner: root
  group: root
  mode: 0644
  action: create
```

Progress

Terminology

Custom Resource

- Composition
- Extension

Progress

Example

```
resource_name :my_custom_resource

property :message, String, default: 'Hello, world!'

action :create do
  file '/tmp/my_custom_file' do
    content new_resource.message
    owner 'root'
    group 'root'
    mode '0644'
    action :create
  end
end
```

Progress

```
resource_name :web_server_setup

property :server_name, String, name_property: true
property :document_root, String, default: '/var/www/html'
property :index_content, String, default: '<h1>Welcome to my web server!</h1>'

action :create do
  # Install the Apache package
  package 'apache2' do
    action :install
  end

  # Hidden... Ensure the Apache service is running and enabled
  # Hidden... Create the document root directory if it doesn't exist

  # Create the index.html file with the specified content
  file "#{new_resource.document_root}/index.html" do
    content new_resource.index_content
    owner 'www-data'
    group 'www-data'
    mode '0644'
    action :create
  end
end
```

Progress

Custom Resources

Benefits

- Encapsulation / Modularity
- Reusability / Readability
- Maintainability
- Idempotency and Convergence
- Better Testing and Debugging
- Easier Chef Client / Cookbook Upgrades

Part 2:

Creating Custom Resources (25 mins)

Progress

Anatomy of a Composite Resource

```
provides :resource_name

property :property_name, RubyType, default: 'value'
default_action :action_a

action :action_a do
    # a mix of built-in Chef Infra resources and Custom Resources
end

action :action_b do
    # a mix of built-in Chef Infra resources and Custom Resources
end
```

Anatomy of a Extension Resource

```
provides :resource_name

property :prop_one, RubyType, default: 'value'

default_action :action_a

load_current_value do |new_resource|
    # Use the load_current_value method to load the specified property values
end

action :action_a do
    converge_if_changed :prop_one do
        # code if prop_one has changed
    end
end

action :action_b do
    converge_if_changed :prop_one do
        # code if any propery has changed
    end
end
```

Progress

Best Practices in Custom Resource Development

- Naming Conventions
- Idempotency
- Current State Management
- Documentation
- Unit Testing / InSpec Tests
- Linting

Progress

Part 3:

Hands-On-Labs

- Writing your first custom (composite) resource
- Writing a recipie that uses that resource
- Testing your resource