Dissecting Your First chef-client Run

The Anatomy of a Chef run

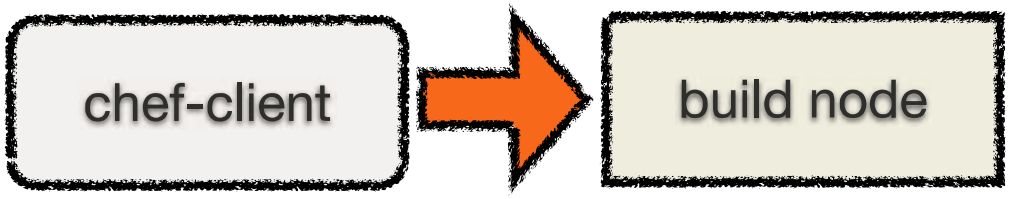
v2.1.6

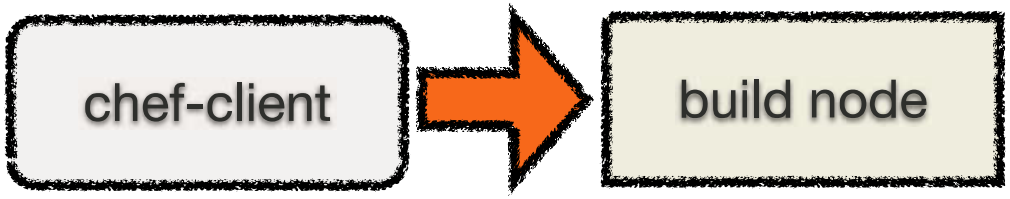
CHEF

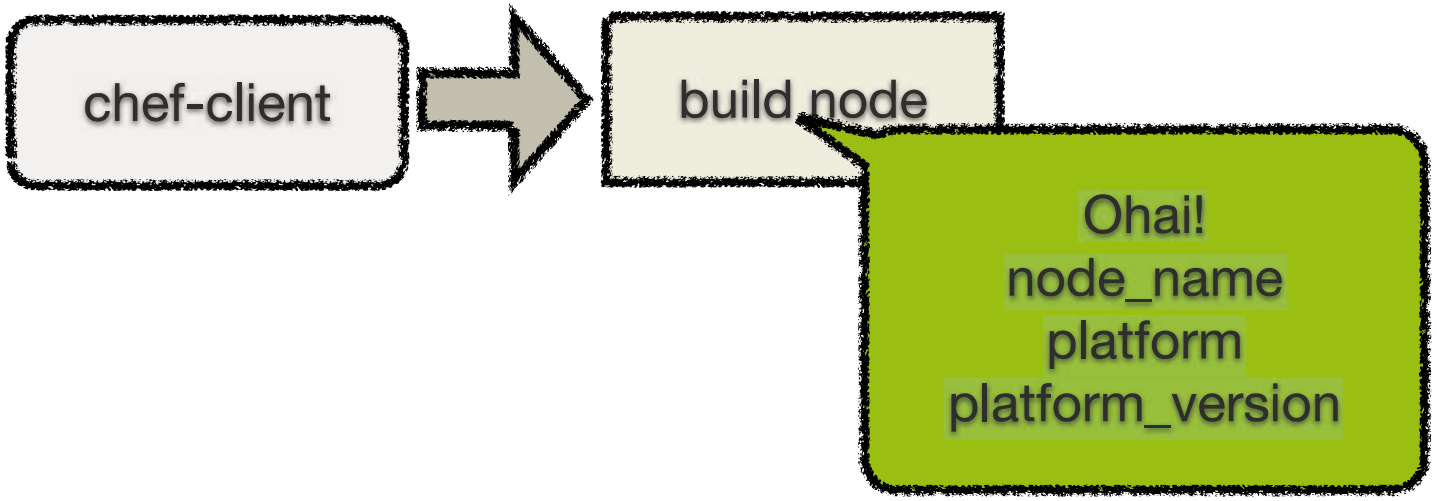
Lesson Objectives

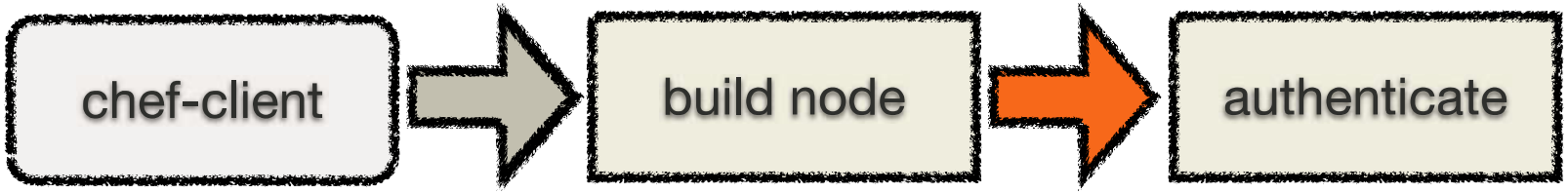
- After completing the lesson, you will be able to
  - List all the steps taken by a chef-client during a run
  - Explain the basic security model of Chef
  - Explain the concepts of the Resource Collection

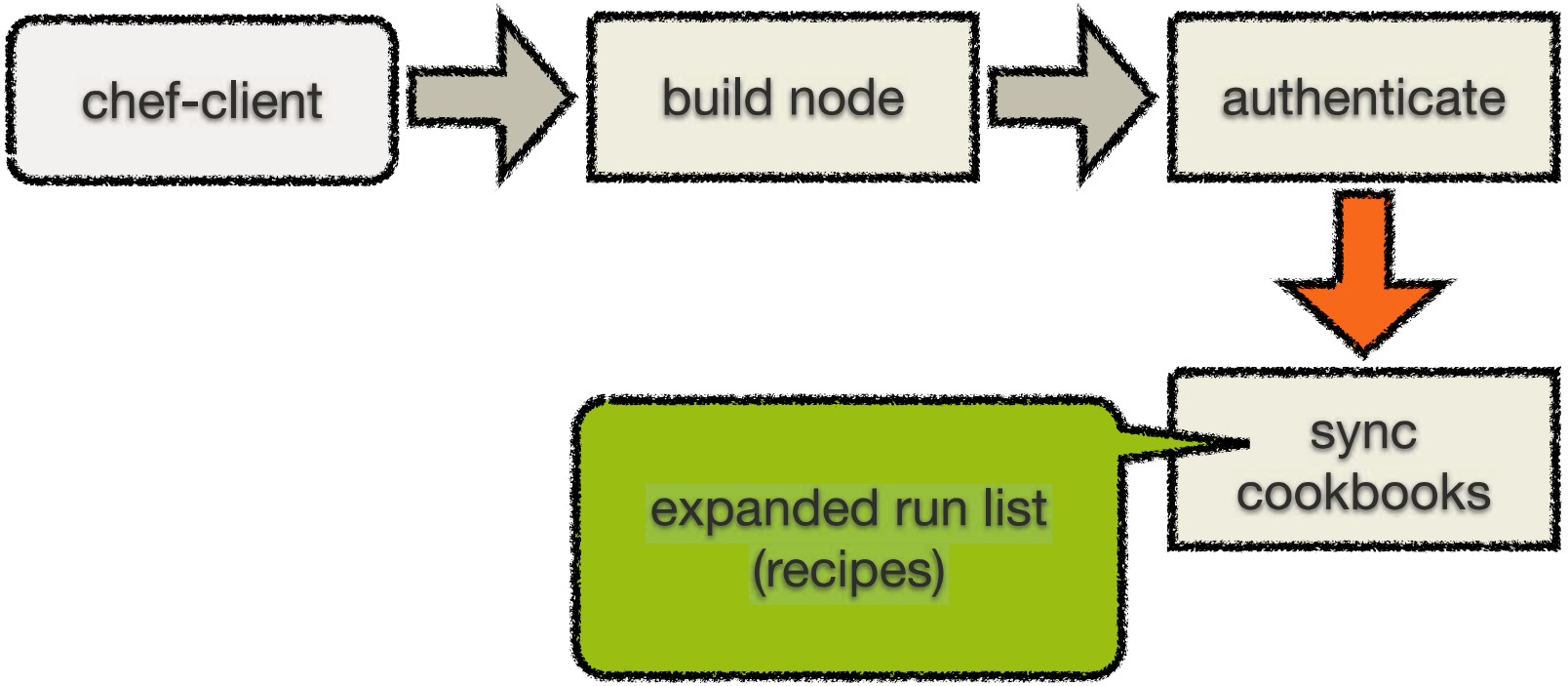

chef-client

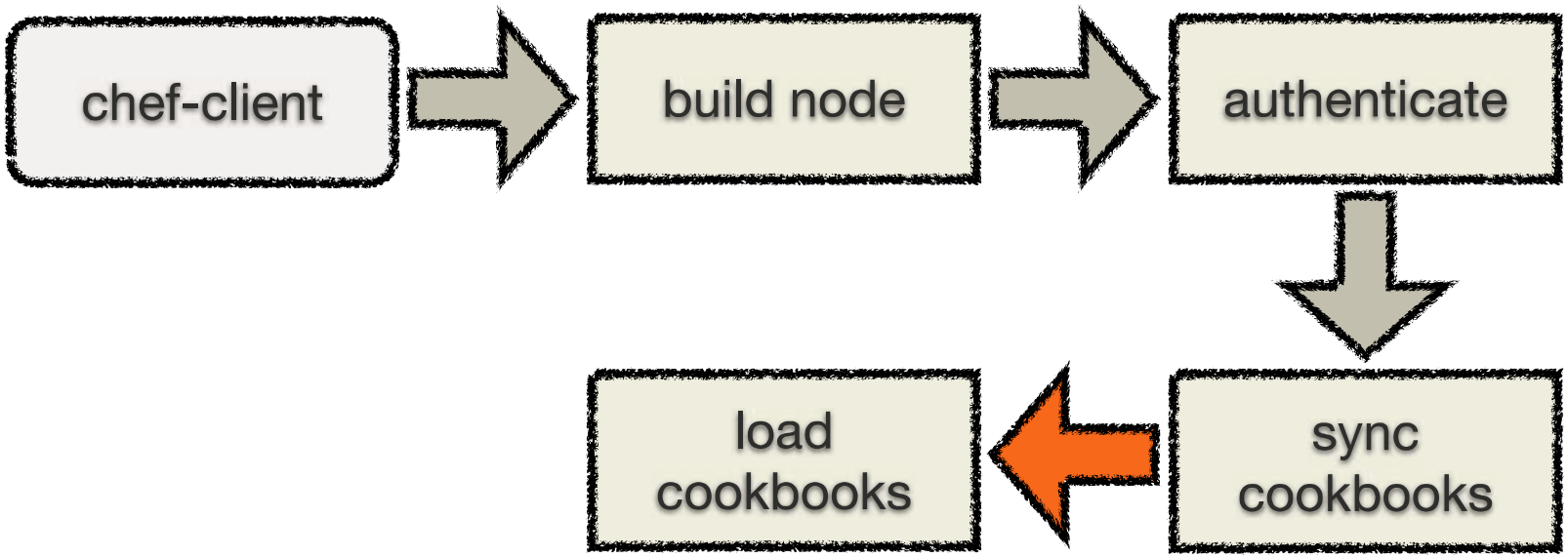

chef-client
build node

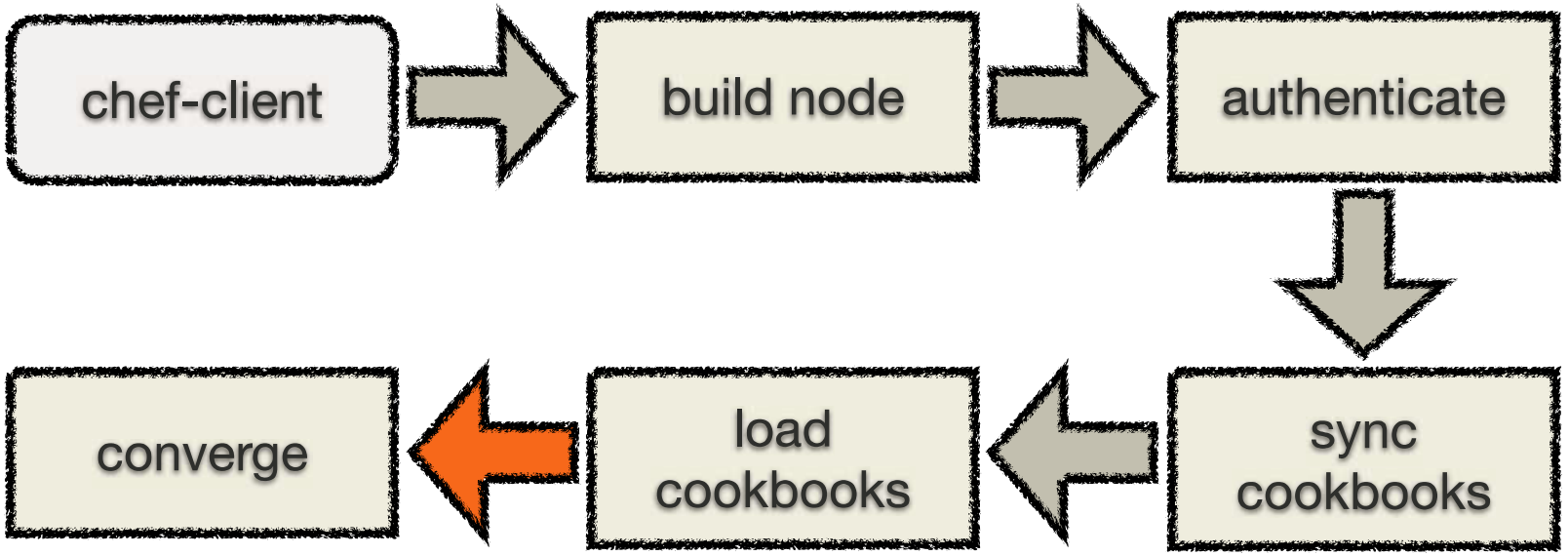

chef-client
build node

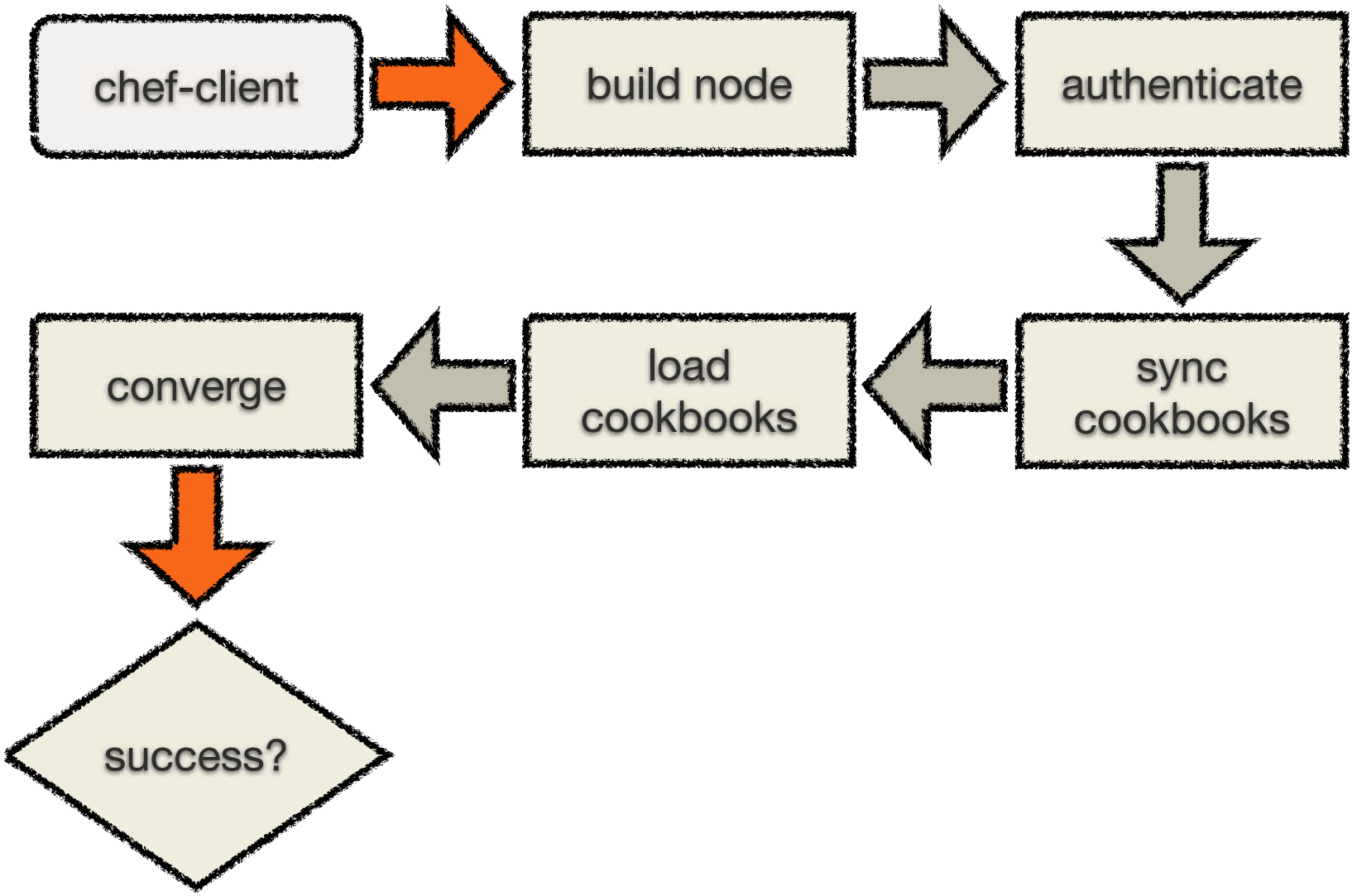

chef-client
build node
Ohai!
node_name
platform
platform_version

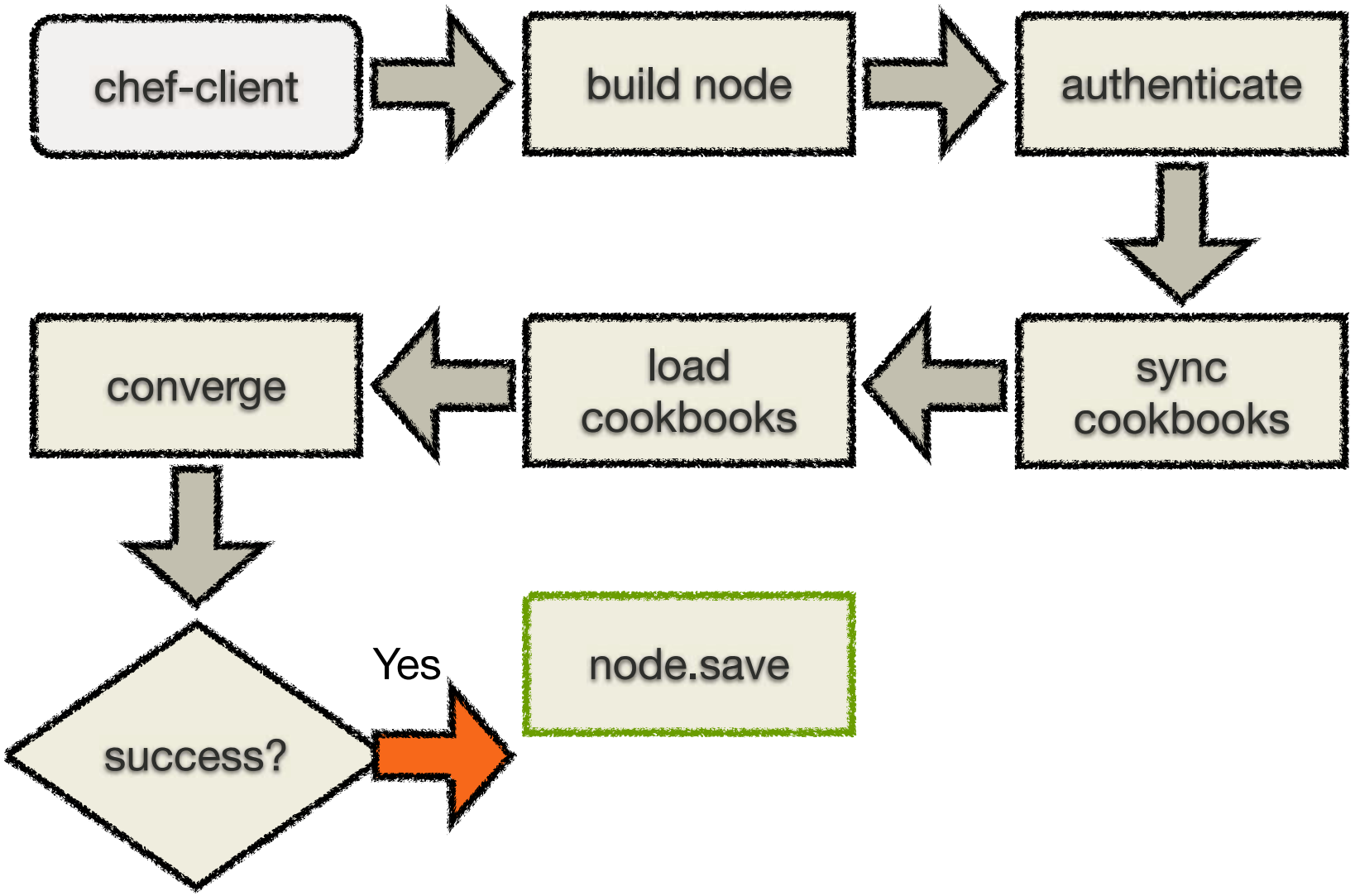

chef-client
build node
authenticate

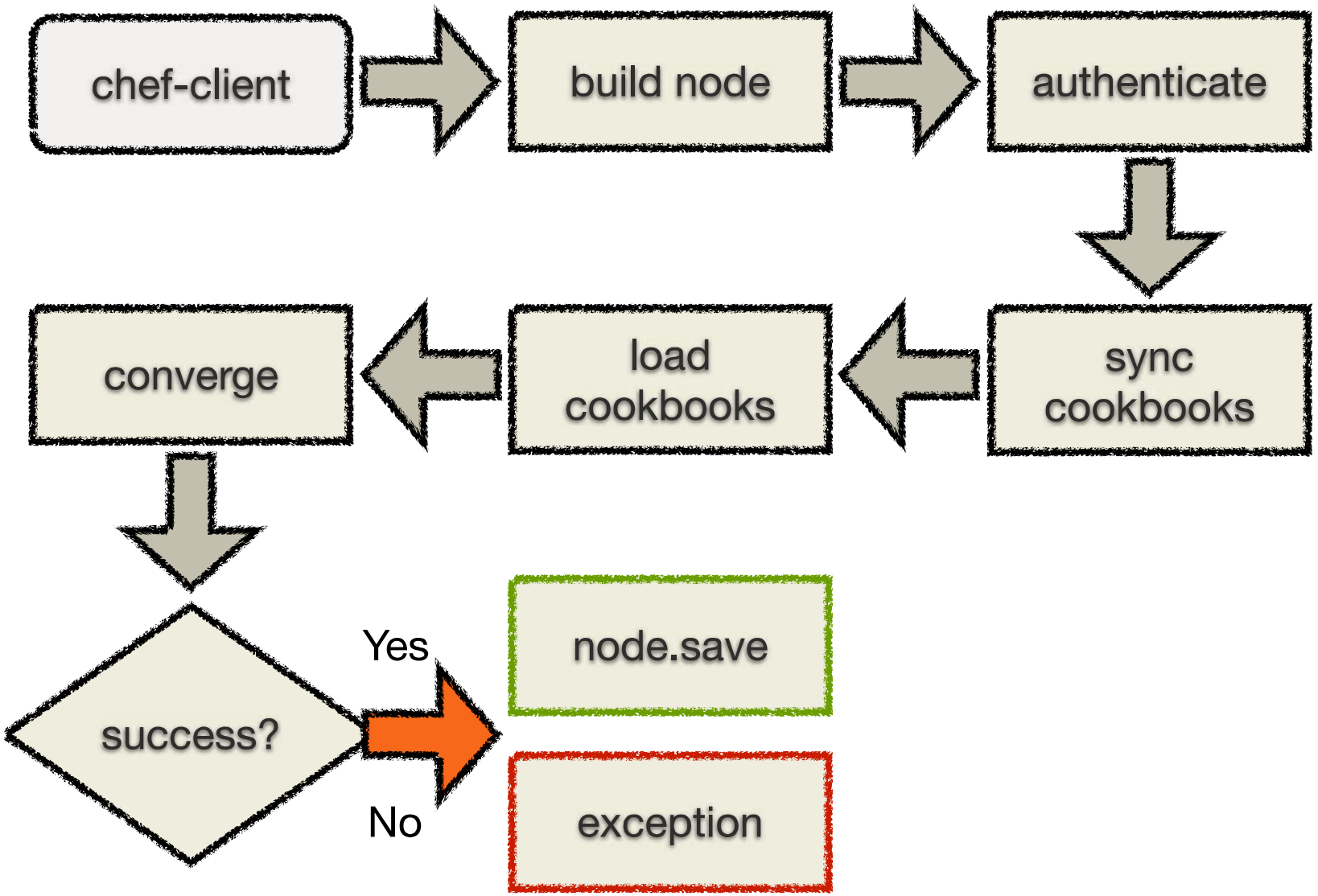

chef-client
build node
authenticate
sync cookbooks
expanded run list (recipes)

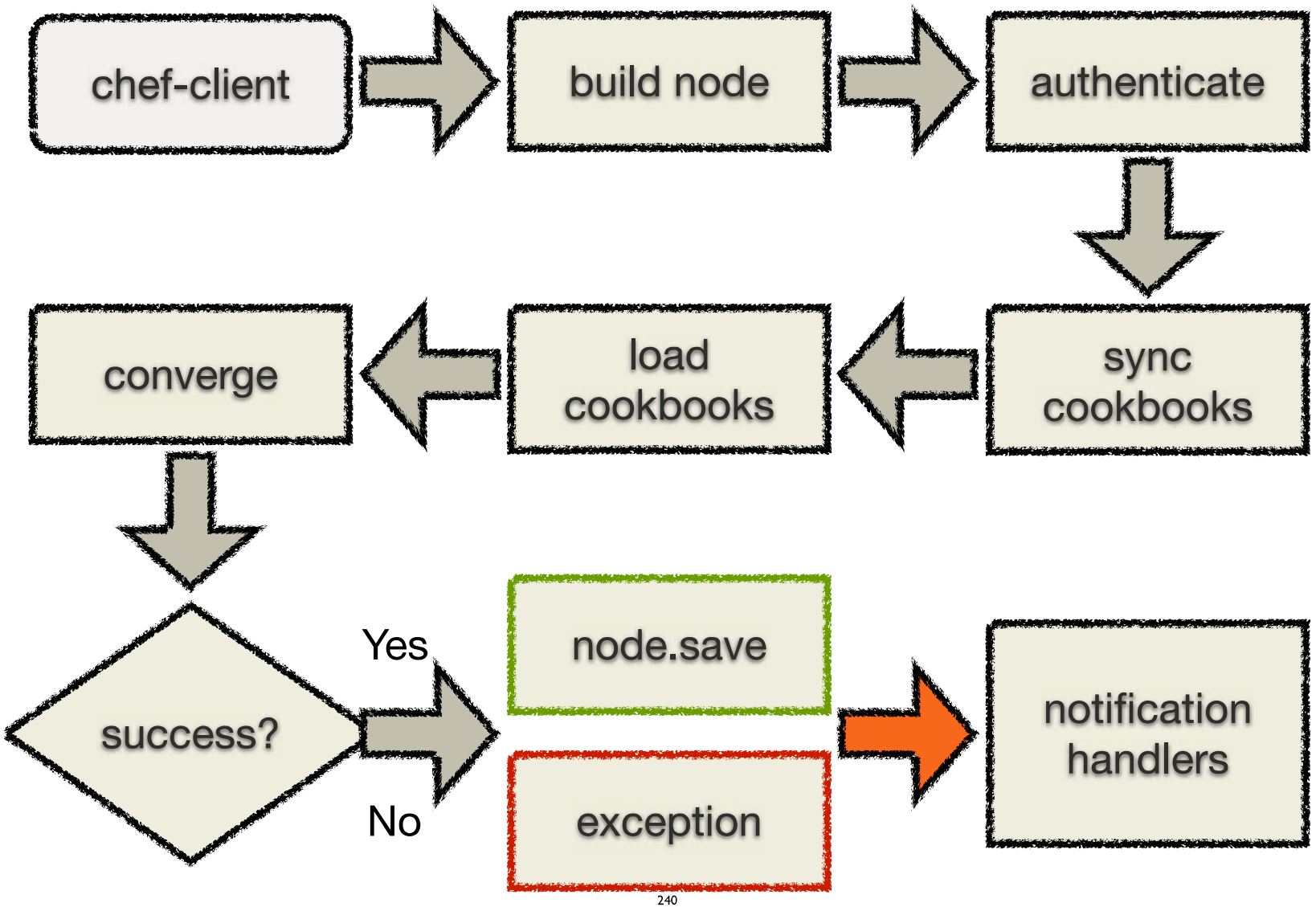

chef-client
build node
authenticate
sync cookbooks
load cookbooks


chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge


chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?

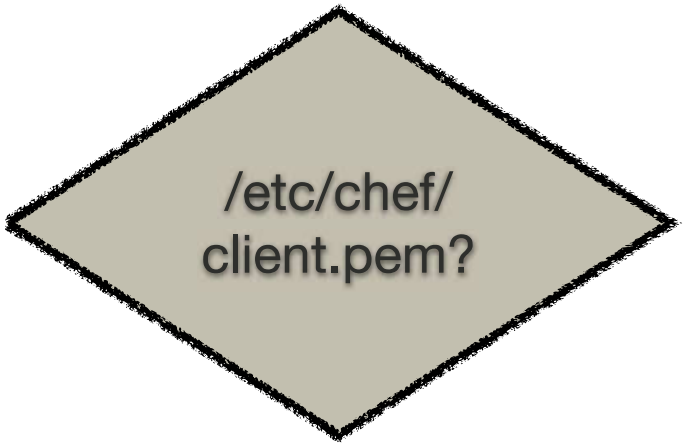

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
node.save

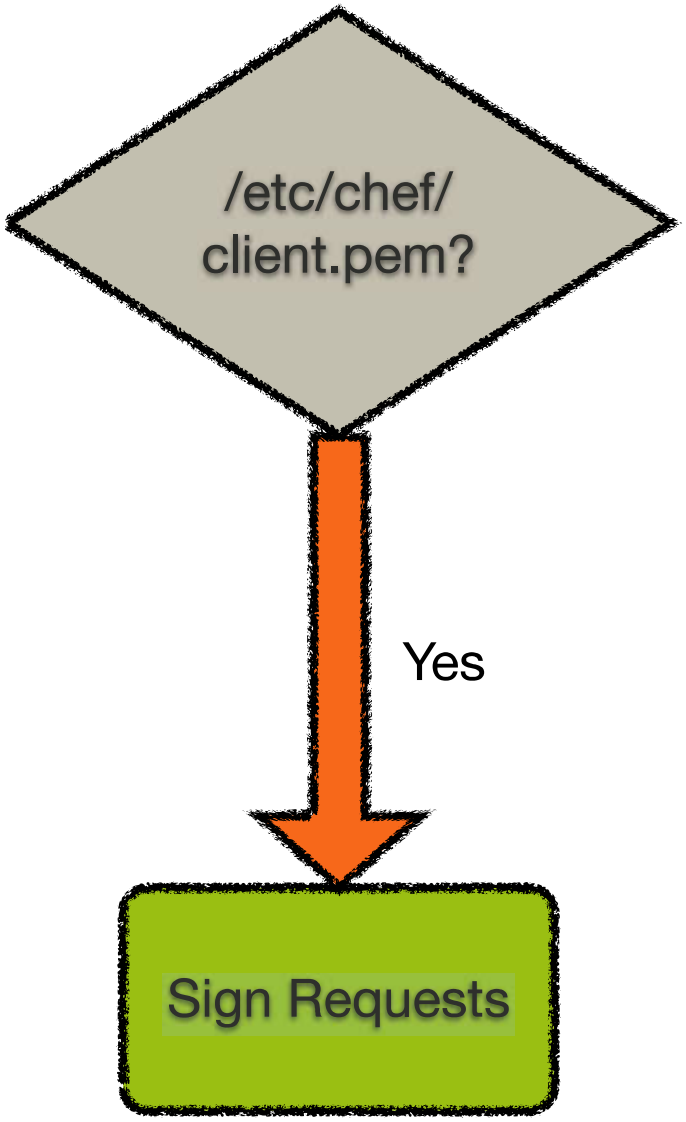

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception

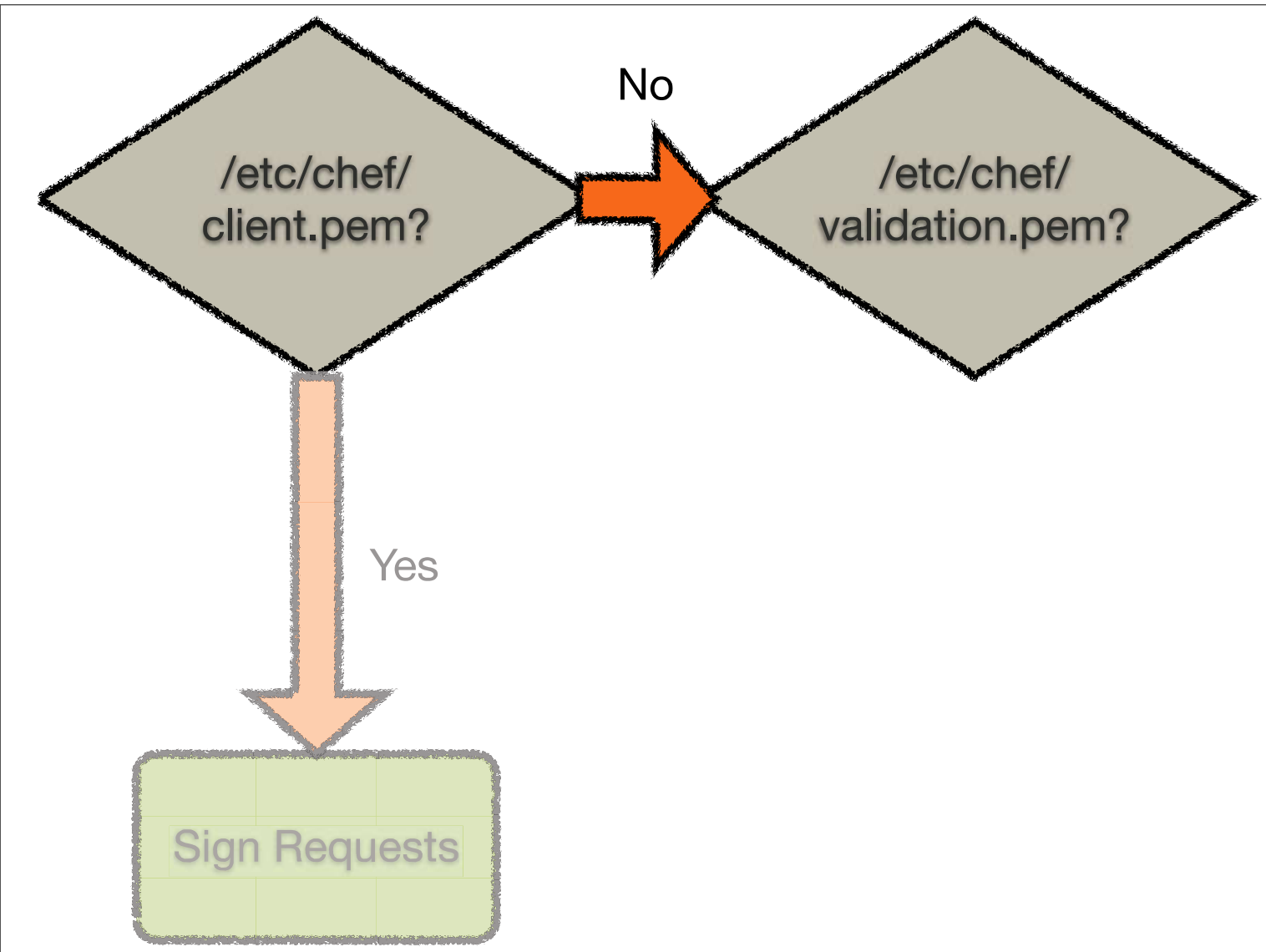

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception
notification handlers

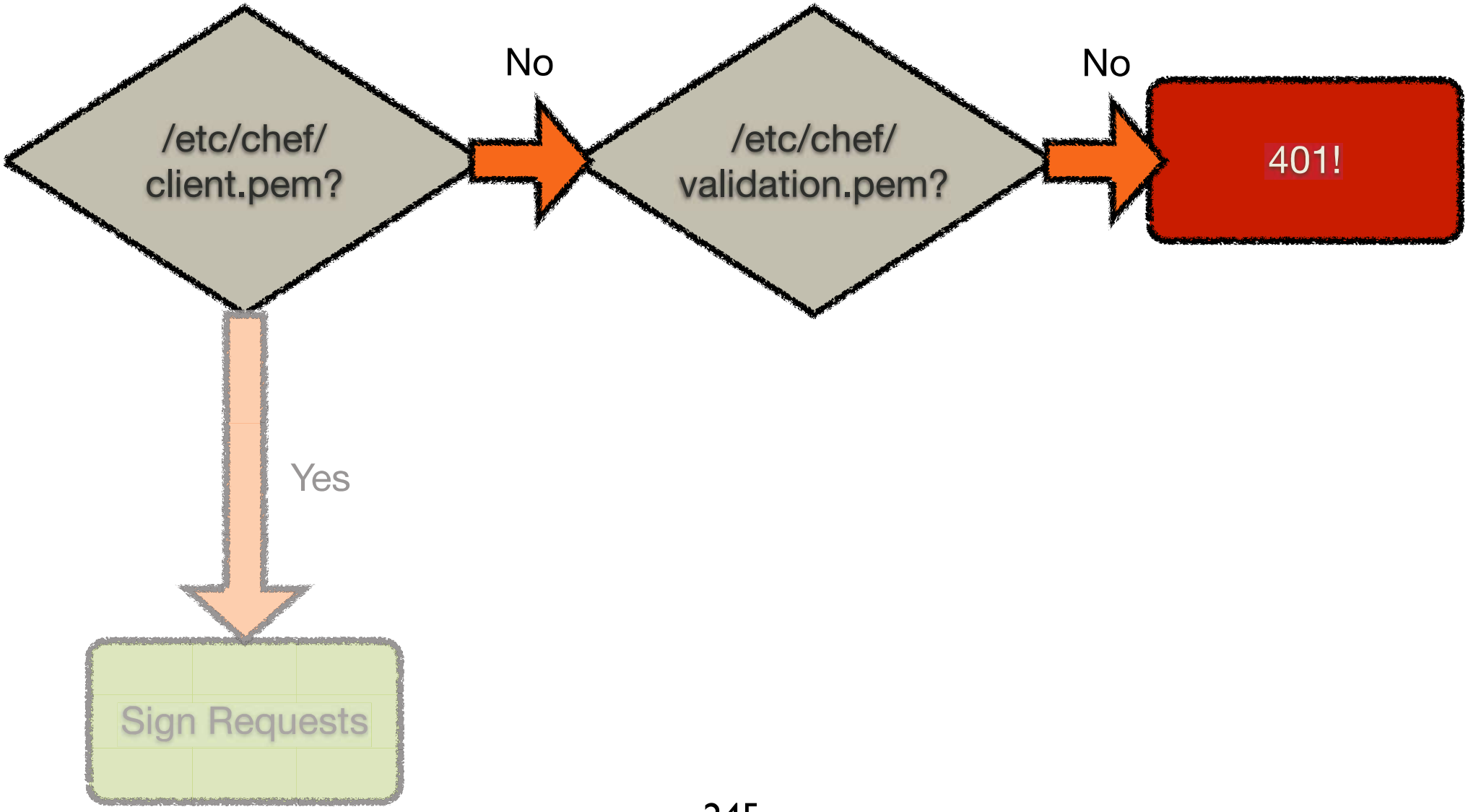
Private Keys

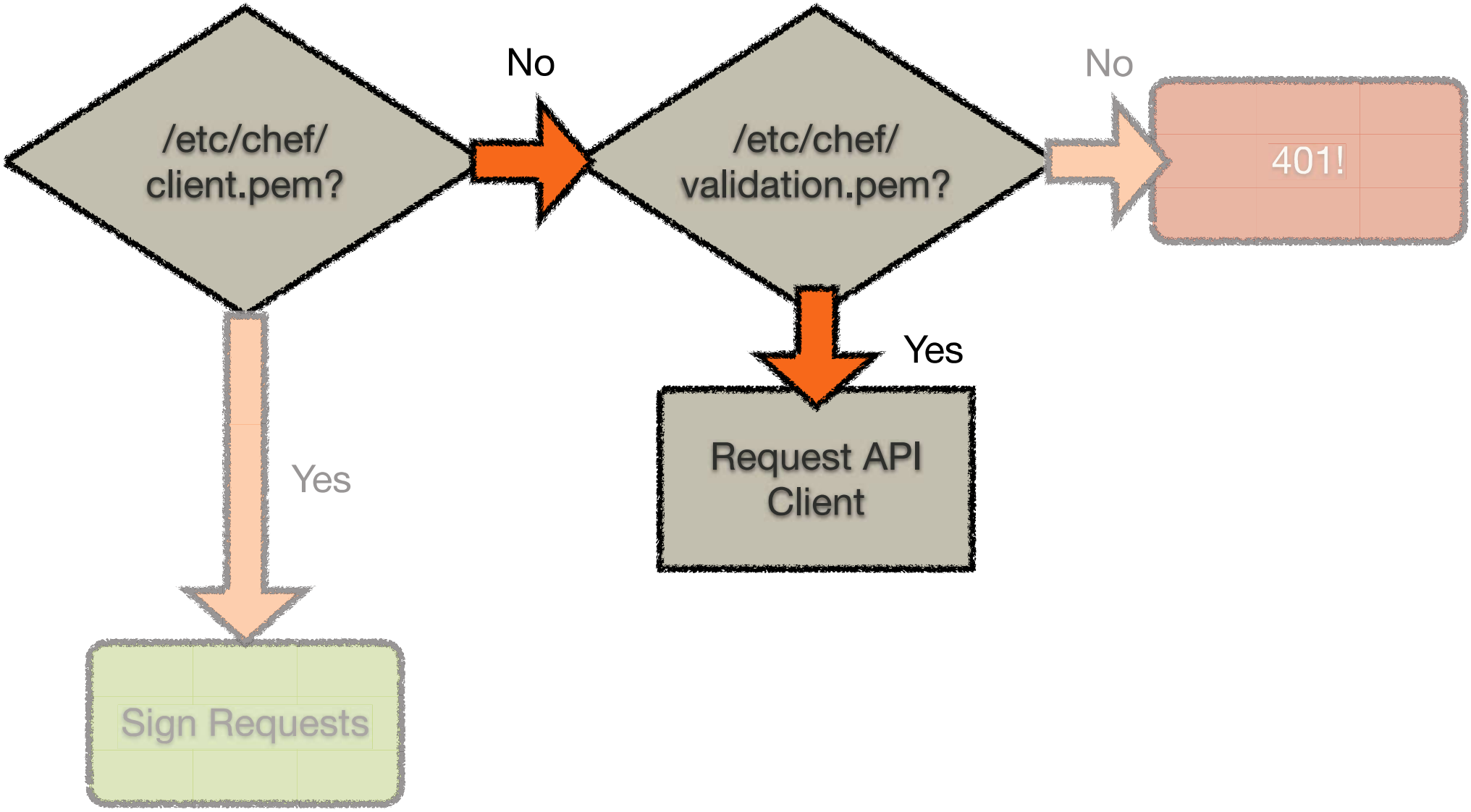
- Chef Server requires keys to authenticate.
  - `client.pem` - private key for API client
  - `validation.pem` - private key for ORGNAME-validator
- Next, let's see how those are used...

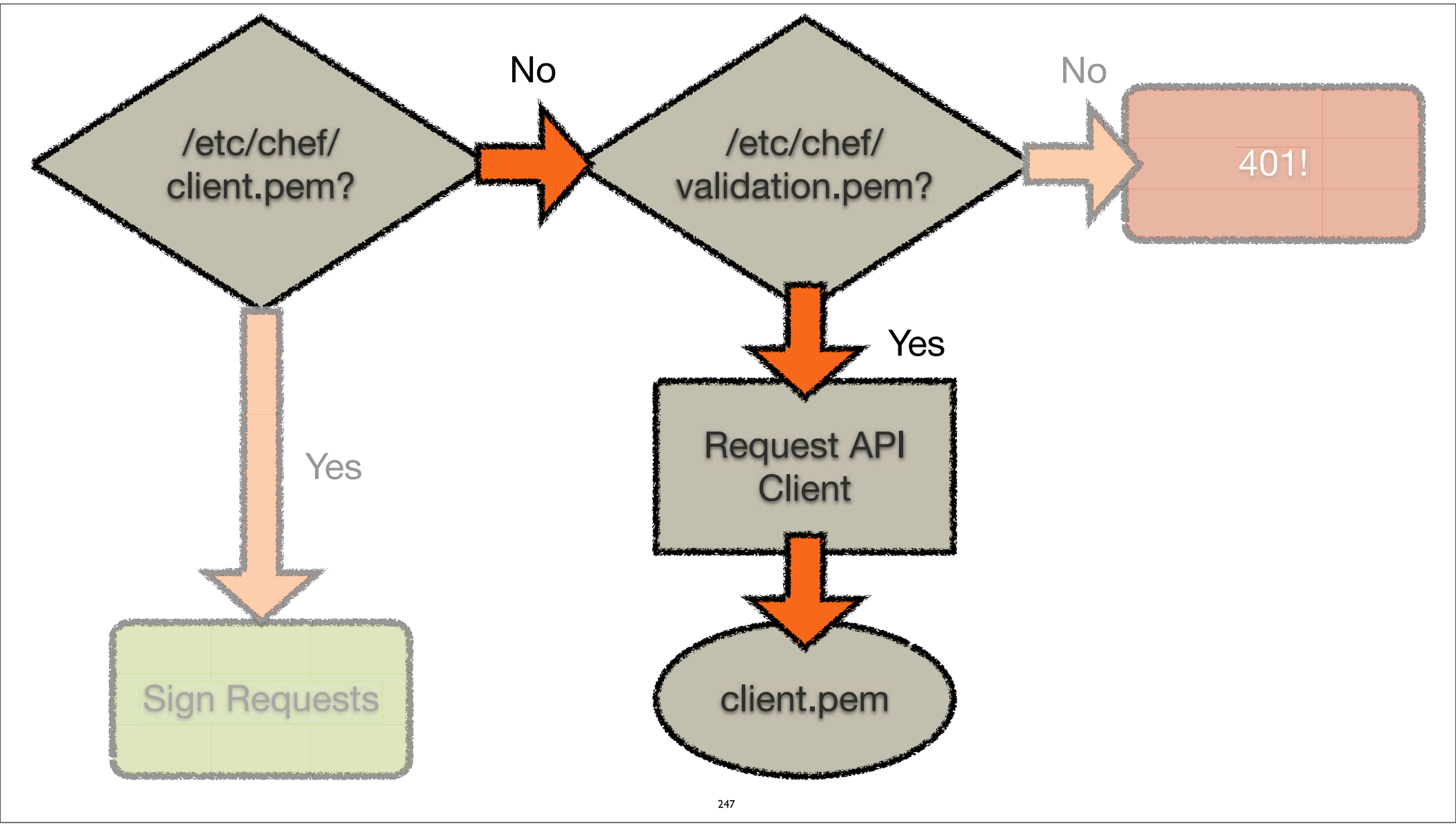

/etc/chef/
client.pem?

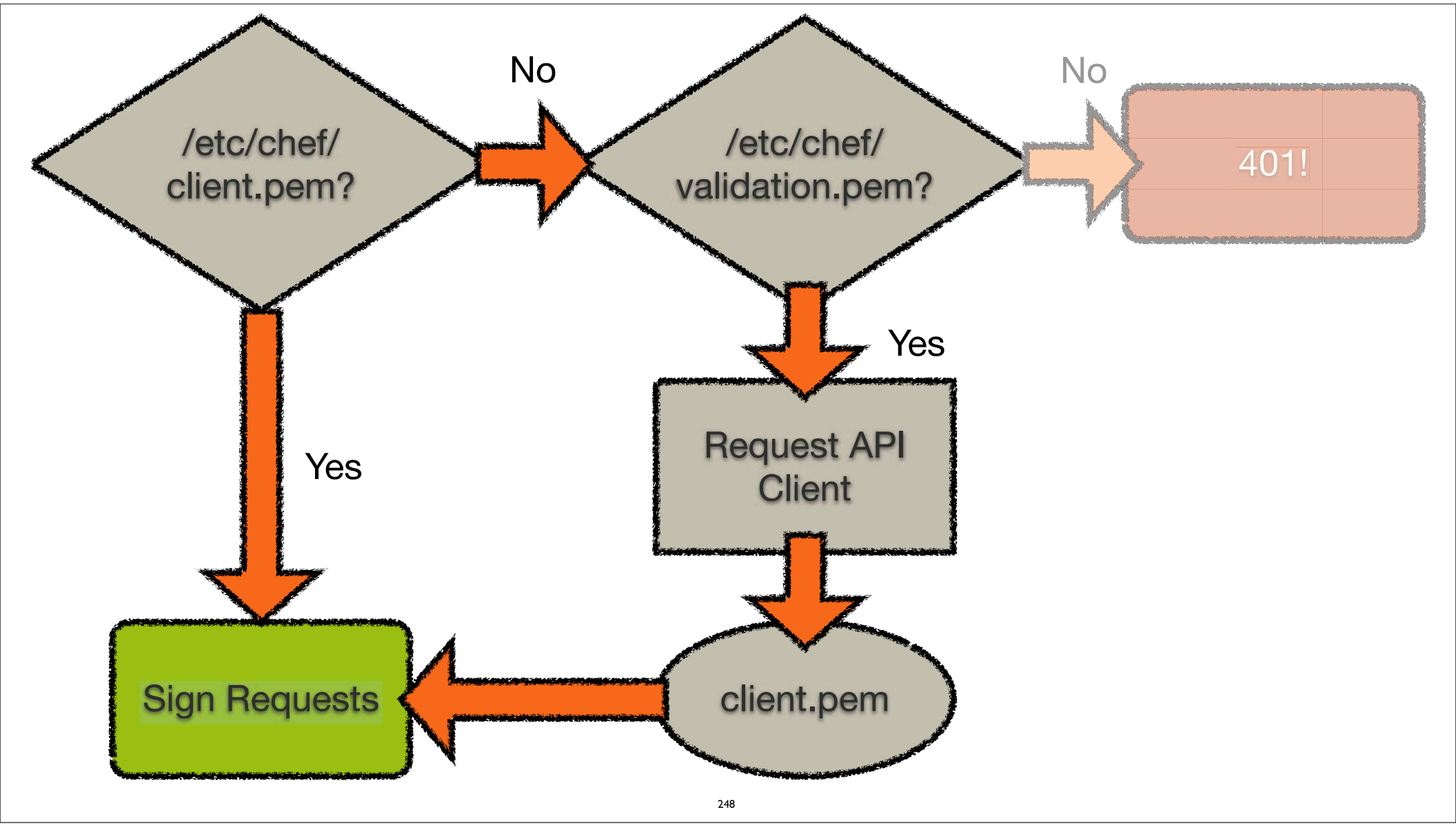

/etc/chef/
client.pem?
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Request API
Client
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Yes
Request API
Client
Sign Requests
client.pem


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Yes
Request API
Client
Sign Requests
client.pem

About the resource collection

v2.1.6

Multiphase Execution - Compile Phase

- During the compile phase, Chef

1. Loads all cookbooks from the run list

2. Reads each recipe to build the resource collection

Multiphase Execution - Execute Phase

- During the execute phase chef takes the resource collection and for each resource it will

1. Check if the resource is in the required state
   - If 'yes' - do nothing
   - If 'no' - bring resource in line with required state
2. Move on to next resource

Resource Collection Phase 1 - Compile Phase

Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
```

Resource Collection Phase 1 - Compile Phase

Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
```

Resource Collection Phase 1 - Compile Phase

Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
```

Resource Collection Phase 1 - Compile Phase

Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Resource Collection Phase 2 - Execute Phase

Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution

Resource Collection Phase 2 - Execute Phase

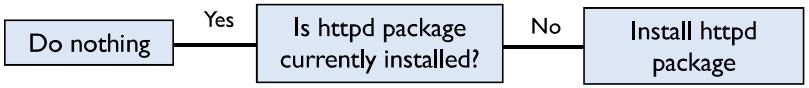
Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution



Resource Collection Phase 2 - Execute Phase

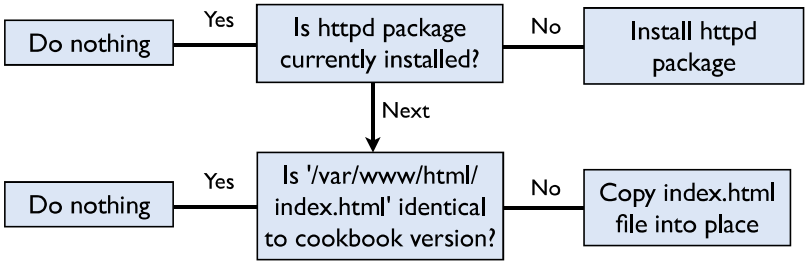
Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution



Resource Collection Phase 2 - Execute Phase

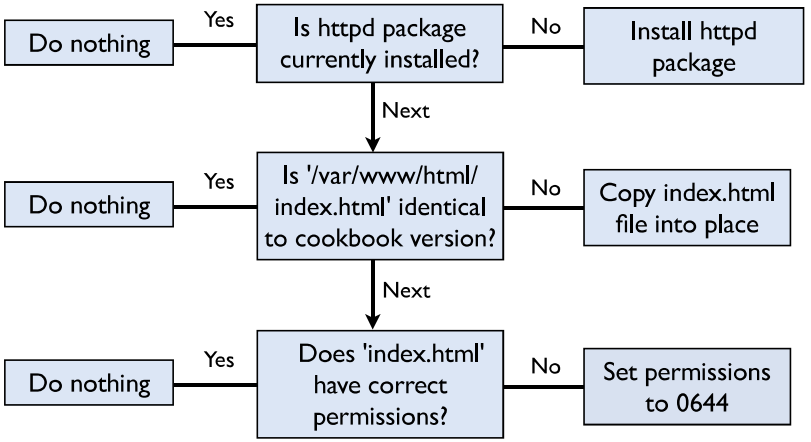
Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution



Resource Collection Phase 2 - Execute Phase

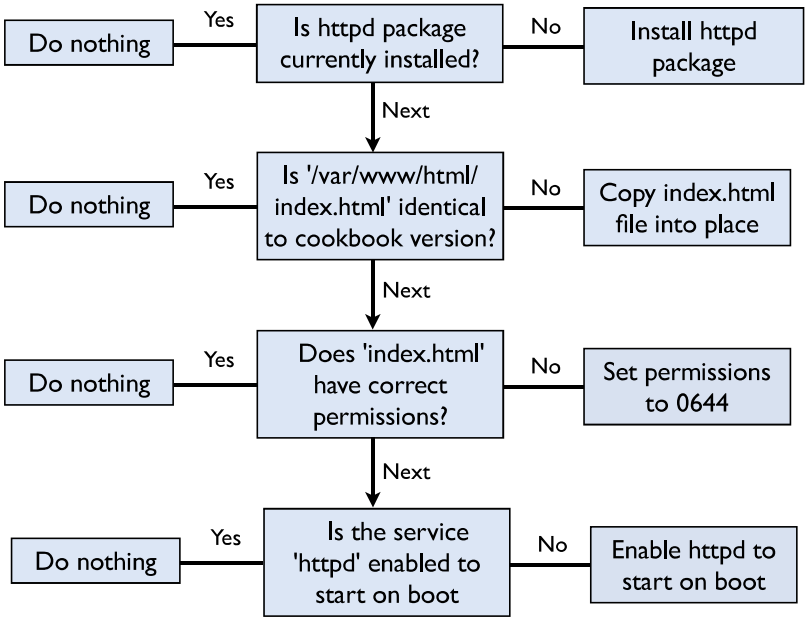
Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution



Resource Collection Phase 2 - Execute Phase

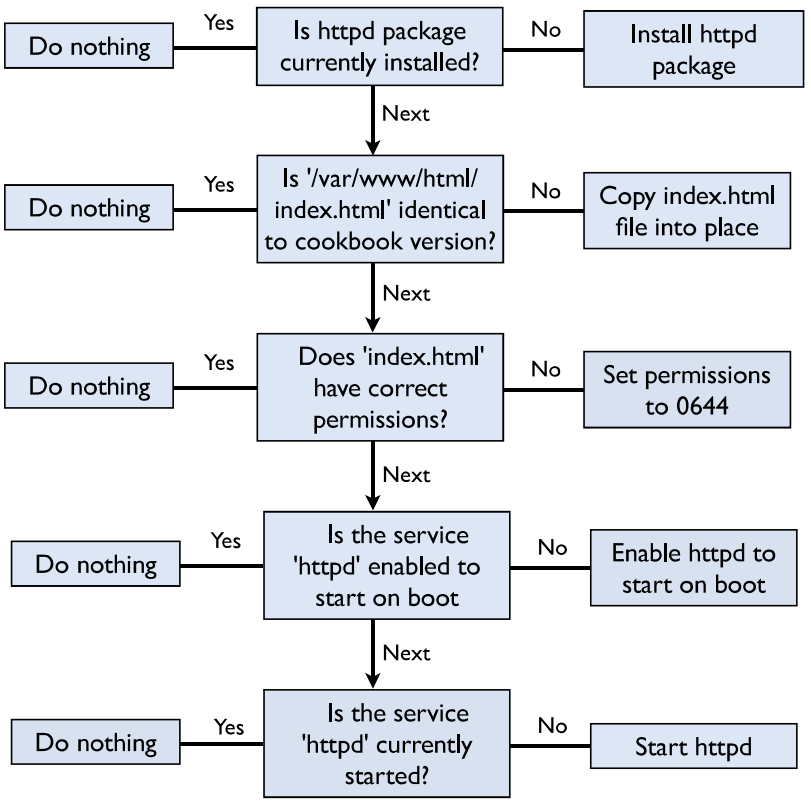
Recipe

```
package "httpd" do
  action :install
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

Resource Collection

```
resource_collection = [
 package["httpd"],
 cookbook_file["/var/www/html/index.html"],
 service ["httpd"]
]
```

Execution



Recipe order is important!

- Recipes are executed in the order they appear in the run list

```
Run List:  recipe[ntp::client], recipe[openssh::server], recipe[apache::server]
```

- These recipes are invoked in the following order

Recipe order is important!

- Recipes are executed in the order they appear in the run list

```
Run List:  recipe[ntp::client], recipe[openssh::server], recipe[apache::server]
```

- These recipes are invoked in the following order

1. recipe[ntp::client]
2. recipe[openssh::server]
3. recipe[apache::server]

Resource Collection - Multiple Recipes

1. recipe[ntp::client]

Resource Collection

```
resource_collection [
```

Resource Collection - Multiple Recipes

1. recipe[ntp::client]

```
package "ntp" do
   action :install
end

template "/etc/ntp.conf" do
   source "ntp.conf.erb"
   owner "root"
   mode "0644"
end


service "ntp" do
   action :start
end
```

Resource Collection

```
resource_collection [
   package[ntp],
   template[/etc/ntp.conf],
   service[ntp]
```

Resource Collection - Multiple Recipes

2. recipe[openssh::client]

Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
```

Resource Collection - Multiple Recipes

2. recipe[openssh::client]

```
package "openssh" do
   action :install
end

template "/etc/sshd/sshd_config" do
   source "sshd_config.erb"
   owner "root"
   mode "0644"
end


service "openssh" do
   action :start
end
```

Resource Collection

```
resource_collection [
   package[ntp],
   template[/etc/ntp.conf],
   service[ntp],
   package[openssh],
   template[/etc/sshd/sshd_config],
   service[openssh]
```

Resource Collection - Multiple Recipes

3. recipe[httpd::server]

Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
  package[openssh],
  template[/etc/sshd/sshd_config],
  service[openssh],
```

Resource Collection - Multiple Recipes

3. recipe[httpd::server]

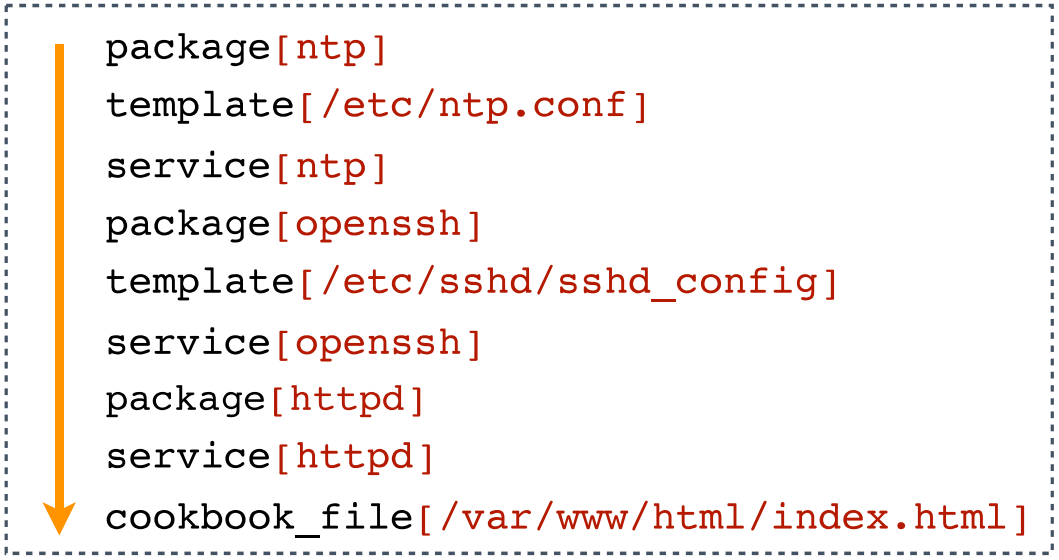
```
package "httpd" do
  action :install
end


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



cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end
```

Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
  package[openssh],
  template[/etc/sshd/sshd_config],
  service[openssh],
  package[httpd],
  service[httpd],
  cookbook_file[/var/www/html/index.html]
]
```

The final resource collection

- So the resources are invoked in the following order during the execute phase



Multiphase Execution

- Plain ruby is executed in the compile phase
- Chef DSL is executed in the execute phase

Recipe

```
%w[sites-available sites-enabled mods-available].each do |dir|
  directory "/var/www/#{dir}" do
    action :create
    mode  '0755'
    owner 'root'
    group node["apache"]["root_group"]
  end
end
```

Resource Collection

Multiphase Execution

- Plain ruby is executed in the compile phase
- Chef DSL is executed in the execute phase

Recipe

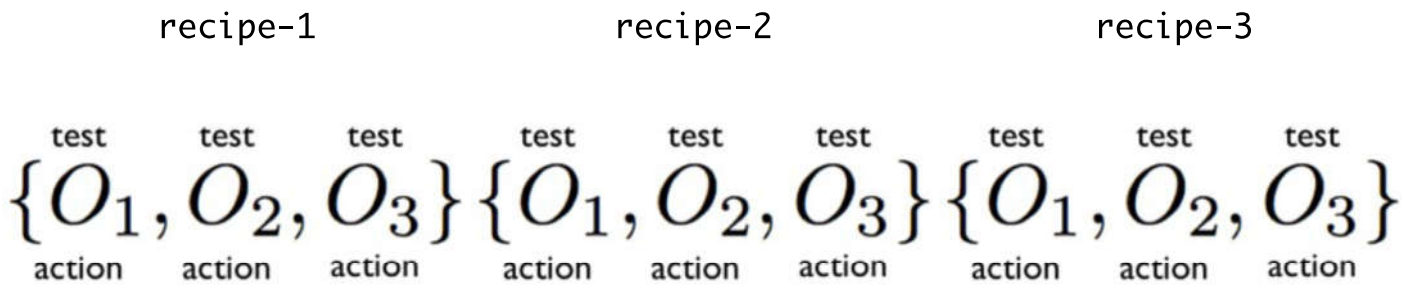
```
%w[sites-available sites-enabled mods-available].each do |dir|
  directory "/var/www/#{dir}" do
    action :create
    mode  '0755'
    owner 'root'
    group node["apache"]["root_group"]
  end
end
```

Resource Collection

```
resource_collection [
  directory["/var/www/sites-available"],
  directory["/var/www/sites-enabled"],
  directory["/var/www/mods-available"],
]
```

Remember - Resource order is important!

- Resources are invoked in the order they appear in the recipe



[---------- resource collection ----------]

Review Questions

- What are the steps in a Chef Client run?
- How does a new machine get a private key with which to authenticate requests?
- If you have the right credentials in place, why else might you not be able to authenticate?
- In which phase of a chef-client run do plain Ruby statements get evaluated?