

Policyfiles

Combining the functions of Berkshelf, Environments and Roles into a single artifact

Objectives

After completing this module, you should be able to

- Explain what Policyfiles are used for
- Use Policyfiles to set run lists for your nodes
- Describe how Policyfiles replace the legacy Roles, Environments, and Berkshelf
- Create a policyfile and a policyfile.lock.json
- Push the policyfile.lock.json to Chef Infra Server and converge a node



Policyfiles

A Policyfile (aka Policyfile.rb) is a file that contains information about a node's:

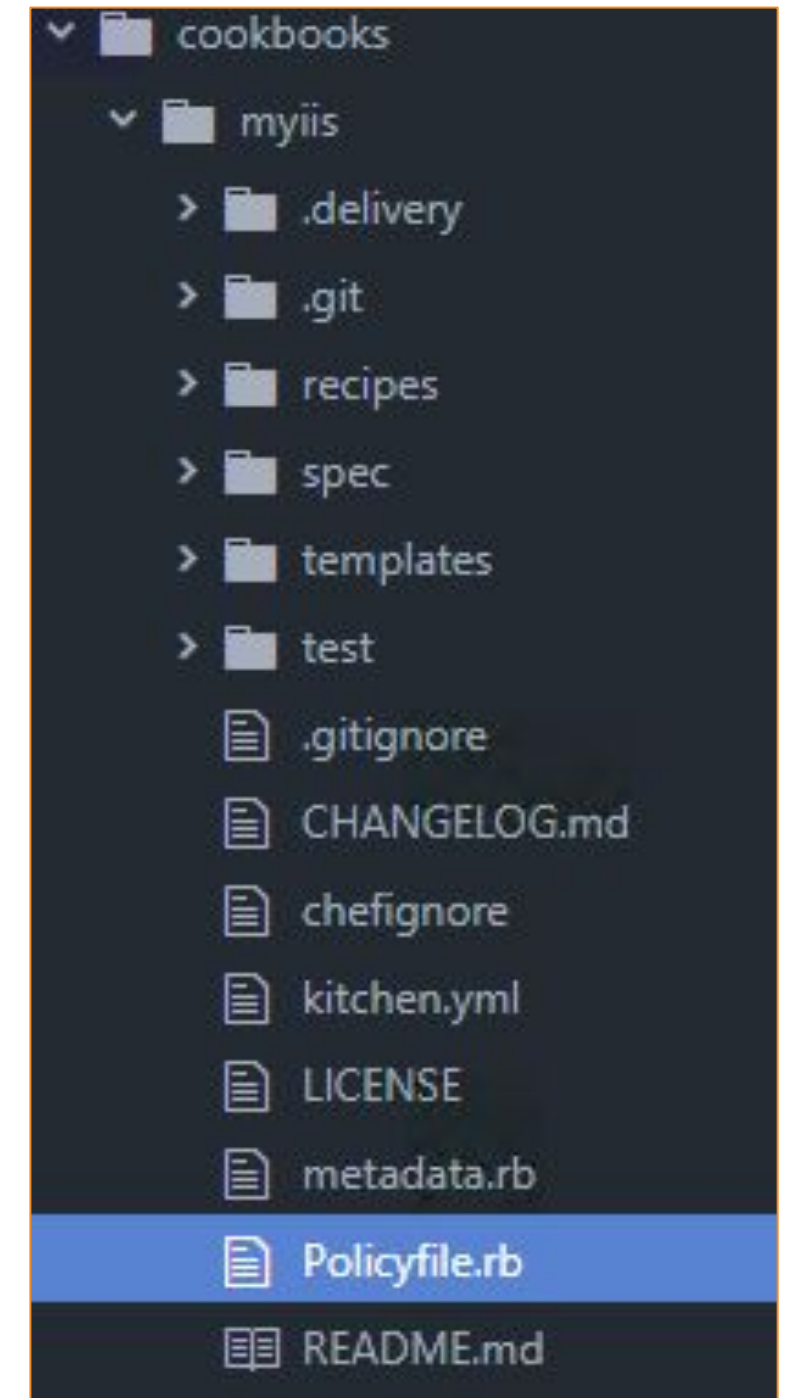
- Policy_name
- Default source for fetching cookbooks
- Run list (or multiple run lists)
- Custom cookbook paths
- Optional cookbook attributes

Policyfiles

When you generate a Chef cookbook using a modern version of Chef Workstation, a Policyfile.rb file is automatically created.(* see speaker notes below slide)

Notice there is no longer a Berksfile generated like in older versions of Chef Workstation/ChefDK.

A Policyfile is the best way to manage run lists, and community cookbook data with a single document that is uploaded to the Chef Infra Server.



Policyfile

This is an example of the Policyfile that was auto generated when you ran `chef generate cookbook myiis` earlier in this course.

name: Used as not just a name for this policyfile, but it replaces the old **role** object. Use a name that reflects the purpose of the machines where the policy will run.

The name must be unique.

Nodes using this policyfile will possess the **myiis** role.

```
Policyfile.rb
1  # Policyfile.rb - Describe how you want Chef Infra Client to
2  #
3  # For more information on the Policyfile feature, visit
4  # https://docs.chef.io/policyfile.html
5
6  # A name that describes what the system you're building with
7  name 'myiis'
8
9  # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order
13 run_list 'myiis::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'myiis', path: '.'
17
```


Policyfile

default_source: Specifies where we get cookbooks if they're not specifically declared in the cookbook section below.

This will usually be the public or a private supermarket or Chef Infra Server.

default_source can also be used for an internal repository where all your cookbooks reside.

```
Policyfile.rb
1  # Policyfile.rb - Describe how you want Chef Infra Client to
2  #
3  # For more information on the Policyfile feature, visit
4  # https://docs.chef.io/policyfile.html
5
6  # A name that describes what the system you're building with
7  name 'myiis'
8
9  # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order
13 run_list 'myiis::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'myiis', path: '.'
17
```

Policyfile

run_list: This sets the run list for any nodes using this Policyfile.

cookbook `COOKBOOK`, path: declares the non-default location where cookbooks can be found.

This may be a path to the top-level of a cookbook repository or to the `./cookbooks` directory within that repository.

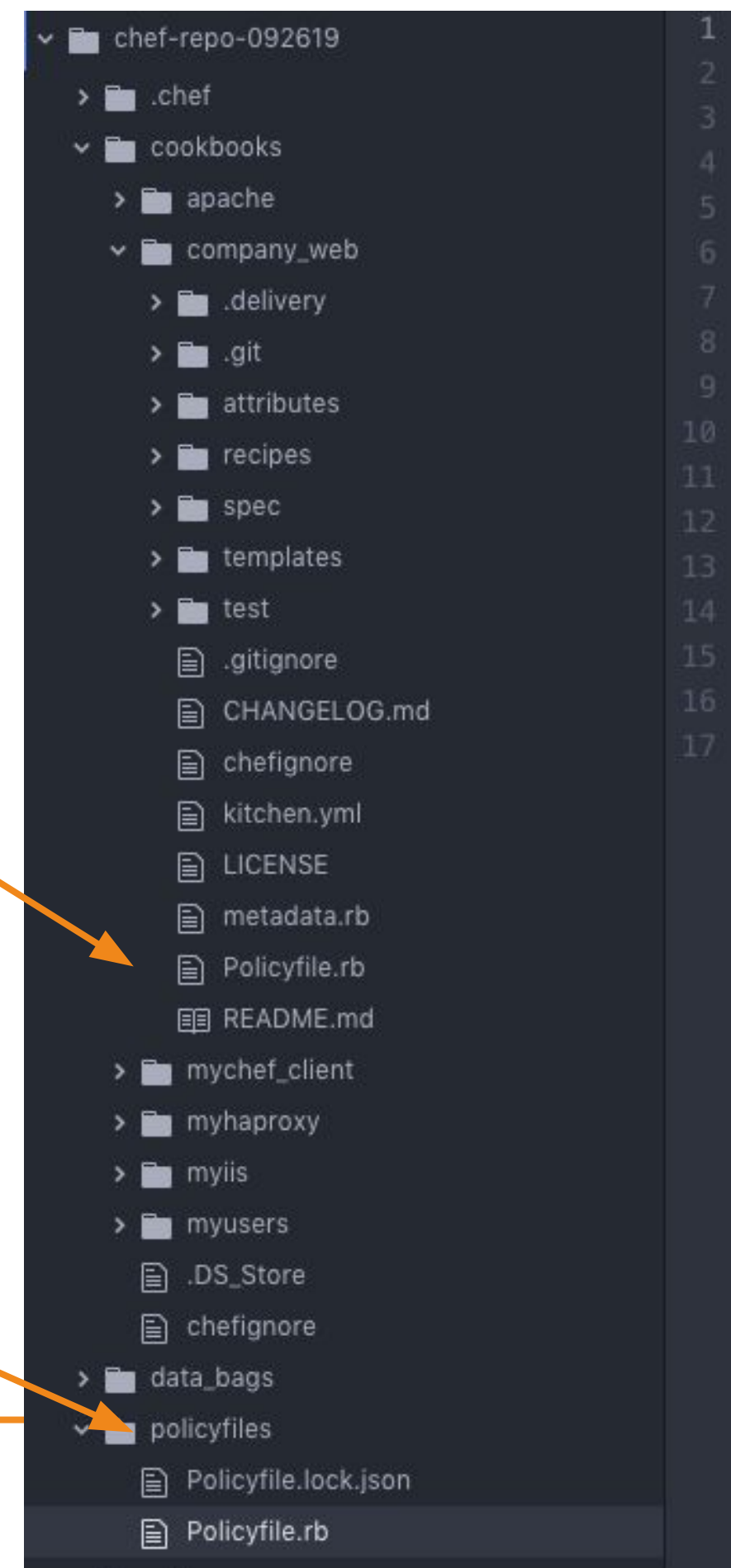
```
Policyfile.rb
1  # Policyfile.rb - Describe how you want Chef Infra Client to
2  #
3  # For more information on the Policyfile feature, visit
4  # https://docs.chef.io/policyfile.html
5
6  # A name that describes what the system you're building with
7  name 'myiis'
8
9  # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order
13 run_list 'myiis::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'myiis', path: '.'
17
```

Policyfile Location

A node can have only one policy, so its policyfile will likely have a run list containing multiple cookbooks. Therefore, it wouldn't make sense for the Policyfile to be in a particular cookbook like this.

So we can use the auto-generated Policyfile as a guideline for creating our own Policyfile or simply ignore it.

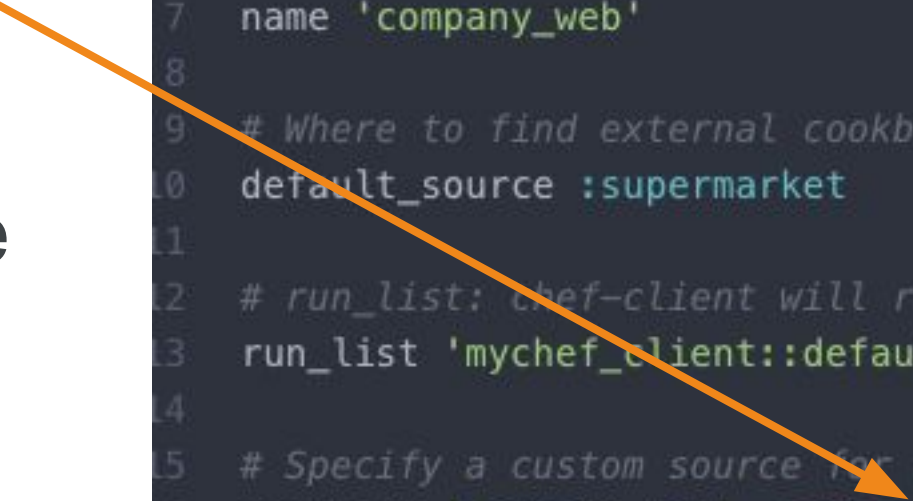
We recommend you create a **policyfiles** directory at the same level as the **cookbooks** directory. Then you can generate all your policyfiles within the **policyfiles** directory.



Cookbook Location

Assuming you place your policyfiles in the recommended policyfiles directory, you would need to specify the location of its cookbooks like in this example, where **'../cookbooks/company_web'** in **this policyfile** is saying, go up one level and then down into the cookbooks directory to find the cookbook(s).

```
company_web.lock.json | company_web.rb
1 # Policyfile.rb - Describe how you want Chef Infra Client to build your system.
2 #
3 # For more information on the Policyfile feature, visit
4 # https://docs.chef.io/policyfile.html
5
6 # A name that describes what the system you're building with Chef does.
7 name 'company_web'
8
9 # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order specified.
13 run_list 'mychef_client::default', 'company_web::default', 'myusers::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'company_web', path: '../cookbooks/company_web'
17 cookbook 'myiis', path: '../cookbooks/myiis'
18 cookbook 'apache', path: '../cookbooks/apache'
19 cookbook 'mychef_client', path: '../cookbooks/mychef_client'
20 cookbook 'myusers', path: '../cookbooks/myusers'
21
```



Cookbook Location

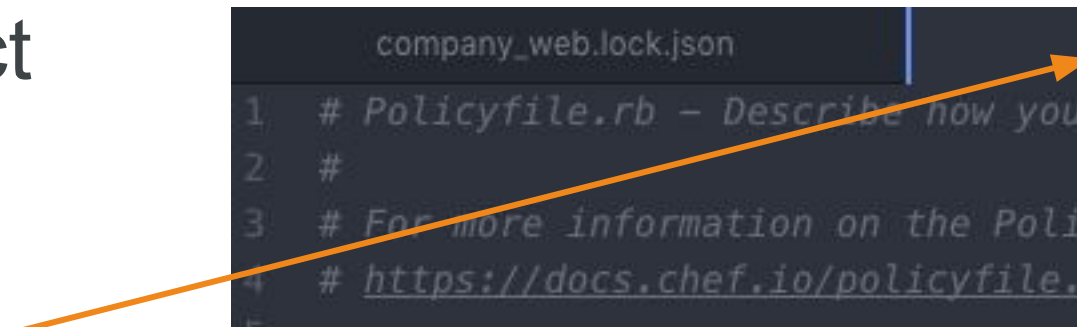
In this way, when you eventually upload your Policyfile (actually a Policyfile.lock.json) to Chef Infra Server, the required cookbooks will also be uploaded simultaneously.

```
company_web.lock.json | company_web.rb
1  # Policyfile.rb - Describe how you want Chef Infra Client to build your system.
2  #
3  # For more information on the Policyfile feature, visit
4  # https://docs.chef.io/policyfile.html
5
6  # A name that describes what the system you're building with Chef does.
7  name 'company_web'
8
9  # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order specified.
13 run_list 'mychef_client::default', 'company_web::default', 'myusers::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'company_web', path: '../cookbooks/company_web'
17 cookbook 'myiis', path: '../cookbooks/myiis'
18 cookbook 'apache', path: '../cookbooks/apache'
19 cookbook 'mychef_client', path: '../cookbooks/mychef_client'
20 cookbook 'myusers', path: '../cookbooks/myusers'
21
```

Policyfile Naming

Also in this example is the correct way to name your policyfiles.

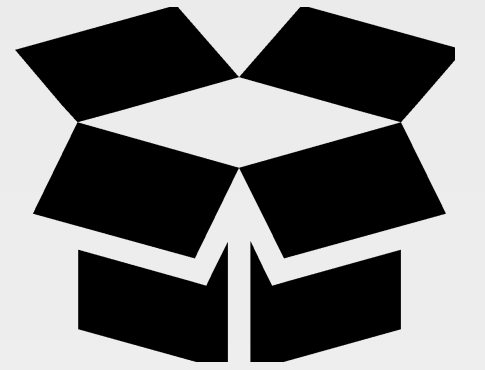
In this example we named the policyfile **company_web.rb** so we can differentiate it from other policyfiles that will reside in the **policyfiles** directory.



```
company_web.lock.json | company_web.rb
1 # Policyfile.rb - Describe how you want Chef Infra Client to build your system.
2 #
3 # For more information on the Policyfile feature, visit
4 # https://docs.chef.io/policyfile.html
5
6 # A name that describes what the system you're building with Chef does.
7 name 'company_web'
8
9 # Where to find external cookbooks:
10 default_source :supermarket
11
12 # run_list: chef-client will run these recipes in the order specified.
13 run_list 'mychef_client::default', 'company_web::default', 'myusers::default'
14
15 # Specify a custom source for a single cookbook:
16 cookbook 'company_web', path: '../cookbooks/company_web'
17 cookbook 'myiis', path: '../cookbooks/myiis'
18 cookbook 'apache', path: '../cookbooks/apache'
19 cookbook 'mychef_client', path: '../cookbooks/mychef_client'
20 cookbook 'myusers', path: '../cookbooks/myusers'
21
```

CONCEPT

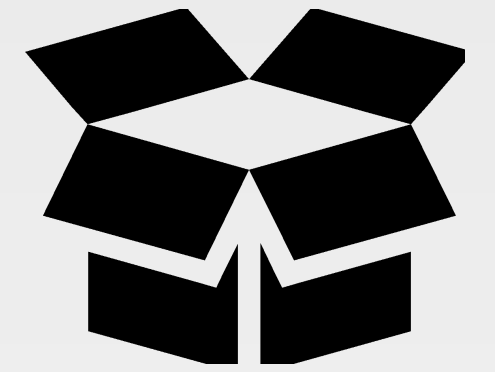
Policyfile.lock.json



Before you upload your Policyfile to Chef Infra Server, you actually need to generate Policyfile.lock.json based on the Policyfile.rb.

In other words, we never upload the Policyfile.rb file. We only upload Policyfile.lock.json, which in turn enables the uploading of any required cookbooks.

CONCEPT



Policyfile.lock.json

Policyfile.lock.json also identifies the checksum of each cookbook in the run_list, then creates a master checksum (the revision ID) adding up the checksum of all the cookbooks.

This revision ID is the unique identifier for this group of cookbooks and their included files. If you and I have the EXACT same cookbooks (identical down to each individual character), we'll have the same revision id.

Example: Policyfile.lock

To generate the Policyfile.lock.json if the policyfile name is **company_web**:

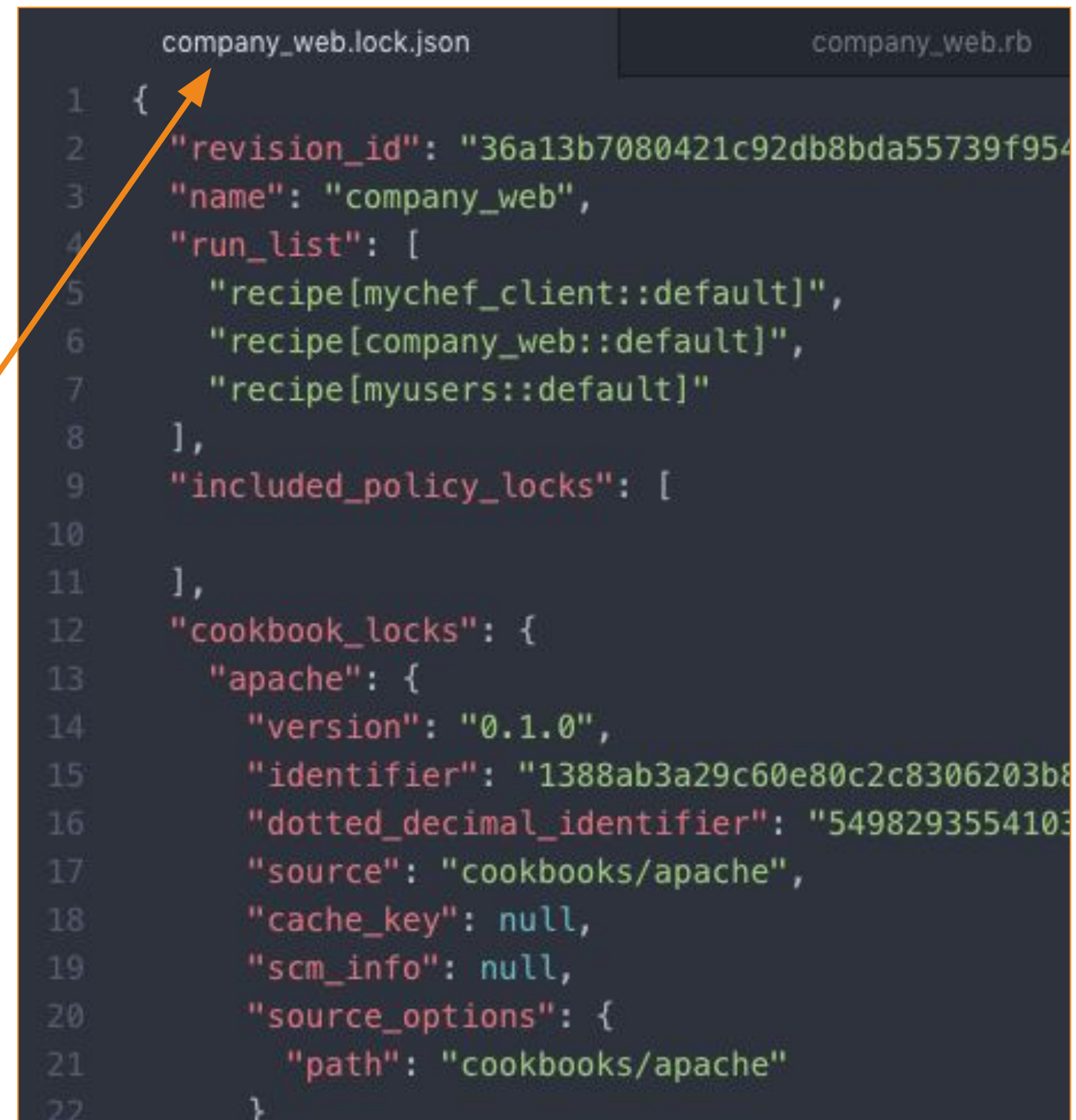
```
chef install company_web.rb
```

That will take the contents of your **company_web.rb** policyfile and convert it to the lock file.

Then you upload the **company_web.lock.json** to the Chef Infra Server with:

```
chef push policy_group company_web
```

Note: We will cover the *policy_group* in a moment.



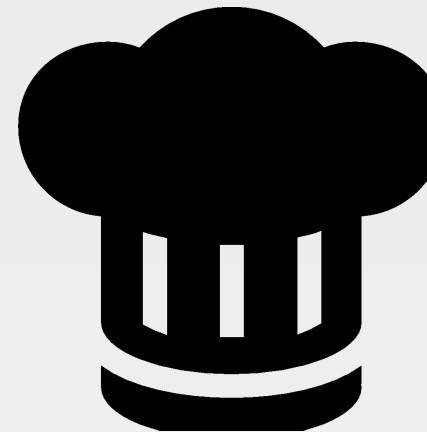
```
company_web.lock.json
1 {
2   "revision_id": "36a13b7080421c92db8bda55739f954",
3   "name": "company_web",
4   "run_list": [
5     "recipe[mychef_client::default]",
6     "recipe[company_web::default]",
7     "recipe[myusers::default]"
8   ],
9   "included_policy_locks": [
10
11   ],
12   "cookbook_locks": {
13     "apache": {
14       "version": "0.1.0",
15       "identifier": "1388ab3a29c60e80c2c8306203b8",
16       "dotted_decimal_identifier": "5498293554103",
17       "source": "cookbooks/apache",
18       "cache_key": null,
19       "scm_info": null,
20       "source_options": {
21         "path": "cookbooks/apache"
22       }
23     }
24   }
25 }
```

Policyfile.lock

When you generate the Policyfile.lock.json file, a **revision_id** is generated in the form of a hash.

That hash is the version number with which you can identify versions of this Policyfile.

```
Policyfile.lock.json
1  {
2    "revision_id": "bd6c581e1a16a3e317f043dd24f4b4f55f08352e8df83a9f5290aef0ae4",
3    "name": "myiis",
4    "run_list": [
5      "recipe[myiis::default]"
6    ],
7    "included_policy_locks": [
8
9    ],
10   "cookbook_locks": {
11     "myiis": {
12       "version": "0.2.1",
13       "identifier": "17ddb9d2c05e62af009d39b21f041e2d01cfc7b",
14       "dotted_decimal_identifier": "6717730919810534.12085874017378800.724424",
15       "source": ".",
16       "cache_key": null,
17       "scm_info": {
18         "scm": "git",
19         "remote": null,
20         "revision": "d691971666b4079580636ca8958ea928cb1ca064",
21         "working_tree_clean": false,
22         "published": false,
23         "synchronized_remote_branches": [
24
25         ]
26       },
27       "source_options": {
28         "path": "."
29       }
30     }
31   }
32 }
```



GL: Generate a Policyfile

In this group lab we'll practice creating a policyfile

Objective:

- ☐ Create a Policyfile directory
- ☐ Generate a Policyfile for Apache, including a run list
- ☐ Generate a Policyfile.lock.json
- ☐ Push a Policyfile.lock.json to Chef Infra Server

GL: Create a policyfiles Directory



```
cd ~/chef-repo  
> mkdir policyfiles
```

GL: Confirm the new policyfiles Directory Exists



```
> ls (or dir for Windows)
```

```
README.md cookbooks policyfiles roles
```


GL: Generate a Policyfile for Apache



```
cd ~/chef-repo/policyfiles>  
> chef generate policyfile apache
```

```
Recipe: code_generator::policyfile
```

```
  * template[/Users/sdelfante/chef-repo/policyfiles/apache.rb] action create  
    - create new file /Users/sdelfante/chef-repo/policyfiles/apache.rb  
    - update content in file /Users/sdelfante/chef-repo/policyfiles/apache.rb  
from none to 65a9ac  
    (diff output suppressed by config)
```

GL: Confirm the new poliyfiles Exists



```
> ls (or dir for Windows)
```

```
apache.rb
```

GL: View the Policyfile



```
> cat apache.rb
```

```
# Policyfile.rb - Describe how you want Chef Infra Client to build your system.
```

```
...
```

```
# A name that describes what the system you're building with Chef does.
```

```
name 'apache'
```

```
# Where to find external cookbooks:
```

```
default_source :supermarket
```

```
# run_list: chef-client will run these recipes in the order specified.
```

```
run_list 'apache::default'
```

```
# Specify a custom source for a single cookbook:
```

```
# cookbook 'example_cookbook', path: '../cookbooks/example_cookbook'
```

In the next step you'll need to add the the path to the apache cookbook.

GL: Edit the New apache.rb Policyfile

 `~/chef-repo/policyfiles/apache.rb`

```
#...skipping for brevity...
# https://docs.chef.io/policyfile.html
# A name that describes what the system you're building with Chef does.
name 'apache'

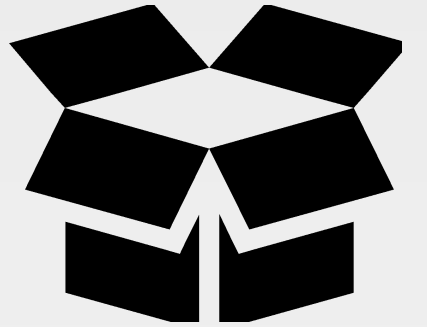
# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list 'apache::default'

# Specify a custom source for a single cookbook: cookbook 'apache', path:
# cookbook 'example_cookbook', path: '../cookbooks/example_cookbook'
cookbook 'apache', path: '../cookbooks/apache'
```

Add the code in green.

CONCEPT



GL: Policyfile.rb and the Policyfile.lock.json

Now that we have our Policyfile.rb (apache.rb) we need to generate the Policyfile.lock.json (apache.lock.json) before we could upload the apache.lock.json to Chef Infra Server.

The `chef install policy_name` command creates the Policyfile.lock.json.

GL: Generate the apache.lock.json



```
~/chef-repo/policyfiles> chef install apache.rb
```

```
Building policy apache
```

```
Expanded run list: recipe[apache::default]
```

```
Caching Cookbooks...
```

```
Installing apache 0.0.5
```

```
Installing apache2 8.3.0
```

```
Lockfile written to /Users/sdelfante/chef-repo/policyfiles/apache.lock.json
```

```
Policy revision id:
```

```
40edea4d9edf5f74e402e6d47d9e719750e451d2b9cb930a1725c28e918c17ec
```

GL: Confirm the new `apache.lock.json` Exists



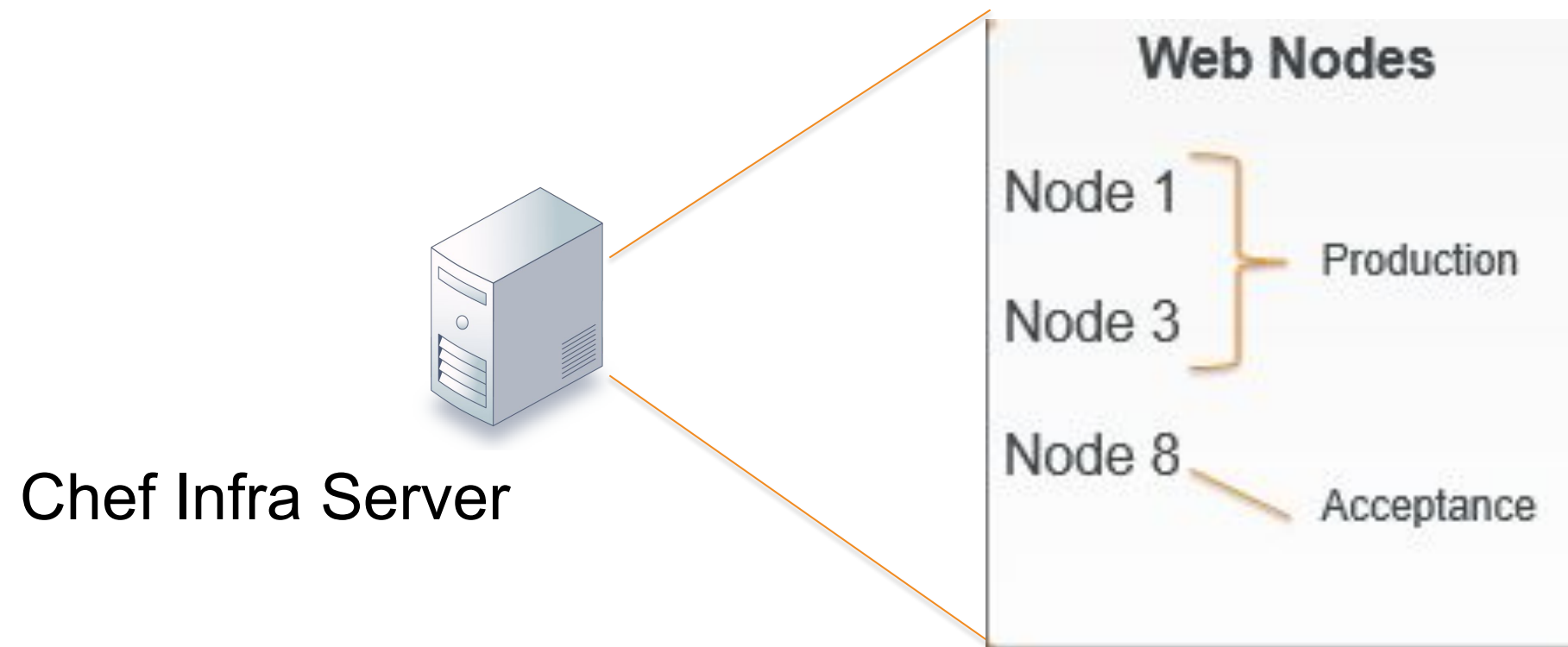
```
ls (or dir for Windows)
```

```
apache.lock.json  apache.rb
```

In practice, at this point you could upload the `apache.lock.json` to Chef Infra Server but we need to define one more thing before we do.

Policy Group = Environment

At the time when you upload the Policyfile.lock.json to the Chef Infra Server is when you specify a policy group, which can act like an environment such as **production** or **acceptance**. You could also think of policy group as a way to group like servers together. **Note:** Policy group replaces the legacy environments.

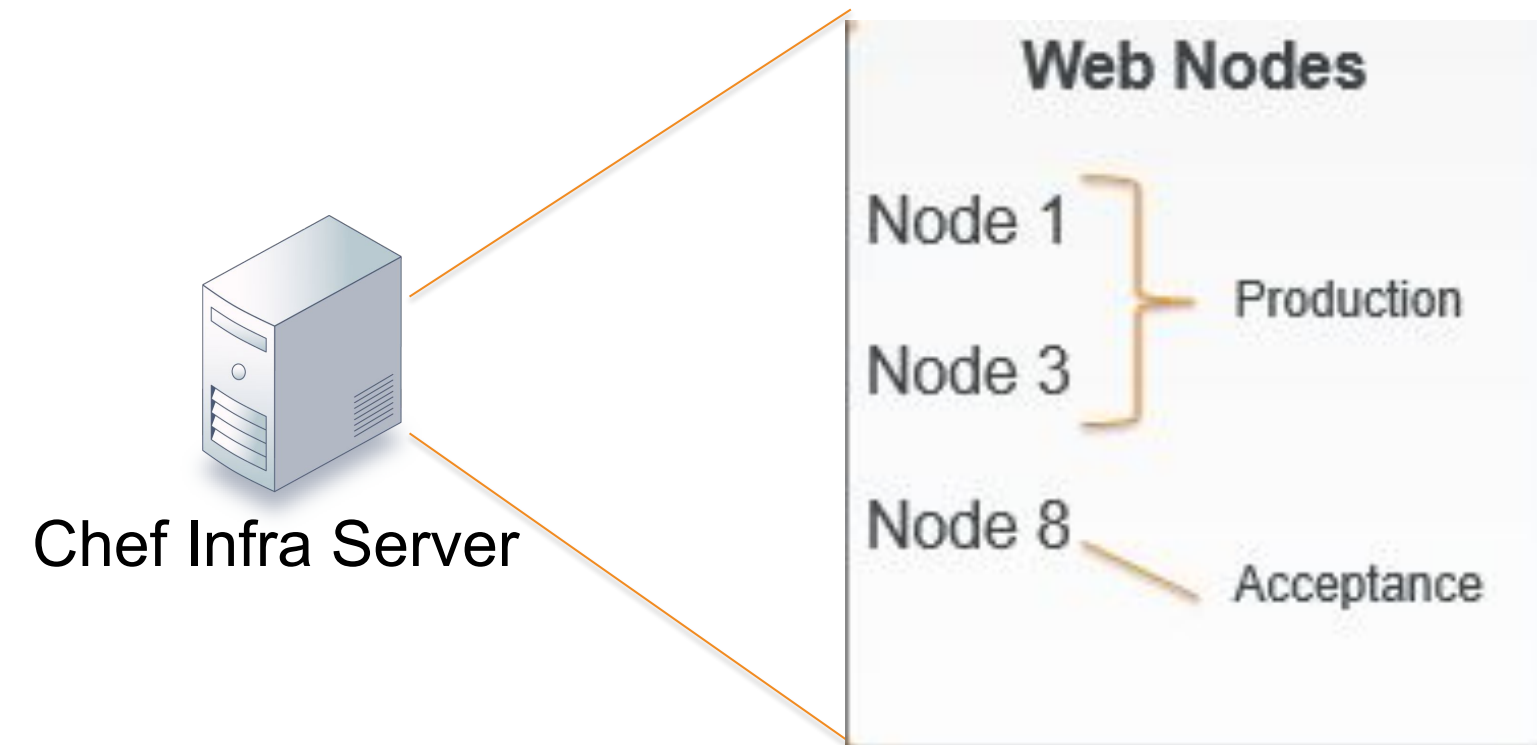


Policy Group = Environment

A policy group can best be defined as a logical separation of nodes.

For example, let's say you have a Production environment and an Acceptance environment.

You could create a **prod** policy group and an **acceptance** policy group and push your policyfile to either one, depending where you want that policyfile's node to reside.



Policy Group

The first time you specify a policy group, that policy group name will be instantiated in Chef Infra Server. For example:

`chef push prod apache.lock.json` will create the policy group named **prod** and also upload the `apache.lock.json` (and its cookbooks) to the Chef Infra Server.

It will upload that `apache.lock.json` within the policy group named **prod**.

Pushing a policyfile to Chef Infra Server

To push a policyfile to Chef Infra Server you need:

Policy name

and...

Policy group

These two items uniquely identify a single policyfile object on the Chef Infra Server.

GL: Push Your apache.lock.json to prod on Chef Infra Server



```
~/chef-repo/policyfiles> chef push prod apache.lock.json
```

```
Uploading policy apache (450fb23145) to policy group prod
```

```
Uploaded apache 0.1.0 (1388ab3a)
```

GL: Verify that Your Policy is on Chef Infra Server



```
> chef show-policy
```

```
apache
```

```
=====
```

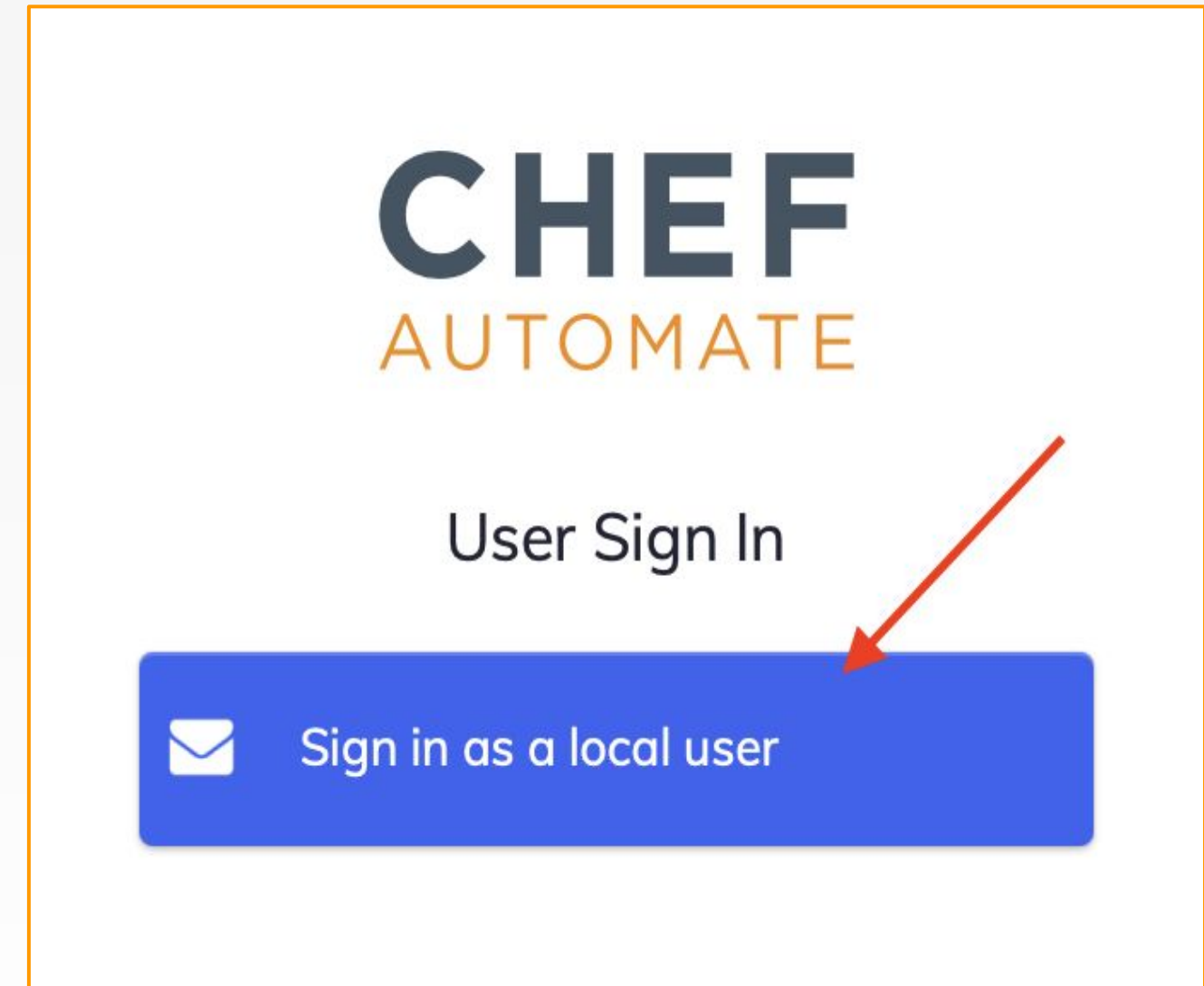
```
* prod: 40edea4d9e
```

This output confirms that the policy is on the Chef infra Server.



Chef Automate Web GUI: Policy (apache)

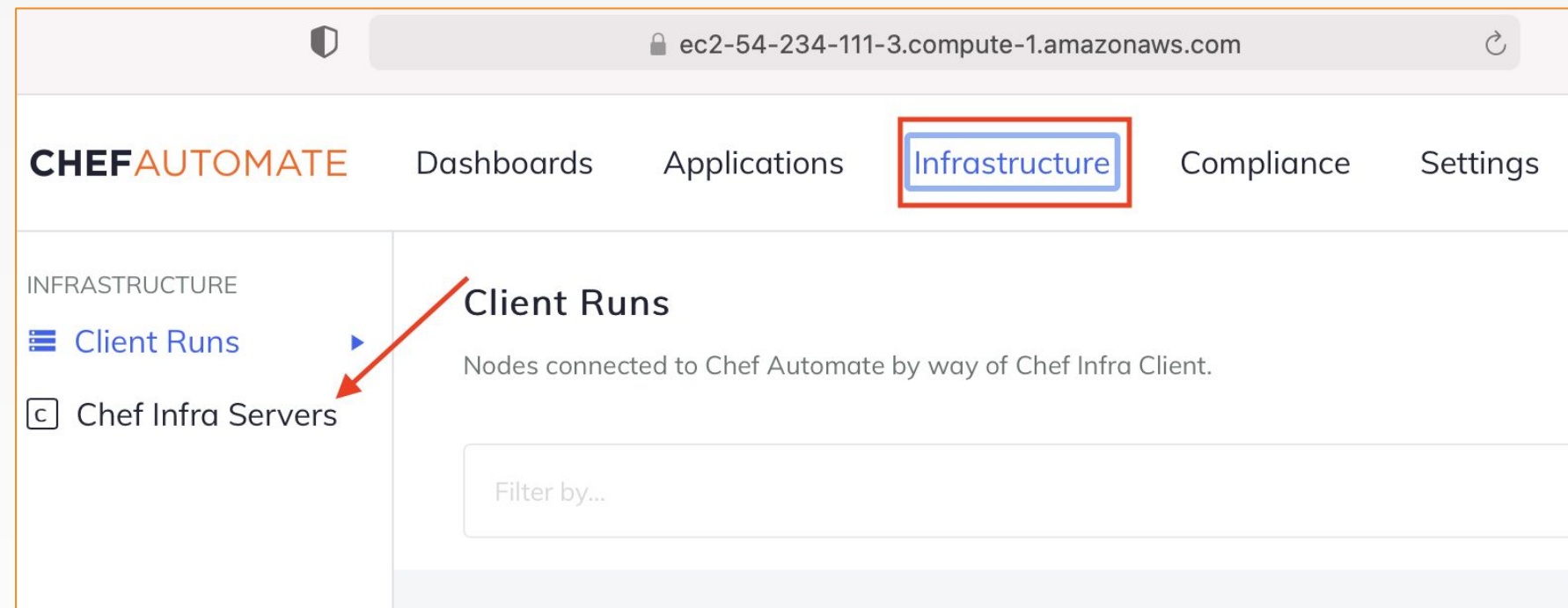
1. Sign In to Chef Automate server.
2. Click on Infrastructure tab and Go to Chef Infra Servers (Under Left Navigation Panel)
3. Click on Listed Chef Infra Server - 'cheftraining'
4. Click on the organization - 'studentxx'
5. Go to Policyfiles tab to view pushed policy.





Chef Automate Web GUI: Policy (apache)

1. Sign In to Chef Automate server.
2. Click on Infrastructure tab and Go to Chef Infra Servers (Under Left Navigation Panel)
3. Click on Listed Chef Infra Server - 'cheftraining'
4. Click on the organization - 'studentxx'
5. Go to Policyfiles tab to view pushed policy.





Chef Automate Web GUI: Policy (apache)

1. Sign In to Chef Automate server.
2. Click on Infrastructure tab and Go to Chef Infra Servers (Under Left Navigation Panel)
3. Click on Listed Chef Infra Server - 'cheftraining'
4. Click on the organization - 'studentxx'
5. Go to Policyfiles tab to view pushed policy.

Chef Infra Servers

Manage Chef Infra Servers with Chef Automate.

Name	FQDN	IP Address	Number Of Orgs
cheftraining	ec2-54-234-111-3.compute-1.amazonaws.com	54.234.111.3	30



Chef Automate Web GUI: Policy (apache)

1. Sign In to Chef Automate server.
2. Click on Infrastructure tab and Go to Chef Infra Servers (Under Left Navigation Panel)
3. Click on Listed Chef Infra Server - 'cheftraining'
4. Click on the organization - 'studentxx'
5. Go to Policyfiles tab to view pushed policy.

[Chef Infra Servers](#) > cheftraining

cheftraining

FQDN	IP Address
ec2-54-234-111-3.compute-1.amazonaws.com	54.234.111.3

Orgs Details

Name	Admin	Projects
student01	student01	student01project



Chef Automate Web GUI: Policy (apache)

1. Sign In to Chef Automate server.
2. Click on Infrastructure tab and Go to Chef Infra Servers (Under Left Navigation Panel)
3. Click on Listed Chef Infra Server - 'cheftraining'
4. Click on the organization - 'studentxx'
5. Go to Policyfiles tab to view pushed policy.

The screenshot shows the Chef Automate web interface for the 'student01' organization. The 'Policyfiles' tab is selected and highlighted with a red box. Below the navigation tabs, a search bar contains the text 'apache'. A table lists the policyfiles, with the first entry 'apache' highlighted by a red arrow. The table has columns for 'Name' and 'Revision ID'.

Name	Revision ID
apache	450fb23145c2b9635544042d413ccf25f c644964aa2f46df6114a5c1a087a94f

knife node policy set Command

In a moment you will apply the policyfile to a node.

To do so you will use the `knife node policy set` command.

Syntax:

```
knife node policy set NODE POLICY_GROUP POLICY_NAME
```

https://docs.chef.io/workstation/knife_node/#policy-set

GL: Apply the apache Policy to Your Linux (apache_web) Node



```
> knife node policy set apache_web prod apache
```

```
Successfully set the policy on node apache_web
```

node name

policy_group

policy_name

'knife node policy set' takes 3 arguments:

- * node name
- * policy group
- * Policy name

GL: View Information About Your Linux Node



```
$ knife node show apache_web
```

```
Node Name:    apache_web
Policy Name:   apache
Policy Group:  prod
FQDN:         ip-172-31-90-167.ec2.internal
IP:           3.236.230.224
Run List:     recipe[apache::default]
Recipes:      apache::default
Platform:     centos 7.6.1810
Tags:
```

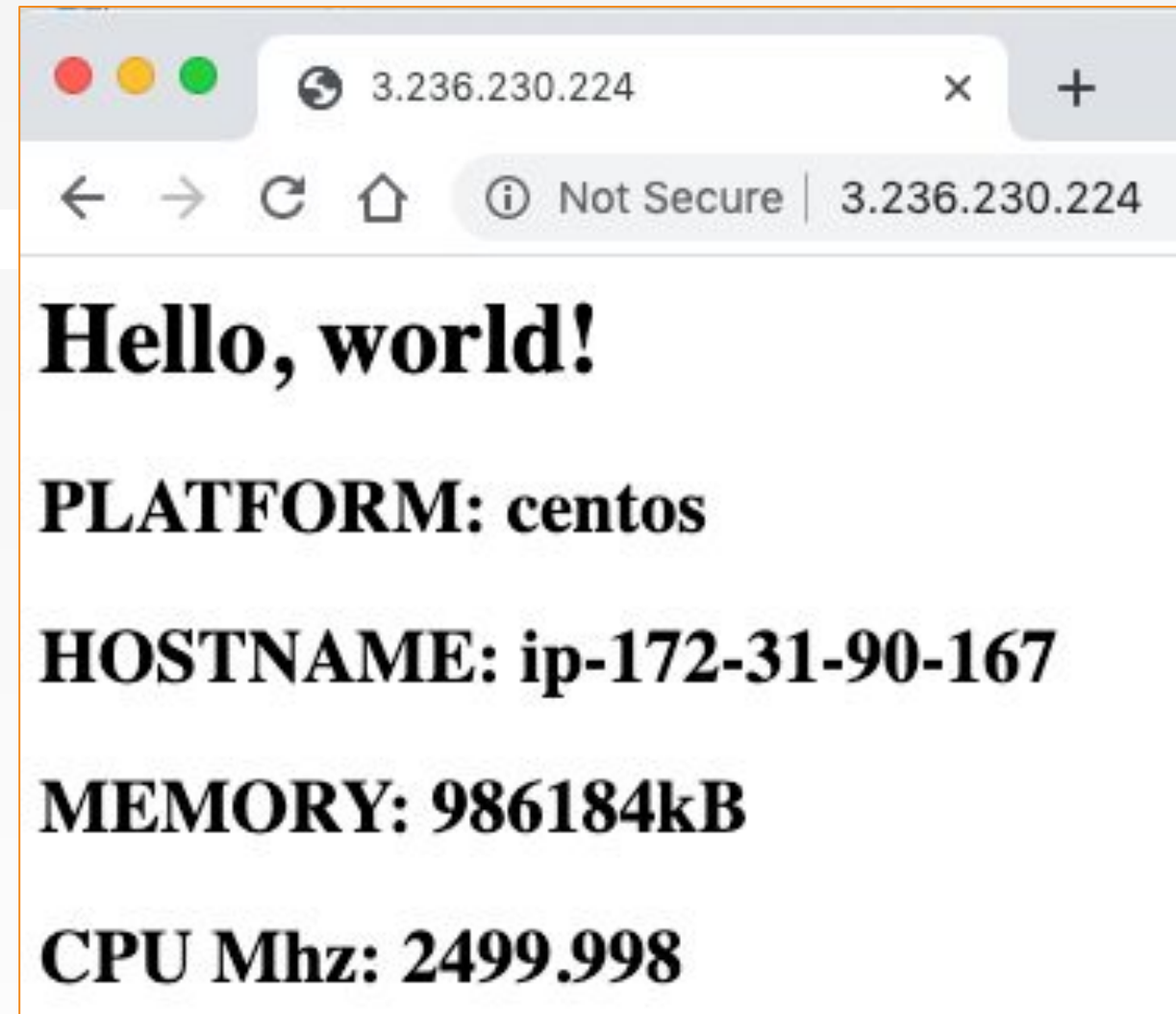

GL: Converge the Linux Node with ssh



```
$ knife ssh IPADDRESS -m -x chef -P PWD 'sudo chef-client'
```

```
3.236.230.224 Starting Chef Infra Client, version 17.3.48
3.236.230.224 Using policy 'apache' at revision
'450fb23145c2b9635544042d413ccf25fc644964aa2f46df6114a5c1a087a94f'
3.236.230.224 resolving cookbooks for run list: ["apache::default@0.1.0 (1388ab3)"]
3.236.230.224 Synchronizing Cookbooks:
3.236.230.224   - apache (0.1.0)
3.236.230.224 Installing Cookbook Gems:
3.236.230.224 Compiling Cookbooks...
3.236.230.224 Converging 3 resources
3.236.230.224 Recipe: apache::server
3.236.230.224   * yum_package[httpd] action install
3.236.230.224     - install version 0:2.4.6-93.el7.centos.x86_64 of package httpd
3.236.230.224   * template[/var/www/html/index.html] action create
3.236.230.224     - create new file /var/www/html/index.html
...
3.236.230.224 Chef Infra Client finished, 4/4 resources updated in 06 seconds
```

GL: Verify that the Linux Node Serves the Page





Review Questions

1. What do policyfiles typically contain?
2. What can a policyfile's `policy_name` be used for?
3. What is a policy group?



Q&A

What questions can we answer for you?



CHEF™