Policyfiles Overview

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

CHEF

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

Policyfile

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

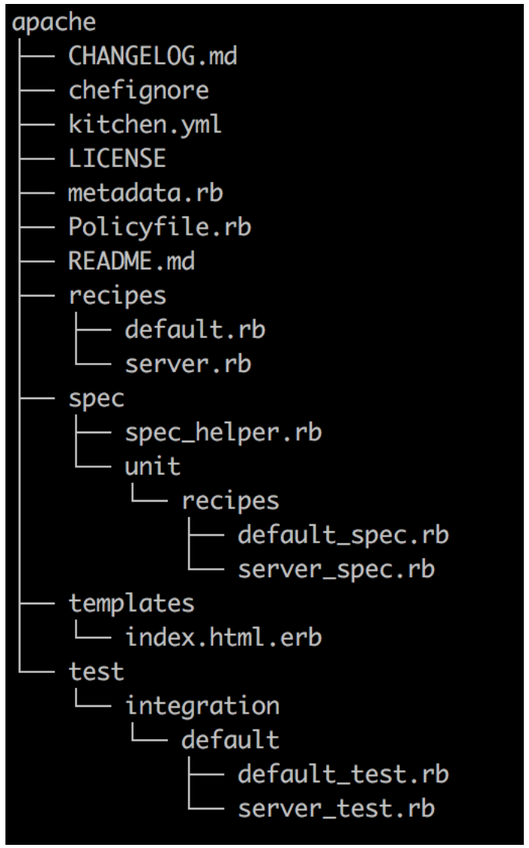
- Default source for fetching cookbooks
- Run list (or multiple run lists via named_run_list)
- Cookbook dependencies and sources
- Optional cookbook attributes
- Policy_name

Policyfiles

When you generate a Chef cookbook using a modern version of Chef Workstation, a Policyfile.rb file is automatically created.*

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

```
# Policyfile.rb - Describe how you want Chef Infra Client to
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with
name 'myiis'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order
run_list 'myiis::default'

# Specify a custom source for a single cookbook:
cookbook 'myiis', path: '.'

```

Policyfile

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

This will usually be the public, or a private, supermarket, or Chef server.

Policyfile.rb

```
# Policyfile.rb - Describe how you want Chef Infra Client to
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with
name 'myiis'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order
run_list 'myiis::default'

# Specify a custom source for a single cookbook:
cookbook 'myiis', path: '.'

```

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

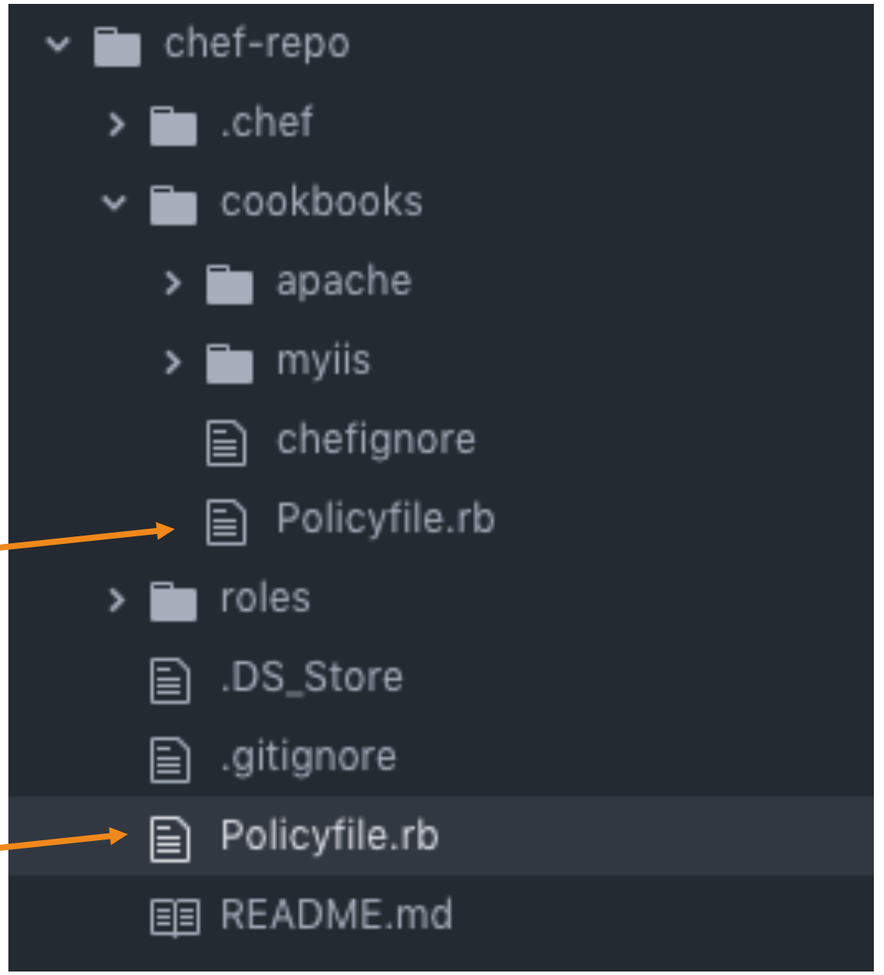
```
# Policyfile.rb - Describe how you want Chef Infra Client to
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with
name 'myiis'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order
run_list 'myiis::default'

# Specify a custom source for a single cookbook:
cookbook 'myiis', path: '.'

```

Policyfile Location

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

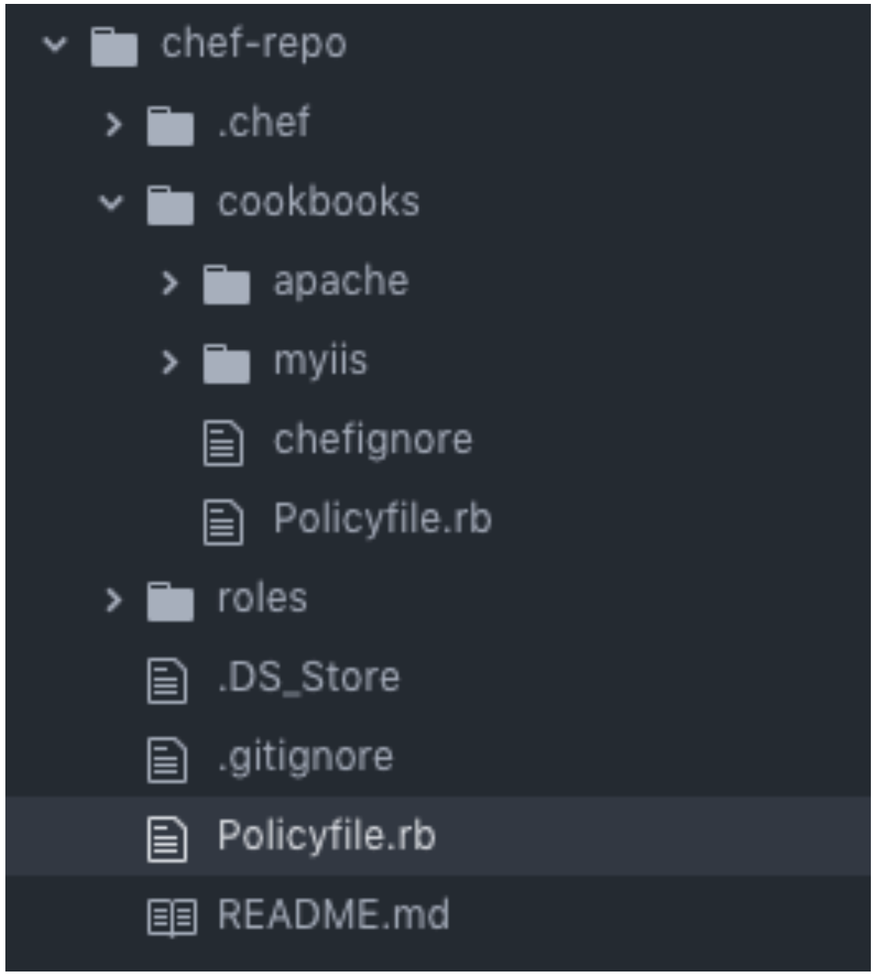
So we can use the auto-generated Policyfile as a guideline for creating our own Policyfile or simply ignore it.

Then we will generate a new Policyfile just above our cookbooks directory and also specify the path to the cookbooks directory so that any required cookbooks can be located by the Policyfile.



Policyfile 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.



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.

Example: Policyfile.lock

To generate the Policyfile.lock.json:

`chef install Policyfile.rb`

That will take the contents of your Policyfile.rb and convert it to the lock file.

Then you upload the Policyfile.lock.json to the Chef Infra Server like this:

`chef push policy_group myiis`

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

Policyfile.lock.json

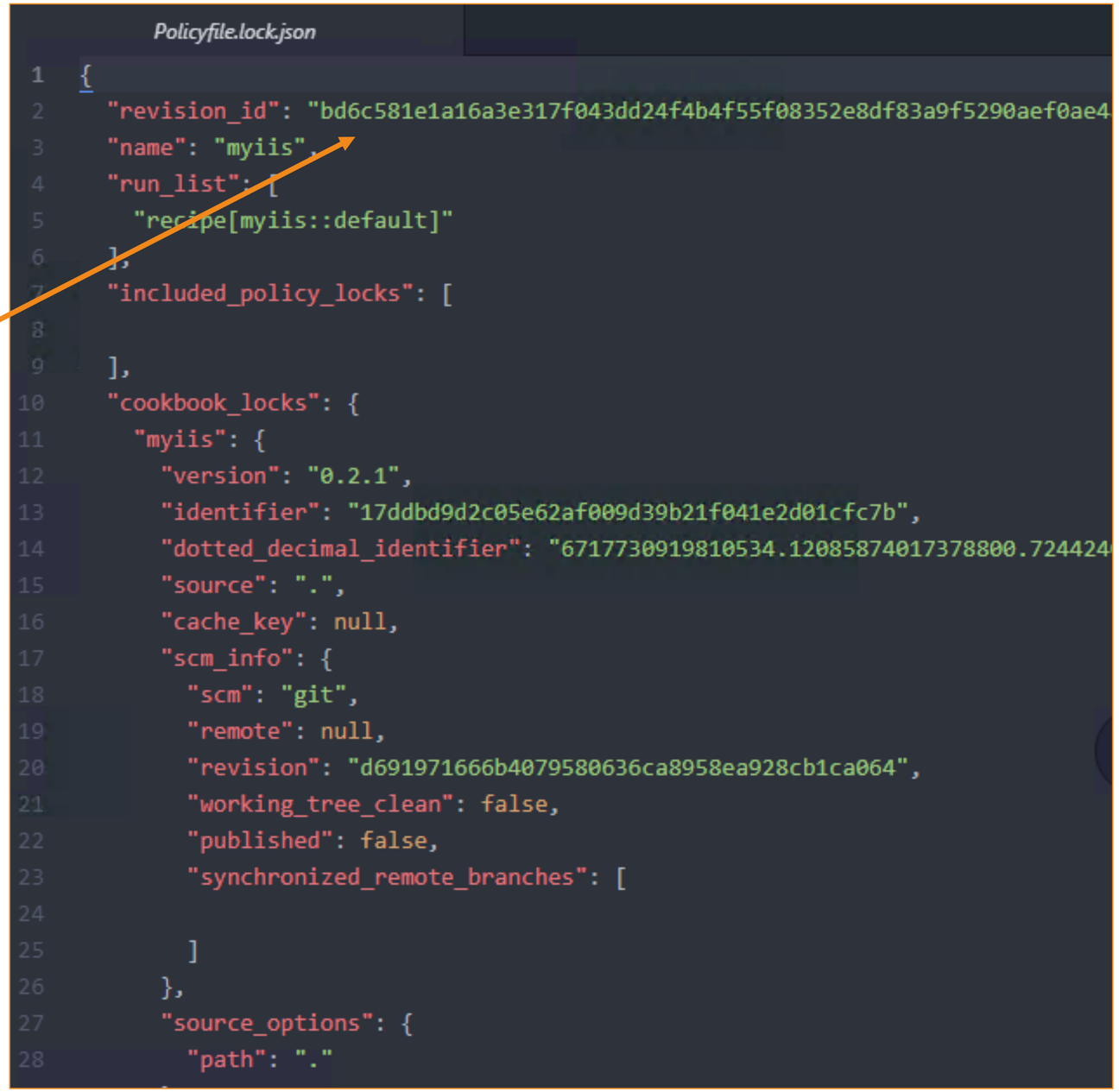

```
{
  "revision_id": "bd6c581e1a16a3e317f043dd24f4b4f55f08352e8df83a9f5290aef0ae4
  "name": "myiis",
  "run_list": [
    "recipe[myiis::default]"
  ],
  "included_policy_locks": [

  ],
  "cookbook_locks": {
    "myiis": {
      "version": "0.2.1",
      "identifier": "17ddbd9d2c05e62af009d39b21f041e2d01cfc7b",
      "dotted_decimal_identifier": "6717730919810534.12085874017378800.724424
      "source": ".",
      "cache_key": null,
      "scm_info": {
        "scm": "git",
        "remote": null,
        "revision": "d691971666b4079580636ca8958ea928cb1ca064",
        "working_tree_clean": false,
        "published": false,
        "synchronized_remote_branches": [

        ]
      },
      "source_options": {
        "path": "."
```


CHEF

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.



CHEF

policy_group

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 **dev**, **acceptance**, and **production**. You could also think of policy_group as a way to group like servers together.

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

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

It will upload that Policyfile.lock.json within the policy group name **`prod.`**

Example: Generating the Policyfile.lock.json

```
~\chef-repo> chef install policyfile.rb
Building policy myiis
Expanded run list: recipe[myiis::default]
Caching Cookbooks...
Installing myiis >= 0.0.0 from path

Lockfile written to
C:/Users/Administrator/cookbooks/myiis/Policyfile.lock.json

Policy revision id:
bd6c581e1a16a3e317f043dd24f4b4f55f08352e8df83a9f5290aef0ae4a3adf
```

Important: Do not run this command yet. This is just an example.

Example: Uploading the Policyfile.lock.json to Chef Infra Server

```
~\cookbooks> chef push prod myiis
```

```
Uploading policy myiis (bfd3af4697) to policy group prod Using myiis 0.0.0
(f37cdfc3)
```

This command uploads the **myiis** Policyfile.lock.json to the **prod** policy_group (a.k.a environment).

CHEF

Review Questions

1. What do policyfiles typically contain?

2. What can a policyfile's policy_name be used for?

Q&A

What questions can we answer for you?

CHEF

CHEF™