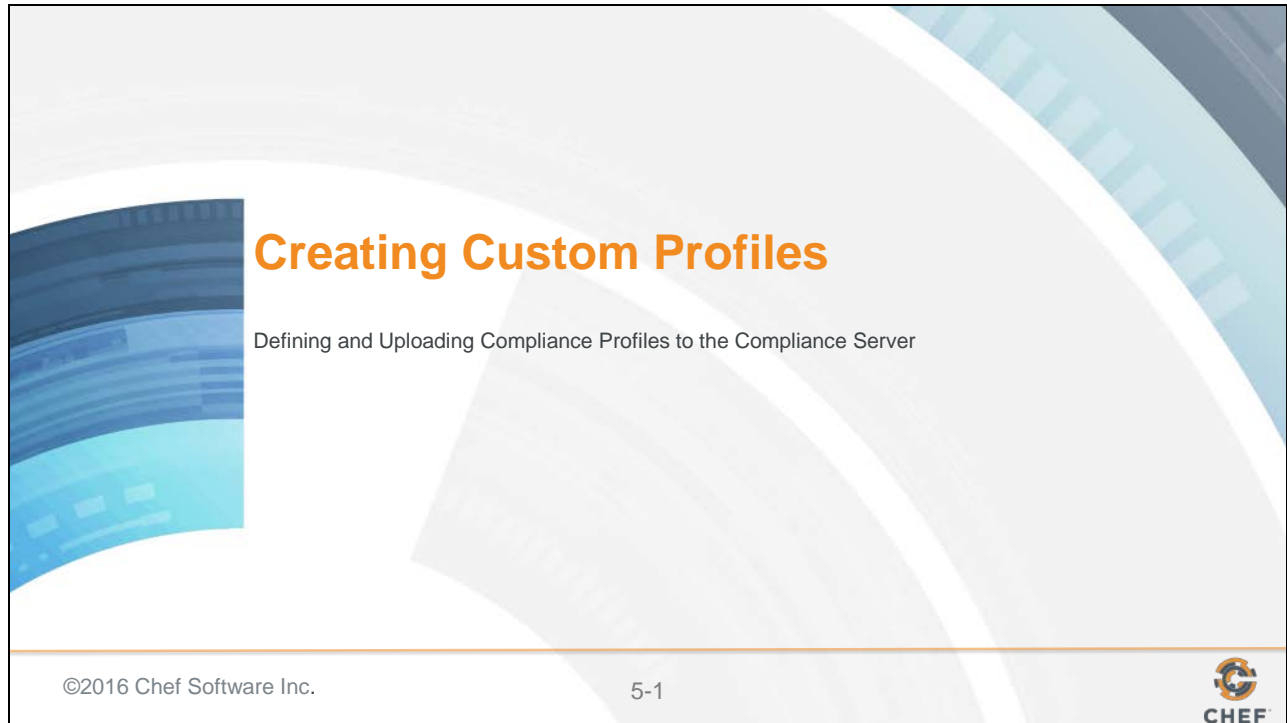


5: Creating Custom Profiles



Slide 2

Objectives

After completing this module, you should be able to:

- Write a custom compliance profile.
- Use InSpec to test your code and your custom profile.
- Upload a custom compliance profile to your Chef Compliance server.
- Test your custom profile.

Slide 3

CONCEPT



Creating a Custom Profile


In this section we will create a custom compliance profile.

Custom profiles are created using InSpec, just like the existing profiles were created.

After you have created a custom profile, you'll learn how to upload it to a Compliance Server and then use it to check for compliance issues.

Slide 4

CONCEPT



InSpec Command Line Interface

In this section we will use the InSpec command line interface (CLI) to help us create Compliance profiles and run audit tests against targets.


The InSpec CLI commands can run audit tests against targets using SSH, WinRM, locally, or on Docker containers.

We'll be using ``inspec init``, ``inspec check`` and ``inspec exec``.

- ``inspec init`` streamlines the creation of new Compliance profiles.

©2016 Chef Software Inc.

5-4



``inspec init`` can create all the directories and basic files that the Compliance Server and ``inspec check`` and ``inspec exec`` require.

Slide 5

CONCEPT




InSpec Command Line Interface

We'll be using ``inspec init``, ``inspec check`` and ``inspec exec``.

- ``inspec check`` just verifies the compliance profile code that you write --it doesn't actually test a system.
- ``inspec exec`` will run the tests against a system.

Slide 6

EXERCISE




Group Lab: Creating a Custom Profile

Creating custom profiles to fit your business needs.

Objective:

- ☐ Create a custom profile.
- ☐ Test your profile with InSpec

©2016 Chef Software Inc. 5-6 

The custom profile you will create will scan nodes to ensure they have a '/tmp' directory and that directory should be owned by the root user.

Note: In the workplace you would likely perform these custom profile tasks on your local workstation and upload them to the Compliance Server. In this class we'll use our target nodes as a workstation to create the profile on since they already have Chef installed on them. Then we'll ultimately upload the customer profile to your Compliance Server.

Slide 7

GL: Using `inspec help`



```
$ inspec help
```

```
[chef@ip-172-31-0-65 ~]$ inspec help
```

```
Commands:
```

```
inspec archive PATH           # archive a profile to tar.gz (defau...
inspec check PATH             # verify all tests at the specified ...
inspec compliance SUBCOMMAND ... # Chef Compliance commands
inspec detect                 # detect the target OS
inspec exec PATHS             # run all test files at the specifie...
inspec help [COMMAND]        # Describe available commands or one...
inspec init TEMPLATE ...     # Scaffolds a new project
inspec json PATH              # read all tests in PATH and generat...
inspec shell                  # open an interactive debugging shell
inspec version                 # prints the version of this tool
```

From your target Linux node/virtual workstation and from your home directory, run `inspec help`. Notice the `inspec` commands and sub commands that are available.

Slide 8

GL: Using `inspec init` help`



```
$ inspec init help
```

```
[chef@ip-172-31-0-65 ~]$ inspec init help
```

```
Commands:
```

```
  inspec init help [COMMAND]  # Describe subcommands or one specific subcommand
```

```
  inspec init profile NAME    # Create a new profile
```

```
# prints the version of this tool
```

The `inspec init` command enables you to create new Compliance profiles with less manual intervention than in previous versions of inspec and Chef Compliance.

Slide 9

GL: Create the Compliance Profile Directories and Files



```
$ inspec init profile secureprofile_01
```

```
Create new profile at /home/chef/secureprofile_01
```

- * Create file README.md
- * Create directory libraries
- * Create directory controls
- * Create file controls/example.rb
- * Create file inspec.yml

Notice the directories and the files that `inspec init profile` creates. The `secureprofile_01` is merely the name of the profile and could be named any way that makes sense in your organization.

Slide 10

GL: View the Compliance Profile Directories and Files



```
$ tree secureprofile_01
```

```
secureprofile_01
├── controls
│   └── example.rb
├── inspec.yml
├── libraries
└── README.md
```

As you can see, the ``tree secureprofile_01`` command shows the new directories and files that inspec requires.

Slide 11

GL: View the inspec.yml File



```
$ cat ~/secureprofile_01/inspec.yml
```

```
name: secureprofile_01
title: InSpec Profile
maintainer: The Authors
copyright: The Authors
copyright_email: you@example.com
license: All Rights Reserved
summary: An InSpec Compliance Profile
version: 0.1.0
```

Let's read the inspec.yml file by issuing `~/secureprofile_01/inspec.yml`.

In the workplace you should modify this file but we'll leave it as-is for now.

GL: Writing a Compliance Profile Control

Compliance profiles must be written within the `controls` directory.

```
secureprofile_01
├── controls
│   └── example.rb
├── inspec.yml
├── libraries
└── README.md
```

Slide 13

GL: Create the `tmp.rb` Control using the `cp` Command



```
$ cp ~/secureprofile_01/controls/example.rb  
~/secureprofile_01/controls/tmp.rb
```

In this example you will use the `cp` command to make a copy of the `example.rb` control and name the copy `tmp.rb`.

Slide 14

GL: Confirm Creation of tmp.rb using `tree`



```
$ tree secureprofile_01
```

```
|— controls  
|   |— example.rb  
|   |— tmp.rb  
|— inspec.yml  
|— libraries  
|— README.md
```

You should now see that `tmp.rb` resides next to the default `example.rb`.

Slide 15

GL: Edit the tmp.rb File - 1 of 3



~/secureprofile_01/controls/tmp.rb

```
# encoding: utf-8
# copyright: 2015, The Authors
# license: All rights reserved

title 'sample section'

# you can also use plain tests
describe file('/tmp') do
  it { should be_directory }
end

# you add controls here
control 'tmp-1.0' do
  impact 0.7
  title 'Create /tmp directory'
  desc 'An optional description...'
  describe file('/tmp') do
    it { should be_directory }
  end
end

# A unique ID for this control
# The criticality, if this control fails.
# A human-readable title
# The actual test
```

Delete everything highlighted in this example.

GL: Edit the tmp.rb File - 2 of 3

 ~/secureprofile_01/controls/tmp.rb

```
# encoding: utf-8
# copyright: 2015, The Authors
# license: All rights reserved
title '/tmp profile'

control "tmp-1.0" do
  impact 0.3
  title "Create /tmp directory"
  desc "A /tmp directory must exist"
  describe file('/tmp') do
    it { should be_directory }
  end
end
end
```

Write the first half of this control as shown here. You'll write the second half below this part in a moment.

Instructor Note: It's generally correct to use single quotes unless string interpolation is used, in which doubles are correct. tbd - Check with inspec team about describe statement correctness. We need to make this example consistent wrt quotes.

Slide 17

GL: Edit the tmp.rb File - 3 of 3

~/compliance_profiles/profile_01/test/tmp.rb

```
...  
control "tmp-1.1" do  
  impact 0.3  
  title "/tmp directory is owned by the root user"  
  desc "The /tmp directory must be owned by the root user"  
  describe file('/tmp') do  
    it { should be_owned_by 'root' }  
  end  
end
```

Write the second half of this control as shown here, just below the code you wrote on the preceding slide. We've pasted the entire control code below so you can see it better.

```
# encoding: utf-8  
# copyright: 2015, The Authors  
# license: All rights reserved  
title '/tmp profile'  
  
control "tmp-1.0" do  
  
  impact 0.3  
  title "Create /tmp directory"  
  desc "A /tmp directory must exist"  
  describe file('/tmp') do  
    it { should be_directory }  
  end  
end  
  
control "tmp-1.1" do  
  impact 0.3  
  title "/tmp directory is owned by the root user"  
  desc "The /tmp directory must be owned by the root user"  
  describe file('/tmp') do  
    it { should be_owned_by 'root' }  
  end  
end
```

Slide 18

GL: Use `inspec check` to Verify Your Profile



```
$ inspec check secureprofile_01/
```

Summary

Location: secureprofile_01/

Profile: secureprofile_01

Controls: 3

Timestamp: 2016-02-26T21:24:52+00:00

Valid: true

Errors

Warnings

Now use 'inspec check' to verify the compliance profile code that you wrote. You should see no errors or warnings.

GL: Use `inspec exec` to Verify Your Profile



```
$ inspec exec secureprofile_01/
```

```
....
```

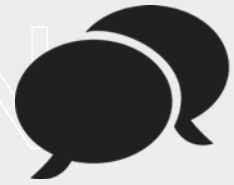
```
Finished in 0.0467 seconds (files took 1.47 seconds to load)
```

```
4 examples, 0 failures
```

Now use 'inspec exec' to test your compliance profile against the node you are working on. You should see no failures.

Since your profile has passed the inspec tests, it is now ready to be uploaded to the Compliance Server. We can assume this new compl

DISCUSSION



Creating a Custom Profile

In the preceding group lab you created a custom Compliance profile and tested your profile with InSpec.

Your code passed the ``inspec check`` test and your system passed the ``inspec exec`` test.

But what would an ``inspec exec`` failure look like?

Slide 21

Example of an `inspec exec` Failure

Let's say you modified your
~ secureprofile_01/controls/tmp.rb
and changed ``should be_owned_by``
`root` to ``should be_owned_by other``
and then ran ``inspec exec`` against that
file...

```
...
control "tmp-1.1" do
  impact 0.3
  title "/tmp directory is owned by the root user"
  desc "The /tmp directory must be owned by the root user"
  describe file('/tmp') do
    it { should be_owned_by other }
  end
end
```

Slide 22

Example: `inspec exec` Failure



```
$ inspec exec secureprofile_01/
```

```
...
Failures:

  1) File /tmp should be owned by "other"
     Failure/Error: DEFAULT_FAILURE_NOTIFIER = lambda { |failure, _opts| raise failure }
       expected `File /tmp.owned_by?("other")` to return true, got false
     # secureprofile_01/controls/tmp.rb:20:in `block (3 levels) in load'

Finished in 0.06284 seconds (files took 1.4 seconds to load)
4 examples, 1 failure

Failed examples:

rspec # File /tmp should be owned by "other"
```

...this is an example of running `inspec exec` against the system using the `~/secureprofile_01/controls/tmp.rb` as modified on the preceding slide.

As you can see, based on the modified control, `inspec exec` expected the /tmp directory to be owned by `other` but in fact /tmp is owned by root.

Instructor Note: As an optional lab, you can have the students modify their `~/secureprofile_01/controls/tmp.rb` as shown on the previous slide and then run the command on this slide. If you do, make sure they change the `~/secureprofile_01/controls/tmp.rb` back so it checks that /tmp is owned by root.

Slide 23

DISCUSSION




Uploading Custom Profiles to Compliance Server

inspec v 0.14.2 and above uses the ``inspec compliance upload PATH`` command to upload profiles from a workstation to the Compliance Server.

That command should be preceded by the ``inspec compliance login SERVER --password=PASSWORD --user=USER`` in order to first log in to the Compliance Server.

EXERCISE




Group Lab: Uploading the Custom Profile to the Compliance Server

Uploading it so it can be used in scans.

Objective:

- ☐ Upload your custom profile to the Compliance server.
- ☐ Run a scan from your Compliance server using your custom profile.

©2016 Chef Software Inc. 5-24 

In addition to the uploading procedure we'll do in the exercise, in the workplace you could also upload custom profiles using an API.

Slide 25

GL: Ensure You Are in the Home Dir



```
$ cd ~  
$ ls
```

```
secureprofile_01
```

From your Linux node, ensure you are in your Home directory and type `ls` to see your compliance profile

GL: Using 'inspec compliance' Commands



```
$ inspec compliance help
```

Commands:

```
inspec compliance exec PROFILE           # executes ...
inspec compliance help [COMMAND]         # Describe ...
inspec compliance login SERVER --password=PASSWORD --user=USER # Log in to...
inspec compliance logout                  # user logo...
inspec compliance profiles                # list all ...
inspec compliance upload PATH             # uploads a...
inspec compliance version                 # displays ...
```

This example shows the options for the `inspec compliance` command. You'll be using the `inspec compliance login SERVER --password=PASSWORD --user=USER` in a moment to first log into your Compliance server.

Slide 27

GL: Logging in to your Compliance Server



```
$ inspec compliance login https://SERVER/api --user=admin --  
password='admin' --insecure
```

```
Successfully authenticated  
[chef@ip-172-31-0-65 ~]$
```

Use your compliance server's IP address in place of SERVER in this example.

Note that the credentials used here are the credentials you created for your Compliance Server UI (admin/admin), not the node's.

Note: We are using the `--insecure` option in this lab because we are not using valid self-signed certificates.

GL: Viewing Your Custom Profile Tree



```
$ tree secureprofile_01
```

```
secureprofile_01
├── controls
│   ├── example.rb
│   └── tmp.rb
├── inspec.yml
├── libraries
└── README.md
```

Notice that even though our VM is now logged into the Compliance server, commands such as `tree` are still executed against the VM we are on.

GL: Viewing Compliance Profiles on Your Compliance Server



```
$ inspec compliance profiles
```

```
Available profiles:
```

```
-----
```

```
* admin/profile  
* admin/profile-cis-3.1  
* admin/profile1  
* base/apache  
* base/linux  
* base/mysql  
* base/postgres  
* base/ssh  
* base/windows  
* cis/cis-ubuntu-level1  
* cis/cis-ubuntu-level2
```

Again, this `inspec compliance profiles` command is executed against the VM we are on.

GL: Uploading Your Custom Profile



```
$ inspec compliance upload secureprofile_01
```

```
I, [2016-03-02T14:56:18.714517 #29237] INFO -- : Checking profile in secureprof      ile_01
I, [2016-03-02T14:56:18.714883 #29237] INFO -- : Metadata OK.
I, [2016-03-02T14:56:18.842213 #29237] INFO -- : Found 3 controls.
I, [2016-03-02T14:56:18.842320 #29237] INFO -- : Verify all controls in control      s/example.rb
I, [2016-03-02T14:56:18.842380 #29237] INFO -- : Verify all controls in control      s/tmp.rb
I, [2016-03-02T14:56:18.842430 #29237] INFO -- : Control definitions OK.
Profile is valid
Generate temporary profile archive at /tmp/secureprofile_0120160302-29237-lhmvbe      8.tar.gz
I, [2016-03-02T14:56:18.925296 #29237] INFO -- : Generate archive /tmp/securepr      ofile_0120160302-29237-
lhmvbe8.tar.gz.
I, [2016-03-02T14:56:18.933640 #29237] INFO -- : Finished archive generation.
Start upload to admin/secureprofile_01
Uploading to https://54.152.196.46/api/owners/admin/compliance/secureprofile_01/      tar
Successfully uploaded profile
```

This `inspec compliance upload secureprofile_01` command is now uploading our custom profile to the Compliance server.

Slide 31

GL: Viewing Compliance Profiles on Your Compliance Server



```
$ inspect compliance profiles
```

```
Available profiles:
```

```
-----
```

```
* admin/profile  
* admin/profile-cis-3.1  
* admin/profile1  
* admin/secureprofile_01  
* base/apache  
* base/linux  
* base/mysql  
* base/postgres  
* base/ssh  
* base/windows  
* cis/cis-ubuntu-level1  
* cis/cis-ubuntu-level2
```

This `inspect compliance profiles` is now executing against our Compliance server, thus we are now looking at the Compliance profiles that reside on the Compliance server. Notice that your secureprofile_01 has been uploaded to the compliance server.

GL: Logging out of Your Compliance server



```
$ inspec compliance logout
```

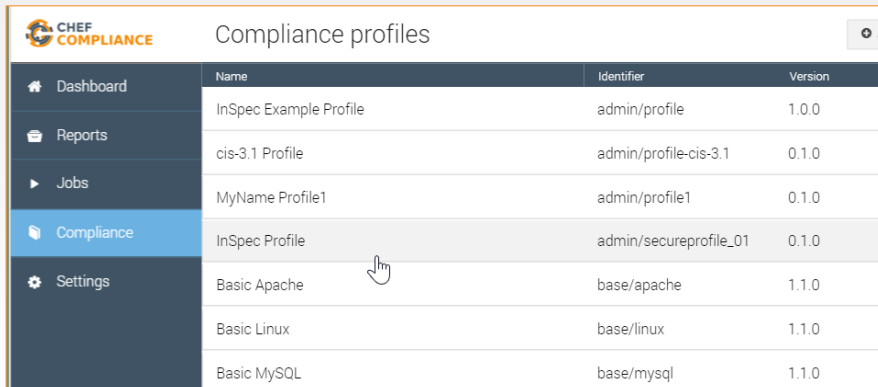
```
Successfully logged out
```


Slide 33

GL: Viewing Your Uploaded Custom Profile

Use a web browser to navigate the Compliance tab of your Compliance server.

Notice that your custom profile is present.

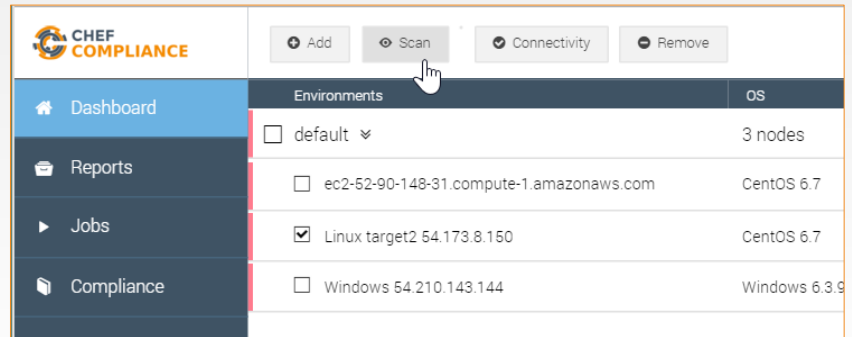


| CHEF COMPLIANCE | | Compliance profiles | | |
|-----------------|------------------------|------------------------|---------|--|
| | Name | Identifier | Version | |
| Dashboard | InSpec Example Profile | admin/profile | 1.0.0 | |
| Reports | cis-3.1 Profile | admin/profile-cis-3.1 | 0.1.0 | |
| Jobs | MyName Profile1 | admin/profile1 | 0.1.0 | |
| Compliance | InSpec Profile | admin/secureprofile_01 | 0.1.0 | |
| Settings | Basic Apache | base/apache | 1.1.0 | |
| | Basic Linux | base/linux | 1.1.0 | |
| | Basic MySQL | base/mysql | 1.1.0 | |

Slide 34

GL: Testing Your Uploaded Custom Profile

Navigate to the Compliance dashboard, click your Linux target, and then click **Scan**.



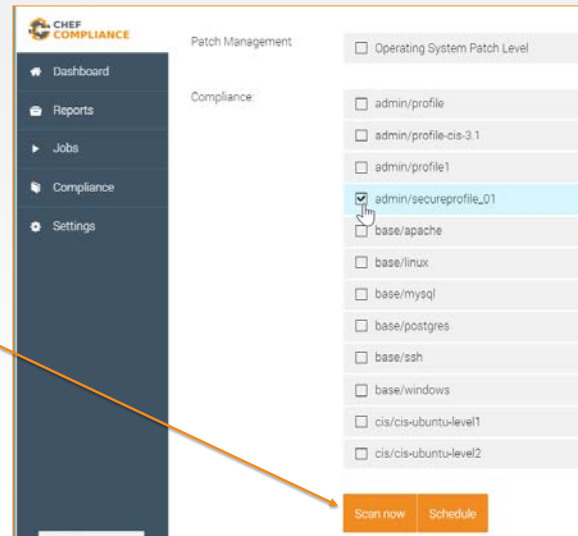
The screenshot shows the Chef Compliance dashboard. On the left is a sidebar with navigation links: Dashboard (selected), Reports, Jobs, and Compliance. The main area has a header with 'CHEF COMPLIANCE' and four buttons: Add, Scan (being clicked by a mouse cursor), Connectivity, and Remove. Below the header is a table titled 'Environments' with columns for 'Environments' and 'OS'. The table lists four environments: 'default' (3 nodes), 'ec2-52-90-148-31.compute-1.amazonaws.com' (CentOS 6.7), 'Linux target2 54.173.8.150' (CentOS 6.7, selected with a checkmark), and 'Windows 54.210.143.144' (Windows 6.3.9).

| Environments | OS |
|---|---------------|
| <input type="checkbox"/> default | 3 nodes |
| <input type="checkbox"/> ec2-52-90-148-31.compute-1.amazonaws.com | CentOS 6.7 |
| <input checked="" type="checkbox"/> Linux target2 54.173.8.150 | CentOS 6.7 |
| <input type="checkbox"/> Windows 54.210.143.144 | Windows 6.3.9 |

Slide 35

GL: Testing Your Uploaded Custom Profile

Select only your custom profile and then click **Scan now**.



Slide 36

GL: Testing Your Uploaded Custom Profile


You should now see that your custom profile works properly and your Linux target is in compliance.

Scan Report

| Hostname | Compliant | Minor Issues | Major Issues | Critical Issues | Skipped |
|--|-----------|--------------|--------------|-----------------|---------|
| Linux target2 54.173.8.150 | 2 | 0 | 0 | 0 | 0 |
| secureprofile_01: Create /tmp directory | Compliant | | | | ■ |
| File /tmp should be directory | | 7.0 | | | |
| secureprofile_01: /tmp directory is owned by the root user | Compliant | | | | ■ |
| File /tmp should be owned by "root" | | 3.0 | | | |

Slide 37

EXERCISE




Group Lab: Uploading the Custom Profile to the Compliance Server

Uploading it so it can be used in scans.

Objective:

- ✓ Upload your custom profile to the Compliance server.
- ✓ Run a scan from your Compliance server using your custom profile.

©2016 Chef Software Inc. 5-37 

We have now completed this group lab.

Slide 38

Review Questions

1. What is the difference between ``inspec check`` and ``inspec exec``?
2. What does ``inspec init profile`` do?

Instructor Note Answers:

1. ``inspec check`` just verifies the code--it doesn't actually test a system. ``inspec exec`` will run the tests against a system.

Additional questions will be added when we test this course on the Late February v1.0 Compliance software since `inspec.yml` will replace the `inspec` requirement of `metadata.rb`. Plus the new ``generate profile`` command will change the way we create profiles too.

2. It creates the directories and baseline files that are required for creating profiles.

Slide 39

