CHEF™

Getting Started with Compliance Automation

CHEF

CHEF™

Remediation

CHEF

InSpec: Turn security and compliance into code

- Translate compliance into Code
- Clearly express statements of policy
- Move risk to build/test from runtime
- Find issues early
- Write code quickly
- Run code anywhere
- Inspect machines, data and APIs

Part of a process of continuous compliance

A simple example of an InSpec CIS rule

```
control 'cis-1.4.1' do
  title '1.4.1 Enable SELinux in /etc/grub.conf'
  desc '
    Do not disable SELinux and enforcing in your GRUB configuration.
    These are important security features that prevent attackers from
    escalating their access to your systems. For reference see …
  '
  impact 1.0
  expect(grub_conf.param 'selinux').to_not eq '0'
  expect(grub_conf.param 'enforcing').to_not eq '0'
end
```

INSPEC

Objectives

After completing this module, you should be able to:

- Remediate a compliance issue.
- Test your remediation locally.
- Test for compliance with InSpec from the CLI
- Rescan the node and ensure compliance.

CONCEPT

Let's Remediate the Issue

Now that we've identified the ssh version issue, let's write a recipe on the target node to remediate the issue.

Then we'll run the compliance scan again to see if we successfully remediated the issue.

Note: In this course we will write a recipe directly on the node that we're running scans on. Of course in a production environment you will likely write such recipes locally and upload them to Chef Server. Then the nodes would convergence the recipes on their next chef-client run.

EXERCISE

GL: Remediating the Issue

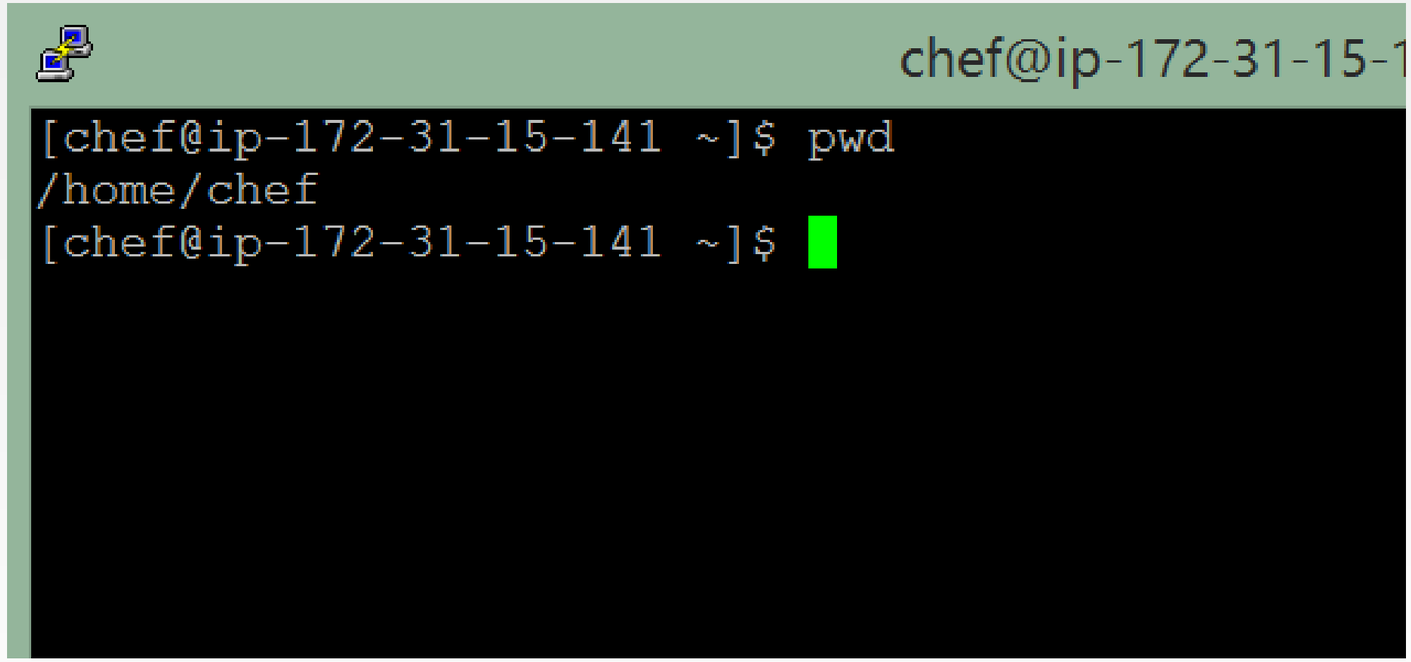
Objective:

- ❑ Start writing a remediation recipe on that node.
- ❑ Test the recipe locally.
- ❑ Test for compliance with InSpec from the command line interface (CLI)
- ❑ Converge the recipe.
- ❑ Rescan the node and ensure compliance.

GL: Remediating the Issue

Log in to your **target** node (not your compliance server node) using ssh and ensure you are in the **home directory**.

Note: emacs, nano, and vim/vi are installed on your Linux nodes. Some tips for using them can be found below in your participant guide.



GL: Create and Change to a ‘cookbooks’ Directory

```
$ mkdir -p cookbooks
$ cd cookbooks
```

From the home directory, create a `**cookbooks**` directory and navigate into it.

GL: Create an SSH Cookbook

```
$ chef generate cookbook ssh
```

```
Generating cookbook ssh
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. Type `cd ssh` to enter it.
…
```

GL: Create an SSH Server Recipe

```
$ chef generate recipe ssh server
```

```
Recipe: code_generator::recipe
  * directory[./ssh/spec/unit/recipes] action create (up to date)
  * cookbook_file[./ssh/spec/spec_helper.rb] action create_if_missing (up to date)
  * template[./ssh/spec/unit/recipes/server_spec.rb] action create_if_missing
    - create new file ./ssh/spec/unit/recipes/server_spec.rb
    - update content in file ./ssh/spec/unit/recipes/server_spec.rb from none to 301b96
    (diff output suppressed by config)
  * directory[./ssh/test/recipes] action create (up to date)
  * template[./ssh/test/recipes/server.rb] action create_if_missing
    - create new file ./ssh/test/recipes/server.rb
    - update content in file ./ssh/test/recipes/server.rb from none to e2c349
    (diff output suppressed by config)
  * template[./ssh/recipes/server.rb] action create
    - create new file ./ssh/recipes/server.rb
    - update content in file ./ssh/recipes/server.rb from none to adc474
    (diff output suppressed by config)
```

GL: Create an SSH Config Template

```
$ chef generate template ssh sshd_config.erb -s /etc/ssh/sshd_config
```

```
Recipe: code_generator::template
  * directory[./ssh/templates/default] action create
    - create new directory ./ssh/templates/default
  * file[./ssh/templates/sshd_config.erb] action create
    - create new file ./ssh/templates/sshd_config.erb
    - update content in file ./ssh/templates/sshd_config.erb from
none to 1c625d
    (diff output suppressed by config)
```

GL: Write the Server Recipe

```
$ ~/cookbooks/ssh/recipes/server.rb
```

```ruby
#
# Cookbook Name:: ssh
# Recipe:: server
#
# Copyright (c) 2016 The Authors, All Rights Reserved.

template '/etc/ssh/sshd_config' do
  source 'sshd_config.erb'
  owner 'root'
  group 'root'
  mode '0644'
end
```

EXERCISE

GL: Testing the Recipe

Objective:

✓ Write a remediation recipe on that node.

- ❑ Test the recipe locally.
- ❑ Test for compliance with InSpec from the command line interface (CLI)
- ❑ Converge the recipe.
- ❑ Rescan the node and ensure compliance.

CHEF

GL: Navigate to your SSH Cookbook

```
$ cd ~/cookbooks/ssh/
```

GL: Edit your .kitchen.yml -- Part 1

~/cookbooks/ssh/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero
```


CHEF

GL: Edit your .kitchen.yml -- Part 2

~/cookbooks/ssh/.kitchen.yml

```
platforms:
#  - name: ubuntu-14.04
  - name: centos-7.2

suites:
  - name: default
    run_list:
      - recipe[ssh::default]
    attributes:
```

GL: Edit your .kitchen.yml -- Part 3

~/cookbooks/ssh/.kitchen.yml

```
platforms:
#  - name: ubuntu-14.04
  - name: centos-7.2

suites:
  - name: server
    run_list:
      - recipe[ssh::server]
    attributes:
```

GL: Run `kitchen list` from ~/cookbooks/ssh/

```
$ kitchen list
```

```
Instance          Driver   Provisioner   Verifier   Transport   Last Action
server-centos-72  Docker   ChefZero      Inspec     Ssh         <Not Created>
```

GL: Run `delivery local deploy`

```
$ delivery local deploy
```

```
Chef Delivery
Running Deploy Phase
-----> Starting Kitchen (v1.11.1)
-----> Creating <server-centos-72>...
       Sending build context to Docker daemon 200.2 kB
       Sending build context to Docker daemon
       Step 0 : FROM centos:centos7
        ---> d83a55af4e75
 …
 Running handlers:
       Running handlers complete
       Chef Client finished, 1/1 resources updated in 01 seconds
       Finished converging <server-centos-72> (0m15.50s).
-----> Kitchen is finished. (0m18.12s)
```

CONCEPT

What We've Done So Far

In the preceding exercises, we began writing a remediation recipe on our target node.

We also tested the recipe locally.

But have we even addressed the "Set the SSH protocol version to 2" issue?

CHEF

EXERCISE

GL: Using InSpec for Verification

Objective:

- ✓ Write a remediation recipe on that node.
- ✓ Test the recipe locally.
- ❑ Test for compliance with InSpec from the command line interface (CLI)
- ❑ Converge the recipe .
- ❑ Rescan the node and ensure compliance.

InSpec Test

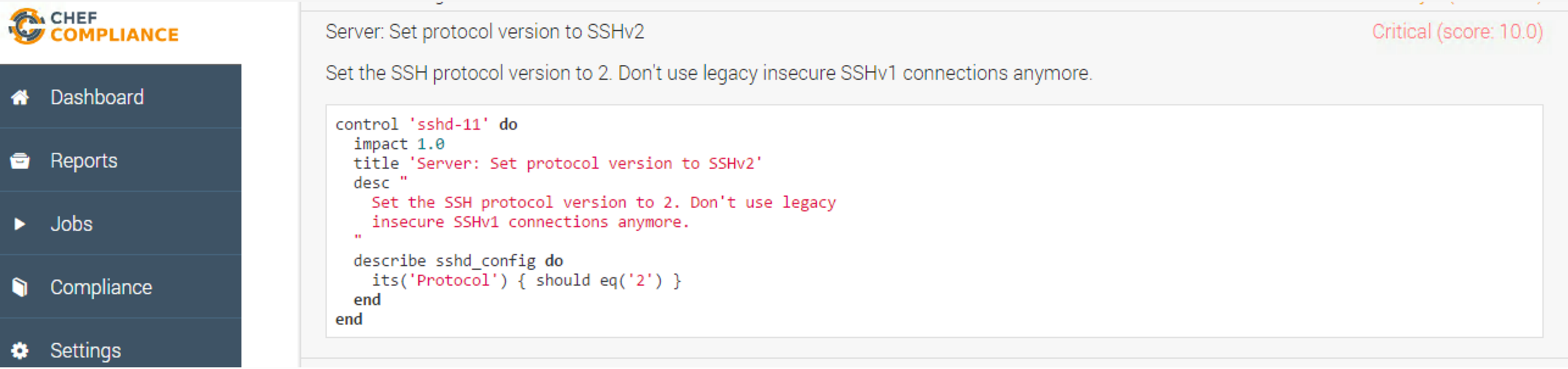
~/cookbooks/ssh/test/recipes/server.rb

```
control 'sshd-11' do
  impact 1.0
  title 'Server: Set protocol version to SSHv2'
  desc "
    Set the SSH protocol version to 2. Don't use legacy
    insecure SSHv1 connections anymore.
  "
  describe sshd_config do
    its('Protocol') { should eq('2') }
  end
end
```

Example of Creating the 'server.rb' file

One handy way to populate the preceding 'server.rb' is to use the Compliance Web UI and copy the InSpec code found in the relevant Compliance profile:

Compliance > Base SSH > Server: Set protocol version to SSHv2



CONCEPT

Running InSpec from the Command Line Interface (CLI)

InSpec is an executable application.

InSpec can execute on remote hosts, including docker containers.

You can use 'inspec exec' to run tests at a specified path.

CHEF

GL: What is your Docker ID?

```
$ sudo docker ps
```

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
5b51a4237437        d5b8fd3299b4        "/usr/sbin/sshd -D -     41 minutes ago      Up 41 minutes       0.0.0.0:32768->22/tcp    grave_davinci
```

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
5b51a4237437        d5b8fd3299b4        "/usr/sbin/sshd -D -     41 minutes ago
```

GL: Running InSpec from the CLI

```
$ inspec exec ~/cookbooks/ssh/test/recipes/server.rb -t
docker://b5813235642a
```

```
Target:
docker://b5813235642a3bb61912cf5cda47a02b2297d40c4ec4d62407580f1138acee
5b

  ✖  sshd-11: Server: Set protocol version to SSHv2 (
     expected: "2"
          got: nil

     (compared using ==)
     )

Summary: 0 successful, 1 failures, 0 skipped
```

GL: Update the Template

```
~/cookbooks/ssh/templates/sshd_config.erb
```

```
# Disable legacy (protocol version 1) support in the server for
new
# installations. In future the default will change to require
explicit
# activation of protocol 1
# Protocol 2
Protocol 2
```

GL: Ensure you are in ~/cookbooks/ssh

```
$ cd ~/cookbooks/ssh
```

GL: Run `delivery local deploy`

```
$ delivery local deploy
```

```
...
 Recipe: ssh::server
         * template[/etc/ssh/sshd_config] action create
           - update content in file /etc/ssh/sshd_config from 1c625d to 9a55f5
           --- /etc/ssh/sshd_config     2016-08-23 09:36:11.709616840 +0000
           +++ /etc/ssh/.chef-sshd_config20160823-542-9zyy0     2016-08-23 09:51:29.745616840
+0000
           @@ -18,7 +18,7 @@
            # Disable legacy (protocol version 1) support in the server for new
            # installations. In future the default will change to require explicit
            # activation of protocol 1
           -# Protocol 2,1
           +Protocol 2

            # HostKey for protocol version 1
            #HostKey /etc/ssh/ssh_host_key
```

GL: Running InSpec from the CLI

```
$ inspec exec ~/cookbooks/ssh/test/recipes/server.rb -t
docker://CONTAINER_ID
```

```
Target:
docker://b5813235642a3bb61912cf5cda47a02b2297d40c4e
c4d62407580f1138acee5b

  ✔  sshd-11: Server: Set protocol version to SSHv2

Summary: 1 successful, 0 failures, 0 skipped
```

Smoke test with delivery

```
$ delivery local smoke
```

```
Chef Delivery
Running Smoke Phase
-----> Starting Kitchen (v1.11.1)
-----> Setting up <server-centos-72>...
$$$$$$ Running legacy setup for 'Docker' Driver
       Finished setting up <server-centos-72> (0m0.00s).
-----> Verifying <server-centos-72>...
       Use `/home/chef/cookbooks/ssh/test/recipes/server` for testing

Target:  ssh://kitchen@localhost:32770

  ✔  sshd-11: Server: Set protocol version to SSHv2
  ○  User root should exist; User root This is an example test, r... (1 skipped)
     This is an example test, replace with your own test.
  ○  Port 80 should not be listening; Port 80 This is an example ... (1 skipped)
     This is an example test, replace with your own test.

Summary: 3 successful, 0 failures, 2 skipped
       Finished verifying <server-centos-72> (0m0.35s).
-----> Kitchen is finished. (0m0.93s)
```

GL: Apply the New SSH Recipe

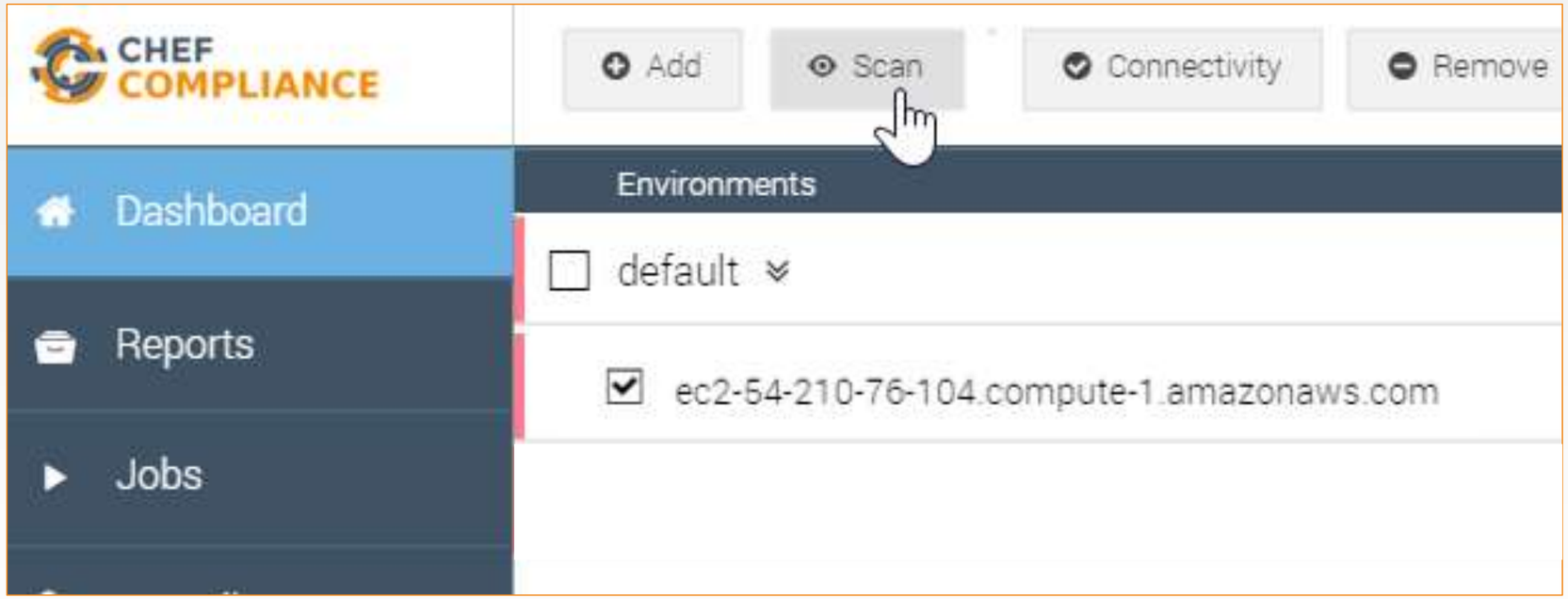
```
$ sudo chef-client --local-mode -r 'recipe[ssh::server]'
```

```
...
+++ /etc/ssh/.ssh_config20151209-10413-hlk9ow       2015-12-09
20:37:07.621689137 +0000
    @@ -37,7 +37,7 @@
     #   IdentityFile ~/.ssh/id_rsa
     #   IdentityFile ~/.ssh/id_dsa
     #   Port 22
    -#   Protocol 2,1
    +Protocol 2
     #   Cipher 3desesources updated in 3.29477735 seconds
```

GL: Re-run the Compliance Scan

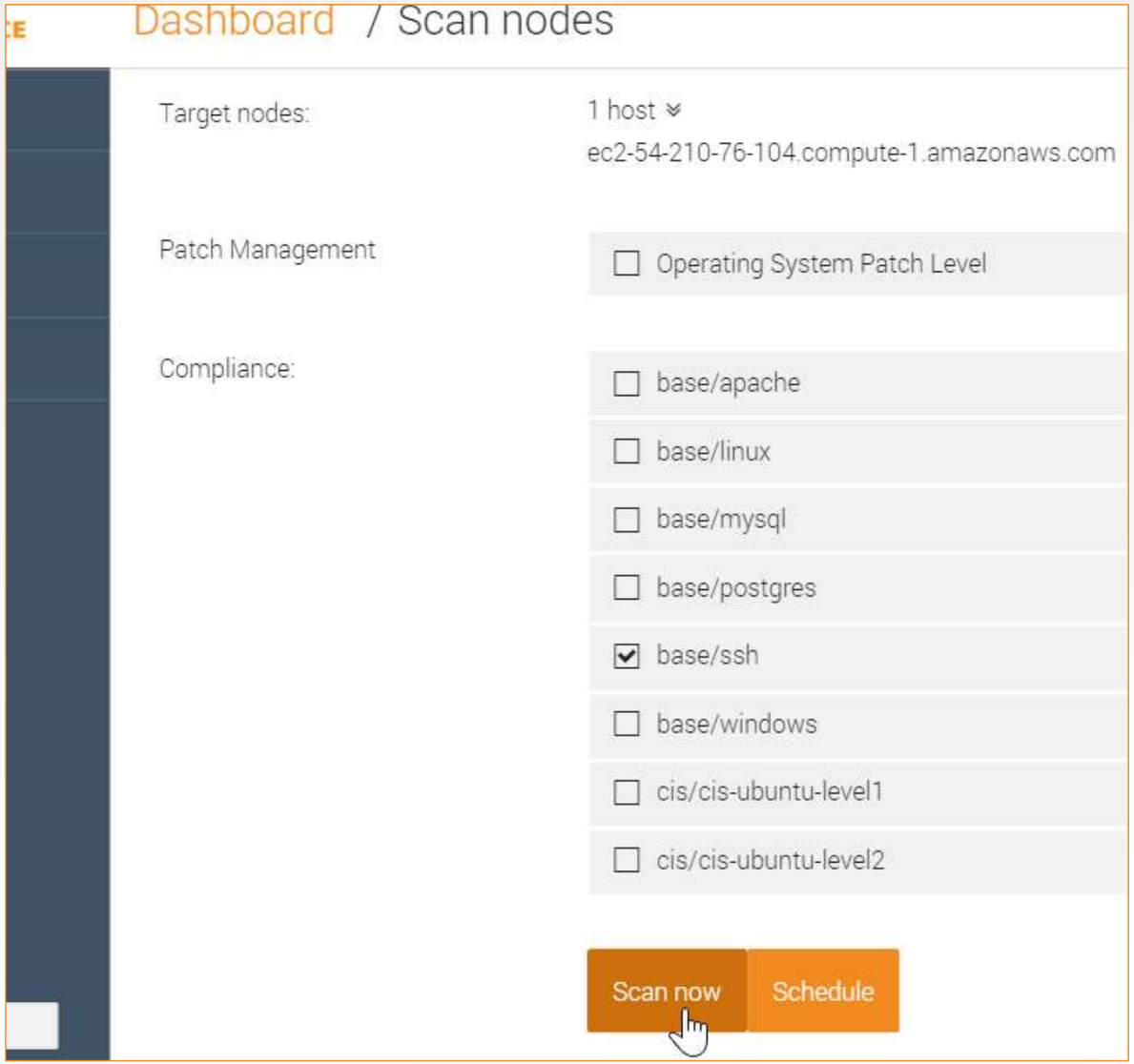
Return to the Compliance Web UI and re-run the scan on your target node.

Be sure to run only the base/ssh scan as shown on the next slide.



GL: Re-run the Compliance Scan

Run only the base/ssh scan.



GL: Results of this Exercise

Your scan should show that the ssh protocol issue is now complaint.

base/ssh: Server: Specify a valid address family	Minor Issues
base/ssh: /etc/ssh should be a directory	Compliant
base/ssh: /etc/ssh should be owned by root	Compliant
base/ssh: sshd_config should be owned by root	Compliant
base/ssh: sshd_config should not be writable/executable to others	Compliant
base/ssh: Client: Set SSH protocol version to 2	Compliant
SSH Configuration Protocol should eq "2"	10.0
base/ssh: Server: Set protocol version to SSHv2	Compliant
base/ssh: Server: Do not permit root-based login	Compliant
base/ssh: sshd_config should not be group-writable/executable	Compliant
base/ssh: sshd_config should not be group-writable/executable	Compliant
base/ssh: Server: Disable challenge-response authentication	Compliant
base/ssh: sshd_config should not be accessible to others	Compliant

DISCUSSION

Conclusion

- ✓ Log in to your target node.
- ✓ Write a remediation recipe on that node.
- ✓ Test the recipe locally.
- ✓ Test for compliance with InSpec from the CLI
- ✓ Converge the recipe.
- ✓ Rescan the node and ensure compliance.

Review Questions

1. When adding a node to the Compliance server's dashboard, should you use the node's FQDN or just its IP address?

2. What can `inspec exec` be used for?

3. How are compliance severities defined?

4. Using the image on the right, what section is the actual test?

```
control 'ssh-4' do
  impact 1.0
  title 'Client: Set SSH protocol version to 2'
  desc "
    Set the SSH protocol version to 2. Don't use legacy
    insecure SSHv1 connections anymore.
  "
  describe ssh_config do
    its('Protocol') { should eq('2') }
  end
end
```

Review Questions

5. If a compliance scan tells you that a node is unreachable, what might you use to troubleshoot the connection?

6. What language is used to define controls?

CHEF™