Audit a Linux node for compliance

Compliance

— Analyze
— Specify
— Test
— Certify

Analyze

Be clear about your compliance requirements and the desired state of your infrastructure.

Specify

Translate your desired state into a formal language that precisely specifies your requirements.

Test

Verify whether the actual state of your infrastructure meets the desired state.

Certify

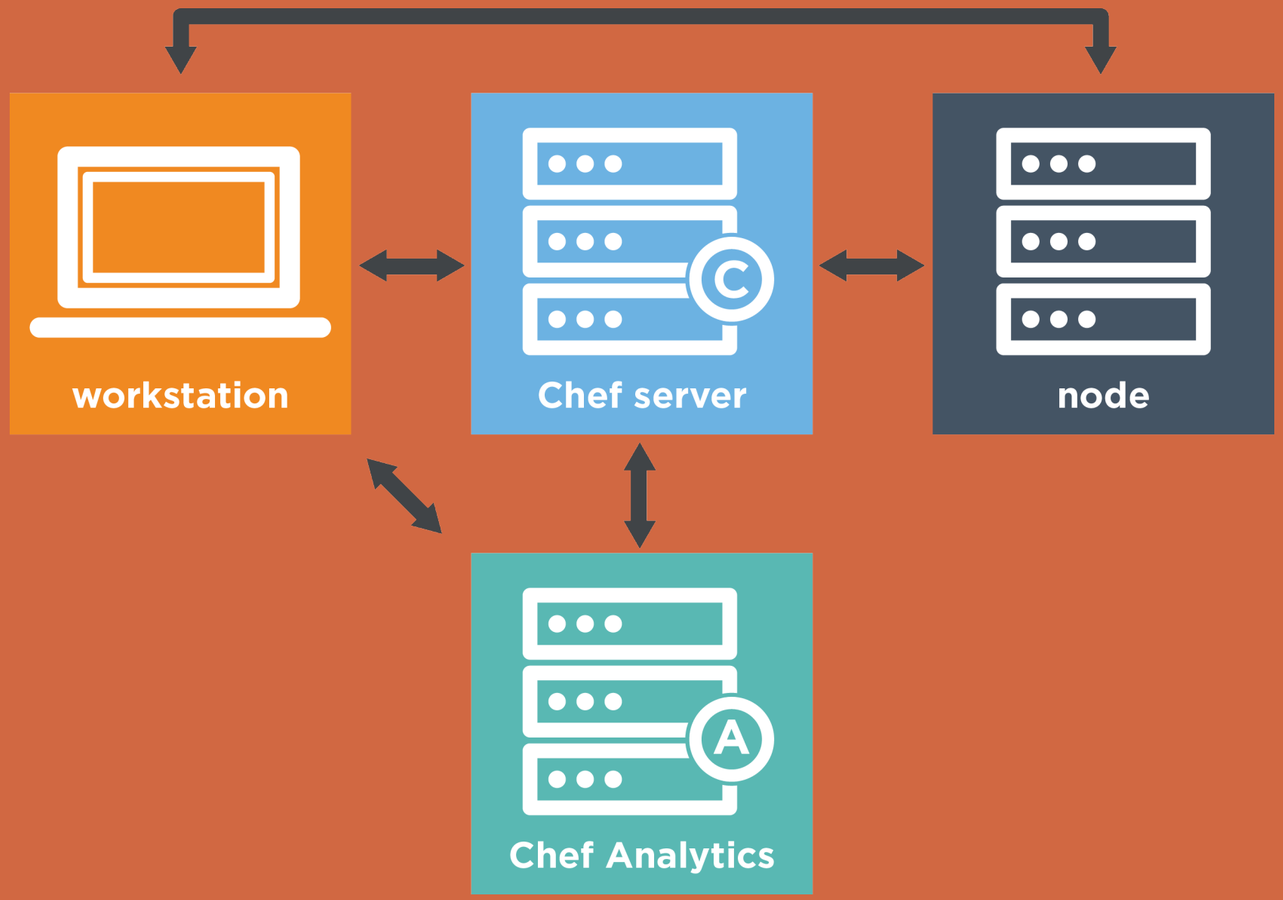
Although not always required, many compliance processes require a final human sign off.

Think about your current compliance and audit process.

How can you *prove* that the actual state of your infrastructure meets the desired state?

How can Chef make this better?

Chef Infrastructure



Audit mode also acts as a form of documentation.

The audit tests formally define the requirements, and from the output you can generate reports that prove whether the actual state of your infrastructure meets those requirements.

You can use audit mode in environments that are not managed by Chef.

You can start by using audit mode in your existing infrastructure to discover audit failures. As a second step, you can write Chef code that address those audit failures.

After completing this workshop, you'll be able to:

- Write and apply controls, both to a local virtual machine and to a node bootstrapped to your Chef server.
- Verify and resolve audit failures.
- Use Chef Analytics to create alerts that signal when your infrastructure falls out of compliance.

Set up your workstation

Everyone will get two playing cards. Match up your cards to the two IP addresses assigned to you.

http://bit.ly/bos-int-chef

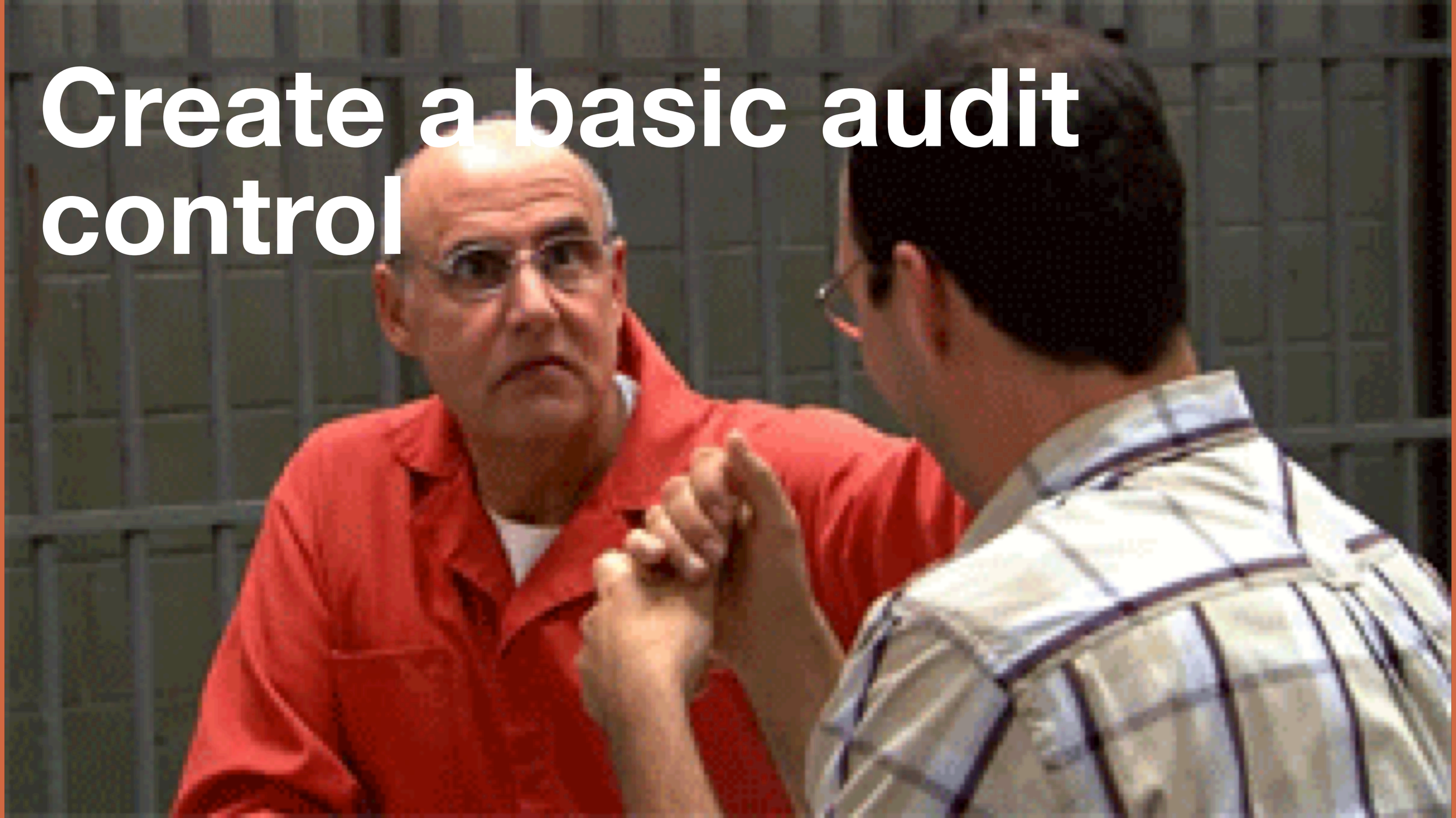
One will be your Chef workstation, the other will be your test node. It doesn't matter which one is which; just keep track.

Connect to your chef workstation

```
ssh chef@10.10.10.10
```

(the password is "chef.io")

Create a basic audit control



In this lesson, you'll create two cookbooks. The first cookbook implements a basic audit control that validates the ownership of web server content.

The second cookbook configures Apache web server and adds a few basic web pages to it. You'll use Test Kitchen to apply both cookbooks to a Ubuntu virtual machine.

You'll see that although the web server cookbook appears correct, it applies changes that violate your organization's compliance policy. You'll complete the lesson by fixing the violation and verifying that the system meets compliance.

Create the audit cookbook

run these commands from the home directory on your chef workstation

```
unzip chef-starter.zip

cd chef-repo/cookbooks

chef generate cookbook YOUR_INITIALS-audit-webserver
```

Add an audit control

Let's say that your organization's internal audit policy states that no web file can be owned by the `root` user. Let's add an audit control that tests for this.

Add the following code to your default recipe, `default.rb`.

```
control_group 'YOUR_INITIALS - web server' do
  control 'home page is not owned by root user' do
    describe file('/var/www/html/index.html') do
      it { should_not be_owned_by 'root' }
    end
  end
end
```

What can you do?

Chef audit controls are based on Serverspec, which is based on RSpec. The Serverspec documentation describes the resource types you can use in your audit controls.

Apply the recipe to a Test Kitchen instance

Now let's apply the audit control to a Ubuntu virtual machine. First, modify your `audit-webserver` **cookbook's** `.kitchen.yml` **file to look like this.**

```
driver:
  name: docker
  use_sudo: false
provisioner:
  name: policyfile_zero
  client_rb:
    audit_mode: :audit_only
platforms:
  - name: ubuntu-14.04
suites:
  - name: default
    attributes:
```

Install cookbooks

Install cookbooks from a Policyfile and generate a locked cookbook set.

```
chef install
```

Run `kitchen list` to verify that the instance has not yet been created.

```
$ kitchen list
-----> Using experimental policyfile mode for chef-client
$$$$$$ The Policyfile feature is under active development.
$$$$$$ For best results, always use the latest chef-client version
Instance             Driver  Provisioner     Verifier  Transport  Last Action
default-ubuntu-1404  Docker  PolicyfileZero  Busser    Ssh        <Not Created>
```

Now run `kitchen converge` to create the instance and apply your audit control.

```
$ kitchen converge
-----> Starting Kitchen (v1.4.2)
-----> Using experimental policyfile mode for chef-client
$$$$$$ The Policyfile feature is under active development.
$$$$$$ For best results, always use the latest chef-client version
-----> Creating <default-ubuntu-1404>...
       Sending build context to Docker daemon 60.42 kB
       Sending build context to Docker daemon
       Step 0 : FROM ubuntu:14.04
        ---> a5a467fddcb8
       Step 1 : RUN dpkg-divert --local --rename --add /sbin/initctl
        ---> Using cache
        ---> 118332391ccd
       Step 2 : RUN ln -sf /bin/true /sbin/initctl
        ---> Using cache
        ---> 68b4627de9c5
       Step 3 : ENV DEBIAN_FRONTEND noninteractive
        ---> Using cache
        ---> 7ed5da3b5a01
       Step 4 : RUN apt-get update
        ---> Using cache
        ---> ee741f5d4fd8
       Step 5 : RUN apt-get install -y sudo openssh-server curl lsb-release
        ---> Using cache
        ---> 740908b79670
       Step 6 : RUN if ! getent passwd kitchen; then useradd -d /home/kitchen -m -s /bin/bash kitche
```

Create the webserver cookbook

Now let's create and apply a second cookbook that configures Apache and adds a few web pages.

First, from your `~/chef-repo` **directory, create the** `webserver` **cookbook**

```
chef generate cookbook cookbooks/
YOUR_INITIALS-webserver
```

Add the following to your webserver cookbook's `recipes/default.rb`

```
# Install the apache2 package.
package 'apache2' do
  action :install
end

# Enable and start the apache2 service.
service 'apache2' do
  action [:start, :enable]
end

# Add a home page to the site.
file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>"
end
```

Configure the `.kitchen.yml` on the webserver cookbook

```
driver:
  name: docker
  use_sudo: false
provisioner:
  name: policyfile_zero
  client_rb:
    audit_mode: :enabled
platforms:
  - name: ubuntu-14.04

suites:
  - name: default
    attributes:
```

Let's test out our webserver cookbook

- `chef install`
  - **generate a locked set of cookbooks**
- `kitchen converge`
  - **see it spin up the instance**
- `kitchen login`
  - **login to the instance to check it out**

Run a few commands to see if it's looking good

- `ls /var/www/html/`
- `curl http://localhost`

Audit your web server configuration

In previous steps, you applied the audit and webserver cookbooks on separate Test Kitchen instances. Let's set things up so that you can run them both from the same instance. Here you'll apply the audit cookbook from the Test Kitchen instance for your webserver cookbook.

Update the Policyfile

Edit the `Policyfile.rb` file in the `webserver` cookbook to look like this:

```
name "webserver"

default_source :supermarket

run_list "YOUR_INITIALS-webserver::default", "YOUR_INITIALS-audit-webserver::default"

cookbook "YOUR_INITIALS-webserver", path: "."
cookbook "YOUR_INITIALS-audit-webserver", path: "../YOUR_INITIALS-audit-webserver"
```

Edit the .kitchen.yml on your webserver cookbook

```
driver:
  name: docker
  use_sudo: false
provisioner:
  name: chef_zero
  client_rb:
    audit_mode: :enabled
platforms:
  - name: ubuntu-14.04

suites:
  - name: default
    attributes:
```

Update Policyfile.lock.json

```
chef update
```


KITCHEN CONVERGE!!
ORANGE

Failure/Error!

```
       Starting audit phase

       web server
         home page is not owned by root user
           File "/var/www/html/index.html"
             should not be owned by "root" (FAILED - 1)

       Failures:

         1) web server home page is not owned by root user File "/var/www/html/index.html" should no$
 be owned by "root"
            Failure/Error: it { should_not be_owned_by 'root'}
       expected `File "/var/www/html/index.html".owned_by?("root")` to return false, got true
            # /tmp/kitchen/cache/cookbooks/audit-webserver/recipes/default.rb:9:in `block (4 levels)
in from_file'

       Finished in 0.11597 seconds (files took 0.27985 seconds to load)
       1 example, 1 failure

       Failed examples:

       rspec  # web server home page is not owned by root user File "/var/www/html/index.html" should
 not be owned by "root"

       Audit phase exception:
         Audit phase found failures - 1/1 controls failed
```

Update your web server configuration to meet compliance

In the previous step, we saw that the home page failed the audit.

webserver/recipe/default.rb

```
# Install the apache2 package.
package 'apache2' do
  action :install
end

# Enable and start the apache2 service.
service 'apache2' do
  action [:start, :enable]
end

# Add a home page to the site.
file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>"
  owner 'www-data'
  group 'www-data'
end
```


KITCHEN CONVERGE AGAIN!!

```
Starting audit phase

web server
  home page is not owned by root user
    File "/var/www/html/index.html"
      should not be owned by "root"

Finished in 0.1117 seconds (files took 0.28909 seconds to load)
1 example, 0 failures
Auditing complete

Running handlers:
Running handlers complete
Chef Client finished, 1/4 resources updated in 02 seconds
  1/1 controls succeeded
Finished converging <default-ubuntu-1404> (0m9.44s).
```

Get an alert when an audit run fails

Next, let's upload your `audit` and `webserver` cookbooks to Chef server, run them on a node, and see how to use Chef Analytics to alert you when your infrastructure falls out of compliance.

Push your Policy to the Chef Server

From `~/chef-repo`

```
knife ssl fetch
```

From `~/chef-repo/cookbooks/YOUR_INITIALS-webserver`

```
chef push staging
```

Upload your cookbooks to the Chef server

You already verified that your `audit` **and** `webserver` **cookbooks behave as you expect on a local virtual machine, so let's begin by uploading these cookbooks to your Chef server.**

Run these commands from the `chef-repo` **directory.**

```
knife cookbook upload -a
rm .chef/chef-boston-validator.pem
```

Apply the webserver cookbook to a node

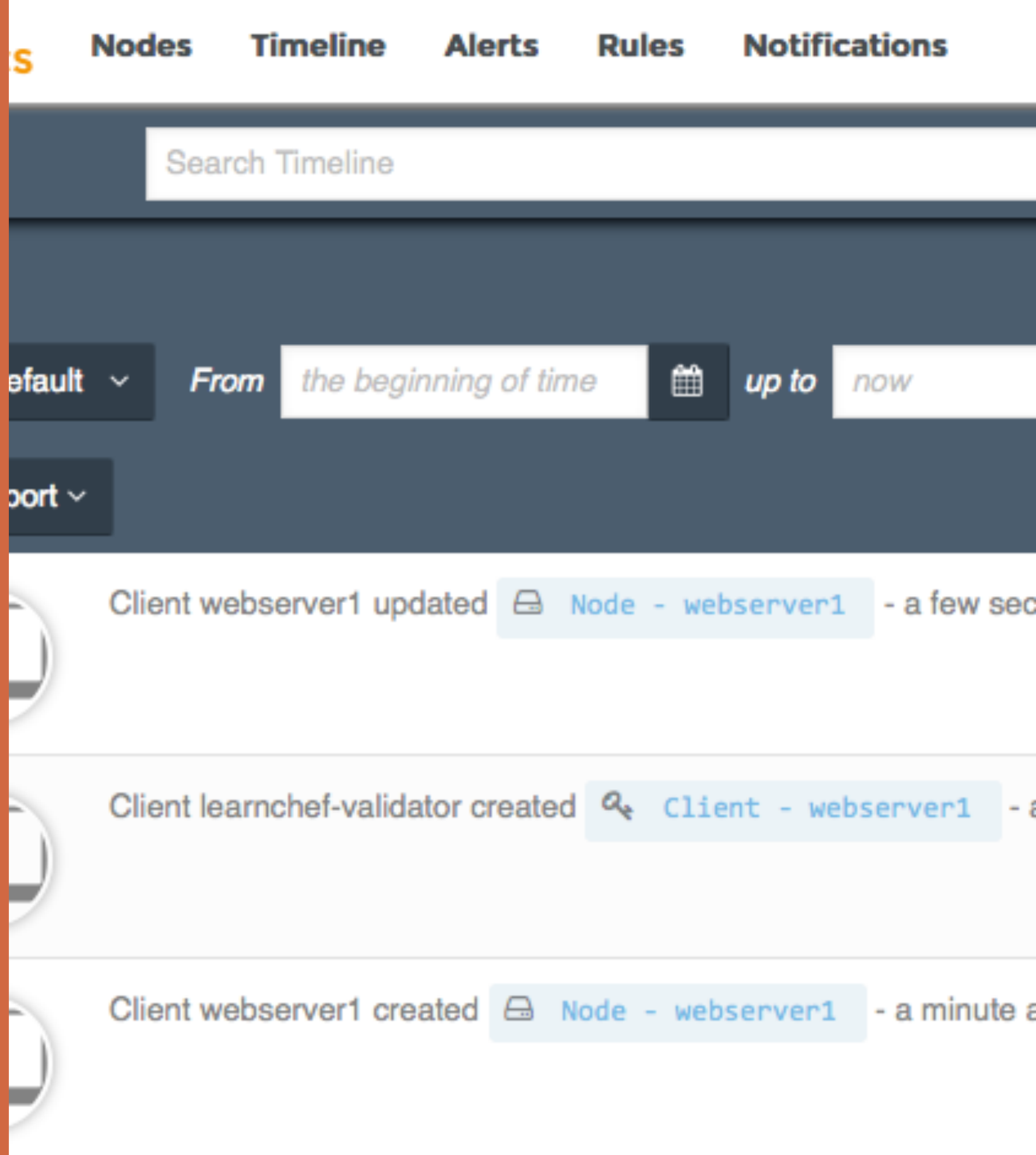
(replace YOUR_IP_ADDRESS with the IP address of your SECOND node)

```
knife bootstrap YOUR_IP_ADDRESS
--ssh-user chef --ssh-password 'chef.io'
--policy-group staging
--policy-name webserver --sudo
-N YOUR_INITIALS-node1
```

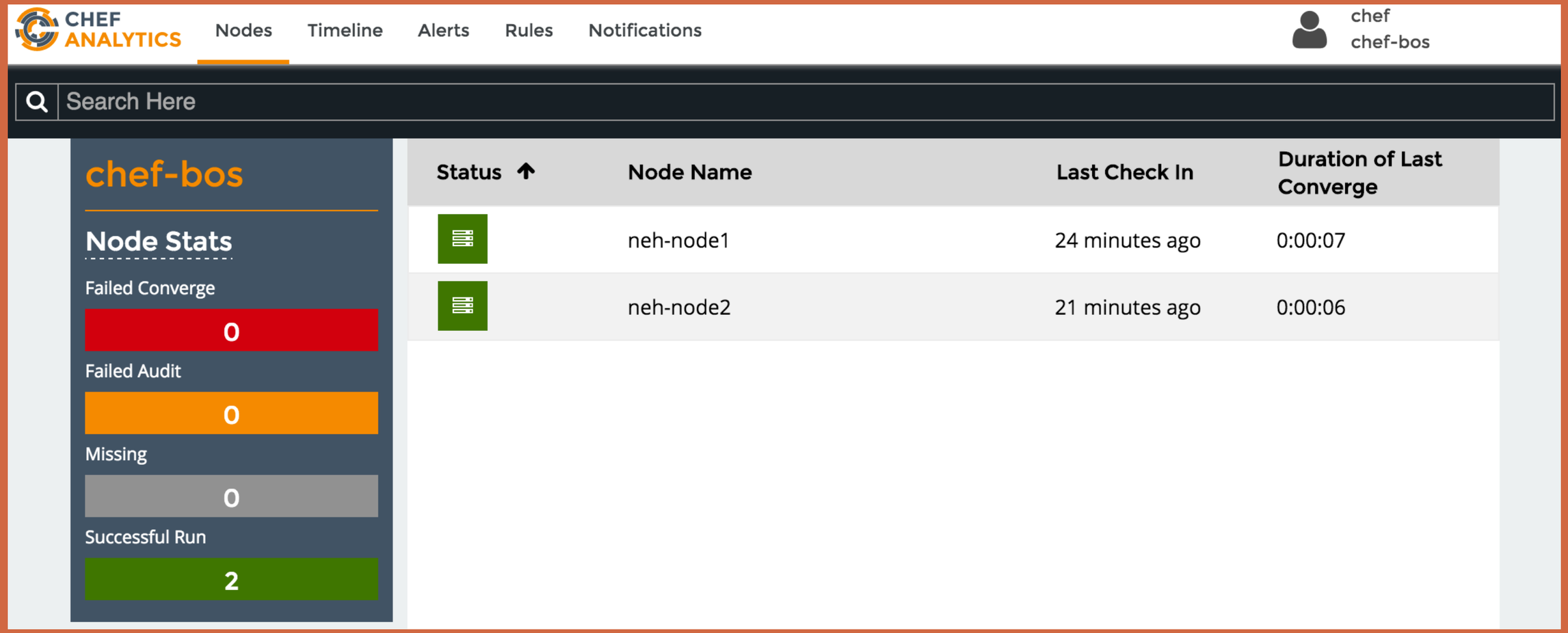
View the events in the Timeline view

Log into http://bit.ly/analytics-bos-chef.

Once logged in, you should see something like this:



The nodes tab should look like this, since you had a successful run.



Add rules that trigger an alert when an audit fails

In this tutorial, instead of using a notification to respond to an event, you'll create an alert. An alert enables you to capture important events that you want to take action on, such as when an audit fails.

You access the alert history through the Chef Analytics web interface. Like a notification, an alert is raised from a rule. You can generate an alert and a notification from the same rule.

Recall that your control looks like this.

```
control_group 'YOUR_INITIALS - web server' do
  control 'home page is not owned by root user' do
    describe file('/var/www/html/index.html') do
      it { should_not be_owned_by 'root' }
    end
  end
end
```

To create a rule that triggers when this audit fails, first navigate to the Chef Analytics interface from your web browser.

From the Rules tab, click + to create a new rule. From the rule editor, click `New Rule Group 1` **and rename it to** `YOUR_INITIALS webserver`**.**

Now add the following code to define your rule:

```
rules 'YOUR_INITIALS webserver'
  rule on run control
  when
    control_group.name =~ 'YOUR_INITIALS - web server' and status != 'success'
  then
    alert:error('YOUR_INITIALS - Run control group {{message.resource_type}} {{message.resource_name}} "{{ message.name }}" failed on {{ message.run.node_name }}.')
  end
end
```

Click Save and you are brought back to the list of rules.

Run the audit cookbook

Now let's run the `audit` cookbook on your node. You already verified that the audit passes when you ran it through Test Kitchen, so let's verify that the alert doesn't trigger.

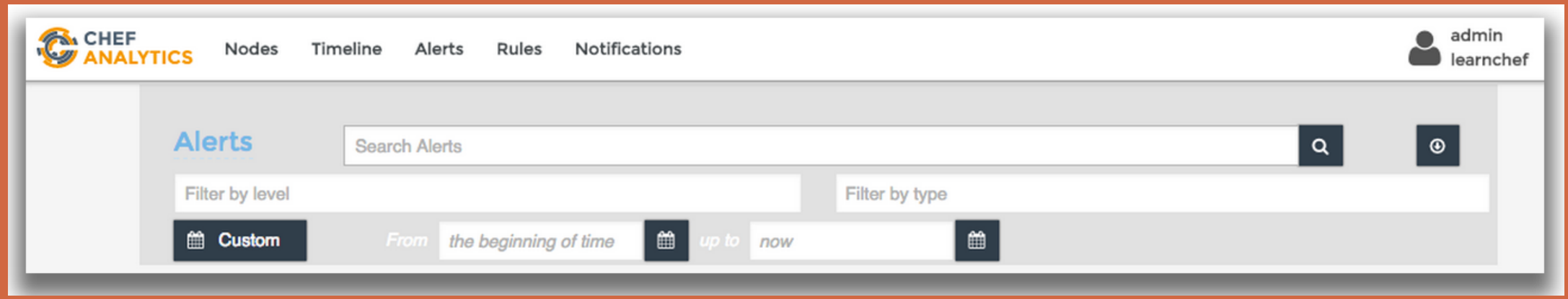
Because you already applied the `webserver` cookbook to your node, this time you'll specify the `--audit-mode audit-only` option to run only the audit code on your node.

Run this command - replace IP with the IP address as before

```
knife ssh 'name:YOUR_INITIALS*'
'sudo chef-client
--audit-mode audit-only' --ssh-user chef
--ssh-password 'chef.io'
```

Verify that the alert didn't trigger

You'll see from the output of your `chef-client` run that the audit tests pass. From the Alerts tab on the Chef Analytics web interface, verify that no alerts appear.



Add a new control

Specifically we want to ensure that `ntp` service is enabled and the `ntp` service is running

Add an audit-ntp cookbook

```
chef generate cookbook cookbooks/
YOUR_INITIALS-audit-ntp
```

Add the following to your audit-ntp cookbook's default recipe:

```
control_group 'YOUR_INITIALS - ntp' do
  control 'ntp is running and enabled' do
    describe service('ntp') do
      it { should be_running }
      it { should be_enabled }
    end
  end
end
```

Edit the webserver Policy to depend on your ntp cokbook

```
run_list "YOUR_INITIALS-webserver::default",
"YOUR_INITIALS-audit-webserver::default",
"YOUR_INITIALS-audit-ntp::default"

cookbook "YOUR_INITIALS-webserver", path: "."
cookbook "YOUR_INITIALS-audit-webserver", path: "../YOUR_INITIALS-audit-webserver"
cookbook "YOUR_INITIALS-audit-ntp", path: "../YOUR_INITIALS-audit-ntp"
```

Increment the version of the webserver cookbook

Edit the `metadata.rb` in the webserver cookbook.

Update the Policyfile.lock.json

```
chef update
```

Test locally

Run `kitchen converge` on the webserver cookbook. It should fail 2 of 3 controls.

Upload the new Policyfile and cookbooks

From the `cookbooks/YOUR_INITIALS-webserver` **directory**

```
chef push staging
```

From the chef-repo directory:

```
knife cookbook upload -a
```

Add the Chef Analytics rules

Navigate to the Chef Analytics interface from your web browser. From the Rules tab, click + to create a new rule. From the rule editor, click `New Rule Group 1` **and rename it to** `YOUR_INITIALS - NTP`**.**

Add the following code to your control

```
rules 'YOUR_INITALS - Ensure NTP is active'
  rule on run_control
  when
    control_group.name =~ 'YOUR_INITIALS - NTP' and
    status != 'success'
  then
    alert:error('YOUR_INITIALS - Run control {{message.resource_type}} {{message.resource_name}} "{{ message.name }}" failed on {{ message.run.node_name }}.')
  end
end
```

Run the audit and watch it fail

Run this command - replace IP with the IP address as before

```
knife ssh 'name:YOUR_INITIALS*'
'sudo chef-client
--audit-mode audit-only' --ssh-user chef
--ssh-password 'chef.io'
```

Log into the Alerts tab in Analytics and see the alert!

Extra Credit

- **On your chef workstation, cd into** `~/cookbooks/audit-shell-shock` **and run** `kitchen converge`
- **Add a new cookbook to remediate the node**
- **Write your own audit control and rule with a partner!**