

# Nathen Harvey

- Community Director at Chef
- Co-host of the Food Fight Show
- Co-organizer of DevOpsDC meetup
- Occasional farmer – <http://bit.ly/farmer-nathen>
- Love Eggs – <http://eggs.chef.io>



- @nathenharvey
- nharvey@chef.io

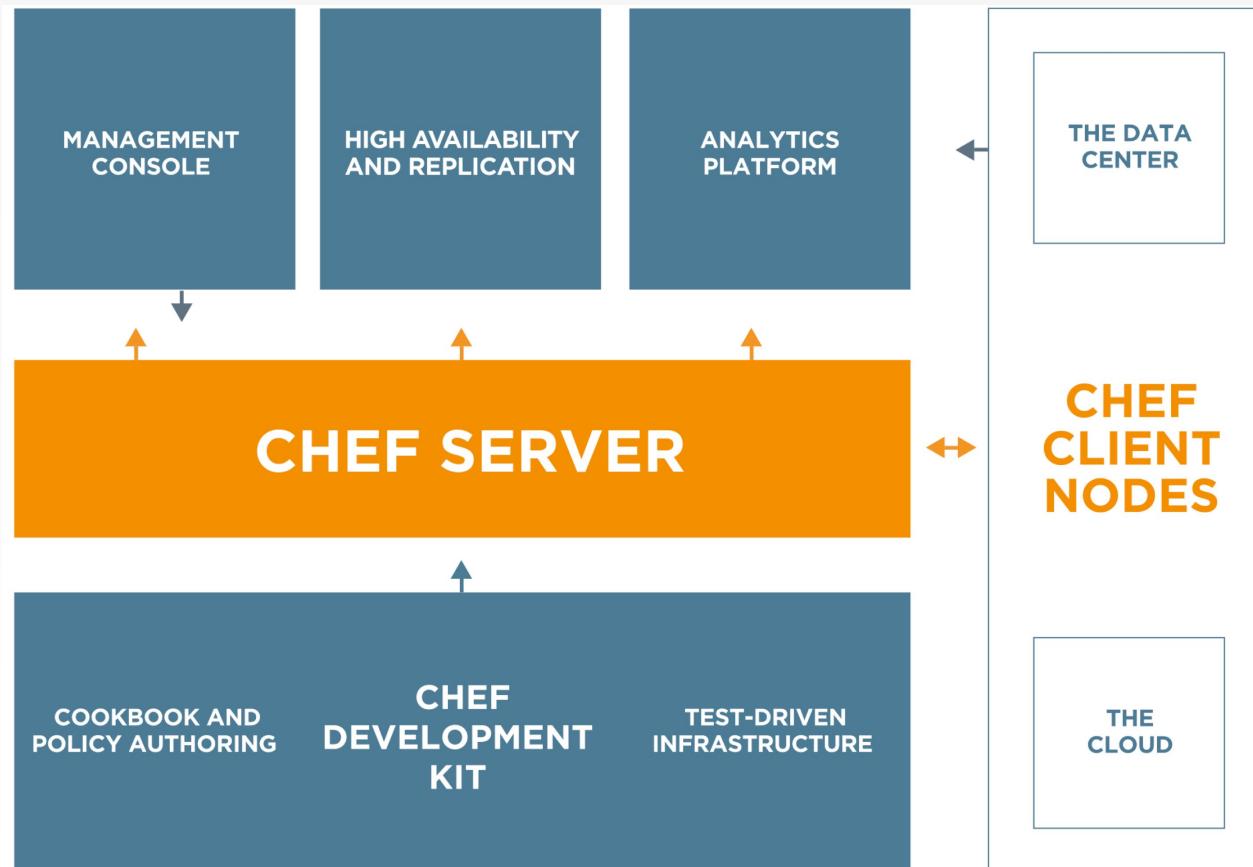


# My SysAdmin Journey

- ssh
- Store notes in ~ / server .txt
- Move notes to the wiki
- Write some scripts ( setup .sh, fixit .sh, etc.)
- Golden images and snapshots
- Remote execution via ssh
- Policy-driven configuration management



# Chef



<https://www.chef.io/chef/>

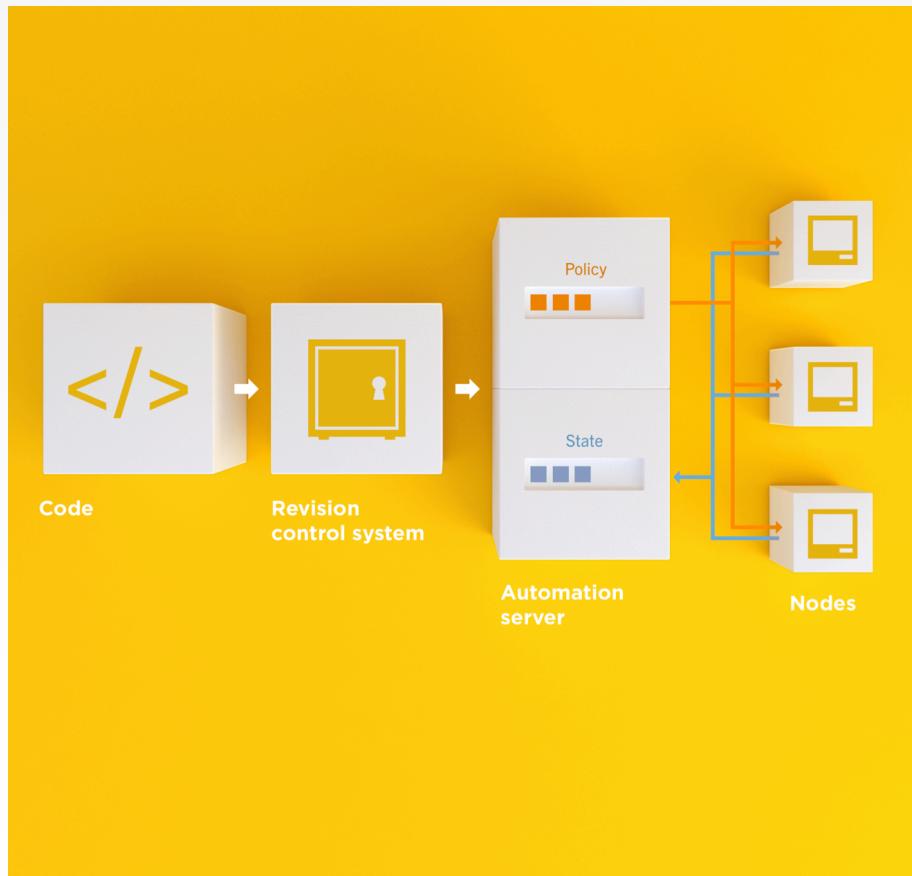


# Automation Platform

- Creates a dependable view of your entire network's state.
- Can handle complex dependencies among the nodes of your network.
- Is fault tolerant.
- Is secure.
- Can handle multiple platforms
- Can manage cloud resources
- Provides a foundation for innovation

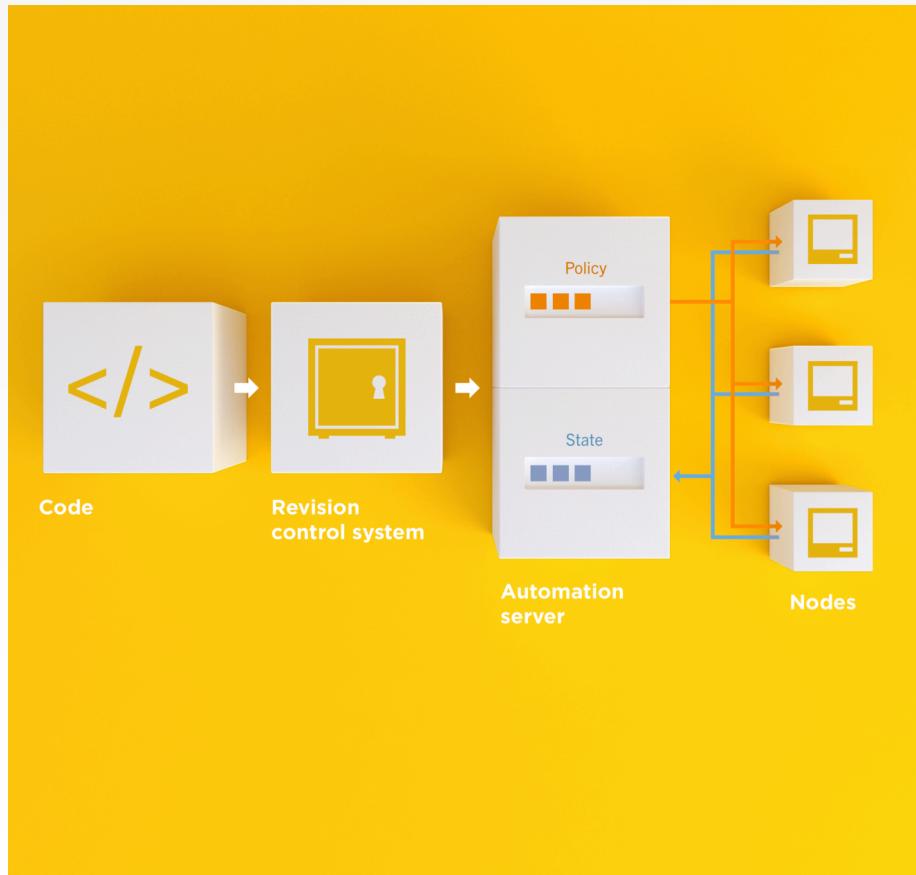


# Infrastructure as Code



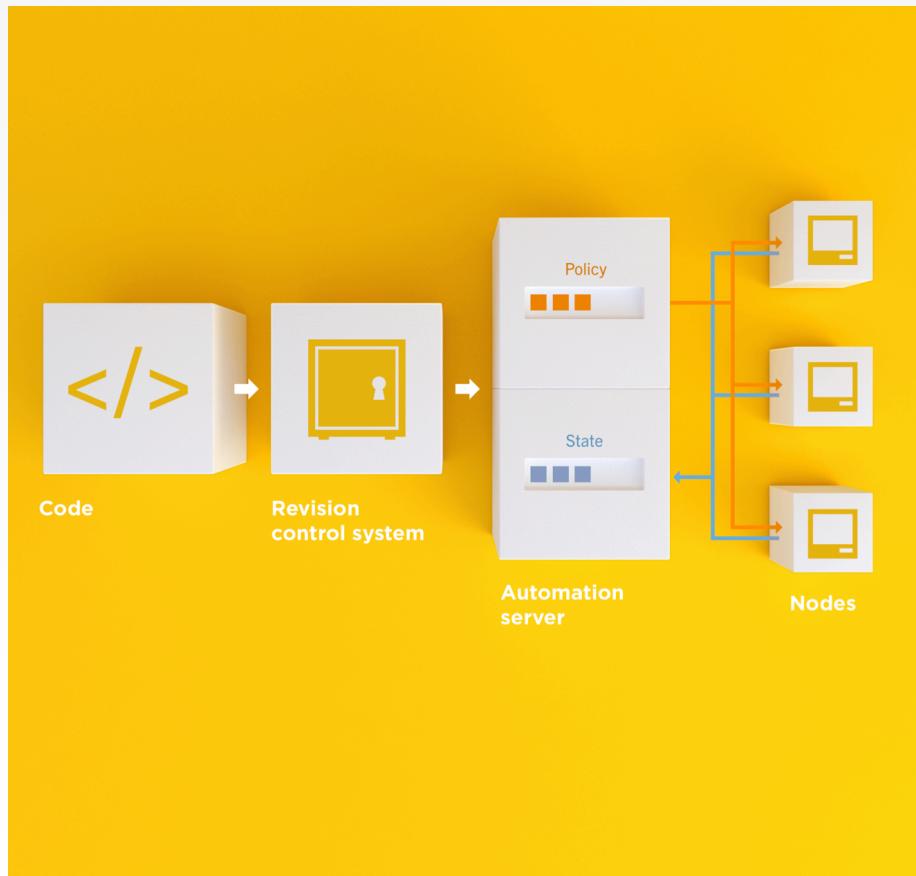
- Programmatically provision and configure components

# Infrastructure as Code



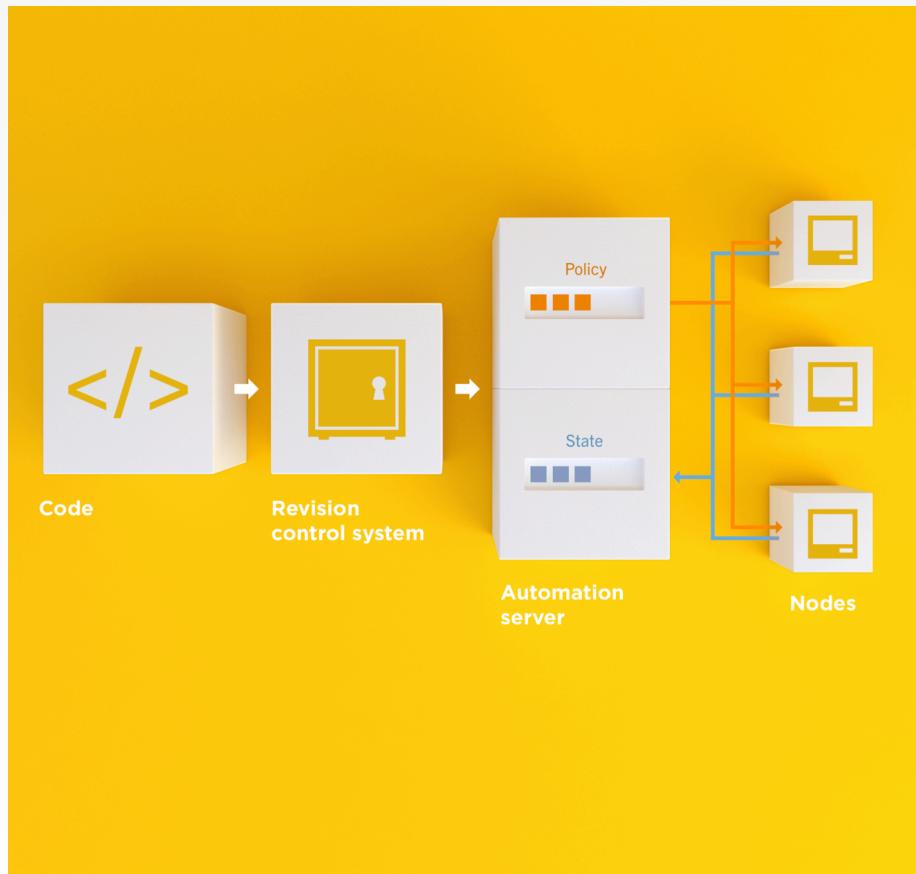
- Treat like any other code base

# Infrastructure as Code



- Reconstruct business from code repository, data backup, and compute resources

# Infrastructure as Code

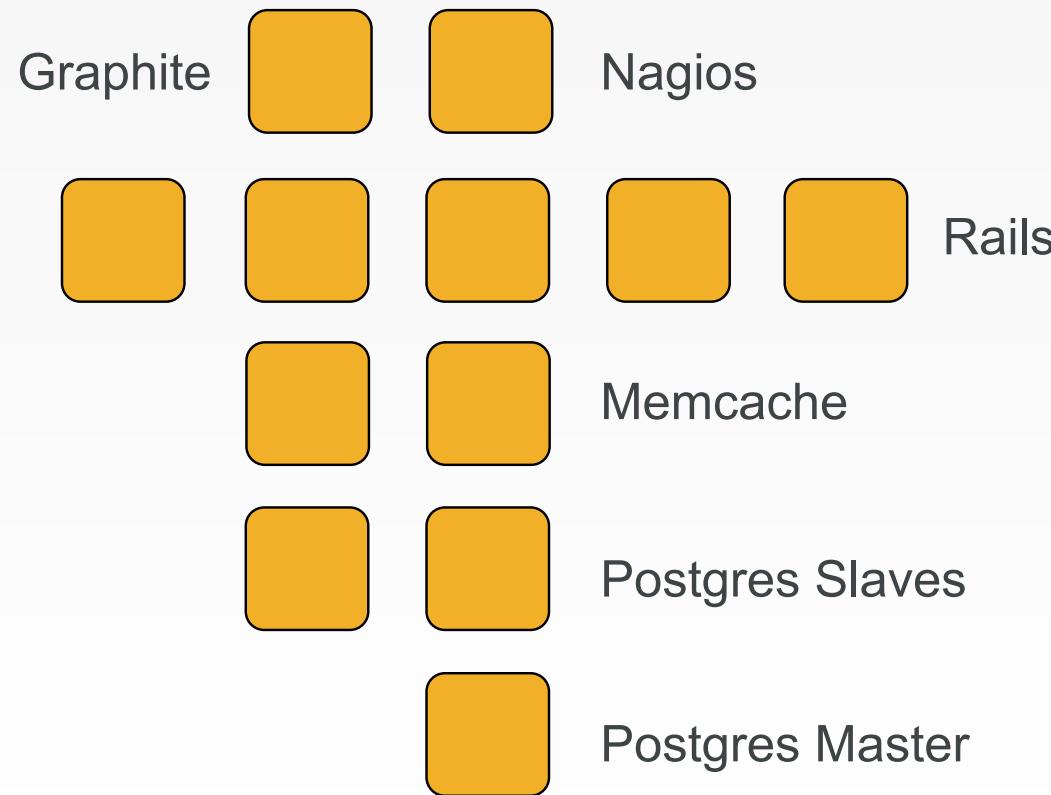


- Programmatically provision and configure components
- Treat like any other code base
- Reconstruct business from code repository, data backup, and compute resources

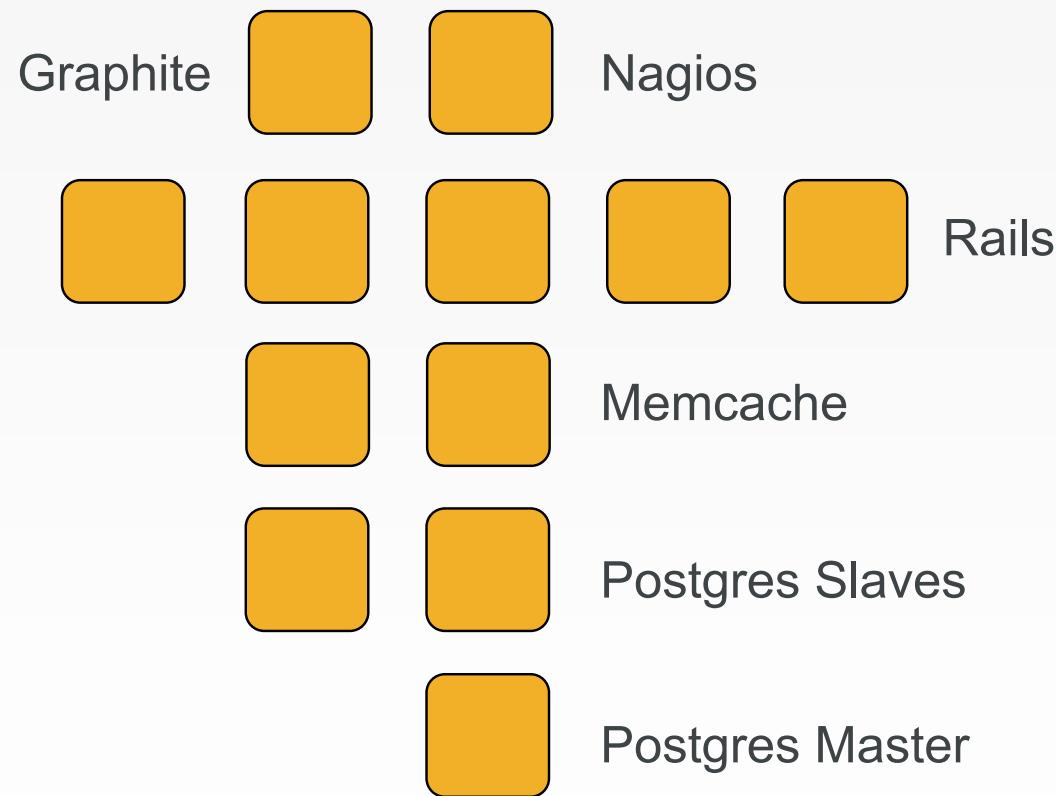
# Policy-based

- You capture the policy for your infrastructure in code
- A program ensures each node in your infrastructure complies with the policy
- A control loop keeps the system stable and allows for change when policy is updated

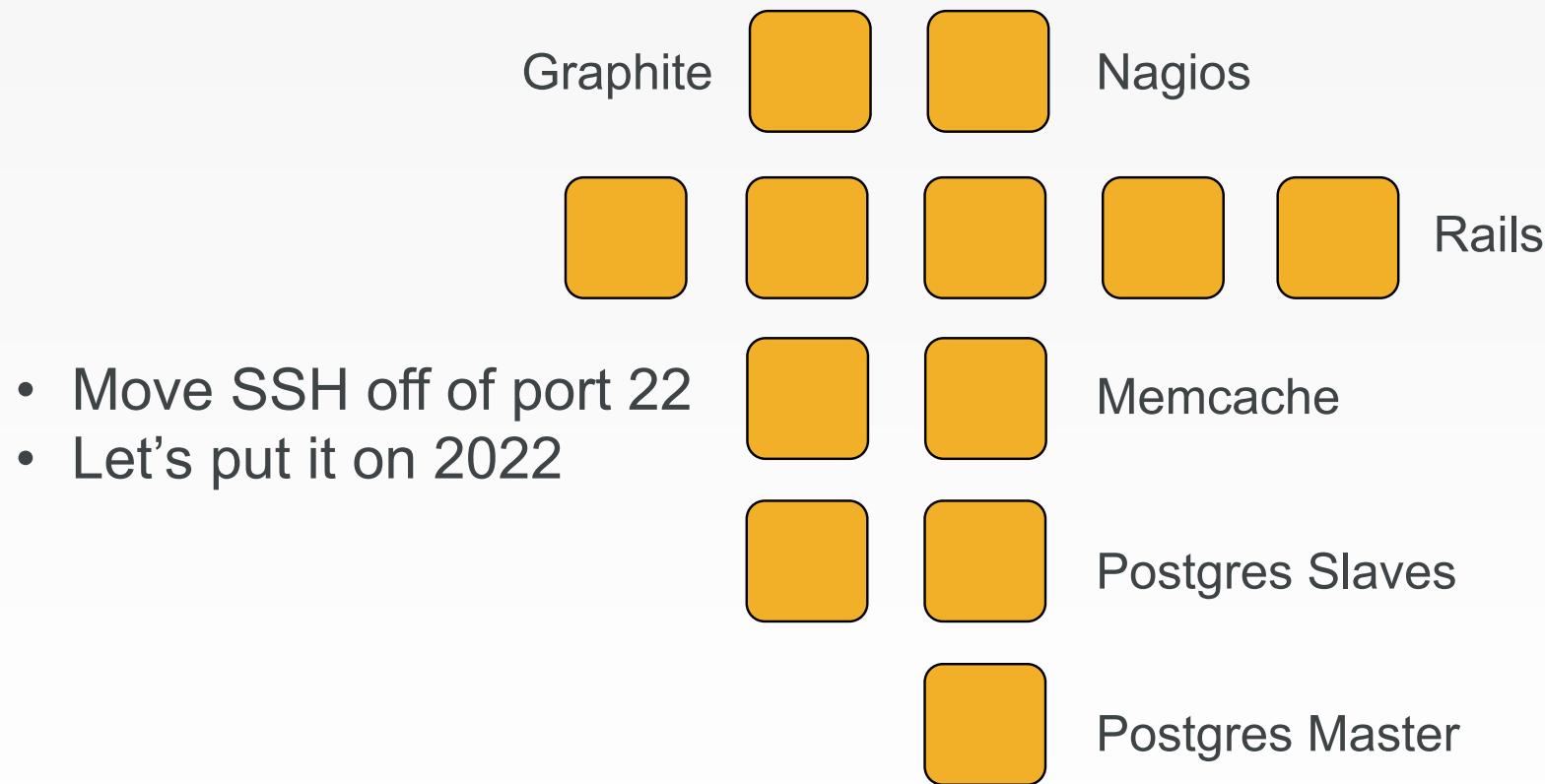
# Sample Infrastructure



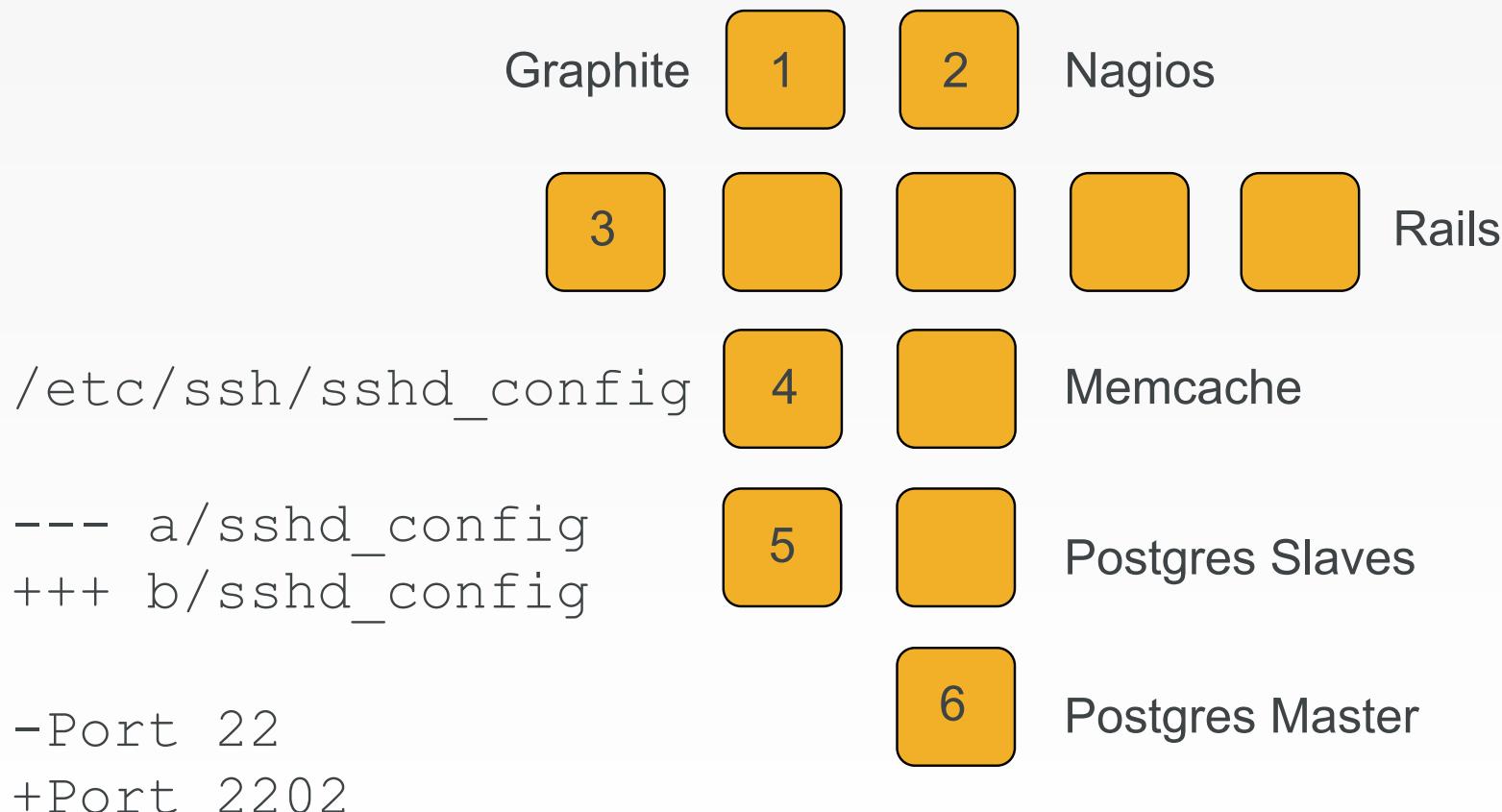
# New Compliance Mandate!



# New Compliance Mandate!

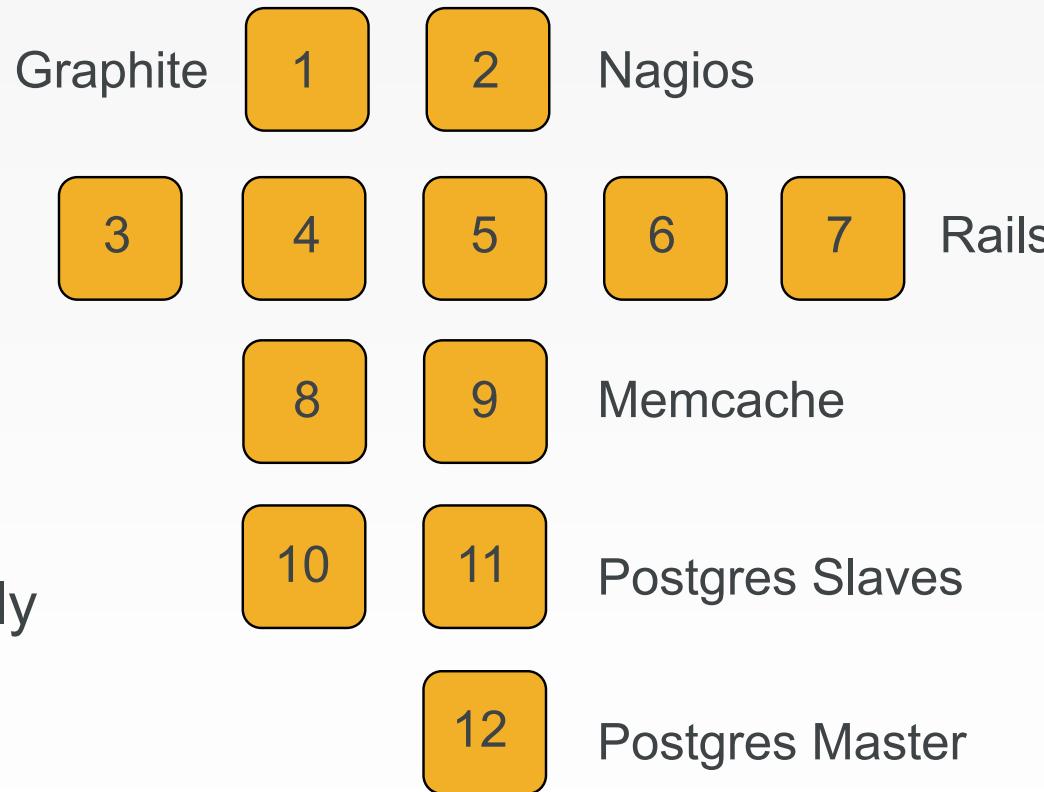


# 6 Golden Images to Update



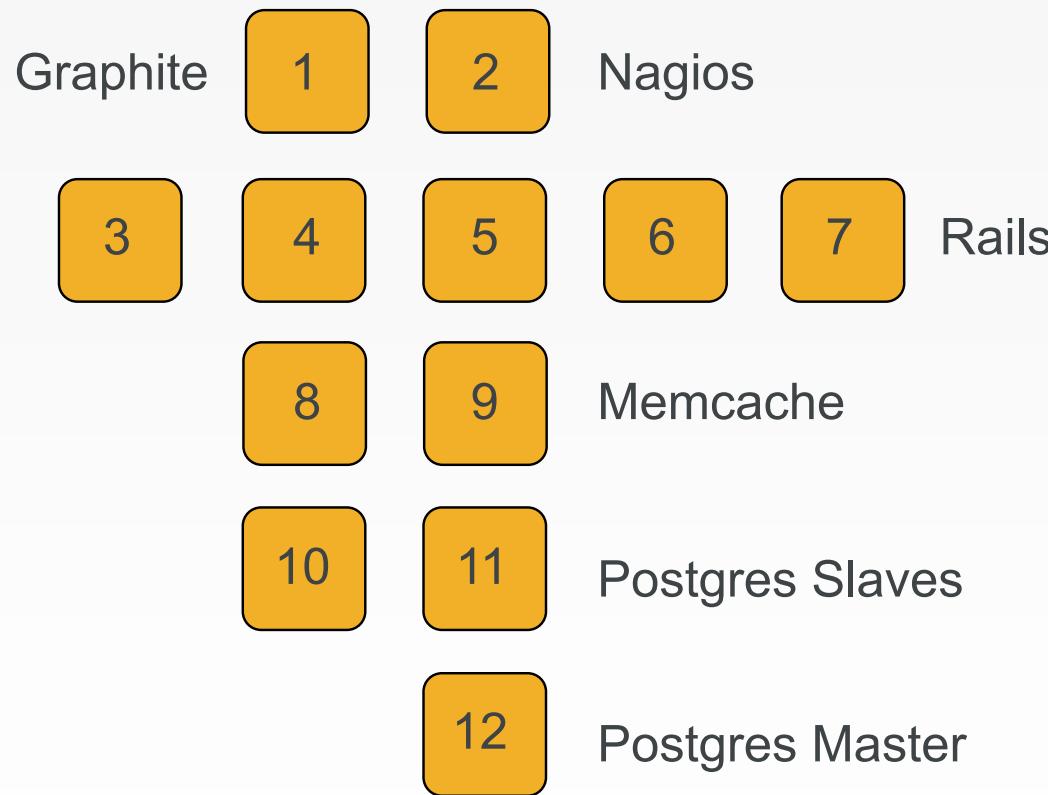
# 12 Instances to replace

- Launch
- Delete
- Repeat
- Typically manually

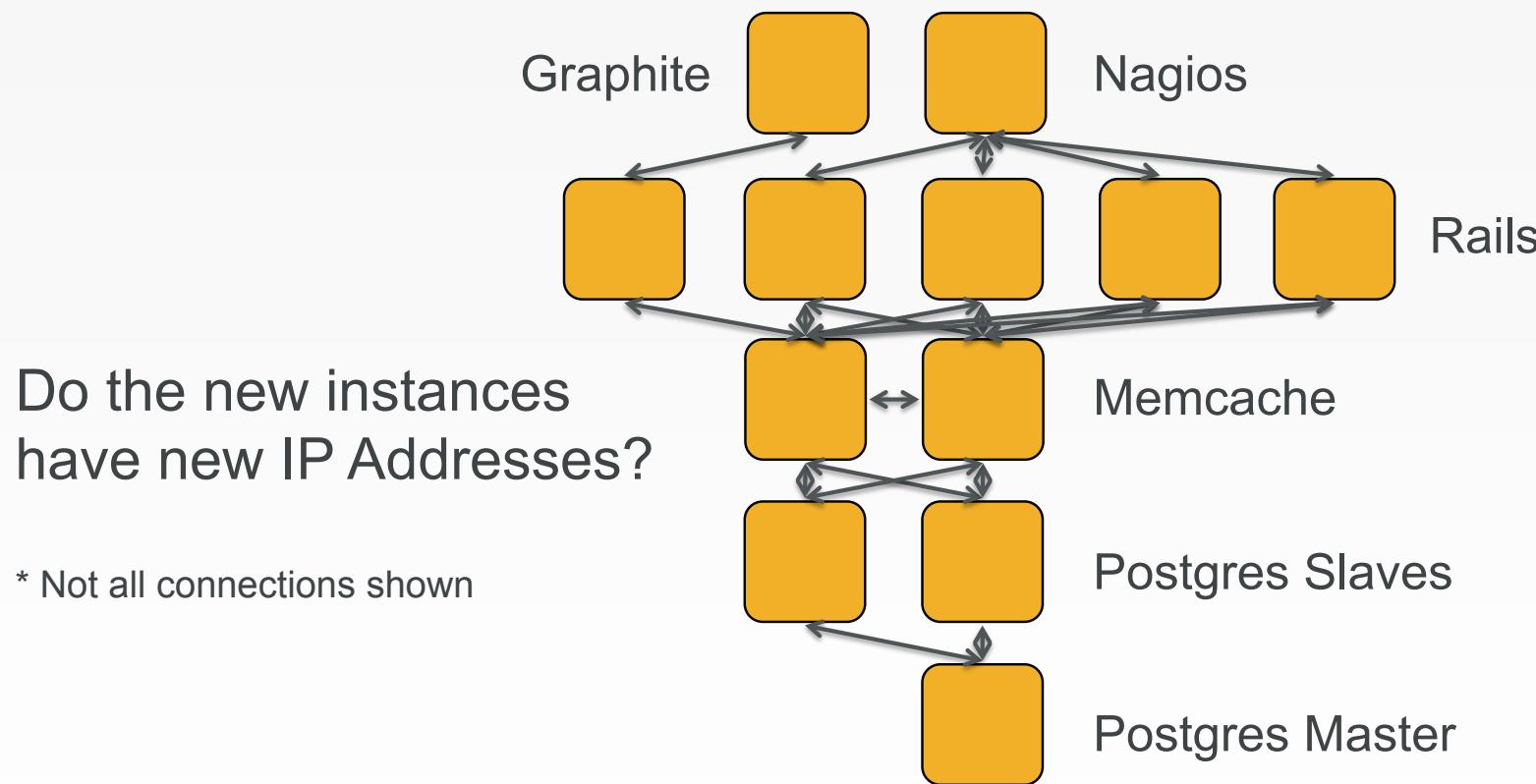


# Done in maintenance window

- High stakes
- Late hours
- Risky change



# New configurations required?





# Chef

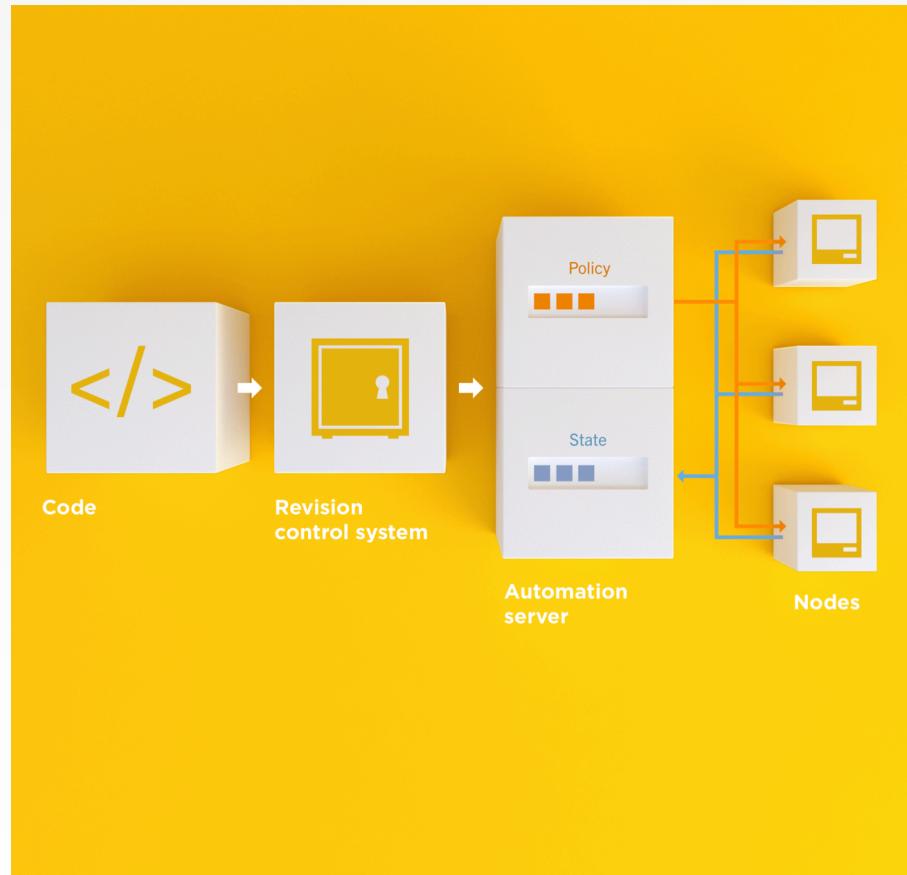
Fast, scalable, flexible IT automation



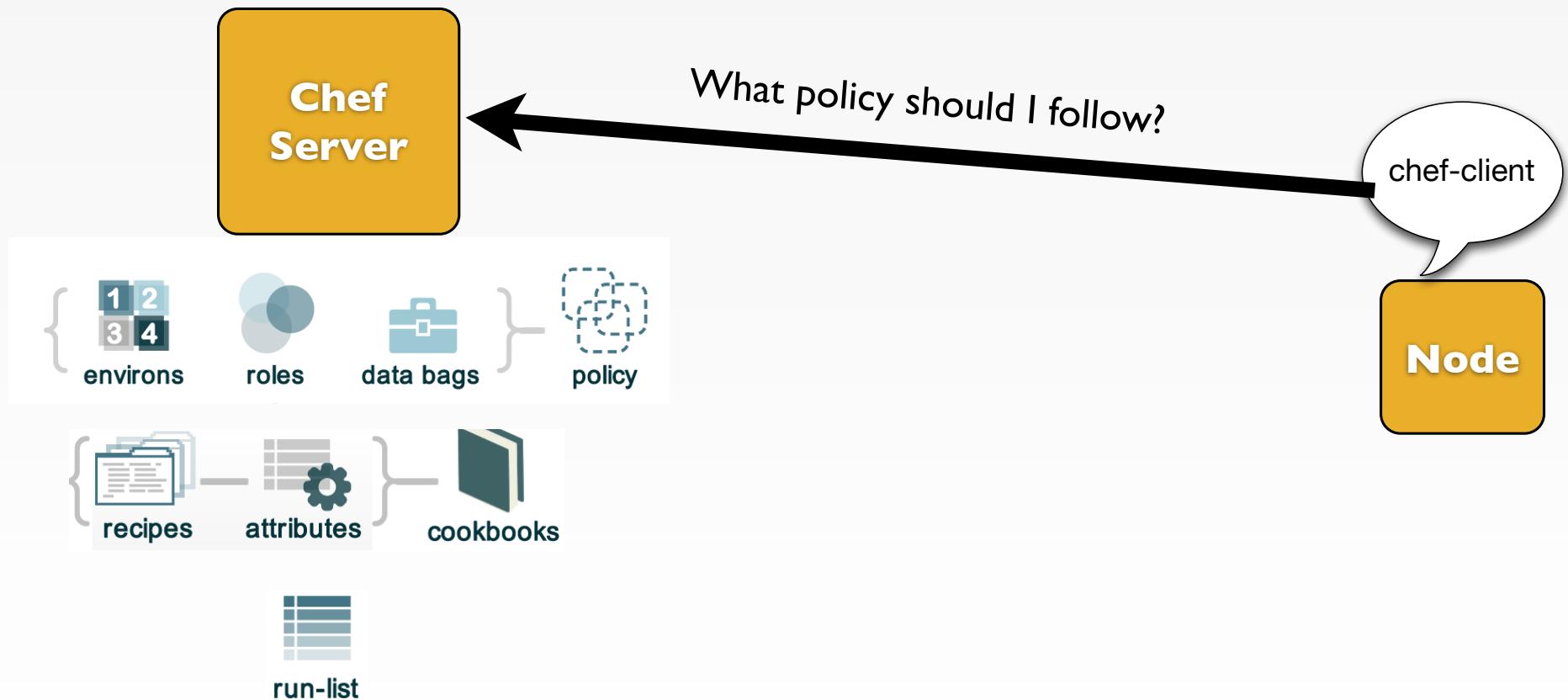
# What is Chef

- Open source framework for managing complexity in your infrastructure through policy-driven automation code
- A community of professionals
- A company

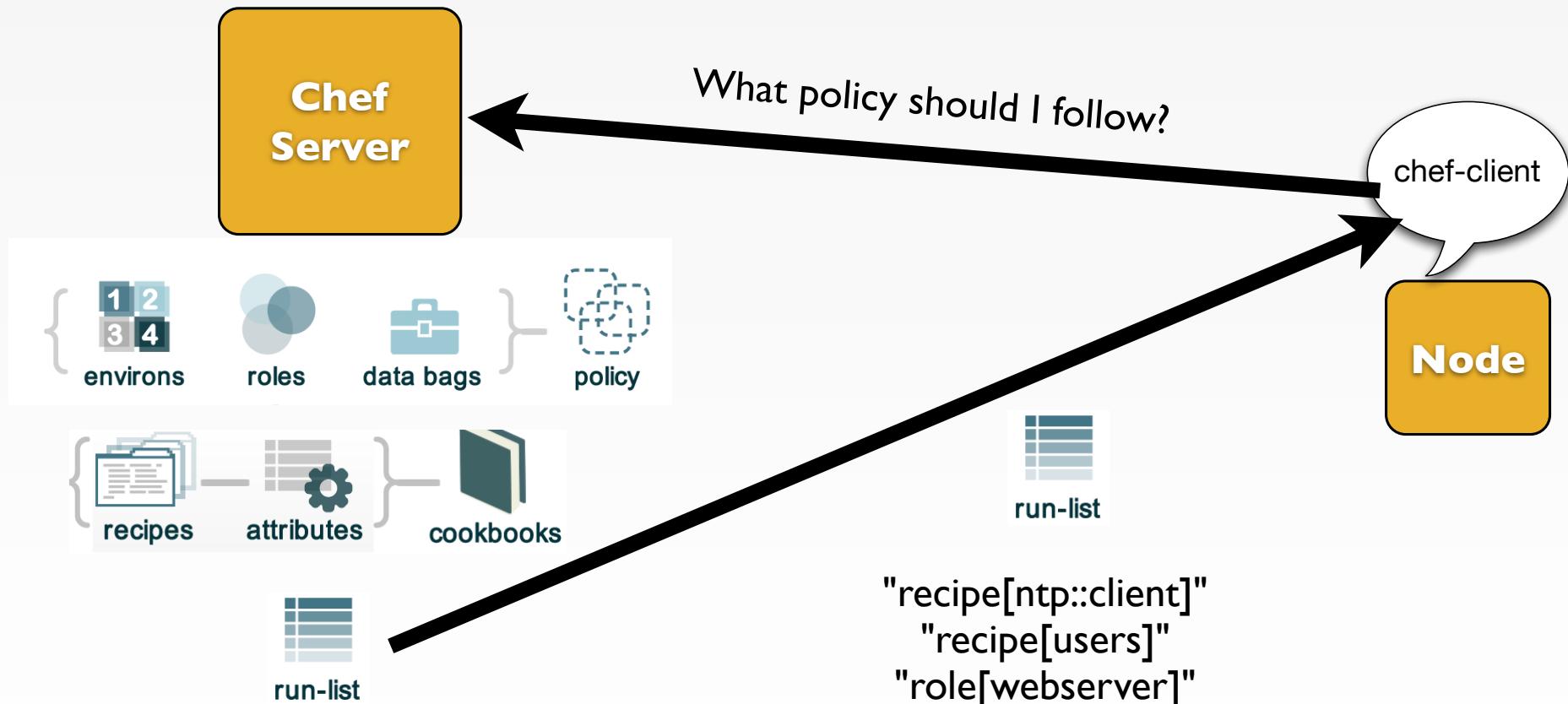
# Chef Server – Policy & State



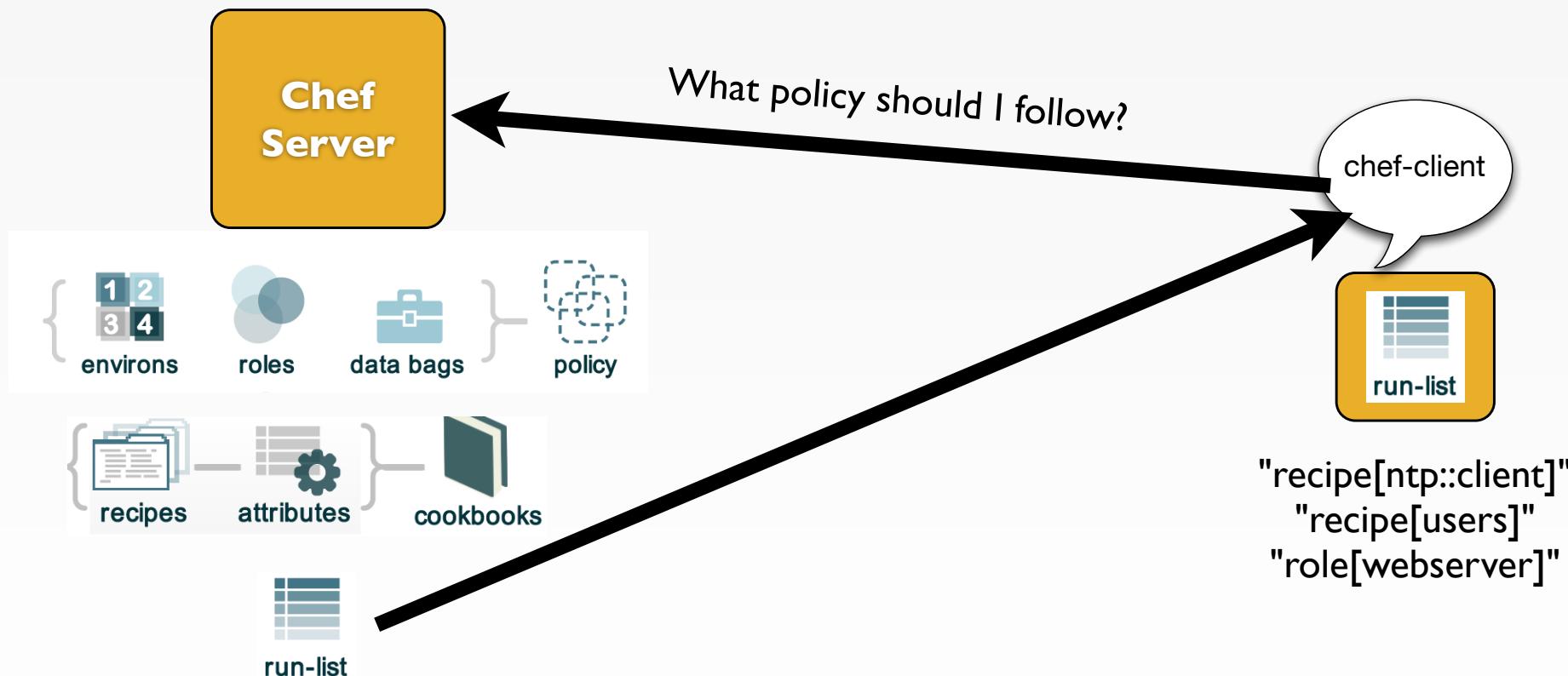
# Desired Configuration



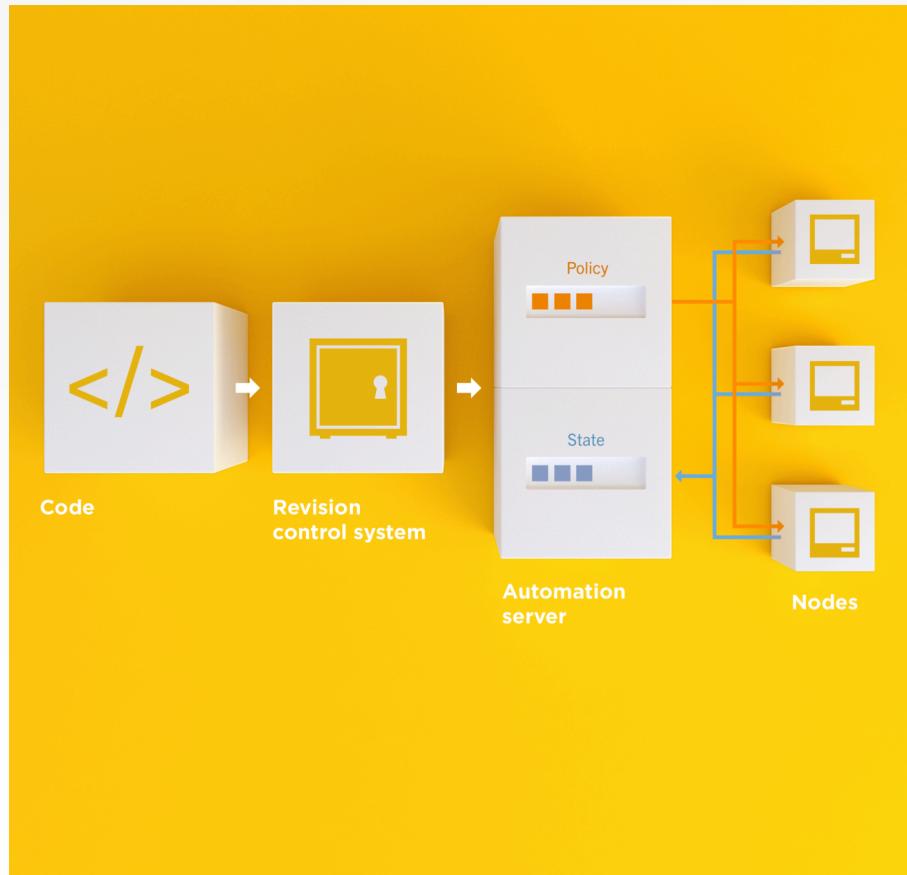
# Desired Configuration



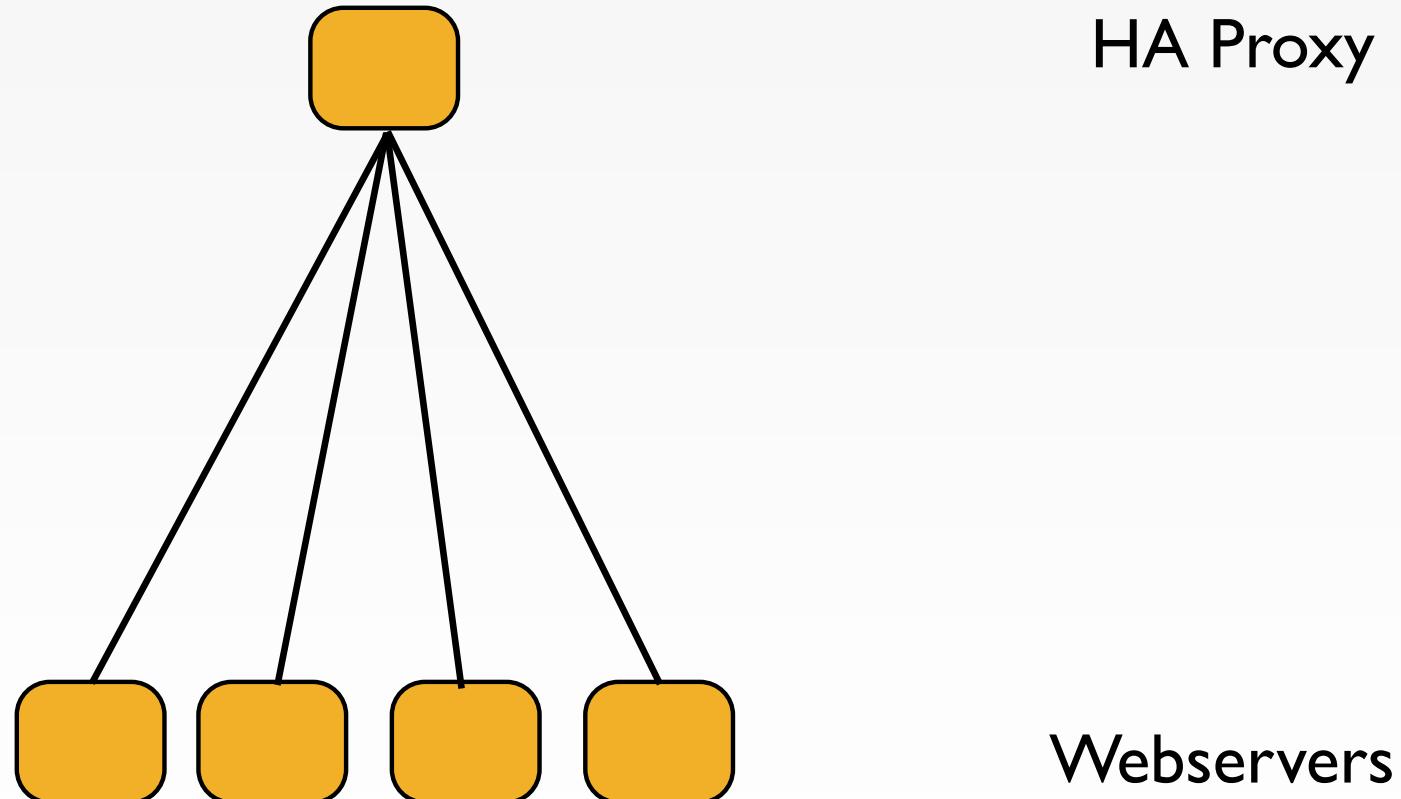
# Desired Configuration



# Chef Server – Desired & Current State



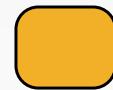
# HA Proxy Configuration



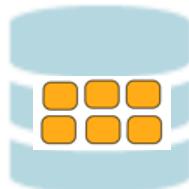
# HA Proxy Configuration



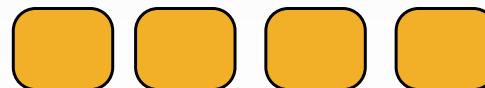
Chef  
Server



HA Proxy



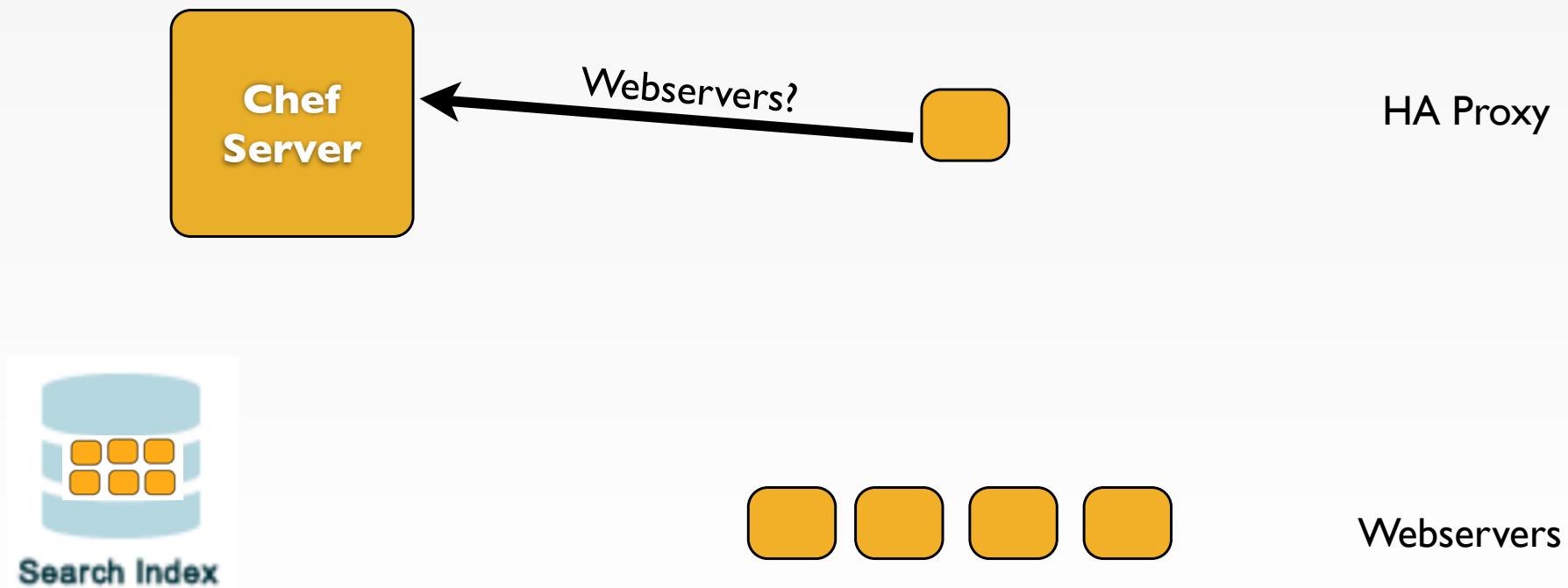
Search Index



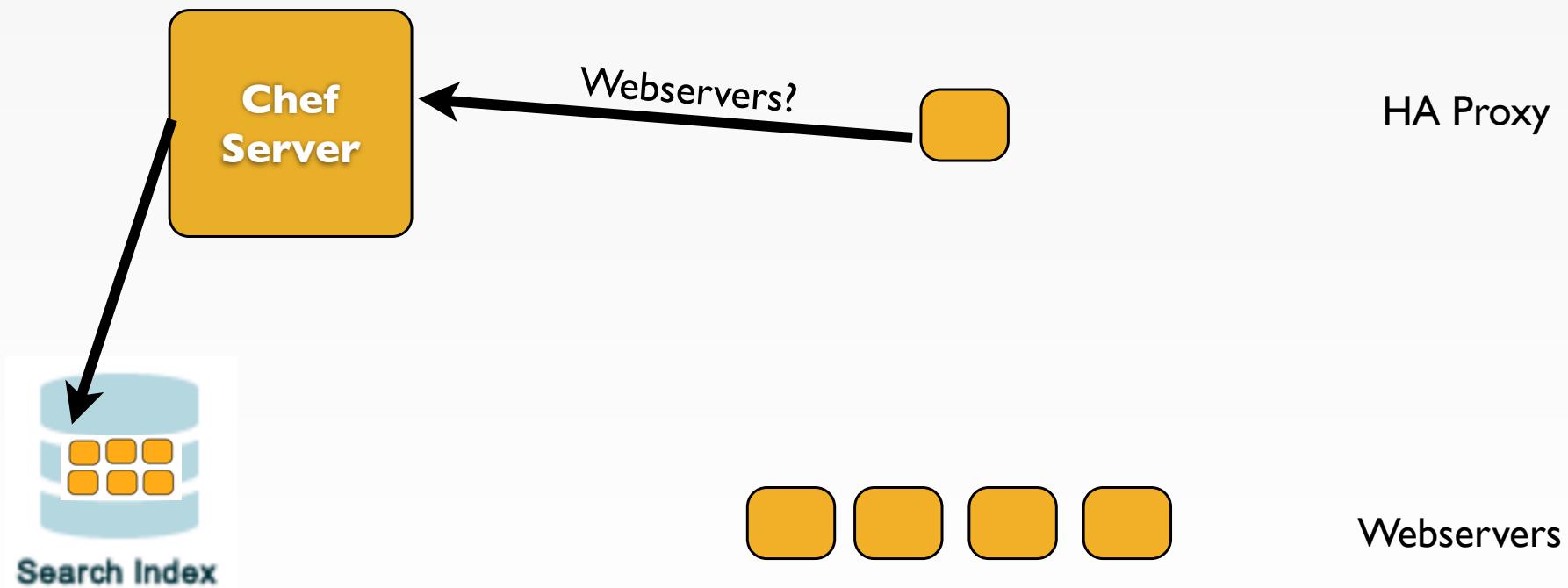
Webservers



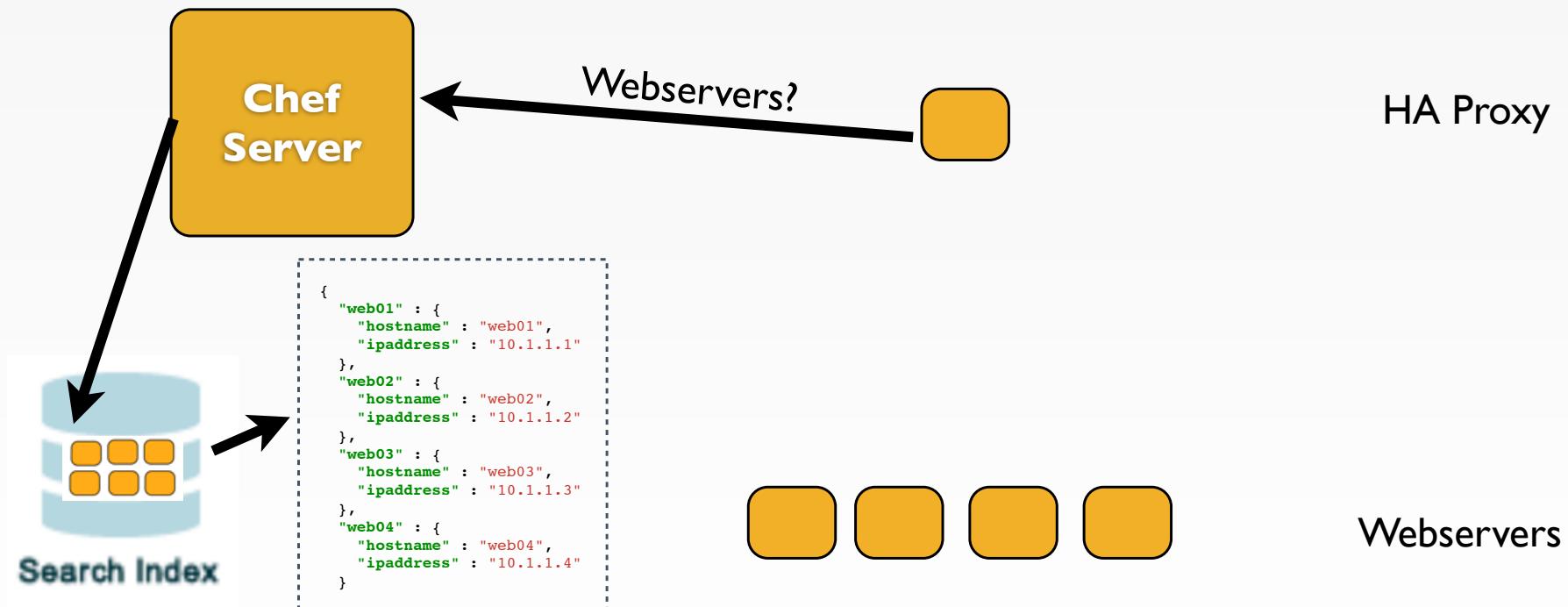
# HA Proxy Configuration



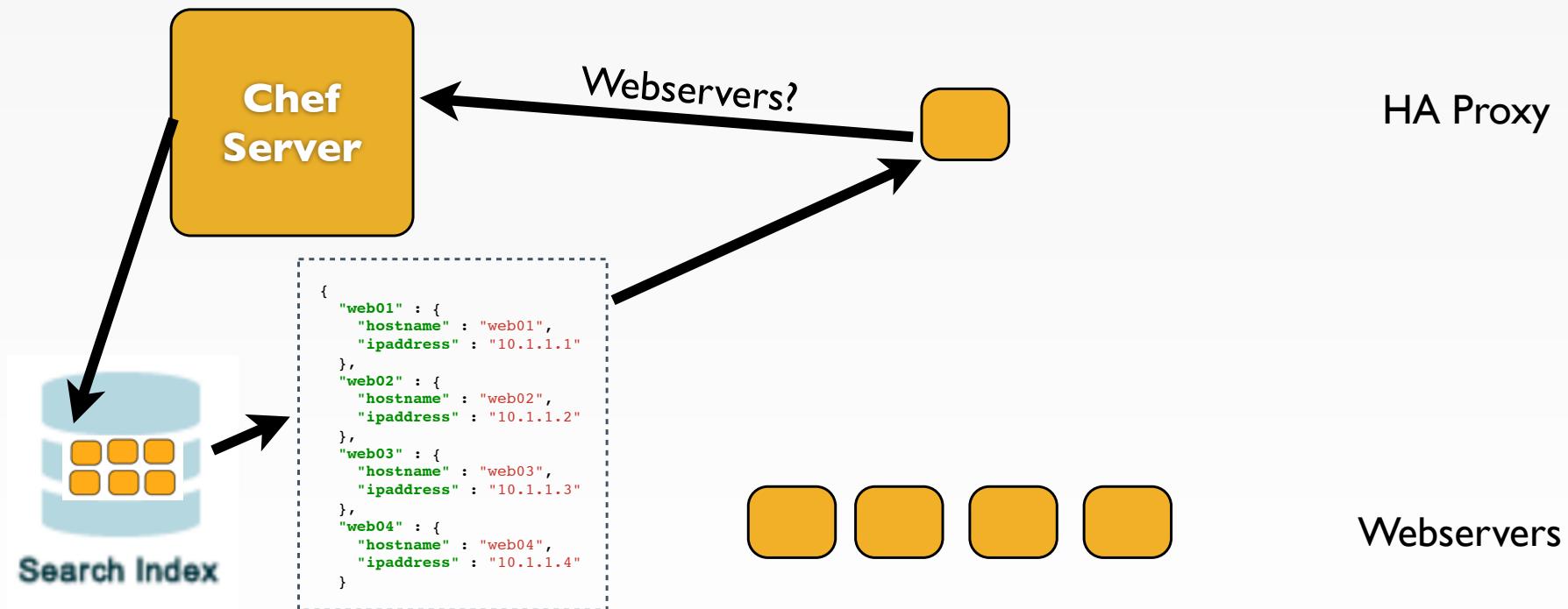
# HA Proxy Configuration



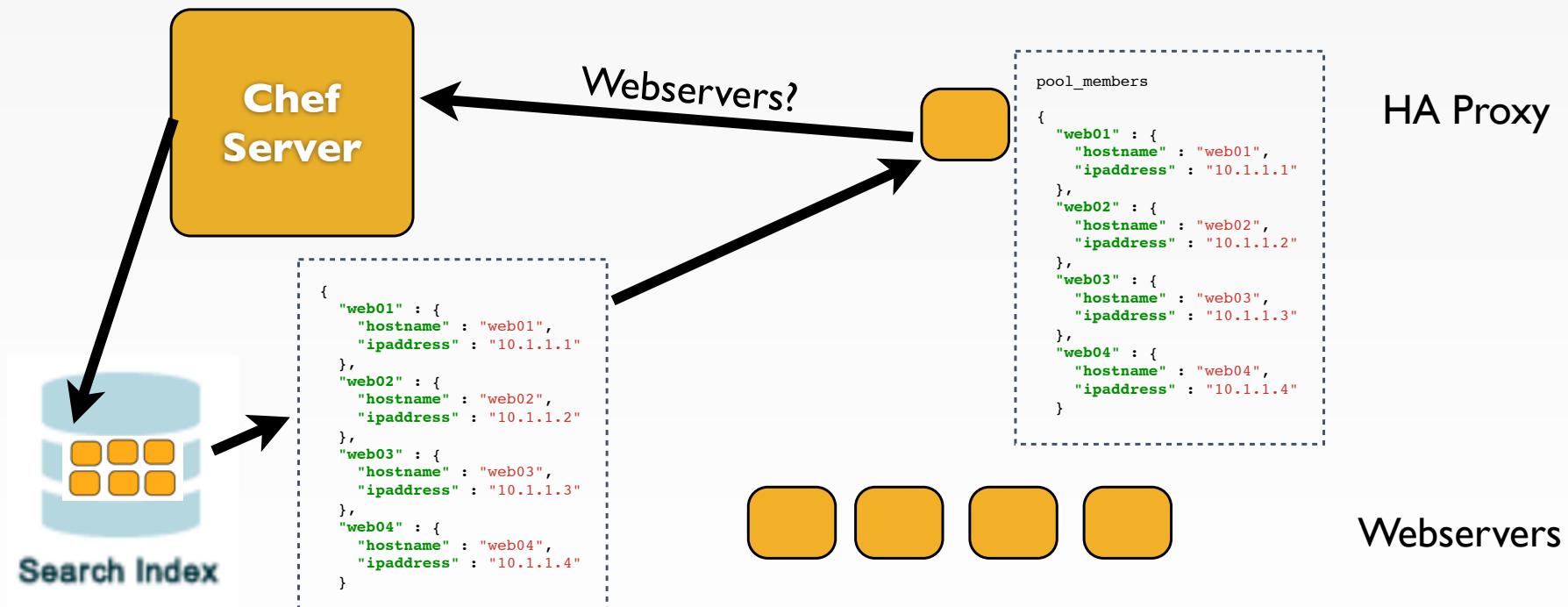
# HA Proxy Configuration



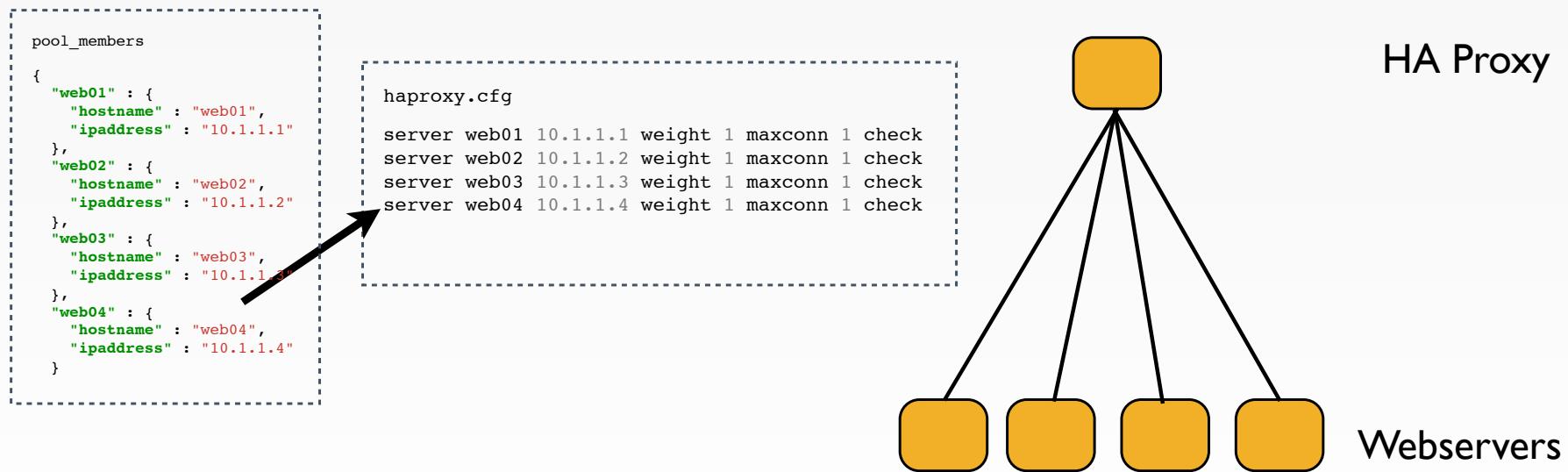
# HA Proxy Configuration



# HA Proxy Configuration



# HA Proxy Configuration





# Building your policy

Resources and Recipes



# Resources

- Piece of the system and its desired state

# Resources - Package

Package that should be installed

```
package "mysql-server" do
  action :install
end
```



# Resources - Service

Service that should be running and restarted on reboot

```
service "iptables" do
  action [ :start, :enable ]
end
```



# Resources - Service

File that should be generated

```
file "/etc/motd" do
  content "Property of Chef Software"
end
```



# Resources - Cron

Cron job that should be configured

```
cron "restart webserver" do
  hour '2'
  minute '0'
  command 'service httpd restart'
end
```

# Resources - User

User that should be managed

```
user "nginx" do
  comment "Nginx user <nginx@example.com>"
  uid 500
  gid 500
  supports :manage_home => true
end
```



# Resources - DSC

DSC resource that  
should be run

```
dsc_script 'emacs' do
  code <<-EOH
    Environment 'texteditor'
  {
    Name = 'EDITOR'
    Value = 'c:\\emacs\\bin\\emacs.exe'
  }
EOH
end
```

# Resources – Registry Key

Registry key that should be created

```
registry_key "HKEY_LOCAL_MACHINE\  
  \SOFTWARE\\Microsoft\\Windows\\  
  \CurrentVersion\\Policies\\System"  
do  
  values [ {  
    :name => "EnableLUA",  
    :type => :dword,  
    :data => 0  
  } ]  
  action :create  
end
```

# Resources

- Piece of the system and its desired state
- <http://docs.chef.io/chef/resources.html>

## Lab – Install a text editor

- **Problem:** Our workstation does not have \$EDITOR installed
- **Success Criteria:** You can edit files with \$EDITOR
- \$EDITOR is your favorite command line text editor: vim, emacs, or nano

# Login to your lab machine

```
$ ssh chef@54.164.75.30
```

```
The authenticity of host '54.165.227.226  
(54.165.227.226)' can't be established.  
RSA key fingerprint is c1:ec:ab:66:fb:22:4a:  
8f:c2:c5:9b:26:77:f3:dd:b3.  
Are you sure you want to continue connecting  
(yes/no)? yes  
Warning: Permanently added  
'54.165.227.226' (RSA) to the list of known  
hosts.  
chef@54.165.227.226's password:
```



# Welcome to your workstation

- ChefDK version 0.4.0 is installed
  - chef --version
- Chef user has passwordless sudo access
  - sudo cat /etc/shadow

# Is \$EDITOR installed?

```
$ which vim
```

```
/usr/bin/which: no vim in (/opt/
chefdk/bin:/home/chef/.chefdk/gem/
ruby/2.1.0/bin:/opt/chefdk/embedded/
bin:/usr/local/bin:/bin:/usr/bin:/
usr/local/sbin:/usr/sbin:/sbin:/
home/chef/bin)
```



# chef-apply

- chef-apply is an executable program that allows you to work with resources
- Is included as part of the ChefDK
- A great way to explore resources
- NOT how you'll eventually use Chef in production

# Install vim

```
$ sudo chef-apply -e "package 'vim'"
```

```
Recipe: (chef-apply cookbook) ::(chef-apply recipe)
* package[vim] action install
  - install version 7.2.411-1.8.el6 of package vim-enhanced
```



# Install emacs

```
$ sudo chef-apply -e "package 'emacs'"
```

```
Recipe: (chef-apply cookbook) :: (chef-apply recipe)
```

- \* package[emacs] action install
  - install version 23.1-25.el6 of package emacs

# Install nano

```
$ sudo chef-apply -e "package 'nano'"
```

```
Recipe: (chef-apply cookbook) :: (chef-apply  
recipe)
```

```
* package[nano] action install  
- install version 2.0.9-7.el6 of package nano
```

# Resources

- Describe the desired state
- Do not need to tell Chef how to get there
- What happens if you re-run the chef-apply command?

# Install \$EDITOR again with chef-apply

```
$ sudo chef-apply -e "package 'vim'"
```

```
Recipe: (chef-apply cookbook) :: (chef-apply
recipe)
  * package[vim] action install (up to date)
```

# Test and Repair

Resources follow a test  
and repair model

```
package "vim"
```



# Test and Repair

Resources follow a **test** and repair model

package "vim"

**Test** Is vim installed?



# Test and Repair

Resources follow a **test** and repair model

package "vim"

Test Is vim installed?

Yes

# Test and Repair

Resources follow a **test** and repair model

package "vim"

Test Is vim installed?

Yes

Done

# Test and Repair

Resources follow a **test** and repair model

package "vim"

Test Is vim installed?

Yes

Done

No

# Test and Repair

Resources follow a **test** and repair model

package "vim"

Test Is vim installed?

Yes

Done

No

Install it

# Test and Repair

Resources follow a **test** and **repair** model

package "vim"

Test Is vim installed?

Yes

Repair

Done

No

Install it

## Resources – Test and Repair

- Resources follow a test and repair model
- Resource currently in the desired state? (test)
  - Yes – Do nothing
  - No – Bring the resource into the desired state (repair)

# Resources

- package
- template
- service
- directory
- user
- group
- dsc\_script
- registry\_key
- powershell\_script
- cron
- mount
- route
- ...and more!



# Recipes

- Policy is defined as a collection of **resources** in **recipes**. There are lots of abstractions on top of this but **resources** are the basic building blocks.



# **Test-driven Infrastructure**

Change policy with confidence



# Our process

- Write policy
  - Apply policy
  - Verify policy
- 
- Not bad for the simple case, will quickly get untenable

# Faster Feedback

- Speed-up the feedback loops with automated testing.
- Have confidence in your changes before you run them in production

# Chef Testing

- Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code follow our style guide?

# Test-driving infrastructure

- We are going to use a relatively simple scenario
- We are going to explore many facets of testing
- We are going to follow a test-first, test-driven model

# Our Scenario

- We want a simple rails app deployed on a server.
- We will only have time to build some of it in today's workshop.
- You will leave with working code

# Create a directory for your Chef project

```
$ chef generate repo chef-repo
```

```
Compiling Cookbooks...
Recipe: code_generator::repo
 * directory[/home/chef/chef-repo] action create
   - create new directory /home/chef/chef-repo
 * template[/home/chef/chef-repo/LICENSE] action create
   - create new file /home/chef/chef-repo/LICENSE
   - update content in file /home/chef/chef-repo/LICENSE from none to aed48c
     (diff output suppressed by config)
 * cookbook_file[/home/chef/chef-repo/README.md] action create
   - create new file /home/chef/chef-repo/README.md
   - update content in file /home/chef/chef-repo/README.md from none to 69567b
     (diff output suppressed by config)
 * cookbook_file[/home/chef/chef-repo/Rakefile] action create
   - create new file /home/chef/chef-repo/Rakefile
```



# Change into that directory

```
$ cd cookbooks
```

# Create an apache cookbook

```
$ chef generate cookbook cookbooks/apache
```

```
Compiling Cookbooks...
```

```
Recipe: code_generator::cookbook
  * directory[/home/chef/chef-repo/cookbooks/apache] action create
    - create new directory /home/chef/chef-repo/cookbooks/apache
  * template[/home/chef/chef-repo/cookbooks/apache/metadata.rb] action
create_if_missing
    - create new file /home/chef/chef-repo/cookbooks/apache/metadata.rb
    - update content in file /home/chef/chef-repo/cookbooks/apache/
metadata.rb from none to 4c0e2d
      (diff output suppressed by config)
  * template[/home/chef/chef-repo/cookbooks/apache/README.md] action
create_if_missing
```



## Questions to ask when testing

- Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

# Chef client success status

- Requirements to verify chef-client success:
  - A place to store the cookbook artifact

# Chef client success status

- Requirements to verify chef-client success:
  - A place to store the cookbook artifact
  - A chef-client with access to the cookbook

# Chef client success status

- Requirements to verify chef-client success:
  - A place to store the cookbook artifact
  - A chef-client with access to the cookbook
  - A target server running the same OS as production

# Test Kitchen

- Test harness to execute code on one or more platforms
- Driver plugins to allow your code to run on various cloud and virtualization providers
- Includes support for many testing frameworks
- Included with ChefDK



# Test Matrix

- Two operating systems

ubuntu-12.04
centos-6.4

# Test Matrix

- Two operating systems
- One recipe

	<b>default</b>
ubuntu-12.04	apache::default
centos-6.4	apache::default

# Test Matrix

- Two operating systems
- Two recipes

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl

# Test Matrix

- Three operating systems
- Two recipes

	<b>default</b>	<b>ssl</b>
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl
ubuntu-14.04	apache::default	apache::ssl

# Configuring the Kitchen



**OPEN IN EDITOR:** cookbooks/apache/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

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

**SAVE FILE!**



# .kitchen.yml

- driver - virtualization or cloud provider

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

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



# .kitchen.yml

- **provisioner** - application to configure the node

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

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



# .kitchen.yml

- platforms - target operating systems

```
---
```

```
driver:
  name: vagrant
```

```
provisioner:
  name: chef_zero
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
```

```
suites:
  - name: default
    run_list:
      - recipe[apache::default]
```

```
    attributes:
```



# .kitchen.yml

- suites - target configurations

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

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



CHEF

# .kitchen.yml

	default
ubuntu-12.04	apache::default
centos-6.4	apache::default

```
---
```

```
driver:
```

```
  name: vagrant
```

```
provisioner:
```

```
  name: chef_zero
```

```
platforms:
```

```
  - name: ubuntu-12.04
```

```
  - name: centos-6.4
```

```
suites:
```

```
  - name: default
```

```
    run_list:
```

```
      - recipe[apache::default]
```



# .kitchen.yml

	<b>default</b>	<b>ssl</b>
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl

```
---
```

```
driver:
```

```
  name: vagrant
```

```
provisioner:
```

```
  name: chef_zero
```

```
platforms:
```

```
  - name: ubuntu-12.04
```

```
  - name: centos-6.4
```

```
suites:
```

```
  - name: default
```

```
    run_list:
```

```
      - recipe[apache::default]
```

```
  - name: ssl
```

```
    run_list:
```

```
      - recipe[apache::ssl]
```



# .kitchen.yml

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl
ubuntu-14.04	apache::default	apache::ssl

```
--  
driver:  
  name: vagrant  
  
provisioner:  
  name: chef_zero  
  
platforms:  
  - name: ubuntu-12.04  
  - name: centos-6.4  
  - name: ubuntu-14.04  
  
suites:  
  - name: default  
    run_list:  
      - recipe[apache::default]  
  - name: ssl  
    run_list:  
      - recipe[apache::ssl]
```



# Move to the apache cookbook directory

```
$ cd ~/chef-repo/cookbooks/apache
```

# Update .kitchen.yml



**OPEN IN EDITOR:** .kitchen.yml

```
---
```

```
driver:
  name: docker
```

```
provisioner:
  name: chef_zero
```

```
platforms:
  - name: ubuntu-12.04
```

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

**SAVE FILE!**



# List the Test Kitchens

```
$ kitchen list
```

Instance	Driver	Provisioner	Last Action
default-ubuntu-1204	Docker	ChefZero	<Not Created>

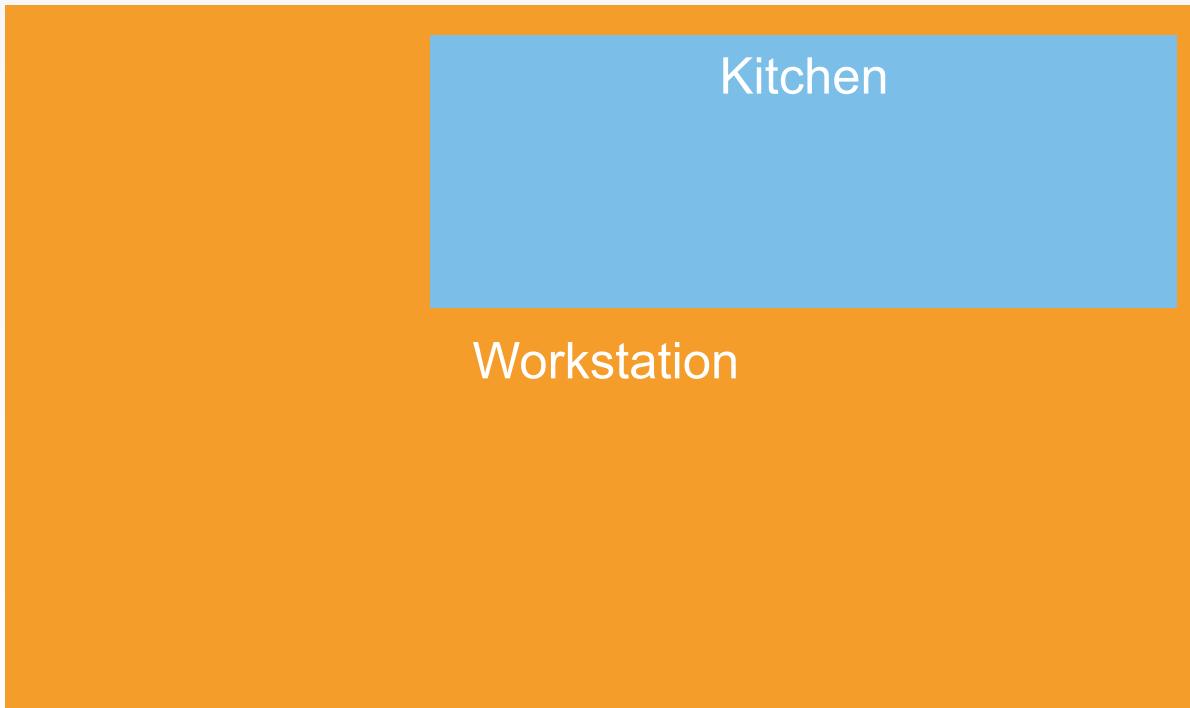
# Create the kitchen

```
$ kitchen create
```

```
----> Starting Kitchen (v1.3.1)
----> Creating <default-ubuntu-1204>...
      Sending build context to Docker daemon  2.56 kB
      Sending build context to Docker daemon
      Step 0 : FROM ubuntu:12.04
        ---> 0b310e6bf058
      Step 1 : RUN dpkg-divert --local --rename --add /sbin/initctl
        ---> Running in 73201c2a8836
      Leaving 'local diversion of /sbin/initctl to /sbin/initctl.distrib'
        ---> c8039cd87665
      Removing intermediate container 73201c2a8836
      Step 2 : RUN ln -sf /bin/true /sbin/initctl
        ---> Running in 4e79ba940fe4
        ---> ecc3ffe49a30
      Removing intermediate container 4e79ba940fe4
```



# Kitchen created



# Login to the kitchen

```
$ kitchen login
```

```
kitchen@localhost's password:
```

# Login to the kitchen

```
$ kitchen login
```

```
kitchen@localhost's password: kitchen
```

# Login to the kitchen

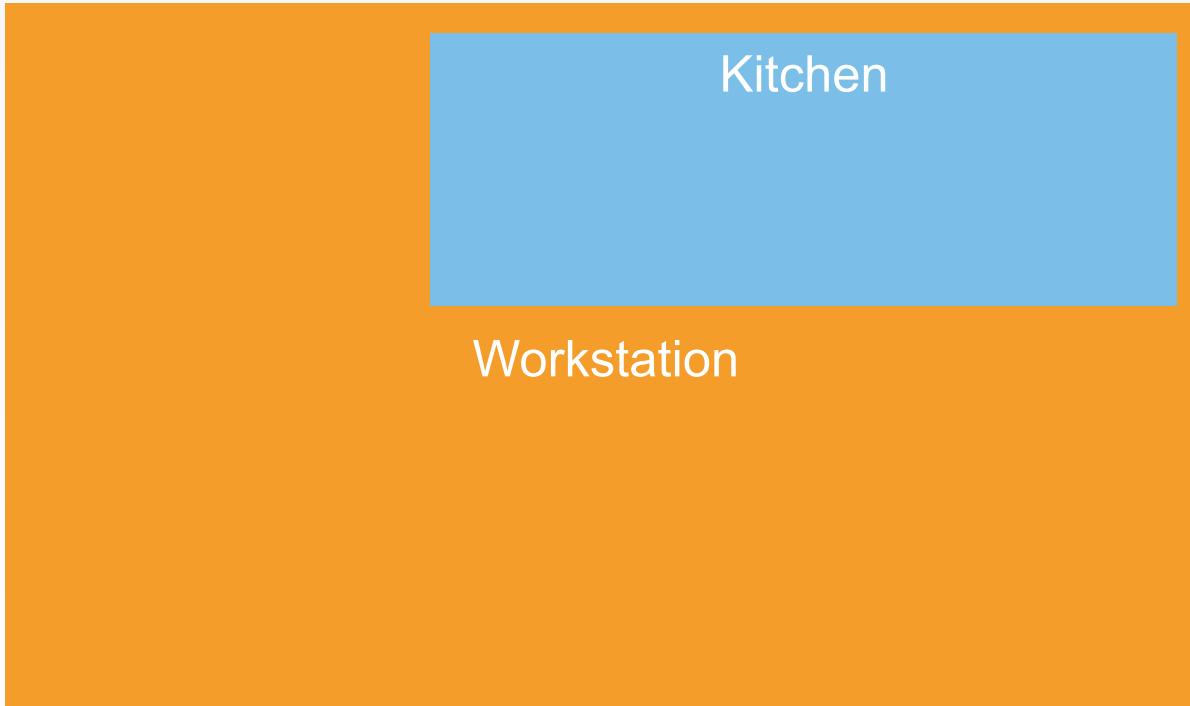
```
$ kitchen login
```

```
kitchen@localhost's password: kitchen
```

```
Last login: Wed Sep 24 04:30:29 2014 from 172.17.42.1
```

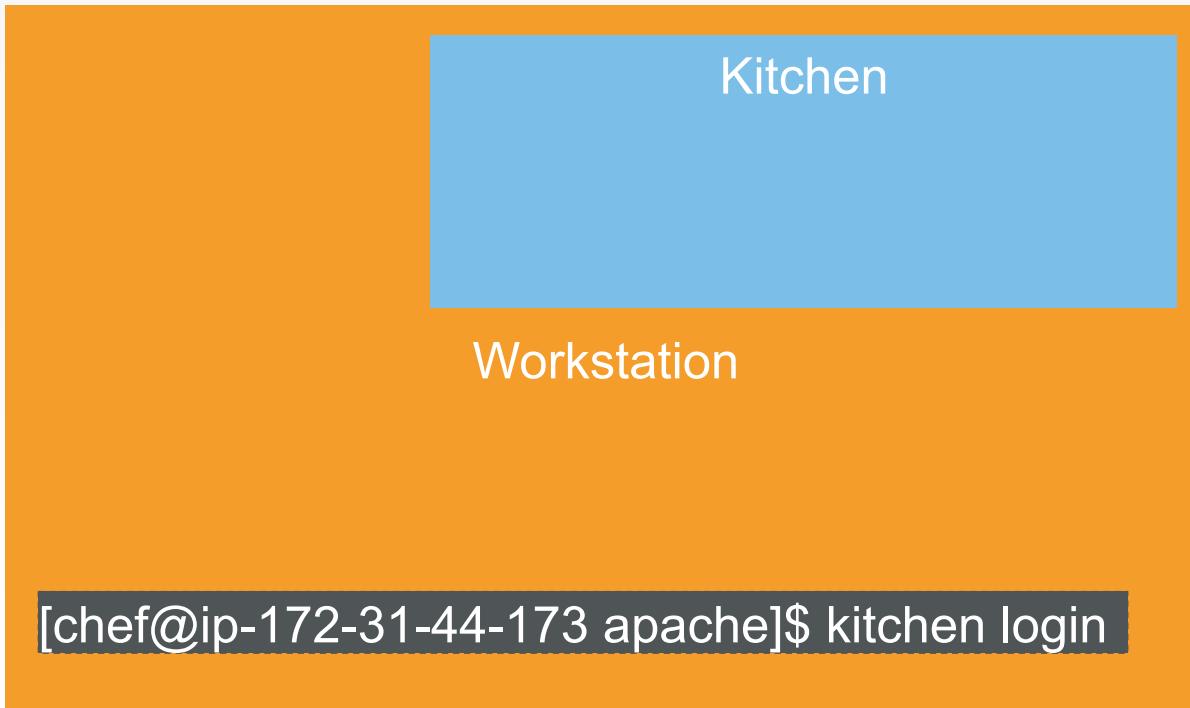


# Kitchen login

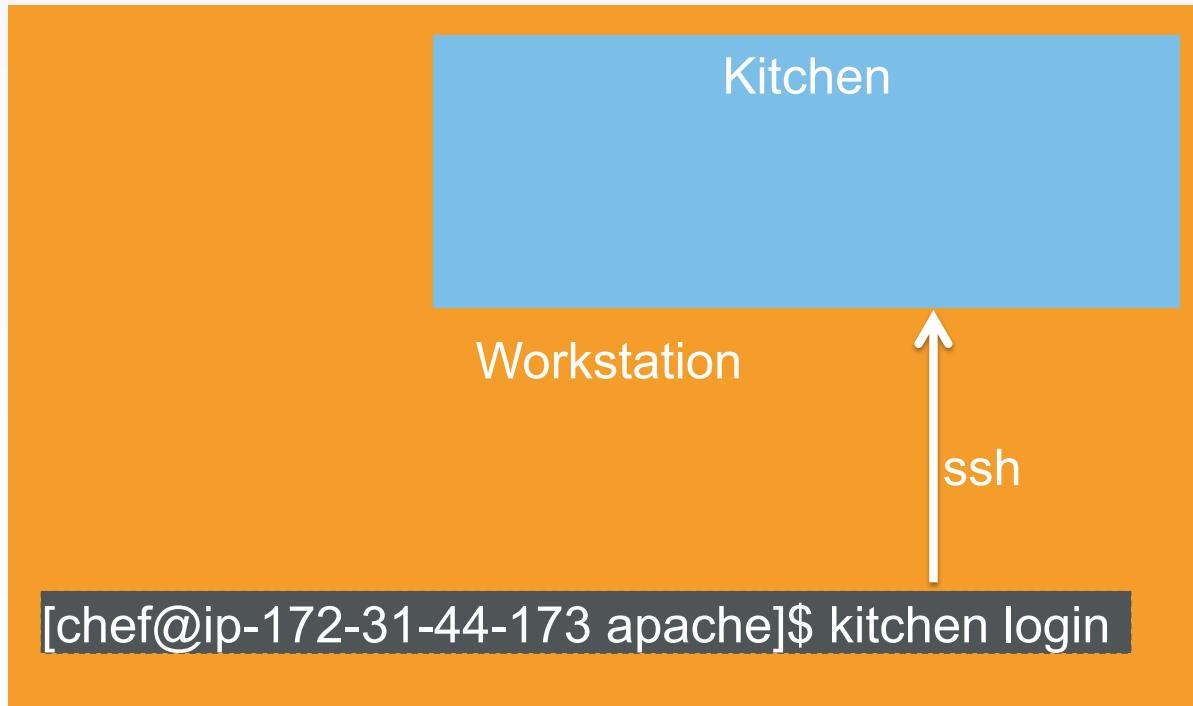


Workstation

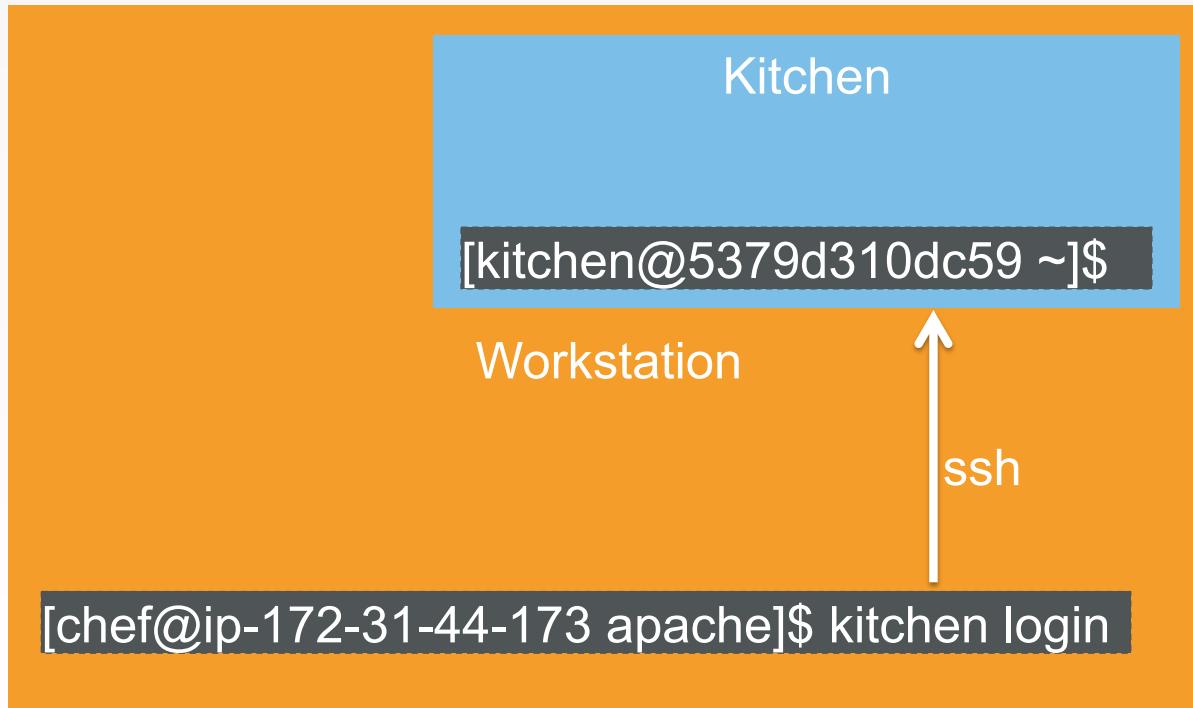
# Kitchen login



# Kitchen login



# Kitchen login



# Chef client success status

- Requirements to verify chef-client success:
  - A target server running the same OS as production
  - A chef-client with access to the cookbook

## Lab – Apply our policy

- **Problem:** We have not applied our policy to the test environment.
- **Success Criteria:** The default apache recipe will be applied in the test environment

# Leave the kitchen

```
$ exit
```

```
logout
```

```
Connection to localhost closed.
```

# Go to the right place

```
$ cd ~/chef-repo/cookbooks/apache
```

# Apply our policy

```
$ kitchen converge
```

```
----> Starting Kitchen (v1.3.1)
----> Converging <default-ubuntu-1204>...
      Preparing files for transfer
      Preparing dna.json
      Resolving cookbook dependencies with Berkshelf 3.2.3...
      Removing non-cookbook files before transfer
      Preparing validation.pem
      Preparing client.rb
----> Installing Chef Omnibus (install only if missing)
      downloading https://www.chef.io/chef/install.sh
          to file /tmp/install.sh
      trying wget...
      sudo: /etc/sudoers.d/kitchen is mode 0644, should be 0440
      Downloading Chef for ubuntu...
```



# Kitchen converge



## Questions to ask when testing

- ✓ Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?



# Verifying node state

Serverspec



# Manually inspect the test node

```
$ kitchen login
```

```
kitchen@localhost's password:
```

# Manually inspect the test node

```
$ kitchen login
```

```
kitchen@localhost's password: kitchen
```

# Manually inspect the test node

```
$ kitchen login
```

```
kitchen@localhost's password: kitchen
```

```
Last login: Wed Sep 24 04:30:29 2014 from 172.17.42.1
```

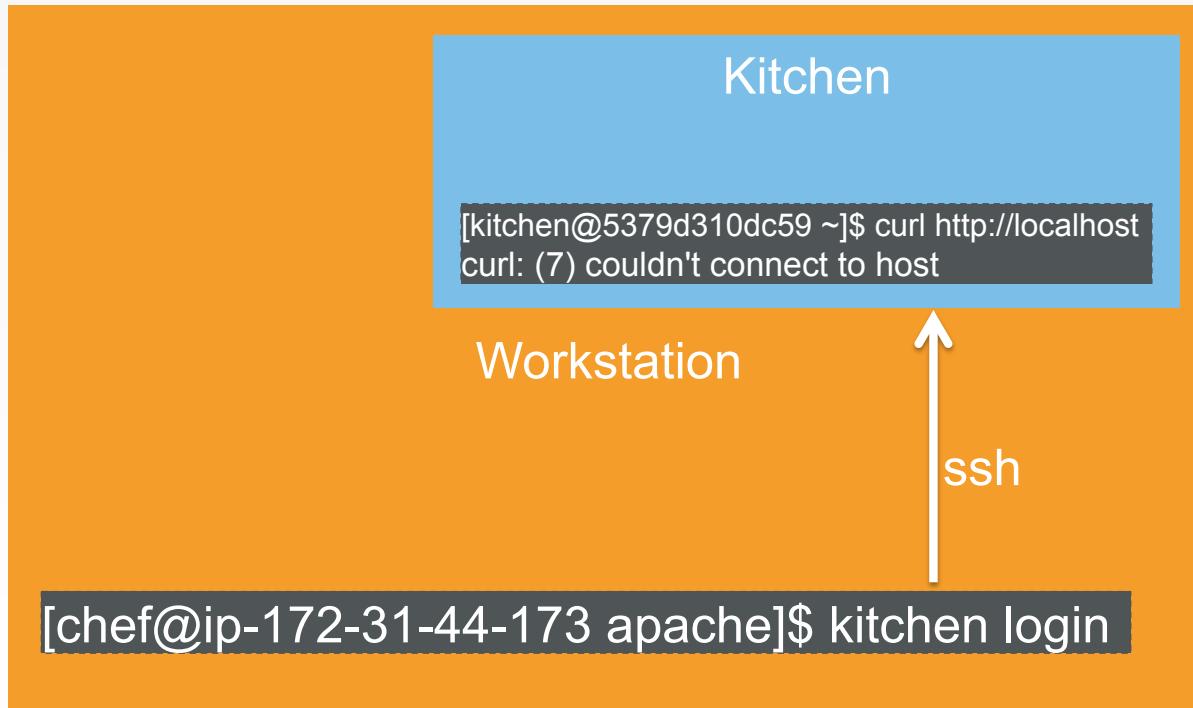


# Manually inspect the test node

```
$ curl http://localhost
```

```
curl: (7) couldn't connect to host
```

# Kitchen login



## Lab – Verify node state

- **Problem:** Manually verifying the state of the test node is tedious and error-prone.
- **Success Criteria:** The end state of the node is automatically tested.

# Serverspec

- Write tests to verify your servers
- Not dependent on Chef
- Defines many resource types
  - package, service, user, etc.
- Works well with Test Kitchen
- <http://serverspec.org/>



# Leave the Kitchen

```
$ exit
```

```
logout
```

```
Connection to localhost closed.
```

# Move to the proper directory

```
$ cd ~/chef-repo/cookbooks/apache
```

# Write a Serverspec test



**OPEN IN EDITOR:** `test/integration/default/serverspec/default_spec.rb`

```
require 'spec_helper'

describe 'apache::default' do

  # Serverspec examples can be found at
  # http://serverspec.org/resource_types.html

  it 'does something' do
    skip 'Replace this with meaningful tests'
  end

end
```

**SAVE FILE!**

# Generic Expectation Form

```
describe "<subject>" do
  it "<description>" do
    expect(thing).to eq result
  end
end
```

# Awesome Expectations



**OPEN IN EDITOR:** test/integration/default/serverspec/default\_spec.rb

```
require 'spec_helper'

describe "apache::default" do
  it "is awesome" do
    expect(true).to eq true
  end
end
```

**SAVE FILE!**



# Run the serverspec test

```
$ kitchen verify
```

```
-----> Running serverspec test suite
/opt/chef/embedded/bin/ruby -I/tmp/busser/suites/serverspec -I/tmp/
busser/gems/gems/rspec-support-3.1.2/lib:/tmp/busser/gems/gems/rspec-
core-3.1.7/lib /opt/chef/embedded/bin/rspec --pattern /tmp/busser/suites/
serverspec/\*\*/\* spec.rb --color --format documentation --default-path /
tmp/busser/suites/serverspec

apache::default
  is awesome

Finished in 0.02823 seconds (files took 0.99875 seconds to load)
1 example, 0 failures
Finished verifying <default-centos-64> (0m5.03s).
```

## How would you test our criteria?

- What would you test to make sure apache is running?

# Verify package is installed



**OPEN IN EDITOR:** [test/integration/default/serverspec/default\\_spec.rb](#)

```
require 'spec_helper'

describe "apache" do
  it "is awesome" do
    expect(true).to eq true
  end

  it "is installed" do
    expect(package("apache2")).to be_installed
  end
end
```

**SAVE FILE!**



# Exercise the test

```
$ kitchen verify
```

```
apache::default
  is awesome
  is installed (FAILED - 1)
```

Failures:

```
1) apache::default is installed
Failure/Error: expect(package 'apache2').to be_installed
expected Package "apache2" to be installed
/bin/sh -c dpkg-query\ -f\ \'\$Status\}\'\ \ -w\ apache2\ \|\ \
grep\ -E\ '\^\\(install\\|hold\\)\\\ ok\\ installed\$\'\'
No packages found matching apache2.
```

# Test is failing, make it pass

- Test-driven development involves
  - Write a test to verify something is working
  - Watch the test fail
  - Write just enough code to make the test pass
  - Repeat

# Update our cookbook



**OPEN IN EDITOR:** `~/chef-reop/cookbooks/apache/recipes/default.rb`

```
package "apache2"
```

**SAVE FILE!**

# Converge the node again

```
$ kitchen converge
```

```
----> Starting Kitchen (v1.3.1)
----> Converging <default-ubuntu-1204>...
      Preparing files for transfer
      Preparing dna.json
      Resolving cookbook dependencies with Berkshelf 3.2.3...
      Removing non-cookbook files before transfer
      Preparing validation.pem
      Preparing client.rb
----> Chef Omnibus installation detected (install only if missing)
      sudo: /etc/sudoers.d/kitchen is mode 0644, should be 0440
      Transferring files to <default-ubuntu-1204>
      sudo: /etc/sudoers.d/kitchen is mode 0644, should be 0440
      Starting Chef Client, version 12.2.1
      [2015-04-15T22:02:01+00:00] WARN: Child with name 'dna.json' found in multiple directories: /tmp/
kitchen/dna.json and /tmp/kitchen/dna.json
      resolving cookbooks for run list: ["apache::default"]
```



# Exercise the test

```
$ kitchen verify
```

```
apache
    is awesome
    is installed
```

```
    Finished in 0.48165 seconds (files took 1.05
seconds to load)
```

```
    2 examples, 0 failures
```

```
    Finished verifying <default-centos-64>
(0m5.64s).
```

```
----> Kitchen is finished. (0m11.84s)
```



## What else will you test?

- Is the service running?
  - Is the port accessible?
  - Is the expected content being served?
- 
- Make sure everything works from a fresh kitchen, too!

# Extend the Serverspec test



**OPEN IN EDITOR:** [test/integration/default/serverspec/default\\_spec.rb](#)

```
describe 'apache' do
  it "is installed" do
    expect(package 'apache2').to be_installed
  end

  it "is running" do
    expect(service 'apache2').to be_running
  end

  it "responds to http requests" do
    expect(command("curl localhost")).exit_status.to eq 0
  end
end
```

**SAVE FILE!**



# Verify the kitchen

```
$ kitchen verify
```

```
Failures:
```

- 1) apache::default is running  
Failure/Error: expect(service 'apache2').to be\_running  
expected Service "apache2" to be running  
/bin/sh -c service\ apache2\ status\ \&\&\ service\ apache2\ status\ \| grep\  
\'running\'  
Apache2 is NOT running.
  
- 2) apache::default responds to http requests  
Failure/Error: expect(command("curl localhost").exit\_status).to eq 0  
expected: 0  
got: 7

# Update our cookbook



**OPEN IN EDITOR:** `~/chef-repo/cookbooks/apache/recipes/default.rb`

```
package "apache2"

service "apache2" do
  action :start
end
```

**SAVE FILE!**



# Converge the node again

```
$ kitchen converge
```

```
----> Starting Kitchen (v1.3.1)
----> Converging <default-ubuntu-1204>...
      Preparing files for transfer
      Preparing dna.json
      Resolving cookbook dependencies with Berkshelf 3.2.3...
      Removing non-cookbook files before transfer
      Preparing validation.pem
      Preparing client.rb
----> Chef Omnibus installation detected (install only if missing)
      sudo: /etc/sudoers.d/kitchen is mode 0644, should be 0440
      Transferring files to <default-ubuntu-1204>
      sudo: /etc/sudoers.d/kitchen is mode 0644, should be 0440
      Starting Chef Client, version 12.2.1
      [2015-04-15T22:02:01+00:00] WARN: Child with name 'dna.json' found in multiple directories: /tmp/
kitchen/dna.json and /tmp/kitchen/dna.json
      resolving cookbooks for run list: ["apache::default"]
```



# Verify the kitchen

```
$ kitchen verify
```

```
apache::default
  is awesome
  is installed
  is running
  responds to http requests
```

```
load)      Finished in 0.9764 seconds (files took 1.22 seconds to
           4 examples, 0 failures
```

```
           Finished verifying <default-ubuntu-1204> (0m9.78s).
-----> Kitchen is finished. (0m11.23s)
```

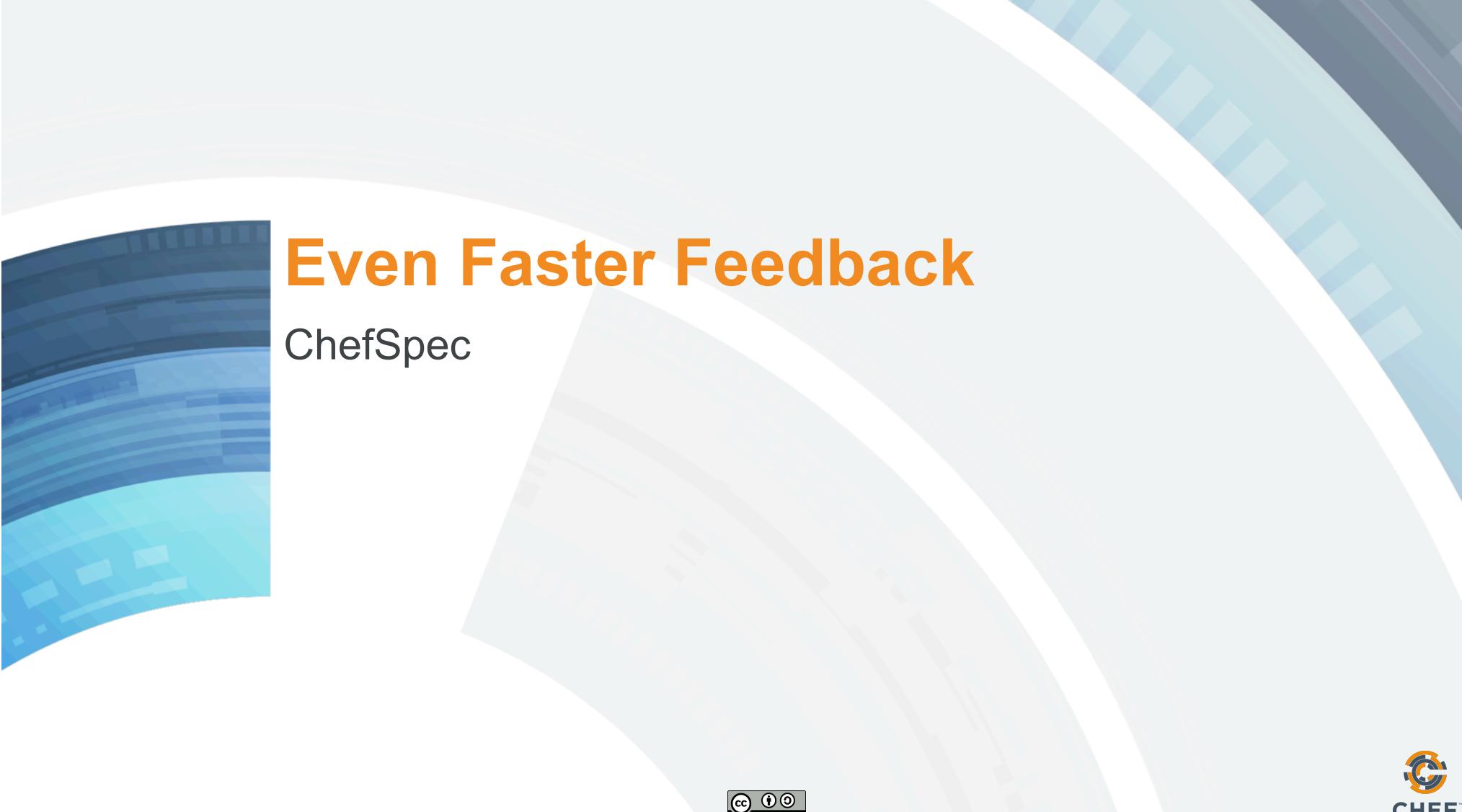


# Kitchen Workflow

- kitchen create
- kitchen converge
- kitchen verify
- kitchen destroy
- All at once with kitchen test

# Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
  - Are the resources properly defined?
  - Does the code following our style guide?



# Even Faster Feedback

ChefSpec



# Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
  - Are the resources properly defined?
  - Does the code following our style guide?

# This is too slow!

- To test our code, we need to spin up a test kitchen, converge a node, execute some tests.
- Our simple test case takes about 2 minutes to fully execute.

# Properly configured resources

- We need a way to verify that the resources in our recipes are properly configured
- We want to get faster feedback

## Lab – Verify the resources

- **Problem:** We should be able to catch errors before we need to converge a node
- **Success Criteria:** Catch a typo prior to converge

# ChefSpec

- Test before you converge
- Get feedback on cookbook changes without the need for target servers



## ChefSpec

gem v4.2.0 build passing

ChefSpec is a unit testing framework for testing Chef cookbooks. ChefSpec makes it easy to write examples and get fast feedback on cookbook changes without the need for virtual machines or cloud servers.

ChefSpec runs your cookbook(s) locally with Chef Solo without actually converging a node. This has two primary benefits:

- It's really fast!
- Your tests can vary node attributes, operating systems, and search results to assert behavior under varying conditions.

<http://sethvargo.github.io/chefspec/>



# Change to the apache cookbook directory

```
$ cd ~/chef-repo/cookbooks/apache
```

# Write a ChefSpec test



**OPEN IN EDITOR:** spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'apache::default' do

  context 'When all attributes are default, on an unspecified platform' do

    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      chef_run # This should not raise an error
    end
  end
end
```

**SAVE FILE!**



# Write a ChefSpec test



**OPEN IN EDITOR:** spec/unit/recipes/default\_spec.rb

```
describe 'apache::default' do
  context 'When all attributes are default, on an unspecified platform' do

    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      chef_run # This should not raise an error
    end

    it 'installs apache' do
      expect(chef_run).to install_package 'apache2'
    end
  end
end
```

**SAVE FILE!**



# Run the ChefSpec tests

```
$ rspec spec
```

```
..
```

```
Finished in 1.19 seconds (files took 13.09 seconds to load)
2 examples, 0 failures
```



# Break the cookbook



**OPEN IN EDITOR:** recipes/default.rb

```
package "apache"
```

```
service "apache2" do
  action :start
end
```

**SAVE FILE!**

# Run the ChefSpec tests

```
$ rspec spec
```

```
.F
```

```
Failures:
```

```
1) apache::default When all attributes are default, on an unspecified platform installs apache
Failure/Error: expect(chef_run).to install_package 'apache2'
expected "package[apache2]" with action :install to be in Chef run. Other package
resources:
```

```
  package[apache]
```

```
  # ./spec/unit/recipes/default_spec.rb:23:in `block (3 levels) in <top (required)>'
```

```
Finished in 1.18 seconds (files took 12.7 seconds to load)
2 examples, 1 failure
```



# Fix the cookbook



**OPEN IN EDITOR:** recipes/default.rb

```
package "apache2"
```

```
service "apache2" do
  action :start
end
```

**SAVE FILE!**

# Run the ChefSpec tests

```
$ rspec spec
```

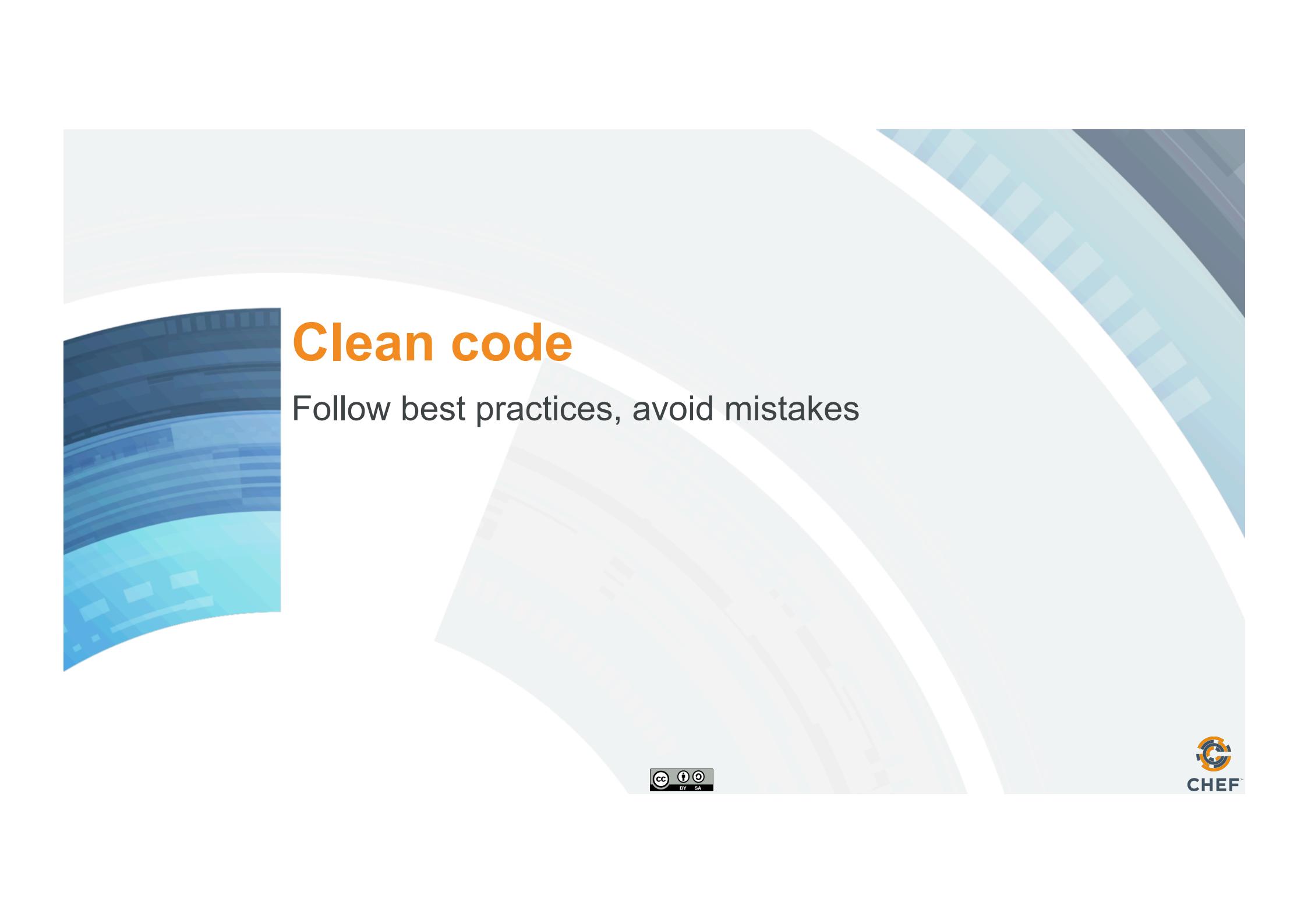
```
..
```

```
Finished in 1.19 seconds (files took 13.09 seconds to load)
2 examples, 0 failures
```



# Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- ✓ Are the resources properly defined?
- Does the code following our style guide?



# Clean code

Follow best practices, avoid mistakes



# Foodcritic

- Check cookbooks for common problems
- Style, correctness, deprecations, etc.
- Included with ChefDK



<http://www.foodcritic.io/>



# Change our recipe



**OPEN IN EDITOR:** recipes/default.rb

```
package_name = "apache2"
```

```
package "#{package_name}"
```

```
service "apache2" do
  action :start
end
```

**SAVE FILE!**

# Run Foodcritic

```
$ foodcritic .
```

```
FC002: Avoid string interpolation  
where not required: ./recipes/  
default.rb:7
```

# Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- ✓ Are the resources properly defined?
- ✓ Does the code following our style guide?

## Next steps

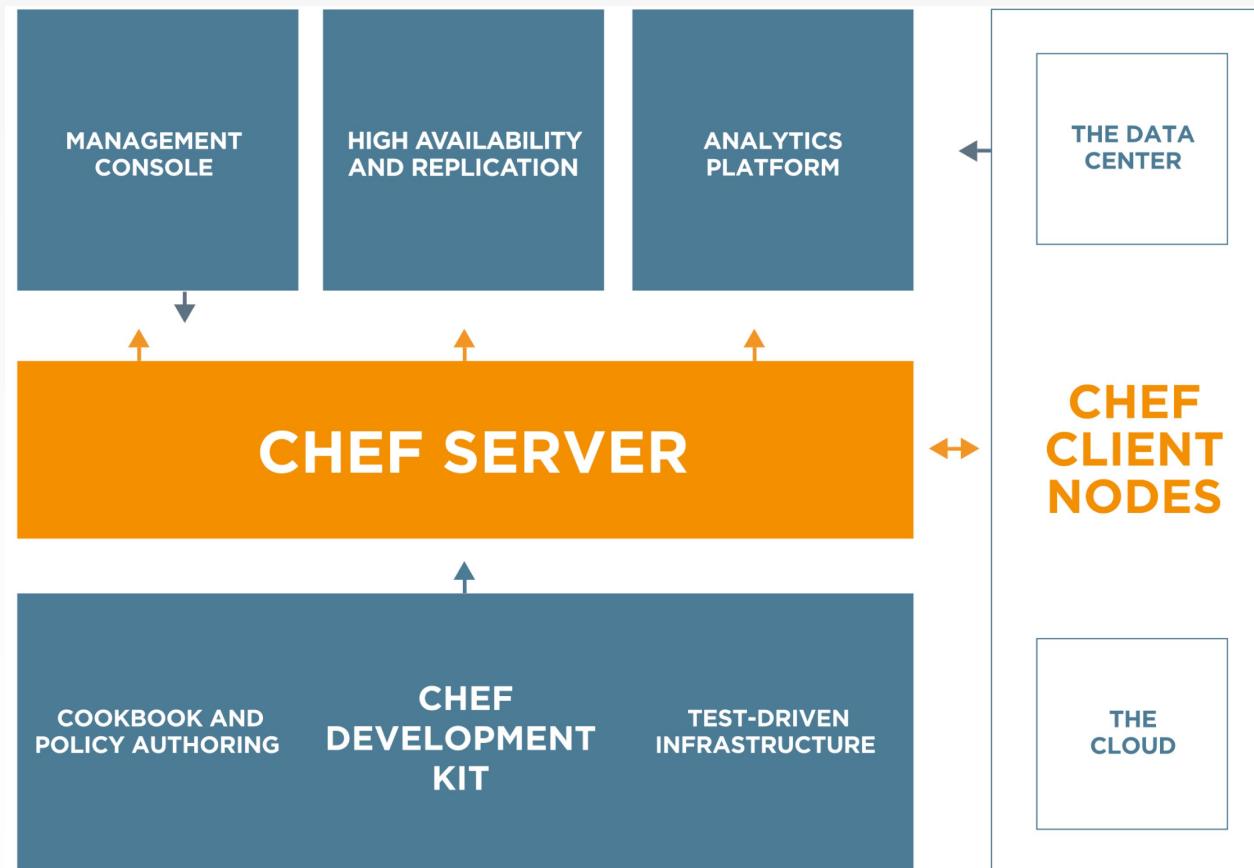
- Install ruby on the system
- Install and configure passenger
- Install and configure a database
- Deploy the application
- Demo the working solution



# Wrapping Up



# We've only scratched the surface



<https://www.chef.io/chef/>

# Build Anything

- Simple internal applications
- Complex external applications
- Workstations
- Hadoop clusters
- IaaS infrastructure
- PaaS infrastructure
- SaaS applications
- Storage systems
- You name it



<http://www.flickr.com/photos/hyku/245010680/>

# And Manage it Simply



<http://www.flickr.com/photos/helico/404640681/>

- Automatically reconfigure everything
- Linux, Windows, Unixes, BSDs
- Load balancers
- Metrics collection systems
- Monitoring systems
- Cloud migrations become trivial

# What questions do you have?

- Ask us anything!

- @nellshamrell

- @nathenharvey

- Thank you!