



Intro to Infrastructure as Code

Introduction to Chef

All Things Open – October 2014



Chef Fundamentals by [Chef Software, Inc.](#) is licensed under a
[Creative Commons Attribution-ShareAlike 4.0 International License](#).



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>



- @nathenharvey
- nharvey@getchef.com



Let's do it live!

Hangouts

Hangouts On Air Schedule

Watch video

Bring your conversations to life with photos, emoji, and even group video calls for free.

<https://www.google.com/+/learnmore/hangouts/>

Hello!

- System Administrator?

Hello!

- System Administrator?
- Developer?

Hello!

- System Administrator?
- Developer?
- Ruby developer?

Hello!

- System Administrator?
- Developer?
 - Ruby developer?
- DevOp?

Hello!

- System Administrator?
- Developer?
 - Ruby developer?
- DevOp?
- Business Person?

Are you experienced?

- Experience with Infrastructure as Code or Configuration Management?

Are you experienced?

- Experience with Infrastructure as Code or Configuration Management?
- Experience with Chef?

Which version control system do you use?

Which version control system do you use?

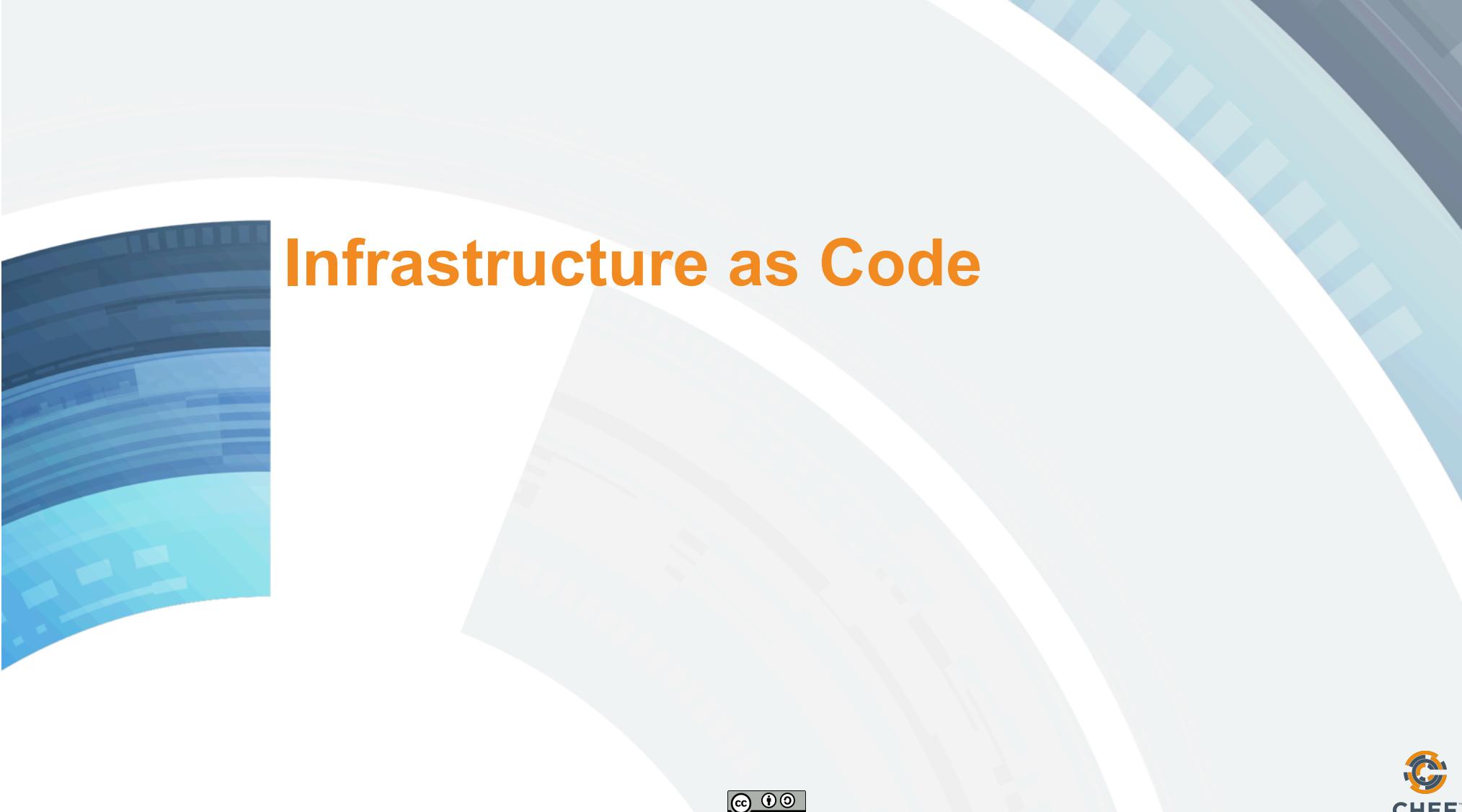
- cp foo foo.bak

Which version control system do you use?

- cp foo foo.bak
- cp foo{, .`date +%Y%m%d%H%M` }

Which version control system do you use?

- cp foo foo.bak
- cp foo{, .`date +%Y%m%d%H%M` }
- cp foo{, .`date +%Y%m%d%H%M` -`\$USER` }



Infrastructure as Code



The Sys Admin's Journey

- ssh

The Sys Admin's Journey

- ssh
- Store notes in ~ / server .txt

The Sys Admin's Journey

- ssh
- Store notes in ~ / server .txt
- Move notes to the wiki

The Sys Admin's Journey

- ssh
- Store notes in ~ / server .txt
- Move notes to the wiki
- Write some scripts (setup .sh, fixit .sh, etc.)

The Sys Admin's Journey

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

The Sys Admin's Journey

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

Benefits of Automation



Dimensions of Scale

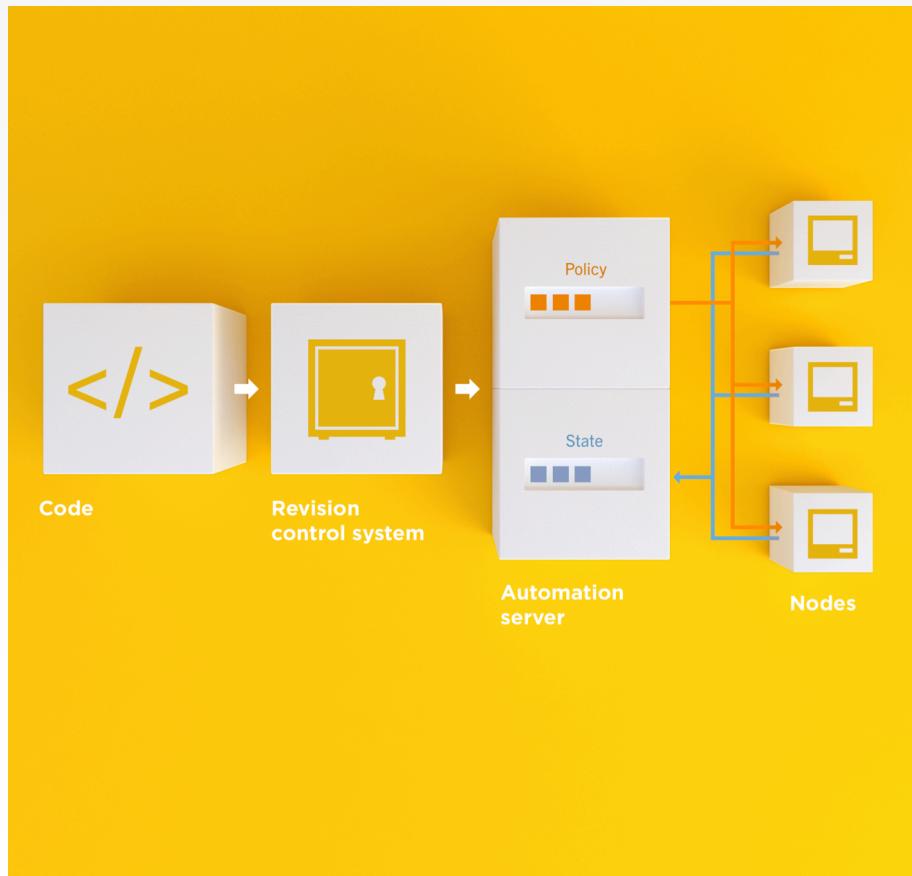


Automation Platform

- 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

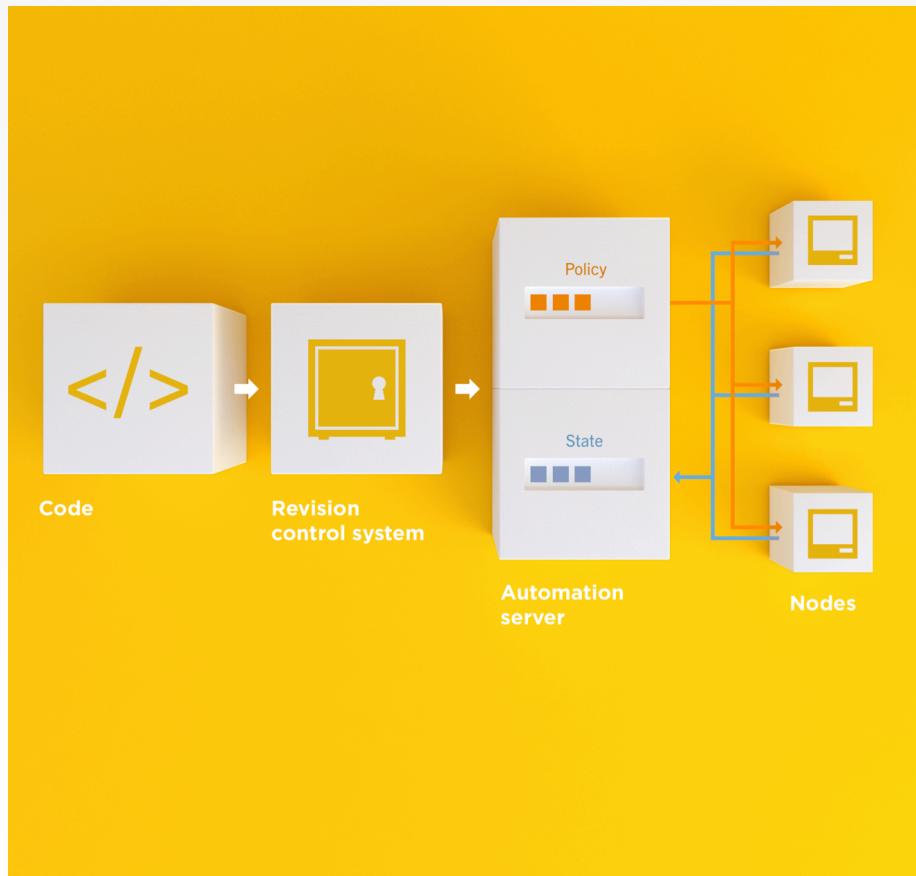


Infrastructure as Code



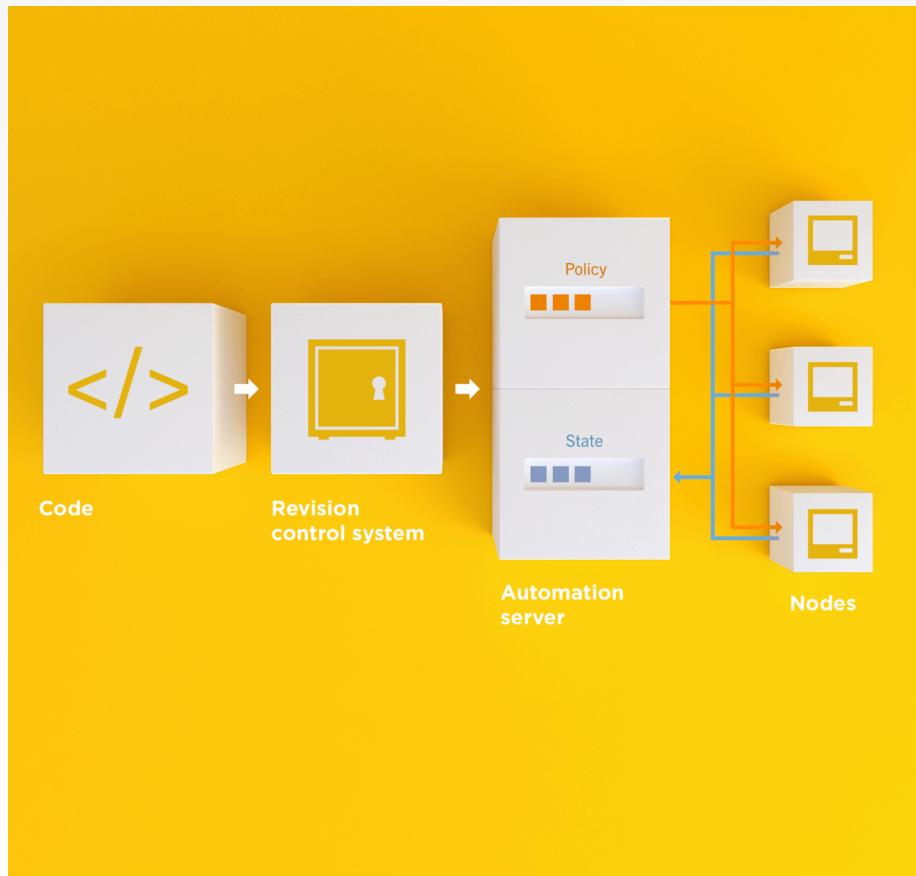
- Programmatically provision and configure components

Infrastructure as Code



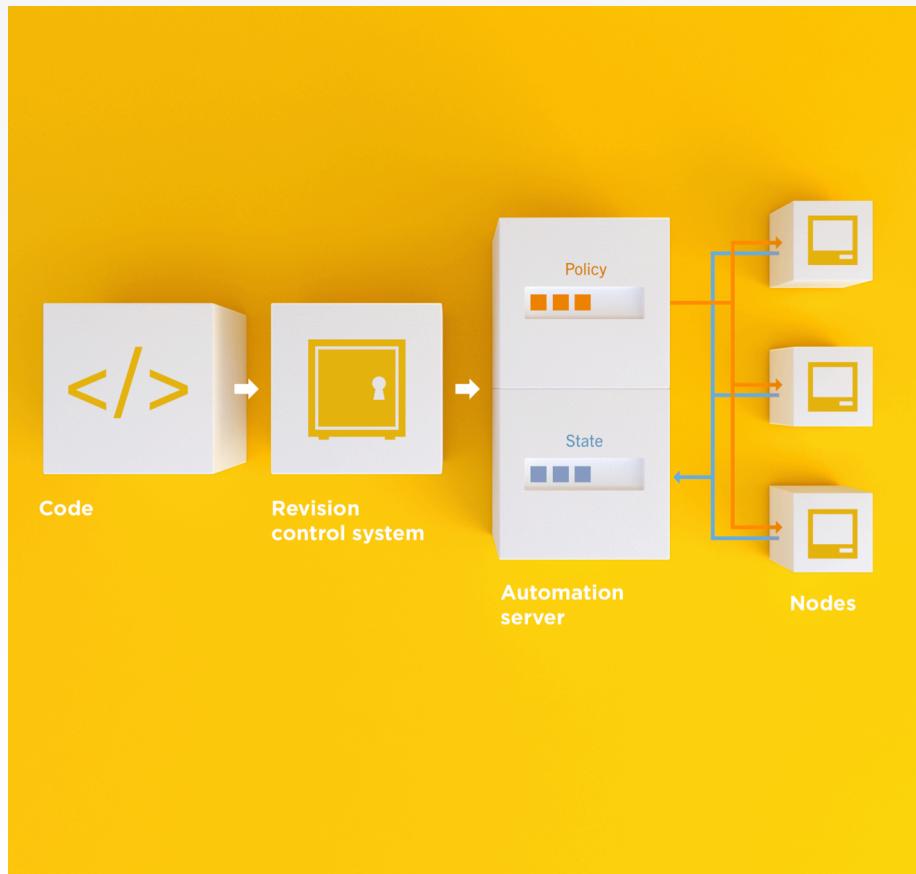
- Treat like any other code base

Infrastructure as Code



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

Infrastructure as Code

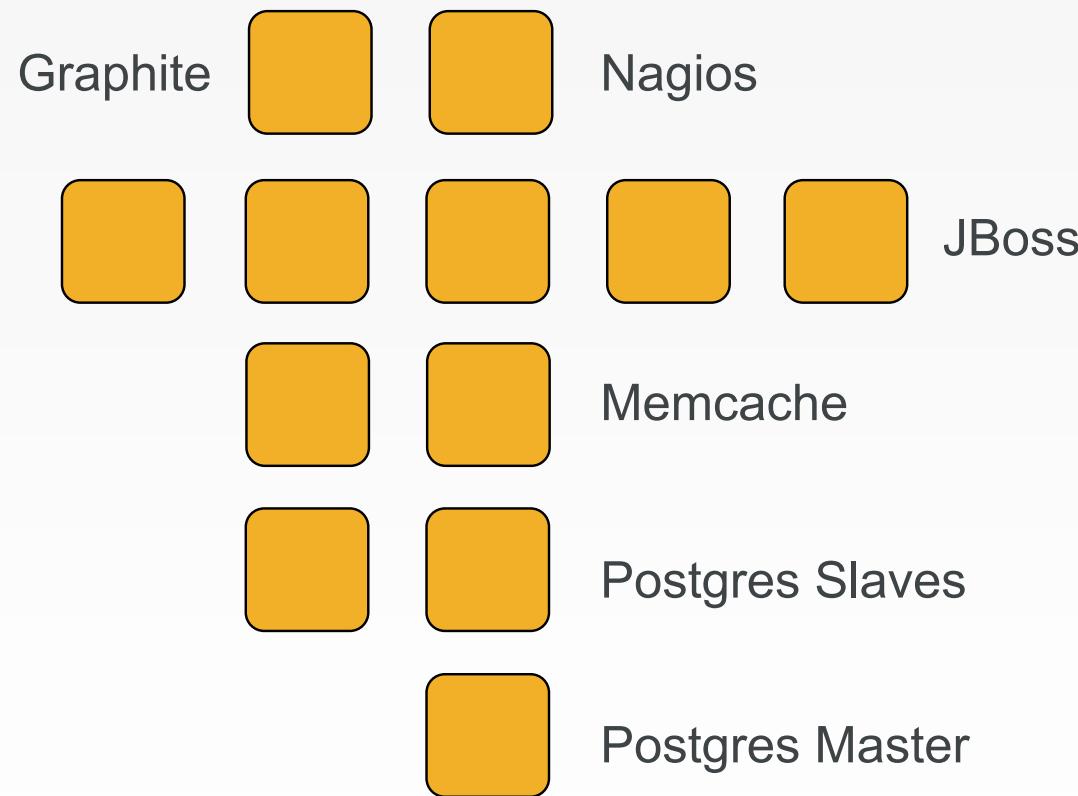


- Programmatically provision and configure components
- Treat like any other code base
- Reconstruct business from code repository, data backup, and compute resources

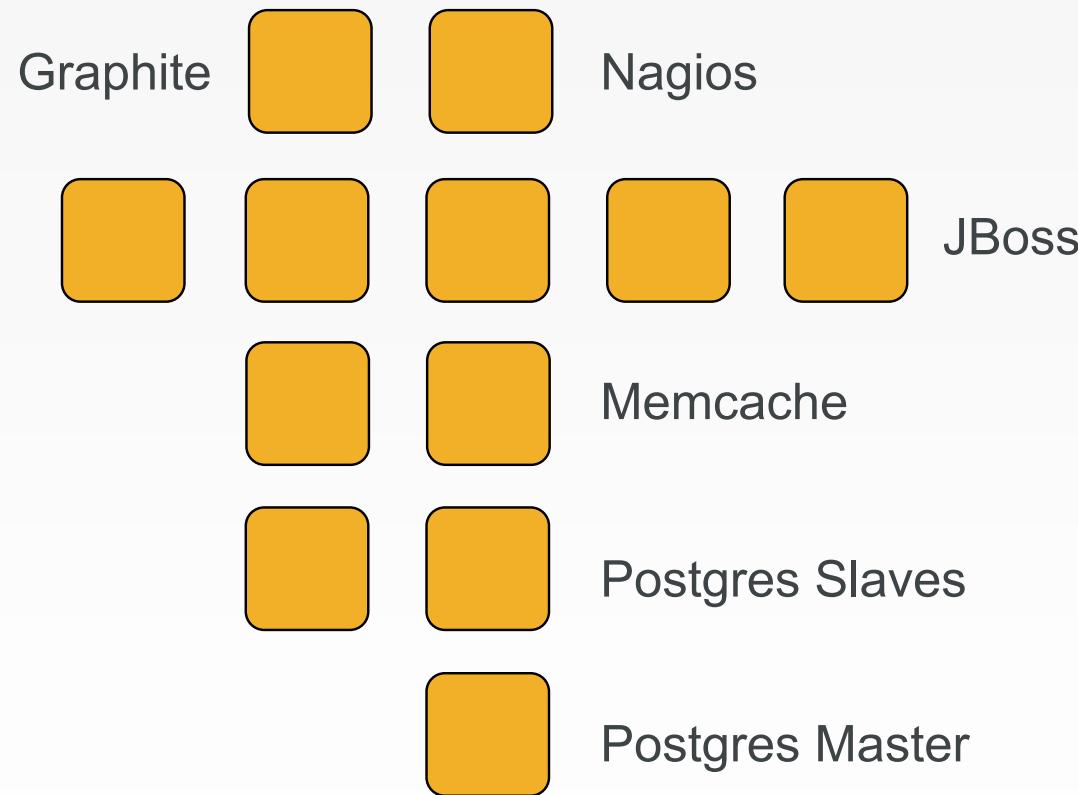
Policy-based

- 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

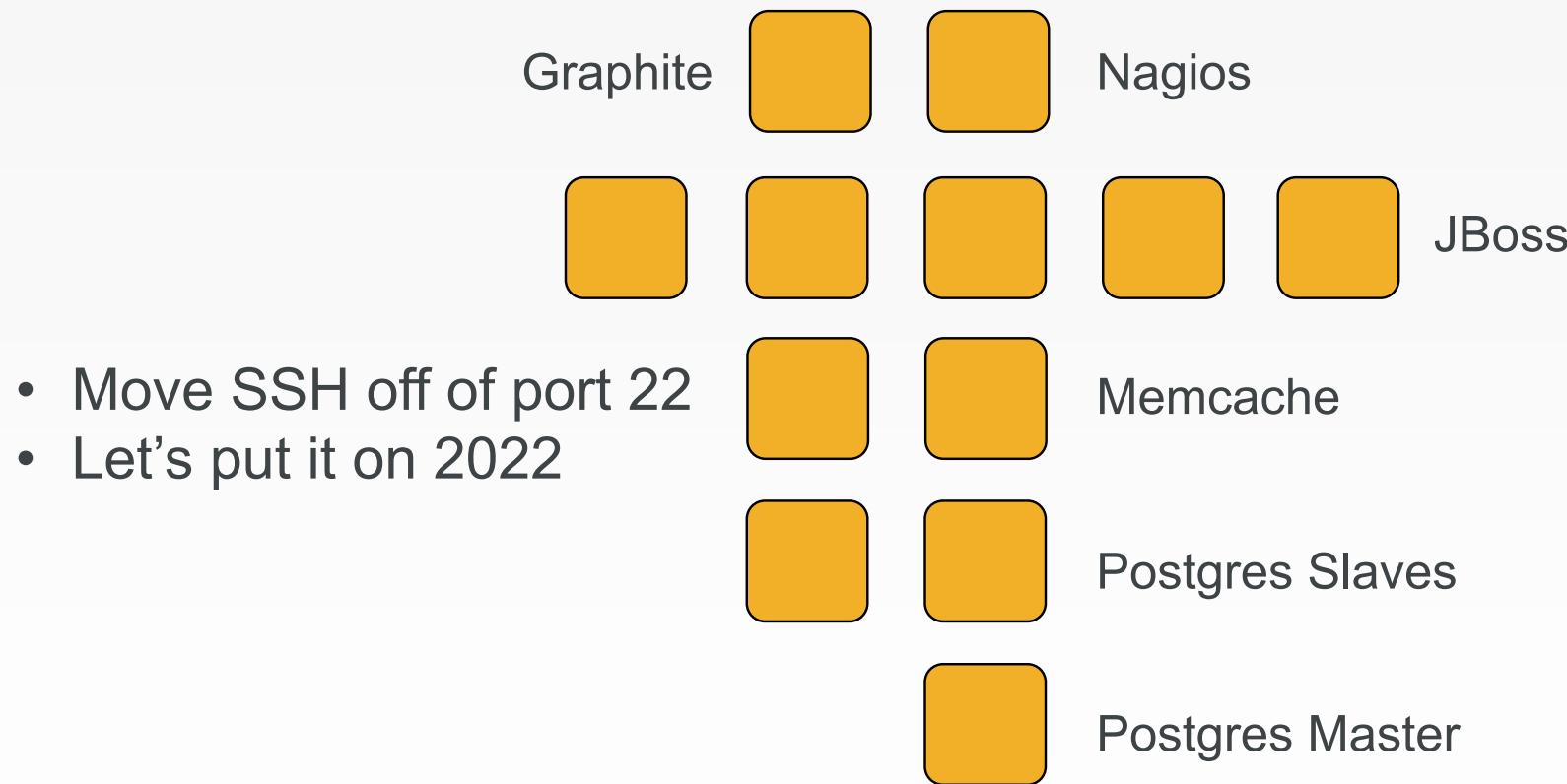
Sample Infrastructure



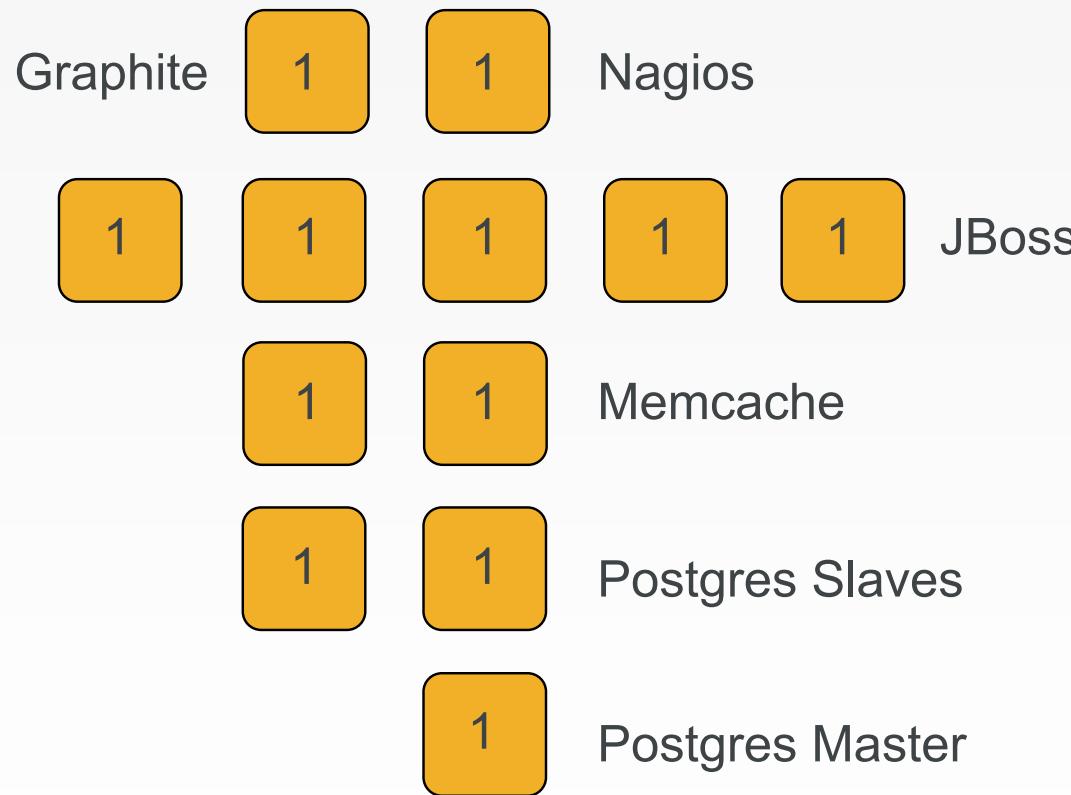
New Compliance Mandate!



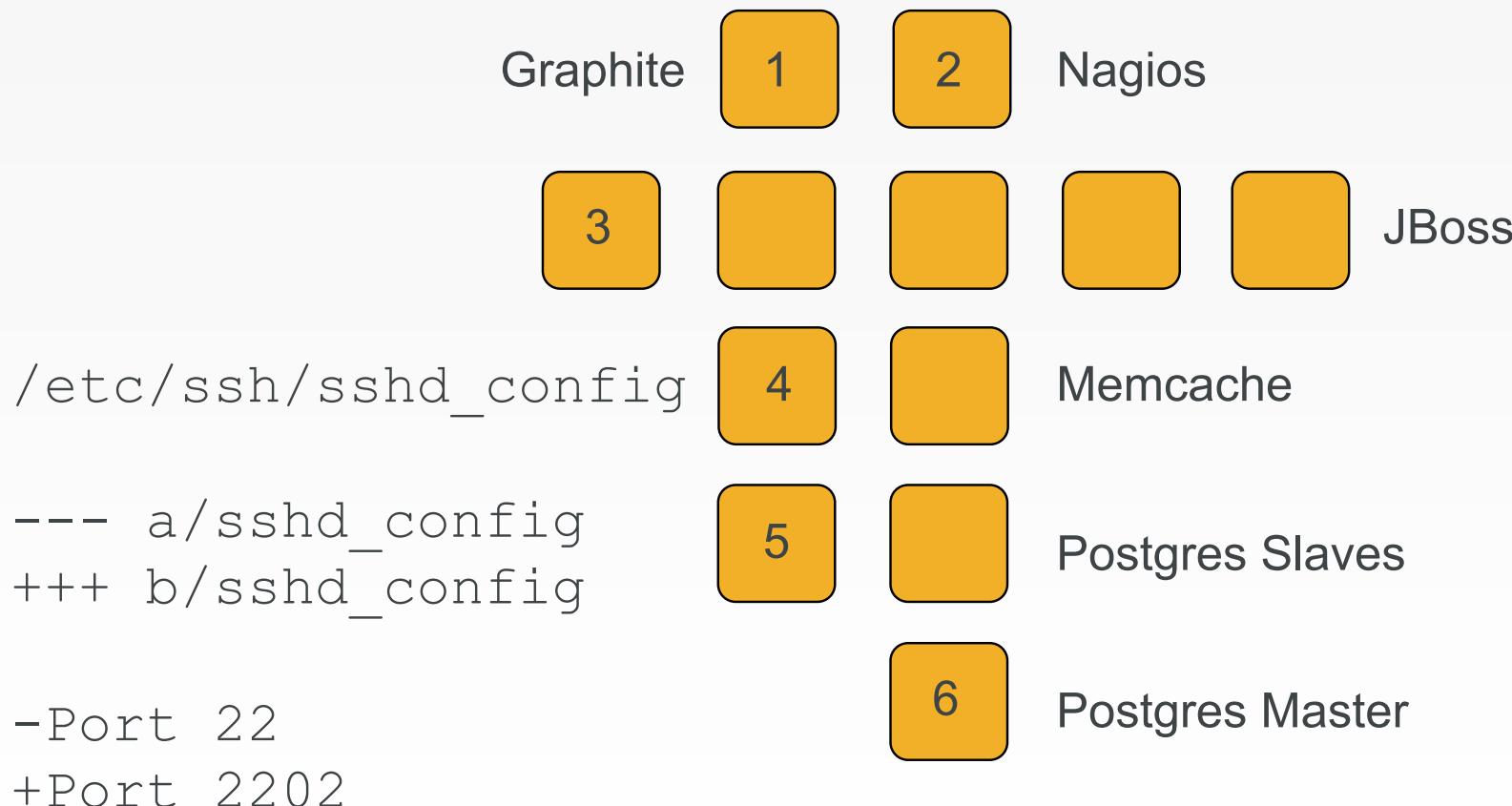
New Compliance Mandate!



6 Golden Images to Update

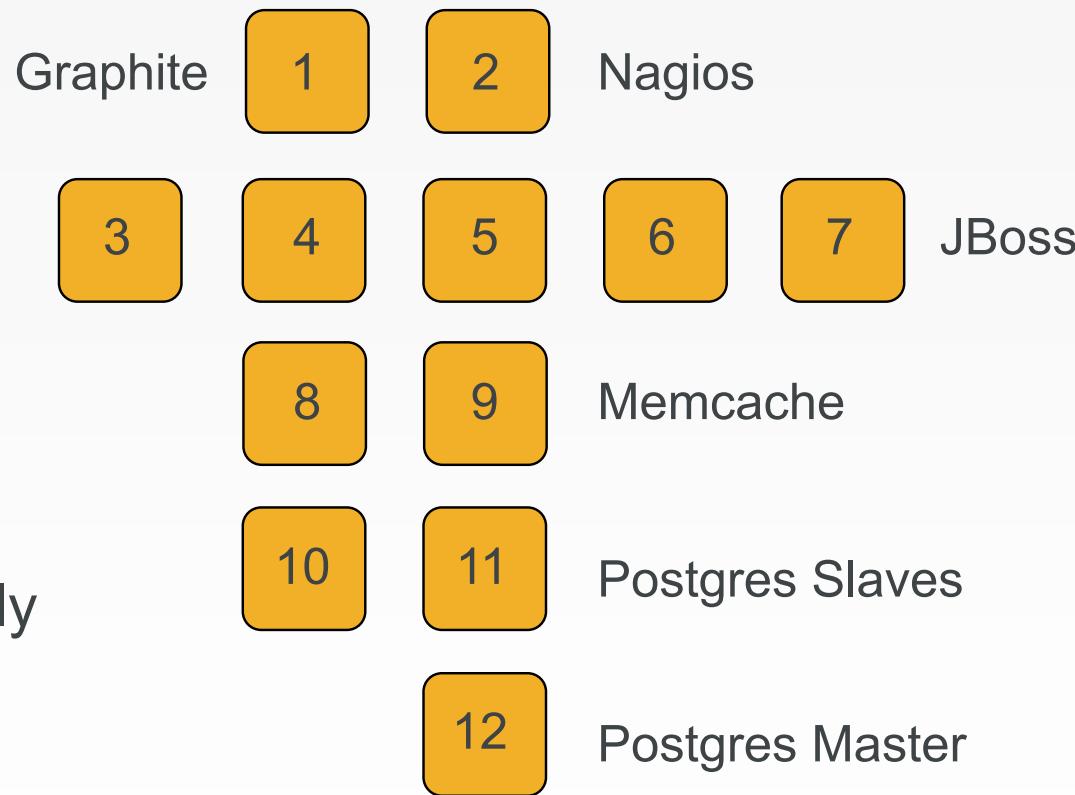


6 Golden Images to Update



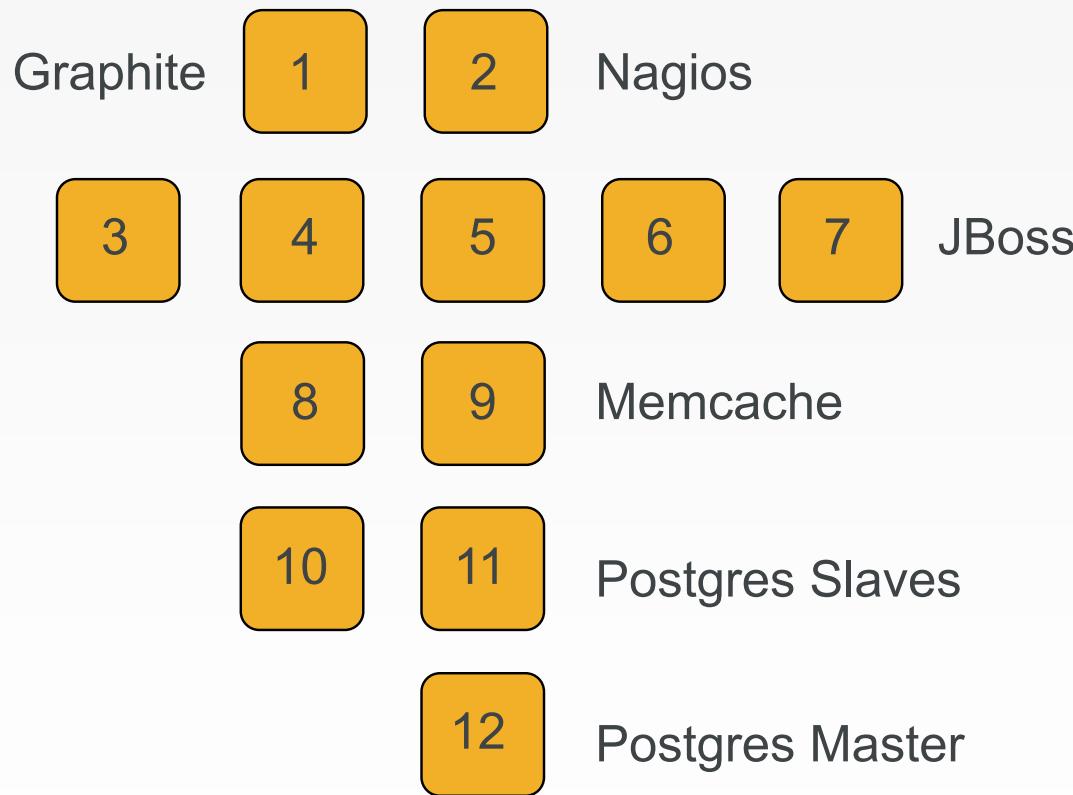
12 Instances to replace

- Launch
- Delete
- Repeat
- Typically manually

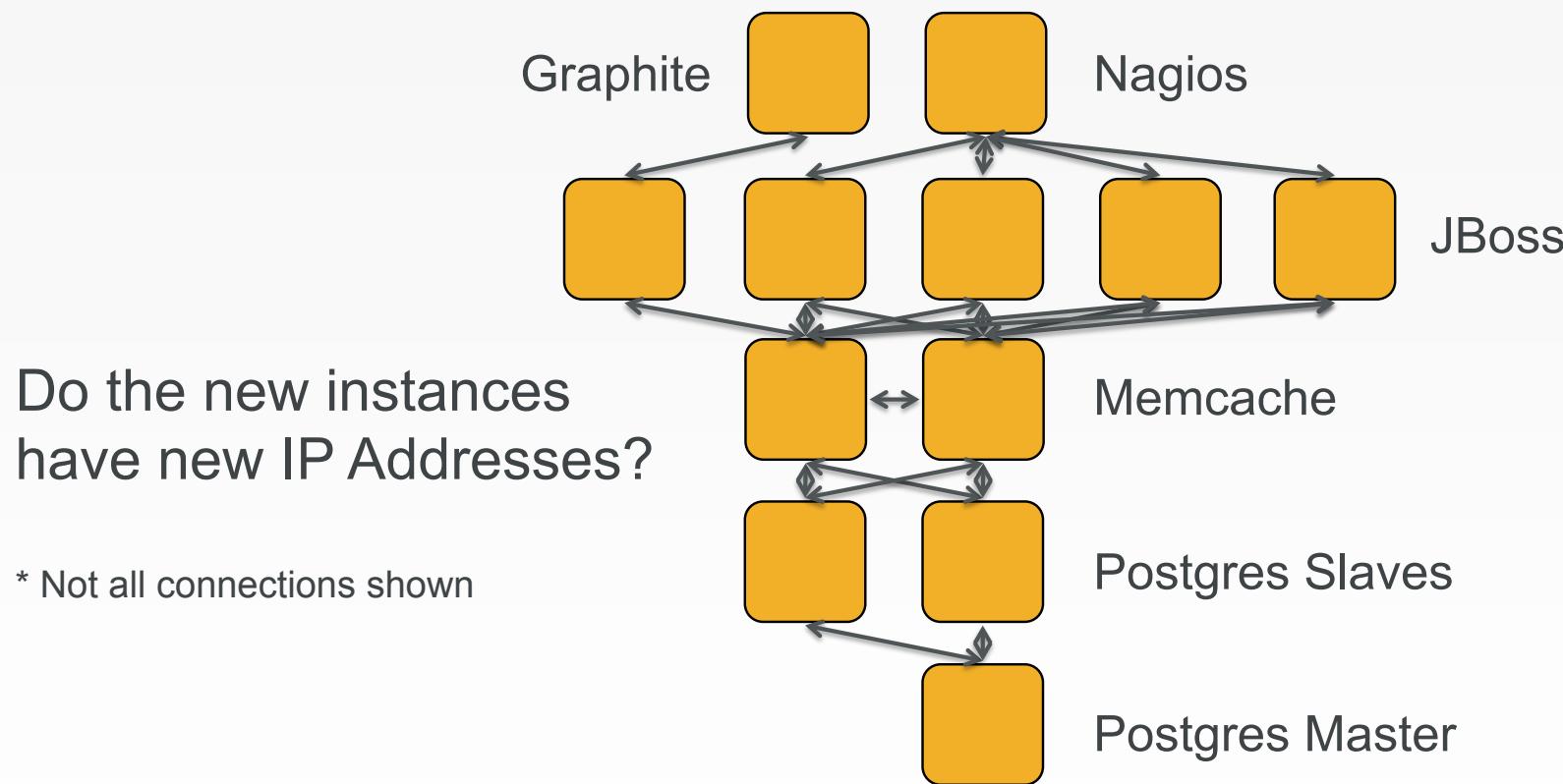


Done in maintenance window

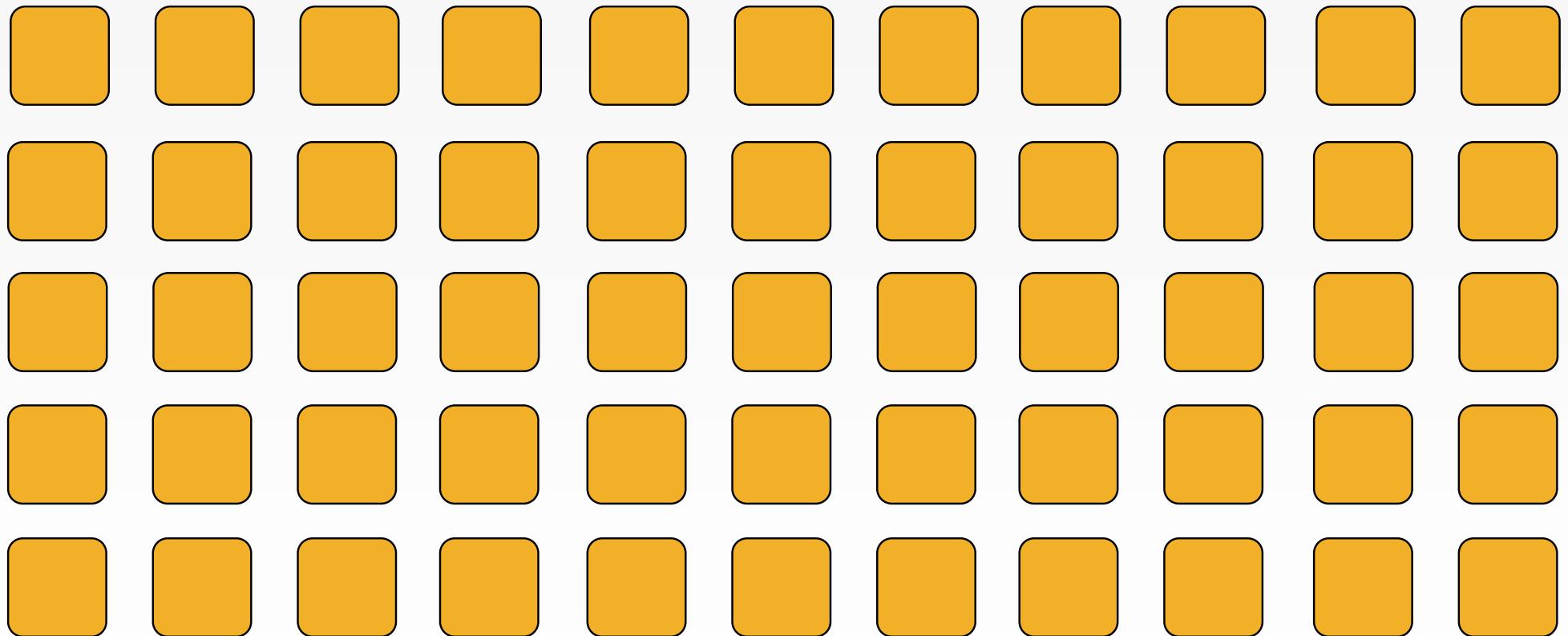
- High stakes
- Late hours
- Risky change



New configurations required?



Golden Images vs. Policy-based





Chef

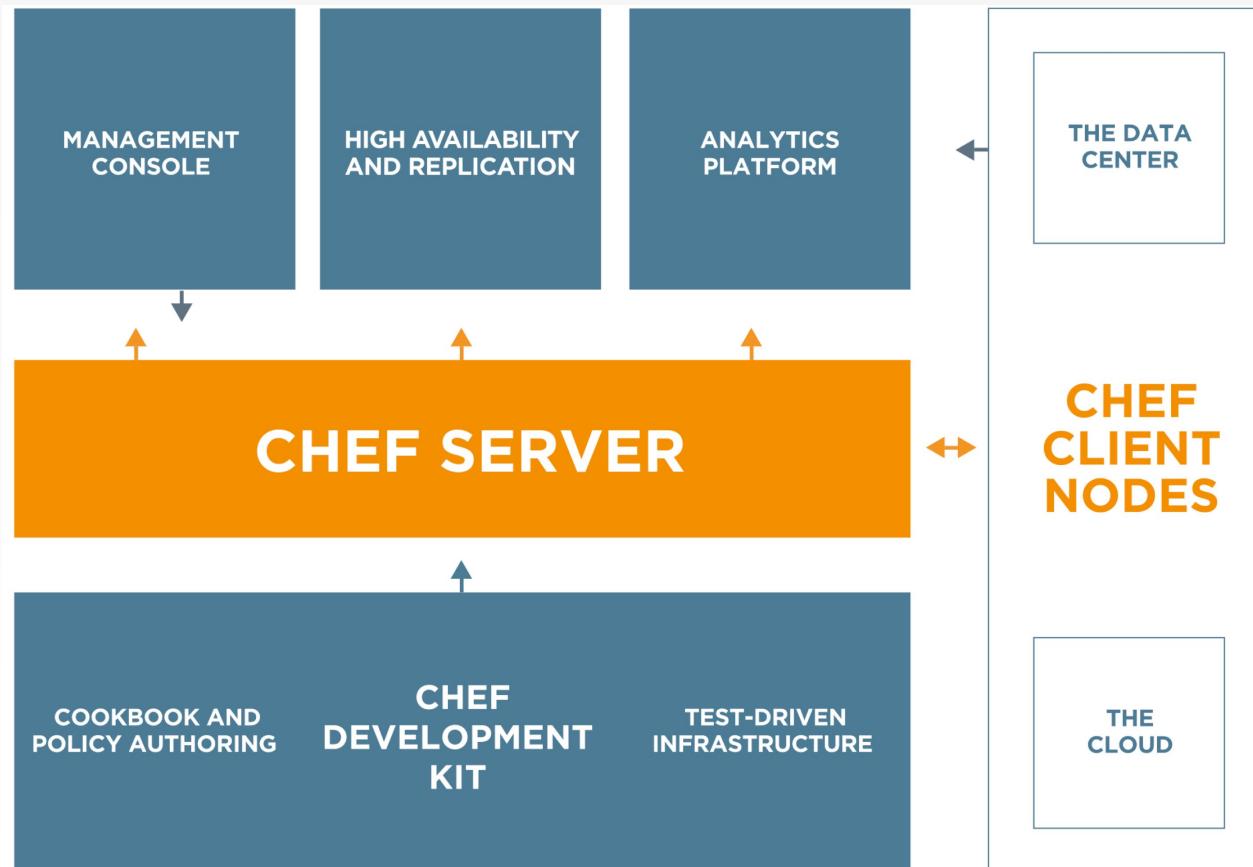
Fast, scalable, flexible IT automation



What is Chef

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

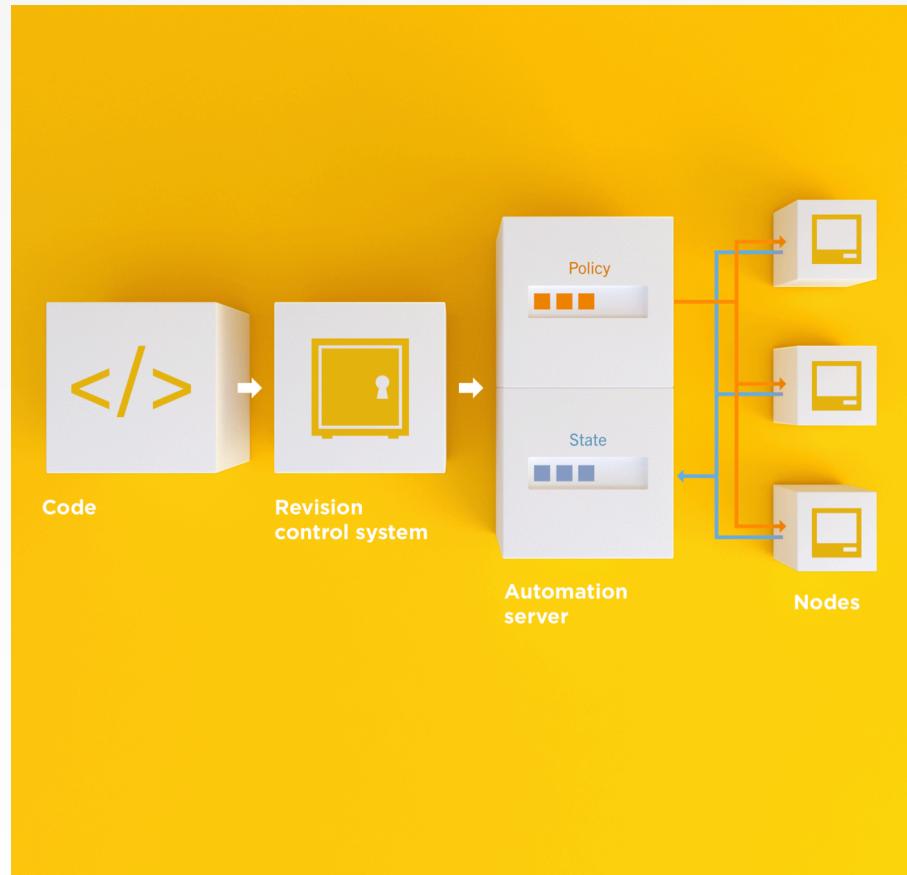
Chef



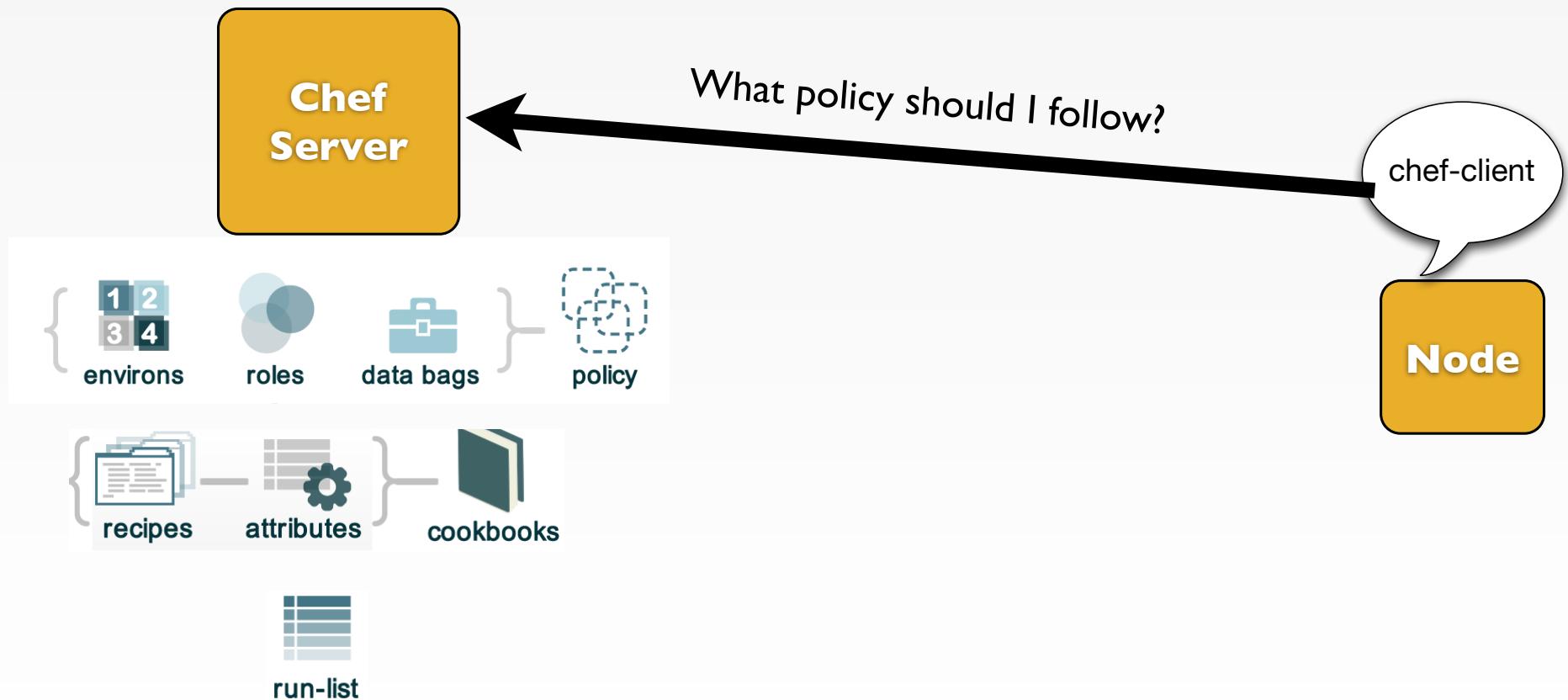
<https://www.getchef.com/chef/>



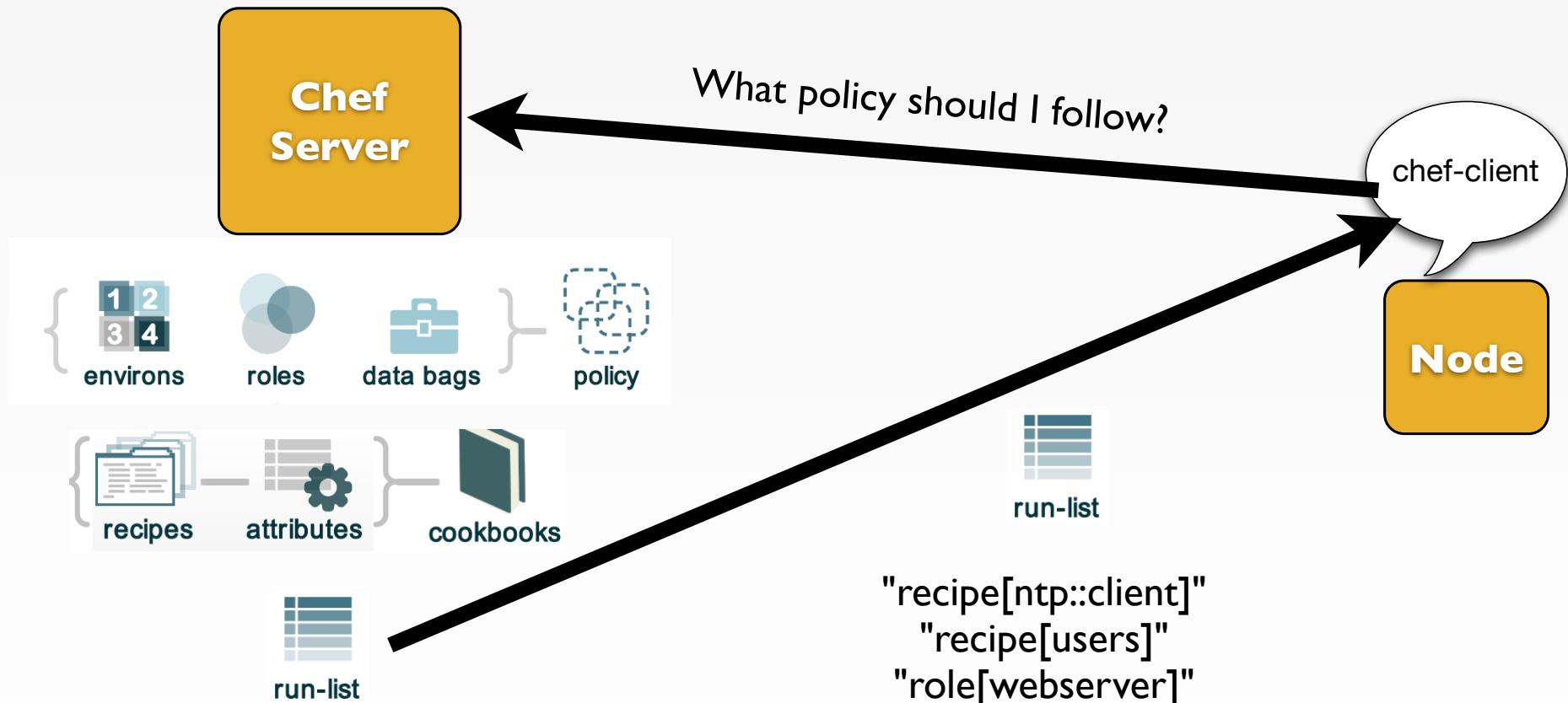
Chef Server – Policy & State



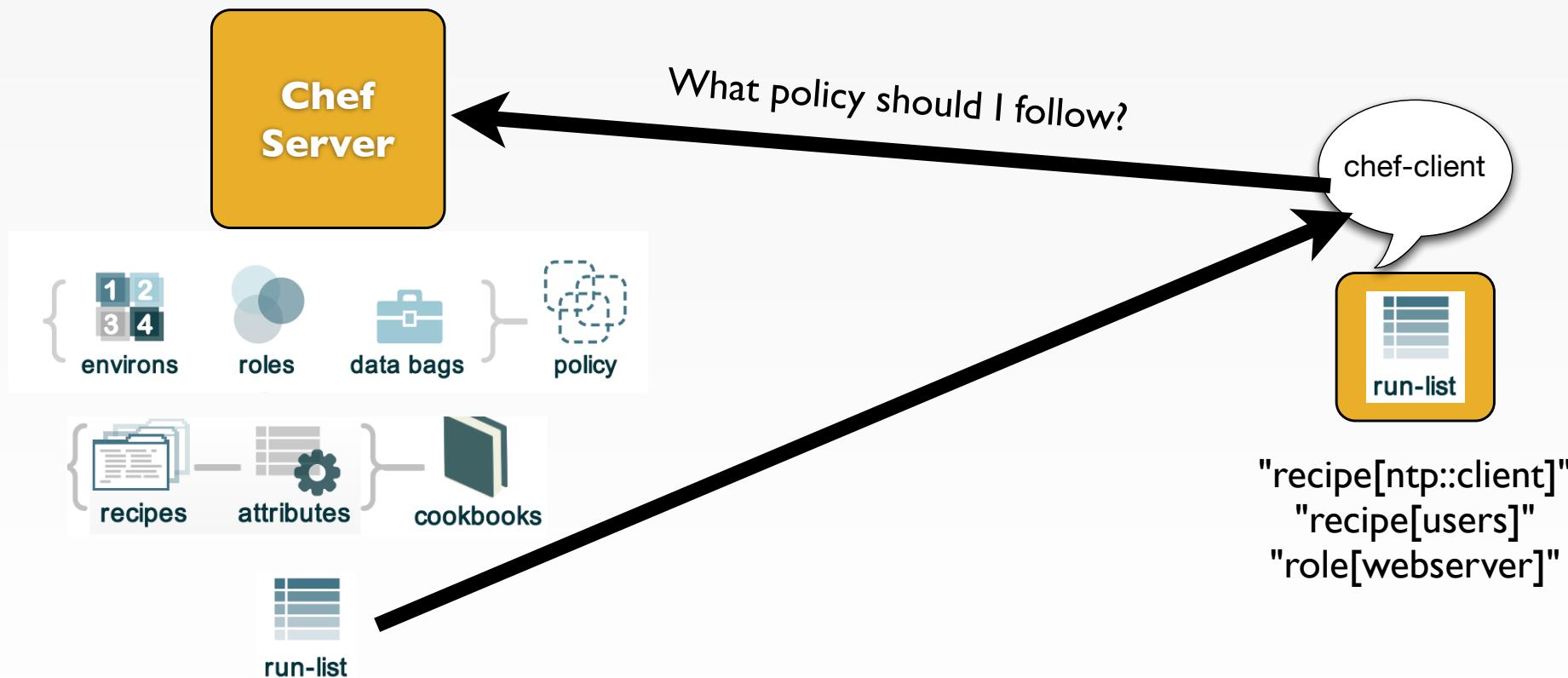
Desired Configuration



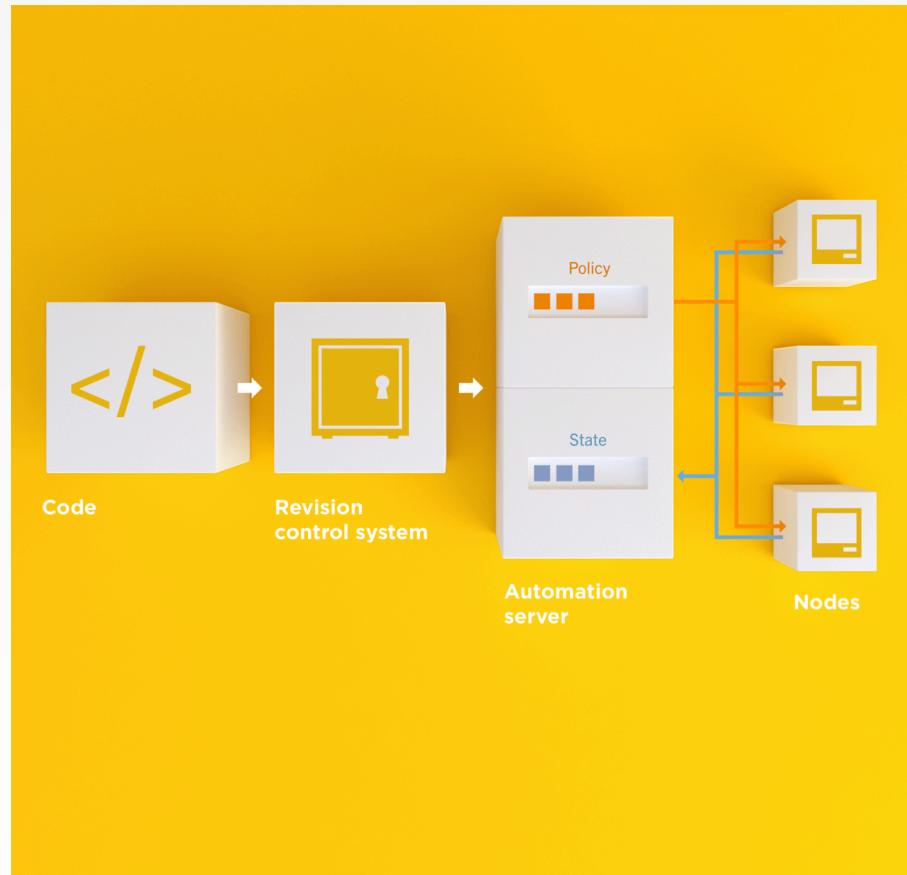
Desired Configuration



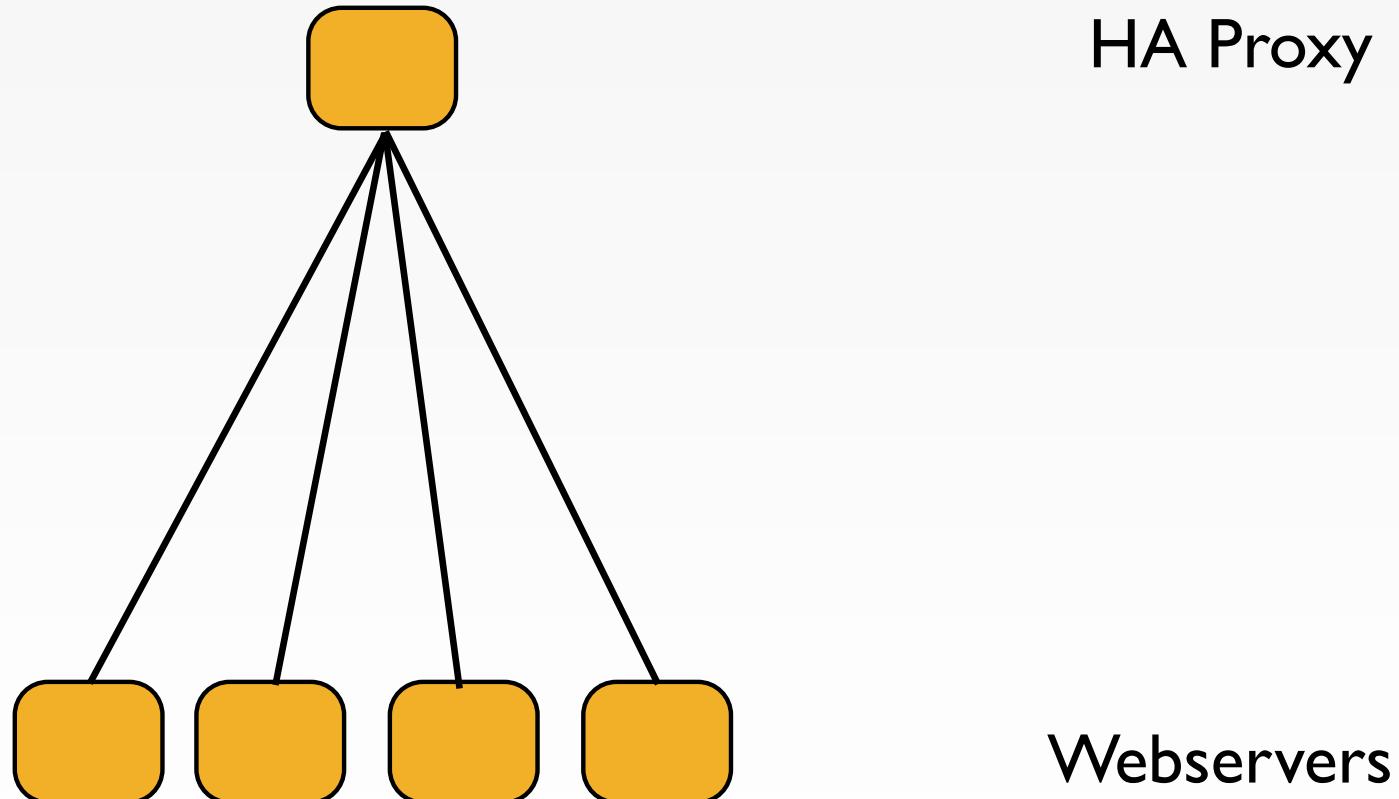
Desired Configuration



Chef Server – Policy & State



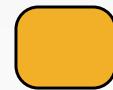
HA Proxy Configuration



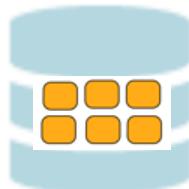
HA Proxy Configuration



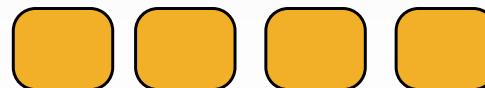
Chef
Server



HA Proxy



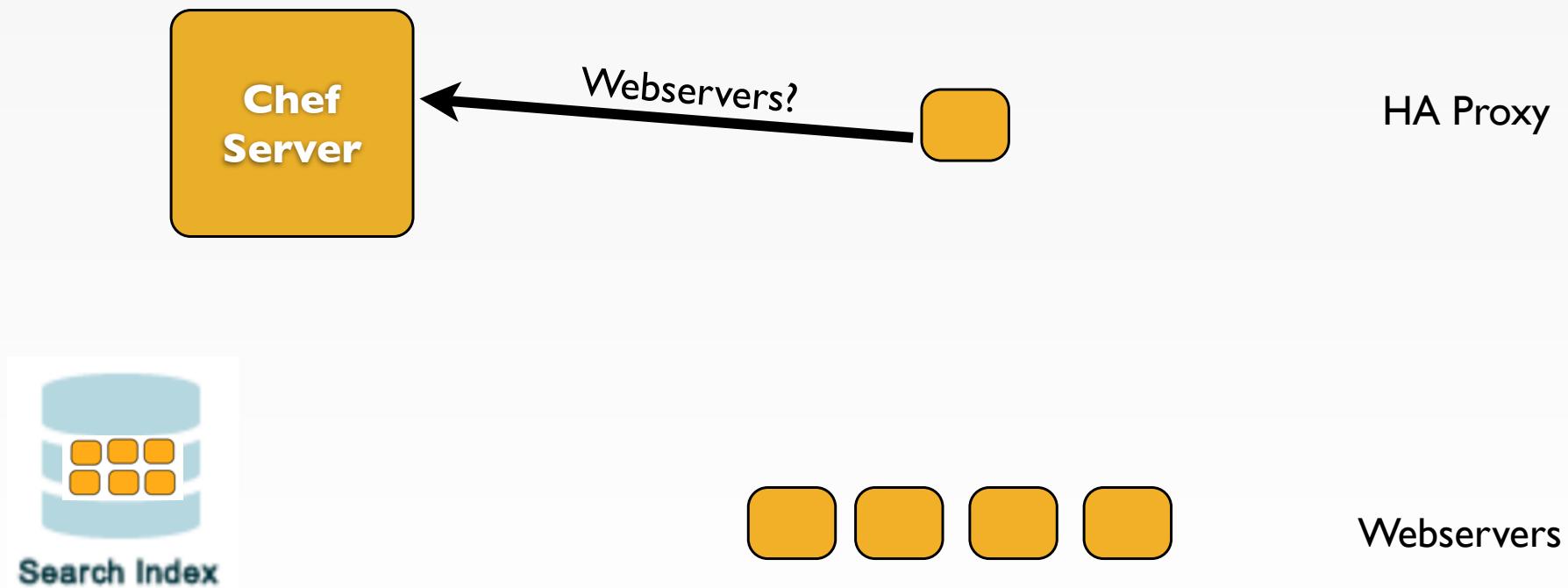
Search Index



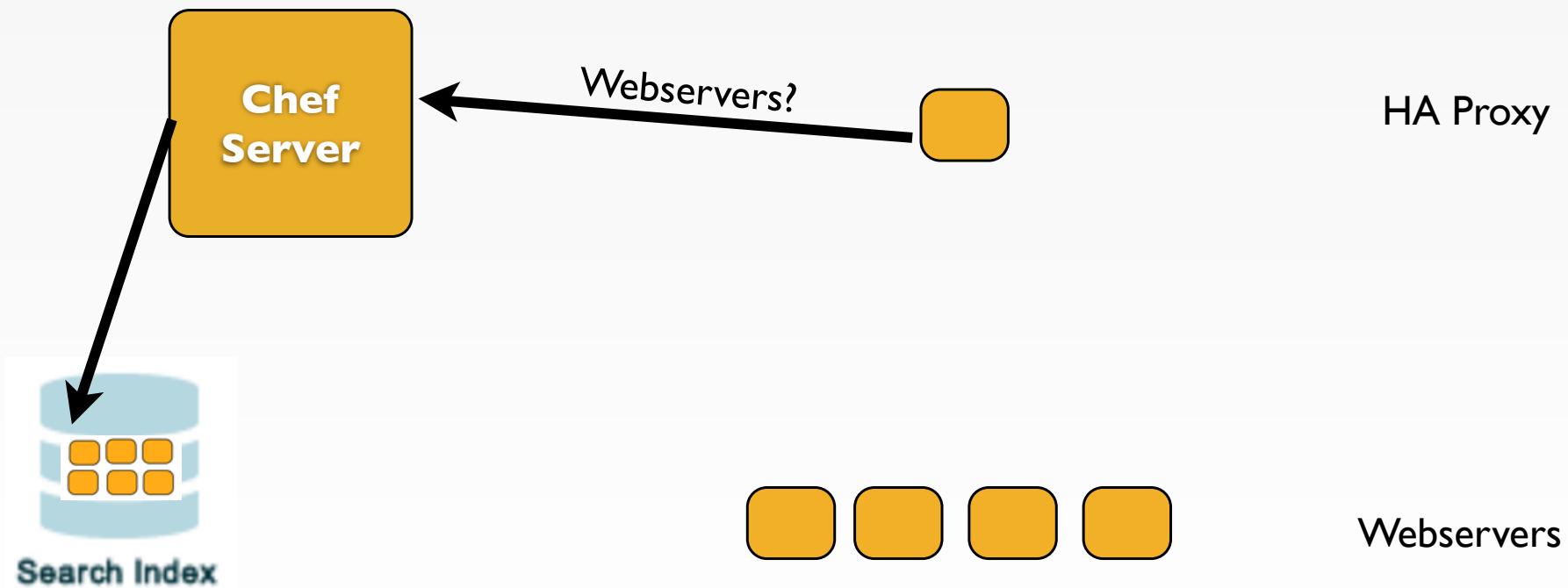
Webservers



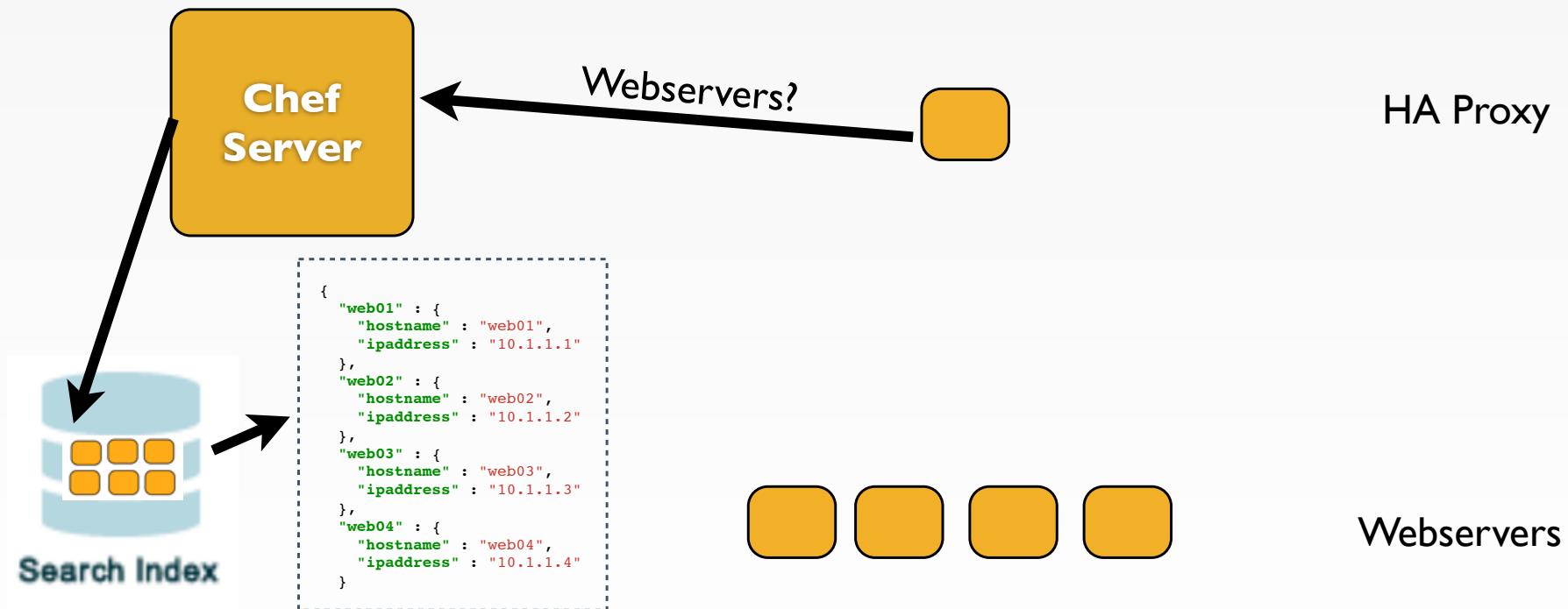
HA Proxy Configuration



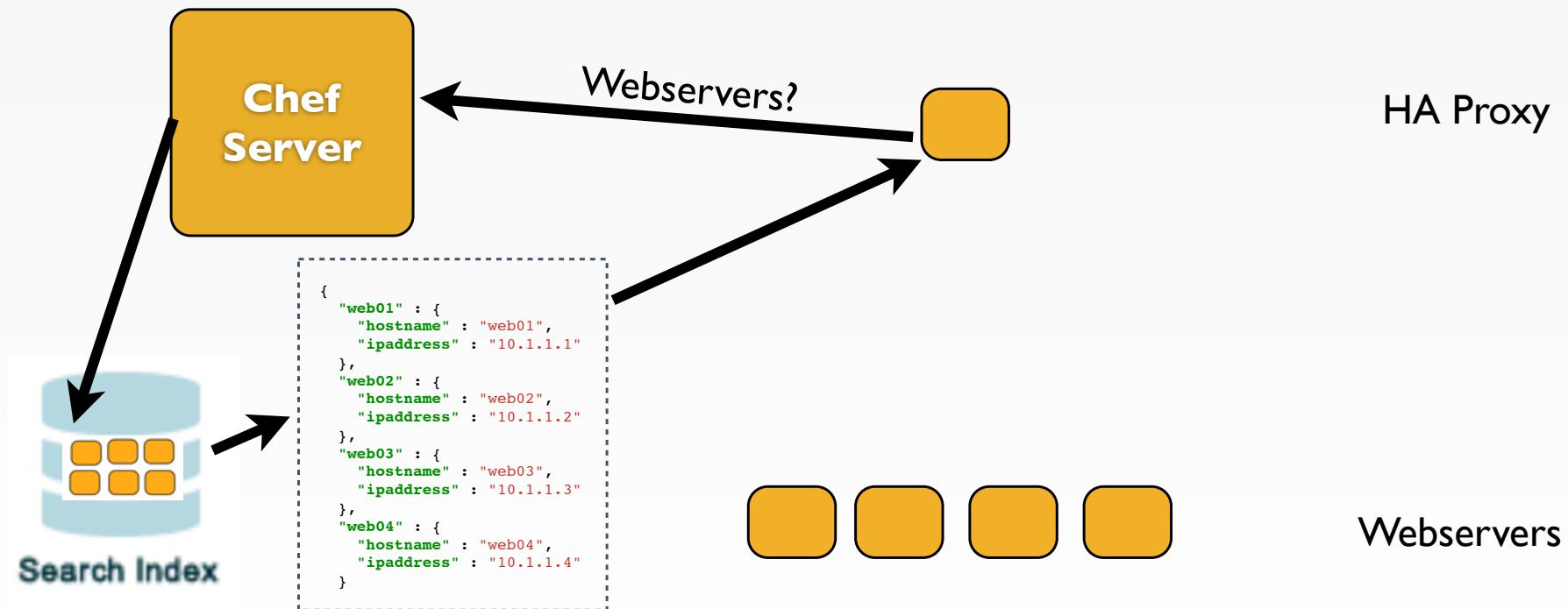
HA Proxy Configuration



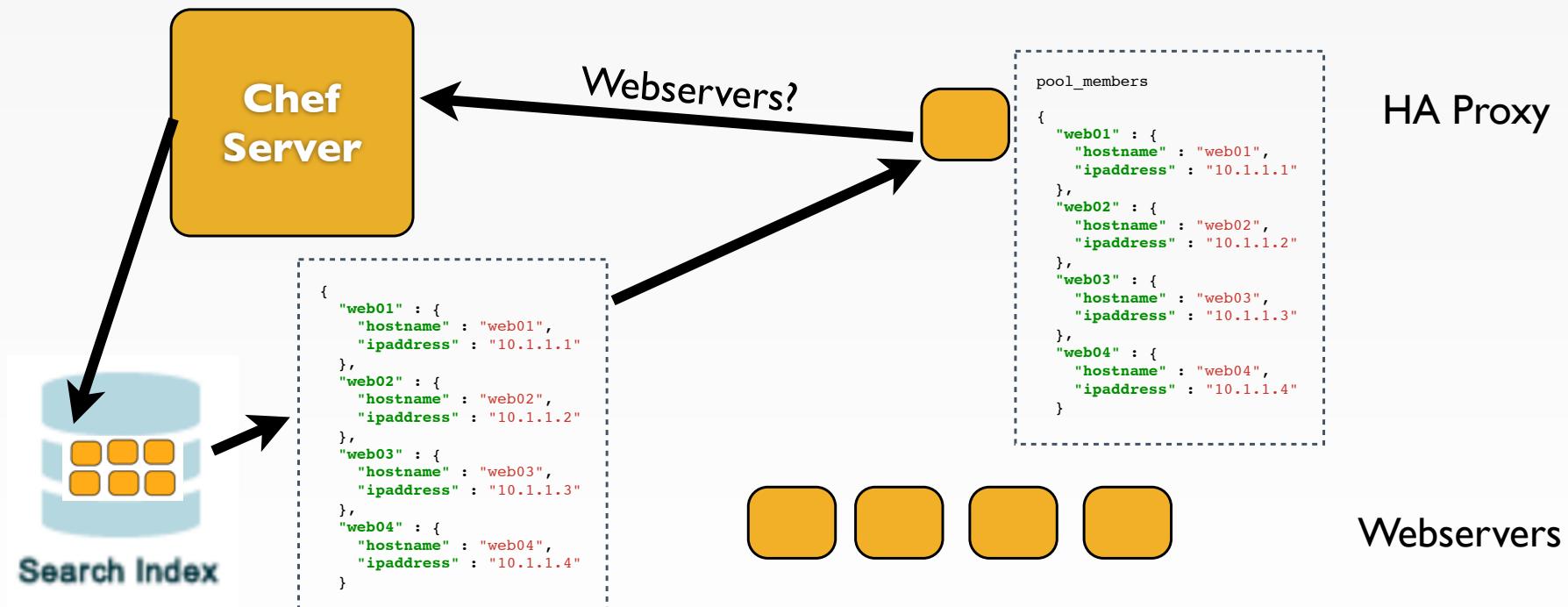
HA Proxy Configuration



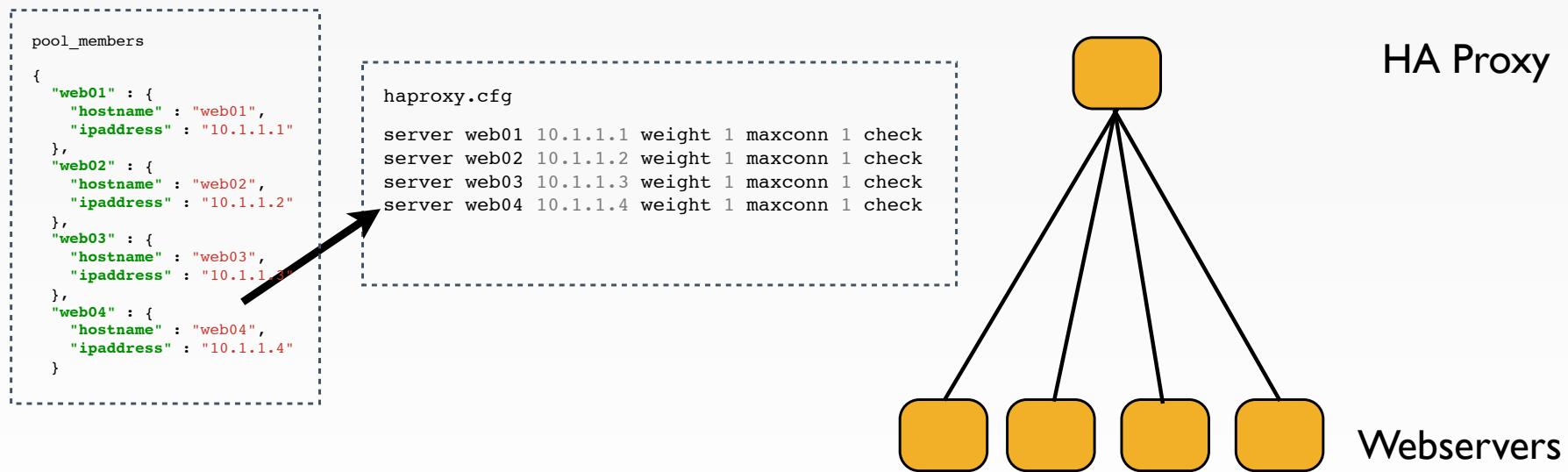
HA Proxy Configuration



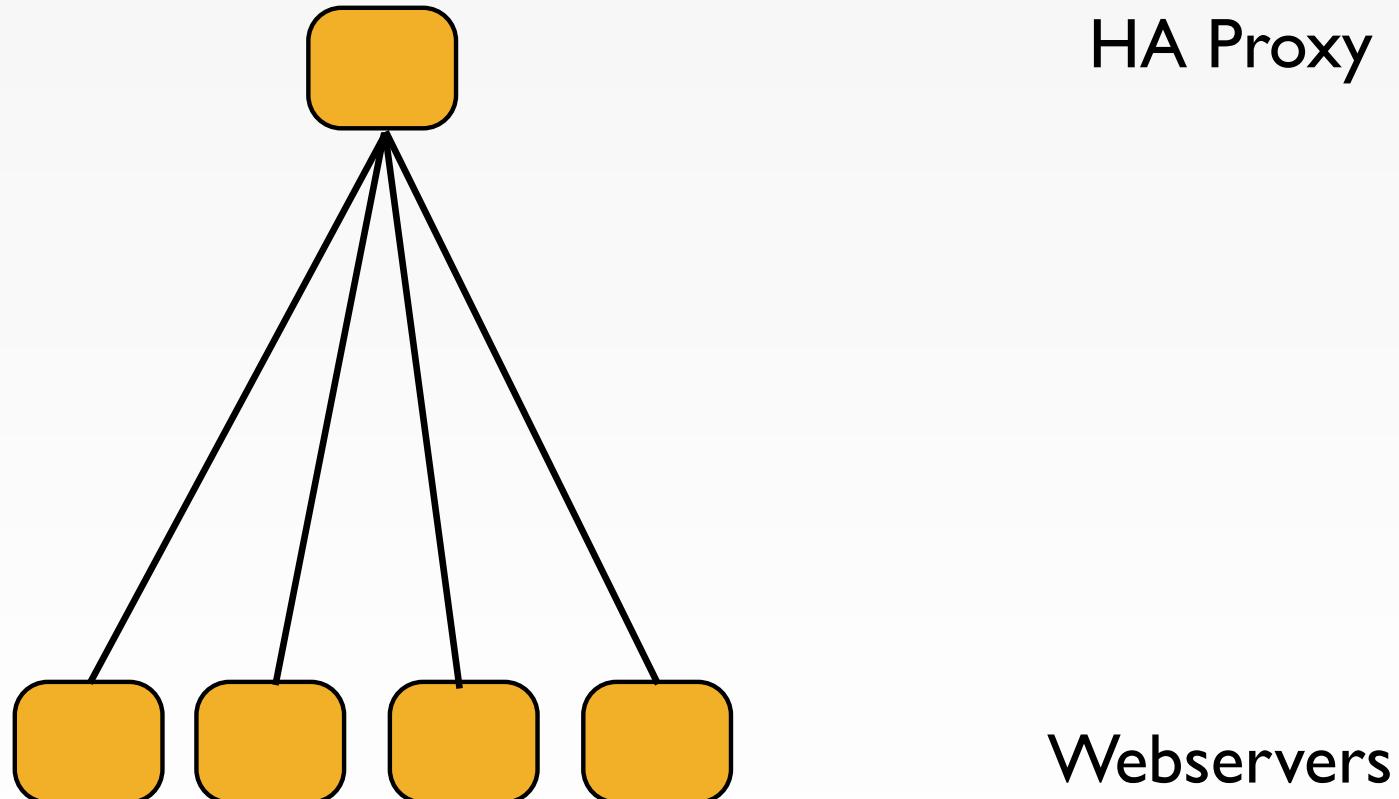
HA Proxy Configuration



HA Proxy Configuration



HA Proxy Configuration





Building your policy

Resources and Recipes



Resources

- Piece of the system and its desired state
 - Package that should be installed
 - Service that should be running
 - File that should be generated
 - Cron job that should be configured
 - User that should be managed
 - And more
- docs.getchef.com/chef/resources.html



Hello, Chef!



OPEN IN EDITOR: ~/hello_chef.rb

```
file "/tmp/hello_chef.txt" do
  content "Hello, Chef"
  mode "0777"
end
```

SAVE FILE!



Apply the policy

```
$ sudo chef-apply hello_chef.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
* file[/tmp/hello_chef.txt] action create
  - create new file /tmp/hello_chef.txt
  - update content in file /tmp/hello_chef.txt from none to 79c290
    --- /tmp/hello_chef.txt      2014-10-22 19:59:04.000000000 -0400
    +++ /tmp/.hello_chef.txt20141022-23075-19aelx1      2014-10-22
19:59:04.000000000 -0400
@@ -1 +1,2 @@
+Hello, Chef
- change mode from '' to '0777'
```



Resources

- Describe the desired state
- Do not need to tell Chef how to get there
- What happens when you re-apply the policy?

Apply the policy

```
$ sudo chef-apply hello_chef.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * file[/tmp/hello_chef.txt] action create (up to date)
```



Resources

- A piece of the system
- Its desired state

```
file "/tmp/hello_chef.txt" do
  content "Hello, Chef"
  mode "0777"
end
```



Change the state of the system

```
$ echo "Hello, #ato2014" > /tmp/hello_chef.txt
```

Apply the policy

```
$ sudo chef-apply hello_chef.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * file[/tmp/hello_chef.txt] action create
    - update content in file /tmp/hello_chef.txt from e453df to 79c290
      --- /tmp/hello_chef.txt      2014-10-22 20:00:20.000000000 -0400
      +++ /tmp/.hello_chef.txt    2014-10-22 23:34:17.570000000 -0400
      20:00:50.000000000 -0400
      @@ -1,2 +1,2 @@
      -"Hello, #ato2014"
      +Hello, Chef
```

Resources – Test and Repair

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

Built-in Resources

- package
- template
- service
- cron
- directory
- mount
- user
- group
- registry_key
- remote_directory
- route
- and many more...

docs.getchef.com/chef/resources.html



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.

Sample Recipe

```
package "httpd"

template "/etc/httpd/conf/httpd.conf" do
  source "httpd.conf.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[httpd]"
end

service "httpd" do
  action [:start, :enable]
end

file "/var/www/html/index.html" do
  content "Hello, ATO!"
end
```





Test-driven Infrastructure

Change policy with confidence



New policy mandate

- Apache should listen on port 81, not the default port!
- Verify policy changes BEFORE applying the changes to production

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 Kitchen Lifecycle

- kitchen create
- kitchen list
- kitchen converge
- kitchen login
- kitchen destroy



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?

Manually Inspect with kitchen login

```
$ kitchen login
```

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



Manually Inspect with kitchen login

```
$ kitchen login
```

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



Manually Inspect with kitchen login

```
$ kitchen login
```

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

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



Manually Inspect with kitchen login

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

```
Hello, ATO!
```



Serverspec

- Write RSpec 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/>



Serverspec Test



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

```
require 'serverspec'
set :backend, :exec

describe "apache" do
  it "has httpd package installed" do
    expect(package('httpd')).to be_installed
  end

  it "is listening on port 80" do
    expect(port(80)).to be_listening
  end

  it "displays our home page" do
    expect(command("curl http://localhost").stdout).to match /ATO/
  end
end
```

SAVE FILE!



Verify the kitchen

```
$ kitchen verify
```

```
----> Setting up Busser
      Creating BUSSER_ROOT in /tmp/busser
      Creating busser binstub
      Plugin serverspec installed (version 0.2.6)
----> Running postinstall for serverspec plugin
      Finished setting up <default-centos-64> (0m32.59s).
----> Verifying <default-centos-64>...
      Suite path directory /tmp/busser/suites does not exist, skipping.
      Uploading /tmp/busser/suites/serverspec/default_spec.rb (mode=0664)
----> Running serverspec test suite
      /opt/chef/embedded/bin/ruby -I/tmp/busser/suites/serverspec -S /opt/chef/embedded/bin/rspec /tmp/busser/suites/serverspec/default_spec.rb
--color --format documentation

      apache
      is installed

      Finished in 0.29547 seconds
      1 example, 0 failures
      Finished verifying <default-centos-64> (0m4.44s).
----> Kitchen is finished. (1m25.74s)
```

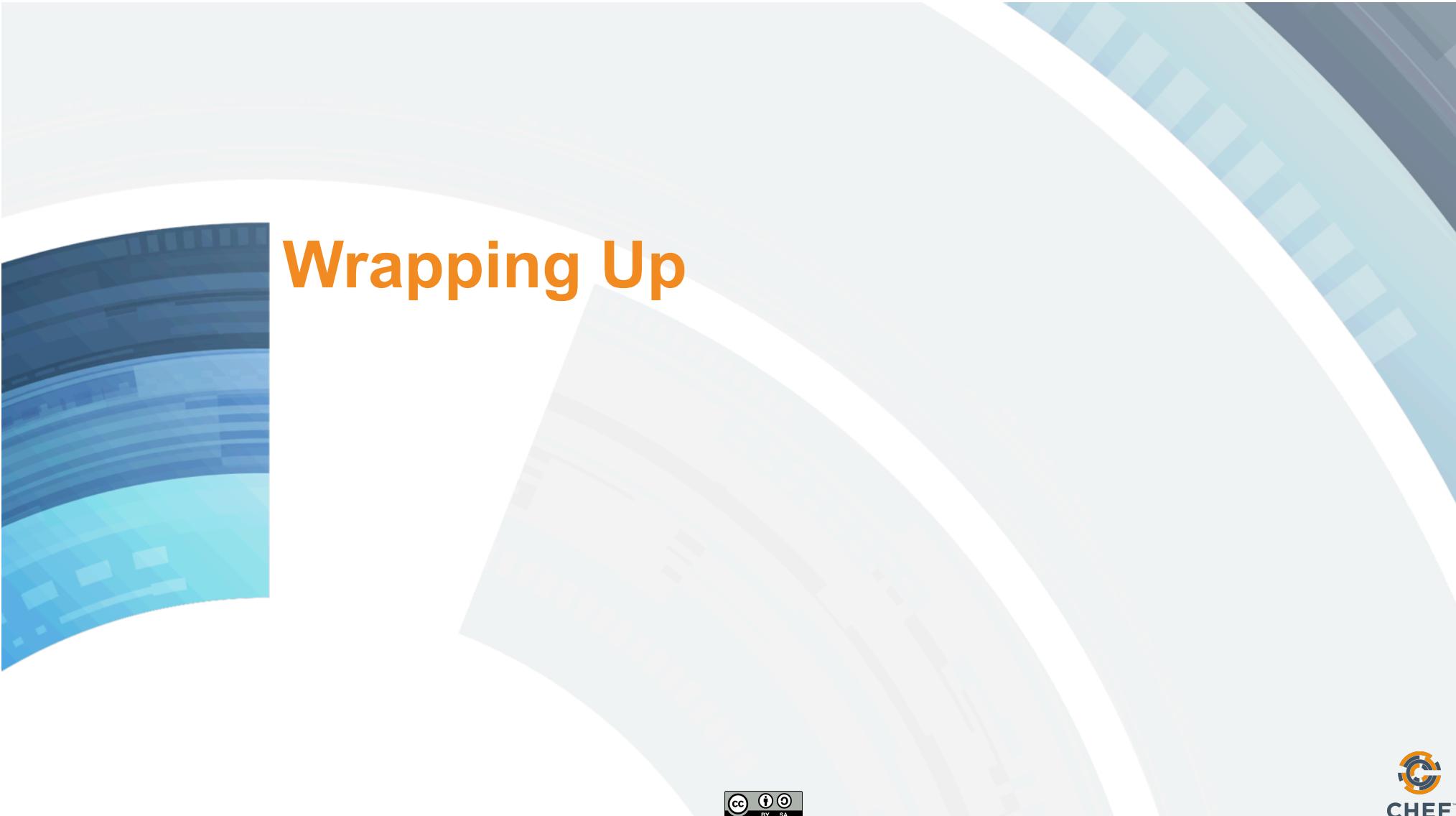


Questions to ask when testing

- ✓ 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?

Now for our new mandate

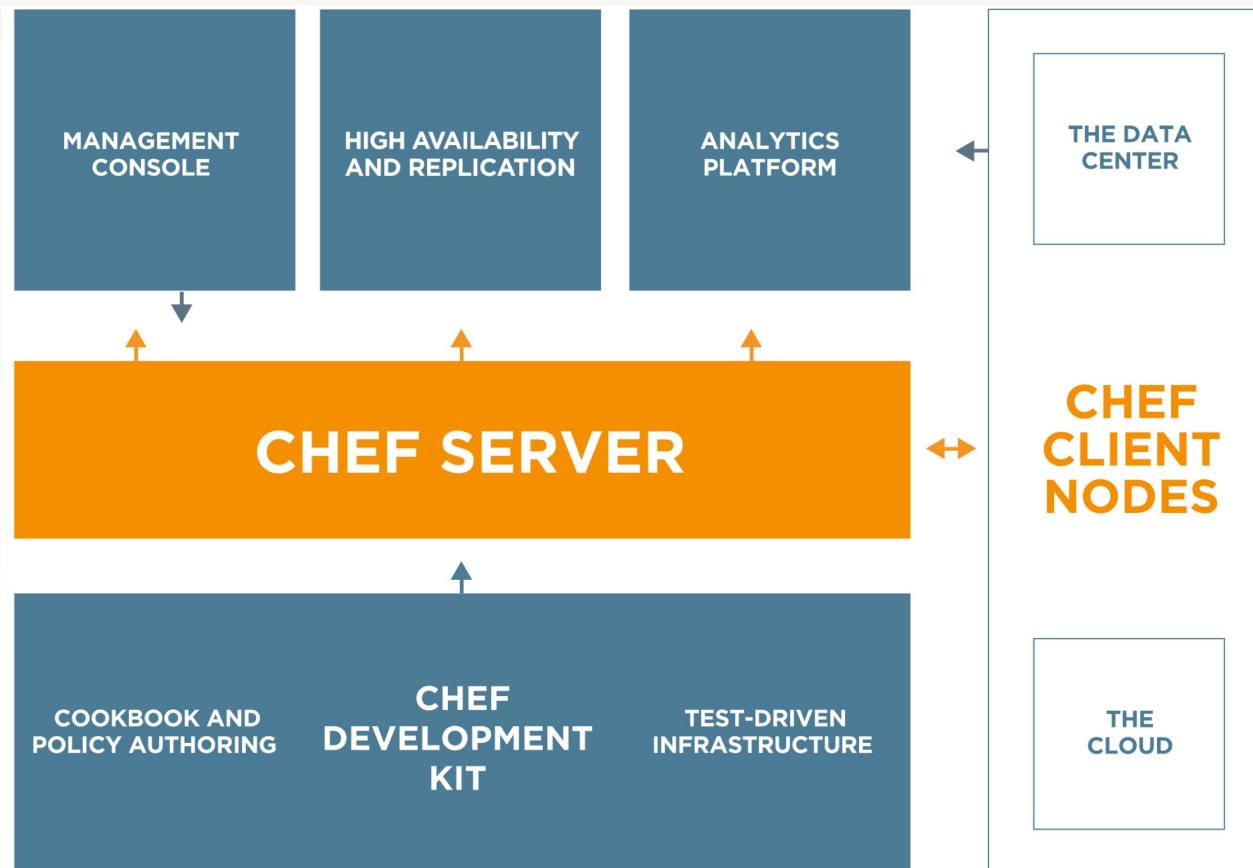
- Update the tests
- Watch them fail
- Update the policy
- See tests pass
- Roll-out changes to production



Wrapping Up



We've only scratched the surface



<https://www.getchef.com/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 me anything!
- @nathenharvey
- nharvey@getchef.com
- Thank you!

