Really large scale systems configuration

Config Management @ Facebook
Phil Dibowitz


METALLICA

Who am I?

Configuration Management Experience

- Co-authored *Spine*
- Authored *Provision*

Scale Experience

- Ticketmaster, Google, Facebook

Passionate about scaling configuration management

Scaling

http://coolinterestingstuff.com/amazing-space-images/

Scaling Configuration Management

How many homogeneous systems can you maintain?

How many heterogeneous systems can you maintain?

How many people are needed?

Can you safely delegate delta configuration?

The Goal

http://www.prathidhwani.org/sports/goal-2012-the-beautiful-game-is-back/

The Goal

- 4 people
- Tens of thousands of heterogeneous systems
- Service owners own/adjust relevant settings

What did we need?

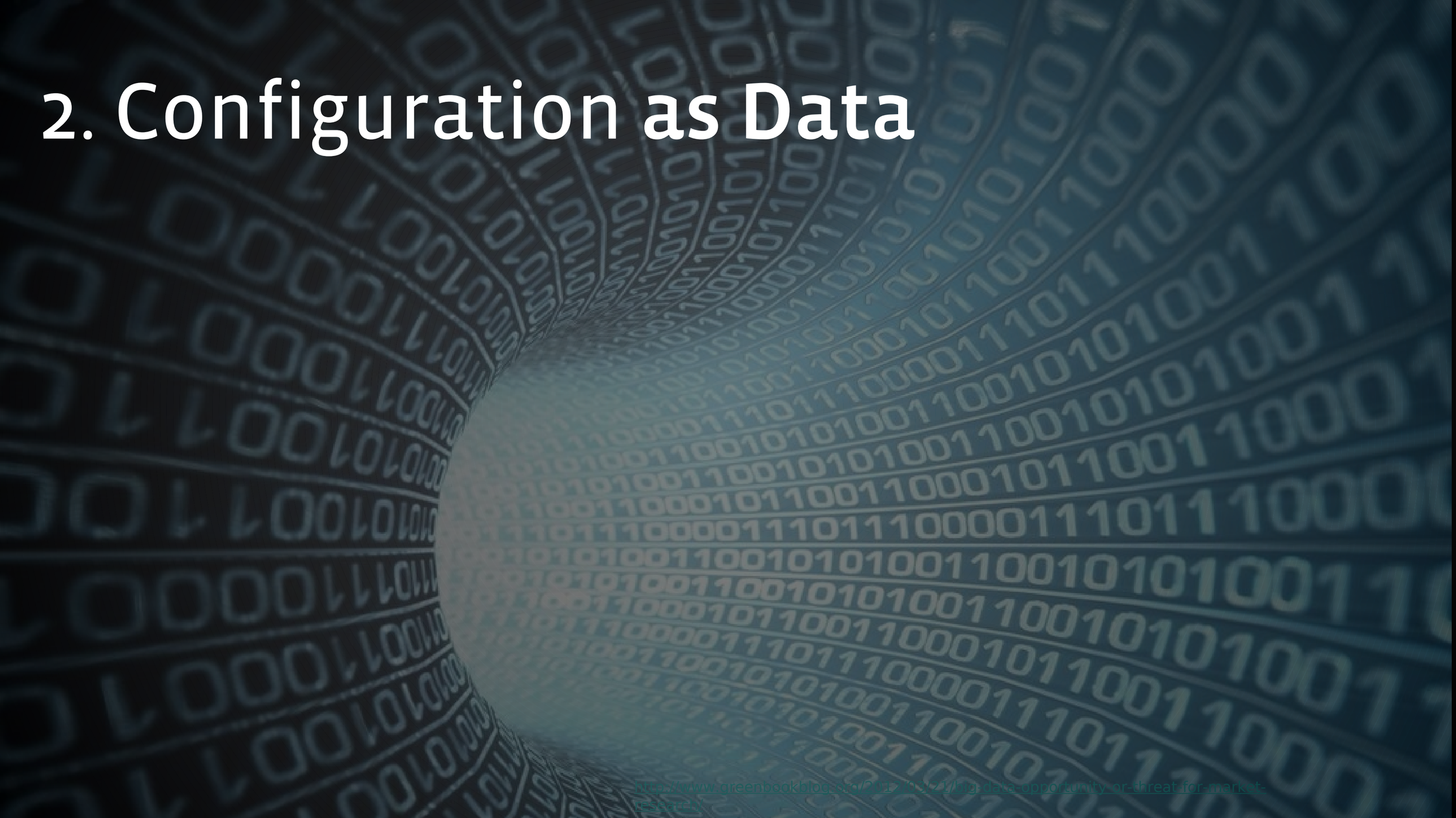
https://www.asburyseminary.edu/elink/my-profile-on-the-hub/

1. Basic Scalable Building Blocks

Basic Scalable Build Blocks

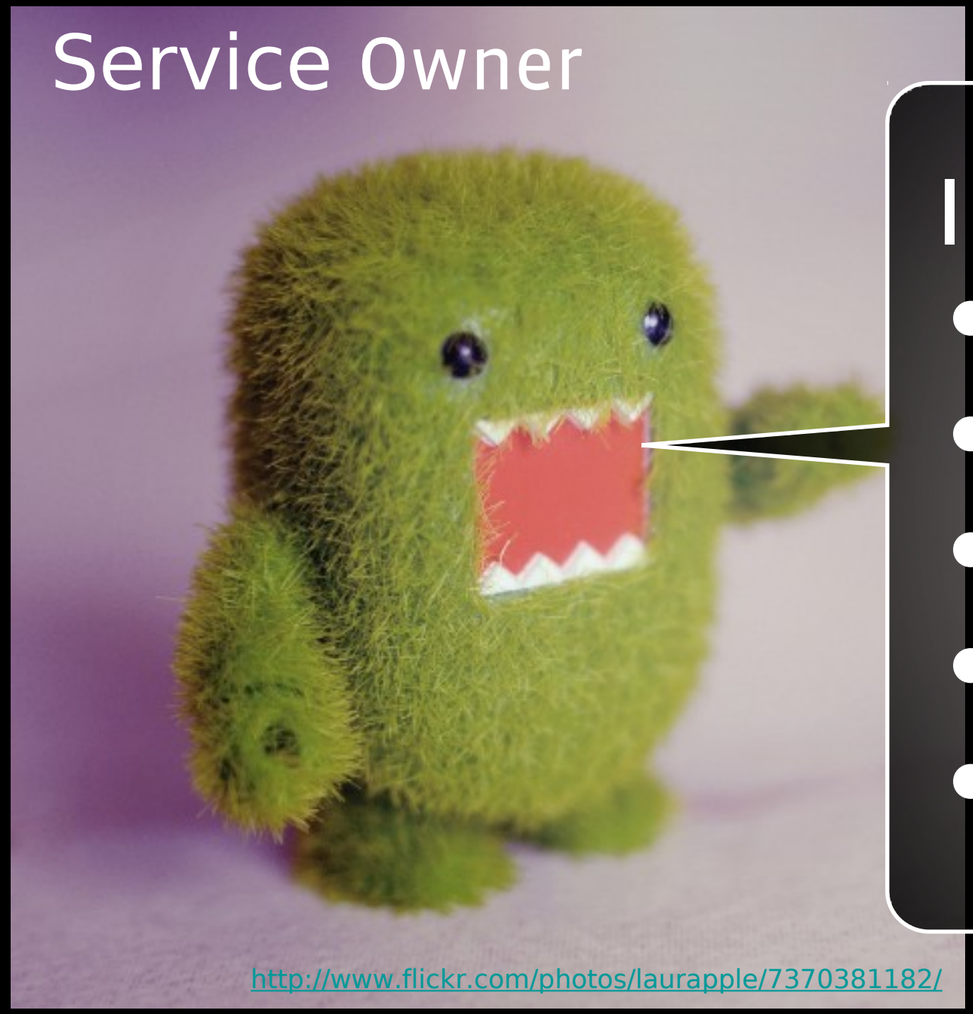
Distributed!	Everything on the client (duh!)
Deterministic!	The system you want on every run
Idempotent!	Only the necessary changes
Extensible!	Tied into internal systems
Flexible!	No dictated workflow

2. Configuration as Data



http://www.greenbookblog.org/2012/09/21/big-data-opportunity-or-threat-for-market-research/

Configuration as Data



http://www.flickr.com/photos/laurapple/7370381182/

I want

- shared mem
- DSR vip
- core files somewhere else
- service running
- less/more/no nscd caching

Configuration as Data

Service Owners **don't** know:

- How to configure DSR
- Optimal sysctl settings
- Network settings
- Authentication settings

3. Flexibility

http://livelovesweat.wordpress.com/2011/12/07/the-importance-of-flexibility/

Flexibility

- Adapt to our workflow
- Super-fast prototyping
- Internal assumptions can be changed - *easily*
- Extend in new ways - *easily*

Flexibility - Example

- Template /etc/sysctl.conf
- Build a hash of default sysctls
- Provide these defaults early in “run”
- Let any engineer munge the bits they want
- /etc/sysctl.conf template interpolated “after”

Picking a tool

http://www.flickr.com/photos/75905404@N00/7126147125/

Many Options

Looked at many options, chose 3 for deep look:

- Spine
- Puppet
- Chef

Other options exist: bcfg2, salt, cfengine3, etc.

Why Chef?

Easier to see from a *problem* with Chef

Chef: The node.save() problem

- node.save() wouldn't scale
  - Can't send that much data from, say, 15k servers every 10-15 minutes (or 5, or 2)

- Standard solution: disable ohai plugins
  - Still too much data
  - Limited the tool unnecessarily

Chef: The node.save() problem

- I want all ohai data for run
- I don’t need it on the chef server
- Solution: use it, but don’t send it!
- Patch Chef? Feature Request?

Chef: whitelist_node_attrs

- New cookbook re-opens Chef::Node.save
- Deletes non-white-listed attrs before saving
- Have as much data as you want during the run
- We send < 1kb back to the server!

Code available:

https://github.com/opscode-cookbooks/whitelist-node-attrs

Chef: whitelist_node_attrs

```
class Chef
  class Node
    alias_method :old_save, :save
    # Overwrite chef’s node.save to whitelist. doesn’t get “later” than this
    def save
      Chef::Log.info(“Whitelisting node attributes”)
      whitelist = self[:whitelist].to_hash
      self.default_attrs = Whitelist.filter(self.default_attrs, whitelist)
      self.normal_attrs = Whitelist.filter(self.normal_attrs, whitelist)
      self.override_attrs = Whitelist.filter(self.override_attrs, whitelist)
      self.automatic_attrs = Whitelist.filter(self.override_attrs, whitelist)
      old_save
    end
  end
end
```

Chef: whitelist_node_attrs

Well… that’s flexible!

Chef: The method_missing problem

```
node.foo('bar')
```

- Ruby: “Is there a method `foo()`?”
- Chef: “If not, is there an attribute `foo`?”
  - “If not, create; assign `bar`”
- OK for…

```
node['foo'] = 'bar'
node.foo = bar
```

- But imagine:

```
node.has_key('foo') # want has_key?()
```

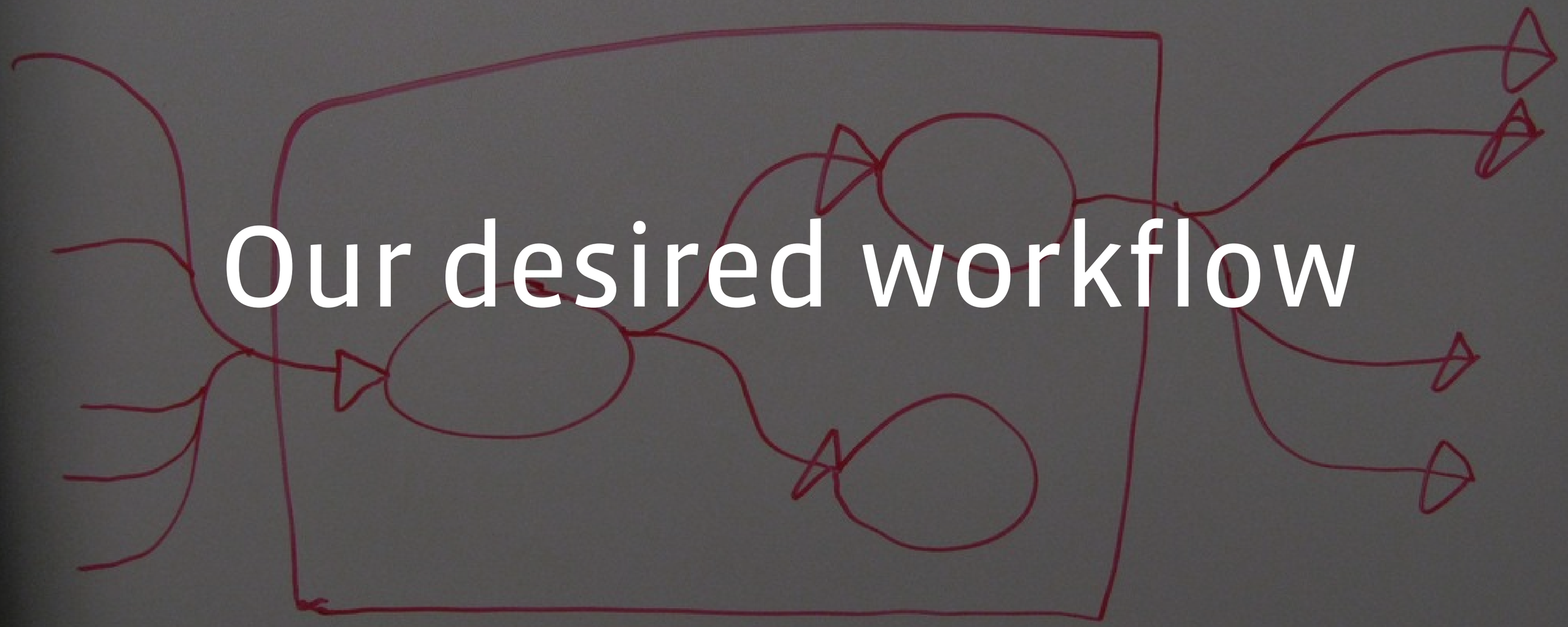
Chef: The method_missing problem

```
class Chef::Node
  def method_missing(method, *args)
    Chef::Log.warn("FB Chef Tweak: Not assuming" +
      " missing method is an attr!")
    Object.send(:method_missing, method, args)
  end
end
```

Chef: The method_missing problem

Again… super flexible!

Our desired workflow



Our Desired Workflow

- Provide API for anyone, anywhere to extend configs by munging data structures
- Engineers don’t need to know what they’re building on, just what they want to change
- Engineers can change their systems without fear of changing anything else
- Testing should be easy
- And…

Something Different

Moving Idempotency “up”

http://www.flickr.com/photos/esi_design/4548531839/sizes/l/in/photostream/

Moving Idempotency Up

- Idempotent records can get stale
  - Remove cron/sysctl/user/etc.
  - Never gets removed => stale entries

- Idempotent systems control *set* of configs
  - Remove cron/sysct/user/etc.
  - No longer rendered in config

Idempotent Records vs. Systems

This is a pain:

```ruby
cron 'tmp_cleaner' do
  minute '5'
  command '/usr/local/bbin/tmp_cleaner'

end


user 'coolsoftd' do
  uid 512
  home '/var/coolsoftd'

end
```

```ruby
# delete after 3/1/13
cron 'tmp_cleaner' do
  minute '5'
  command '/usr/local/bbin/tmp_cleaner'
  action :delete
end

# delete after 3/1/13
user 'coolsoftd' do
  uid 512
  home '/var/coolsoftd'
  action :delete
end
```

Idempotent Records vs. Systems

This is better:

```
cron 'tmp_cleaner' do
  minute '5'
  command '/usr/local/bbin/tmp_cleaner'
end

user 'coolsoftd' do
  uid 512
  home '/var/coolsoftd'
end
```

Case Studies

Case Study 1: sysctl

- fb_sysctl/attributes/default.rb
  - Provides defaults looking at hw, kernel, etc.
- fb_sysctl/recipes/default.rb
  - Defines a template
- fb_sysctl/templates/default/sysctl.erb
  - 3-line template

Case Study 1: sysctl

Template:

```
# Generated by Chef, do not edit directly!
<%- node['fb']['fb_sysctl'].keys.sort.each do |key| %>
<%= key %> = <%= node['fb']['fb_sysctl'][key] %>
<%- end %>
```

Result:

```
# Generated by Chef, do not edit directly!
...
net.ipv6.conf.eth0.accept_ra = 1
net.ipv6.conf.eth0.accept_ra_pinfo = 0
net.ipv6.conf.eth0.autoconf = 0
...
```

Case Study 1: sysctl

In the cookbook for the DB servers:

database/recipes/default.rb

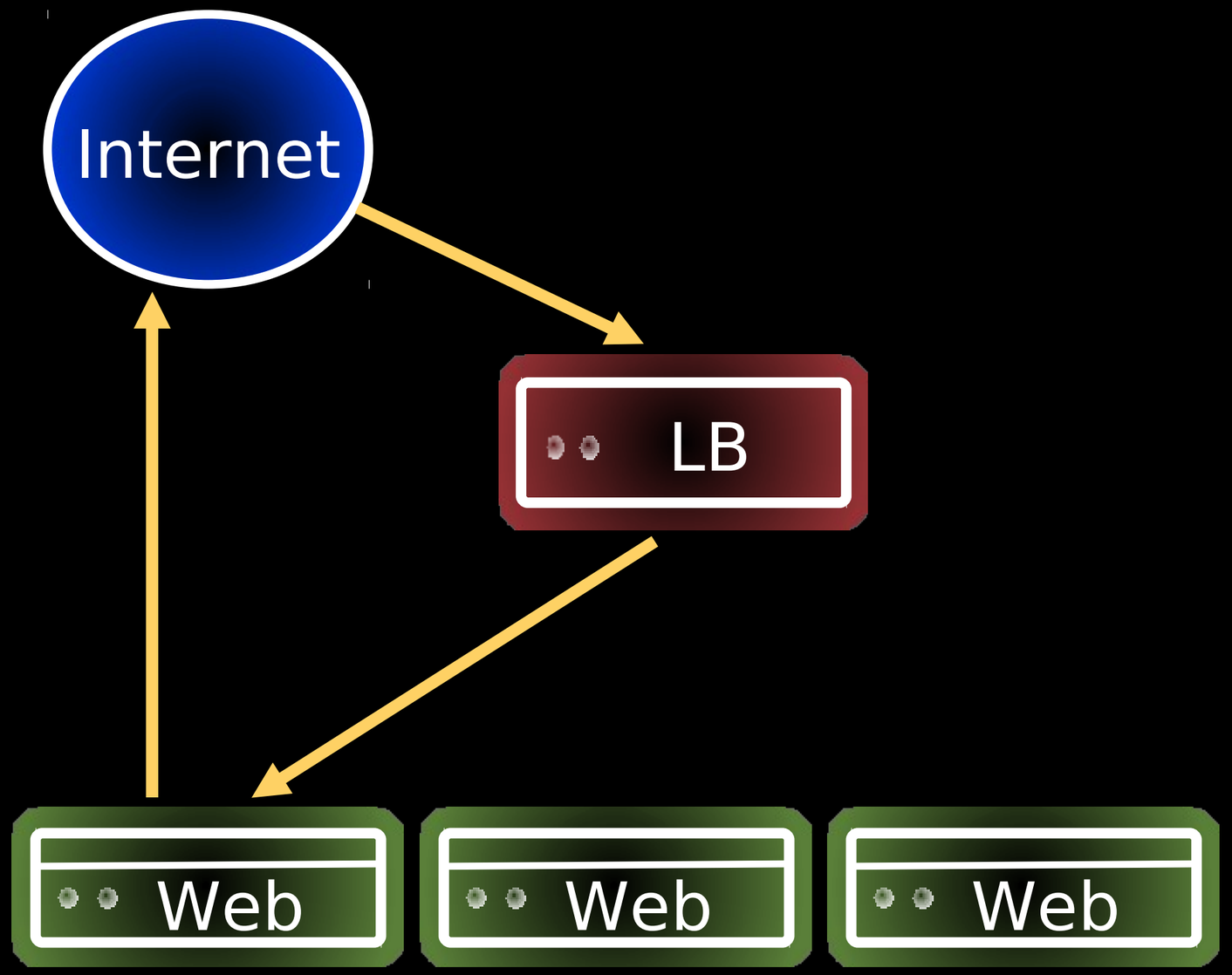
```
node.default['fb']['fb_sysctl']['kernel.shmmax'] = 19541180416
node.default['fb']['fb_sysctl']['kernel.shmall'] = 5432001
```

Case Study 1: sysctl

How does this help us scale?

- Significantly better heterogenous scale
- Fewer people need to manage configs
- Delegation is simple

Case Study 2: DSR



Case Study 2: DSR

- DSR VIPs are hard:
  - L2 networks: dummyX (which one?!)
  - L3 networks: tunl0
  - V6 vips: ip6tnl0
  - May need special routing considerations
- For us:
  - node.add_dsr_vip('10.1.1.2')

Case Study 2: DSR

How does this help us scale?

- Far fewer people

  [only add_dsr_vip() author(s) needs to understand the details]

- More heterogeneous systems
- Delegation is easy

Other Examples

Want IPv6?

```
node.default['fb']['fb_networking']['want_ipv6'] = true
```

Want to know what kind of network?

```
node.is_layer3?()
```

New cronjob?

```
node.default['fb']['fb_cron']['jobs']['myjob'] = {
  'time' => '*/15 * * * *',
  'command' => 'thing',
  'user' => 'myservice',
}
```

Our Chef Infrastructure

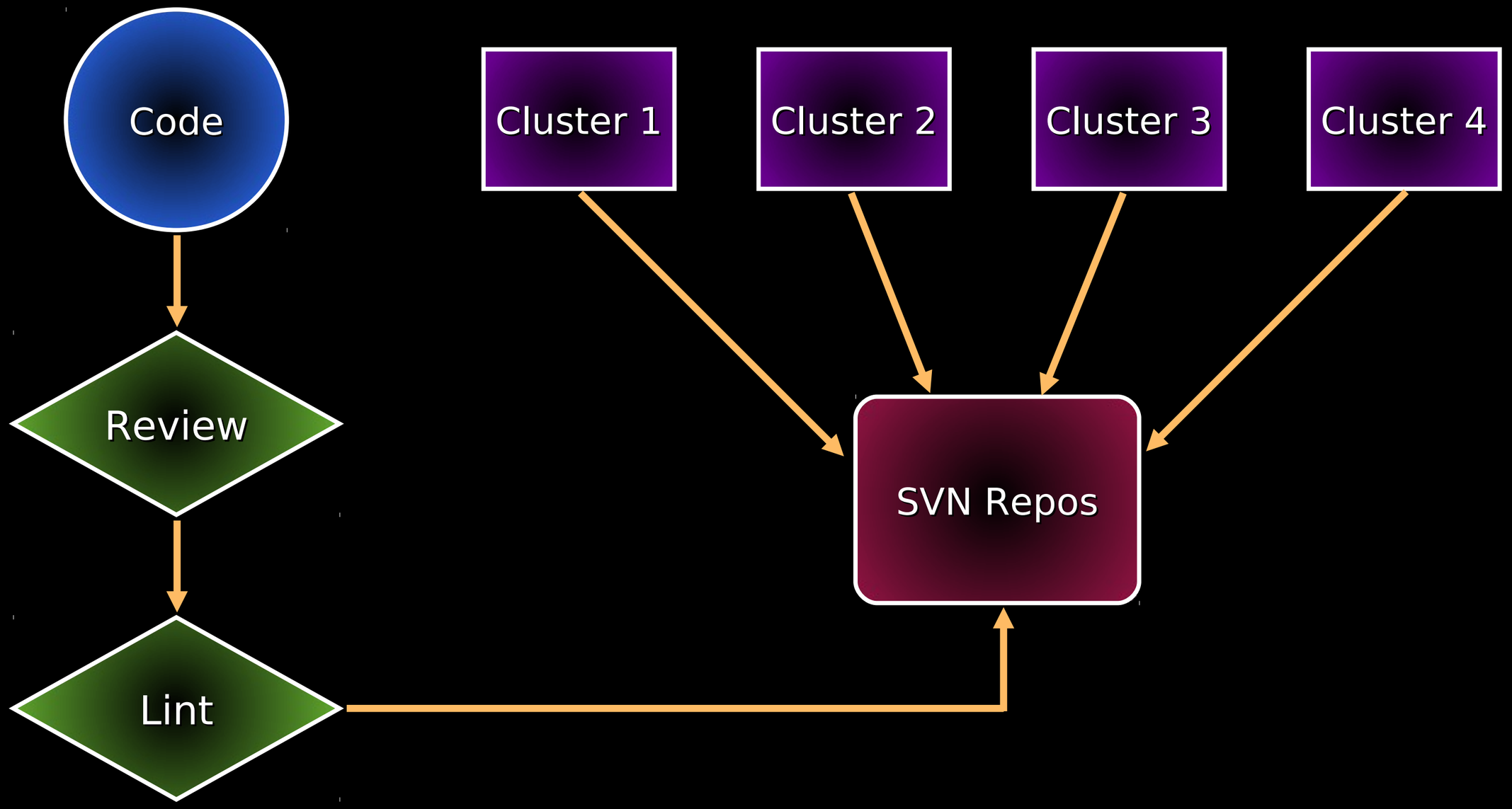
Our Chef Infrastructure

OSC and OPC

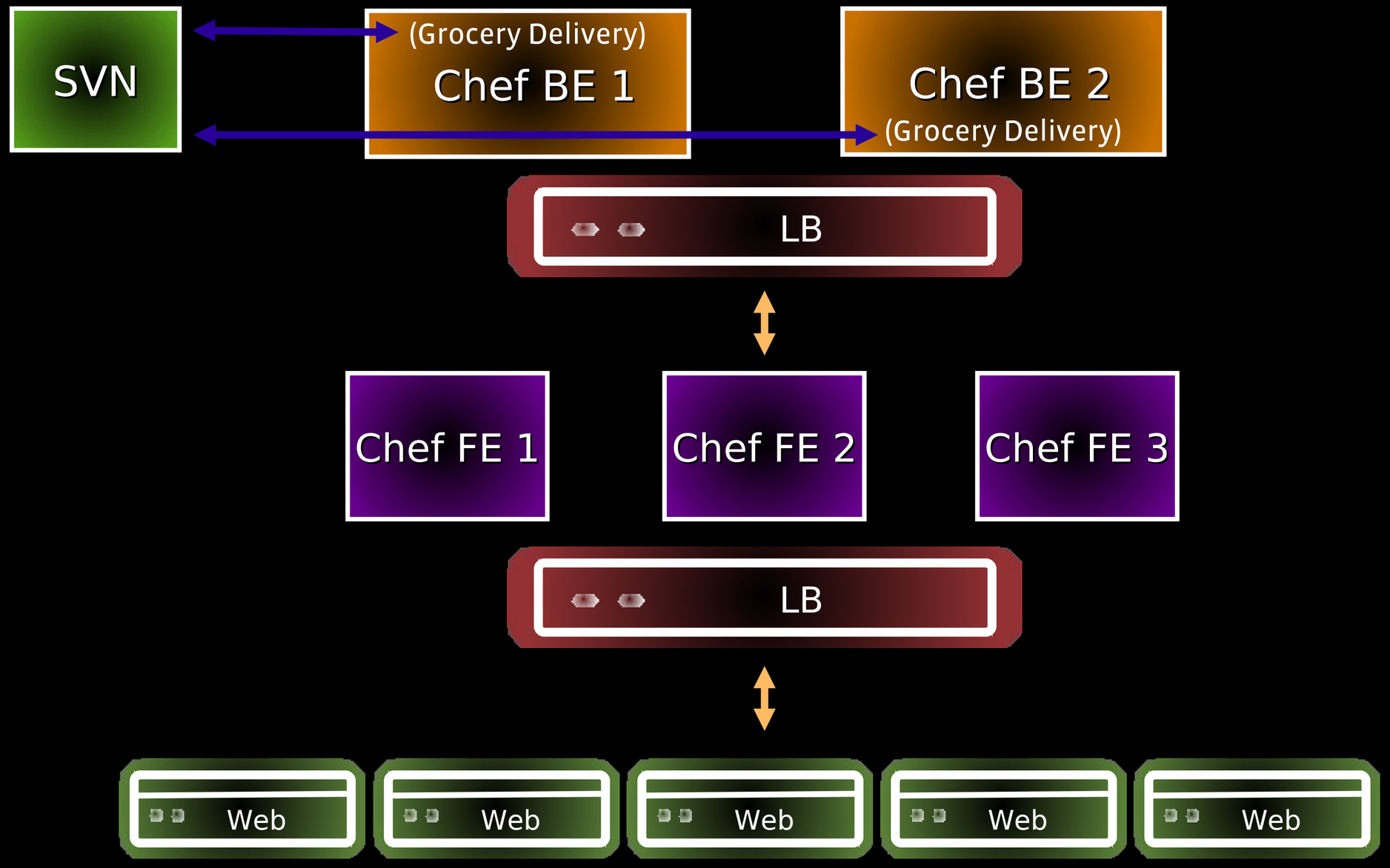
Our Chef Infrastructure - Customizations

- Stateless Chef Servers
  - No search
  - No databags
- Separate Failure Domains
- Tiered Model

Production: Global



Production: Cluster



Assumptions

- Server is basically stateless
  - Node data not persistent
  - No databags
  - grocery_delivery keeps roles/cookbooks in sync
- Chef only knows about the cluster it is in

Implementation Details

- Persistent data needs to come from FB SORs
- Ohai is tied into necessary SORs
- Runlist is forced on every run

Implementation Details: Client

- Report Handlers feed data into monitoring:
  - Last exception seen
  - Success/Failure of run
  - Number of resources
  - Time to run
  - Time since last run
  - Other system info

Implementation Details: Server

- Fed into monitoring :
  - Stats (postgres, authz [opc], etc.)
  - Errors (nginx, erchef, etc.)
  - More…
- Script open source:
  - https://github.com/facebook/chef-utils


But does it scale?

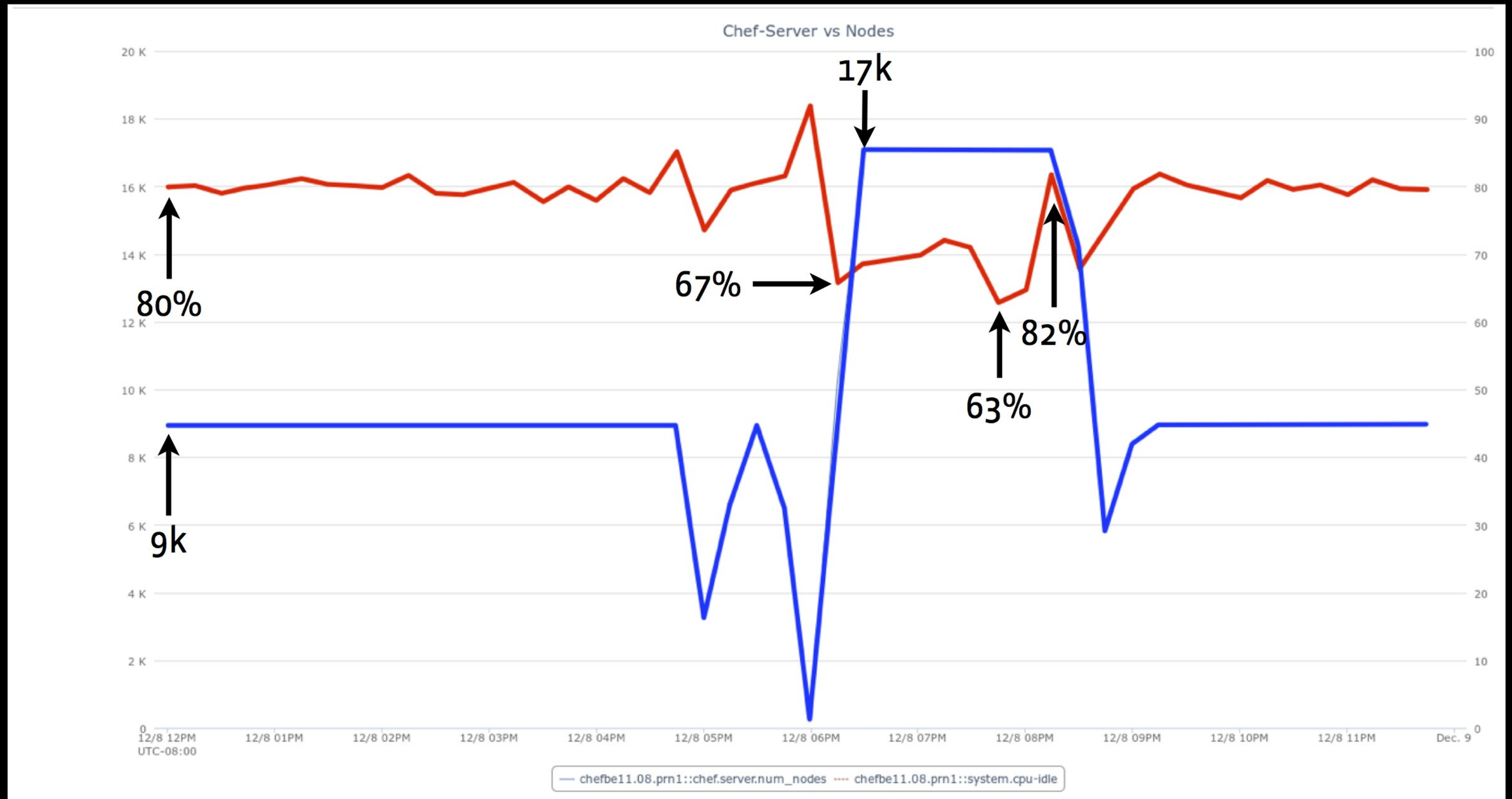
Scale

- Cluster size ~10k+ nodes
- 15 minute convergence (14 min splay)
- grocery_delivery runs every minute
- Lots of clusters

Scale - OSS Chef

Let’s throw more than a cluster at a Chef instance!

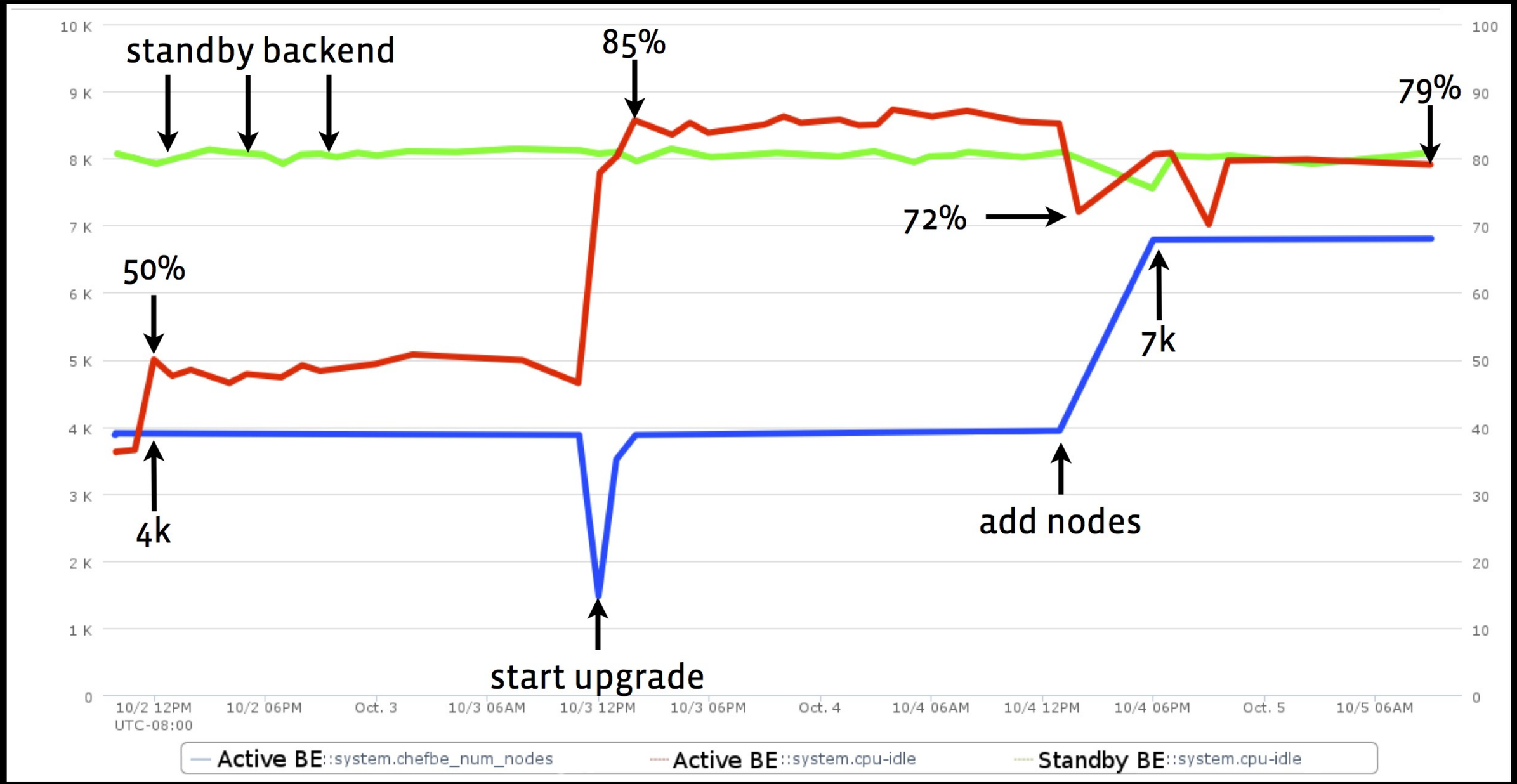
Scale - OSS Chef

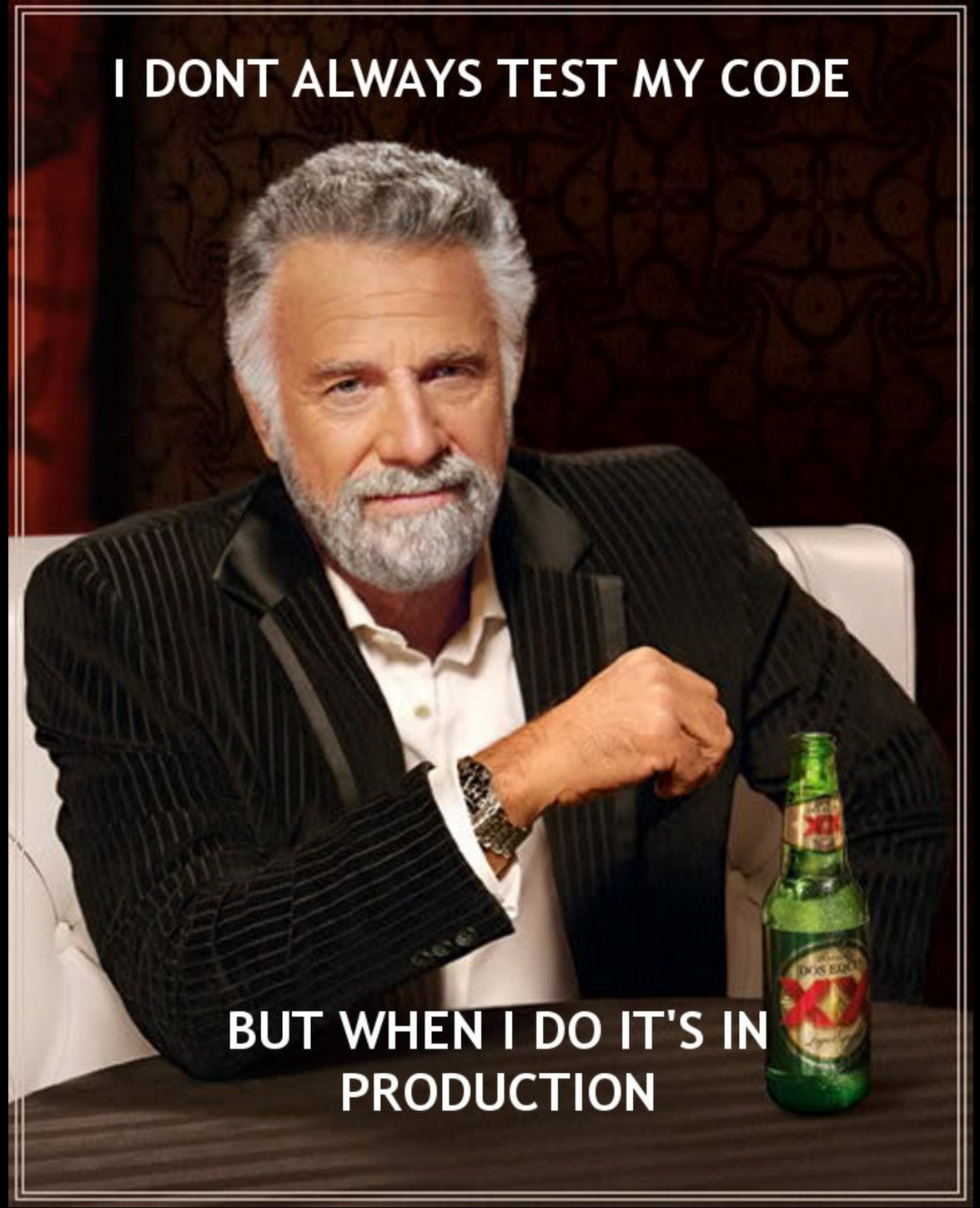


Scale - Erchef (OPC)

Pre-erchef vs Post-erchef

Scale - Erchef (OPC)




I DONT ALWAYS TEST MY CODE
BUT WHEN I DO IT'S IN PRODUCTION

Testing: Desires

- Test on a real production host and pull dependencies
- Don’t rely on people to clean up after themselves
- Should be easy!
- Can test before commit (commits go to prod)

Testing: Approach

- Multi-tenancy
- Everyone gets their own “logical” chef server
- Could be approximated with OSC and some automation

Testing: Approach

Create user and org

```
$ chef_test init
```

Sync your repo to org, test on a server

```
$ chef_test test -s <server>
```

Run Chef on test server

```
server# chef-client
```

Fix bugs, re-sync

```
$ vim … ; chef_test upload
```

Lessons

Lessons

- Idempotent systems > idempotent records
- Delegating delta config == easier heterogeneity
- Full programming languages > restrictive DSLs
- Scale is more than just a number of clients
- Easy abstractions are critical
- Testing against real systems is useful and necessary

Summary

So how about those types of scale?

Summary

How many homogeneous systems can you maintain?	>> 17k
How many heterogeneous systems can you maintain?	> 17k
How many people are needed?	~4
Can you safely delegate delta configuration?	Yes

Thanks

- Opscode
  - Adam Jacob, Chris Brown, Steven Danna & the erchef team
- Andrew Crump
  - foodcritic rules!
- Everyone I work with
  - KC, Larry, David, Pedro, Bethanye

facebook
INFRASTRUCTURE