

Introduction to Chef for SCALE 10x

someara@opscode.com
www.opscode.com
@someara





Chnef

Congratulations!!!



- U has a cloud
- Now what?

http://www.flickr.com/photos/ian_munroe/4758240536/

APIs are awesome

- You can provision **compute** resources in seconds
- You can provision **storage** resources in seconds
- That's cool.

<http://www.flickr.com/photos/jdhancock/3634246981/>



Chef can help with that



- knife ec2 server create
- knife rackspace server create
- knife terremark server create
- knife voxel server create
- knife gandi server create
- knife cloudstack server create
- knife vsphere server create
- knife eucalyptus server create
- knife openstack server create

<http://www.flickr.com/photos/kyz/3122499444/>

But then what?





You need to configure them

Applications





Collection of Resources

- Nodes
 - Networking
 - Files
 - Directories
 - Symlinks
 - Mounts
- Routes
 - Users
 - Groups
 - Tasks
 - Packages
 - Software
 - Services
 - Configurations
 - Stuff



Acting in concert



RULE THE

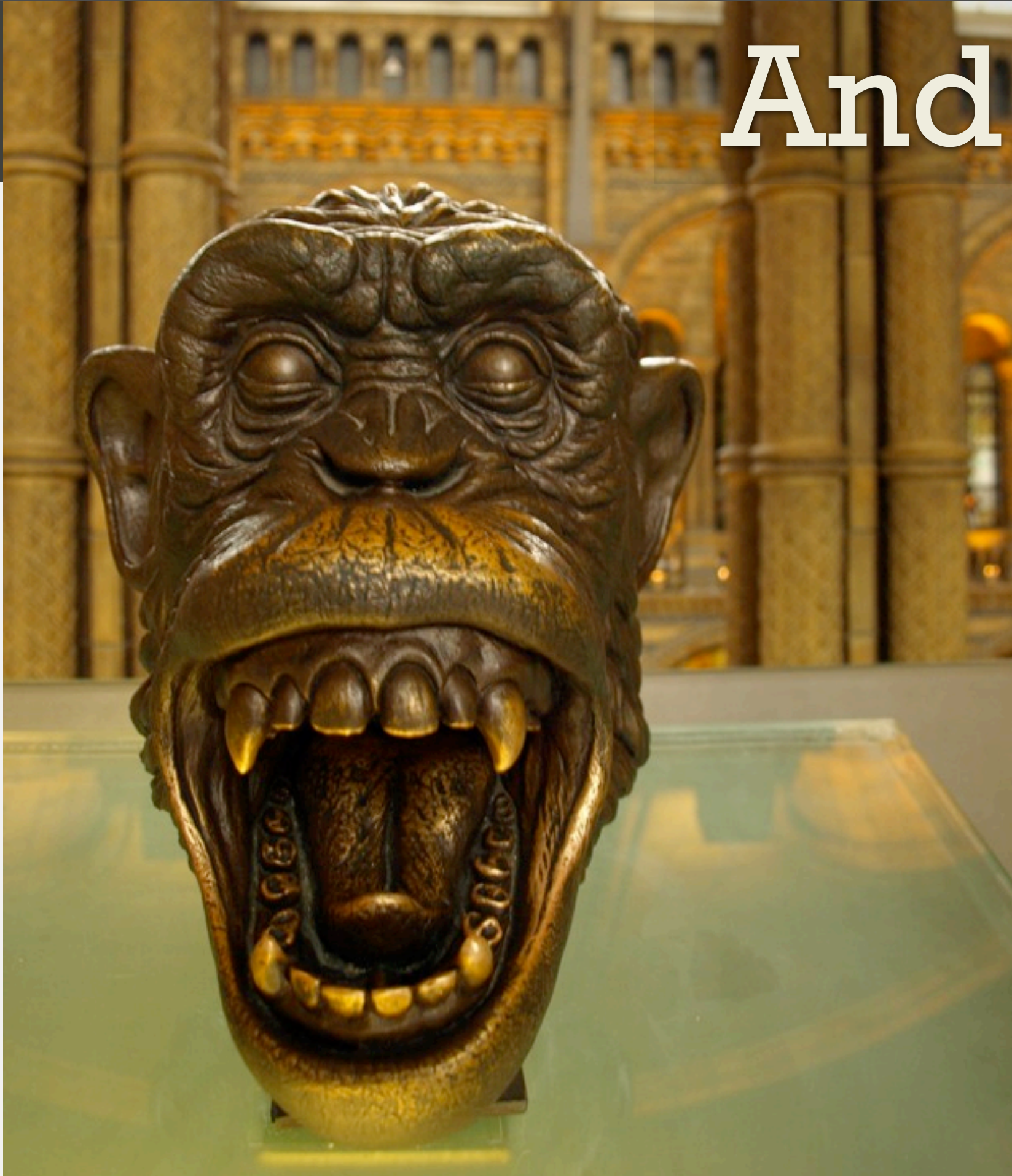
<http://www.flickr.com/photos/glowjangles/4081048126/>

To provide a Service

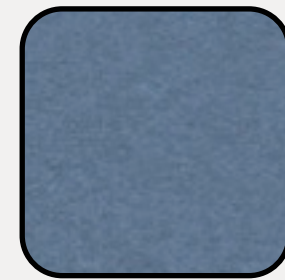


<http://www.flickr.com/photos/28309157@N08/3743455858/>

And it *evolves*



<http://www.flickr.com/photos/16339684@N00/2681435235/>



Application

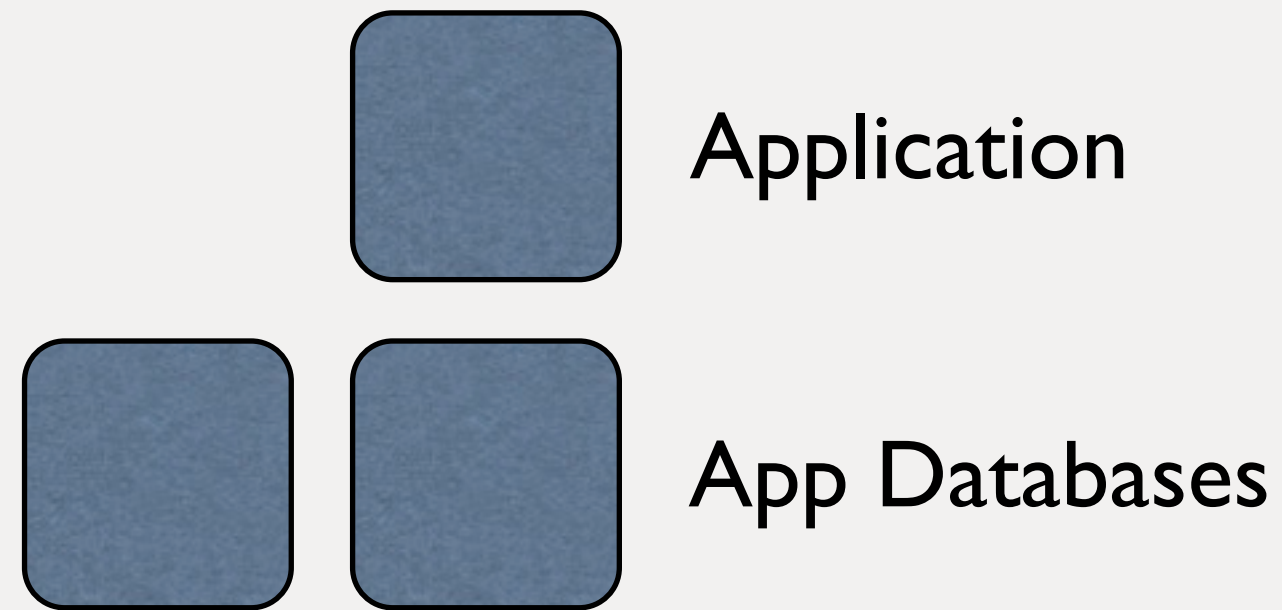


Application

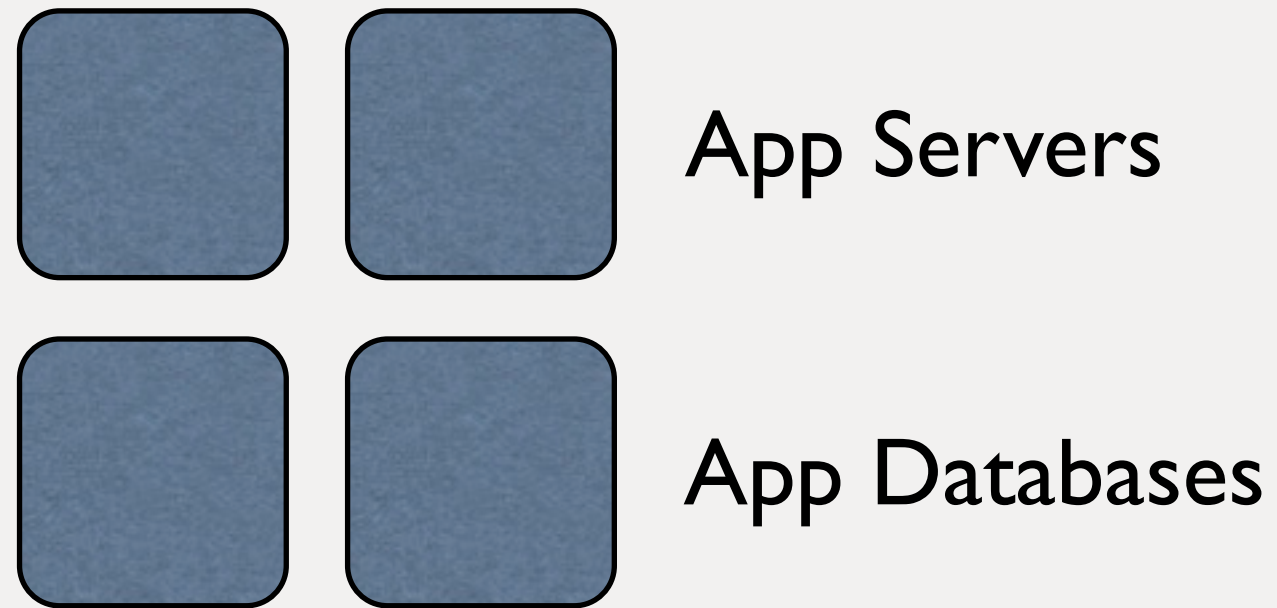


Application Database

See Nodes Grow



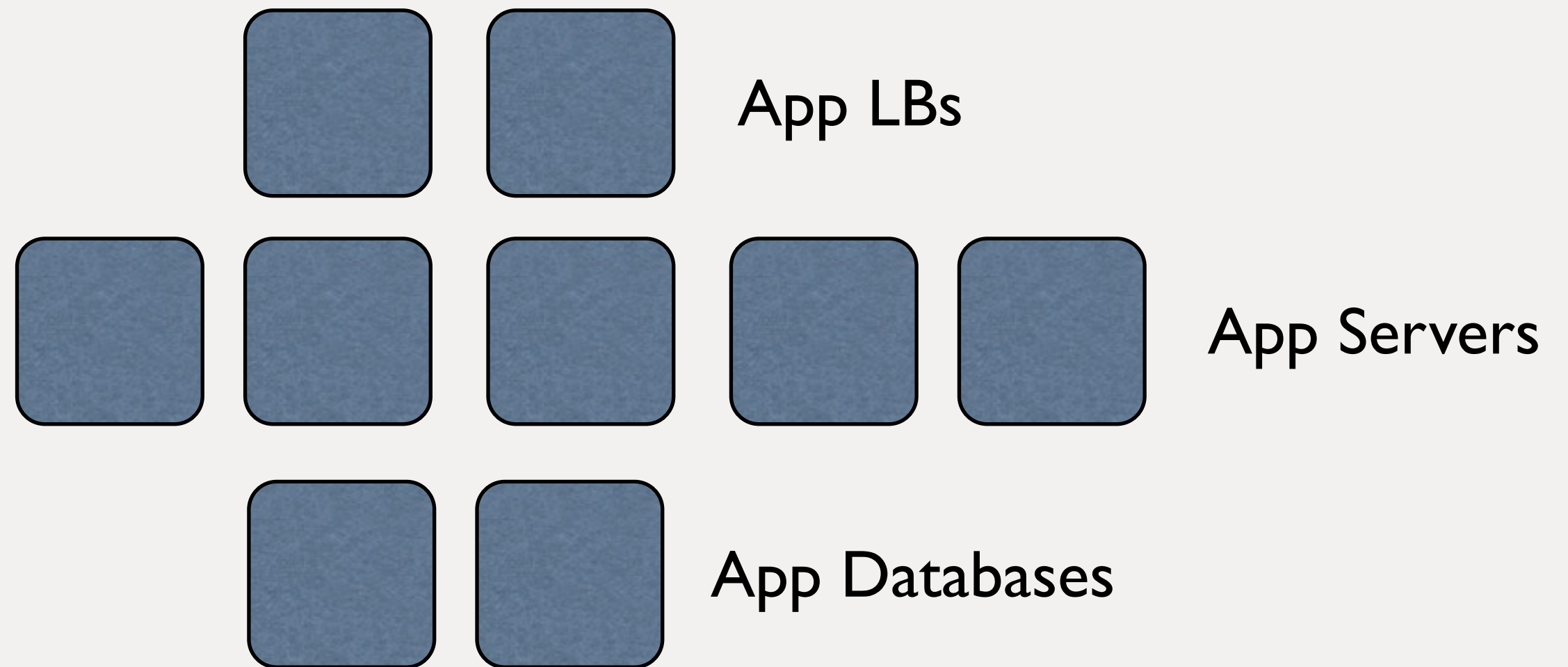
See Nodes Grow



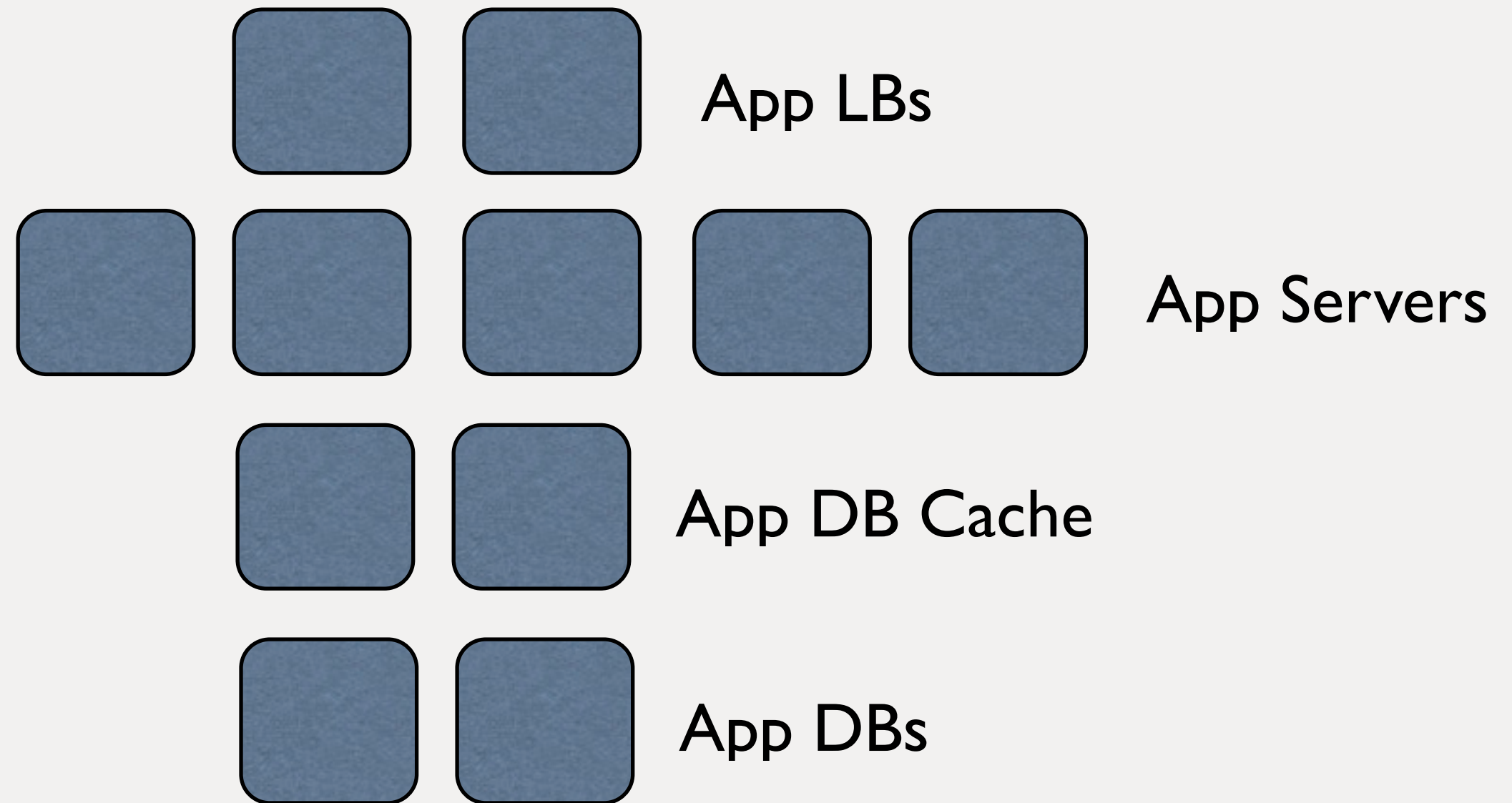
See Nodes Grow



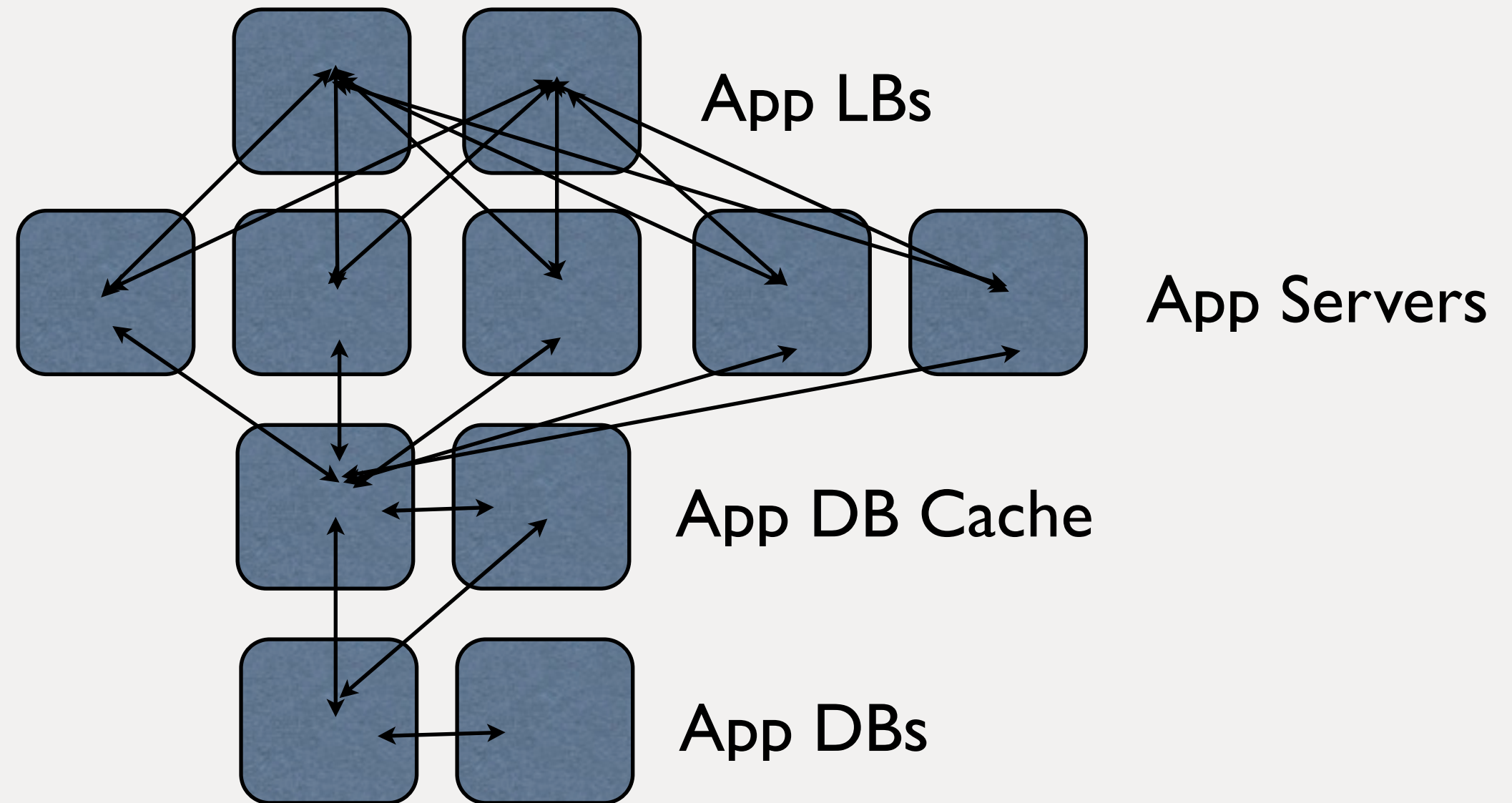
See Nodes Grow



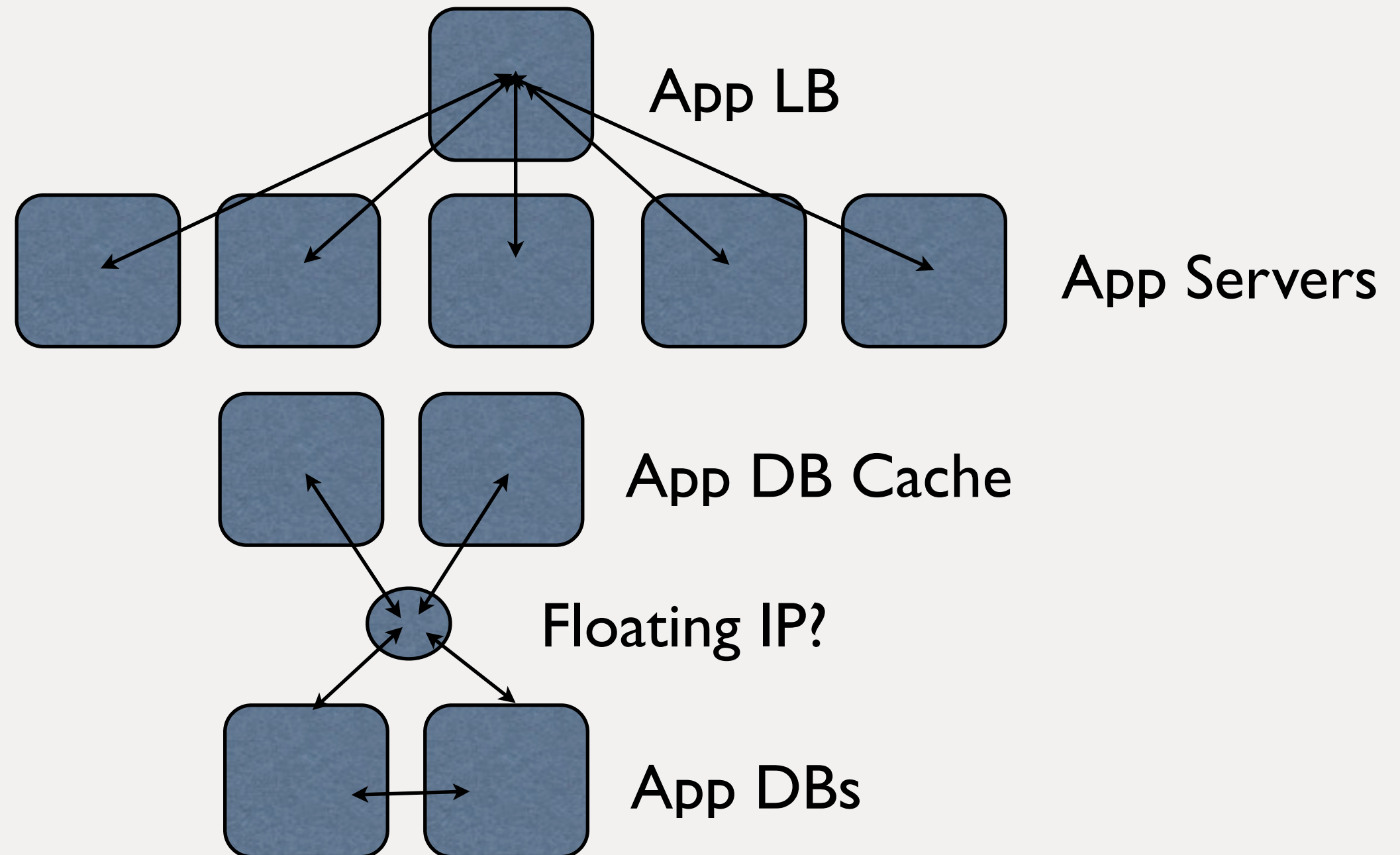
See Nodes Grow



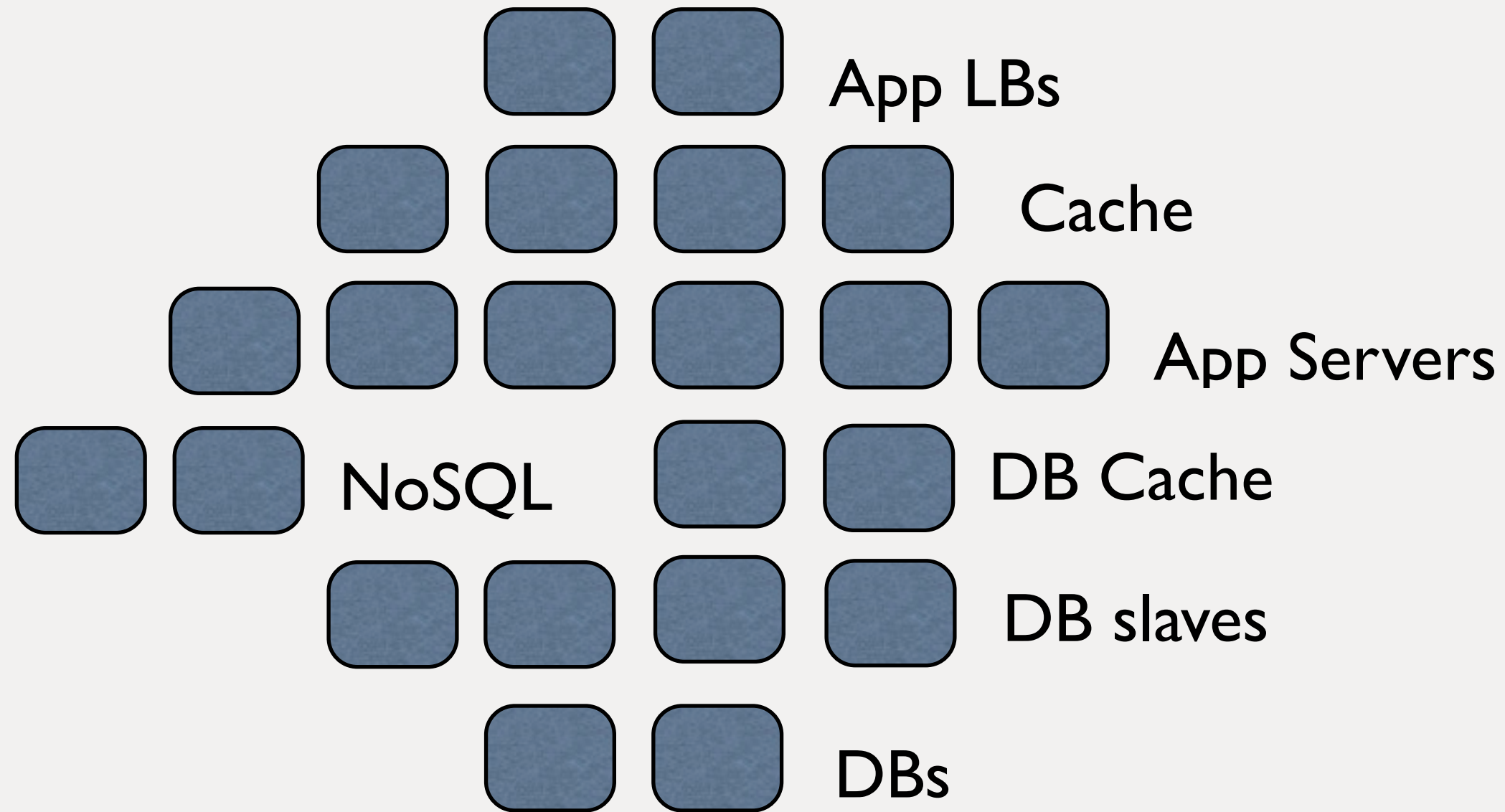
Stitched together with configs

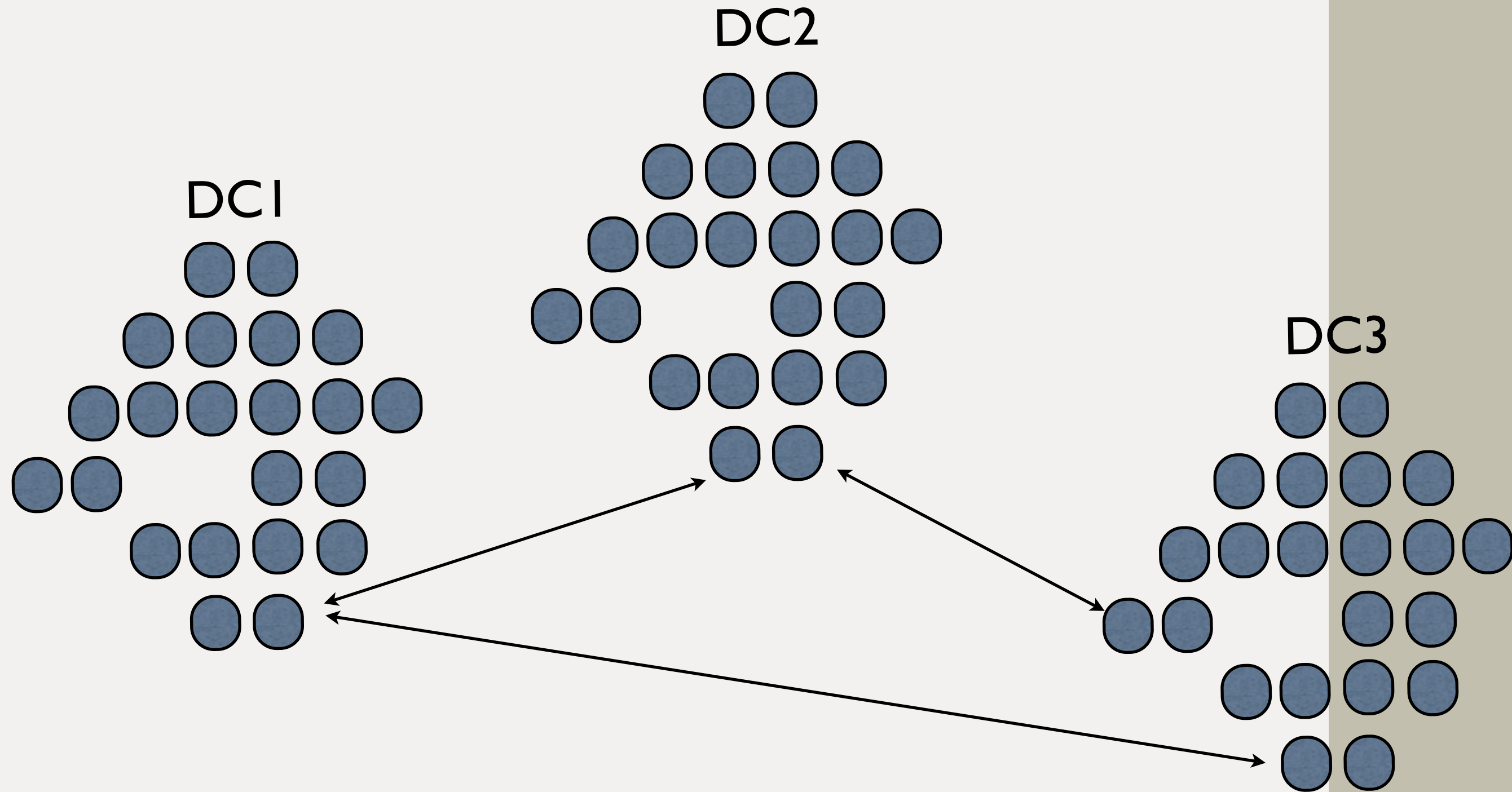


Your Infrastructure is a snowflake



Complexity increases quickly







Configuration Management

<http://www.flickr.com/photos/philliecasablanca/3354734116/>

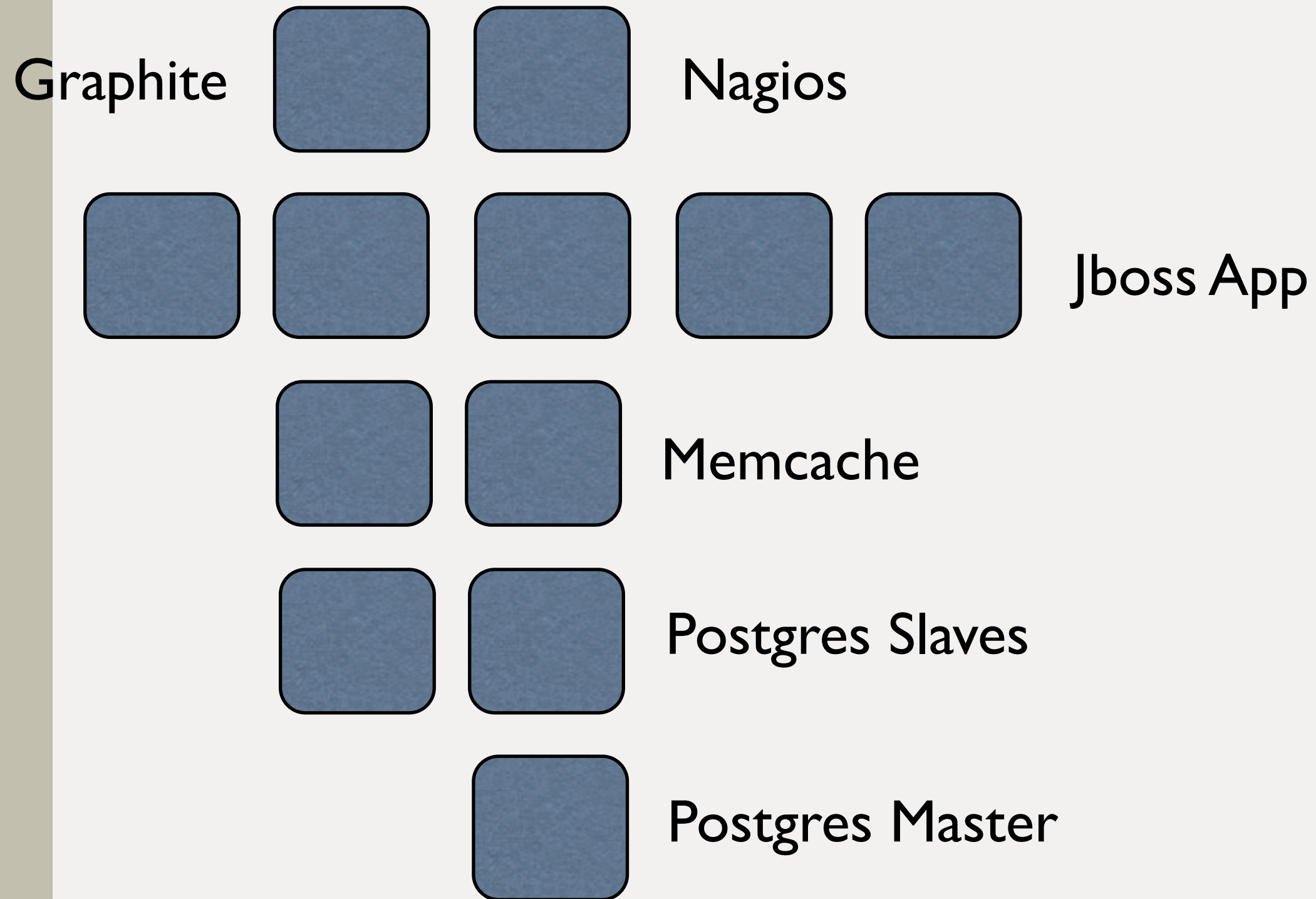
Golden Images are not the answer

- Gold is heavy
- Hard to transport
- Hard to mold
- Easy to lose configuration detail



<http://www.flickr.com/photos/garysoup/2977173063/>

Typical Boring Infrastructure



New Compliance Mandate

Graphite



Nagios



Jboss App



Memcache



Postgres Slaves



Postgres Master

- Move SSH off port 22

- Lets put it on 2022

6 Golden Image Updates

Graphite

1

2

Nagios

3

Jboss App

4

Memcache

5

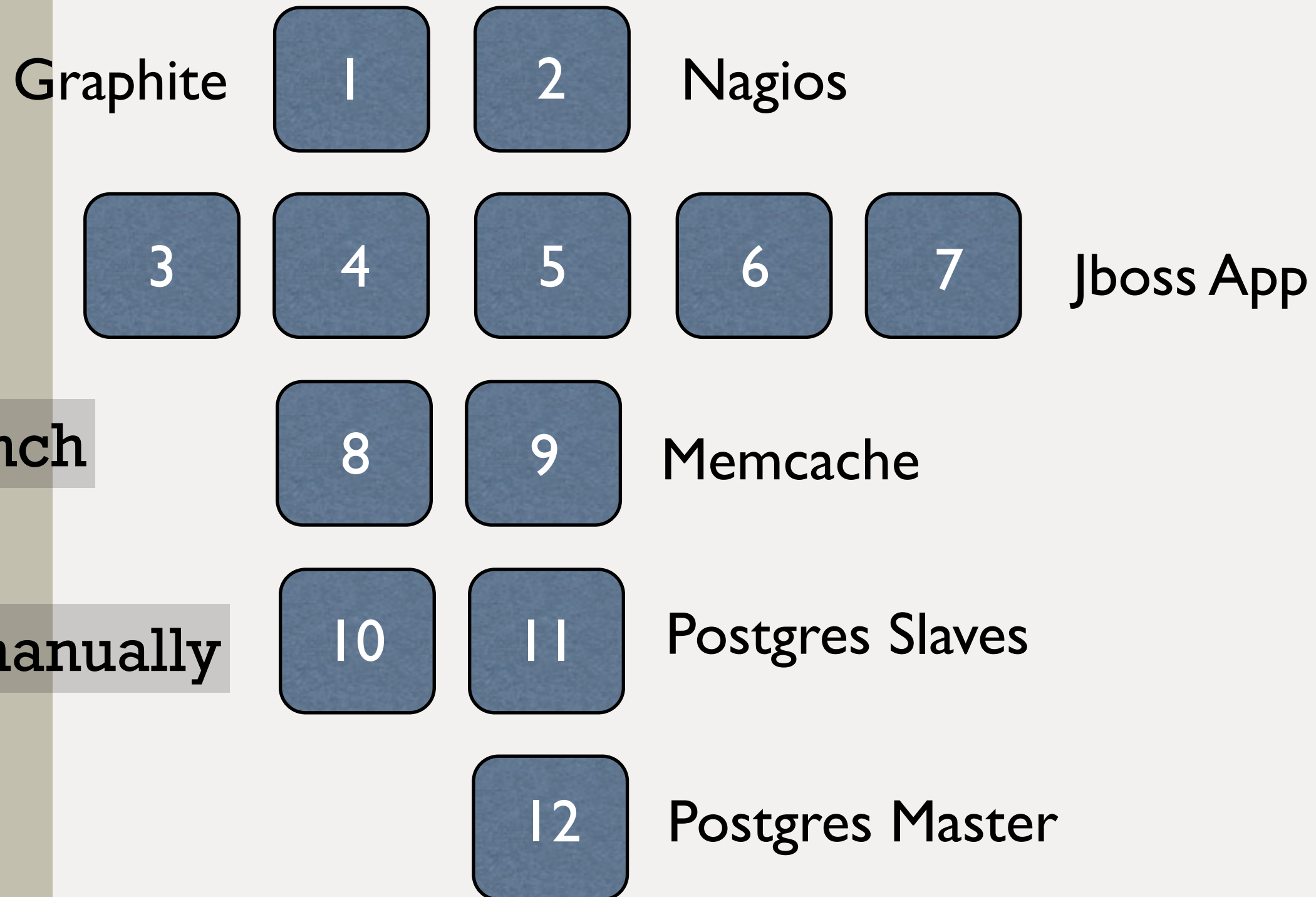
Postgres Slaves

6

Postgres Master

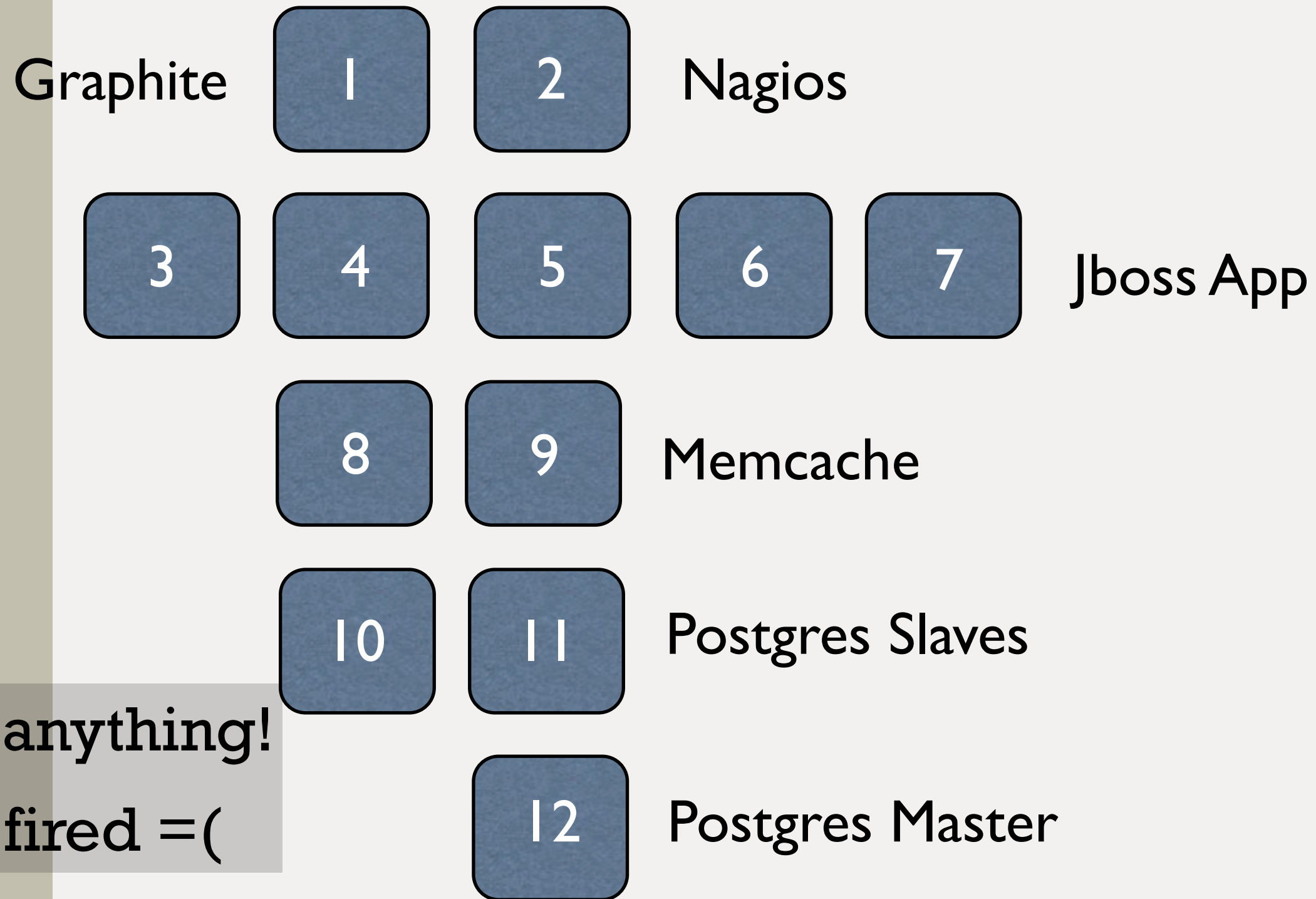
- `edit /etc/ssh/sshd_config`

12 Instance Replacements



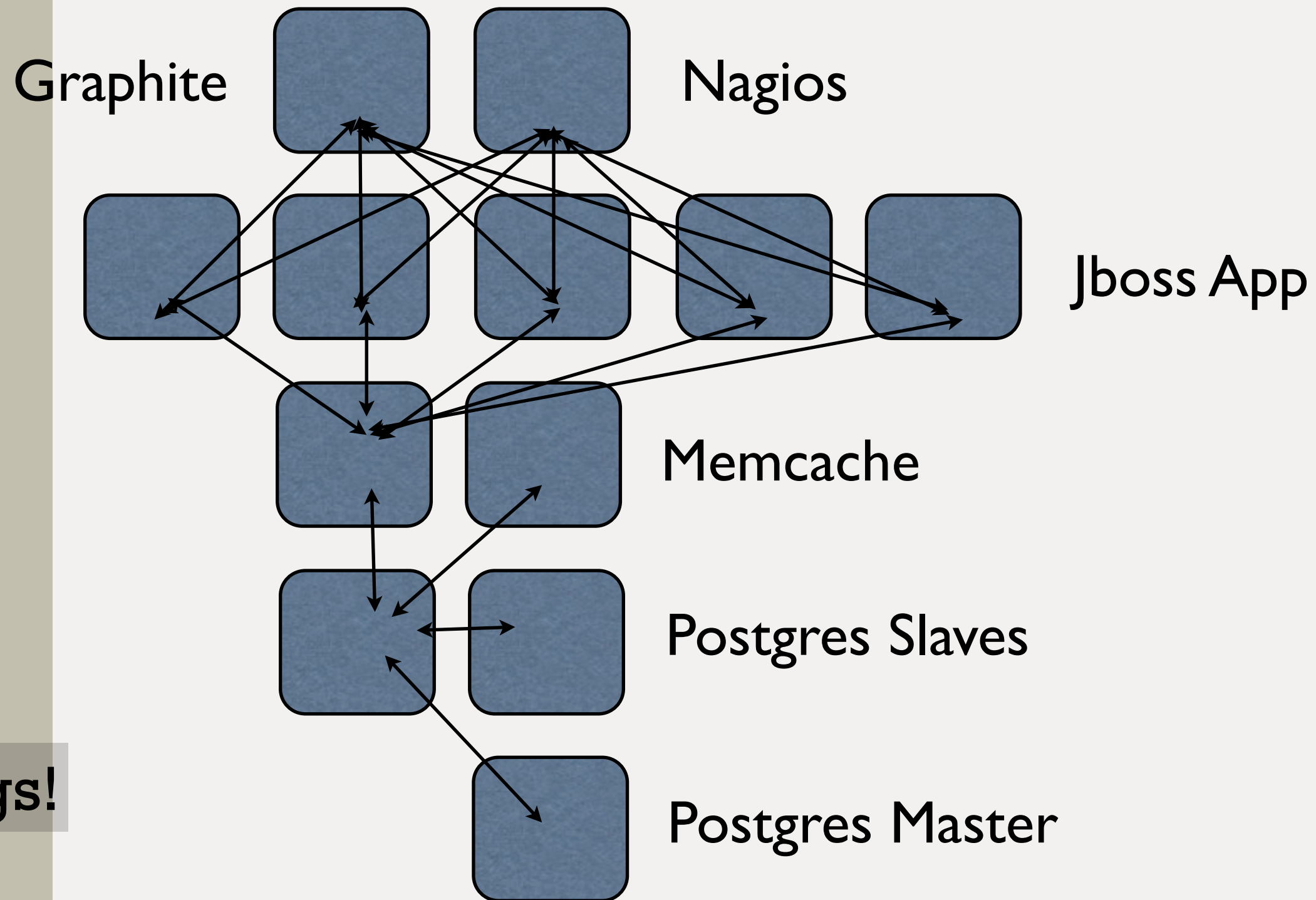
- Delete, launch
- Repeat
- Typically manually

Done in Maintenance Windows



- Don't break anything!
- Bob just got fired =(

Different IP Addresses?



● Invalid configs!

Configuration Desperation



RULE THE

<http://www.flickr.com/photos/francoforeshock/5716969942/>

Chef Solves This Problem



- But you already guessed that, didn't you?

RULE THE CLOUD

Programs!

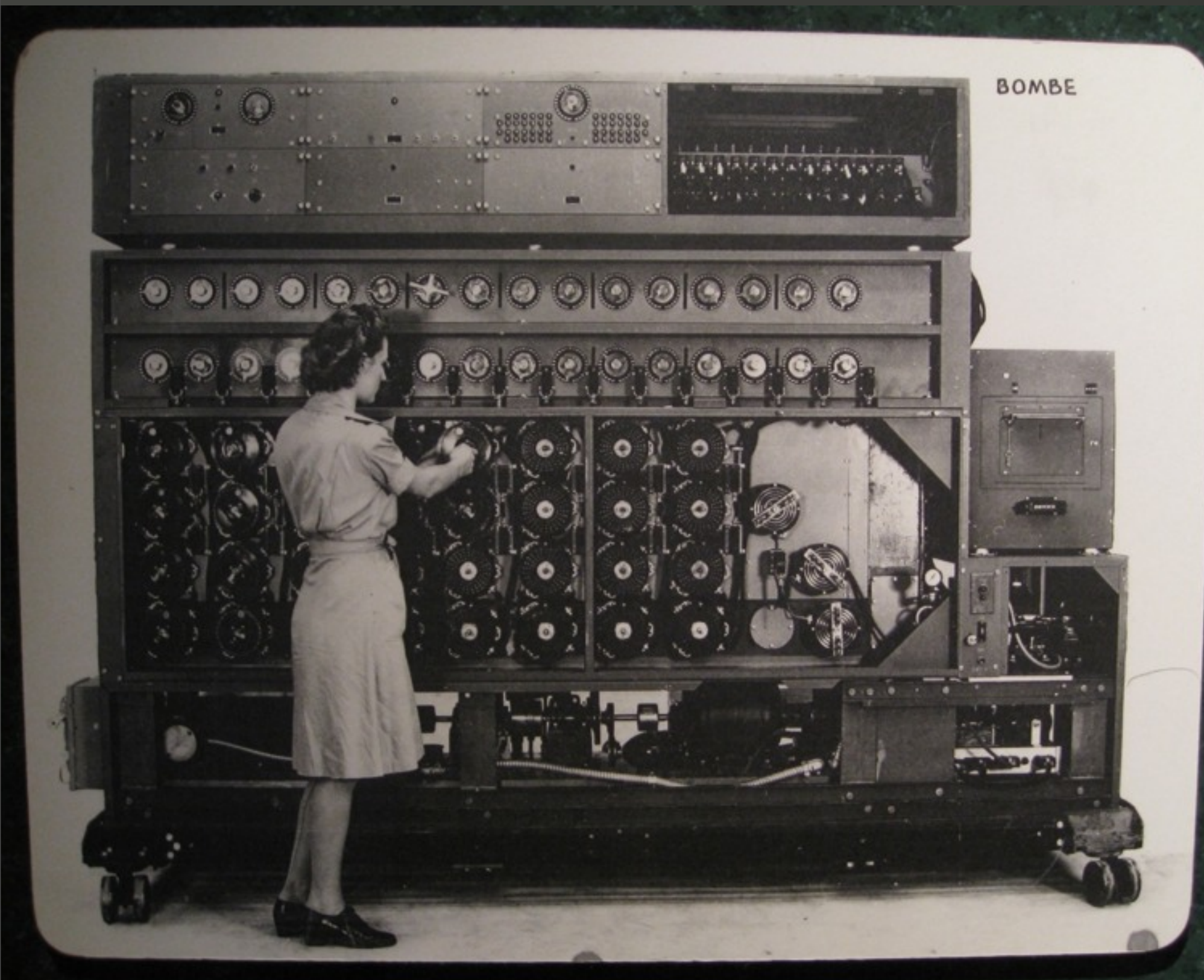
- Generate configurations directly on nodes
- Reduce management complexity
- Version control the programs



- Define policy
- Say what, not how
- Pull not Push



Chef is Infrastructure as Code



- Programmatically provision and configure
- Treat like any other code base
- Reconstruct business from code repository, data backup, and bare metal resources.

<http://www.flickr.com/photos/louisb/4555295187/>

That looks like this

```
package "ntp" do
  action :install
end
```

```
template "/etc/ntpd.conf" do
  source "ntpd.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
  variables(:time_server => "time.example.com")
  notifies :restart, "service[ntpd]"
end
```

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

```
package "net-snmp" do
  action :install
end
```

```
template "/etc/snmpd.conf" do
  source "snmpd.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
  variables(:community_string => "not_public")
  notifies :restart, "service[snmpd]"
end
```

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

Ohai!

```
"hostname": "server-1",
"fqdn": "server-1.example.com",
"domain": "example.com",
"network": {
  "interfaces": {
    "eth0": {
      "type": "eth",
      "number": "0",
      "encapsulation": "Ethernet",
      "addresses": {
        "00:0C:29:43:26:C5": {
          "family": "lladdr"
        },
        "192.168.177.138": {
          "family": "inet",
          "broadcast": "192.168.177.255",
          "netmask": "255.255.255.0"
        },
        "fe80::20c:29ff:fe43:26c5": {
          "family": "inet6",
          "prefixlen": "64",
          "scope": "Link"
        }
      }
    }
  },
}
```

```
"memory": {
  "swap": {
    "cached": "0kB",
    "total": "4128760kB",
    "free": "4128760kB"
  },
  "total": "2055676kB",
  "free": "1646524kB",
  "buffers": "35032kB",
  "cached": "210276kB",
  "active": "125336kB",
  "inactive": "142884kB",
  "dirty": "8kB",
  "writeback": "0kB",
  "anon_pages": "22976kB",
  "mapped": "8416kB",
  "slab": "121512kB",
  "slab_reclaimable": "41148kB",
  "slab_unreclaim": "80364kB",
  "page_tables": "1784kB",
  "nfs_unstable": "0kB",
  "bounce": "0kB",
  "commit_limit": "5156596kB",
  "committed_as": "74980kB",
  "vmalloc_total": "34359738367kB",
  "vmalloc_used": "274512kB",
  "vmalloc_chunk": "34359449936kB"
},
```

```
"block_device": {
  "ram0": {
    "size": "32768",
    "removable": "0"
  },
  "ram1": {
    "size": "32768",
    "removable": "0"
  },
  "ram2": {
    "size": "32768",
    "removable": "0"
  },
}
```

Decide what to declare

```
execute "load sysctl" do
  command "/sbin/sysctl -p"
  action :nothing
end

bytes = node['memory']['total'].split("kB")[0].to_i * 1024 / 3,
pages = node['memory']['total'].split("kB")[0].to_i * 1024 / 3 / 2048

# adjust shared memory and semaphores
template "/etc/sysctl.conf" do
  source "sysctl.conf.erb"
  variables(
    :shmmax_in_bytes => bytes,
    :shmall_in_pages => pages
  )
  notifies :run, "execute[load sysctl]", :immediately
end
```

```
size = ((2 * 3) * 4) / 2

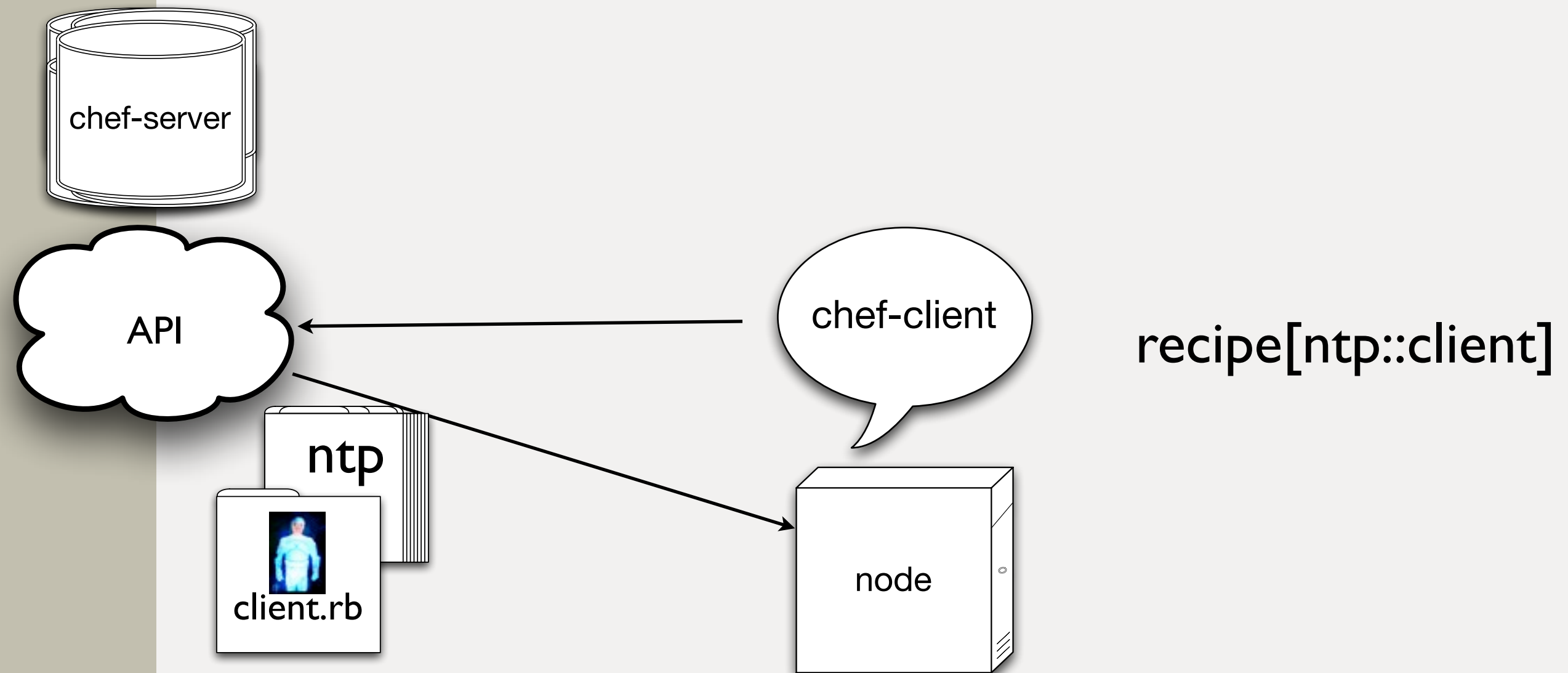
99.downto(1) do |i|
  beer_bottle "bottle-#{i}" do
    oz size
    action [ :take_down, :pass_around ]
  end
end
```

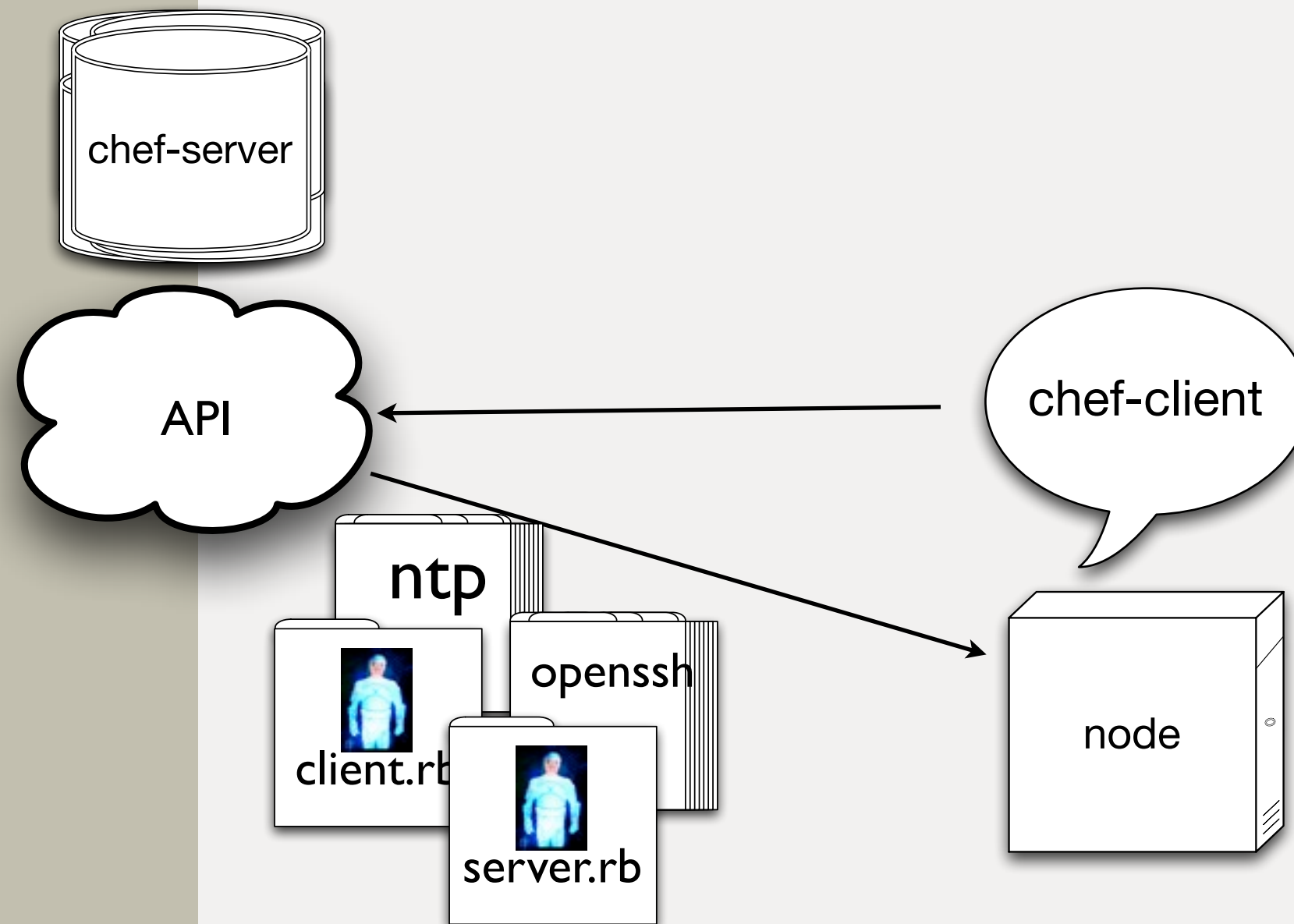
Recipes and Cookbooks

- Recipes are collections of Resources
- Cookbooks contain recipes, templates, files, custom resources, etc
- Code re-use and modularity

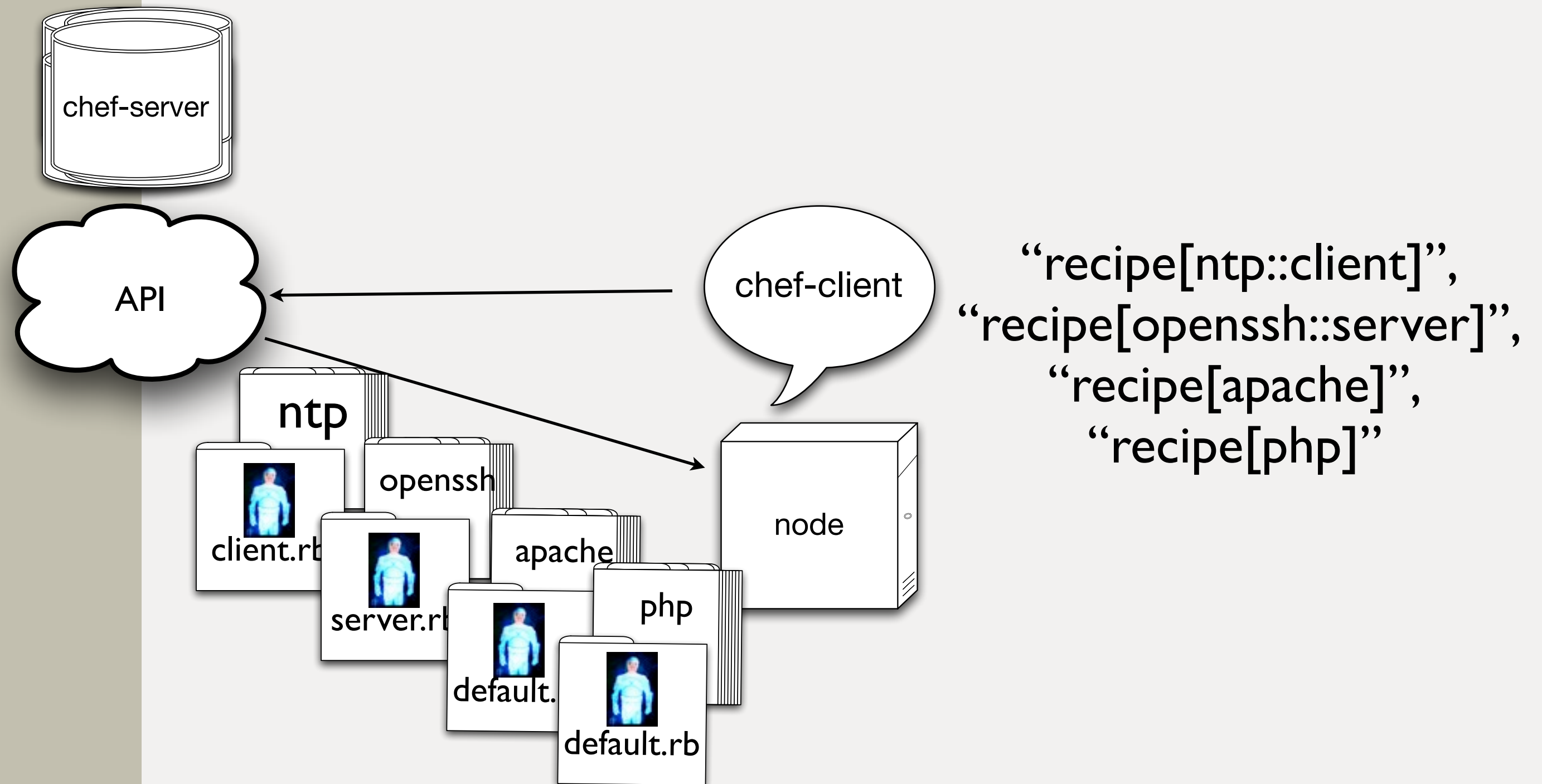
<http://www.flickr.com/photos/shutterhacks/4474421855/>





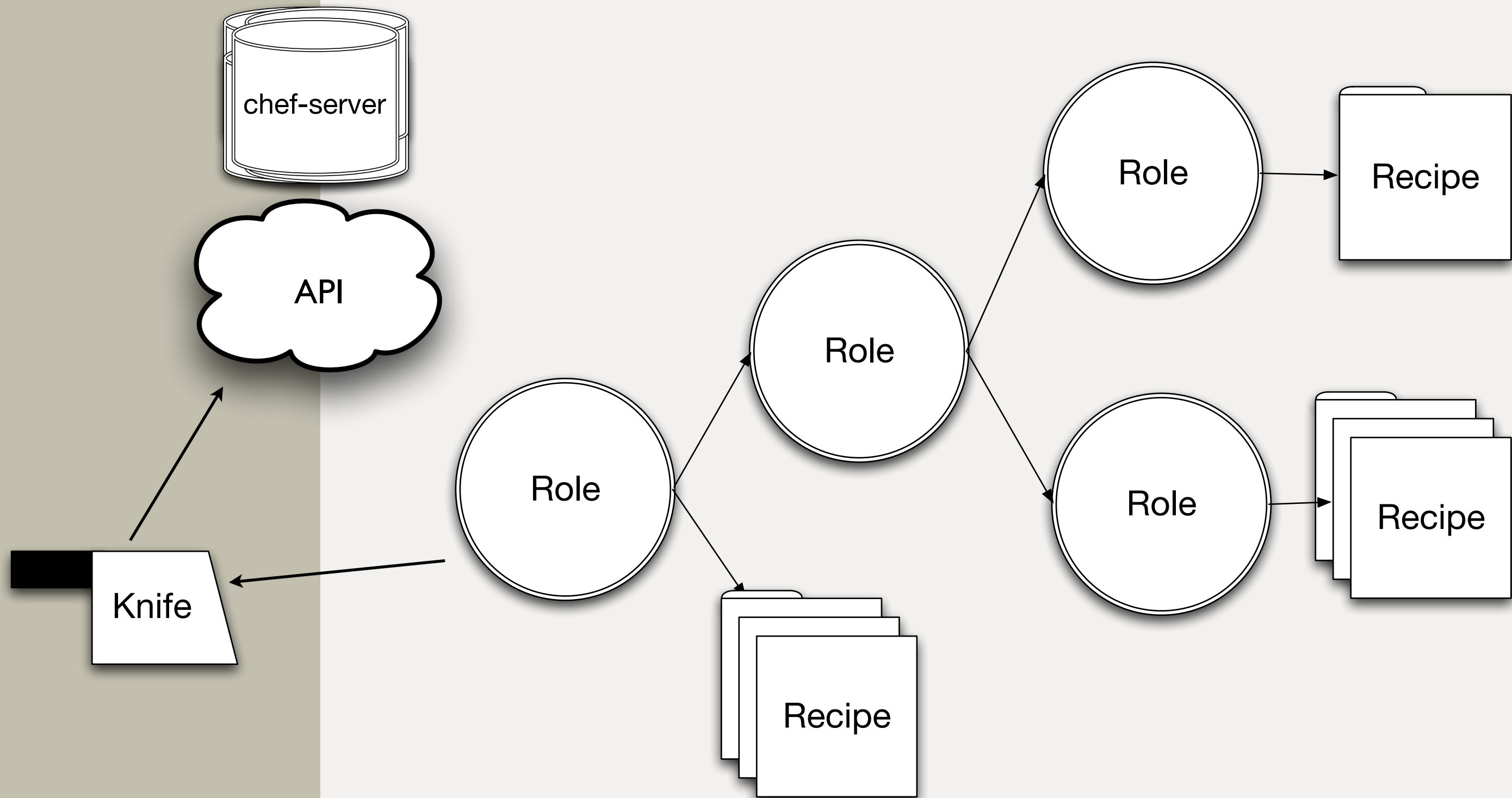


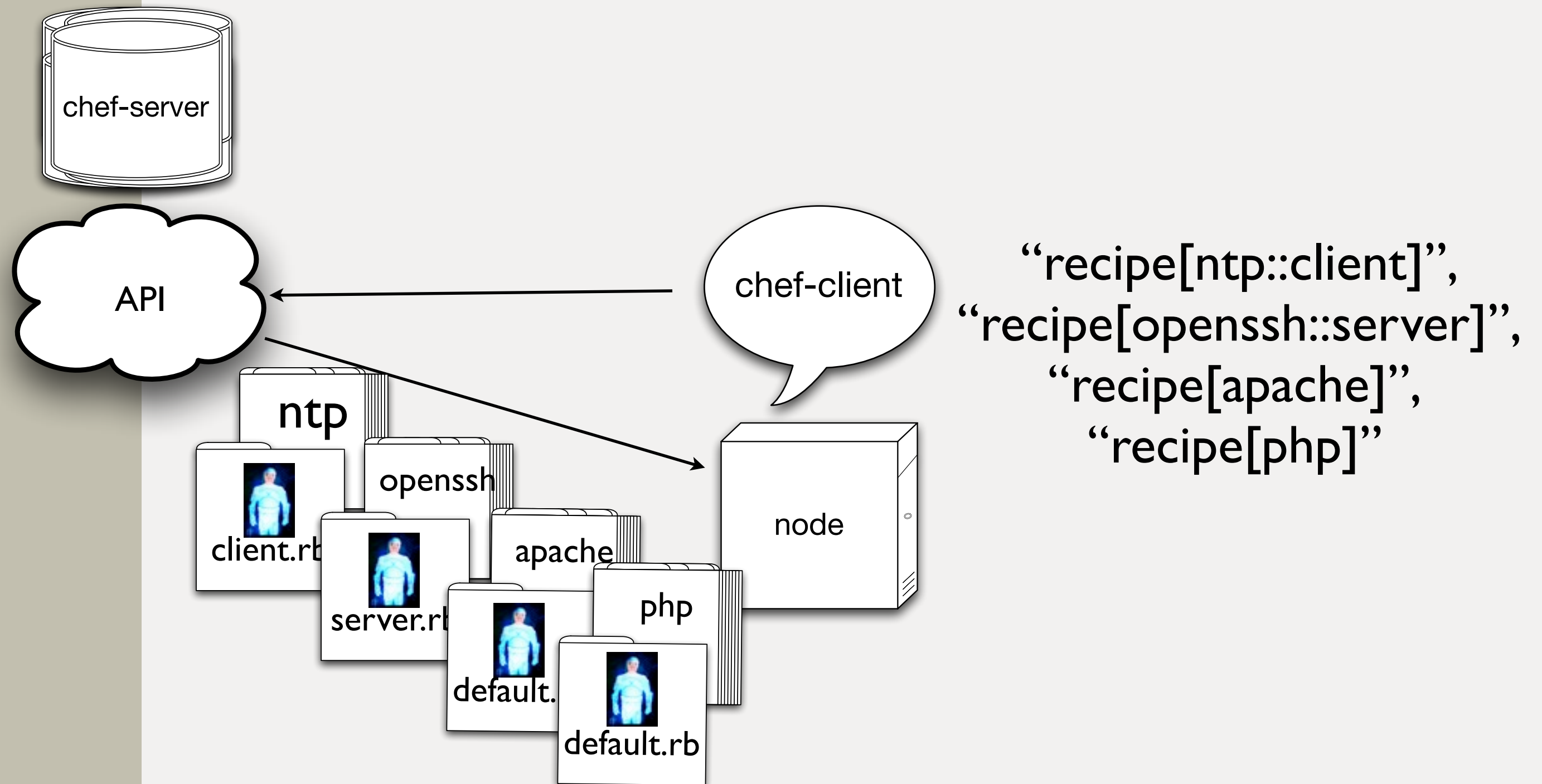
“ntp::client”,
“openssh::server”

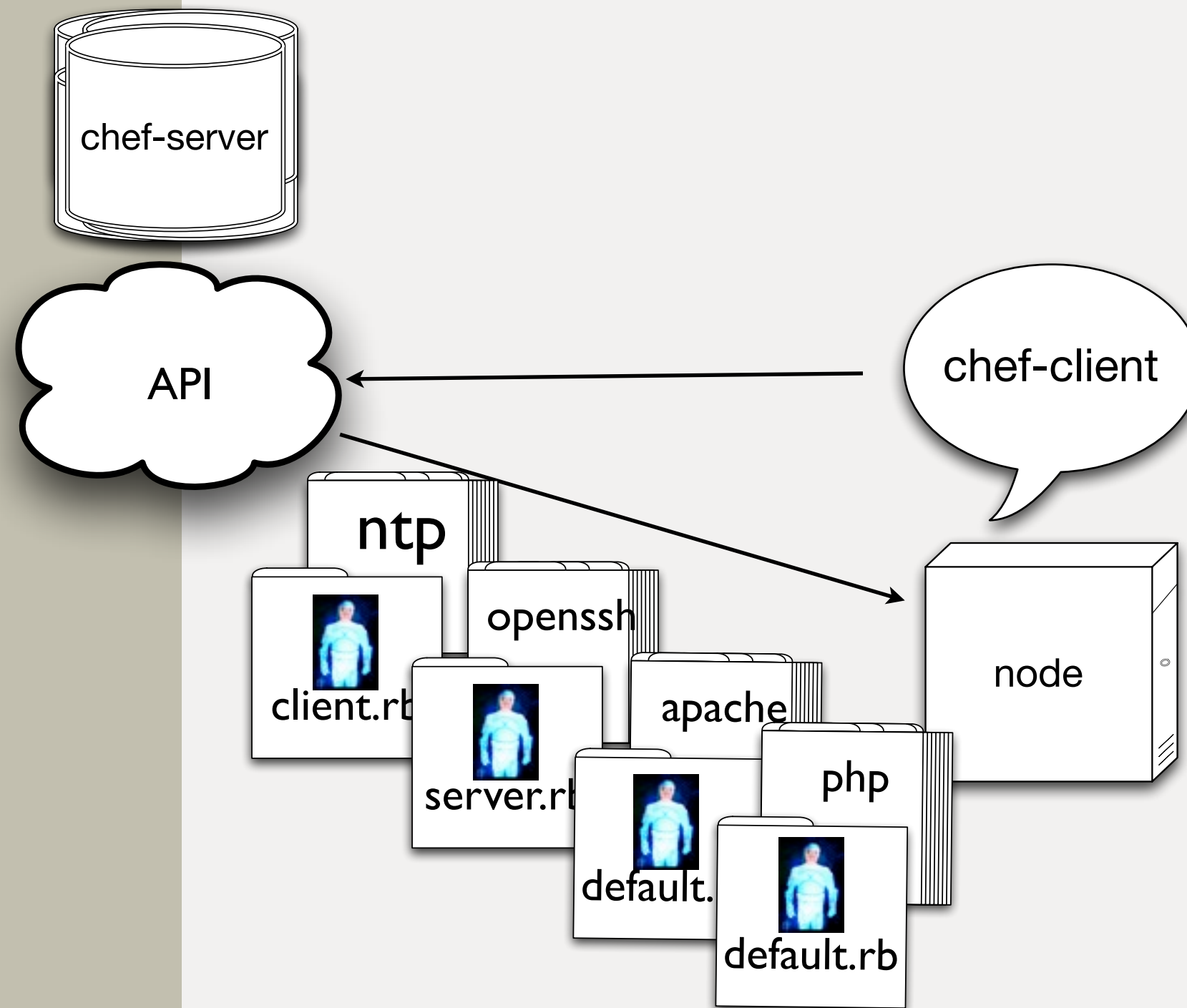


```
name "webserver"
description "webserver server"
run_list [
  "role[base]",
  "recipe[nginx::server]"
]
```

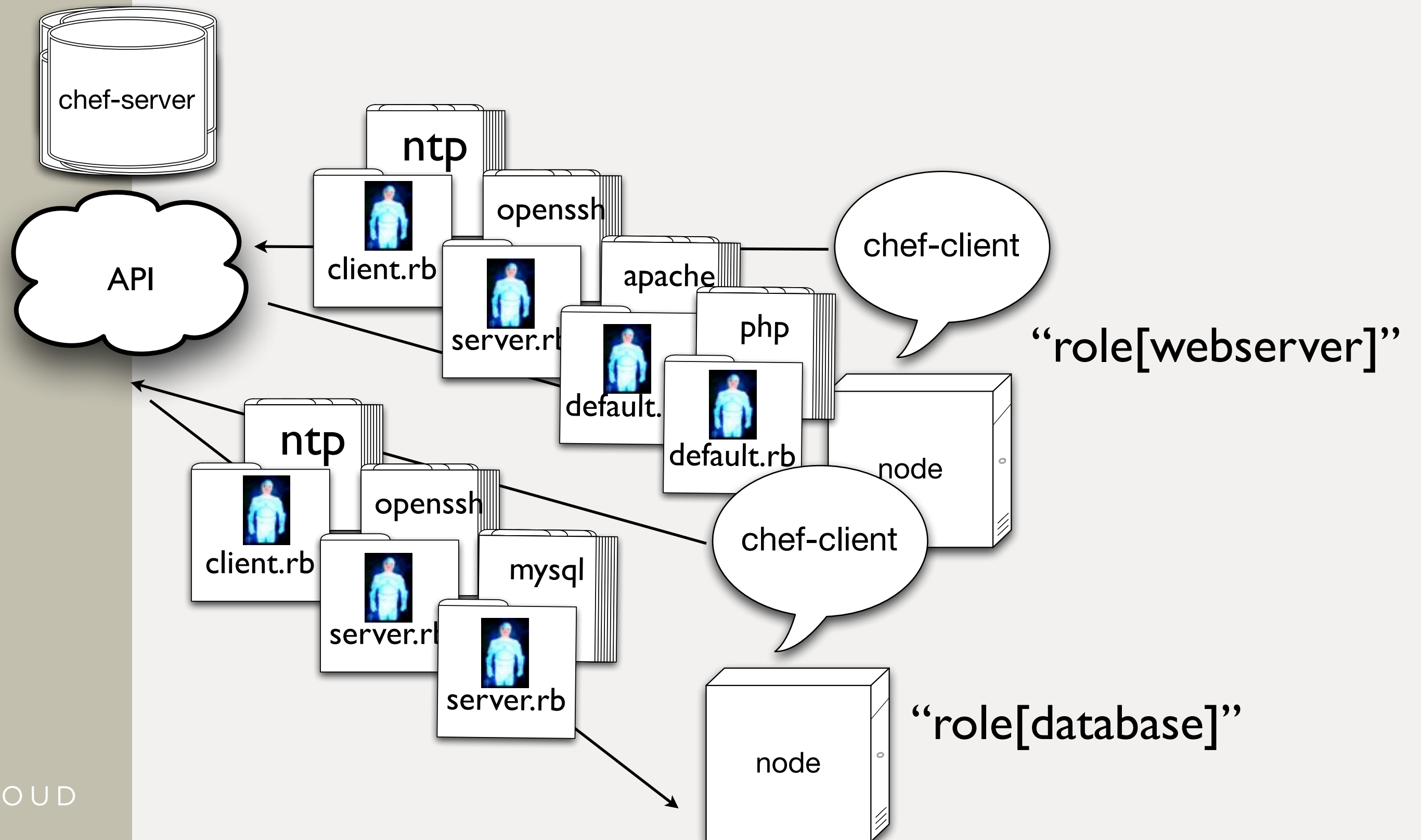
```
name "base"
description "base"
run_list [
  "recipe[selinux::disabled]",
  "recipe[etchosts]",
  "recipe[yum::epel]",
  "recipe[debugtools]"
]
```







“role[base]”,
“role[webserver]”





- Search for nodes with Roles
- Find configuration data

- IP addresses
- Hostnames
- FQDNs

<http://www.flickr.com/photos/kathycsus/2686772625>

```
pool_members = search("node", "role:webserver")

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy-app_lb.cfg.erb"
  owner "root"
  group "root"
  mode 0644
  variables :pool_members => pool_members.uniq
  notifies :restart, "service[haproxy]"
end
```

Pass results into Templates

```
# Set up application listeners here.  
listen application 0.0.0.0:80  
  balance roundrobin  
  <% @pool_members.each do |member| -%>  
    server <%= member[:hostname] %> <%= member[:ipaddress] %>: weight 1 maxconn 1 check  
  <% end -%>  
<% if node["haproxy"]["enable_admin"] -%>  
listen admin 0.0.0.0:22002  
  mode http  
  stats uri /  
<% end -%>
```

munin::server example

```
node.set[:munin][:server] = true
munin_clients = search(:node, "munin_client:true")

cookbook_file "/etc/cron.d/munin" do
  source "munin-cron"
  mode "0644"
  owner "root"
  group "root"
end

template "/etc/munin/munin.conf" do
  source "munin.conf.erb"
  mode 0644
  variables(:munin_clients => munin_clients)
end
```

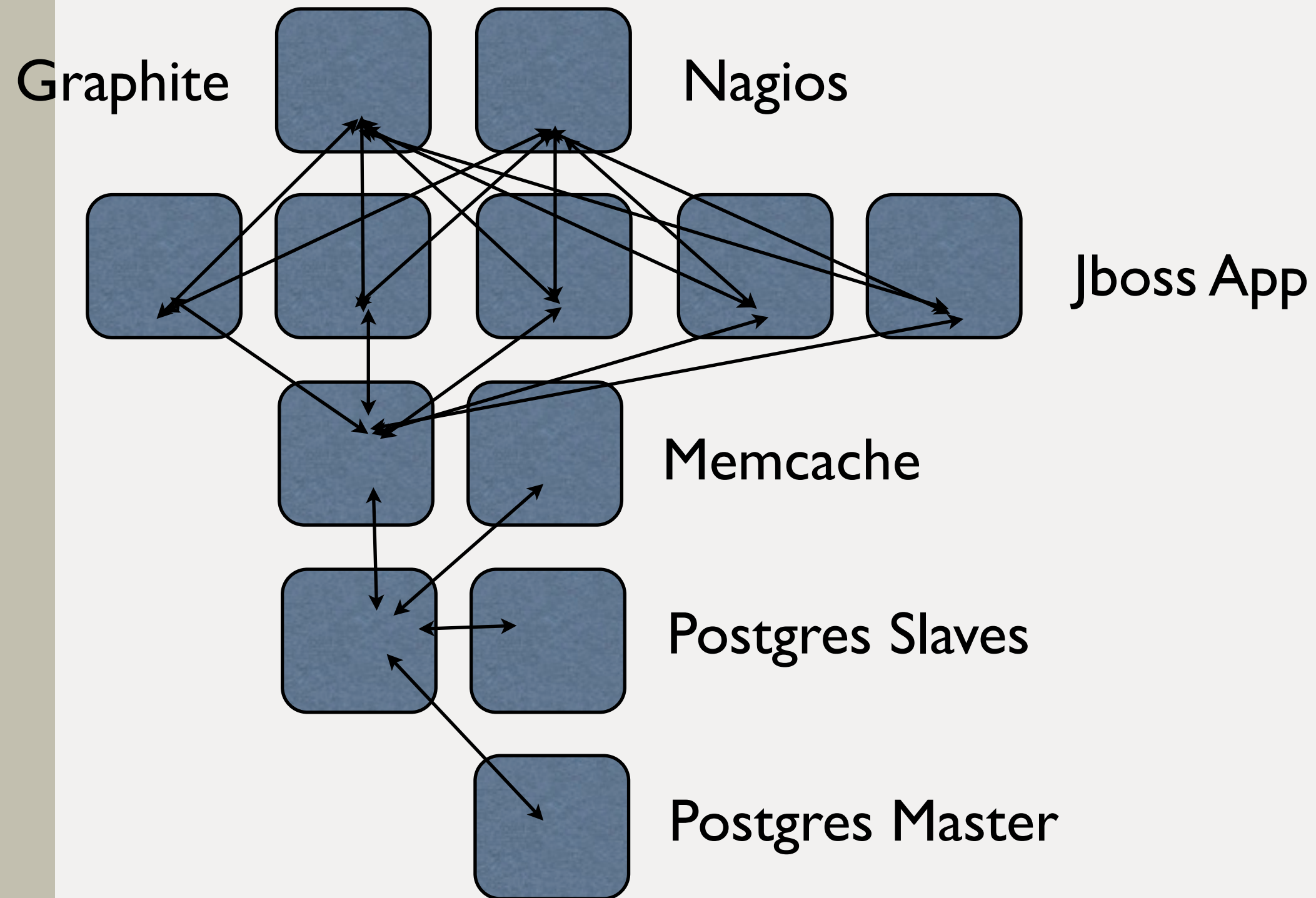
munin::client example

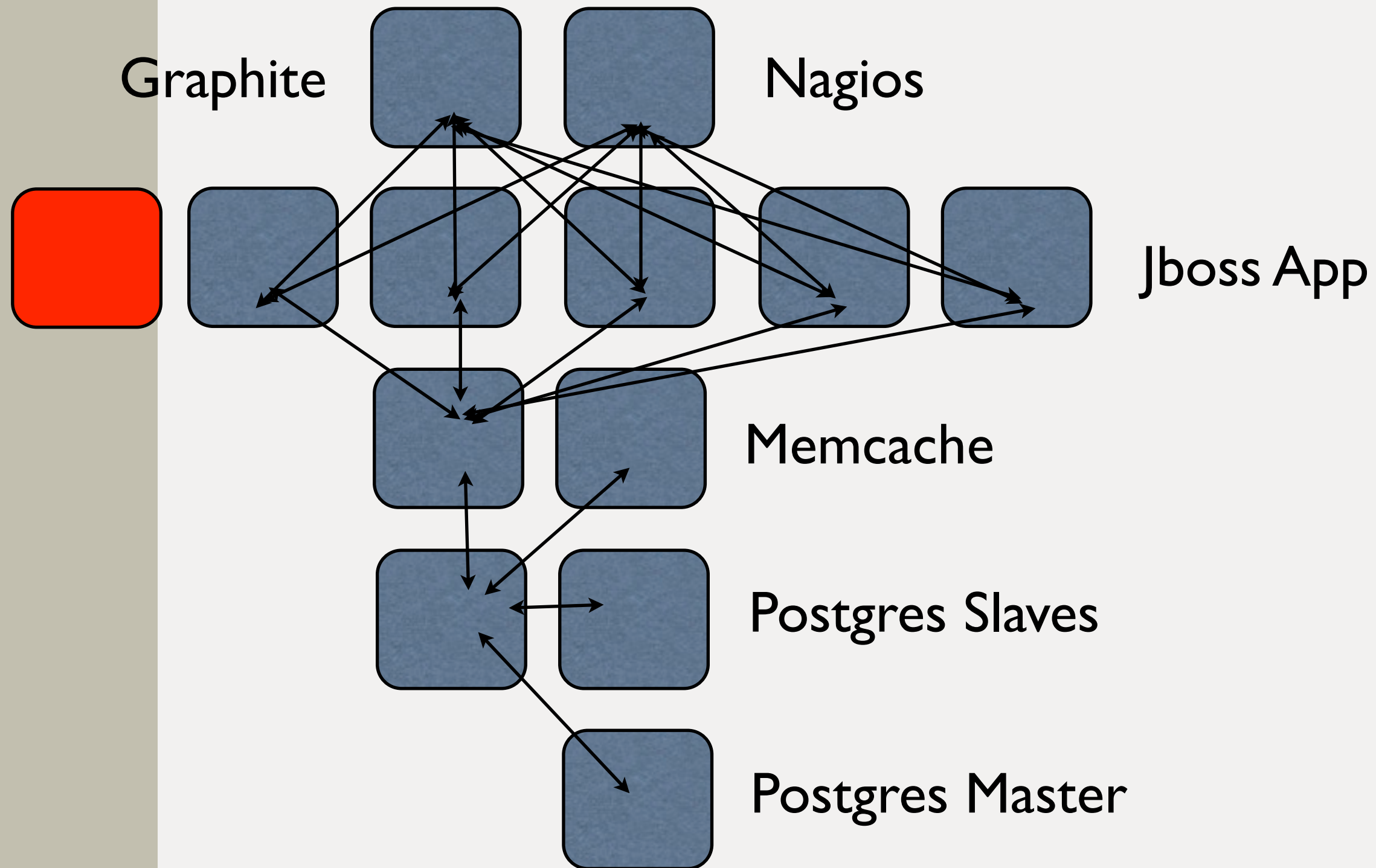
```
node.set[:munin][:client] = true
munin_servers = search(:node, "munin_server:true")

unless munin_servers.empty?
  package "munin-node" do
    action :install
  end

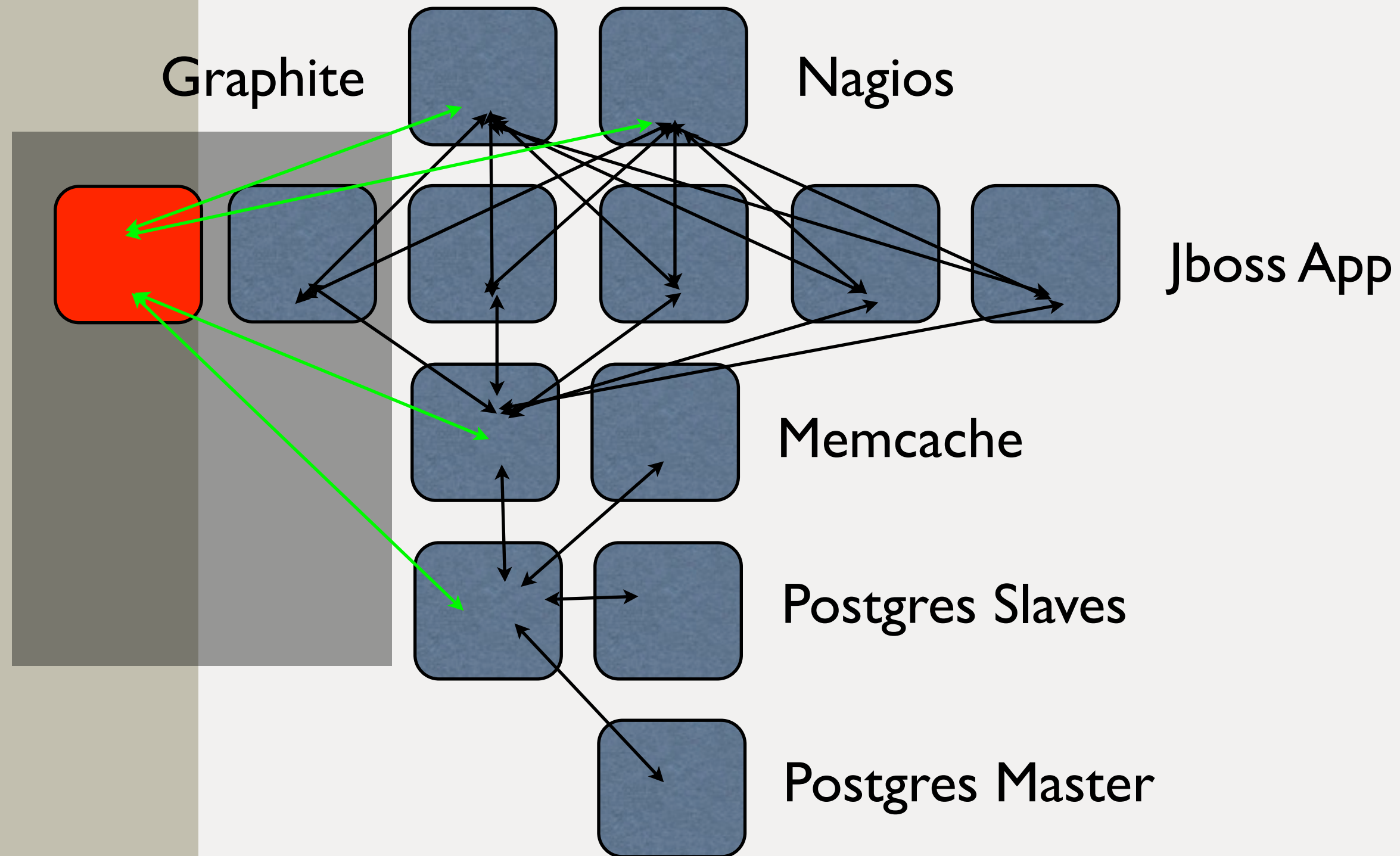
  template "/etc/munin/munin-node.conf" do
    source "munin-node.conf.erb"
    mode 0644
    variables :munin_servers => munin_servers
    notifies :restart, "service[munin-node]"
  end

  service "munin-node" do
    supports :restart => true
    action [ :enable, :start ]
  end
end
```

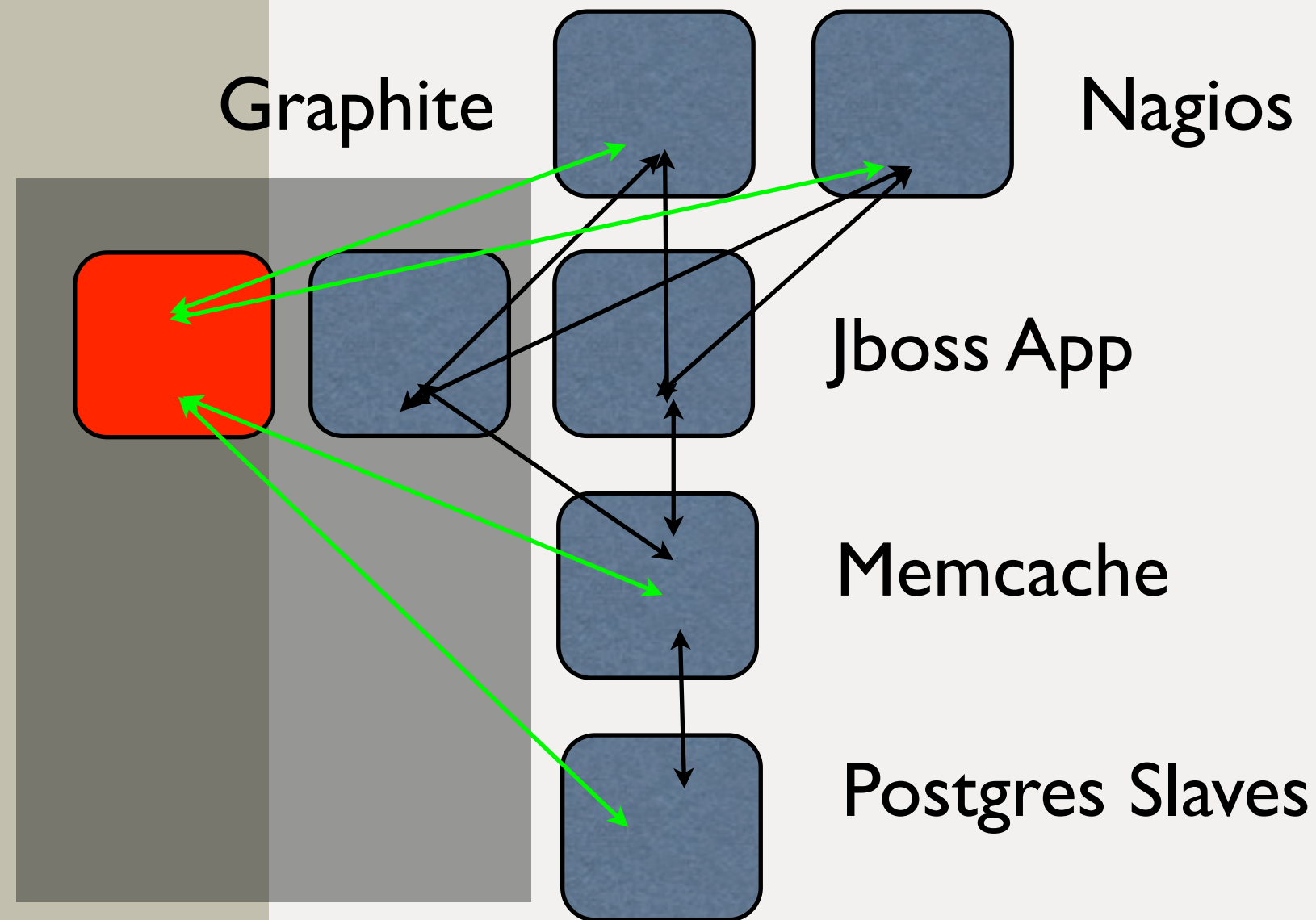




This can happen automatically



Count the resources



- 12+ resource changes for 1 node addition

- Load balancer config
- Nagios host ping
- Nagios host ssh
- Nagios host HTTP
- Nagios host app health
- Graphite CPU
- Graphite Memory
- Graphite Disk
- Graphite SNMP
- Memcache firewall
- Postgres firewall
- Postgres authZ config

CLONING CANNOT COPE WITH THIS



● Chef can.

<http://www.flickr.com/photos/evelynishere/2798236471/>



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

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

- Automatically reconfigure everything
- Load balancers
- Metrics collection systems
- Monitoring systems
- Whatever
- Cloud migrations become trivial

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



Questions?

sales@opscode.com
www.opscode.com

