# Complex Made Simple

## Sleep Better With **TorqueBox**

**Bob McWhirter**
Director of Polyglot
Red Hat

JBoss Community

# Complex Made Simple

## Sleep Better With **TorqueBox**

**Bob McWhirter**
Director of Polyglot
Red Hat

JBoss Community

# Who am I?

## Bob McWhirter

- Director of Polyglot, JBoss Fellow @ **Red Hat**
- **TorqueBox** project founder
- Part of the **project:odd** team
- The Codehaus, Drools, Jaxen, Groovy

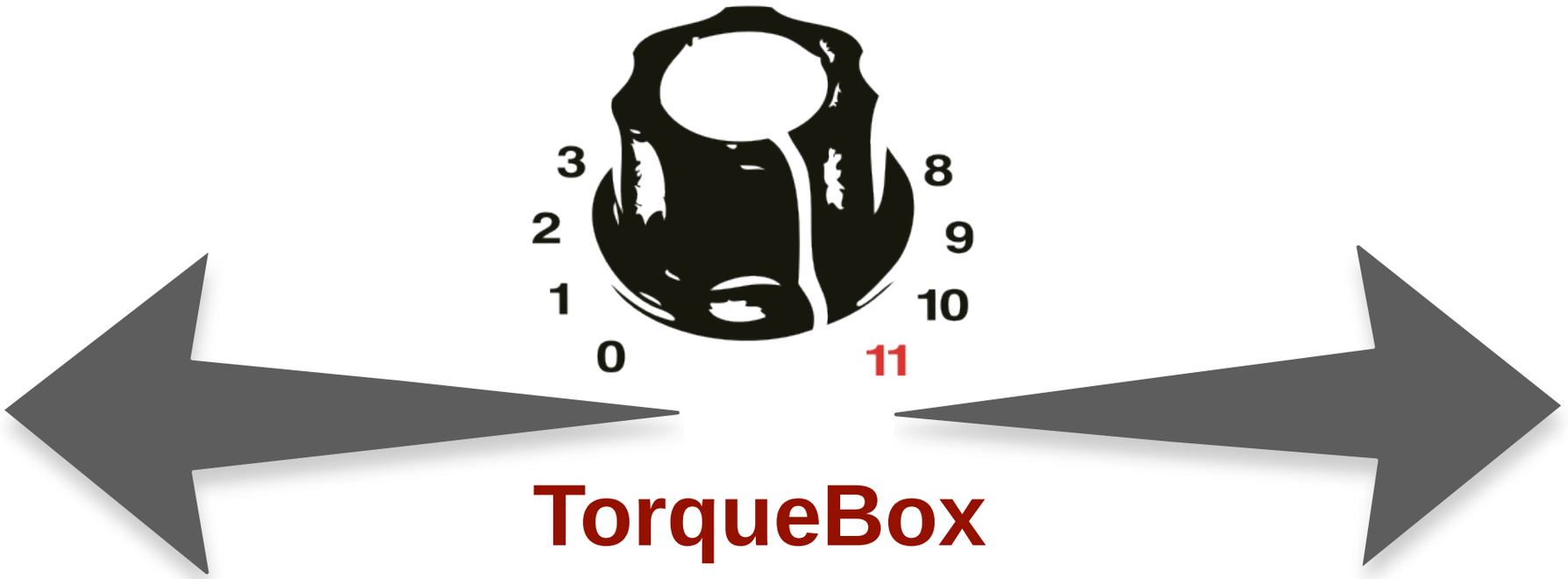
# Scenario

You've got a Rails application to deploy, for production.

# But...

The app you deploy **today** will grow in the **future**, if you're successful.

# Deploy Continuum




**TorqueBox**

**Roll your own**

**Platform as a Service**

# What is TorqueBox?

**TorqueBox** is a
Ruby Application Server.

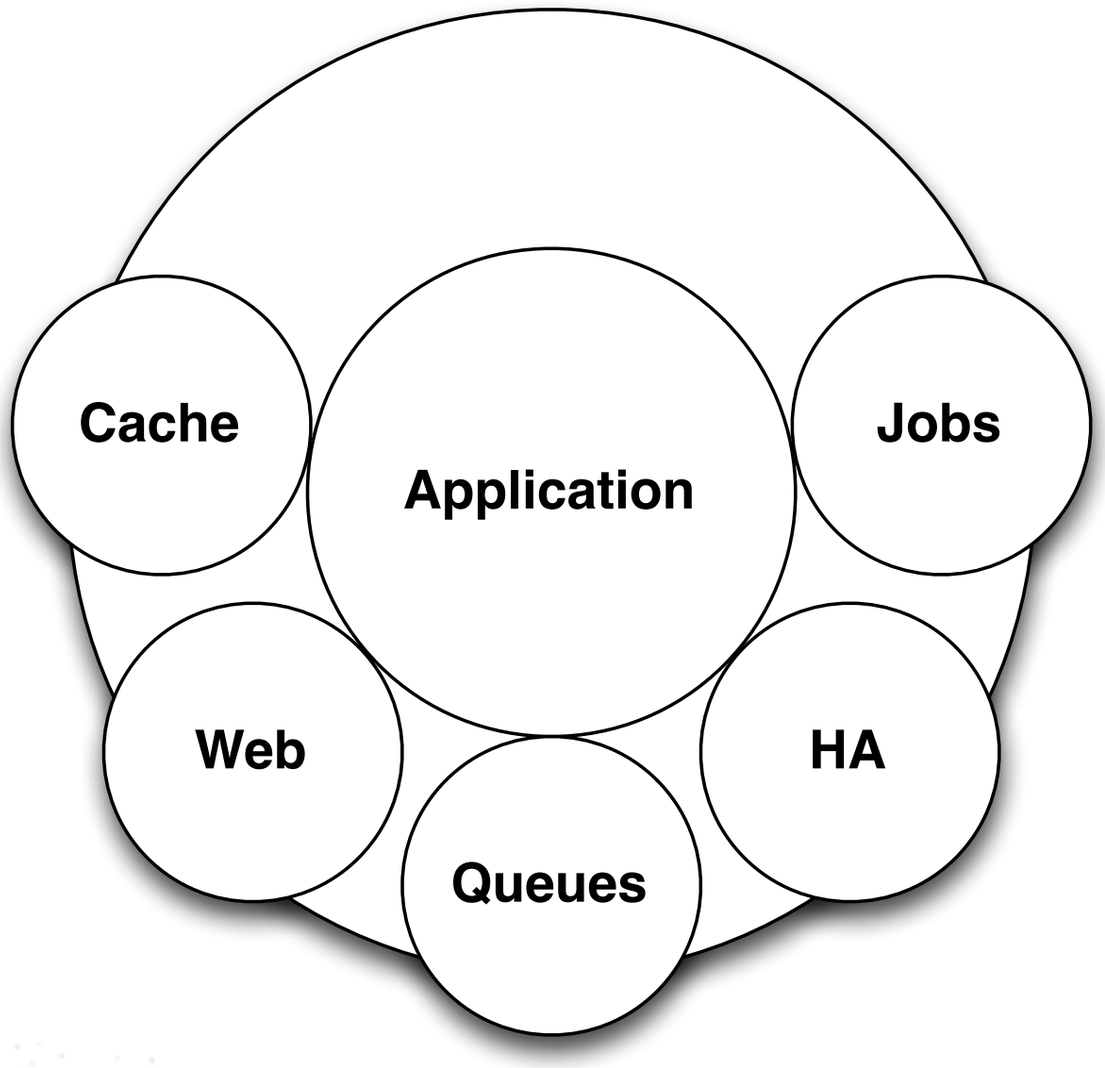
Based on JRuby and
JBoss AS7.

# What is an app-server?

*If you're not from the Java world...*

An application server is a process that **hosts your app** and **provides a multitude of services** and facilities.

# App-servertastic!



# Anyhow...

# Roll your own

- Install OS
- Install packages
- Configure everything
- Manage everything

# Roll your own

- Apache httpd
- Load balancer
- Unicorn, pack of Mongrels
- crontab
- SMTP
- memcached
- Database
- deployment
- monitoring

# Or outsource some...

# Roll your own >>

- Install OS
- Install packages
- Configure most things
- Manage most things
- Outsource some things

# Roll your own >>

- Apache httpd
- Load balancer
- Unicorn, pack of Mongrels
- crontab
- ~~SMTP~~ **Amazon SES**, **SendGrid**
- memcached
- Database
- ~~deployment~~ **Capistrano**
- ~~monitoring~~ **New Relic**

# Use a Paas (Heroku)

Outsource **everything** but your app.

# Heroku

- ~~Apache httpd~~ **Heroku**
- ~~Load balancer~~ **Heroku**
- ~~Unicorn, pack of Mongrels~~ **Heroku**
- ~~crontab~~ **Heroku**
- ~~SMTP~~ **SendGrid**
- ~~memcached~~ **Heroku**
- ~~Database~~ **Heroku**
- ~~deployment~~ **Heroku**
- ~~monitoring~~ **New Relic**

# Or middle-ground

- **Decrease** number of things to manage
- Increase **scalability**
- Increase **availability** (optionally)

# TorqueBox

- Apache httpd
- ~~Load balancer~~ **JBoss mod_cluster**
- ~~Unicorn, pack of Mongrels~~ **TorqueBox**
- ~~crontab~~ **TorqueBox**
- ~~SMTP~~ **Amazon SES**, **SendGrid**
- ~~memcached~~ **TorqueBox**
- Database
- ~~deployment~~ **Capistrano**
- ~~monitoring~~ **NewRelic**

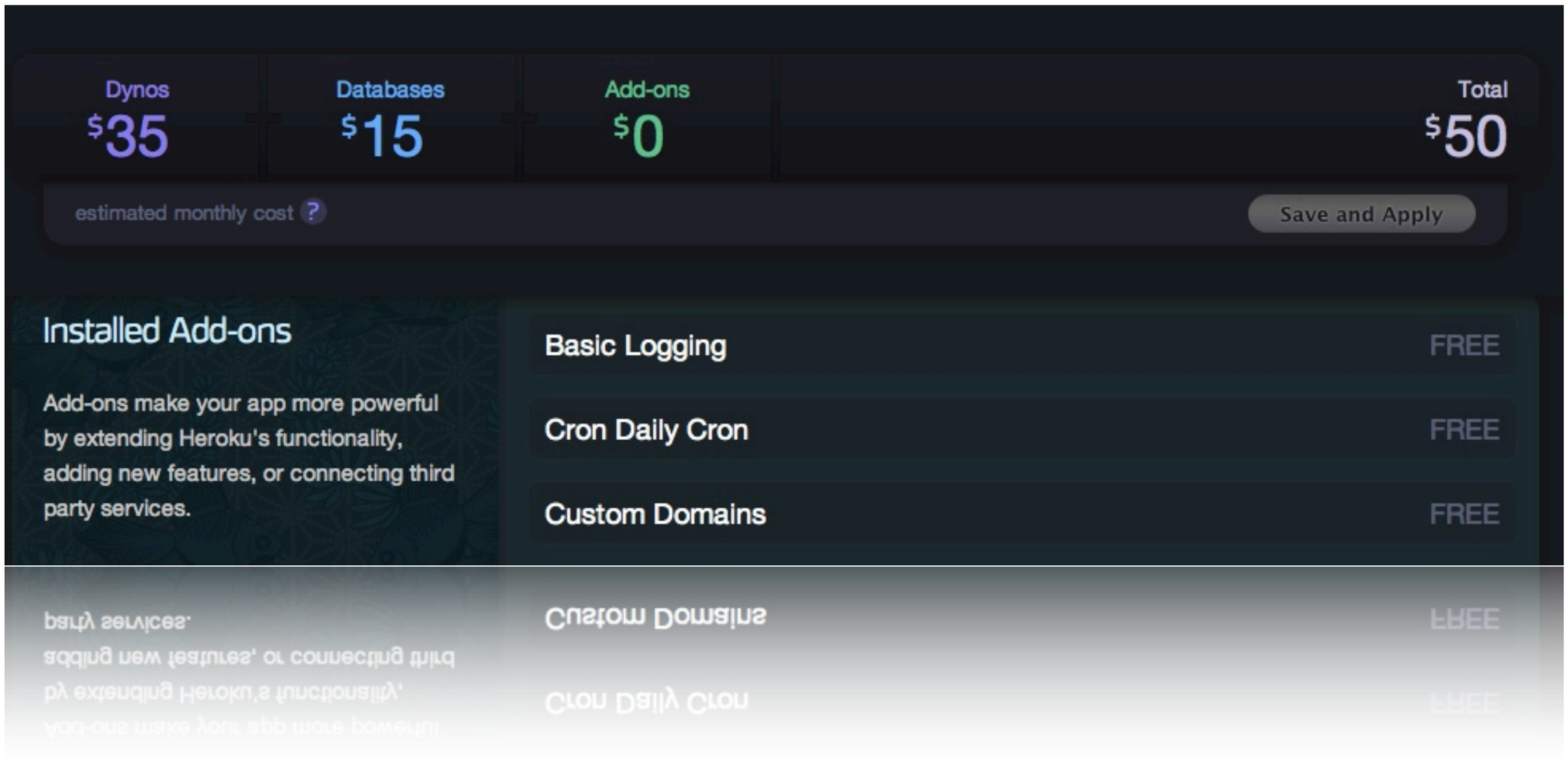
# Case Study!

# Appalachian Sustainable Agriculture Project (ASAP)

# Application Needs

- Rails 2.x web app
- Caching
- Background tasks
- Cron jobs
- Deployment from SCM
- Sending email
- Database queries
- Monitoring & metrics

# As deployed on Heroku



# Migration Motivation

In ***this particular case***, client feels it could save money, and have a better system by moving to an affordable VPS, letting **TorqueBox** absorb some of the roll-your-own complexity.

# Migrating to TorqueBox

```
$ rvm install jruby-1.6.7
```

# Migrating to TorqueBox

```
$ gem install torquebox-server \
  --pre --version=2.0.0.cr1
```

*Simpler once we're no longer --pre*

# Using torquebox.rb template

```
$ cd myapp
$ torquebox rails
```

# torquebox.rb template

- Add torquebox gems to Gemfile
- Add **activerecord-jdbc-adapter**
- Add **torquebox-rake-support**
- Create directories for **services**, **jobs**, **etc**
- Initialize **web session** store
- Initialize **Rails.cache** with TorqueBox
- Initialize ActiveRecord for **background jobs**

# “Porting” the application

# Delayed::Job

App uses
**`Delayed::Job`**

Replace with TorqueBox
**`Backgroundable`**

# Delayed Job

**controller.rb**

```
def create
  @export = ExcelExport.create(...)
  job = ExcelExportJob.new
  job.excel_export_id = @export.id
  Delayed::Job.enqueue job
  @export.set_status "Queued for Export"
  @export.save!
end
```

# Delayed Job

## excel_export_job.rb

```
class ExcelExportJob < Struct.new(:excel_export_id)
  def perform
    ExcelExport.find(self.excel_export_id).generate_report
  end
end
```

# Delayed Job

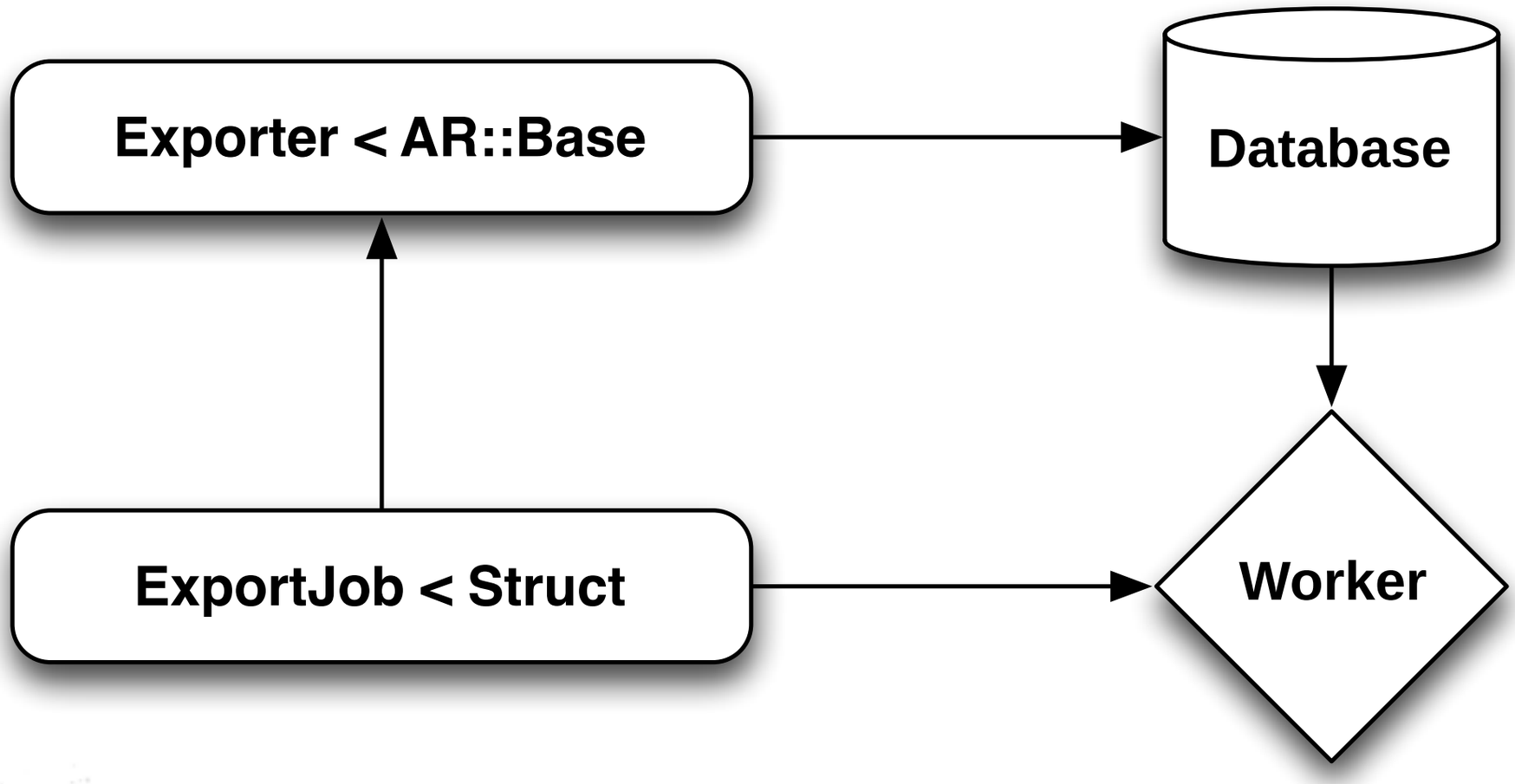
## excel_export.rb

```
class ExcelExport < ActiveRecord::Base
  def generate_report
    begin
      set_status 'Processing'
      spreadsheet = Spreadsheet.new(self.businesses)

      set_status 'Writing'
      spreadsheet.write_workbook self.workbook_file

      set_status 'Storing on Amazon S3'
      File.open(self.workbook_file) {|out|self.output = out}

      set_status 'Cleaning Up'
      File.delete(self.workbook_file)

      set_status 'Complete'
    rescue
      set_status 'Export Error'
    end
  end
```

# Delayed Job

Mayhap you need to run a **worker**, also...

# Delayed::Job



# That's pretty explicit

**TorqueBox** allows you to very easily run methods **asynchronously**.

# Backgroundable

## excel_export.rb

```
class ExcelExport < ActiveRecord::Base
  always_background :generate_report

  def generate_report
    begin
      set_status 'Processing'
      ...
    rescue
      set_status 'Export Error'
    end
  end

end
```

# Backgroundable

## excel_export.rb

```
class ExcelExport < ActiveRecord::Base
  always_background :generate_report

  def generate_report
    begin
      set_status 'Processing'
      ...
    rescue
      set_status 'Export Error'
    end
  end

end
```

# Backgroundable

**export_controller.rb**

```
def create
  @export = ExcelExport.create(...)
  @export.set_status "Queued for Export"
  @export.save
  @export.generate_report
end
```

# Backgroundable

## excel_export_job.rb

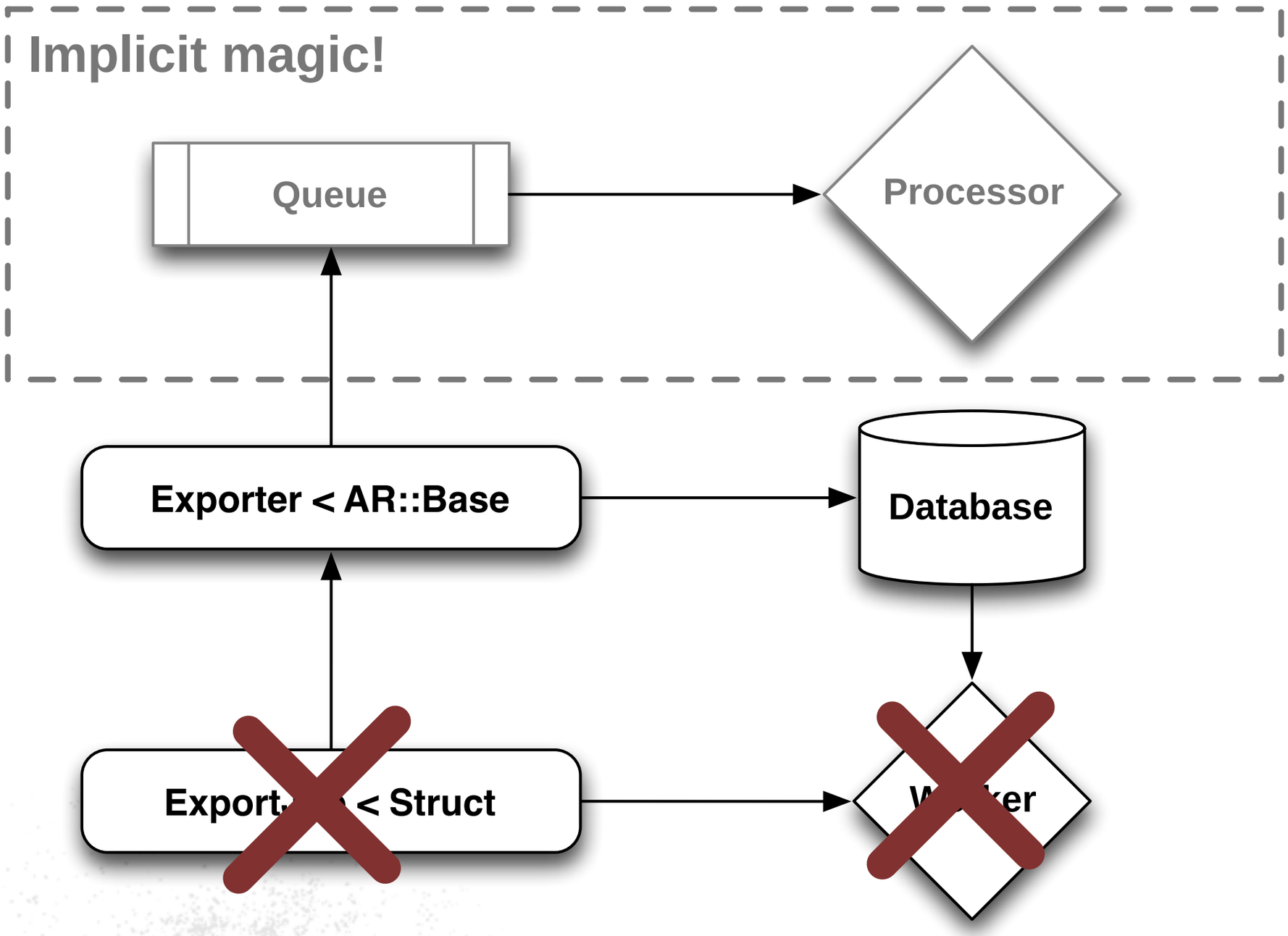
```
$ rm excel_export_job.rb
```

Kill it!

# Trade!

Trade **some code** and **a process** you have to manage for **less code** and an **in-container service**.

# Backgroundable



# Caching

- **template.rb** does most of the work
  - Web sessions
  - Fragment and other caching

# *No more*

# memcached!

# Scheduled Jobs

- Replaces cron jobs
- Deploys with your application
- Includes application environment
- Cron-like scheduling
- HA-clustering (optionally)

# Heroku Cron

- Free version runs 1x daily
- Implemented as a rake task
- Includes application environment

# Roll your own

- crontab
- `script/runner` (boots rails every time)
- deployment strategy
- source control
- etc.

# On Heroku

## Rakefile

```
desc "This task is called by the heroku cron add-on"
task :cron => [:environment,
               :heroku_backup,
               :refresh_trip_planner]

desc "Refresh the trip planner business data cache"
task :refresh_trip_planner => :environment do
  #
  # update GIS cache
  #
end
```

# Scheduled Jobs

In TorqueBox, a scheduled job is a **component** of **your application**.

# Straight-forward

refresh_trip_planner.rb

```
class RefreshTripPlanner
  def run()
    #
    # update the GIS cache
    #
  end
end
```

# Configuration

## torquebox.yml

```
jobs:
  cache.updater:
    job:         RefreshTripPlanner
    cron:        '0 0 2 * * ? '
    description: Refresh trip cache
```

# But...

Doing a huge query **once-a-day** isn't ideal, it was just "free".

# TorqueBox Services

- Avoid the huge query
- Keep data more fresher

# Keep it fresher

## trip_planner_service.rb

```
TripPlannerService
 def initialize( options={} )
   @queue = TorqueBox::Messaging::Queue.new( '/queues/trip_planner' )
 end

 def start
   Thread.new do
     initialize_cache
     while should_run
       update_cache @queue.receive
     end
   end
 end

 def update_cache( business_id )
   TripPlanner.insert_or_update business_id
 end
 # initialize_cache(), stop(), etc
end
```

# Keep it fresher

## trip_planner_service.rb

```
def initialize( options={} )
  @queue = TorqueBox::Messaging::Queue.new( ... )
end
```

```
      initialize_cache
      while should_run
        update_cache @queue.receive
      end
    end
  end

  def update_cache( business_id )
    TripPlanner.insert_or_update business_id
  end
  # initialize_cache(), stop(), etc
end
```

# Keep it fresher

## trip_planner_service.rb

```
TripPlann
 def init
   @queue
 end

 def star
   Thread
     init
     whil
       up
     end
   end
 end

 def upd
   TripP
 end
 # initialize_cache(), stop(), etc
end
```

```
def start
  Thread.new do
    initialize_cache
    while should_run
      update_cache @queue.receive
    end
  end
end
```

# Keep it fresher

## app/models/business.rb

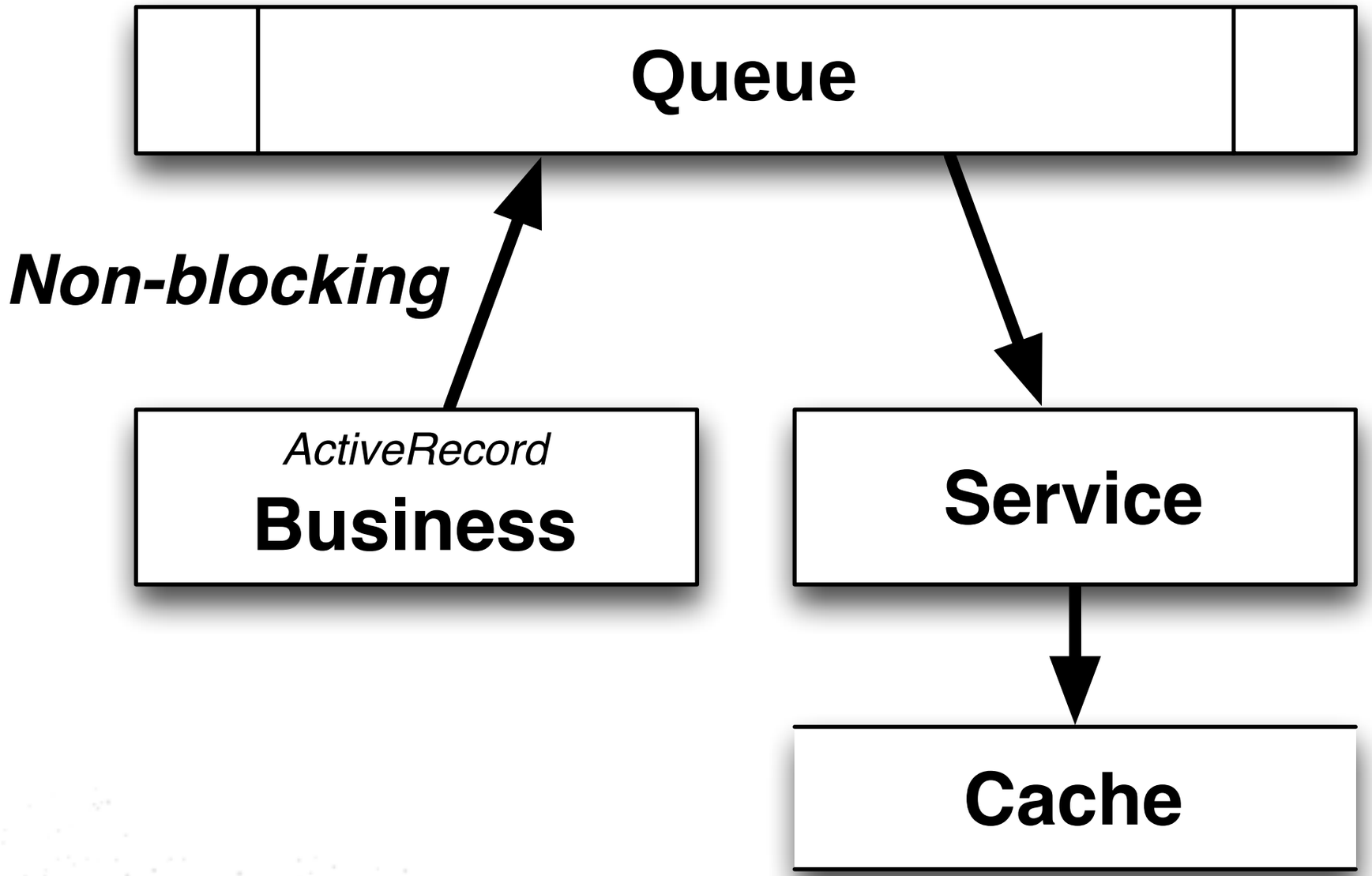
```
require 'torquebox-messaging'

class Business < ActiveRecord::Base

  after_save :update_trip_planner

  def update_trip_planner
    queue = TorqueBox::Messaging::Queue.new('/queues/trip_planner')
    queue.publish( self.id )
  end

end
```

# Keep it fresher

## app/models/business.rb

```
require 'torquebox-messaging'

class Business < ActiveRecord::Base

  after_save :update_trip_planner

  d
    queue = TorqueBox::Messaging::Queue.new(...)
    queue.publish( self.id )
  e

end
```

# Queue, Service, Cache



# Production!


JBoss Community


(Remember, we work for **Red Hat**)

# Set up the server

```
$ yum install \
      java-1.6.0-openjdk \
      httpd \
      mod_cluster \
      git \
      postgresql-server
```

# Capistrano

Regular **capistrano** deployments are supported given the recipes provided by **`torquebox-capistrano-support.gem`**

# Capistrano

## deploy.rb

```
require 'torquebox-capistrano-support'
require 'bundler/capistrano'

# source code
set :application,       "buyappalachian.org"
set :repository,        "git@github.com:account/repo.git"
set :branch,            "torquebox-2.0"
set :user,              "torquebox"
set :scm,               :git
set :scm_verbose,       true
set :use_sudo,          false
```

# Capistrano

## deploy.rb (continued)

```
# Production server
set :deploy_to,           "/opt/apps/buyappalachian.org"
set :torquebox_home,      "/opt/torquebox/current"
set :jboss_init_script,   "/etc/init.d/jboss-as-standalone"
set :app_environment,     "RAILS_ENV: production"
set :app_context,         "/"

ssh_options[:forward_agent] = false

role :web, "torquebox.buyappalachian.org"
role :app, "torquebox.buyappalachian.org"
role :db,  "torquebox.buyappalachian.org",
     :primary => true
```

# Launch on boot

# init.d

Install
**`/etc/init.d/jboss-as`**
from the stock distribution

# JBoss configuration

## /etc/jboss-as/jboss-as.conf

```
# General configuration for the init.d script
# Place this file in /etc/jboss-as/jboss-as.conf
# and copy $JBOSS_HOME/bin/init.d/jboss-as-standalone.sh
# to /etc/init.d/jboss-as-standalone

JBOSS_USER=torquebox
JBOSS_HOME=/opt/torquebox/current/jboss
JBOSS_PIDFILE=/var/run/torquebox/torquebox.pid
JBOSS_CONSOLE_LOG=/var/log/torquebox/console.log
JBOSS_CONFIG=standalone-ha.xml
```

# Load Balance with mod_cluster

JBoss Community

# mod_cluster

## httpd.conf

```
LoadModule slotmem_module        modules/mod_slotmem.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module      modules/mod_advertise.so
LoadModule manager_module        modules/mod_manager.so

<Location /mod_cluster_manager>
    SetHandler mod_cluster-manager
    AllowDisplay On
</Location>

Listen torquebox-balancer:6666
```

# mod_cluster

## httpd.conf (continued)

```
<VirtualHost torquebox-balancer:6666>

  <Directory />
    Order deny,allow
    Deny from all
    Allow from all
  </Directory>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0

  EnableMCPMReceive

  ManagerBalancerName torquebox-balancer
  AllowDisplay On
  AdvertiseFrequency 5
  AdvertiseSecurityKey secret

</VirtualHost>
```

# Deployed

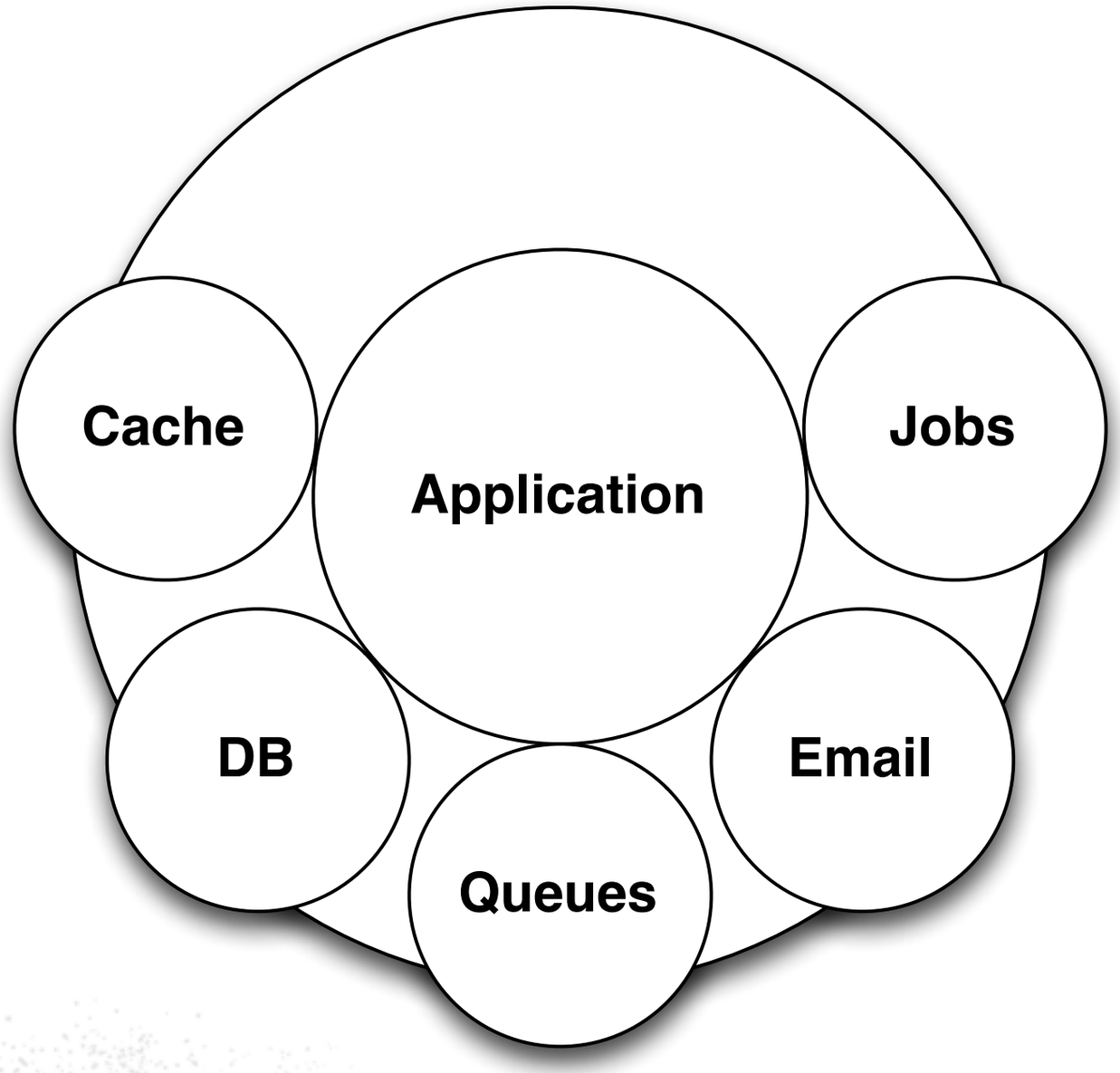
```
[root@torquebox opt]# ls -l apps/buyappalachian.org/
total 8
lrwxrwxrwx 1   23 Mar 16 15:10 current -> releases/20120315230553
drwxrwxr-x 5 4096 Mar 15 19:05 releases
drwxrwxr-x 6 4096 Mar 15 17:29 shared
```

# And then it grows...

As your app **evolves**, you might add stuff like **WebSockets**/STOMP, or **HA** or other facilities already in the tin.

# Can we go further?

# What's a PaaS?



# TorqueBox on OpenShift

openshift.redhat.com

# Roadmap & Status

- Current **2.0.0.CR1**
- **2.0.0.Final** Real Flippin' Soon
- Continuing investment from Red Hat
- Interop with other languages, such as **Clojure** via **Immutant**

# Resources

http://torquebox.org/

@torquebox on Twitter

#torquebox on IRC

# Question everything.

# JBoss Community