TorqueBox

The Beauty of Ruby with the Power of Java

Toby Crawley

OSCON Java
July 2011

Toby Crawley

- @tcrawley
- TorqueBox Core Developer
- Red Hat Senior Engineer

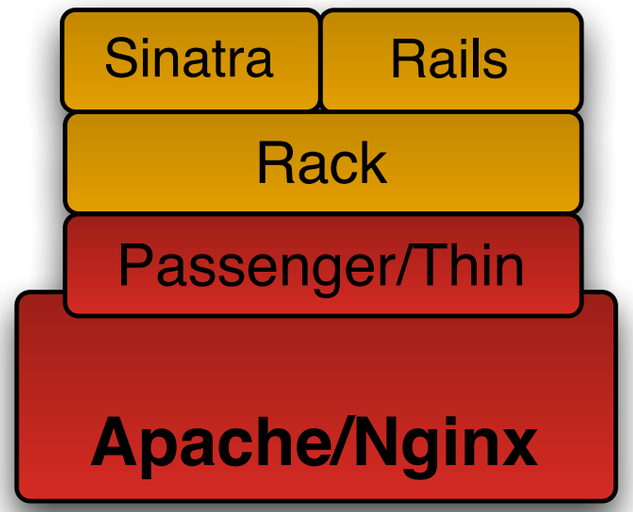
but it's a team.

What is TorqueBox?

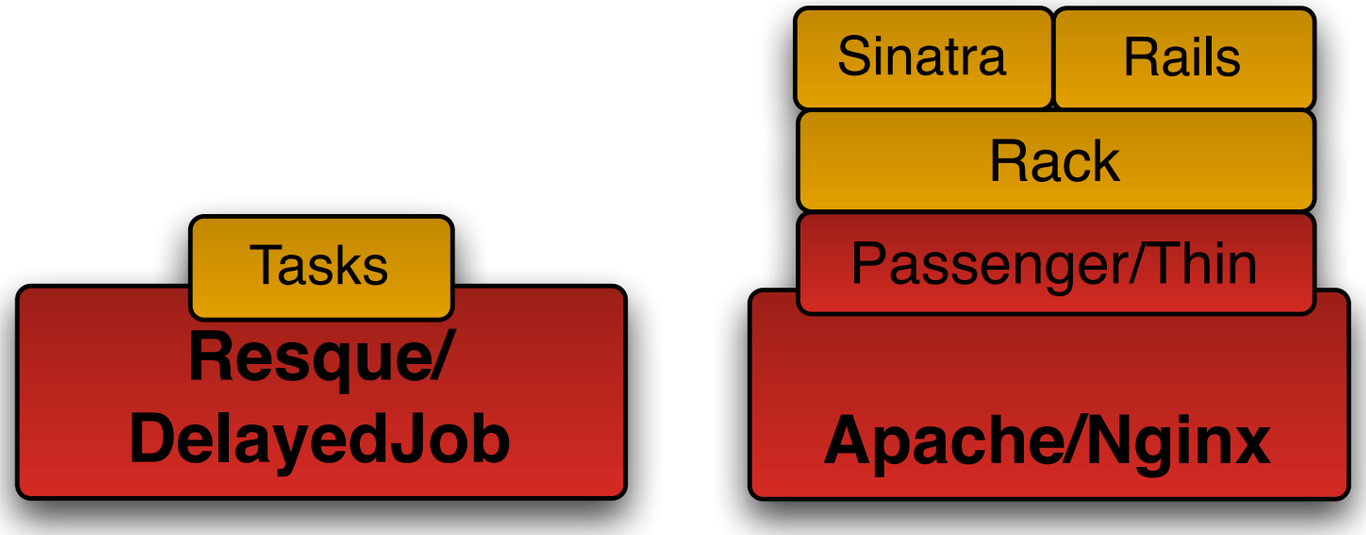
The mating of **JRuby** to **JBoss AS**.

Target Market

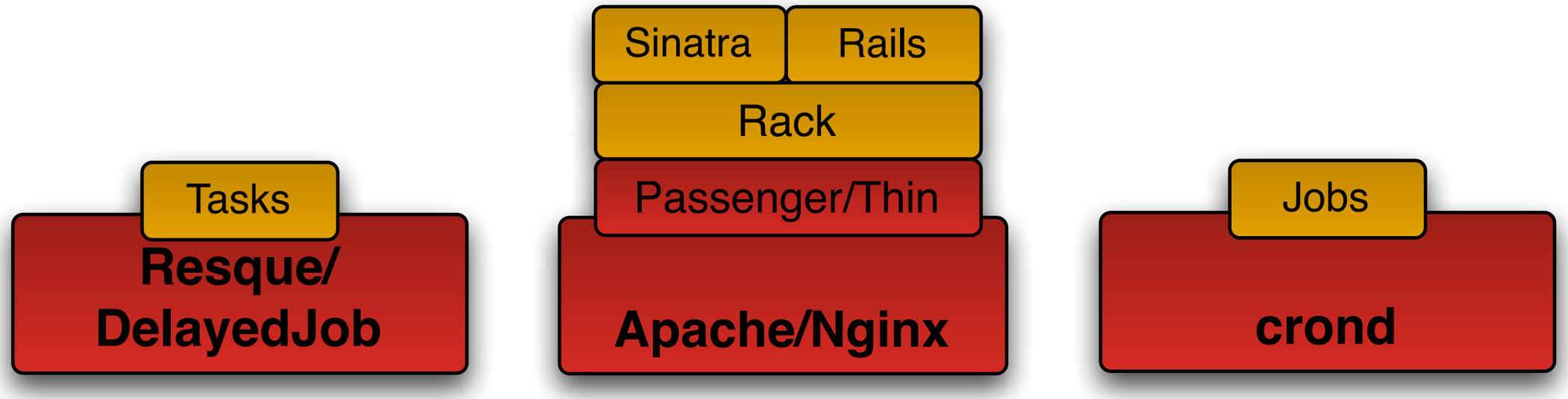
Application Server?



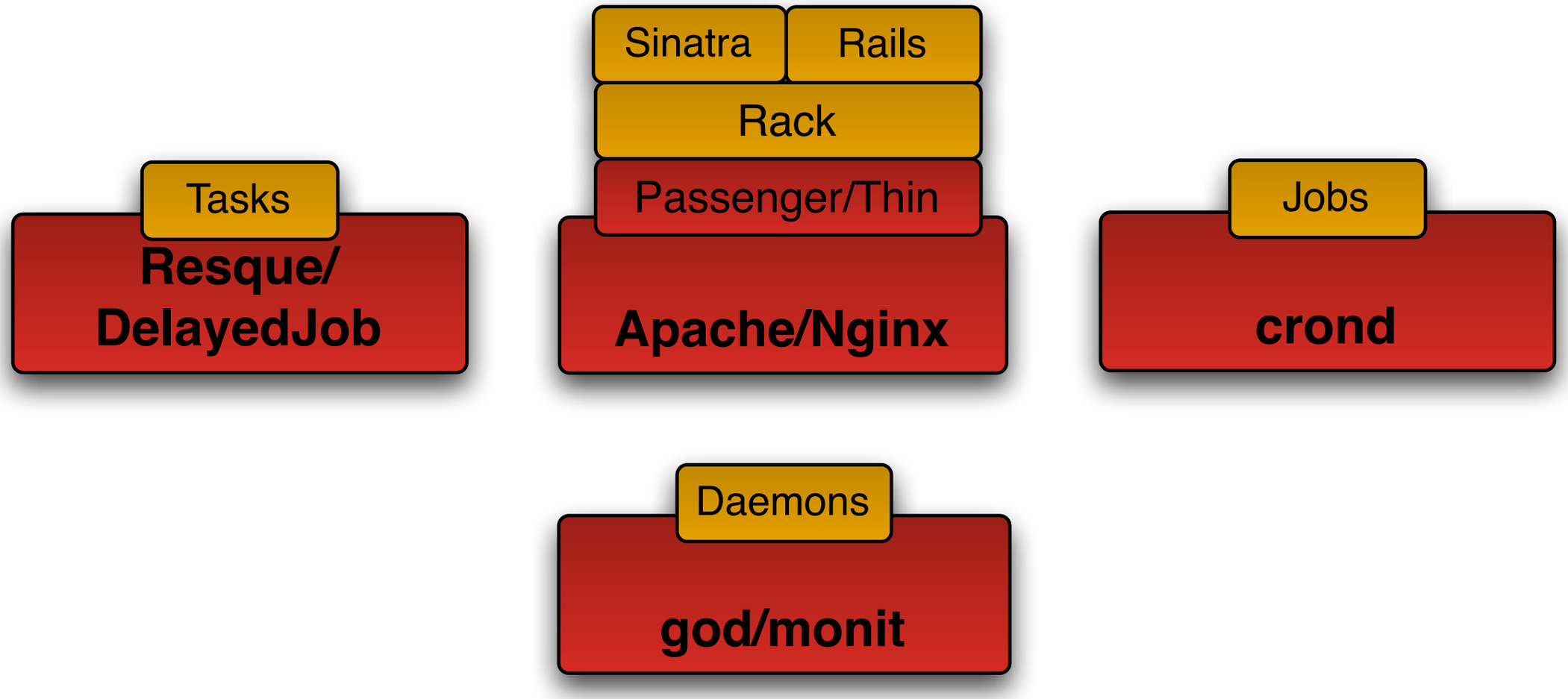
Application Server?



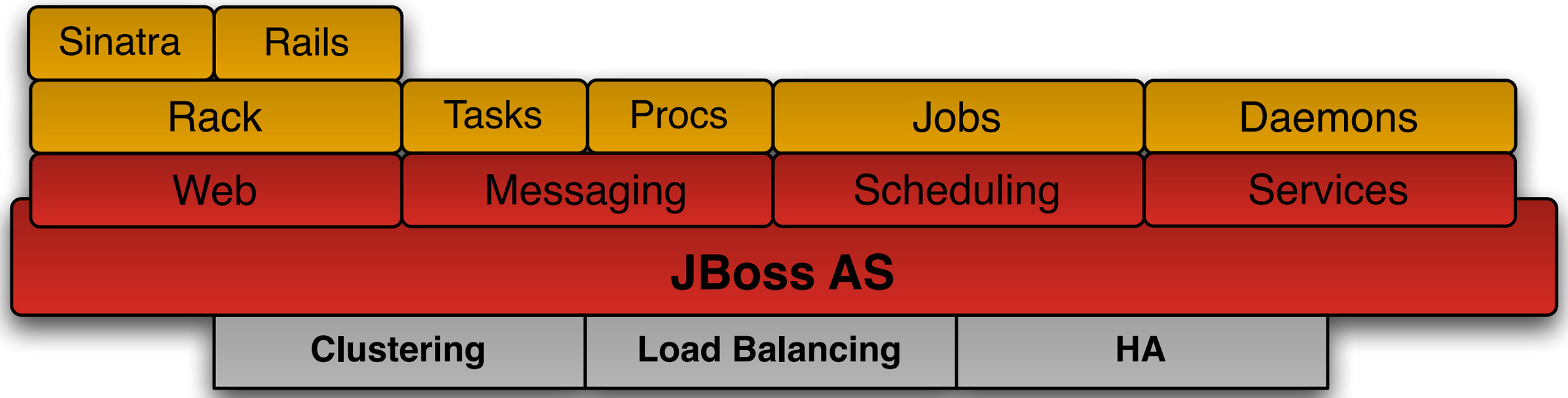
Application Server?



Application Server?



TorqueBox AS



Goals

- Support Ruby web frameworks
  - Rails
  - Sinatra
  - Rack

Goals

- Support Ruby web frameworks
  - Rails
  - Sinatra
  - Rack
- Go **beyond** the web
  - Messaging
  - Jobs
  - Services

Why Ruby?

- No compilation
- Highly expressive
- Lots of shiny frameworks
- Few specifications (more fun!)
- Meta-programming

Expressive

com/foo/Anything.java

```
Set<Person> people = new HashSet<Person>();

for ( Team each : teams ) {
  people.addAll( each.getMembers() );
}

for ( Person each : people ) {
  each.promote();
}
```

Expressive

anything.rb

```
teams.
 collect(&:members).
 flatten.uniq.each(&:promote!)
```

Why *JRuby*?

Why *JRuby*?

```
require 'java'

system = java.lang.System

system.properties.each do |k,v|
  system.properties[k] = v.upcase
end
```

Why *JRuby*?

- Very fast runtime
- Real threads
- Java libraries
- Java tools
- Healthy community

TorqueBox Details

Builds upon and requires JBoss AS 6.x.

Tight integration with the JBoss stack.

Easy Install

(download 1.1 from torquebox.org)

```
$ unzip torquebox-dist-1.1-bin.zip
```

```
$ export TORQUEBOX_HOME=$PWD/torquebox-1.1
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby

$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install

(download 1.1 from torquebox.org)


```
$ unzip torqu

$ export TO                                                    1.1
$ export JBO                                                   /jboss
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby

$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install

(download 1.1 from torquebox.org)

```
$ unzip torquebox-dist-1.1-bin.zip
```

```
$ export TORQUEBOX_HOME=$PWD/torquebox-1.1
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby

$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install

(download 1.1 from torquebox.org)

$ unzip torquebox-dist-1.1-bin.zip

$ export TORQUEBOX_HOME=$PWD/torquebox-1.1
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby

$ export PATH=$JRUBY_HOME/bin:$PATH

Easy Install

(download 1.1 from torquebox.org)

$ unzip torquebox-dist-1.0.0.CR1-b

$ export TORQUEBOX_HOME=

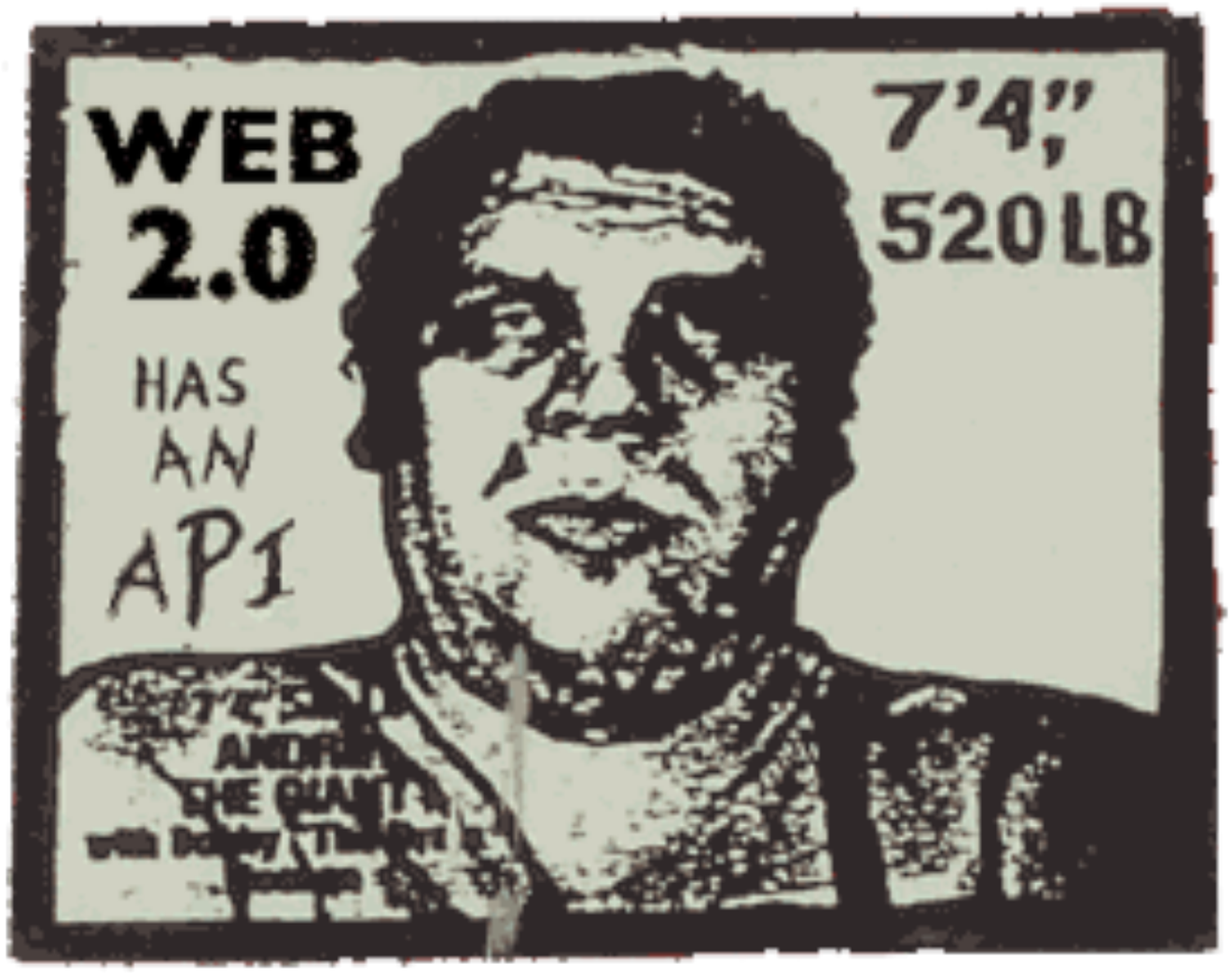
$ export JBOSS_HOME=$TORQ

$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby

$ export PATH=$JRUBY_HOME/bin:$PATH

Make sure the jruby found in your path is in $JRUBY_HOME/bin.

Web



Web

Run Ruby web-apps within the context of the Servlet container.

Without compilation.

A rails application

```
RAILS_ROOT/
  app/
    models/
      person.rb
    views/
      person/
        index.html.haml
        show.html.haml
    controllers/
      persons_controller.rb
  lib/
    my-java-components.jar
```

Deployment

rake **torquebox:run**

Run TorqueBox server

rake **torquebox:deploy**[context_path]

Deploy the app in the current directory

rake **torquebox:undeploy**

Undeploy the app in the current directory

But continue editing

Deployment does **not** create archives (by default).

Continue live-editing of running app:

models,**views**, **controllers**...

Non-surprising.

Almost boring.

Web (Session Integration)

```
class SomeController

  def index
    session[:password] = 'sw0rdfish'
  end

end
```

Web (Session Integration)

```
public class SomeServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse resp) {

    request.getSession().getValue("password");

  }
}
```

Clustering

Ruby applications participate fully in AS clustering.

Can use JBoss **mod_cluster**.

mod_cluster

A reverse proxy implemented as an Apache module with JBoss awareness.

Constantly gathers load statistics and deployment availability for intelligent request distribution.

Let's go *beyond* the web...

Scheduled Jobs

Jobs

app/jobs/newsletter_sender.rb

```
class NewsletterSender

  def run()
    subscriptions = Subscription.find(:all)
    subscriptions.each do |e|
      send_newsletter( e )
    end
  end

end
```

Jobs

config/torquebox.yml

```
jobs:
  monthly_newsletter:
    description: first of month
    job: NewsletterSender
    cron: '0 0 0 1 * ?'

process_tps_reports:
  job: TPSReportProcessor
  cron: '0 0 0 0 MON ?'
  singleton: true
```

Async Tasks

Regular Class

```
class Something

  def foo()
  end

  def bar()
  end

end
```

Blocking invocations

```
something = Something.new

something.foo

something.bar
```

Backgroundable

```
class Something

  include TorqueBox::Messaging::Backgroundable

  def foo()
  end

  def bar()
  end

end
```

Explicitly Non-blocking

```
something = Something.new

something.background.foo

something.background.bar
```

Choices

```
class Something

  include TorqueBox::Messaging::Backgroundable
  always_background :foo

  def foo()
  end

  def bar()
  end

end
```

Implicitly Non-blocking

```
something = Something.new

something.foo
```

See The Future

something = Something.new

future = something.foo

See The Future

future.started?

future.complete?

future.error?

future.result

See The Future

```
class Something

  def foo()
    ...
    count += 1
    future.status = count
    ...
  end

end
```

See The Future

```
# on the 'client' side

future.status_changed?

future.status # => 42
```

Messaging

Messaging

- JMS behind the scenes
- HornetQ is the JBoss JMS implementation

Destinations

config/torquebox.yml

```
queues:
  /queues/questions:
  /queues/answers:
    durable: false
topics:
  /topics/new_accounts
  /topics/notifications
```

Processors

config/torquebox.yml

```
messaging:
  /topics/orders:
    - PrintHandler
    - ShoutHandler
/queues/receipts:
  PrintHandler:
    concurrency: 5
    config:
      printer: the_little_one
```

Processors

app/models/print_handler.rb

```
include TorqueBox::Messaging

class PrintHandler < MessageProcessor
  def initialize(opts)
    @printer = opts['printer'] || default
  end
  def on_message(body)
    puts "Processing #{body} of #{message}"
  end
end
```

Queues

contrived example

```
questions = Queue.new('/queues/questions')
answers = Queue.new('/queues/answers')

Thread.new do
  questions.publish "What time is it?"
  puts answers.receive( :timeout => 1000 )
end

puts questions.receive
answers.publish Time.now
```

Queues

contrived example

```
questions = Queue.new('/queues/questions')
answers = Queue.new('/queues/answers')

Thread.new do
  questions.publish "What time is it?"
  puts answers.receive( :timeout => 1000 )
end

puts questions.receive
answers.publish Time.now
```

Queues

contrived example

```
questions = Queue.new('/queues/questions')
answers = Queue.new('/queues/answers')

Thread.new do
  questions.publish "What time is it?"
  puts answers.receive( :timeout => 1000 )
end

puts questions.receive
answers.publish Time.now
```

Queues

contrived example

```
questions = Queue.new('/queues/questions')
answers = Queue.new('/queues/answers')

Thread.new do
  questions.publish "What time is it?"
  puts answers.receive( :timeout => 1000 )
end
```

puts questions.receive
answers.publish Time.now

Queues

“on the fly”

```
include TorqueBox

queue = Messaging::Queue.new '/queues/foo'
queue.create
  ...
queue.destroy
```

Topics

- behavior is different, but interface is the same.
- all subscribers of a **topic** see each message, but only one subscriber will see any message from a **queue**
- use TorqueBox::Messaging::Topic

Services


A NEW ORLEANS Style CELEBRATION OF
SERVICE AS A STRATEGY!
SERVICE NATION
CITIES OF SERVICE
ServiceAsAStrategy.org

Services

Long-running, non-web "**daemons**" that share the runtime environment and deployment lifecycle of your app.

Services

- Represented as a class with optional initialize(Hash), start() and stop() methods, which should each return quickly.
- Typically will start a long-running loop in a thread and respond to external events.

Services

config/torquebox.yml

```
services:
  TimeMachine:
    queue: /queue/morris_day

  IrcBot:
    server: freenode.net
    channel: #torquebox
    publish: /topics/irc
    singleton: true
```

Services

app/services/time_machine.rb

```
class TimeMachine
  def initialize(opts)
    @queue = Queue.new(opts['queue'])
  end

  def start
    Thread.new { run }
  end

  def stop
    @done = true
  end
end
```

Services

app/services/time_machine.rb

```
class TimeMachine
  def initialize(opts)
    @queue = Queue.new(opts['queue'])
  end

  def start
    Thread.new { run }
  end

  def stop
    @done = true
  end
end
```

Services

app/services/time_machine.rb

```
class TimeMachine
  def initialize(opts)
    @queue = Queue.new(opts['queue'])
  end

  def start
    Thread.new { run }
  end

  def stop
    @done = true
  end
end
```

Services

app/services/time_machine.rb

```
class TimeMachine
  def initialize(opts)
    @queue = Queue.new(opts['queue'])
  end

  def start
    Thread.new { run }
  end

  def stop
    @done = true
  end
end
```

Services

app/services/time_machine.rb

```
class TimeMachine

  def run
    until @done
      @queue.publish(Time.now)
      sleep(1)
    end
  end

end
```

Services

app/services/time_machine.rb

```
class TimeMachine

  def run
    until @done
      @queue.publish(Time.now)
      sleep(1)
    end
  end

end
```

Caching

Caching

Integration with **JBoss Infinispan**, a distributed/ replicated object store.

Transparent Infinispan

Easily used for all of the implicit caching within Rails.

Replace in-memory, or **memcached** caches.

Opaque Infinispan

```
include ActiveSupport::Cache

myCache =
  TorqueBoxStore.new(:name => 'MyCache',
                     :mode => :replicated,
                     :sync => true)
```

Resource Injection

Injection?

Letting the container figure out how to help you wire up complex component models.

aka

Inversion of Control

Guice, PicoContainer, JBoss Microcontainer

Java CDI

```
package com.mycorp;

@ApplicationScoped
class Something {
  @Inject
  private SomethingElse elsewise;
}

@ApplicationScoped
class SomethingElse {

}
```

CDI Injection

```
class MyService
  include TorqueBox::Injectors

  def initialize opts={}
    @thing = inject(com.mycorp.Something)
  end
end
```

But there's more than just CDI you can inject. There's also queues, topics, heroin and other things.

Destination Injection

```
class MyService
  include TorqueBox::Injectors

  def initialize opts={}
    @inbound  = inject( "/topics/questions" )
    @outbound = inject( "/queues/answers" )
  end
end
```

JNDI Injection

```
class MyService
  include TorqueBox::Injectors

  def initialize opts={}
    @factory = inject("java:comp/env/jdbc/myDS")
  end
end
```

JBossMC Injection

```
class MyService
  include TorqueBox::Injectors

  def initialize opts={}
    @heroin = inject( "SomeMCBean" )
  end
end
```

Why MC Beans?

All internal plumbing of **JBoss AS** is stitched together using MC.

Grab the **WebServer**, the **CacheManager**, whatevs.

But how does it perform?

BENchmarks

Real-world Rail application:

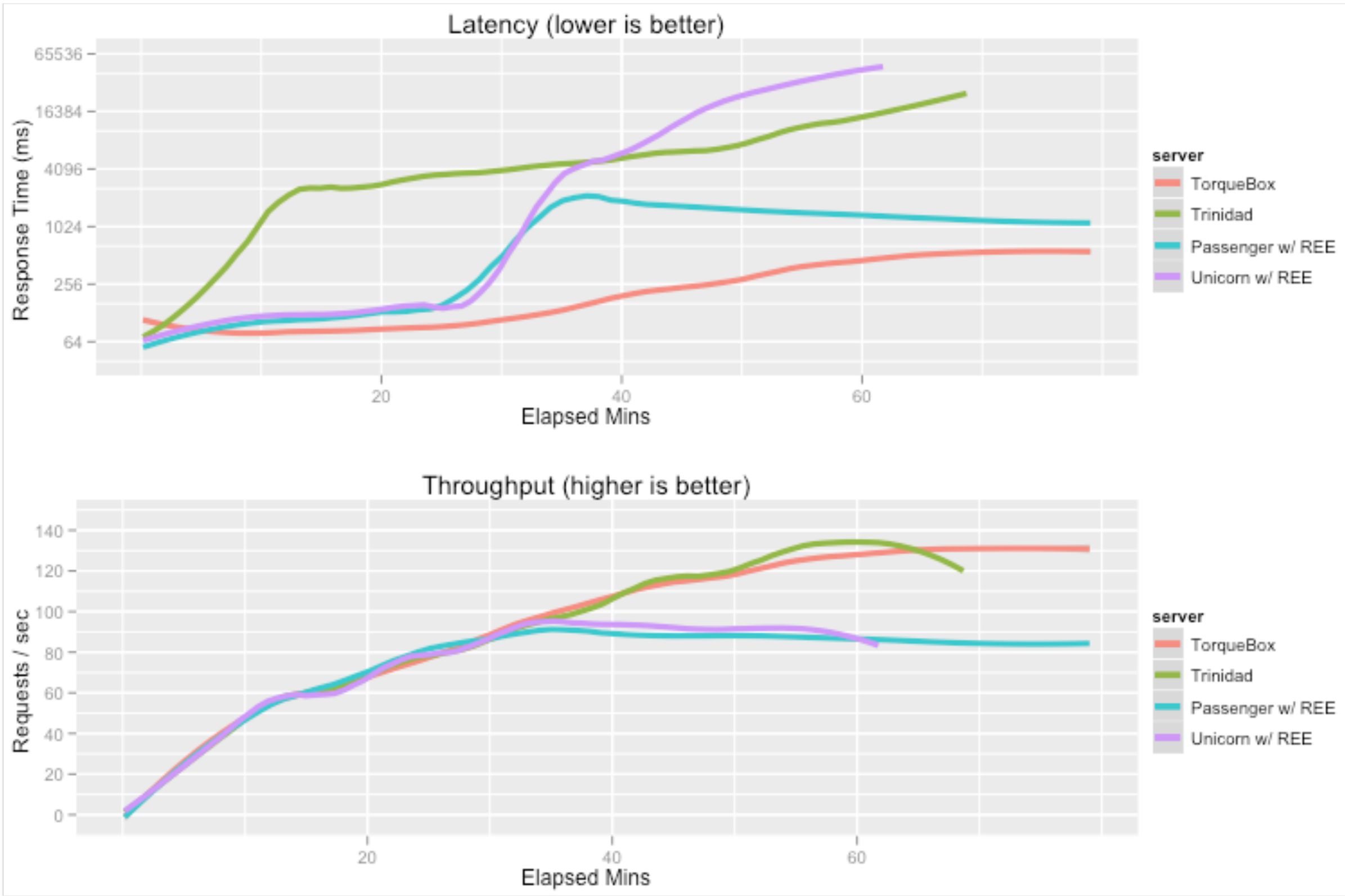
Redmine

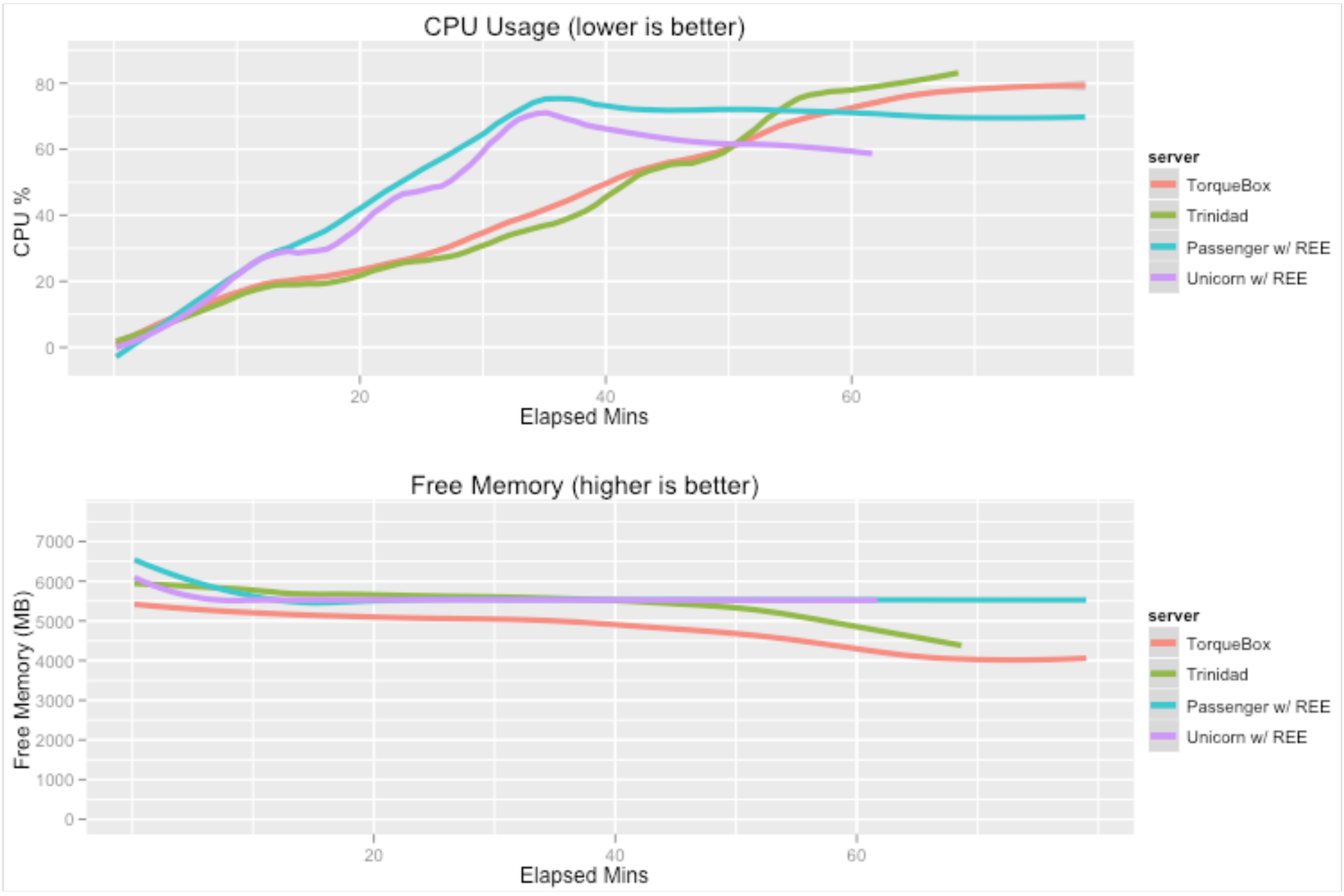
Comparisons:

TorqueBox, Trinidad, Glassfish, Passenger, Unicorn, Thin

Runtimes:

JRuby, MRI, RubyEE


Latency (lower is better)
Response Time (ms)
65536
16384
4096
1024
256
64
20
40
60
Elapsed Mins
server
TorqueBox
Trinidad
Passenger w/ REE
Unicorn w/ REE
Throughput (higher is better)
Requests / sec
140
120
100
80
60
40
20
0
20
40
60
Elapsed Mins
server
TorqueBox
Trinidad
Passenger w/ REE
Unicorn w/ REE


CPU Usage (lower is better)
CPU %
80
60
40
20
0
20
40
60
Elapsed Mins
server
TorqueBox
Trinidad
Passenger w/ REE
Unicorn w/ REE
Free Memory (higher is better)
Free Memory (MB)
7000
6000
5000
4000
3000
2000
1000
0
20
40
60
Elapsed Mins
server
TorqueBox
Trinidad
Passenger w/ REE
Unicorn w/ REE

The TorqueBox Ecosystem

BackStage

Dashboard to inspect and control Ruby components.

And a RESTful API.

TorqueBox::Backstage

Apps | Queues | Topics | Msg. Processors | Jobs | Services

Name	App	Status	Messages	Delivering
ExpiryQueue	n/a	Running	0	0
DLQ	n/a	Running	0	0
/queues/a-kitchen-sink-queue	n/a	Running	7	0
MessageProducerTask	kitchen-sink	Paused	0	0
Backgroundable	kitchen-sink	Running	0	0

StompBox

Easy **Heroku**-esque
git-based deployments.

Stomp Box :: Dashboard

Repositories Managed

ballast-sinatra
master →

ballast-sinatra
test →

buyappalachian.org
torquebox →

Pushes Received

	Date	Status	Commit	Reposit
Push	February 09 - 16:13	received deploy	2dee90f	buyappala

Commits

Lance
fb656070c7e8370d1ca8ccd47b9392fc26ce20b6
2011-02-09T13:12:20-08:00
Add tmp dir for auto deployment

Lance
2dee90fff7d87821126734889623e7cd9a06bd76
2011-02-09T13:12:50-08:00
Merge branch 'torquebox' of github.com:lance/buyappalachian.org into torquebox

Push	February 09 - 14:44	undeployed deploy	351e092	buyappala

Commits

Lance Ball
351e0923cc9fe021b28b98011239115c3c95ca78
2011-02-09T11:44:45-08:00
Use specific liquid version

Push	February 08 - 16:53	undeployed deploy	012e814	buyappala

Commits

Lance Ball

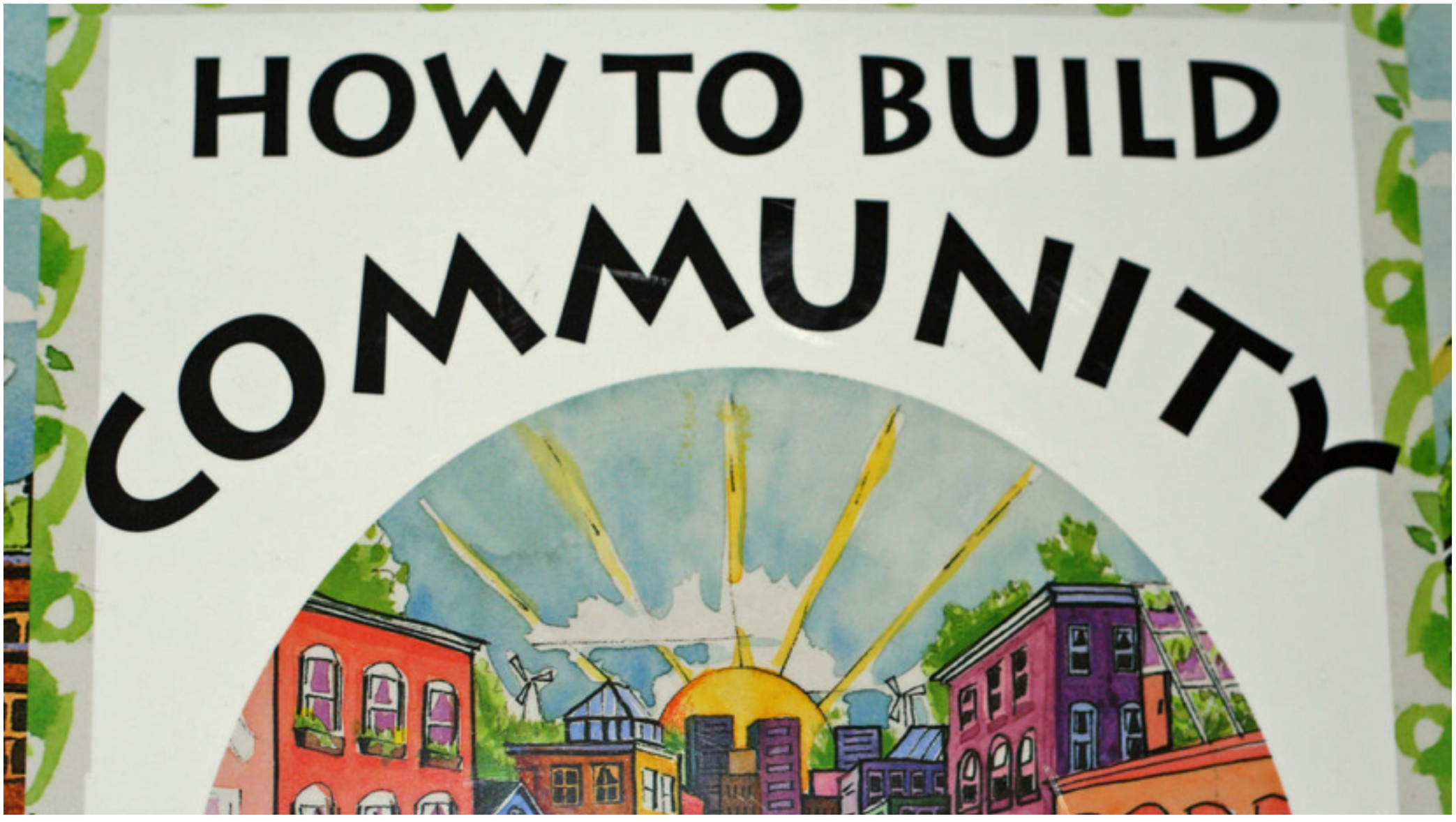
Roadmap

- 1.1 Stable
- 2.0 Development (based on JBoss AS7)

Oh yeah...

It works on **Windows**.

Open Source



Resources

- http://torquebox.org/
- http://github.com/torquebox
- #torquebox on FreeNode
- @torquebox

Thanks!
Questions?