Introduction to Riak

Sean Cribbs

system “whoami”

system “whoami”

- BCS (Tulsa), three years as a musician

system “whoami”

- BCS (Tulsa), three years as a musician
- Started with Ruby in early 2006

system “whoami”

- BCS (Tulsa), three years as a musician
- Started with Ruby in early 2006
- Rails freelancer 2007-2010

system “whoami”

- BCS (Tulsa), three years as a musician
- Started with Ruby in early 2006
- Rails freelancer 2007-2010
- Radiant CMS

system “whoami”

- BCS (Tulsa), three years as a musician
- Started with Ruby in early 2006
- Rails freelancer 2007-2010
- Radiant CMS
- March 2010 - Developer Advocate

500 Rails Error

http://railscantscale.com/

Site is down cause it's built on rails, lol

For additional support, view the FAQ.

Ruby on Rails Billing
Simple subscription billing for Ruby on Rails websites. Make money!
recurly.com/rubyonrails

Ruby on Rails Developer
Latest Technology, Key Projects, Top Ruby Teams, Great Benefits
www.mymatrixjobs.com/RubyonRails

Ruby on Rails Tutorials
Online training videos, lessons and more! Learn from the experts
www.lynda.com

Ads by Google

Rails Can’t Scale

Rails Can’t Scale

what does it mean?

Ruby ~~Rails~~ Can’t Scale

what does it mean?

MySQL ~~Ruby~~ ~~Rails~~ Can’t Scale

what does it mean?

insert straw man here

~~MySQL~~ ~~Ruby~~ ~~Rails~~ Can't Scale

what does it mean?

Mythbusting “scalability”

Mythbusting “scalability”

- Scalability is not a yes/no question

Mythbusting “scalability”

- Scalability is not a yes/no question
- It’s a ratio of **benefit** to **cost**

Mythbusting “scalability”

- Scalability is not a yes/no question
- It’s a ratio of **benefit** to **cost**
  - **Benefits**: low latency, throughput, uptime, concurrency, reliability

Mythbusting “scalability”

- Scalability is not a yes/no question
- It’s a ratio of **benefit** to **cost**
  - **Benefits**: low latency, throughput, uptime, concurrency, reliability
  - **Costs**: CPU, RAM, disk, bandwidth, power, hw/sw

Mythbusting “scalability”

- Scalability is not a yes/no question
- It’s a ratio of **benefit** to **cost**
  - **Benefits**: low latency, throughput, uptime, concurrency, reliability
  - **Costs**: CPU, RAM, disk, bandwidth, power, hw/sw
- **scalability = bang for your buck (ROI)**

Scaling goes both ways

Scaling goes both ways

- **Up/Out** - high traffic, “Big Data”, multi-datacenter

Scaling goes both ways

- **Up/Out** - high traffic, “Big Data”, multi-datacenter
- **Down/In** - laptop, set-top, mobile

How do you scale?

How do you scale?

- Vertically - get bigger, expensive machines

How do you scale?

- Vertically - get bigger, expensive machines
- Horizontally - get more commodity machines

What is Riak?

(the horizontal scaling bits)

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)
  - key-value storage

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)
  - key-value storage
  - masterless, peer-to-peer replication

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)
  - key-value storage
  - masterless, peer-to-peer replication
  - consistent hashing

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)
  - key-value storage
  - masterless, peer-to-peer replication
  - consistent hashing
  - eventual consistency

What is Riak?

(the horizontal scaling bits)

- Based on Amazon's Dynamo (2007)
  - key-value storage
  - masterless, peer-to-peer replication
  - consistent hashing
  - eventual consistency
  - failover - quorums, hinted handoff

“O/RM is the Vietnam of Computer Science.”

-- Ted Neward, 2004/6

“O/RM is the Vietnam of Computer Science.”

-- Ted Neward, 2004/6

slippery slope

“O/RM is the Vietnam of Computer Science.”

-- Ted Neward, 2004/6

slippery slope

diminishing returns

Impedance mismatch

Impedance mismatch

- Relational = predicate logic, truth statements

Impedance mismatch

- Relational = predicate logic, truth statements
- Object = identity, state, behavior, encapsulation

Pick your poison

Pick your poison

- Make the objects fit the database

Pick your poison

- Make the objects fit the database
  - Active Record

Pick your poison

- Make the objects fit the database
  - Active Record
- Make the database fit the objects

Pick your poison

- Make the objects fit the database
  - Active Record
- Make the database fit the objects
  - OODBMS, Graph DBs

Middle ground

Middle ground

- Document databases (*identity* and *state)*

Middle ground

- Document databases (*identity* and *state)*
  - No JOINs - denormalized / composed

Middle ground

- Document databases (*identity* and *state)*
  - No JOINs - denormalized / composed
  - Loosely structured

Middle ground

- Document databases (*identity* and *state)*
  - No JOINs - denormalized / composed
  - Loosely structured
  - Friendly transport formats, e.g. JSON

What is Riak?

(the document database bits)

What is Riak?

(the document database bits)

- Store your objects as JSON (or any format)

What is Riak?

(the document database bits)

- Store your objects as JSON (or any format)
- Link between objects (like hypertext)

What is Riak?

(the document database bits)

- Store your objects as JSON (or any format)
- Link between objects (like hypertext)
- No SQL - query with Javascript Map-Reduce

DevOps - agile infrastructure

DevOps - agile infrastructure

programming your infrastructure

DevOps - agile infrastructure

programming your infrastructure
better communication and planning

Befriend your sysadmin

Befriend your sysadmin

- Integrate with existing infrastructure

Befriend your sysadmin

- Integrate with existing infrastructure
- Reduce the amount of hands-on work

Befriend your sysadmin

- Integrate with existing infrastructure
- Reduce the amount of hands-on work
- Have a growth plan

What is Riak?

(the ops-friendly bits)

What is Riak?

(the ops-friendly bits)

- Web-shaped storage

What is Riak?

(the ops-friendly bits)

- Web-shaped storage
  - Store data in its original format

What is Riak?

(the ops-friendly bits)

- Web-shaped storage
  - Store data in its original format
  - It’s just HTTP - same techniques apply

What is Riak?

(the ops-friendly bits)

- Web-shaped storage
  - Store data in its original format
  - It's just HTTP - same techniques apply
    - load balancing, proxy caches, round-robin DNS

What is Riak?

(the ops-friendly bits)

- Web-shaped storage
  - Store data in its original format
  - It's just HTTP - same techniques apply
    - load balancing, proxy caches, round-robin DNS
- No node is special - grow horizontally

What is Riak?

(the ops-friendly bits)

- Web-shaped storage
  - Store data in its original format
  - It's just HTTP - same techniques apply
    - load balancing, proxy caches, round-robin DNS
- No node is special - grow horizontally
- Get some sleep, fix it in the morning

To review, Riak is...

To review, Riak is...

- Magical Horizontally-Scaling Unicorns

To review, Riak is...

- Magical Horizontally-Scaling Unicorns
- Application-Friendly Rainbows in the Cloud

To review, Riak is...

- Magical Horizontally-Scaling Unicorns
- Application-Friendly Rainbows in the Cloud
- Data-Shredding MapReduce Ninjas

soapbox **do**

http://www.flickr.com/photos/mullica/2769082212/

http://www.flickr.com/photos/bike/236125877/

Stop Cargo-Culting

You are not 37signals, Twitter, or github.

http://www.flickr.com/photos/library_of_virginia/2898496299/

Ignore the Salesperson

Do your own evaluation. Don’t parrot the hype.

Understand your Data

One size does not fit all. Give up your crutches.

Plan for Failure

Your code is not perfect. Your server will go down.

http://www.solarnavigator.net/animal_kingdom/humans/simon_hall.htm

Take Responsibility

Own your choice. Understand the consequences.

end

Getting Started with Riak

http://192.168.0.87/~sean/riak-training/

Starting up a cluster

Starting up a cluster

- make all devrel

Starting up a cluster

- make all devrel
- dev{1,2,3}/bin/riak start

Starting up a cluster

- make all devrel
- dev{1,2,3}/bin/riak start
- dev{2,3}/bin/riak-admin join dev1@127.0.0.1

Riak CRUD

http://127.0.0.1:8091/

Riak CRUD

http://127.0.0.1:8091/

- POST /riak/bucket (C - random key)

Riak CRUD

http://127.0.0.1:8091/

- POST /riak/bucket (C - random key)
- GET /riak/bucket/key (R)

Riak CRUD

http://127.0.0.1:8091/

- POST /riak/bucket (C - random key)
- GET /riak/bucket/key (R)
- PUT /riak/bucket/key (U,C - known key)

Riak CRUD

http://127.0.0.1:8091/

- POST /riak/bucket (C - random key)
- GET /riak/bucket/key (R)
- PUT /riak/bucket/key (U,C - known key)
- DELETE /riak/bucket/key (D)

gem install riak-client

http://192.168.0.87/~sean/riak-training/

require ‘riak’

require ‘riak’

- **Make a client object**
  client = Riak::Client.new

require ‘riak’

- **Make a client object**
  client = Riak::Client.new
- **Get a bucket**
  bucket = client.bucket(‘foo’) # Riak::Bucket

require ‘riak’

- **Make a client object**
  client = Riak::Client.new
- **Get a bucket**
  bucket = client.bucket(‘foo’) # Riak::Bucket
- **Get an object from the bucket**
  obj = bucket.get(‘bar’) # Riak::RObject

require ‘riak’

- **Make a client object**
  client = Riak::Client.new
- **Get a bucket**
  bucket = client.bucket(‘foo’) # Riak::Bucket
- **Get an object from the bucket**
  obj = bucket.get(‘bar’) # Riak::RObject
- **Initialize a new object**
  obj = bucket.new(‘baz’)

Riak::RObject

Riak::RObject

- **Get/set object key**
  obj.key = “bar”

Riak::RObject

- **Get/set object key**
  obj.key = “bar”
- **Get/set content-type**
  obj.content_type = ‘application/json’

Riak::RObject

- **Get/set object key**
  obj.key = “bar”
- **Get/set content-type**
  obj.content_type = ‘application/json’
- **Get/set the object body data (JSON auto enc/dec)**
  obj.data = {“name” => “Sean”}

Riak::RObject

- **Get/set object key**
  obj.key = “bar”
- **Get/set content-type**
  obj.content_type = ‘application/json’
- **Get/set the object body data (JSON auto enc/dec)**
  obj.data = {“name” => “Sean”}
- **Store the object**
  obj.store

LAB

LAB

- Build Riak 3-node developer setup
  (install Erlang as necessary)

LAB

- Build Riak 3-node developer setup (install Erlang as necessary)
- Store an image in Riak via REST API with curl

LAB

- Build Riak 3-node developer setup (install Erlang as necessary)
- Store an image in Riak via REST API with curl
- Store a JSON object via Ruby client

Riak Influences

Riak Influences

- CAP Theorem

Riak Influences

- CAP Theorem
- Amazon Dynamo Paper

Riak Influences

- CAP Theorem
- Amazon Dynamo Paper
- Operational experience running large networks

CAP Theorem

CAP Theorem

- **Consistency** - globally consistent state

CAP Theorem

- **Consistency** - globally consistent state
- **Availability** - system accepts reads and writes

CAP Theorem

- **Consistency** - globally consistent state
- **Availability** - system accepts reads and writes
- **Partition Tolerance** - system handles network failure

CAP: Pick two

Riak: Pick two...

for each operation

Amazon Dynamo

Amazon Dynamo

- originally for their shopping cart storage

Amazon Dynamo

- originally for their shopping cart storage
- masterless, peer-to-peer replication

Amazon Dynamo

- originally for their shopping cart storage
- masterless, peer-to-peer replication
- evenly-distributed key space “ring” (consistent hashing)

Amazon Dynamo

- originally for their shopping cart storage
- masterless, peer-to-peer replication
- evenly-distributed key space “ring” (consistent hashing)
- eventually consistent (AP)

Amazon Dynamo

- originally for their shopping cart storage
- masterless, peer-to-peer replication
- evenly-distributed key space “ring” (consistent hashing)
- eventually consistent (AP)
- robust failure tolerance

Amazon Dynamo

- originally for their shopping cart storage
- masterless, peer-to-peer replication
- evenly-distributed key space “ring” (consistent hashing)
- eventually consistent (AP)
- robust failure tolerance
- hinted handoff

Dynamo Quorums

Dynamo Quorums

- N = number of replicas

Dynamo Quorums

- N = number of replicas
- R = quorum for a successful read

Dynamo Quorums

- N = number of replicas
- R = quorum for a successful read
- W = quorum for a successful write

Dynamo Math

Dynamo Math

- N - R = read fault tolerance

Dynamo Math

- N - R = read fault tolerance
- N - W = write fault tolerance

Dynamo Math

Dynamo Math

- N = 4, W = 2, R = 2

Dynamo Math

- N = 4, W = 2, R = 2
- N - R = 2 (nodes can be down and Riak can still perform reads)

Dynamo Math

- N = 4, W = 2, R = 2
- N - R = 2 (nodes can be down and Riak can still perform reads)
- N - W = 2 (nodes that will receive hinted writes)

Riak Improvements

R=W -- “read your writes consistency”
R+W > N -- “strong consistency”

Riak Improvements

- N can vary *per bucket*

R=W -- “read your writes consistency”
R+W > N -- “strong consistency”

Riak Improvements

- N can vary *per bucket*
- R and W can vary *per operation*

R=W -- “read your writes consistency”
R+W > N -- “strong consistency”

Setting N (replication factor)

Setting N (replication factor)

```
PUT /riak/bucket
Content-Type: application/json

{“props”:{“n_val”:5}}
```

Setting N (replication factor)

```
PUT /riak/bucket
Content-Type: application/json

{“props”:{“n_val”:5}}
```

bucket.n_value = 5

Request-time quorums

Request-time quorums

- GET /riak/bucket/key?r=1

  bucket.get('key', :r => 1)

Request-time quorums

- GET /riak/bucket/key?r=1
  bucket.get('key', :r => 1)
- PUT /riak/bucket/key?w=4
  object.store(:w => 4)

Other quorums

Other quorums

- DW = durable writes (stored to disk)

Other quorums

- DW = durable writes (stored to disk)
- RW = quorum for deletes (delete is a write!)

LAB

LAB

- Change the n_val for a bucket

LAB

- Change the n_val for a bucket
- Stop a node, test R/W quorums

Links

Links

- Simple, one-way relationships between objects

Links

- Simple, one-way relationships between objects
- Like <a> or <link> tags in HTML

Links

- Simple, one-way relationships between objects
- Like <a> or <link> tags in HTML
- Can be efficiently followed via “link-walking”

The Link Header

The Link Header

bucket

tag

Link: </riak/demo/test1>; riaktag="userinfo"

key

Links in Ruby API

Links in Ruby API

- **Create a link**
  Riak::Link.new("/riak/bucket/key", "tag")

Links in Ruby API

- **Create a link**
  Riak::Link.new("/riak/bucket/key", "tag")
- **Read an object's links**
  obj.links # Set<Riak::Link>

Links in Ruby API

- **Create a link**
  Riak::Link.new("/riak/bucket/key", "tag")
- **Read an object's links**
  obj.links # Set<Riak::Link>
- **Convert an object to a link**
  obj.links << obj2.to_link("next")
  obj.store

Link-Walking (traversal)

Link-Walking (traversal)

- Ask Riak to “walk” a sequence of links filtered by bucket, tag, or a combination

Link-Walking (traversal)

- Ask Riak to “walk” a sequence of links filtered by bucket, tag, or a combination
- Optionally, collect results along the way and return them

Link-Walking (traversal)

- Ask Riak to “walk” a sequence of links filtered by bucket, tag, or a combination
- Optionally, collect results along the way and return them
- Can be arbitrarily deep

Link-Walking (traversal)

- Ask Riak to “walk” a sequence of links filtered by bucket, tag, or a combination
- Optionally, collect results along the way and return them
- Can be arbitrarily deep
- Response is doubly-nested multipart/mixed
  riak-client handles this for you - Array of Arrays of RObjects

Link-Walking Example

Link-Walking Example

GET /riak/demo/test1/_,_,1

Start at demo/test1, follow all links and return the objects

```
obj = client['demo']['test1']
obj.walk(:keep => true)
```

Link-Walking Example

Link-Walking Example

GET /riak/demo/test1/**demo**,_,1

Start at demo/test1, follow all links pointing to the **demo bucket** and return the objects

```
obj.walk(:bucket => “demo”, :keep => true)
```

Link-Walking Example

Link-Walking Example

GET /riak/demo/test1/_,**friend**,1

Start at demo/test1, follow all links **tagged "friend"** and return the objects

```
object.walk(:tag => 'friend', :keep => true)
```

Link-Walking Example

Link-Walking Example

GET /riak/demo/test1/_,_,_/_,_,1

Start at demo/test1, follow all links, follow all links from those, return everything from the last set

```
obj.walk({},{:keep => true})
```

LAB

LAB

- Create a network of mutual friends using Ruby:
  Justin, Andy, Bryan, Tony

LAB

- Create a network of mutual friends using Ruby: Justin, Andy, Bryan, Tony
- Find all of Justin's friends' friends via curl

LAB

- Create a network of mutual friends using Ruby:
  Justin, Andy, Bryan, Tony
- Find all of Justin's friends' friends via curl
- Find all of Tony's first and second-tier friends in Ruby

Riak Map-Reduce

I believe I did, Bob.

Riak Map-Reduce

Riak Map-Reduce

- Based on Google's Map-Reduce idea

Riak Map-Reduce

- Based on Google's Map-Reduce idea
- Some similarities to Hadoop and CouchDB

Riak Map-Reduce

- Based on Google's Map-Reduce idea
- Some similarities to Hadoop and CouchDB
- High data-locality

Riak Map-Reduce

- Based on Google's Map-Reduce idea
- Some similarities to Hadoop and CouchDB
- High data-locality
- Phases can be Javascript or Erlang

Map-Reduce Terminology

Map-Reduce Terminology

- **Job:** a query, composed of inputs and phases

Map-Reduce Terminology

- **Job:** a query, composed of inputs and phases
- **Phase:** a single map or reduce function application

Map-Reduce Terminology

- **Job:** a query, composed of inputs and phases
- **Phase:** a single map or reduce function application
- **Map:** phase applied to each item - filter, transform

Map-Reduce Terminology

- **Job:** a query, composed of inputs and phases
- **Phase:** a single map or reduce function application
- **Map:** phase applied to each item - filter, transform
- **Reduce:** phase applied to the collection - collate, aggregate, compute

Map Phases

Map Phases

- Inputs are bucket-key pairs (with optional key-specific data)

Map Phases

- Inputs are bucket-key pairs (with optional key-specific data)
- Must return a list

Map Phases

- Inputs are bucket-key pairs (with optional key-specific data)
- Must return a list
- Parallel results are aggregated

Map Built-ins

Map Built-ins

- Riak.mapValues

Map Built-ins

- Riak.mapValues
- Riak.mapValuesJson

Reduce Phases

Reduce Phases

- Performed on a single node

Reduce Phases

- Performed on a single node
- Two processes per reduce phase increase parallelism

Reduce Phases

- Performed on a single node
- Two processes per reduce phase increase parallelism
- Must return a list

Reduce Built-ins

Reduce Built-ins

- Riak.reduceMin

Reduce Built-ins

- Riak.reduceMin
- Riak.reduceMax

Reduce Built-ins

- Riak.reduceMin
- Riak.reduceMax
- Riak.reduceSort

Build a M/R Job

Build a M/R Job

- Specify inputs

Build a M/R Job

- Specify inputs
  - bucket/key, bucket/key/key-specific-arg, bucket

Build a M/R Job

- Specify inputs
  - bucket/key, bucket/key/key-specific-arg, bucket
- Specify phases

Build a M/R Job

- Specify inputs
  - bucket/key, bucket/key/key-specific-arg, bucket
- Specify phases
  - Each can receive a static argument

Build a M/R Job

- Specify inputs
  - bucket/key, bucket/key/key-specific-arg, bucket
- Specify phases
  - Each can receive a static argument
  - Each can return intermediate results

Map-Reduce on HTTP

Map-Reduce on HTTP

- Job is a JSON object

Map-Reduce on HTTP

- Job is a JSON object
- POST /mapred

A simple M/R job

```
{“inputs”: “goog”,
 “query”: [{“map”:
            {“language”:”javascript”,
             “name”: “Riak.mapValuesJson”,
                “keep”: true}]}
```

A simple M/R job

```
{“inputs”: “goog”,
 “query”: [{“map”:
            {“language”:”javascript”,
             “name”: “Riak.mapValuesJson”,
                 “keep”: true}]}
```

```
Riak::MapReduce.new(c).add(‘goog’).
        map(‘Riak.mapValuesJson’, :keep => true).run
```

Another M/R Job

Another M/R Job

{“inputs”: “goog”,

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                   “name”: “App.findHighGreater”,
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                “name”: “App.findHighGreater”,
                “arg”: 600.0,
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                “name”: “App.findHighGreater”,
                “arg”: 600.0,
                “keep”: false},
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                “name”: “App.findHighGreater”,
                “arg”: 600.0,
                “keep”: false},
            {“reduce”:{“language”:”javascript,
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                “name”: “App.findHighGreater”,
                “arg”: 600.0,
                “keep”: false},
           {“reduce”:{“language”:”javascript,
                “name”: “Riak.reduceMax”,
```

Another M/R Job

```
{“inputs”: “goog”,
 “query”: [{“map”:{“language”:”javascript”,
                “name”: “App.findHighGreater”,
                “arg”: 600.0,
                “keep”: false},
            {“reduce”:{“language”:”javascript,
                “name”: “Riak.reduceMax”,
                “keep”: true}]}
```

Another M/R Job

```
{"inputs": "goog",
 "query": [{"map":{"language":"javascript",
                   "name": "App.findHighGreater",
                   "arg": 600.0,
                   "keep": false},
            {"reduce":{"language":"javascript,
                       "name": "Riak.reduceMax",
                       "keep": true}]}
```

```
Riak::MapReduce.new(c).add('goog').
        map('App.findHighGreater', :arg => 600.0).
        reduce("Riak.reduceMax", :keep => true).run
```

LAB

LAB

- Load the Google historical stock data

LAB

- Load the Google historical stock data
- Use Map-Reduce to:

LAB

- Load the Google historical stock data
- Use Map-Reduce to:
  - Find the highest high price in the first week of March

LAB

- Load the Google historical stock data
- Use Map-Reduce to:
  - Find the highest high price in the first week of March
  - Find the day with the widest price range since the beginning of the year

Ripple

```
gem install ripple
```

Ripple

```
gem install ripple
```

- Document-style persistence/modeling library

Ripple

```
gem install ripple
```

- Document-style persistence/modeling library
- ActiveModel niceties

Ripple

```
gem install ripple
```

- Document-style persistence/modeling library
- ActiveModel niceties
- Callbacks

Ripple

```
gem install ripple
```

- Document-style persistence/modeling library
- ActiveModel niceties
  - Callbacks
  - Validations

Ripple

```
gem install ripple
```

- Document-style persistence/modeling library
- ActiveModel niceties
  - Callbacks
  - Validations
  - ActionPack compatibility

Ripple Roadmap

Ripple Roadmap

- Associations

Ripple Roadmap

- Associations
- Indexes

Ripple Roadmap

- Associations
- Indexes
- Use a property/method as the key

Ripple Roadmap

- Associations
- Indexes
- Use a property/method as the key
- Dynamic finders

Ripple Roadmap

- Associations
- Indexes
- Use a property/method as the key
- Dynamic finders
- Session store

Create a Ripple::Document

Create a Ripple::Document

```
class Person
  include Ripple::Document

  property :name, String, :presence => true
end
```

Sample app walkthrough

Sample app walkthrough

http://github.com/seancribbs/riak-url-shortener

Create a shortened URL

Create a shortened URL

```
post '/' do
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
    obj.content_type = "text/plain"
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
    obj.content_type = "text/plain"
    obj.data = params[:url]
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
    obj.content_type = "text/plain"
    obj.data = params[:url]
    obj.store
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
    obj.content_type = "text/plain"
    obj.data = params[:url]
    obj.store
  end
```

Create a shortened URL

```
post '/' do
  client = Riak::Client.new
  key = create_shortcode.call(params[:url])
  bucket = Riak::Bucket.new(client, 'urls')
  Riak::RObject.new(bucket, key).tap do |obj|
    obj.content_type = "text/plain"
    obj.data = params[:url]
    obj.store
  end
  status 201
  @url = "http://#{request.host_with_port}/#{key}"
  headers "Location" => @url
  haml :created
end
```

Embiggen the short URL

Embiggen the short URL

```
get '/:code' do
  begin
    client = Riak::Client.new
```

Embiggen the short URL

```
get '/:code' do
  begin
    client = Riak::Client.new
    obj = Riak::Bucket.new(client, 'urls').get(params[:code])
```

Embiggen the short URL

```
get '/:code' do
  begin
    client = Riak::Client.new
    obj = Riak::Bucket.new(client, 'urls').get(params[:code])
    url = obj.data
```

Embiggen the short URL

```
get '/:code' do
  begin
    client = Riak::Client.new
    obj = Riak::Bucket.new(client, 'urls').get(params[:code])
    url = obj.data
    halt 301, {"Location" => url}, []
```

Embiggen the short URL

```
get '/:code' do
  begin
    client = Riak::Client.new
    obj = Riak::Bucket.new(client, 'urls').get(params[:code])
    url = obj.data
    halt 301, {"Location" => url}, []
  rescue Riak::FailedRequest => fr
    case fr.code
    when 404
      not_found(haml(:not_found))
    when 500..599
      error(503, haml(:error))
    end
  end
end
```

fin

sean@basho.com
@seancribbs
“seancribbs” on Freenode IRC #riak

Questions?

http://wiki.basho.com
riak-users@lists.basho.com