

# The Holy Grail of Databases



The Secret of Perfect Data Management

“Sponsored” By



# I WAS ON RAILS



## BEFORE IT WAS COOL

# Who I am



- ❧ Eric Redmond
- ❧ “Seven Databases in Seven Weeks”
  - ❧ Pragmatic Press
  - ❧ Out by the end of Summer

# Complex Data



# Complex Data

---

“Complexity is a symptom of confusion, not a cause.”

-- *Jeff Hawkins (of Palm Pilot fame)*

A photograph of a man dressed as a medieval knight or templar. He wears a chainmail coif and a white tunic with a red cross. He holds a golden chalice in his left hand and a white mug with a green logo in his right hand. A sign in front of him reads "CHOOSE NICELY".

What is the Holy Grail?

# Terms

---



## ❖ Database

- ❖ a system intended to organize, store, and retrieve large amounts of data easily. It consists of an organized collection of data for one or more uses... - wikipedia

## ❖ Datastore

- ❖ A data store is a data repository of a set of integrated objects. These objects are modeled using classes defined in database schemas. - wikipedia
- ❖ Clearly this second sentence is wrong – it wouldn't include Riak or CouchDB.

# Terms



- ❖ SQL
  - ❖ Not NoSQL
- ❖ NoSQL
  - ❖ Linear Scalability (business decision: known estimatable requirements to grow in a consistent way)
  - ❖ Ability to be Distributed
  - ❖ Low Latency

# Acronyms



❖ ACID

❖ Transaction-based (generally SQL)

❖ BASE

❖ Request-based (NoSQL)

❖ CAP

❖ Consistency

❖ Availability

❖ Partition Tolerance

# ACID



- ❖ **Atomic**
  - ❖ Transactions are “all or nothing”
- ❖ **Consistent**
  - ❖ The system data will have integrity – data will never be in an *inconsistent state*
- ❖ **Isolated**
  - ❖ Transactions cannot see each other – data from one transaction is unavailable until it is complete
- ❖ **Durable**
  - ❖ Can recover from failures – generally some underlying disk writes

# BASE



- ❖ Basically Available
- ❖ Soft state
- ❖ Eventual consistency

# CAP Theorem



- ❖ Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services
  - ❖ Nancy Lynch and Seth Gilbert
  - ❖ "...it is impossible to reliably provide atomic, consistent data when there are partitions in the network. It is feasible, however, to achieve any two of the three properties: consistency, availability, and partition tolerance."

# Consistency

---



- ❖ A request to any connectable node in the system returns the same data
- ❖ Strong Consistency
  - ❖ aka: Strict, Linearizable or Atomic
  - ❖ When an update completes, subsequent access returns the new result
- ❖ Weak Consistency
  - ❖ For most NoSQL purposes, we mean Eventual
    - ❖ When an update completes, subsequent access will eventually return the new result

# Correct Consistency



- ❖ List of cities
- ❖ DNS is eventually consistent

# Availability

---

- ❖ Colloquial definition
  - ❖ The data is available when I want it.
  - ❖ Wrong! (latency) It could take forever
- ❖ “Technical-er” definition
  - ❖ Nodes which may sustain pack-loss continue serving requests.
  - ❖ Or: Is it possible to be *unavailable*?

# Partition Tolerance

---



- ❖ Despite message loss, the DB continues to operate.
- ❖ A DB is either P or not.
- ❖ “...the choice is almost always between sequential consistency and high availability”
- ❖ <http://www.cloudera.com/blog/2010/04/cap-confusion-problems-with-partition-tolerance>

# Consistency & Availability





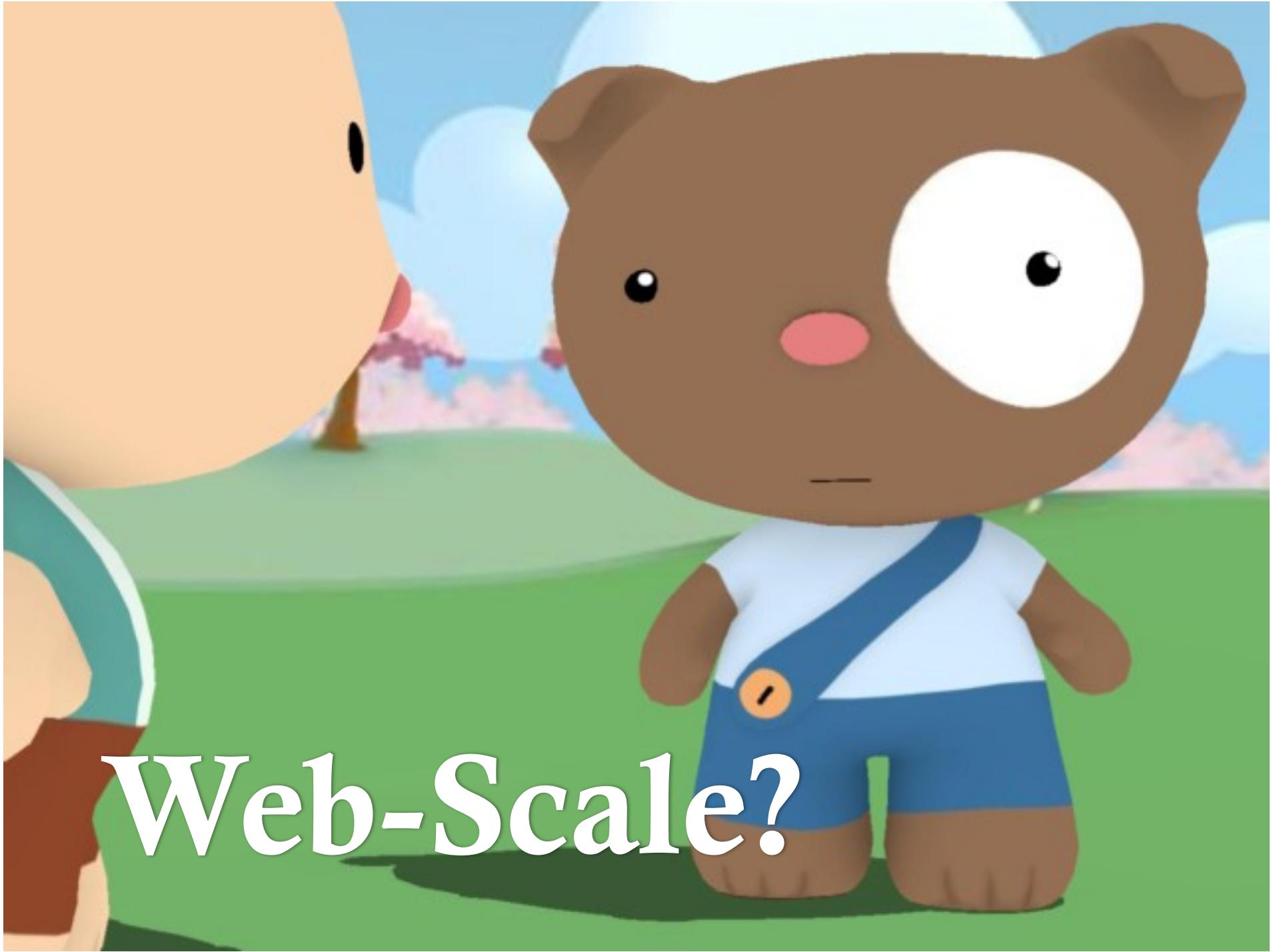
A clear glass bottle of SGS Grand Marnier liqueur is shown floating in the middle of a vast, calm blue ocean. The bottle is tilted slightly, with its corked top pointing towards the upper right. The label on the bottle clearly displays the brand name "SGS" and "Grand Marnier". In the background, a tropical landscape with palm trees and distant hills is visible under a bright sky.

Eventual  
Consistency

# Other Concerns: Latency

---

- ❖ Not addressed in CAP
- ❖ The focus of many “web-scale” NoSQL solutions
- ❖ Case in point:
  - ❖ PNUTS (Yahoo database) gives up BOTH C and A



Web-Scale?

# Common Patterns



- ❖ Replication
- ❖ N/R/W
- ❖ Consistent Hashing
- ❖ Mapreduce

# Replication

---

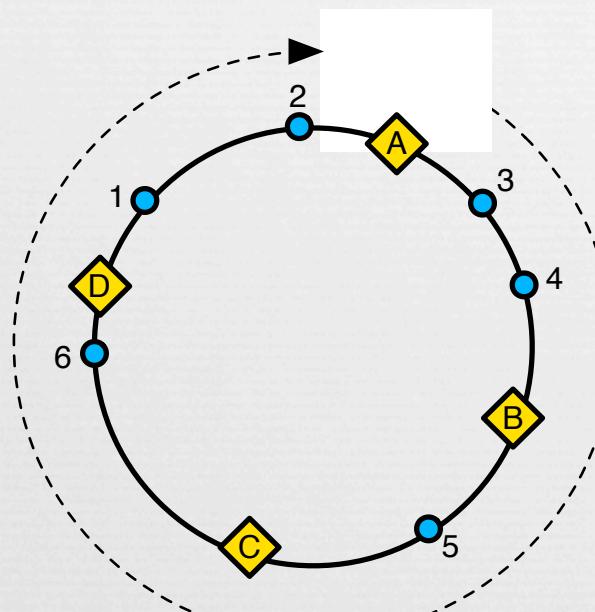
- ❖ Copying data amongst nodes in a distributed database
- ❖ Lazy (Optimistic) replication
  - ❖ Gossip (nodes communicate to stay in sync)
- ❖ Master/Slave
- ❖ Master/Master
  - ❖ Vector Clocks (keep track of write order per client)
  - ❖ MVCC (subversion)

# N/R/W

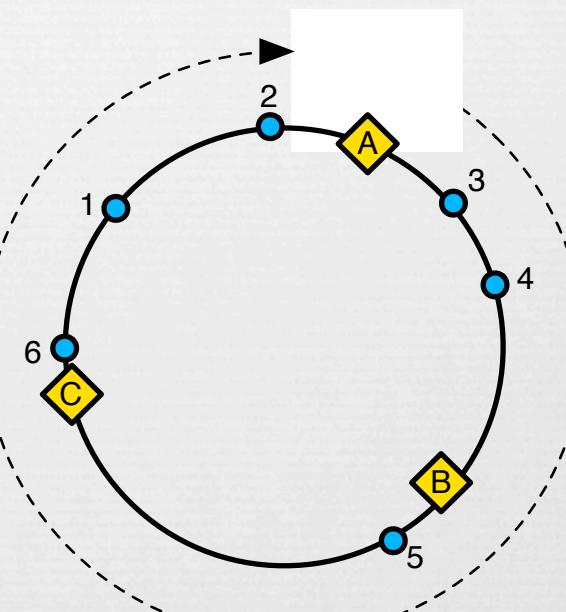


- ❖ N/R/W
  - ❖ N = Nodes to write to (per bucket)
  - ❖ W = Nodes written to before success
  - ❖ R = Nodes read from before success
- ❖ Support both CP and AP in one database
- ❖ Used by Cassandra and Riak

# Consistent Hashing

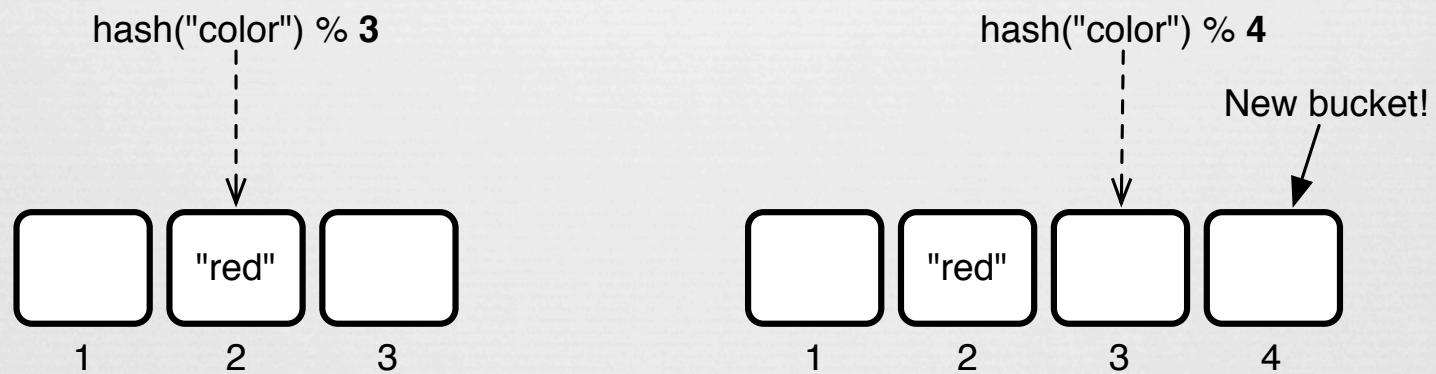


A = [1, 2]  
B = [3, 4]  
C = [5]  
D = [6]



A = [1, 2, 6]  
B = [3, 4]  
C = [5]  
D = []

# (in)Consistent Hashing



# Mapreduce

---

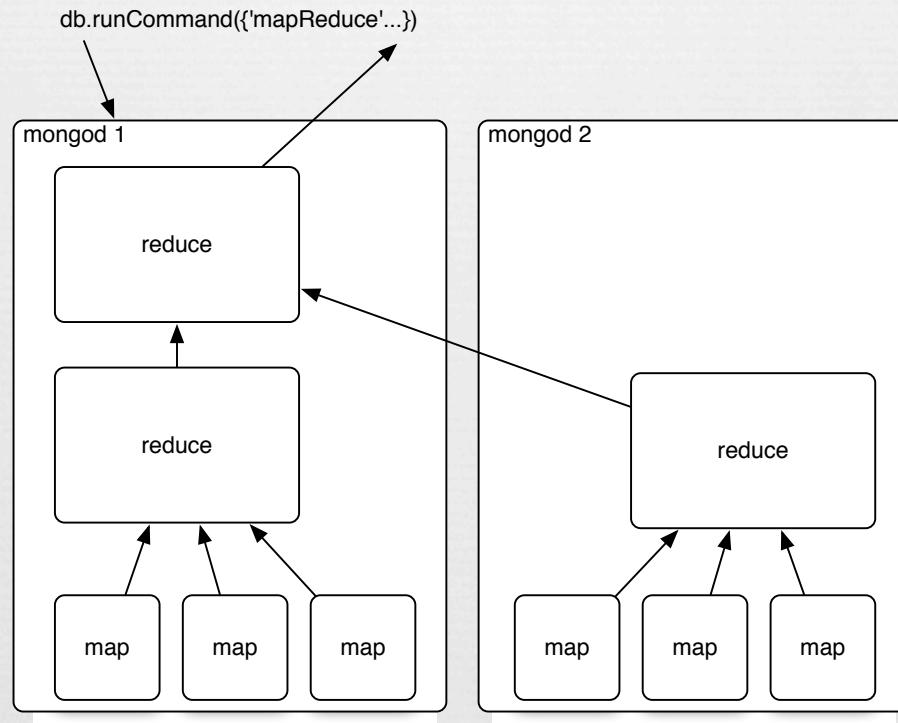


```
rooms = Room.all
```

```
caps = rooms.map{|room| room.capacity}
```

```
result = caps.reduce(0){|sum, capacity| sum+capacity}
```

# Mapreduce



The Holy Grail of DBs is



# Coming Soon



# Our Fine Selection



## Column A

❖ MySQL

❖ PostgreSQL

❖ Riak

❖ Cassandra

❖ HBase

❖ MongoDB

## Column B

❖ CouchDB

❖ Neo4j

❖ FlockDB

❖ Memcached

❖ Kyoto Cabinet

❖ Redis

# Relational Models



- ≈ PostgreSQL (full featured)
  - ≈ <http://bitbucket.org/ged/rubyPg>
  - ≈ [http://github.com/Casecommons/pg\\_search](http://github.com/Casecommons/pg_search)
  - ≈ <http://github.com/tenderlove/textile>
- ≈ MySQL (lighter)
  - ≈ <http://rubygems.org/gems/mysql<=turd>
  - ≈ <http://github.com/brianmario/mysql2>
  - ≈ <http://github.com/oldmoe/mysqlplus>
  - ≈ <https://github.com/igrigorik/em-mysqlplus> <= defunct
- ≈ Drizzle (lightest)
  - ≈ <http://drizzle.org/>
  - ≈ <https://github.com/jakedouglas/libdrizzle-ruby-ffi>

# Benchmark / Numbers



	user	system	total	real
<b>Mysql2</b>				
	0.750000	0.180000	0.930000	(1.821655)
<b>do_mysql</b>				
	1.650000	0.200000	1.850000	(2.811357)
<b>Mysql</b>				
	7.500000	0.210000	7.710000	(8.065871)

Gemfile — rc2011

File database.yml

```
source 'http://rubygems.org'

gem 'rails', '3.0.5'
gem 'sqlite3'

# Use unicorn as the web server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger (ruby-debug for Ruby 1.8.7+, ruby-debug19
for Ruby 1.9.2+)
# gem 'ruby-debug'
# gem 'ruby-debug19', :require => 'ruby-debug'

# Bundle the extra gems:
# gem 'bj'
# gem 'nokogiri'
# gem 'sqlite3-ruby', :require => 'sqlite3'
# gem 'aws-s3', :require => 'aws/s3'
```

Column: 6 | Ruby | Soft Tabs: 2 | -

rc2011

- ▶ app
- ▶ config
- ▶ config.ru
- ▶ db
- ▶ doc
- ▶ Gemfile
- ▶ lib
- ▶ log
- ▶ public
- ▶ Rakefile
- ▶ README
- ▶ script
- ▶ test
- ▶ tmp
- ▶ vendor

The image shows a screenshot of a software development environment, likely a Ruby on Rails project. On the left is a code editor window titled 'Gemfile — rc2011' containing the contents of a Gemfile. The code includes dependencies like rails 3.0.5 and sqlite3, configuration for unicorn, deployment with capistrano, and various debugger configurations. On the right is a file tree view showing the directory structure of the project 'rc2011', which includes subfolders for app, config, db, doc, lib, log, public, Rakefile, README, script, test, tmp, and vendor. At the bottom of the interface are standard file operations buttons (New, Open, Save, etc.) and a toolbar with icons for file operations.

# Sounds Like “kristmus”



- ❖ Trigram
- ❖ Algorithm for misspellings
- ❖ Metaphone
- ❖ Algorithm for similar sounds

## Terminal — ruby — 80x22

bash

ruby

bash

```
:id => 223,  
:iso => "EH",  
:name => "WESTERN SAHARA"
```

```
?23] #<Country:0x000001009315d8> {  
  :id => 224,  
  :iso => "YE",  
  :name => "YEMEN"
```

```
?24] #<Country:0x00000100930c28> {  
  :id => 225,  
  :iso => "ZM",  
  :name => "ZAMBIA"
```

```
?25] #<Country:0x0000010092fe68> {  
  :id => 226,  
  :iso => "ZW",  
  :name => "ZIMBABWE"
```

```
l  
l.9.2-p0 > █
```

# ARel?



```
SELECT *, cube_distance(ranks, '1,0,0') dist  
FROM movie_genres  
WHERE cube_enlarge('(1,0,0)::cube, 0, 3) @> ranks  
ORDER BY dist;
```

# Bigtable/Columnar Style

---



## ❖ HBase

- ❖ <http://github.com/greglu/hbase-stargate> (slow)
- ❖ <http://github.com/sqs/rhino> (defunct)
- ❖ <http://rubygems.org/gems/thrift>

## ❖ Cassandra

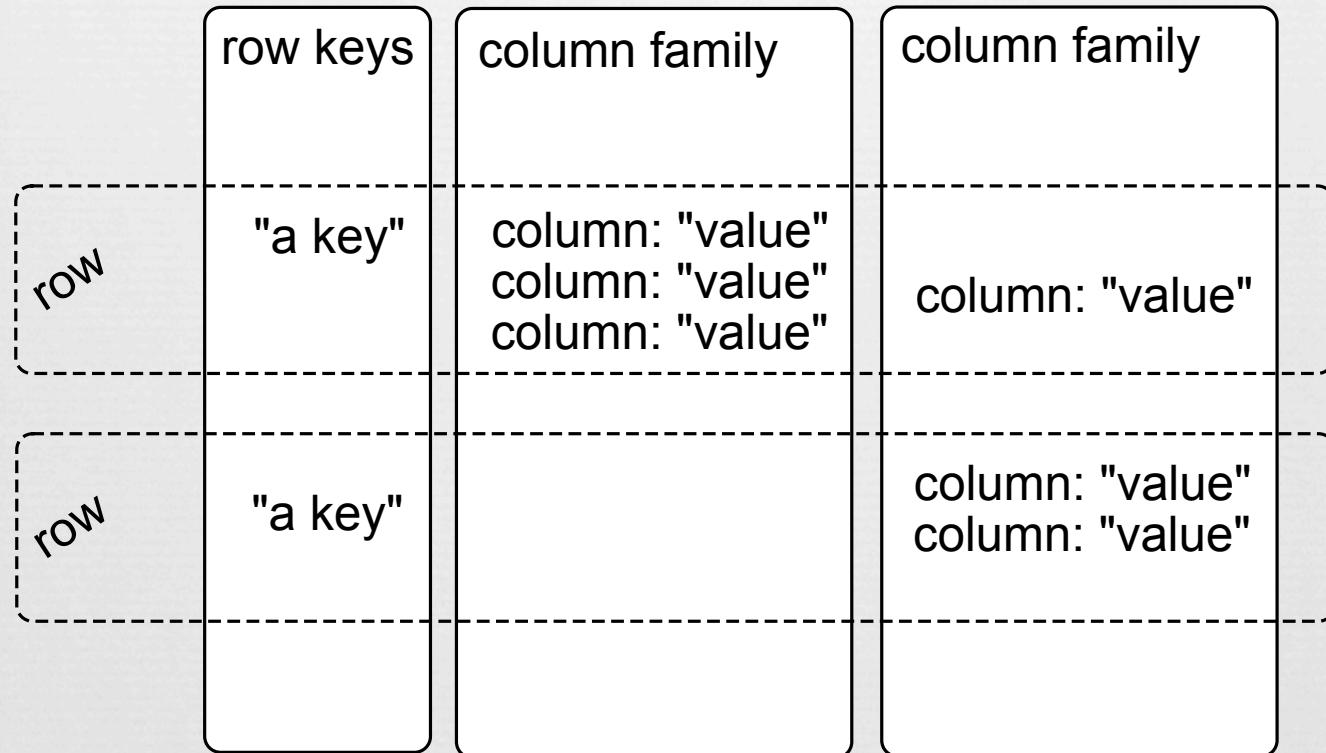
- ❖ Hybrid. Node architecture like dynamo – data structure like BigTable w/ column families
- ❖ <http://github.com/fauna/cassandra>
- ❖ [http://github.com/NZKoz/cassandra\\_object](http://github.com/NZKoz/cassandra_object)

# HBase



- ❖ Google BigTable implementation
- ❖ Born of Hadoop (Java mapreduce engine)
- ❖ JRuby CLI!

# “BigTable” Columns



## Terminal — bash — 80x22

bash

ruby

java

```
(0.1ms) SET client_min_messages TO 'panic'  
(0.1ms) SET standard_conforming_strings = on  
(0.1ms) SET client_min_messages TO 'notice'  
(0.4ms) SET time zone 'UTC'  
(0.1ms) SHOW TIME ZONE  
it Load (0.9ms)  SELECT "events".* FROM "events"  
(2.3ms)   SELECT a.attname, format_type(a.atttypid, a.atttypmod), d.adisnull  
           :notnull  
           pg_attribute a LEFT JOIN pg_attrdef d      [  
           attrelid = d.adrelid AND a.attnum = d.adnum  
           : a.attrelid = '"events"'::regclass  
           .attnum > 0 AND NOT a.attisdropped  
           ] BY a.attnum  
(0.3ms)  SHOW TABLES  
itry Load (0.4ms)  SELECT `countries`.* FROM `countries` WHERE `country`  
           = 'us' LIMIT 1  
reated events/index.html.erb within layouts/application (77.4ms)  
et 200 OK in 96ms (Views: 78.0ms | ActiveRecord: 4.8ms)  
[1-04-24 15:07:14] INFO  going to shutdown ...  
[1-04-24 15:07:14] INFO  WEBrick::HTTPServer#start done.  
ig  
11$
```

# launch\_hbase.rb

```
require 'hbase'  
class Object  
  include Apache::Hadoop::Hbase::Thrift  
  def thrift  
    unless defined?(@@hclient)  
      @@tsocket = Thrift::Socket.new( '127.0.0.1', 9090 )  
      @@ttransport = Thrift::BufferedTransport.new( @@tsocket )  
      @@tprotocol = Thrift::BinaryProtocol.new( @@ttransport )  
      @@hclient = Hbase::Client.new( @@tprotocol )  
    end  
    @@ttransport.open  
    yield @@hclient  
  ensure  
    @@ttransport.close  
  end  
end
```

# Hbase Migration

---

```
class CreateWikis < ActiveRecord::Migration
  def self.up
    thrift do |hbase|
      hbase.createTable( 'wiki', [
        ColumnDescriptor.new(:name => 'text:', :maxVersions=>10),
        ColumnDescriptor.new(:name => 'title:')
      ])
    end
  end
```

# wiki.rb

```
def self.all( start = " )  
    wikis = []  
    thrift do |hbase|  
        scanner = hbase.scannerOpen( 'wiki', start, ['title:', 'text:] )  
        while (row = hbase.scannerGet(scanner)).present?  
            row.each do |v|  
                wikis << Wiki.new( :title => v.columns['title:'].value,  
                                :text => v.columns['text:'].value)  
            end  
        end  
    end  
    wikis  
end
```

# wiki.rb

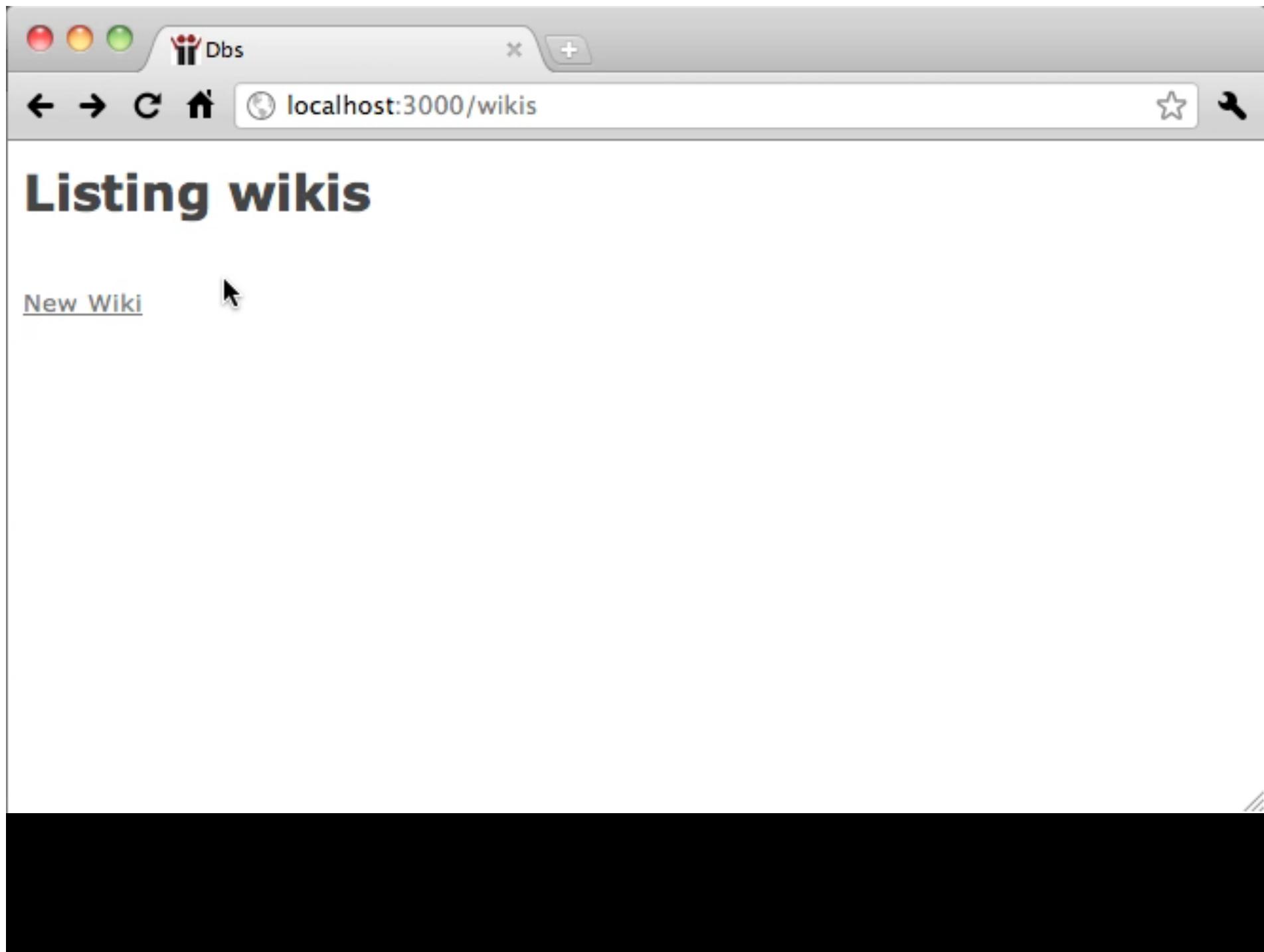


```
def self.find(title)
  thrift do |hbase|
    hbase.getRow('wiki', title).each do |v|
      return Wiki.new( :title => v.columns['title:'].value,
                      :text => v.columns['text:'].value )
    end
  end
end
```

# wiki.rb



```
def history
  historical_text = []
  thrift do |hbase|
    hbase.getVer( 'wiki', title, 'text:', 10 ).each do |v|
      historical_text << v.value.dup
    end
  end
  historical_text
end
```



# HBase Benefits

---

- ❖ Strong (and flexible) columnar schema
- ❖ Sequential Reads and Column Versioning
- ❖ Mapreduce via Hadoop integration
- ❖ Consistent (configurable to Available)
- ❖ Great for Wide Area Networks
- ❖ (*Google, Facebook*)

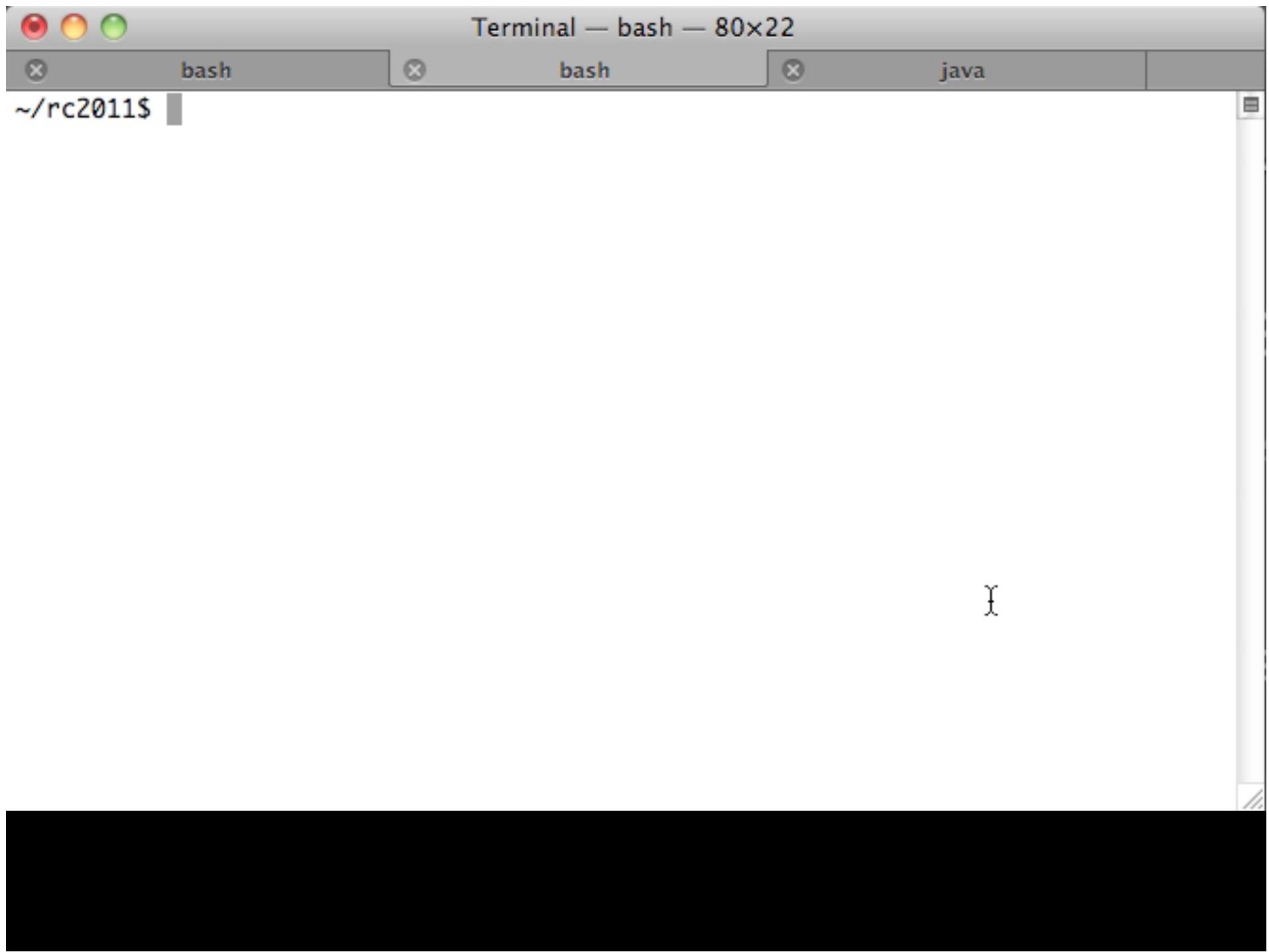
# storage-conf.xml ☹



```
<Keyspace Name="CassandraObject">
    <ColumnFamily CompareWith="UTF8Type" Name="Customers"/>
    <ColumnFamily CompareWith="TimeUUIDType"
Name="CustomersByLastName" />
    <ColumnFamily CompareWith="UTF8Type"
Name="Appointments" />
    ...
</Keyspace>
```

## Terminal — bash — 80x22

```
bash                                bash                                bash
invoke    test_unit
create     test/unit/wiki_test.rb
create     test/fixtures/wikis.yml
route     resources :wikis
invoke    scaffold_controller
create     app/controllers/wikis_controller.rb
invoke    erb
create     app/views/wikis
create     app/views/wikis/index.html.erb
create     app/views/wikis/edit.html.erb
create     app/views/wikis/show.html.erb
create     app/views/wikis/new.html.erb
create     app/views/wikis/_form.html.erb
invoke    test_unit
create     test/functional/wikis_controller_test.rb
invoke    helper
create     app/helpers/wikis_helper.rb
invoke    test_unit
create     test/unit/helpers/wikis_helper_test.rb
invoke    stylesheets
identical  public/stylesheets/scaffold.css
~/rc2011$
```



# Cassandra

---



- ❖ Sequential Reads of Ordered Keys (scannable)
- ❖ Columnar schema
- ❖ Built-in versioning
- ❖ Available (configurable to Consistent)
- ❖ Optimized for hundreds of nodes
- ❖ (*Digg, Twitter*)

# INTERMISSION



Let's Enjoy Some Art

OS -ef | grep

Ceci n'est pas une pipe.

# Document Datastore

---



## ❖ MongoDB

- ❖ <http://mongoid.org/>
- ❖ <http://mongomapper.com/> (rails3 branch)

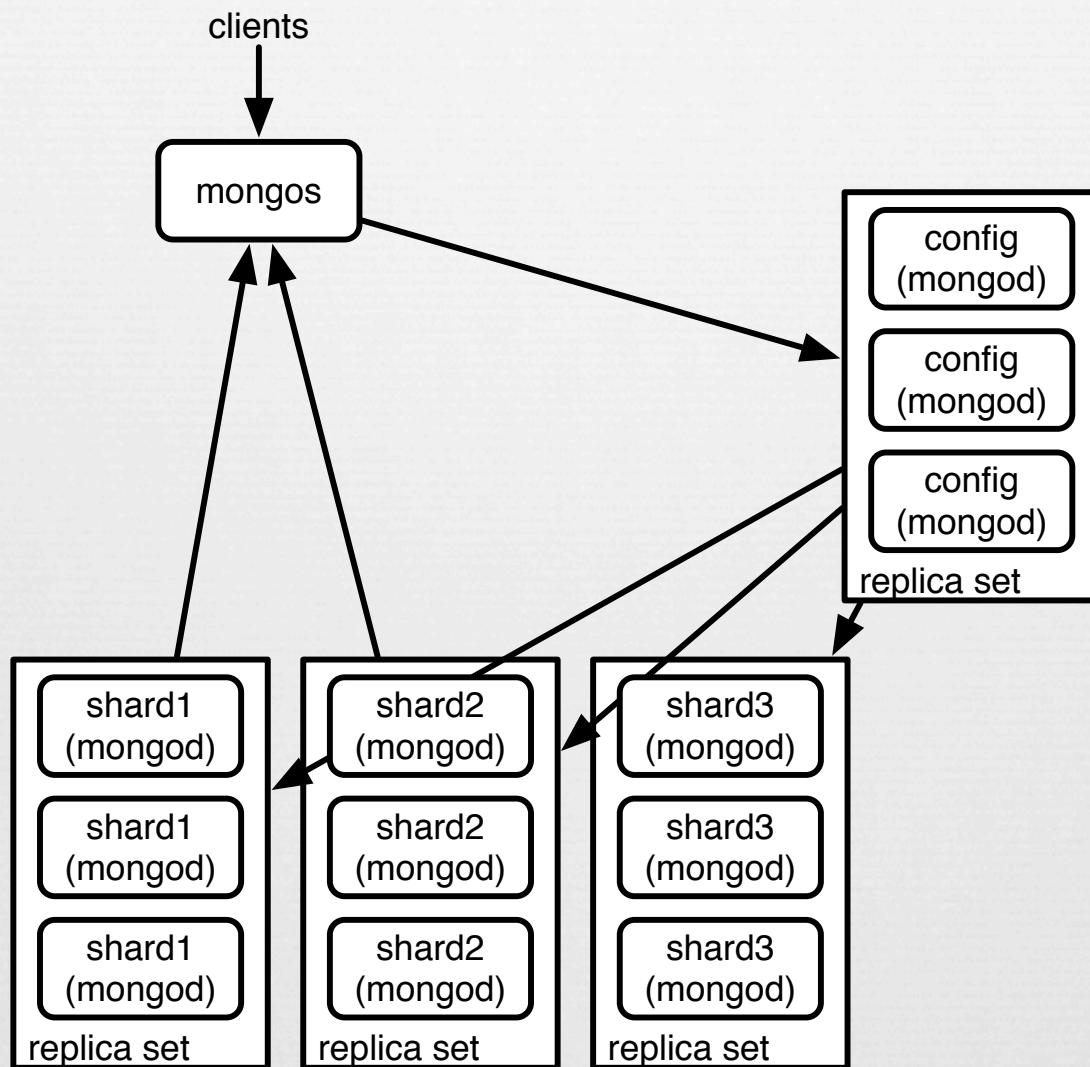
## ❖ CouchDB

- ❖ [http://github.com/couchrest/couchrest\\_model](http://github.com/couchrest/couchrest_model)
- ❖ [http://github.com/peritor/simply\\_stored](http://github.com/peritor/simply_stored)
- ❖ <http://tilgovi.github.com/couchdb-lounge/> (clustering)

# Document

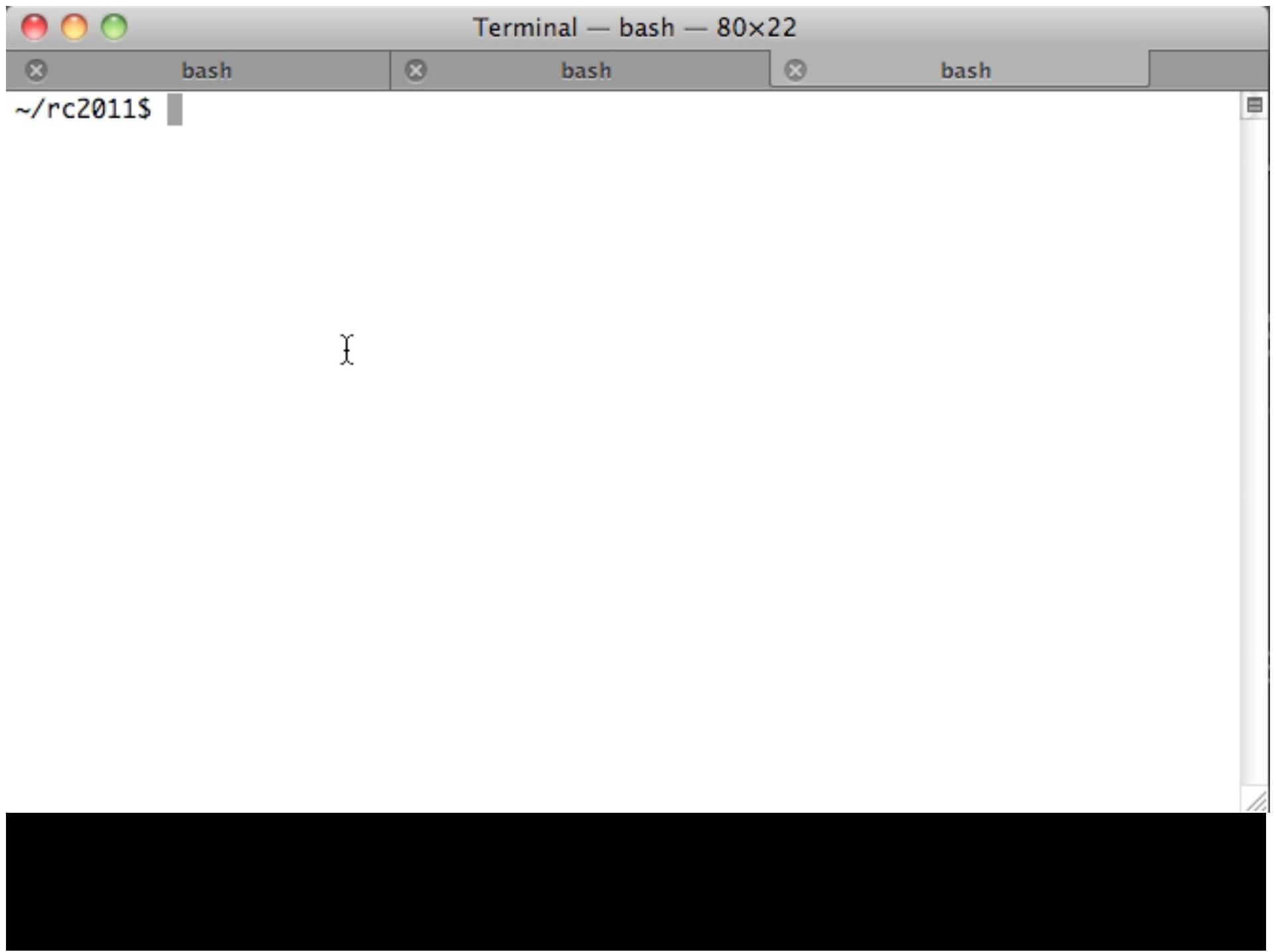


```
{  
    "_id" : ObjectId("4db7ca268e236e5bf9a52224"),  
    "_rev" : "2612672603",  
    "name" : "Sant Julià de Lòria",  
    "country" : "AD",  
    "timezone" : "Europe/Andorra",  
    "population" : 8022,  
    "location" : {  
        "latitude" : 42.46372,  
        "longitude" : 1.49129  
    }  
}
```



## Terminal — bash — 80x22

```
bash                                bash                                bash
~/rc2011$ rails c
Loading development environment (Rails 3.0.5)
ruby-1.9.2-p0 > include Cassandra::Constants
=> Object
ruby-1.9.2-p0 > store = Cassandra.new('CassandraObject')
=> #<Cassandra:2173056880, @keyspace="CassandraObject", @schema={}, @servers=["127.0.0.1:9160"]>
ruby-1.9.2-p0 > store.insert(:Customers, '1234', {'name' => 'Peter Griffin', 'buys' => 'beer'})
=> nil
ruby-1.9.2-p0 > ap store.get(:Customers, '1234')
{
  "buys" => "beer",
  "name" => "Peter Griffin"
}
=> nil
ruby-1.9.2-p0 > ap store.get(:Customers, '1234', 'name')
"Peter Griffin"
=> nil
ruby-1.9.2-p0 > exit
~/rc2011$
```



Terminal — ruby — 80x22

bash

ruby

bash

```
~/rc2011$ rails c
```

```
Loading development environment (Rails 3.0.5)
```

```
ruby-1.9.2-p0 > City.first
```

```
=> #<City _id: 4db7ca268e236e5bf9a52224, _type: nil, _id: BSON::ObjectId('4db7c  
a268e236e5bf9a52224'), name: "Sant Julià de Lòria", country: "AD", population: 8  
022, timezone: "Europe/Andorra">
```

```
ruby-1.9.2-p0 > City.first.location
```

```
=> #<Location _id: 4db7ca5e360b01c3d9000001, _type: nil, _id: BSON::ObjectId('4  
db7ca5e360b01c3d9000001'), latitude: 42.46372, longitude: 1.49129>
```

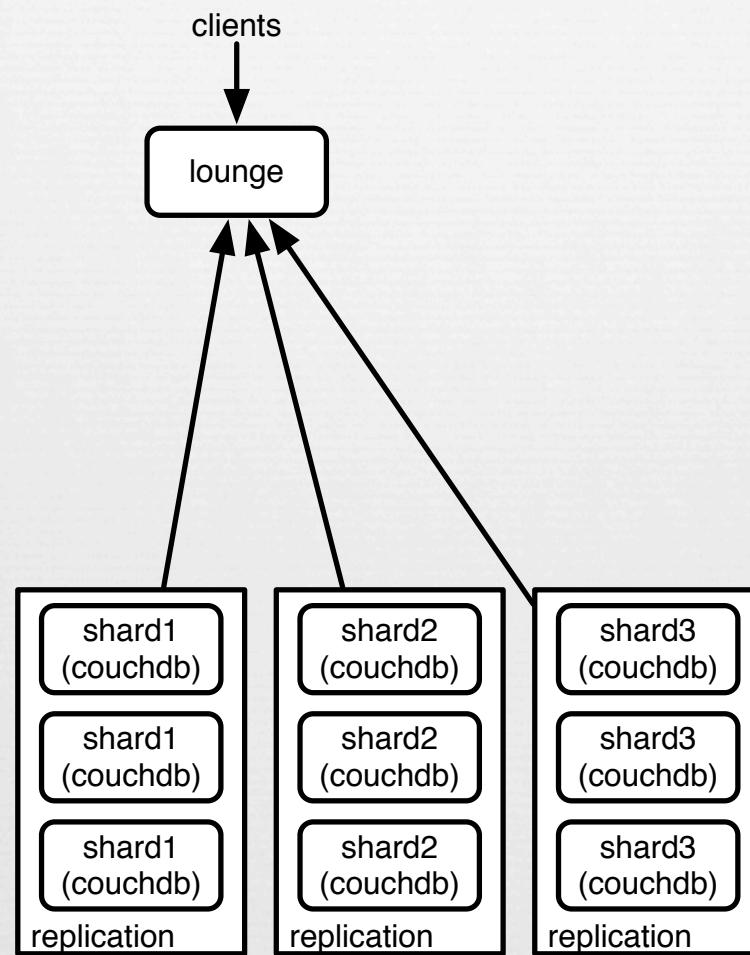
```
ruby-1.9.2-p0 > █
```

# Couch

---



- ❖ Interacting with Couch in Rails is similar to Mongo.  
The difference is a heavier reliance on map/reduce to  
create views.
- ❖ Futon (web console)
- ❖ Lounge (clustering, sharding)
- ❖ BigCouch (Dynamo-style NRW)



Apache CouchDB - Futon: Br

127.0.0.1:5984/\_utils/database.html?\_users

Overview → \_users

New Document Jump to: Document ID View: All documents Stale views

Security... Compact & Cleanup... Delete Database...

Key ▼	Value
"_design/_auth" ID: _design/_auth	{rev: "1-91285b0279dc582d8e1549c84c9c1406"}

Showing 1-1 of 1 row ← Previous Page | Rows per page: 10 | Next Page →

**CouchDB**  
relax

Tools

- Overview
- Configuration
- Replicator
- Status
- Test Suite

Recent Databases

\_users

Welcome to Admin Party!  
Everyone is admin. Fix this

Futon on Apache CouchDB 1.0.1

# Relax



```
$ sudo couchdb
```

Apache CouchDB 1.0.1 (LogLevel=info) is starting.

Apache CouchDB has started. **Time to relax.**

```
[info] [<0.31.0>] Apache CouchDB has started on
```

<http://127.0.0.1:5984/>

```
$ curl http://127.0.0.1:5984/
```

```
{"couchdb":"Welcome","version":"1.0.1"}
```

# Mongo v Couch

---

- ❖ Consistency Focused
- ❖ Master/Slave
- ❖ Ad-hoc queries
- ❖ Comfortable to SQL users
- ❖ Built to run on clusters
- ❖ Availability Focused
- ❖ Master/Master
- ❖ Mapreduce views
- ❖ Comfortable to client/server authors
- ❖ Runs on nearly anything

# Dynamo K/V Style

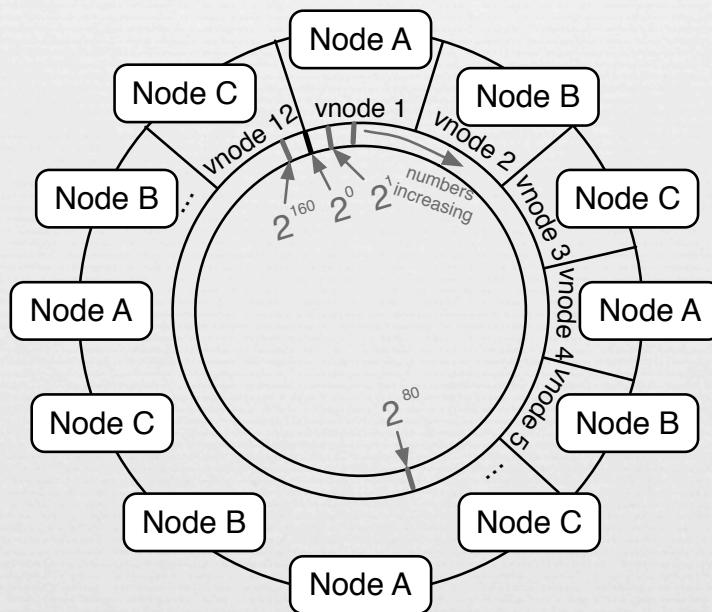


- ❖ Riak
  - ❖ Pretty “documenty”
  - ❖ Risky <https://github.com/aphyr/risky>
  - ❖ Ripple <http://seancribbs.github.com/ripple>
  - ❖ Riak Session <http://rubygems.org/gems/riak-sessions>

# “The Ring”



N=3  
Q=12



## Terminal — ruby — 80x22

```
bash          ruby          bash
mezone: "America/New_York">>,
[21] #<City _id: 4db7ca2a8e236e5bf9a6764e, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a6764e'), name: "Hampton", country: "US", population: 5119, ti
mezone: "America/New_York">>,
[22] #<City _id: 4db7ca2a8e236e5bf9a676d7, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a676d7'), name: "West Elkridge", country: "US", population: 28
734, timezone: "America/New_York">>,
[23] #<City _id: 4db7ca2a8e236e5bf9a6762b, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a6762b'), name: "Elkridge", country: "US", population: 22042,
timezone: "America/New_York">>,
[24] #<City _id: 4db7ca2a8e236e5bf9a676ac, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a676ac'), name: "Riverside", country: "US", population: 6507,
timezone: "America/New_York">>,
[25] #<City _id: 4db7ca2a8e236e5bf9a67684, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a67684'), name: "New Windsor", country: "US", population: 1358
, timezone: "America/New_York">>,
[26] #<City _id: 4db7ca2a8e236e5bf9a675e3, _type: nil, _id: BSON::ObjectId('4db7ca2a8e236e5bf9a675e3'), name: "Baltimore", country: "US", population: 610892
, timezone: "America/New_York">
]
=> nil
ruby-1.9.2-p0 >
```

The screenshot shows a software interface for editing Ruby on Rails code. The main window displays the file `images_controller.rb` with the following content:

```
1 class ImagesController < ApplicationController
2   def index
3   end
4
5 end
```

The code editor has tabs for `Gemfile`, `index.html.erb`, `images_controller.rb` (which is the active tab), and `routes.rb`. The status bar at the bottom shows "Line: 1 Column: 1", "Ruby on Rails", "Soft Tabs: 2", and "ImagesCon...".

To the right of the code editor is a sidebar showing the project structure of the `rc2011` application:

- `rc2011`
  - `app`
  - `config`
  - `config.ru`
  - `db`
  - `doc`
  - `Gemfile`
  - `Gemfile.lock`
  - `lib`
  - `log`
  - `public`
    - `Rakefile`
    - `README`
  - `script`
  - `test`
  - `tmp`
  - `vendor`

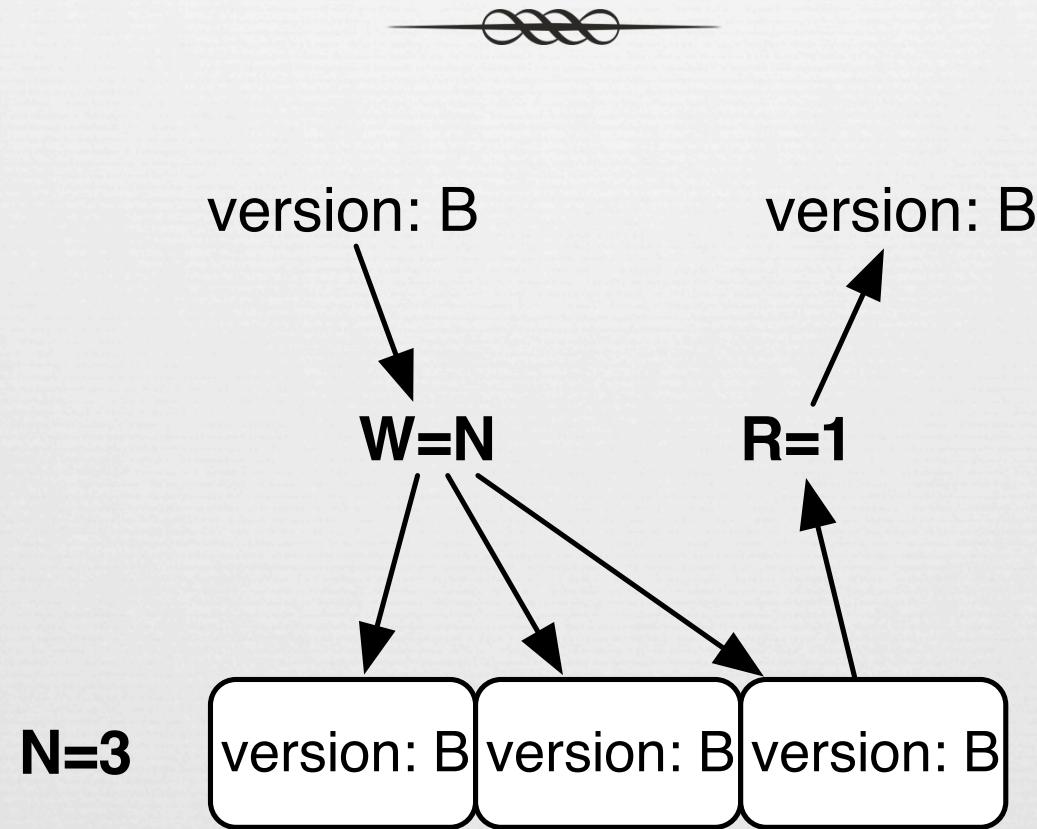
At the bottom of the sidebar are icons for creating new files and folders, settings, and help.

# N/R/W

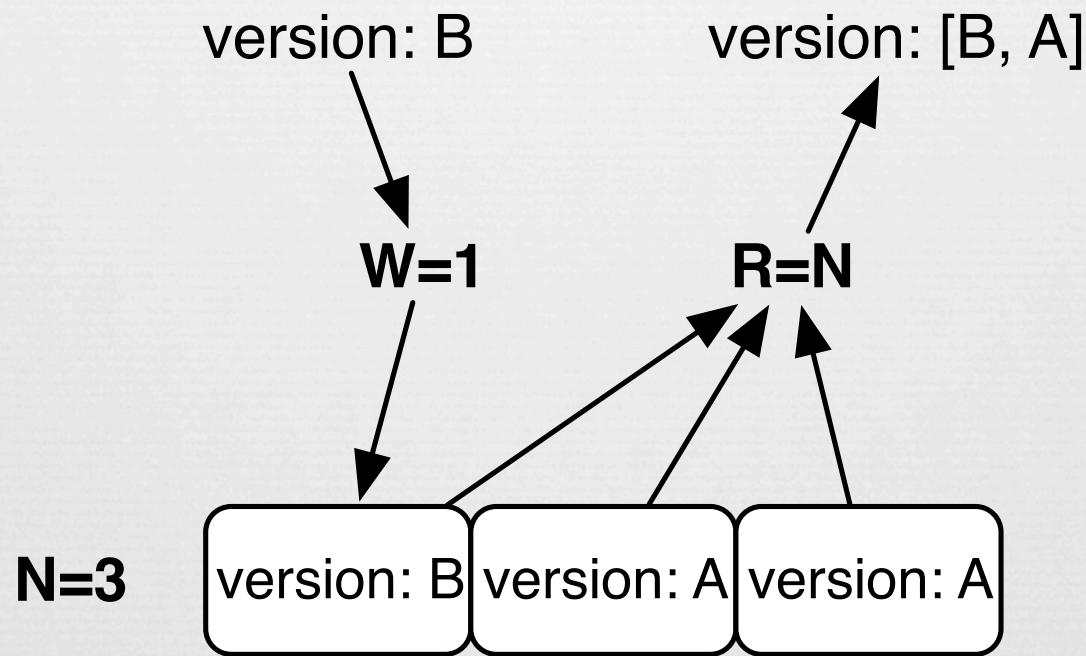


- ❖ CAP can't be beat – but it can be tweaked
- ❖ N/R/W
  - ❖ N = Nodes to write to (per bucket)
  - ❖ W = Nodes written to before success
  - ❖ R = Nodes read from before success
- ❖ What does this mean?
  - ❖ Support both CP and AP in one database

# Write Consistency

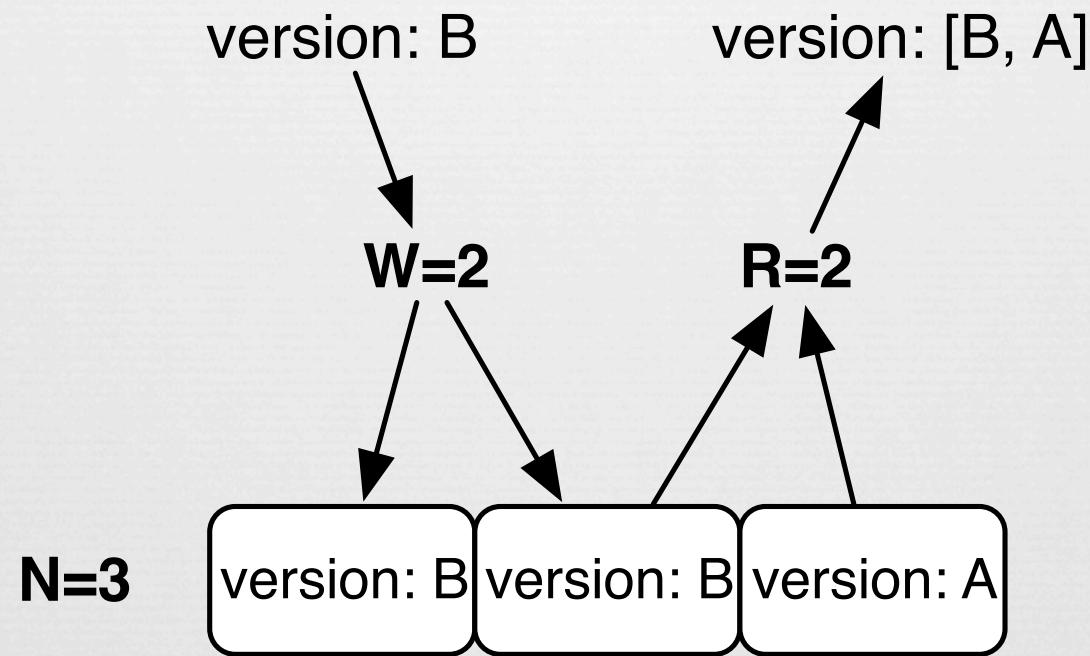


# Read Consistency

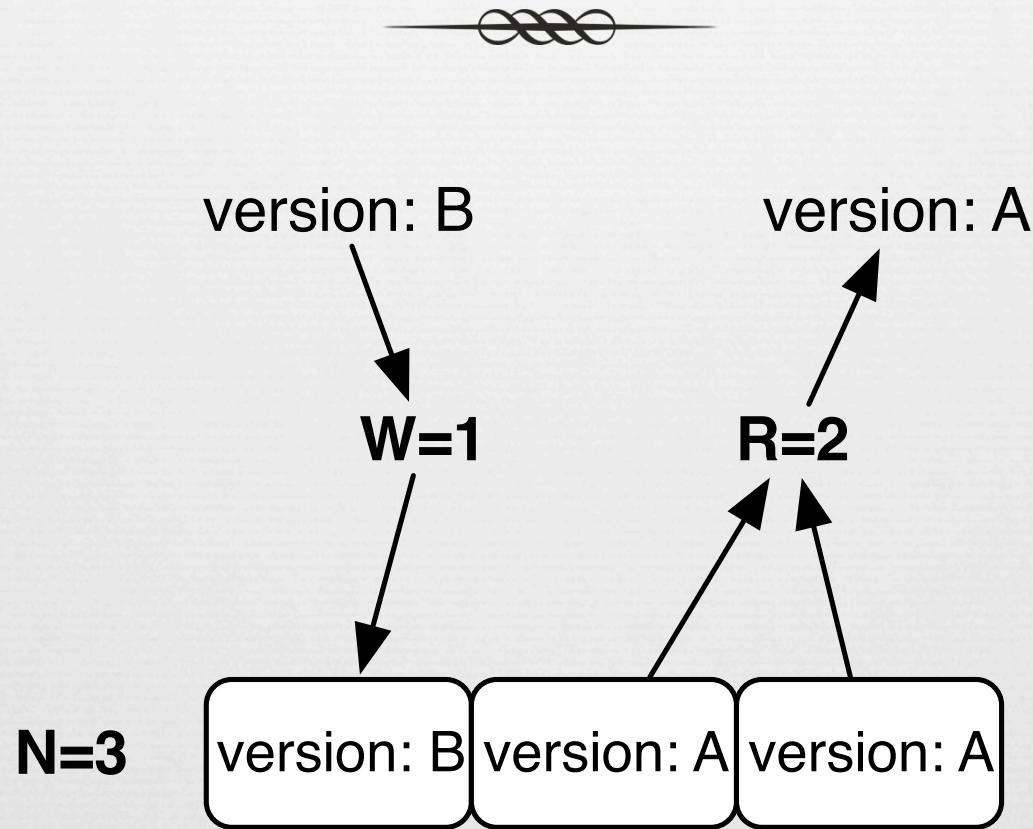


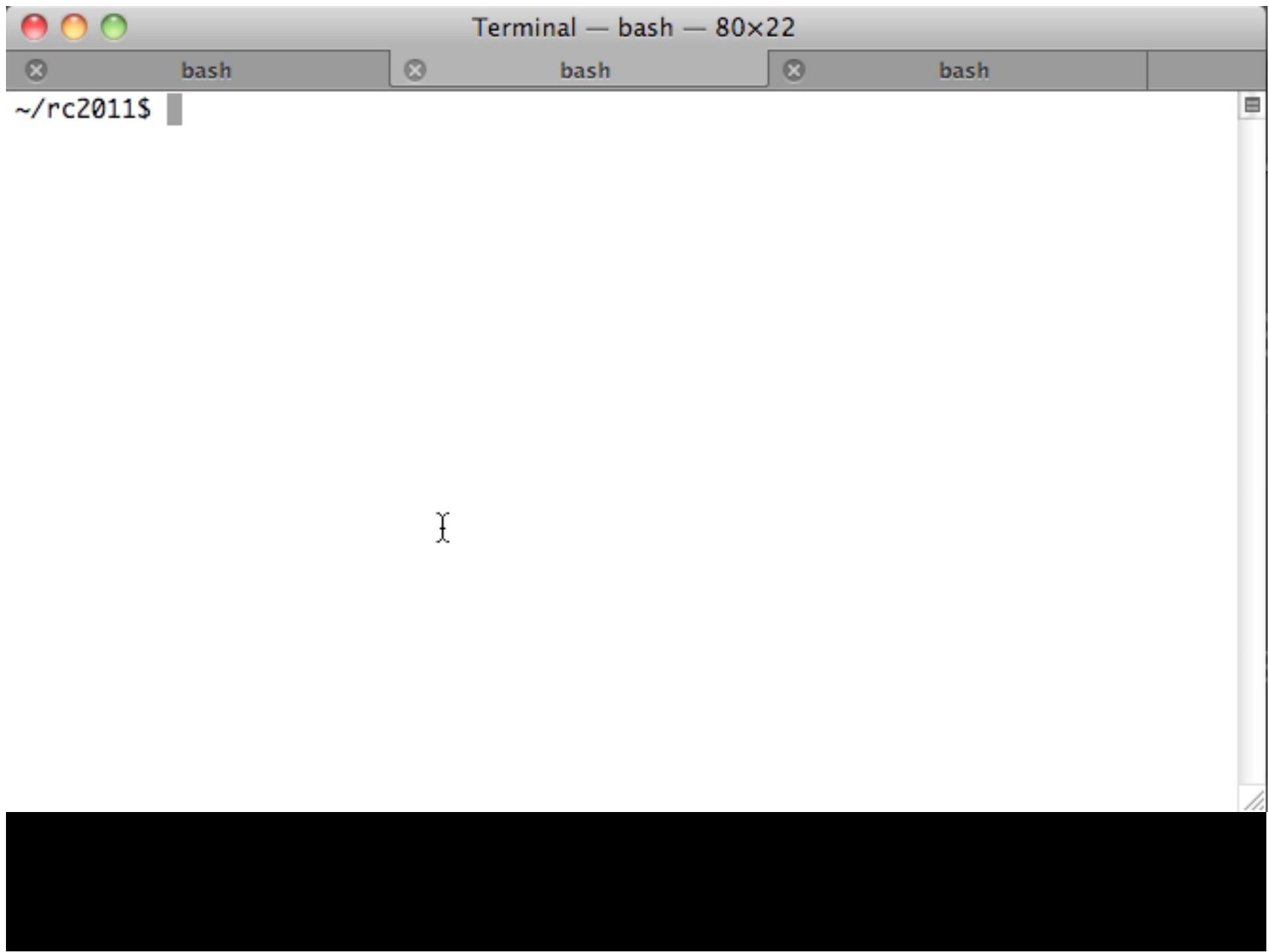
# Quorum

---



# Weak Consistency





# Key/Value Stores

---



- ❖ Memcached
- ❖ Kyoto Cabinet
- ❖ Redis
  - ❖ <http://github.com/ezmobius/redis-rb>
  - ❖ <http://github.com/nateware/redis-objects>
  - ❖ <http://github.com/jodosha/redis-store>
  - ❖ <http://github.com/defunkt/resque>
  - ❖ [http://www.paperplanes.de/2010/2/16/a\\_collection\\_of\\_redis\\_use\\_cases.html](http://www.paperplanes.de/2010/2/16/a_collection_of_redis_use_cases.html)

# Redis Knows Sets



- ≈ redis.sadd 'person', 'Eric'
- ≈ redis.sadd 'person', 'Jim'
- ≈ redis.smembers 'person'
  - ≈ ['Eric', 'Jim']
- ≈ redis.sadd 'owns\_pet', 'Eric'
- ≈ redis.sinter 'person', 'owns\_pet'
  - ≈ ['Eric']

Gemfile — rc2011

\* Gemfile \* index.html.erb \* images\_controller.rb \* routes.rb

```
1 source 'http://rubygems.org'  
2  
3 gem 'rails', '3.0.5'  
4 gem 'awesome_print'  
5 gem 'mysql2'  
6 gem 'pg'  
7 gem 'pg_search'  
8 gem 'thrift'  
9 gem 'cassandra'  
10 gem 'mongoid'  
11 gem 'bson_ext'  
12 gem 'ripple'  
13  
14 # Use unicorn as the web server  
15 # gem 'unicorn'  
16  
17 # Deploy with Capistrano  
18 # gem 'capistrano'  
19  
20 # To use debugger (ruby-debug for Ruby 1.8.7+, ruby-debug19  
# for Ruby 1.9.2+)
```

Line: 13 Column: 1 | L Ruby | Soft Tabs: 2 | — |

rc2011  
► app  
► config  
► config.ru  
► db  
► doc  
► Gemfile  
► Gemfile.lock  
► lib  
► log  
► public  
► Rakefile  
► README  
► script  
► test  
► tmp  
► vendor

The screenshot shows a software interface for editing Ruby on Rails code. The main window displays the contents of `shorts_controller.rb`. The code defines a controller named `ShortsController` that inherits from  `ApplicationController`. It contains two methods: `new` and `create`, both enclosed in an `end` block. The code is color-coded, with class names in blue and method names in orange. The interface includes a toolbar at the top with icons for file operations, a sidebar on the right showing the project structure, and status bars at the bottom providing information about the current file and the Ruby on Rails environment.

```
class ShortsController < ApplicationController
  def new
  end

  def create
  end
end
```

Project Structure:

- rc2011
  - app
  - config
  - config.ru
  - db
  - doc
  - Gemfile
  - Gemfile.lock
  - lib
  - log
  - public
    - Rakefile
    - README
  - script
  - test
  - tmp
  - vendor

Bottom Status Bar:

- Line: 3 Column: 6
- Ruby on Rails
- Soft Tabs: 2
- new

# Graph Datastore

---



¤ Neo4j

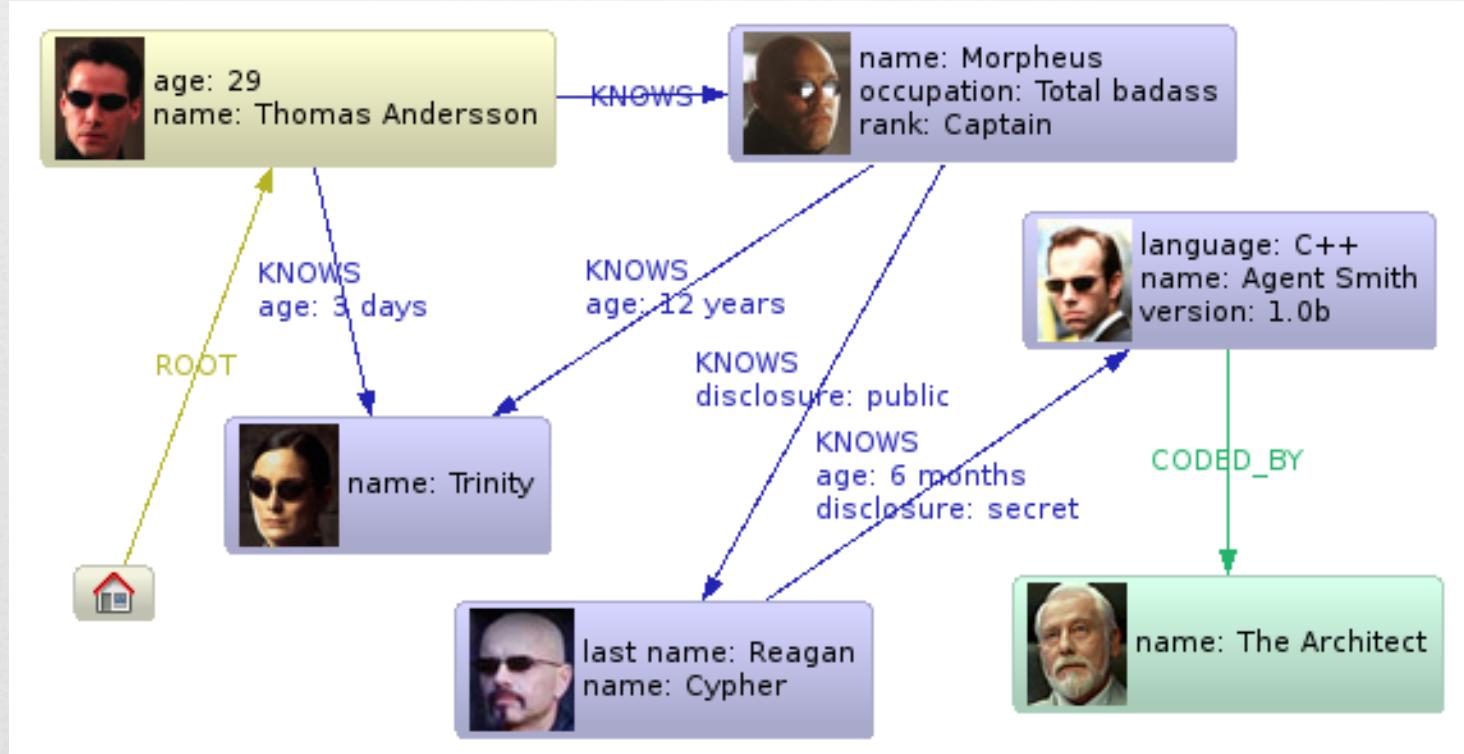
¤ Neo4j.rb <http://github.com/andreasronge/neo4j>

¤ Neography <http://github.com/maxdemarzi/neography>

¤ FlockDB

¤ FlockDB client <http://github.com/twitter/flockdb-client>

# The Matrix



# Neo4j Shell (JRuby)



```
≈ neo4j-sh (3, Morpheus)$ cd 4
≈ neo4j-sh (4)$ set name Cypher
≈ neo4j-sh (4, Cypher)$ mkrel -ct KNOWS
≈ neo4j-sh (4, Cypher)$ ls -rd out
≈ (me) --<KNOWS>--> (5)
≈ neo4j-sh (4, Cypher)$ cd 5
≈ neo4j-sh (5)$ set name "Agent Smith"
≈ neo-sh (5, Agent Smith)$ mkrel -cvt CODED_BY
≈ Node (6) created
≈ Relationship <6, CODED_BY> created
≈ neo4j-sh (5, Agent Smith)$ cd 6
≈ neo4j-sh (6)$ set name "The Architect"
≈ neo4j-sh (6, The Architect)$
```

# DB Tools



❖ DataMapper

❖ <http://github.com/datamapper/dm-rails>

❖ Chimera

❖ <http://github.com/benmyles/chimera>

# Try Them All!



- ❖ Why not? It's a big decision.
- ❖ Download the example from this talk @

MySQL/ Postgres	CA	relational	bank
Hbase	CP	columnar	search engine
Cassandra	AP	columnar	SETI
Mongo	CP	document	insurance
Couch	AP	document	mobile interfaces
Neo4j	CA	graph	genealogy
FlockDB	AP	graph	social network
Riak	AP	key/value	huge catalog
Memcached/ Kyoto Cabinet/ Redis	AP	key/value	session data

# Education



You know what? Just use RDBMS...

# Sites

---



- ❖ <http://nosql-database.org/>
  - ❖ A great list
- ❖ <http://sevenweeks.org/>
  - ❖ The book website (it's a wiki!)
- ❖ <https://github.com/coderoshi/holy-grail-dbs>
  - ❖ The project
  - ❖ The slides

# Papers

---

- ❖ Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services
  - ❖ [people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf](http://people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf)
- ❖ Dynamo: Amazon's Highly Available Key-value Store
  - ❖ [allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf](http://allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf)
- ❖ Bigtable: A Distributed Storage System for Structured Data
  - ❖ [labs.google.com/papers/bigtable-osdi06.pdf](http://labs.google.com/papers/bigtable-osdi06.pdf)
- ❖ MapReduce: Simplified Data Processing on Large Clusters
  - ❖ [labs.google.com/papers/mapreduce.html](http://labs.google.com/papers/mapreduce.html)

# Papers

---

- ❖ Megastore: Providing Scalable, Highly Available Storage for Interactive Services
  - ❖ [http://www.cidrdb.org/cidr2011/Papers/  
CIDR11\\_Paper32.pdf](http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf)
- ❖ Design and Evaluation of a Continuous Consistency Model for Replicated Services
  - ❖ [http://citeseerx.ist.psu.edu/viewdoc/download?  
doi=10.1.1.34.7743&rep=rep1&type=pdf](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.7743&rep=rep1&type=pdf)
- ❖ Indexed Database API
  - ❖ <http://www.w3.org/TR/IndexdB>

# PS: Get a Mac



- ❖ brew install mysql
- ❖ brew install mongodb
- ❖ brew install postgresql
- ❖ brew install couchdb
- ❖ brew install hbase
- ❖ brew install memcached
- ❖ brew install cassandra
- ❖ brew install kyoto-cabinet
- ❖ brew install riak
- ❖ brew install redis