

# CouchDB and Rails on the Cloud

Rocky Jaiswal

*McKinsey & Company*

2<sup>nd</sup> IndicThreads.com Conference On  
Cloud Computing

3,4 JUNE 2011



PUNE, INDIA


# ABOUT ME

Rocky Jaiswal

Daytime job – Software Architect at McKinsey & Co

Programmer / Agilist at heart

Have been programming for almost 9 years, plan to do it for a long loooong time

I  Java/Ruby/JRuby/jQuery and anything to do with web application development

Want to build good looking, scalable and performing websites that help people

Blog – [www.rockyj.in](http://www.rockyj.in)

Twitter – [www.twitter.com/whatsuprocky](http://www.twitter.com/whatsuprocky)

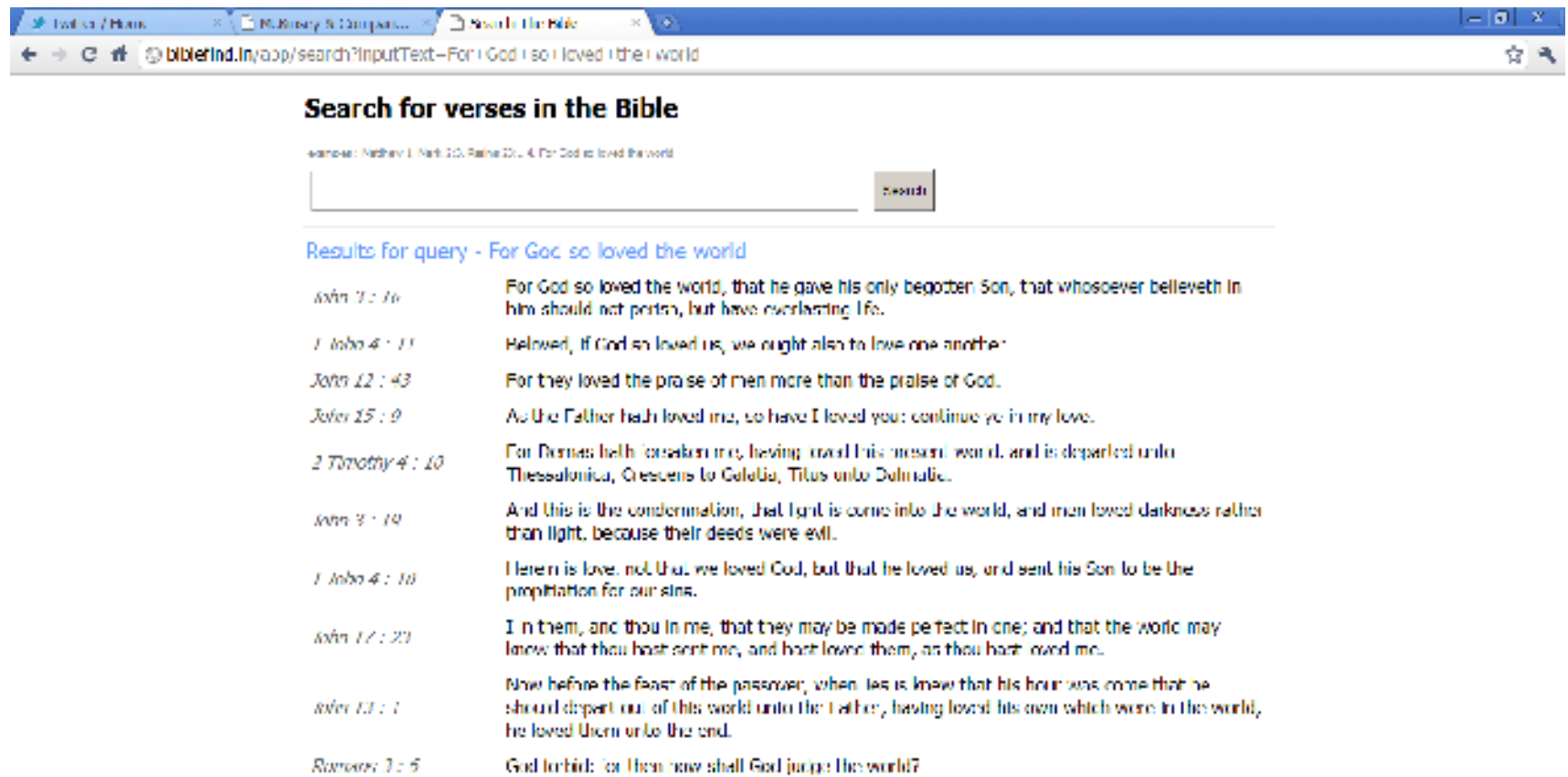


# Why we need the cloud

- I am a developer. Don't want the hassle to maintain infrastructure.
- We are a small organization. We want cheap and flexible infrastructure.
- I / we want to scale easily. Be it scale up or scale down.
- \*Choice of technology also determines how easily you can scale. e.g. Use of NoSQL instead of a RDBMS



# A WORKING EXAMPLE



Search for verses in the Bible

Keywords: Matthew 1, Mark 10, Luke 10, 4, For God so loved the world

Results for query - For God so loved the world

<i>John 1 : 16</i>	For God so loved the world, that he gave his only begotten Son, that whosoever believeth in him should not perish, but have everlasting life.
<i>1 John 4 : 11</i>	Beloved, if God so loved us, we ought also to love one another.
<i>John 12 : 43</i>	For they loved the praise of men more than the praise of God.
<i>John 15 : 9</i>	As the Father hath loved me, so have I loved you: continue ye in my love.
<i>2 Timothy 4 : 10</i>	For Demas hath forsaken me, having loved this present world, and is departed unto Thessalonica, Crescens to Galatia, Titus unto Dalmatia.
<i>John 3 : 19</i>	And this is the condemnation, that light is come into the world, and men loved darkness rather than light, because their deeds were evil.
<i>1 John 4 : 10</i>	Herein is love, not that we loved God, but that he loved us, and sent his Son to be the propitiation for our sins.
<i>John 17 : 23</i>	I in them, and thou in me, that they may be made perfect in one; and that the world may know that thou hast sent me, and hast loved them, as thou hast loved me.
<i>John 13 : 1</i>	Now before the feast of the passover, when Jesus knew that his hour was come that he should depart out of this world unto the Father, having loved his own which were in the world, he loved them unto the end.
<i>Romans 7 : 5</i>	God forbid: for then how shall God judge the world?

<http://biblefind.in>



# HELLO COUCHDB

Apache CouchDB is a document-oriented database that can be queried and indexed in a MapReduce fashion using JavaScript.

CouchDB also offers incremental replication with bi-directional conflict detection and resolution.

CouchDB provides a RESTful JSON API than can be accessed from any environment that allows HTTP requests.



+ It offers an easy introduction to the world of NoSQL



# HELLO COUCHDB – DOCUMENT ORIENTED

A different way to model your data.

Data stored in documents

Think of it as a de-normalized table row

EMPLOYEE MODEL  
MULTIPLE

C \_\_\_\_\_ C \_\_\_\_\_  
I \_\_\_\_\_ I \_\_\_\_\_  
T \_\_\_\_\_ T \_\_\_\_\_  
Y \_\_\_\_\_ Y \_\_\_\_\_  
P \_\_\_\_\_ P \_\_\_\_\_

NAME	TITLE	EMPLOYEE	SALARY

SUMMARY

NAME	TITLE	SALARY	EMPLOYEE



# HELLO COUCHDB - MAPREDUCE

MapReduce – Divide and Rule for programmers

How would you count the occurrences of each word in a book given a group of helpers



# HELLO COUCHDB – JSON/HTTP

When we talk of databases we talk of drivers ☺

CouchDB's protocol is HTTP

The data exchange + storage language is JSON



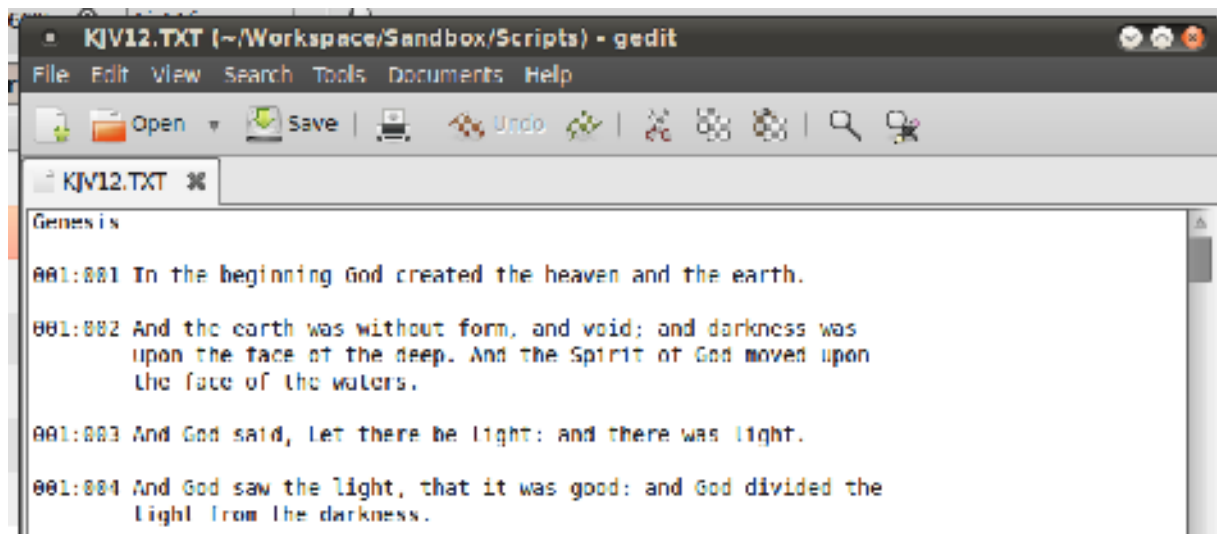
```
{  
  "Subject" : "I like JSON",  
  "Author" : "Rocky",  
  "Tags" : [  
    "Web", "Programming", "Data Exchange"  
  ]  
}
```

And the queries are written in JavaScript

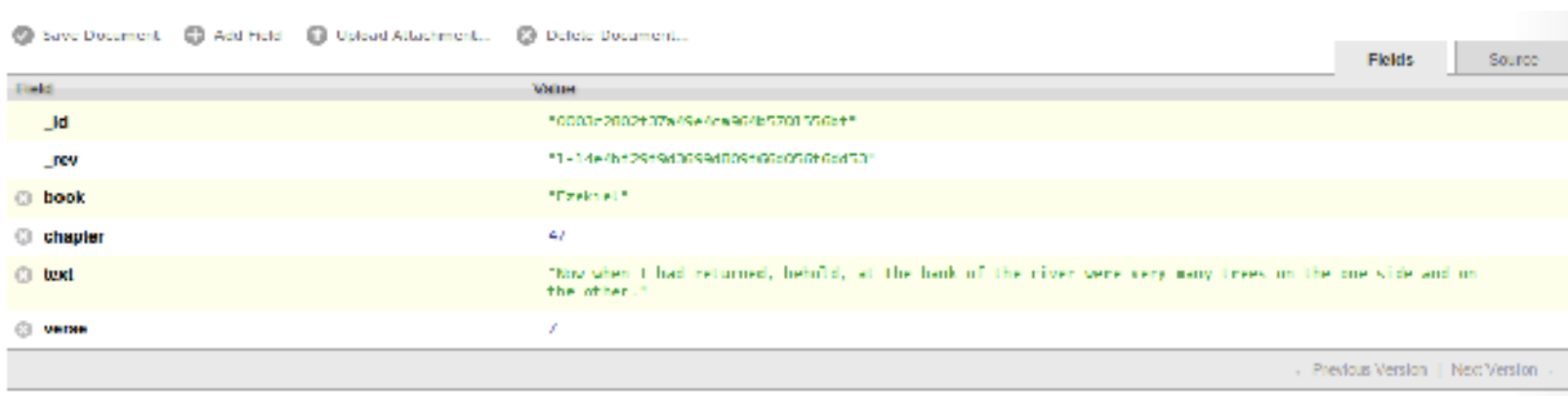




# PARSING THE BIBLE AND STORING IT IN COUCHDB



```
KJV12.TXT (~/Workspace/Sandbox/Scripts) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
KJV12.TXT
Genesis
001:001 In the beginning God created the heaven and the earth.
001:002 And the earth was without form, and void; and darkness was
upon the face of the deep. And the Spirit of God moved upon
the face of the waters.
001:003 And God said, Let there be light: and there was light.
001:004 And God saw the light, that it was good: and God divided the
light from the darkness.
```



save Document Add Field Upload Attachment... Delete Document...

Fields	Source
fields	value
<input type="checkbox"/> _id	"0001-0002+07a45a7e9a30b070110001"
<input type="checkbox"/> _rev	"1-14a7b12694009d009e0000000000"
<input checked="" type="checkbox"/> book	"Genesis"
<input checked="" type="checkbox"/> chapter	47
<input checked="" type="checkbox"/> text	"Now when I had returned, behold, at the bank of the river were very many trees on the one side and on the other."
<input checked="" type="checkbox"/> version	/

Previous Version | Next Version



# CREATING VIEWS IN COUCHDB

Views are like queries. Hmmm... more like Stored Procedures

Views are expressed as Map + Reduce functions written in JavaScript

For example my view to query all the verses –

```
{
  "lookup": {
    "map": "function(doc){
      if (doc.book && doc.chapter && doc.verse && doc.text){
        key = [doc.book, doc.chapter, doc.verse];
        emit(key, doc.text);
      }
    }"
  }
}
```

CouchDB runs the function for every document in DB and stores results in a B-Tree.



# THE COUCHREST GEM

**CouchRest** lightly wraps CouchDB's HTTP API, managing JSON serialization, and remembering the URI-paths to CouchDB's API endpoints so you don't have to.

```
@db = CouchRest.database!  
("http://127.0.0.1:5984/the_bible")
```

```
@db.save_doc(  
{:key => 'value', 'another key' => 'another value'}  
)
```



# THE RAILS APPLICATION

So our back-end is set

We only need a front-end now

Nothing much needs to be done

1 Controller

1 View

Some jQuery for autocomplete



# REGEX NIGHTMARES

Matthew 1 – **One whole chapter**

Mark 2:3 – **One verse**

Psalms 23:1-4 – **A set of verses**

For God so loved the world – **Free Text**

```
/Everybody stand back/  
i know regular expressions
```



# LUCENE INTEGRATION

Couchdb-lucene (<https://github.com/rnewson/couchdb-lucene>)

## Java project

CouchDB View –

```
@db.save_doc({
  "_id" => "_design/lucene",
  :fulltext => {
    :by_text => {
      :index => "function(doc) { var ret=new Document(); ret.add(doc.text);
                return ret }"
    }
  }
})
```



# LUCENE INTEGRATION

## CONTD..

CouchDB config -

```
[external]
```

```
fti=python /home/rocky/Apps/couchdb-lucene-0.7-  
SNAPSHOT/tools/couchdb-external-hook.py
```

```
[httpd_db_handlers]
```

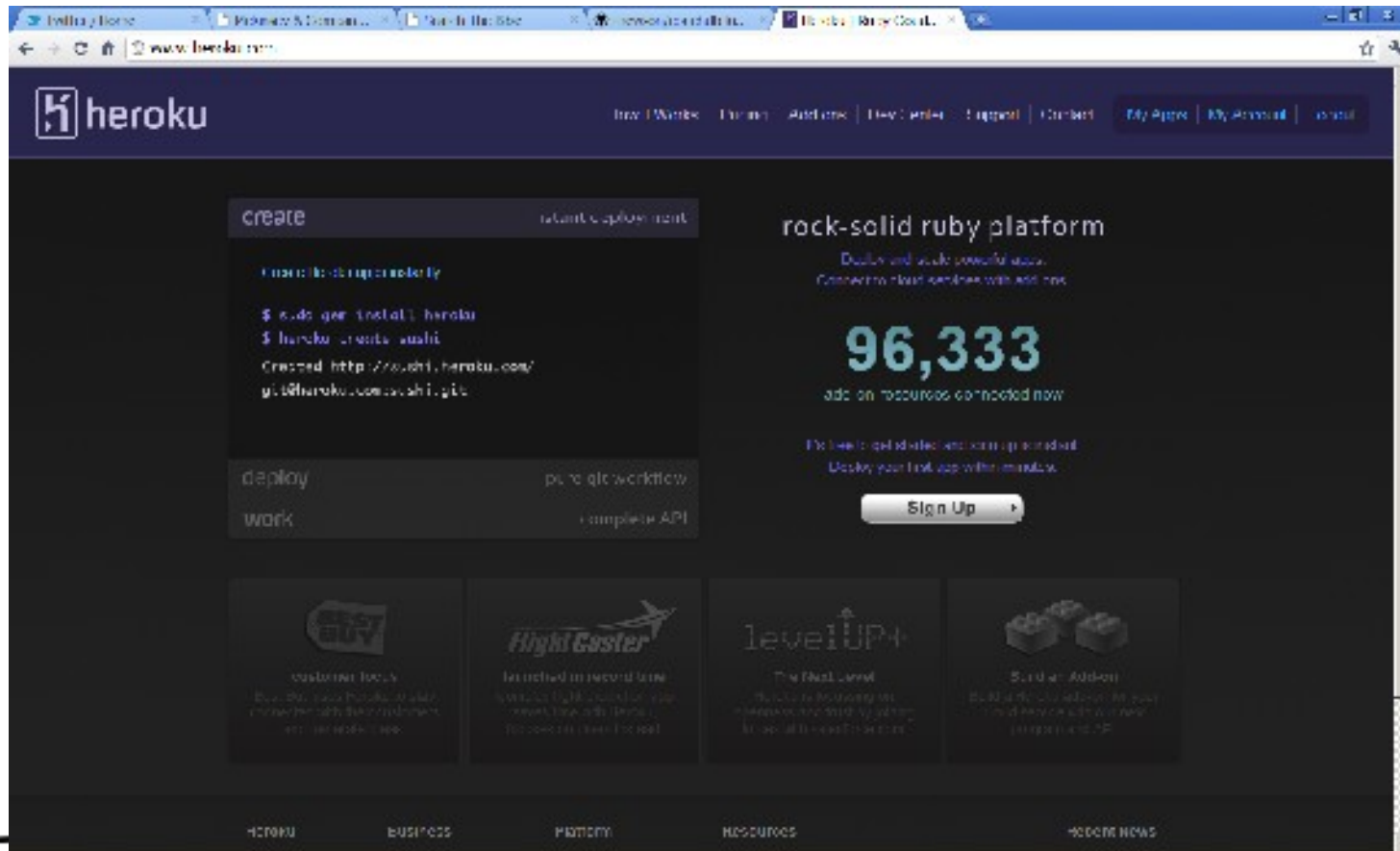
```
_fti = {couch_httpd_external, handle_external_req,  
<<"fti">>}
```



# USING HEROKU

Rails application hosting provider

**Free** for 1 "Dyno" + 1 Shared database



The screenshot shows the Heroku website homepage. At the top, there is a navigation bar with links for Home, Work, Pricing, Add-ons, Dev Center, Support, Contact, My Apps, My Account, and Logout. The main content area features a 'create' button with a 'rails capture track' label, a 'deploy' button with a 'push to git workflow' label, and a 'work' button with a 'complete API' label. The central focus is the 'rock-solid ruby platform' section, which includes the text 'Deploy and scale powerful apps. Connect to cloud services with add-ons.' and a large number '96,333' representing the number of add-on resources connected now. Below this is a 'Sign Up' button. The footer contains logos for customer focus, FlightCaster, levelUP+, and Build an Add-on, along with navigation links for Heroku, Business, Platform, Resources, and Recent News.



# USING HEROKU CONTD..

```
create instant deployment

Create Heroku apps instantly:

$ sudo gem install heroku
$ heroku create sushi

Created http://sushi.heroku.com/
git@heroku.com:sushi.git

deploy
work
```

```
create instant deployment
deploy pure git workflow

Full deployment is simply a git push.

$ git push heroku master
-----> Heroku receiving push
-----> Rails app detected
-----> Launching..... done
        http://sushi.heroku.com deployed

work complete API
```



# USING SLICEHOST

## Our Plans

Plan	RAM	Storage	I/O	Monthly Cost	
256 slice	256MB	10GB	100GB	\$20	<a href="#">SIGN UP</a>
384 slice	384MB	15GB	225GB	\$25	<a href="#">SIGN UP</a>
512 slice	512MB	20GB	300GB	\$30	<a href="#">SIGN UP</a>
768 slice	768MB	30GB	450GB		
1GB slice	1024MB	40GB	600GB		
1.5GB slice	1536MB	60GB	900GB		
2GB slice	2048MB	80GB	1200GB		
3GB slice	3072MB	120GB	1800GB		
4GB slice	4096MB	160GB	2500GB		
8GB slice	8192MB	320GB	5000GB		
15.5GB slice	16384MB	620GB	25000GB		

## Included in Every Slice™

- 64 bit and 32 bit Linux images available
- Full root access and rebooting
- Choice of Linux distros
- Dedicated IP address and tier-1 redundant bandwidth
- RAD-10 disk storage
- Reserved RAM
- Guaranteed CPU share and more when available

```
login as: rockyj
rockyj@          password:
Linux rockyj couchdb 2.6.35.1 rscldud #1 SMP Wed Aug 11 18:40:09 UTC 2010 i686 g
NT/TLinux
Ubuntu 10.04 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/
Last login: Tue May 10 05:29:38 2011 from
rockyj@rockyj-couchdb:~$ ls -l
total 8
drwxr-xr-x 3 rockyj users 4096 May  9 11:56 Apps
drwxr-xr-x 2 rockyj users 4096 May  9 12:45 Scripts
rockyj@rockyj-couchdb:~$ cd Apps/
rockyj@rockyj-couchdb:~/Apps$ ls -l
total 4
drwxr-xr-x 0 rockyj users 4096 May  9 12:02 couchdb-lucene-0.7-SNAPSHOT
rockyj@rockyj-couchdb:~/Apps$ cd couchdb-lucene-0.7-SNAPSHOT/bin
rockyj@rockyj-couchdb:~/Apps/couchdb-lucene-0.7-SNAPSHOT/bin$ ls -l
total 12
-rwxr--r-- 1 rockyj users 26 May  9 11:56 kill_ppid
-rw      1 rockyj users 570 May 10 05:30 nohup.out
-rwxr--r-- 1 rockyj users 413 May  9 11:56 run
rockyj@rockyj couchdb:~/Apps/couchdb lucene 0.7 SNAPSHOT/bin$
```



# PUTTING IT OUT IN THE BIG BAD WORLD

Buy a domain name from <http://www.godaddy.com>

Or a domain provider of your choice

In Heroku add the domain name

Add the Zerigo add-on in Heroku

In godaddy's admin console point your nameservers to  
Zerigo's name servers

Wait ...



# SCALING

Heroku makes scaling dead easy



# SCALING

Heroku makes scaling dead easy (if we were using SQL)

**Dedicated Databases**

Dedicated databases are available for large scale production applications. All plans have a 2TB database max. Features include consistent high performance, direct pool access, stored procedures, and more.

						
<a href="#">Read more about the new database offerings.</a>	<b>Ronin</b>	<b>Fugu</b>	<b>Ika</b>	<b>Zilla</b>	<b>Baku</b>	<b>Mecha</b>
	16 connections	20 connections	50 connections	100 connections	200 connections	400 connections
	1.7 GB RAM	1.7 GB RAM	7.5 GB RAM	17 GB RAM	34 GB RAM	60 GB RAM
	1 comp. unit	5 comp. unit	1 comp. unit	6.0 comp. unit	13 comp. unit	20 comp. unit
	<b>\$ 200</b>	<b>\$ 400</b>	<b>\$ 800</b>	<b>\$ 1,600</b>	<b>\$ 3,200</b>	<b>\$ 6,400</b>
	<b>Add</b>	<b>Add</b>	<b>Add</b>	<b>Add</b>	<b>Add</b>	<b>Add</b>



# SCALING

In NoSQL world, replication is a first class citizen

Apache CouchDB Replicator

Replicate changes from:

Local database: test\_write\_dbwith\_slashes

Remote database: http://

In:

Local database: test\_write\_dbwith\_slashes

Remote database: http://

Replicate

Event

Event
No replication

CouchDB

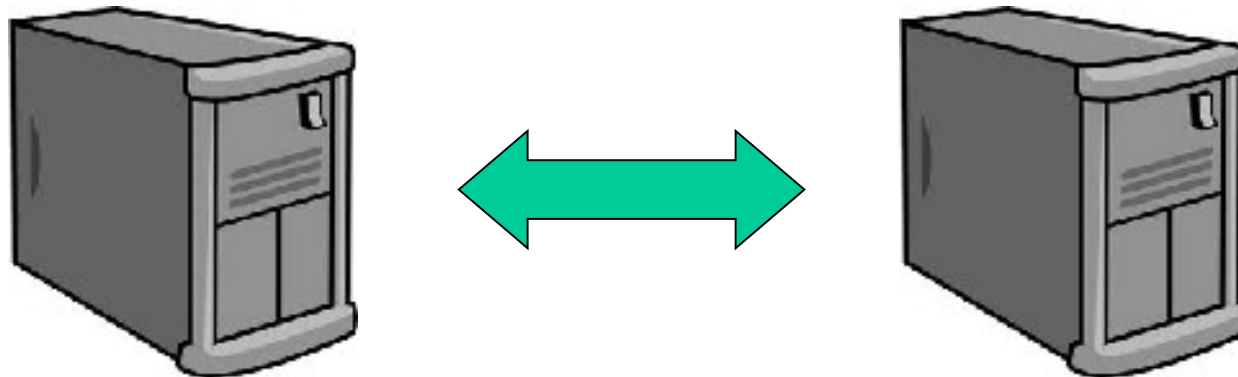
Tools

- Overview
- Configuration
- Replicator

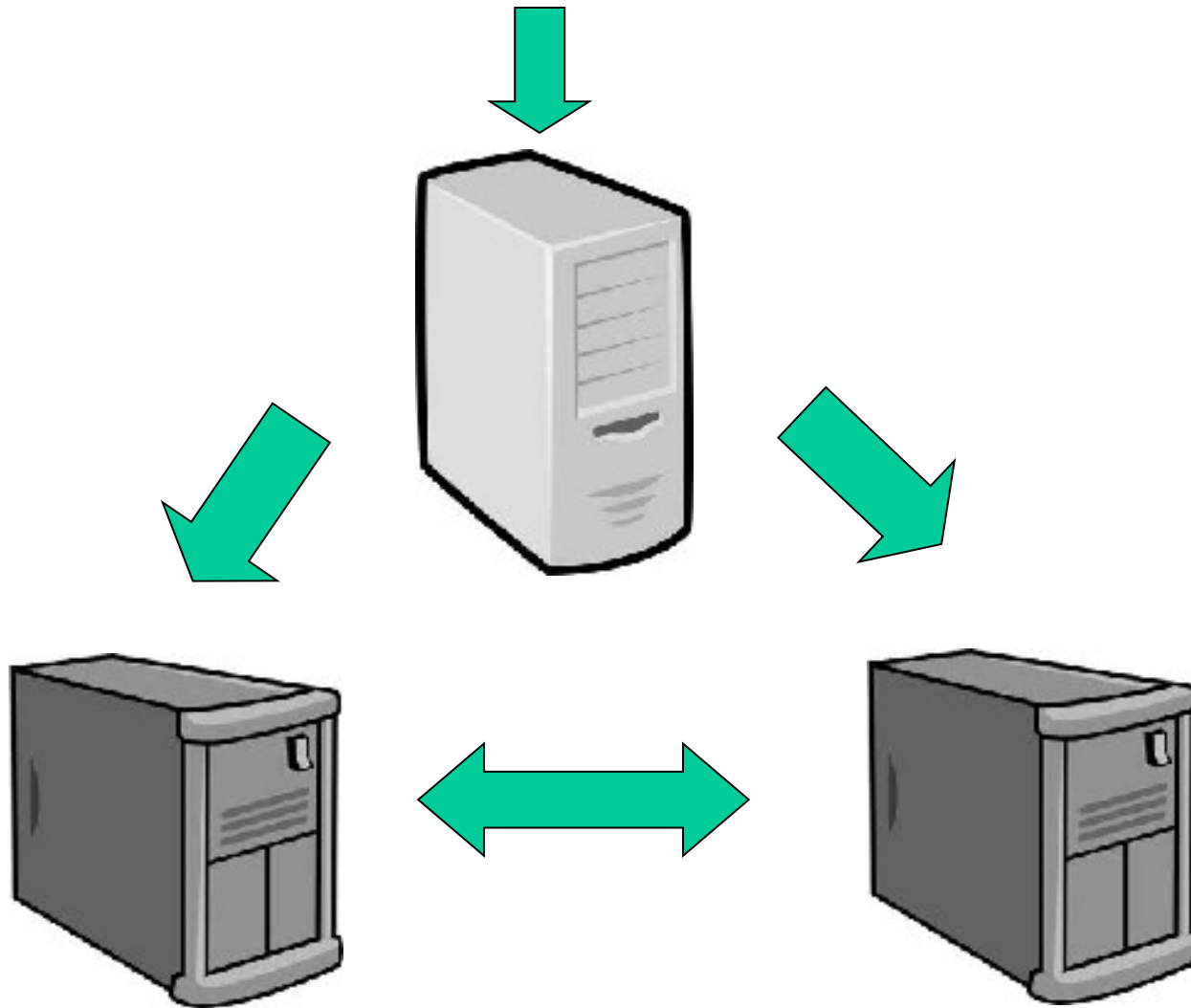
```
POST /_replicate {  
  "source": "a", "target": "b", "continuous": "true"  
}
```



# SCALING (HOT BACKUP)



# SCALING WITH A DUMB PROXY





# SCALING WITH COUCHDB LOUNGE

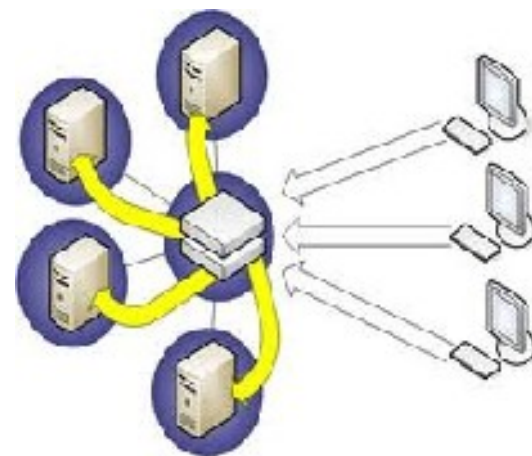
Have a look at CouchDB Lounge

It consists of –

- a dumb proxy that is a module for nginx
- a smart proxy that distributes work

All in all, make a cluster –

- Have continuous replication
- Use Lounge to distribute load



# WHEN NOT TO USE COUCHDB

When the number of writes far exceed the number of reads (plus the data volume is very high)

- This would create a bottleneck for replication
- And you may encounter more conflicts

When you need ad-hoc queries

- You cannot use the power of views in this case

Use CouchDB's brother MongoDB in this case.



# THANKS

