CouchDB and Rails on the Cloud

Rocky Jaiswal

McKinsey & Company

2nd 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

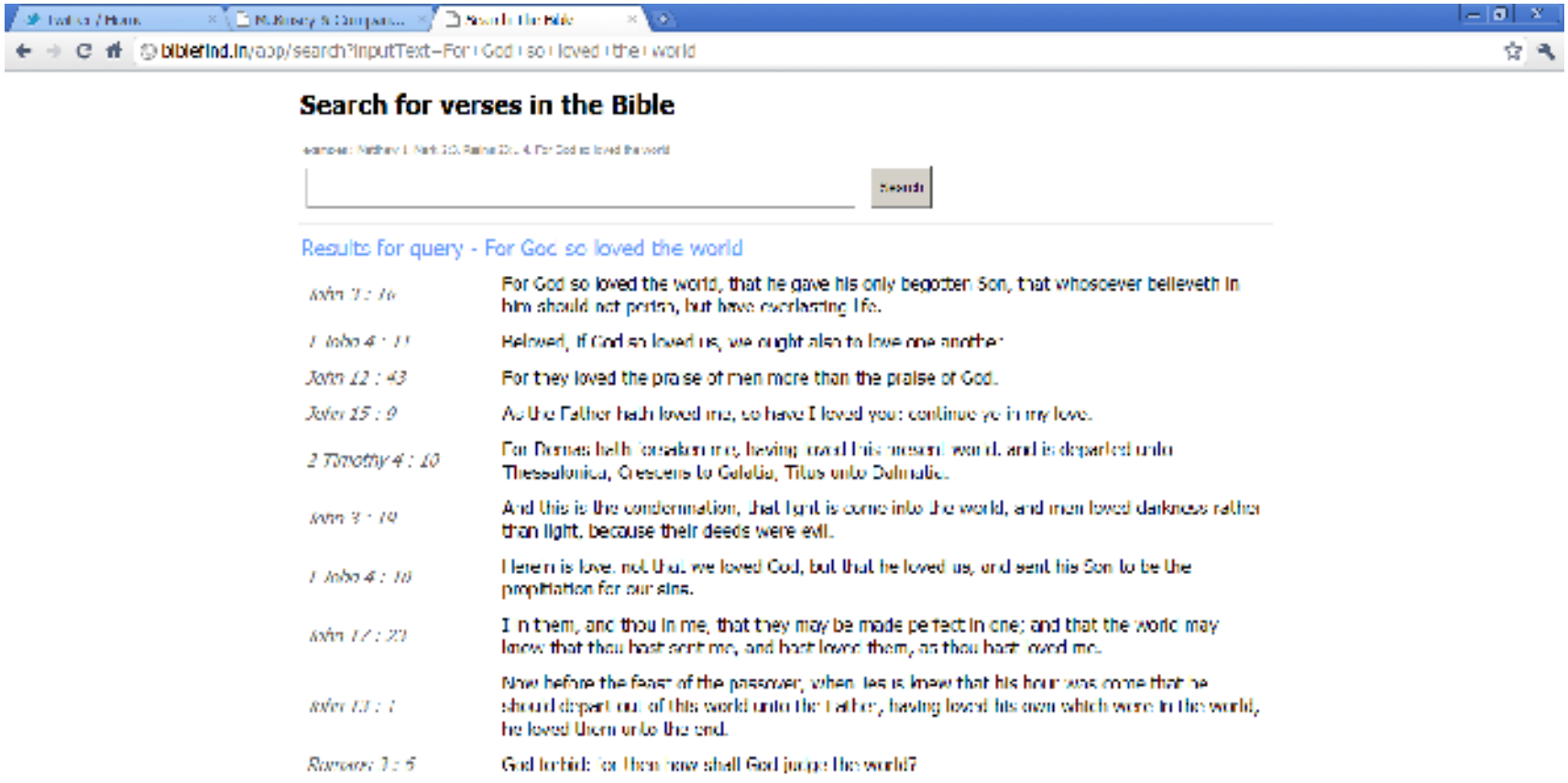
Blog – www.rockyj.in

Twitter – 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



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.

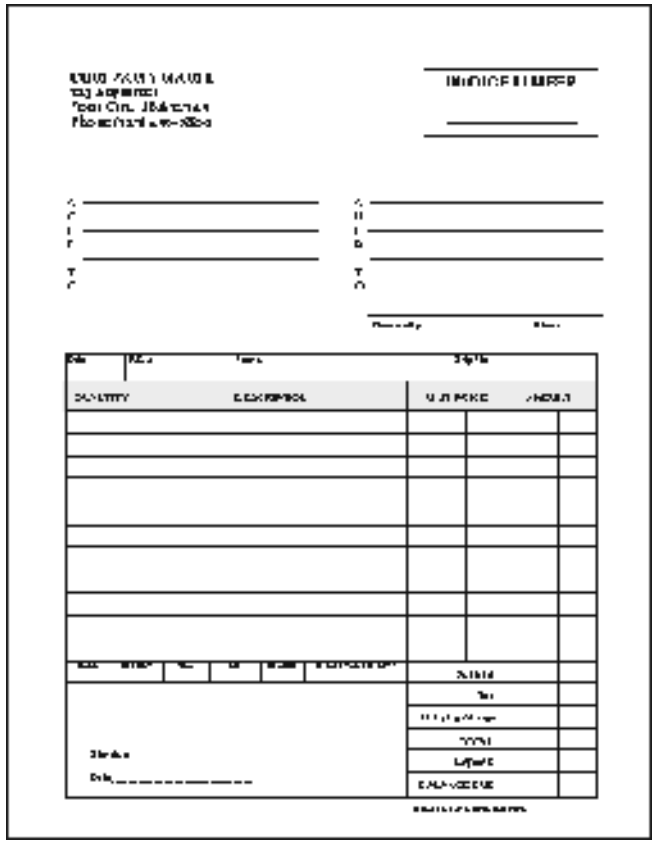
+ It offers an easy introduction to the world of NoSQL

HELLO COUCHDB – DOCUMENT ORIENTED

A different way to model your data.

Data stored in documents

Think of it as a de-normalized table row



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 ☺

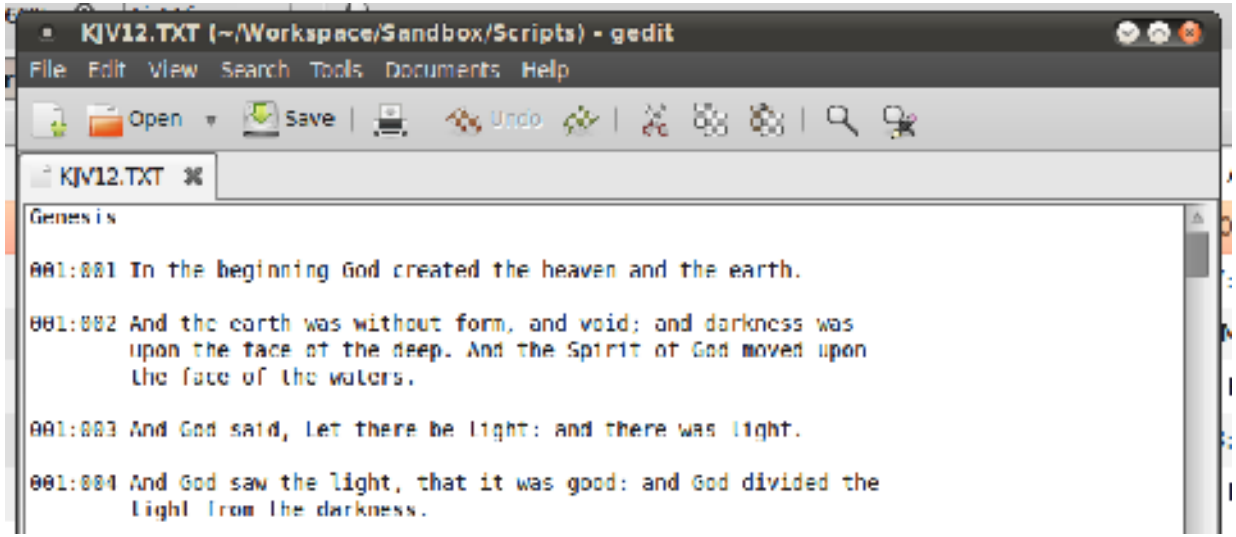
CouchDB's protocol is HTTP

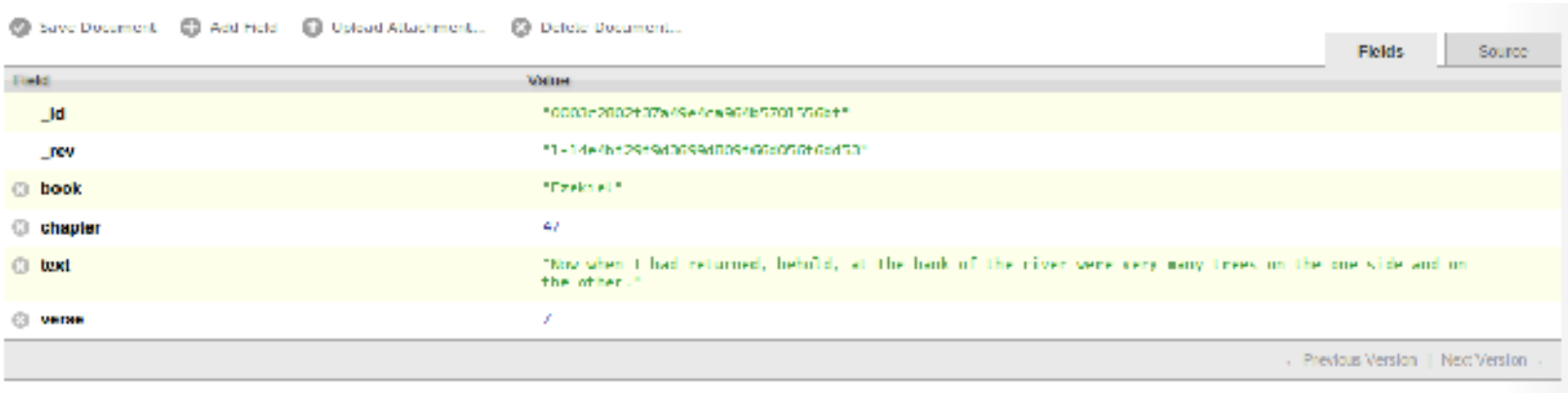
The data exchange + storage language is JSON

```
{
        "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





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

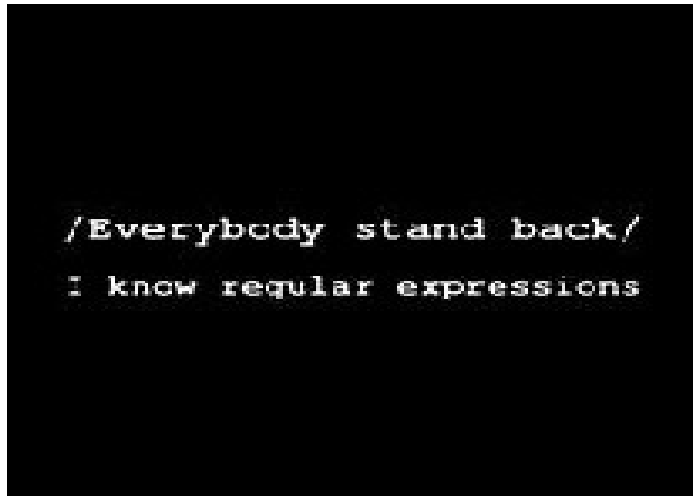
- 1 Controller
- 1 View
- Some jQuery for autocomplete

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**

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 -

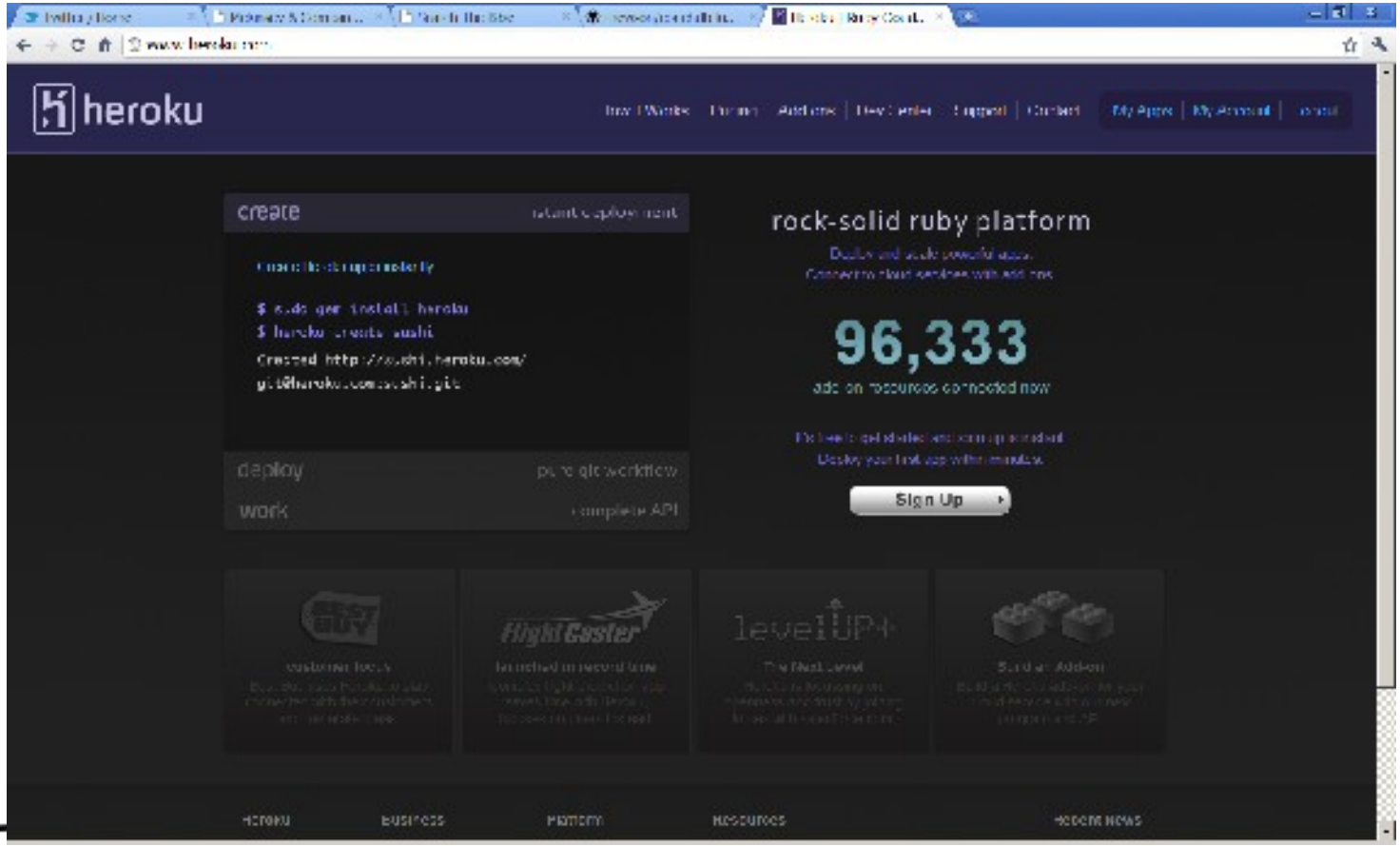
```
[external]
fti=python /home/rocky/Apps/couchdb-lucene-0.7-SNAPSHOT/tools/couchdb-external-hook.py

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

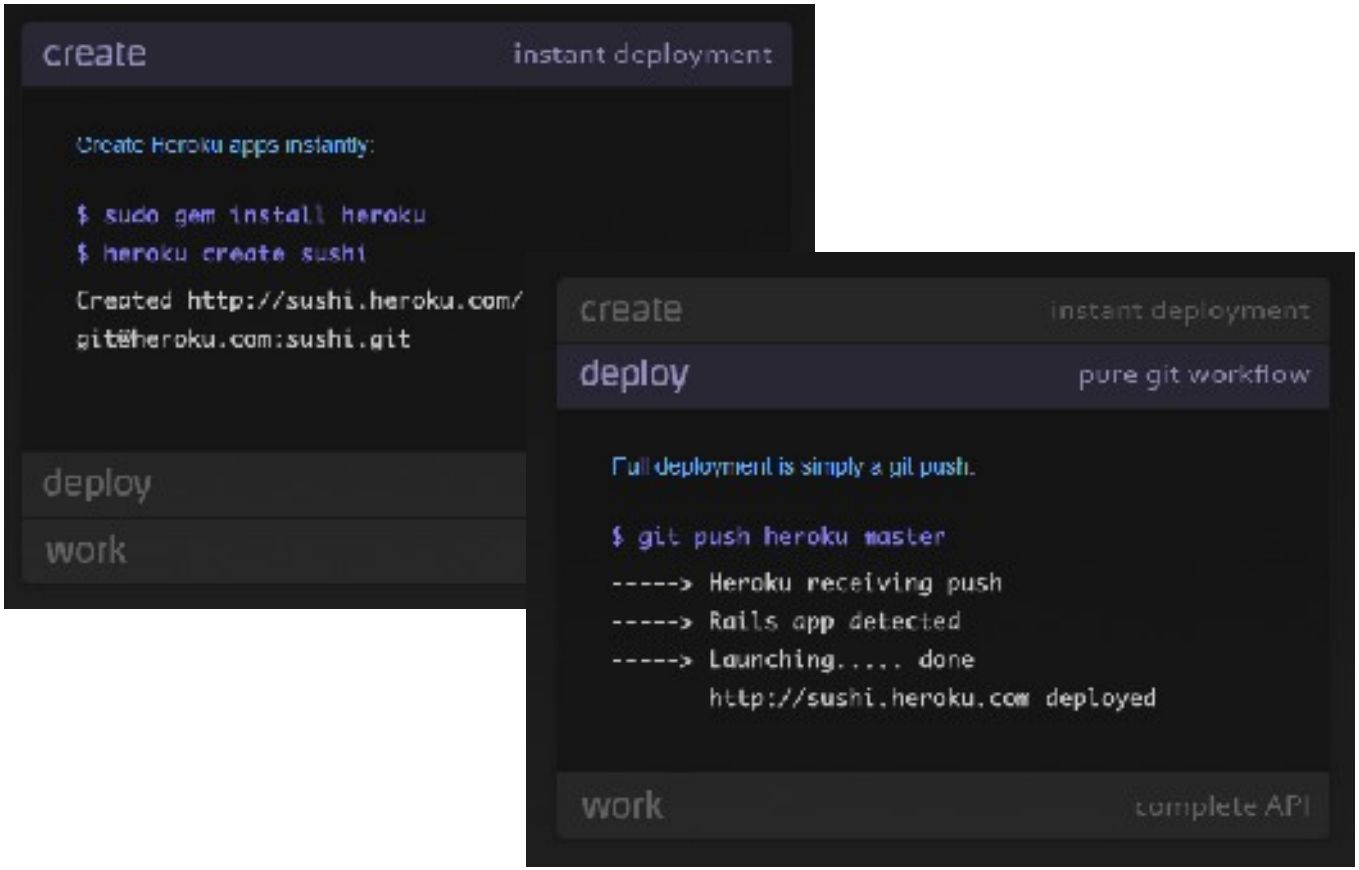
USING HEROKU

Rails application hosting provider

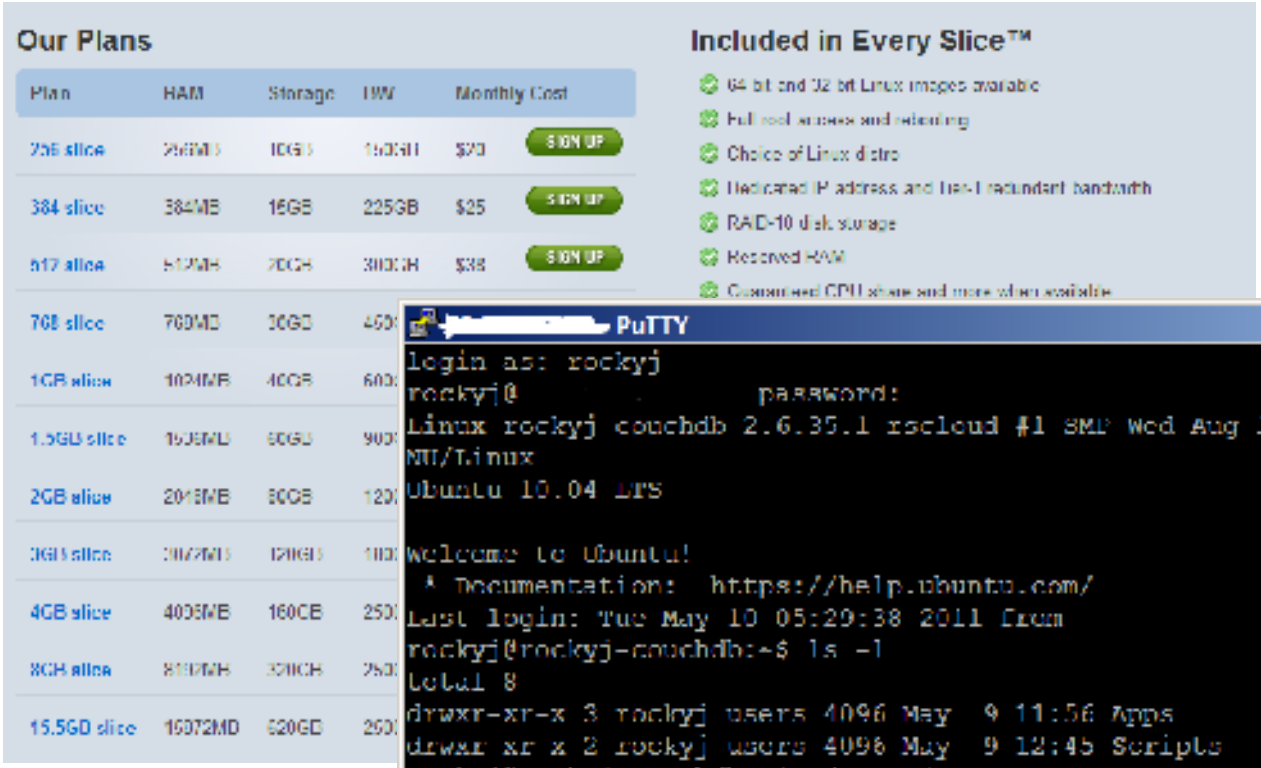
Free for 1 "Dyno" + 1 Shared database

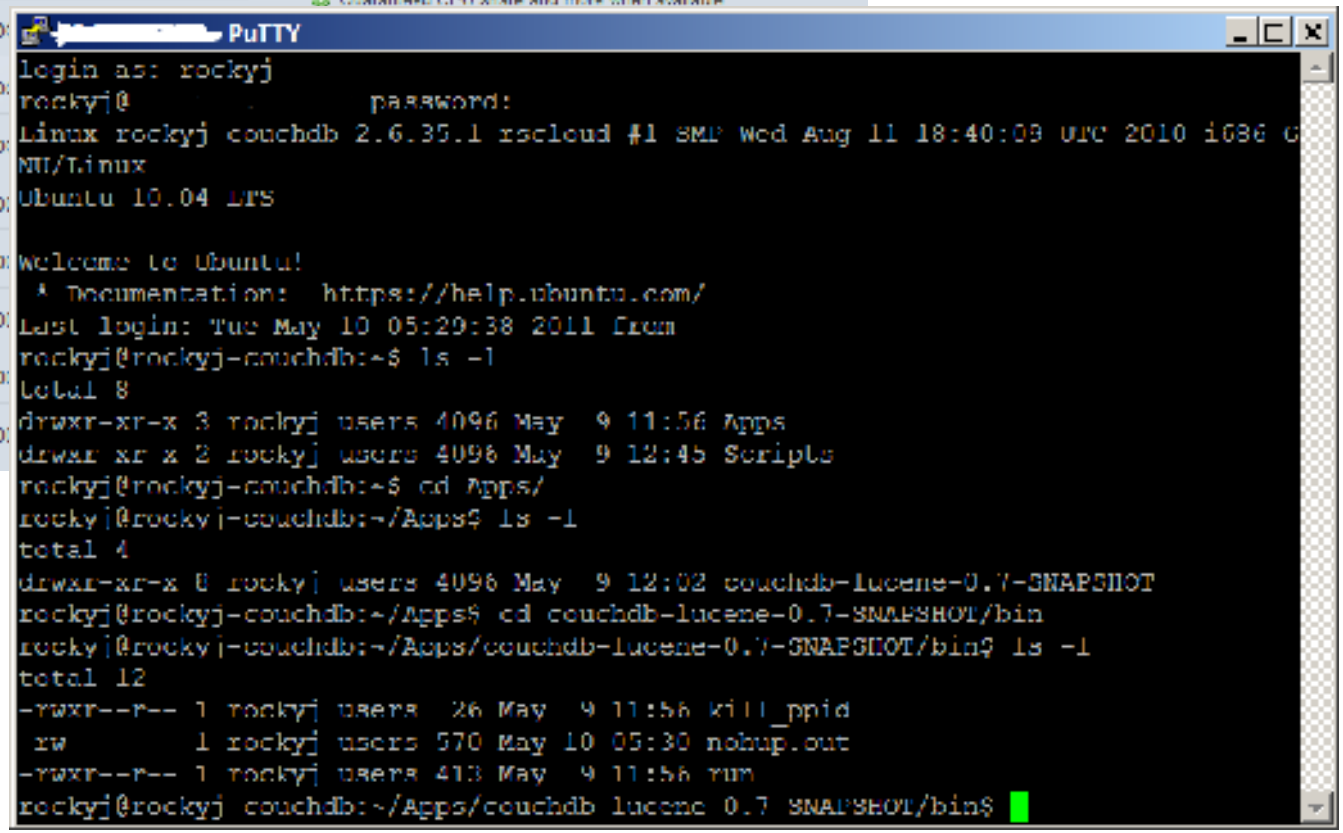


USING HEROKU CONTD..



USING SLICEHOST





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

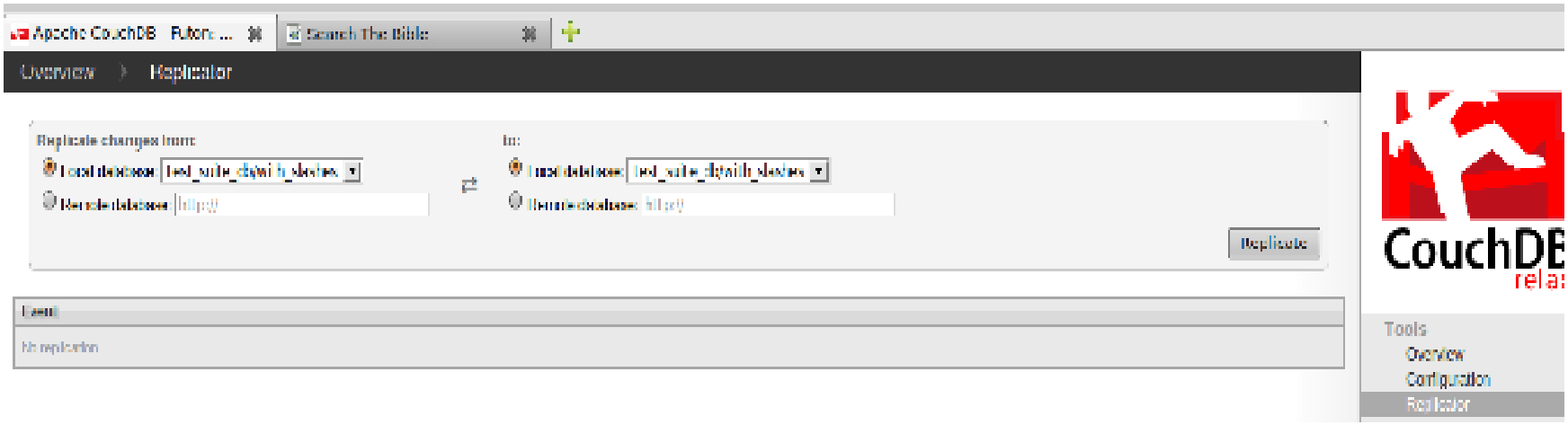
Heroku makes scaling dead easy

SCALING

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

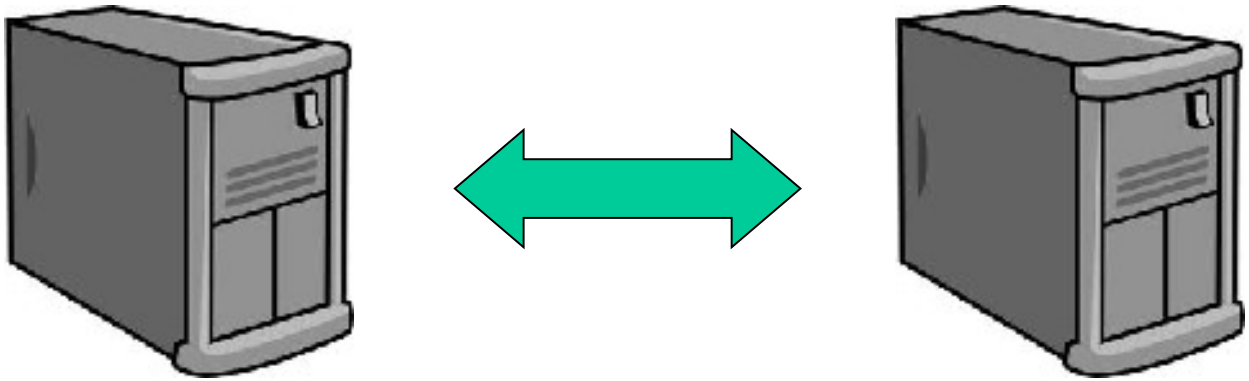
SCALING

In NoSQL world, replication is a first class citizen

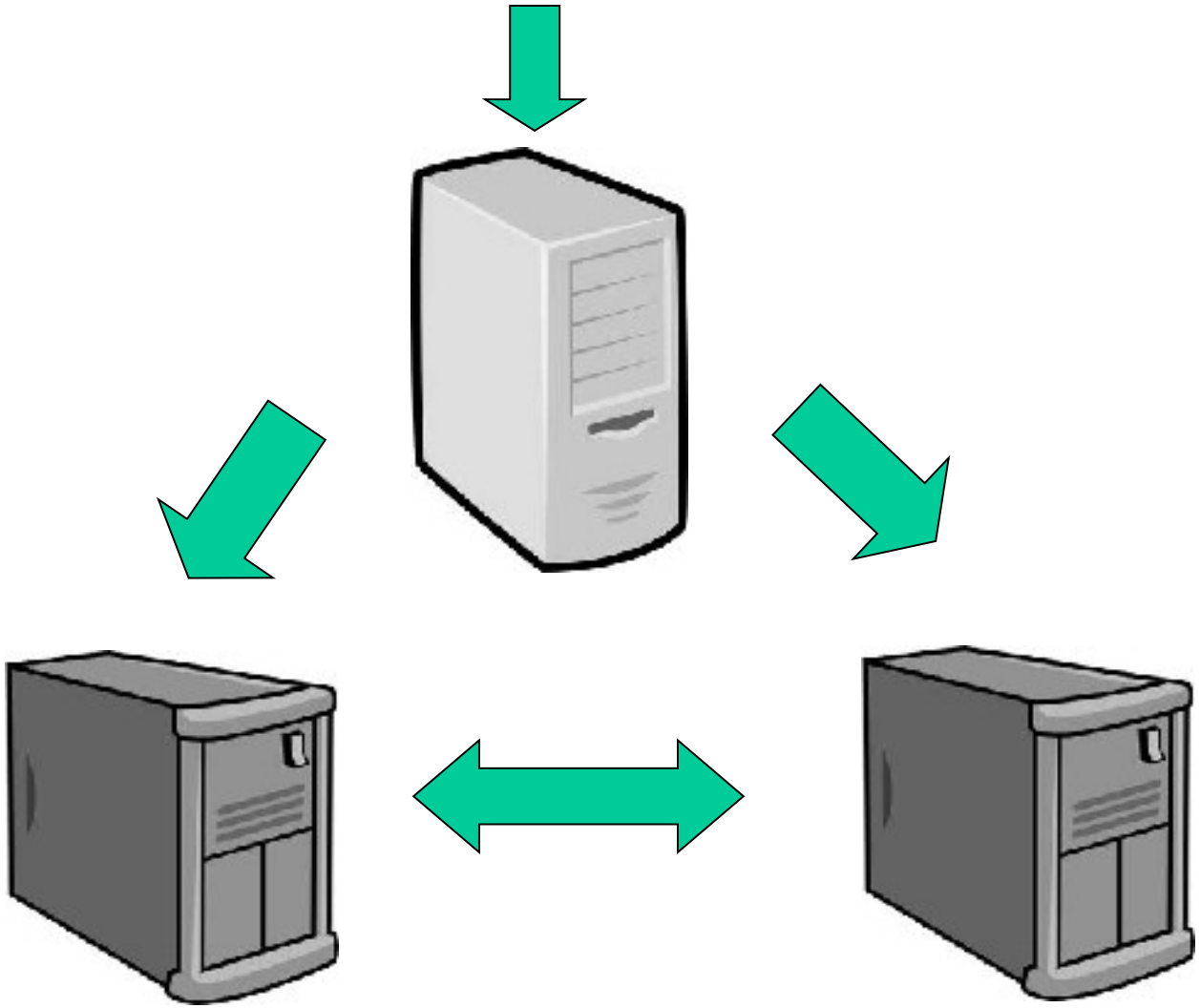


POST /_replicate {
"source":"a", "target":"b", "continuous":"true"
}

SCALING (HOT BACKUP)



SCALING WITH A DUMB PROXY



SCALING WITH COUCHDB LOUNGE

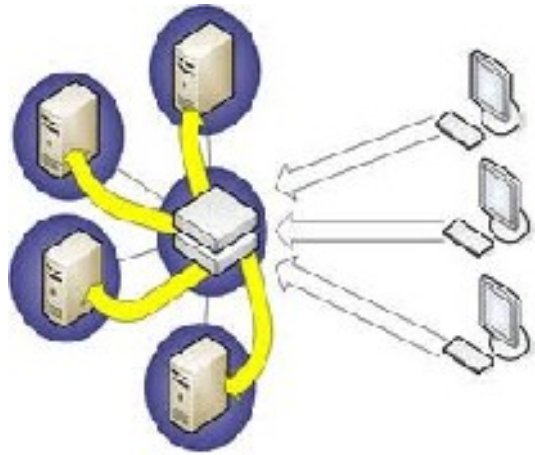
Have a look at CouchDB Lounge

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