Couchbase

Full-Stack JavaScript

Couchbase, node.js, and some AngularJS

Couchbase

Who am I?

Philipp Fehre - Developer Advocate

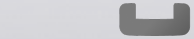
Twitter: @ischi Github: sideshowcoder

Couchbase

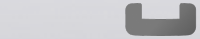
Let’s build something

Couchbase

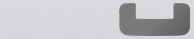
JSON and JavaScript



Responsiveness



Scalable



Efficiency

How Do We Get There?

Couchbase

node.js

- Event driven programming model
  - Leads your software development to be fast
- Scale out ready
  - Because of the loose coupling, node can scale out as far as you have machines
  - However, this means it *needs* Couchbase
- Very efficient use of system resources
  - Single threaded but multi-process, event driven model ensures good handoff from compute to IO

Couchbase

Couchbase

- Sub-millisecond latency
  - Gives you the consistent performance needed to build complex, interactive game play
- Designed for Scale
  - Add and remove nodes as needed
- Efficiency
  - Couchbase manages system resources such as memory and CPU efficiently

Couchbase

- Document database
- Store different data types
  - Counters
  - Numbers
  - Strings
  - JSON

JSON

we all know and love

Couchbase

JSON in node.js

```
/* $ cat example.json
 * {
 *   "foo": "bar"
 * }
 */

var fs = require("fs");
var rawData = fs.readFileSync("./example.json");
var data = JSON.parse(rawData);
console.log("property foo of data:", data.foo);

/* $ node read.js
 * property for of data: bar
 */
```

Couchbase & JSON

gamesim-sample > Documents

Aaron0

```
{
  "experience": 14746,
  "hitpoints": 20210,
  "jsonType": "player",
  "level": 146,
  "loggedIn": true,
  "name": "Aaron0",
  "uuid": "3b49dd18-1d56-478e-8ab1-fb38e31ce7e2"
}
```

JSON & Couchbase

- Native Data format
  - Special support for JSON documents is provided
  - Couchbase recognises JSON as a Datatype
- Complex queries
  - JSON can be handled by the View engine
  - Build indices via Map / Reduce

Couchbase Views

VIEW CODE

Map

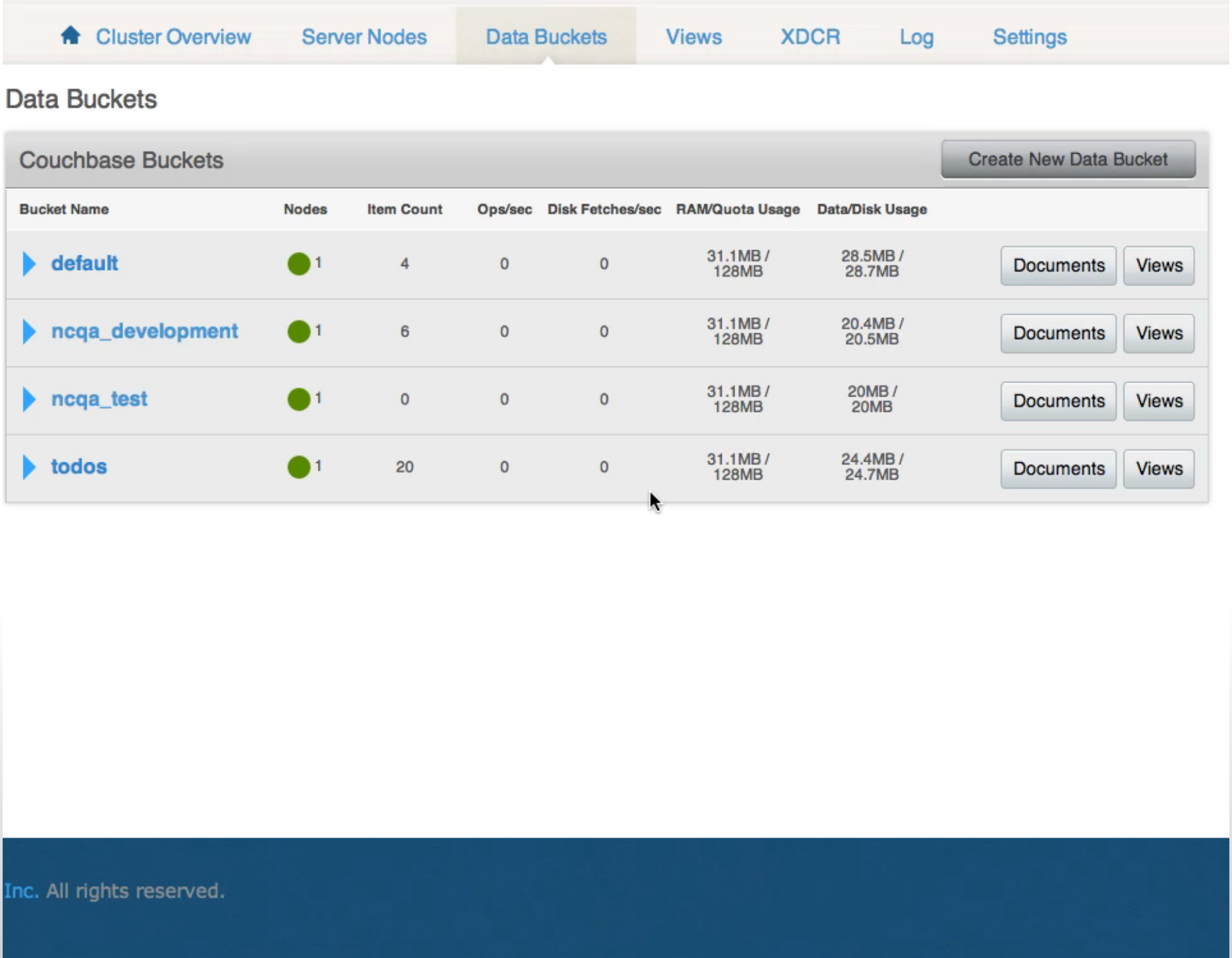
```
1  function (doc, meta) {
2    if (doc.jsonType == "player") {
3    emit(meta.id, null);
4    }
5  }
```

Reduc

```
1
```

Filter Results ?stale=false&connection_timeout=60000&limit=10&skip=0

Key	Value
"Aaron0" Aaron0	null


Cluster Overview
Server Nodes
Data Buckets
Views
XDCR
Log
Settings
Data Buckets
Couchbase Buckets
Create New Data Bucket
Bucket Name
Nodes
Item Count
Ops/sec
Disk Fetches/sec
RAM/Quota Usage
Data/Disk Usage
default
1
4
0
0
31.1MB / 128MB
28.5MB / 28.7MB
Documents
Views
ncqa_development
1
6
0
0
31.1MB / 128MB
20.4MB / 20.5MB
Documents
Views
ncqa_test
1
0
0
0
31.1MB / 128MB
20MB / 20MB
Documents
Views
todos
1
20
0
0
31.1MB / 128MB
24.4MB / 24.7MB
Documents
Views
Inc. All rights reserved.

JSON as the API

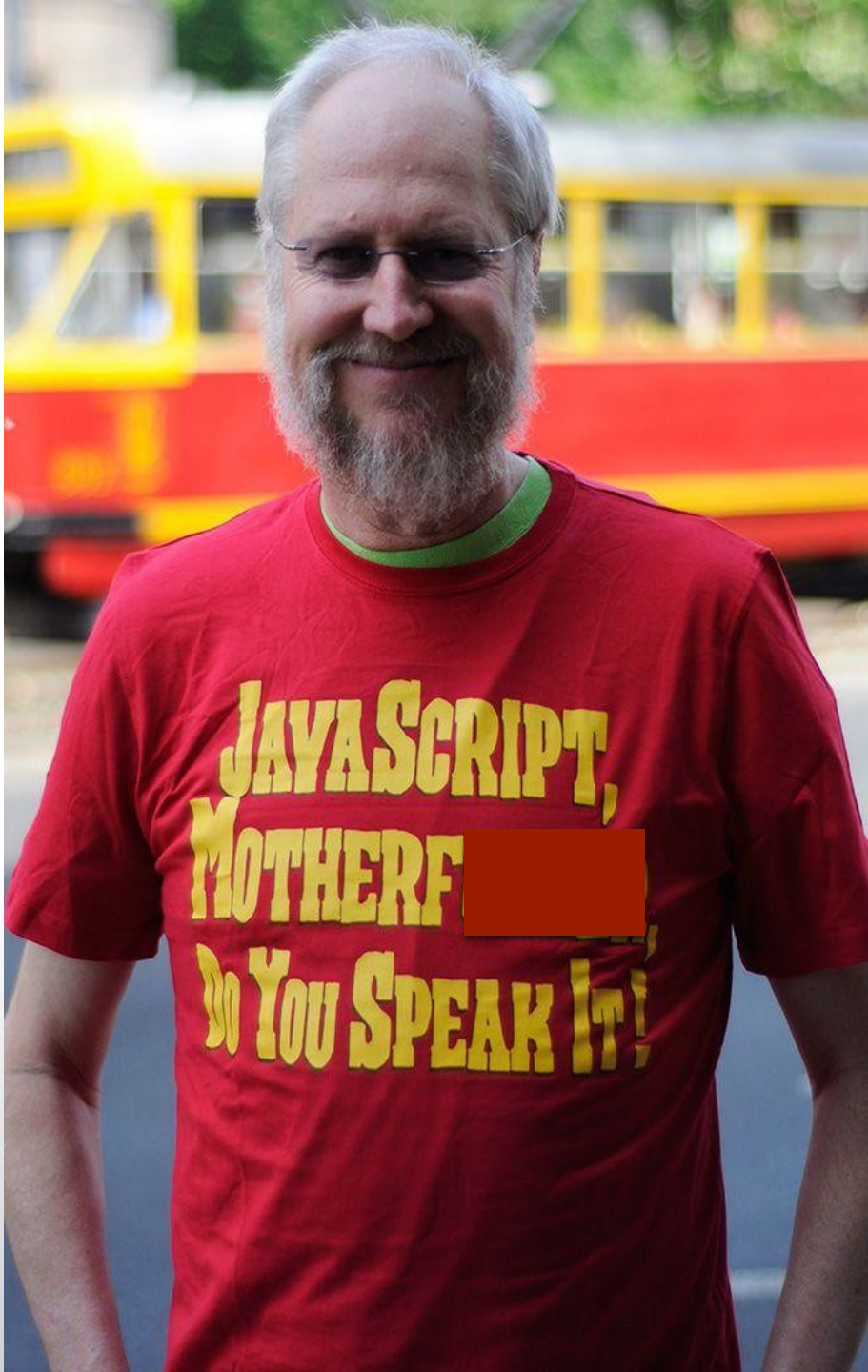
curl https://api.github.com/users/sideshowcoder


```
{
  "login": "sideshowcoder",
  "id": 108488,
  "avatar_url": "https://avatars.githubusercontent.com/u/…",
  "gravatar_id": "5cde19029032f151ca09687f7c8783eb",
  "url": "https://api.github.com/users/sideshowcoder",
  "html_url": "https://github.com/sideshowcoder",
  "followers_url": "https://api.github.com/users/…",
  "following_url": "https://api.github.com/users/…",
  ...
}
```

JSON in the Browser

```
<html>
  <body>
    <script src="/jquery-1.11.0.min.js"></script>
    <div id="user-name"></div>
    <div id="last-active"></div>
    <script>
      $.getJSON("https://api.github.com/users/sideshowcoder",
      function (data) {
          $("#user-name").text(data.login);
          $("#last-active").text(data.updated_at);
      })
    </script>
  </body>
</html>
```

JavaScript is the Programming Language of the Web



Demo

Couchbase

Building an API

with Couchbase, node.js, and AngularJS

Couchbase

Users

```
app.post("/signup", function (req, res) {
  var credentials = req.body
  User.create(credentials, function (err, user) {
    if (err) {
      res.render("signup", { message: err })
    } else {
      req.session.userId = user.userId
      res.redirect("/")
    }
  })
})
```

```
app.post("/signin", function (req, res) {
  var credentials = req.body
  User.authenticate(credentials, function (err, user) {
    if (err) {
      res.render("signup", { message: err })
    } else {
      req.session.userId = user.userId
      res.redirect("/")
    }
  })
})
```

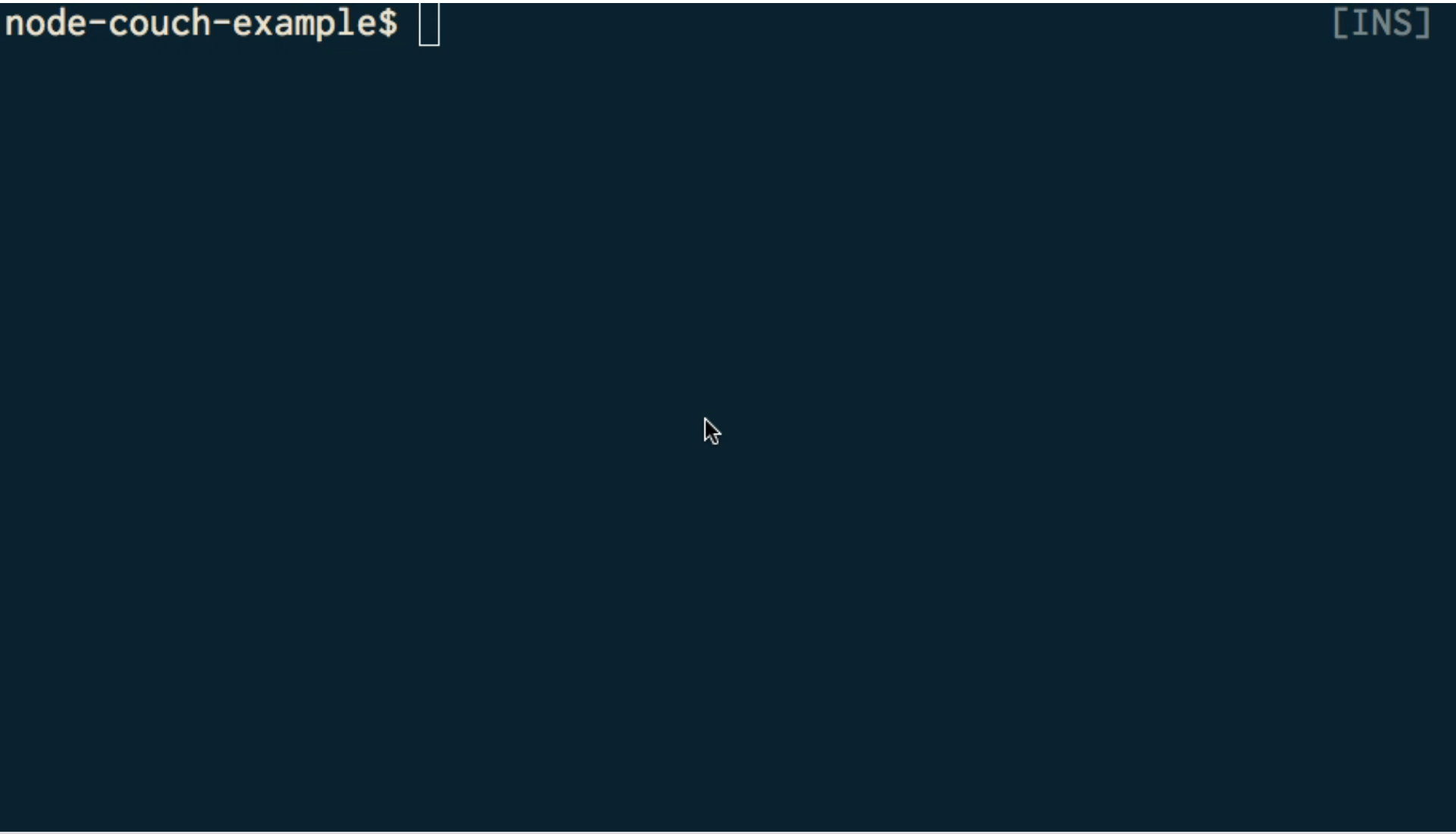
The Database

Using Couchbase from node.js

```
var couchbase = require("couchbase")
var cluster = new couchbase.Cluster()
var bucket = cluster.openBucket("default")

var doc = { store: "json we can", multiple: "values it can have" }

bucket.upsert("my-key", doc, function (err, res) {
  if (err) throw err
  bucket.get("my-key", function (err, res) {
    if (err) throw err
    console.log(res)
    bucket.disconnect()
  })
})
```


node-couch-example$
[INS]

Create a connection

```
new couchbase.Connection(dbConfig, function(){})
```

Smart client

- Abstracts the cluster
  - managing connections to all servers
  - reestablishes failed connection
- Manages connection handshake
  - Transfers and continuously updates the cluster map
  - Detects cluster configuration changes and abstracts them for the user

Reuse your database connection

```
var _db = null;

var db = function (cb) {
  if (_db) return cb(null, _db);

  _db = new couchbase.Connection(dbConfig, function(err) {
    if (err) return cb(err);
    cb(null, _db);
  })
}
```

Keying

- Use a Unique value for key (email, username, sku, isbn, etc.)
  - example u::phil
- Predictable Keys can follow Key-Value patterns
  - Users typically can be done this way and are the most numerous items
- Referential Keys
  - u::philipp.fehre@gmail.com => u::phil
  - u::ischi => u::phil

Creating a new user

Authentication

Keep in mind GET is quick

```
var User = require("./user")

var auth = function (req, res, next) {
  if (req.session.userId) {
    User.validateId(req.session.userId, function (err, user) {
      if (err) {
        res.redirect("/signin")
      } else {
        req.user = user
        next()
      }
    })
  } else {
    res.redirect("/signin")
  }
}

module.exports = auth
```

Game State

Routes

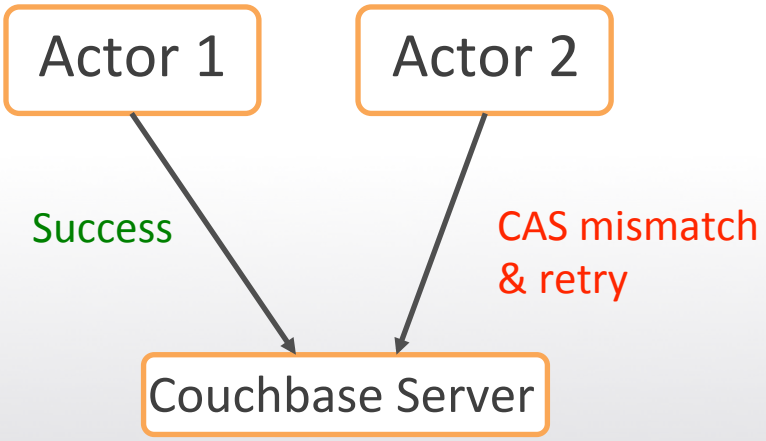
```
app.get("/questions", auth, function (req, res) {
  res.set("Content-Type", "application/json")
  QuestionList.forUser(req.user, function (err, list) {
    if (err) {
      res.statusCode = 500
      res.end()
    } else {
      res.end(JSON.stringify(list))
    }
  })
})
```

```
app.post("/questions", auth, function (req, res) {
  var questions = req.body
  QuestionList.saveForUser(req.user, questions, function (err) {
    if (err) {
      res.statusCode = 500
      return res.end()
    } else {
      QuestionList.forUser(req.user, function (err, list) {
        if (err) {
          res.statusCode = 500
          res.end()
        } else {
          res.end(JSON.stringify(list))
        }
      })
    }
  })
})
```

Question List

Dealing with concurrency

- **If a question is answered from two devices at the same time, they collide**
  - We don't want to accidentally overwrite an answer edit since we're sending a full document update right after theirs
- **Retry the operation, *if appropriate***




```
[
  {
    "id": "1",
    "text": "What is couchbases upcoming
query language called?",
    "choices": [
      {
        "text": "N1QL",
        "state": true,
        "count": 1,
        "id": "a",
        "$$hashKey": "00Q"
      },
      {
        "text": "SQL",
        "state": false,
        "count": 0,
        "id": "b",
        "$$hashKey": "00R"
```

Couchbase

```
QuestionList.prototype.load = function (includeCount, cb) {
  var that = this
  db.get(this.key, function (err, data) {
    if (err) {
    …
    } else {
      that.cas = data.cas
      that.questions = data.value
      cb(null, that)
    }
  })
}

QuestionList.prototype.save = function (cb) {
  db.set(this.key, this, { cas: this.cas }, cb)
}
```

Consistent Data

No merging or quorum read needed

The Frontend

Couchbase

AngularJS

- Provide views template data bindings
- Encapsulate logic in Model-View-“Whatever works”
- Templates are based on “enhanced” HTML

Rendering JSON responses

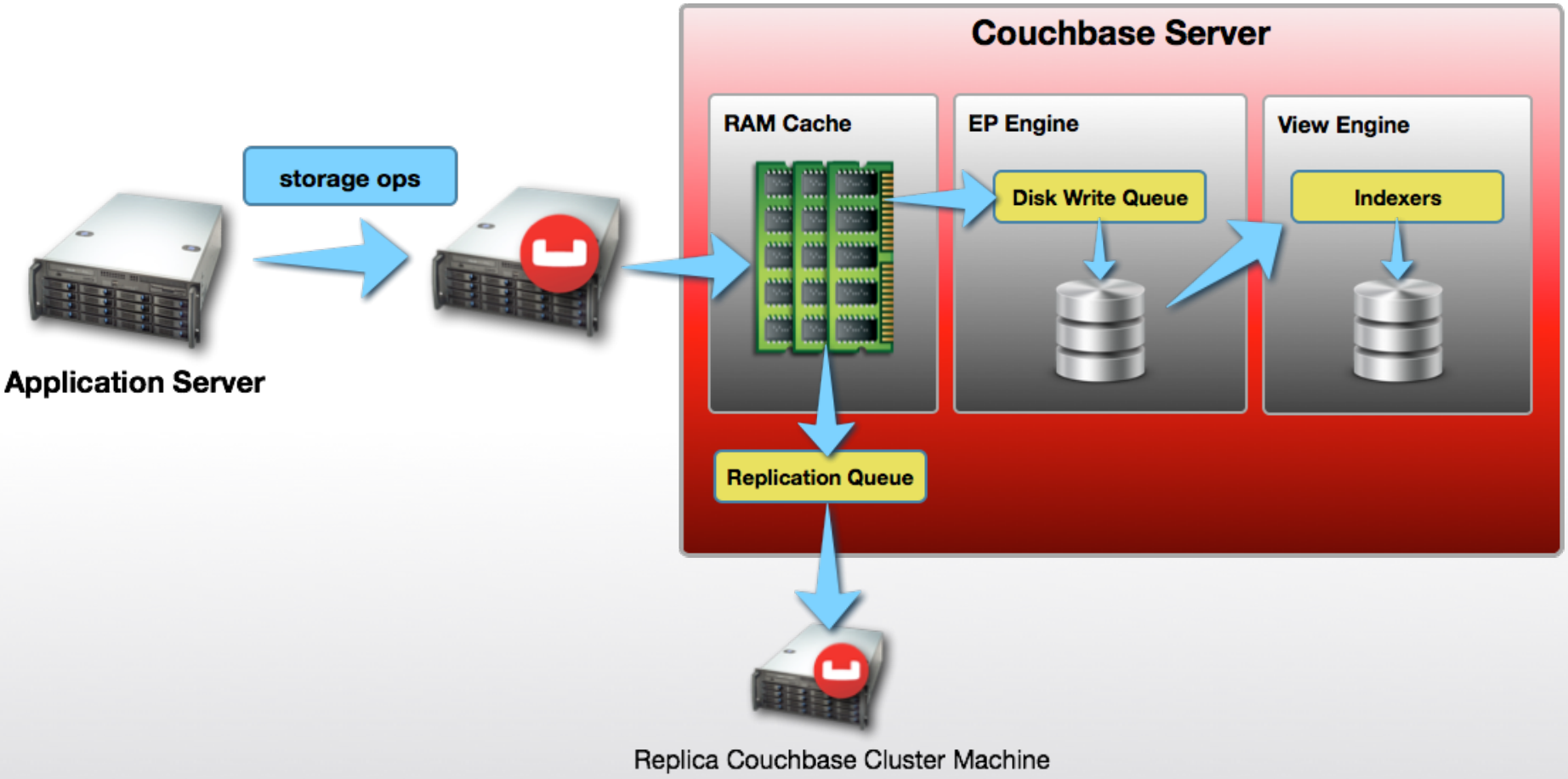
One more thing

Couchbase

Showing the answer count

Couchbase Views

Storage to Index



Creating views with code

```
var countsDDoc = {
  "views": {
    "counts": {
      "map": "function (doc, meta) { if(doc[0]) {…}”,
      "reduce": "_sum"
    }
  }
}

db.setDesignDoc("ncqa", countsDDoc, function (err, data) {
  if (err) {
    console.log("failed to setup view.", err)
  } else {
    console.log("setup view done.")
  }
  db.shutdown()
})
```

Querying the View

```
QuestionList.prototype.loadCounts = function (cb) {
  var that = this
  db.conn(function (err, conn) {
    var opts = {stale: false, group: true, keys: choicesKeys}
    var q = conn.view("ncqa", "counts", opts)
    q.query(function (err, results) {
       …
    })
  })
}
```

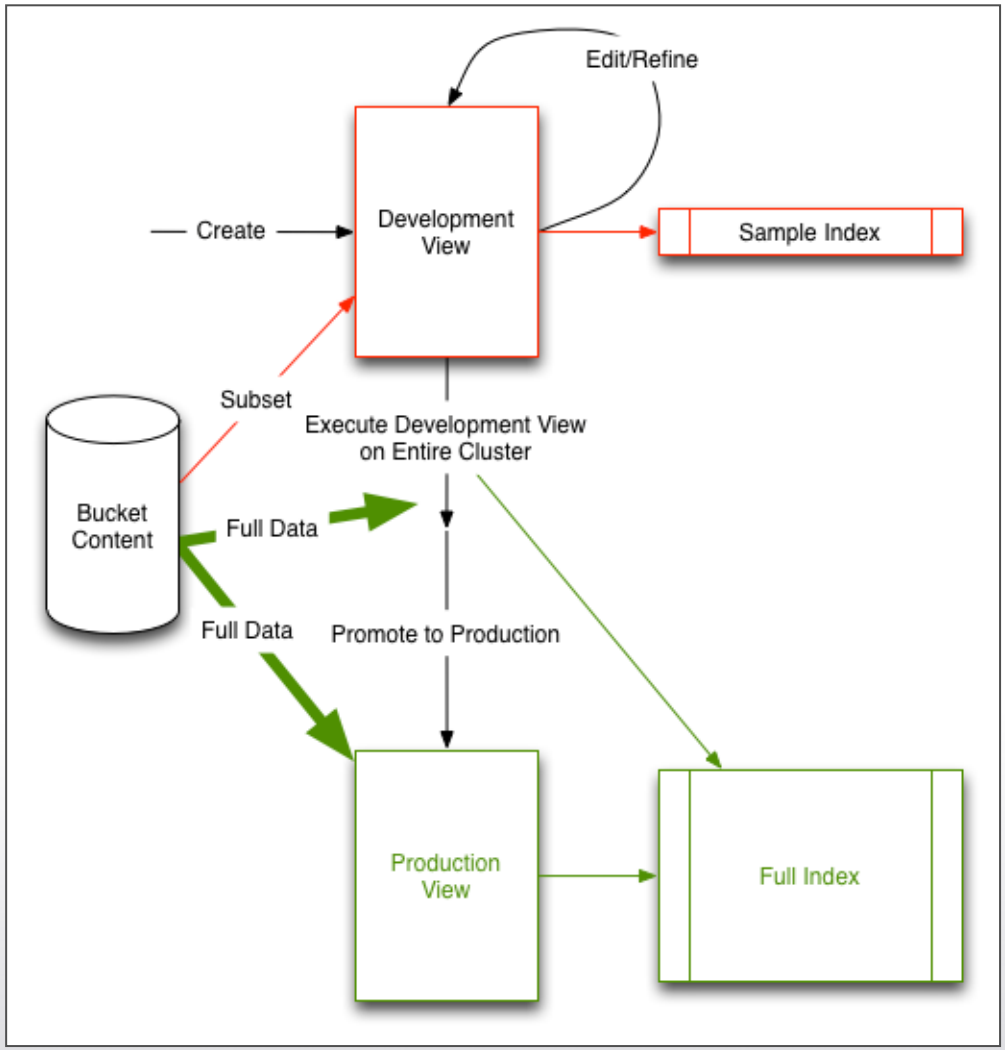
Eventual Persistence

When to use and when not to use

- views operate on the persisted data
- don’t use for “login”
- data can be “stale”

Development vs. Production Views

- Development views index a subset of the data.
- Publishing a view builds the index across the entire cluster.
- Queries on production views are scattered to all cluster members and results are gathered and returned to the client.



Couchbase 3.0

Many improvements to views are coming

Watch out for N1QL coming up!

More Demo

What to do next…

- **get the code, set it up, run**
  - https://github.com/couchbaselabs/node-couch-qa
- **Read Brett's blogs**
  - https://blog.couchbase.com/game-servers-and-couchbase-nodejs-part-1
- **Write your own! We help you along!**
  - http://www.couchbase.com/communities/nodejs

Resources

- https://github.com/couchbaselabs/node-couch-qa
- https://github.com/couchbase/couchnode
- https://blog.couchbase.com/game-servers-and-couchbase-nodejs-part-1
- https://blog.couchbase.com/game-servers-and-couchbase-nodejs-part-2
- https://blog.couchbase.com/game-servers-and-couchbase-nodejs-part-3
- https://github.com/brett19/node-gameapi

Thank you!

philipp@couchbase.com

Github: @sideshowcoder

Twitter: @ischi

Blog: sideshowcoder.com