Official Beta!

Apache CouchDB

INTRODUCTIONS

- Mike Miller
- *not* an Apache CouchDB committer
- Particle Physics Faculty at U. Washington
- Cloudant cofounder, Chief Scientist
- Background: machine learning, analysis, big data, globally distributed systems

CouchDB

- Schema-free document database server
- Robust, highly concurrent, fault-tolerant
- RESTful JSON API
- Futon web admin console
- MapReduce system for generating custom views
- Bi-directional incremental replication
- couchapp: lightweight HTML+JavaScript apps served directly from CouchDB using views to transform JSON

PARAPHRASING THE MASSES

- Why CouchDB?
  - Simple, robust, concurrent, fun
  - successful in production
- Why not couch?
  - Missing features (ad hoc queries, auth/auth, sharding, etc)
  - “Too slow”
  - too new: (“*still alpha*”)

FROM INTEREST TO ADOPTION

- 100+ production users
- 3 books being written
- Vibrant, open community
- Active commercial development
- Rapidly maturing

OF THE WEB

> Django may be built *for* the Web, but CouchDB is built *of* the Web. I've never seen software that so completely embraces the philosophies behind HTTP ... this is what the software of the future looks like.

Jacob Kaplan-Moss
October 17 2007

http://jacobian.org/writing/of-the-web/

SHOULD YOU CARE?

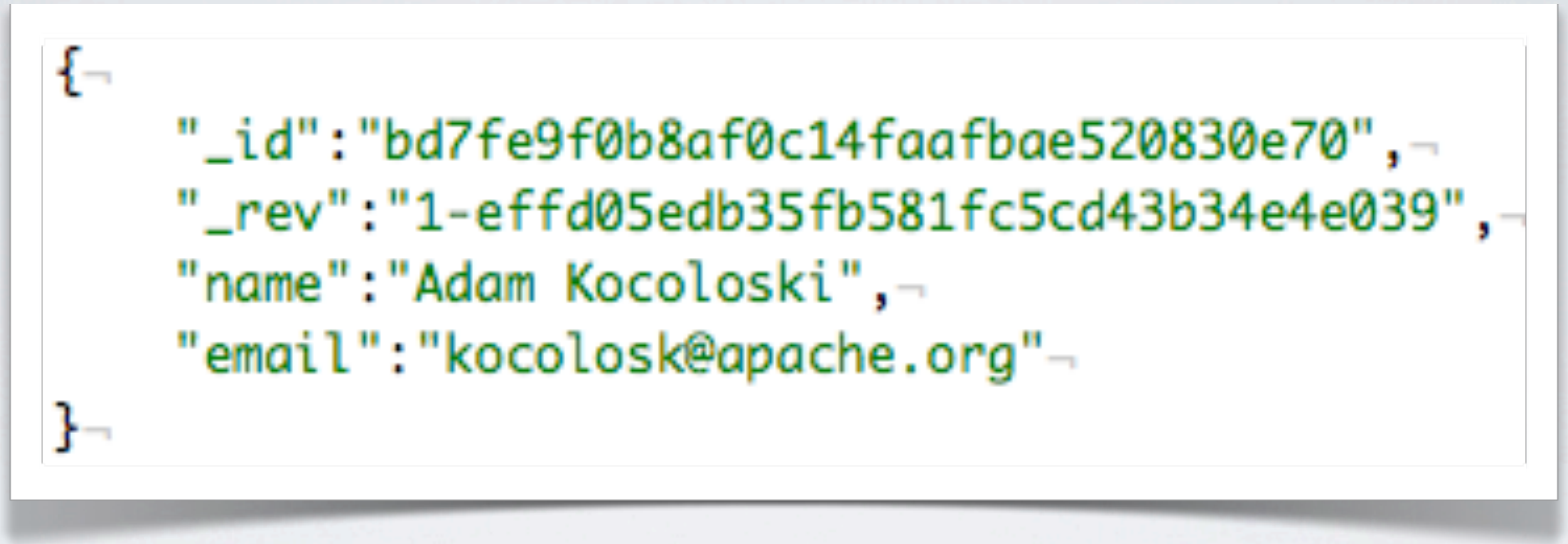
The internet happened, and we ignored it.
In retrospect that was a mistake.

Paraphrased from Bill Warner (Founder Avid, Wildfire)
Summer, 2008

New technologies enable new business

LETS TAKE A LOOK

DOCUMENTS



```
{
    "_id":"bd7fe9f0b8af0c14faafbae520830e70",
    "_rev":"1-effd05edb35fb581fc5cd43b34e4e039",
    "name":"Adam Kocoloski",
    "email":"kocolosk@apache.org"
}
```

- Documents are JSON Objects
- Underscore-prefixed fields are reserved
- Documents can have binary attachments
- MVCC _rev deterministically generated from doc content

ROBUST

- Never overwrite previously committed data
- In the event of a server crash or power failure, just restart CouchDB -- there is no “repair”
- Take snapshots with “cp”
- Configurable levels of durability: can choose to fsync after every update, or less often to gain better throughput

CONCURRENT

- Erlang approach: lightweight processes to model the natural concurrency in a problem
- For CouchDB that means one process per TCP connection
- Lock-free architecture; each process works with an MVCC snapshot of a DB.
- Performance degrades gracefully under heavy concurrent load

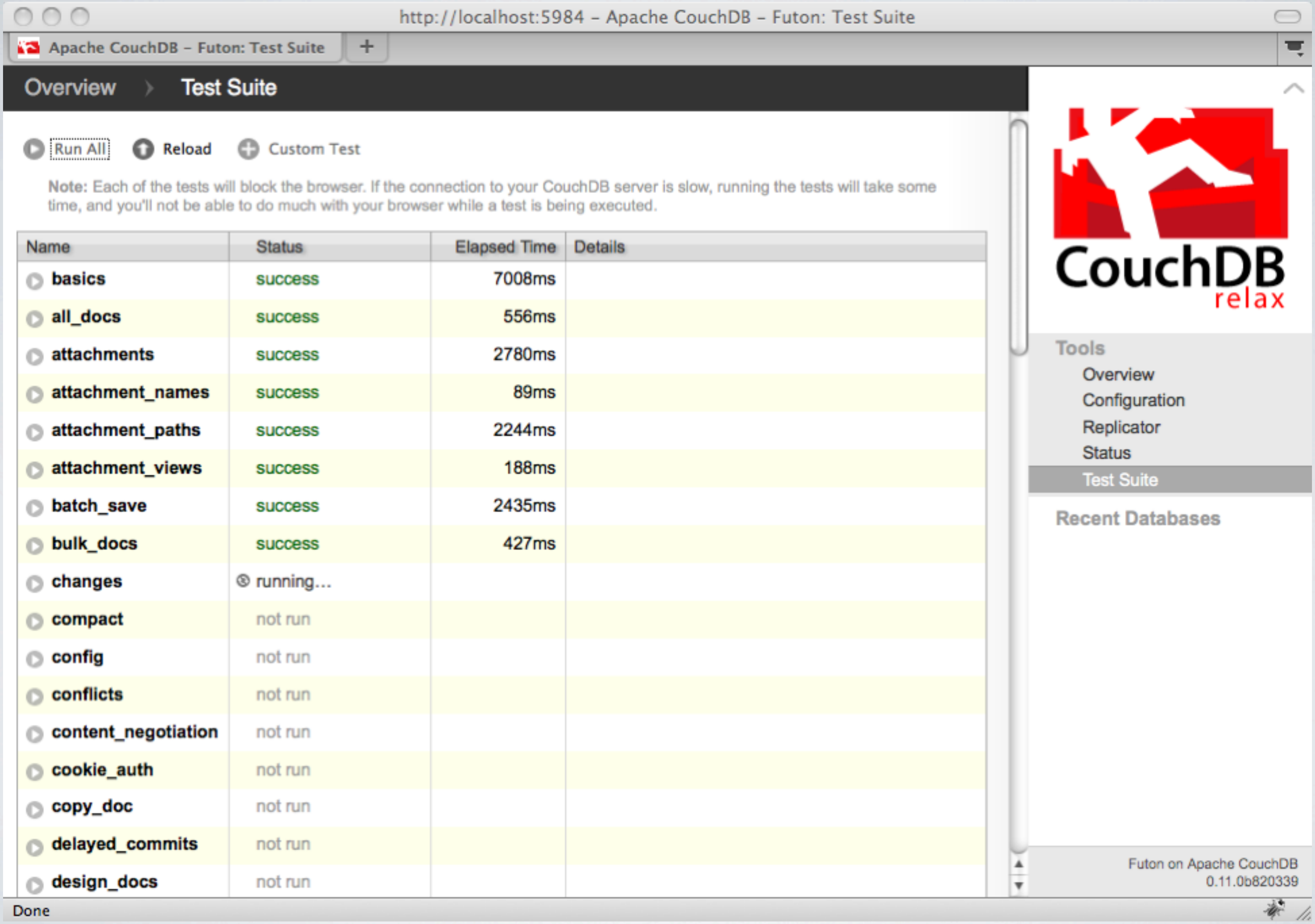
REST API

- **C**reate
  PUT /mydb/mydocid

- **R**etrieve
  GET /mydb/mydocid

- **U**pdate
  PUT /mydb/mydocid

- **D**elete
  DELETE /mydb/mydocid



Note: Each of the tests will block the browser. If the connection to your CouchDB server is slow, running the tests will take some time, and you'll not be able to do much with your browser while a test is being executed.

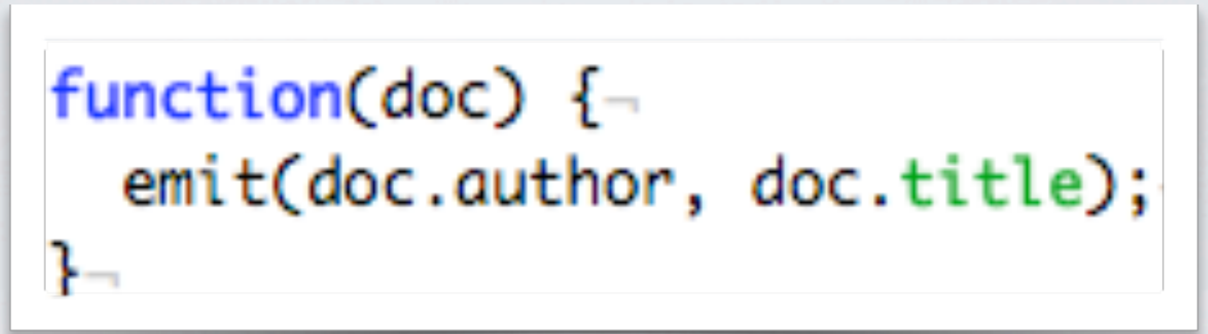
Name	Status	Elapsed Time	Details
basics	success	7008ms	
all_docs	success	556ms	
attachments	success	2780ms	
attachment_names	success	89ms	
attachment_paths	success	2244ms	
attachment_views	success	188ms	
batch_save	success	2435ms	
bulk_docs	success	427ms	
changes	running...		
compact	not run		
config	not run		
conflicts	not run		
content_negotiation	not run		
cookie_auth	not run		
copy_doc	not run		
delayed_commits	not run		
design_docs	not run		

Futon on Apache CouchDB 0.11.0b820339

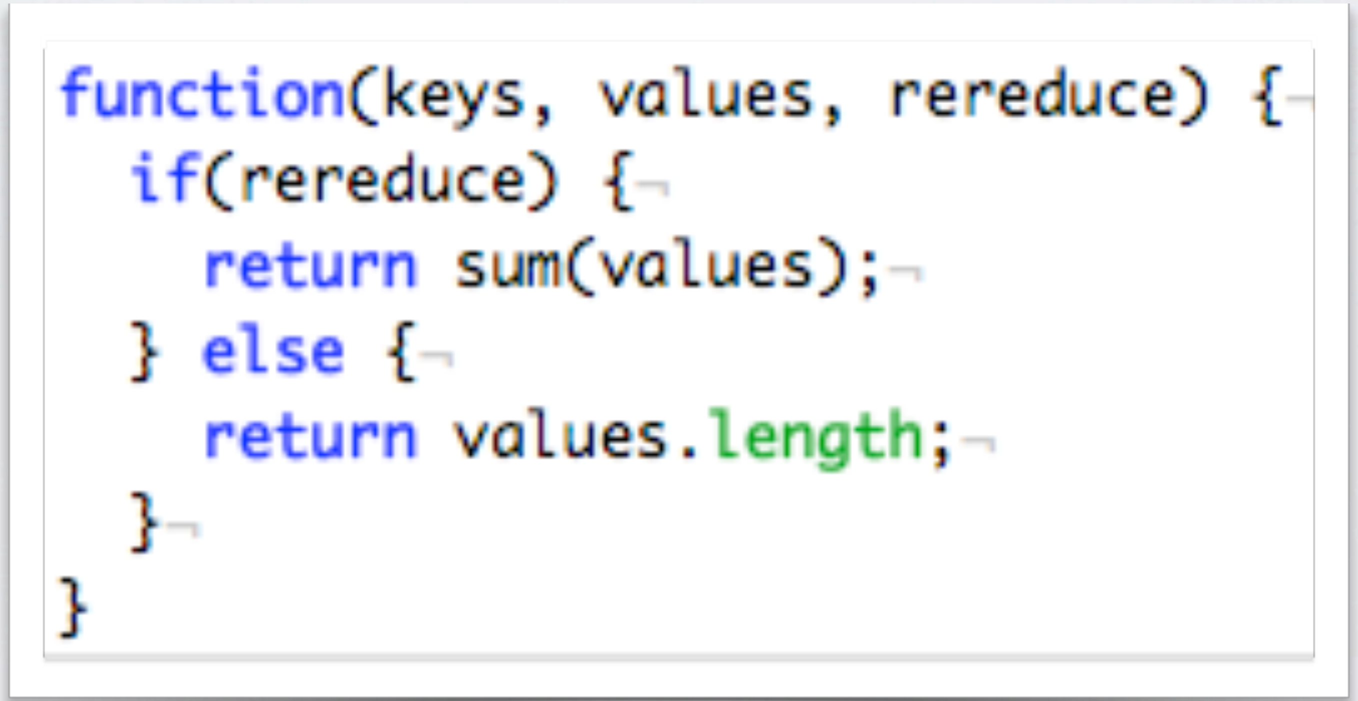
VIEWS

- Custom, persistent representations of document data
- "Close to the metal" -- no dynamic queries in production, so you know exactly what you're getting
- Generated using MapReduce functions written in JavaScript (and other languages)
- Each view must have a **map** function and may also have a **reduce** function
- Leverages view collation, rich view query API

DOCUMENTS BY AUTHOR

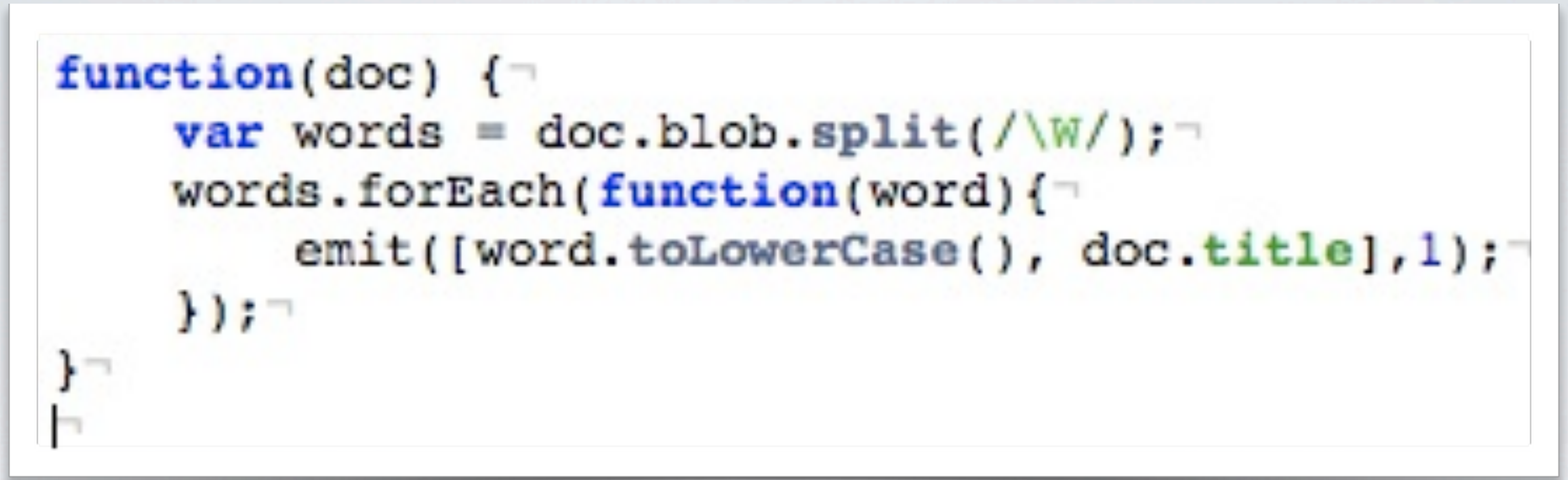


```
function(doc) {¬
  emit(doc.author, doc.title);
}¬
```

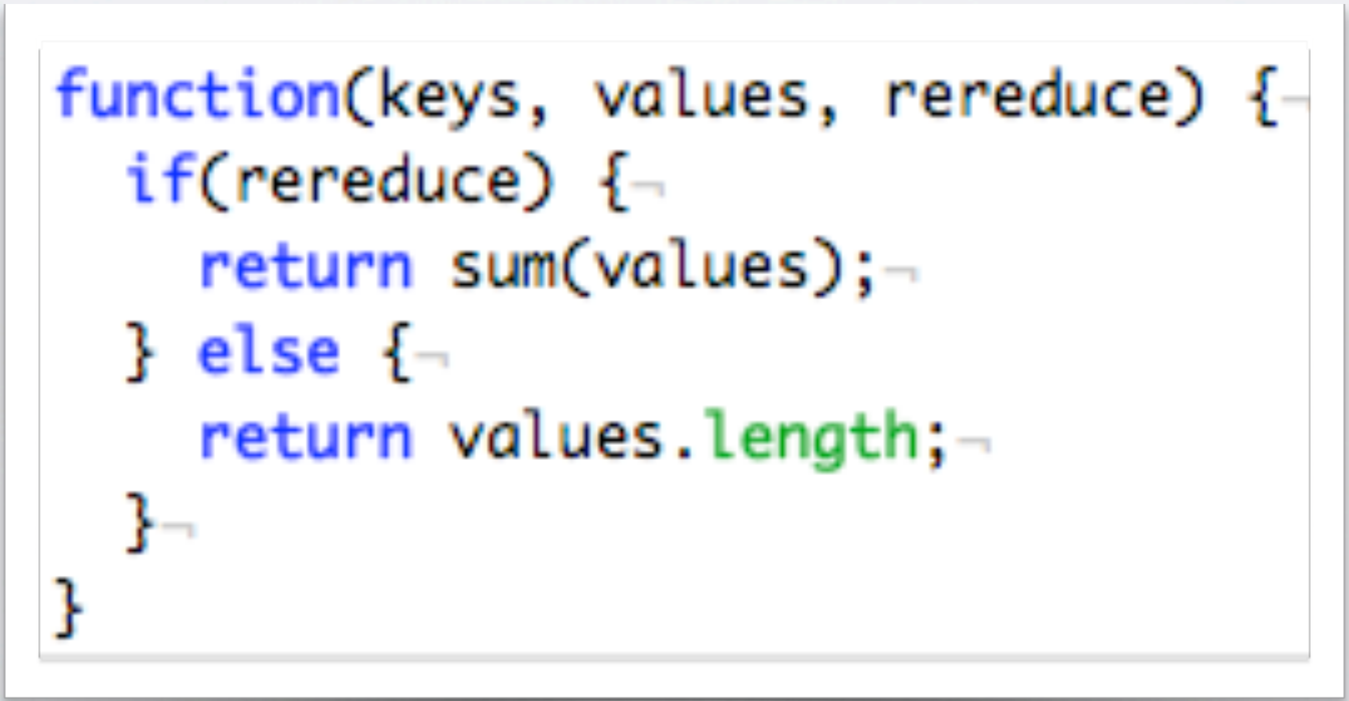


```
function(keys, values, rereduce) {¬
  if(rereduce) {¬
    return sum(values);¬
  } else {¬
    return values.length;¬
  }¬
}
```

WORD COUNT



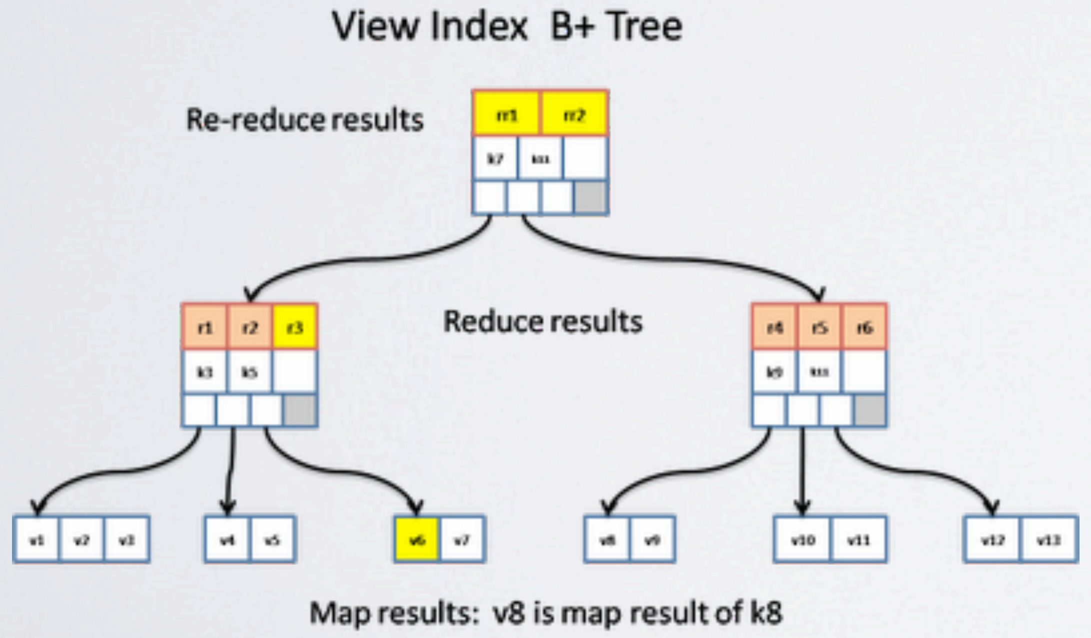
```
function(doc) {
    var words = doc.blob.split(/\W/);
    words.forEach(function(word){
        emit([word.toLowerCase(), doc.title],1);
    });
}
```

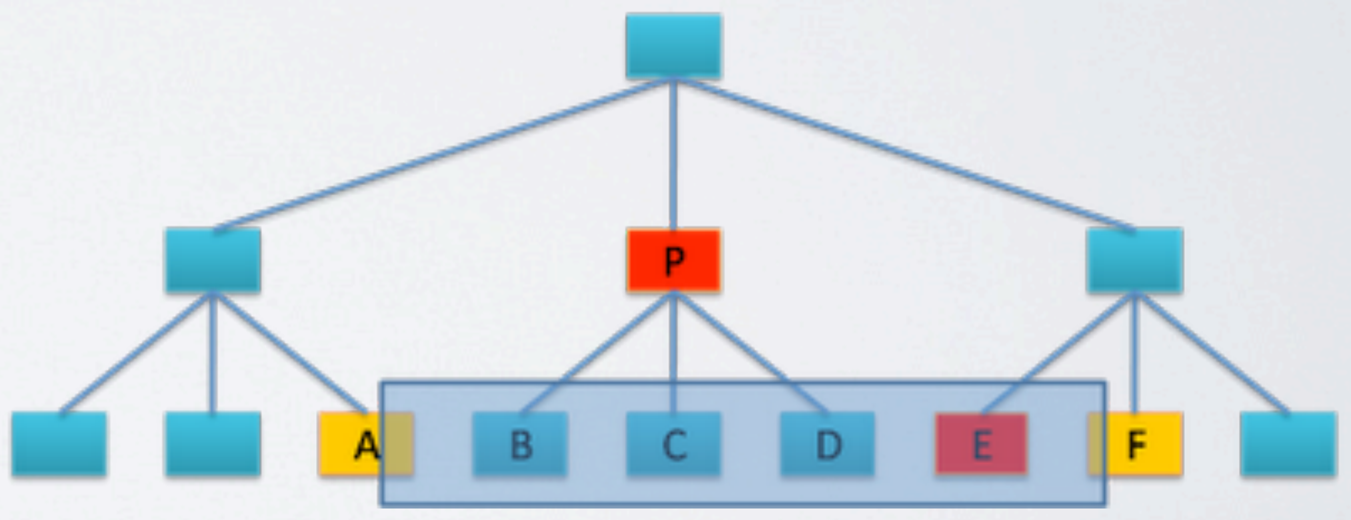


```
function(keys, values, rereduce) {
  if(rereduce) {
    return sum(values);
  } else {
    return values.length;
  }
}
```

INCREMENTAL

- Computing a view can be expensive, so CouchDB saves the result in a B-tree and keeps it up-to-date
- Leaf nodes store map results, inner nodes store reductions of children



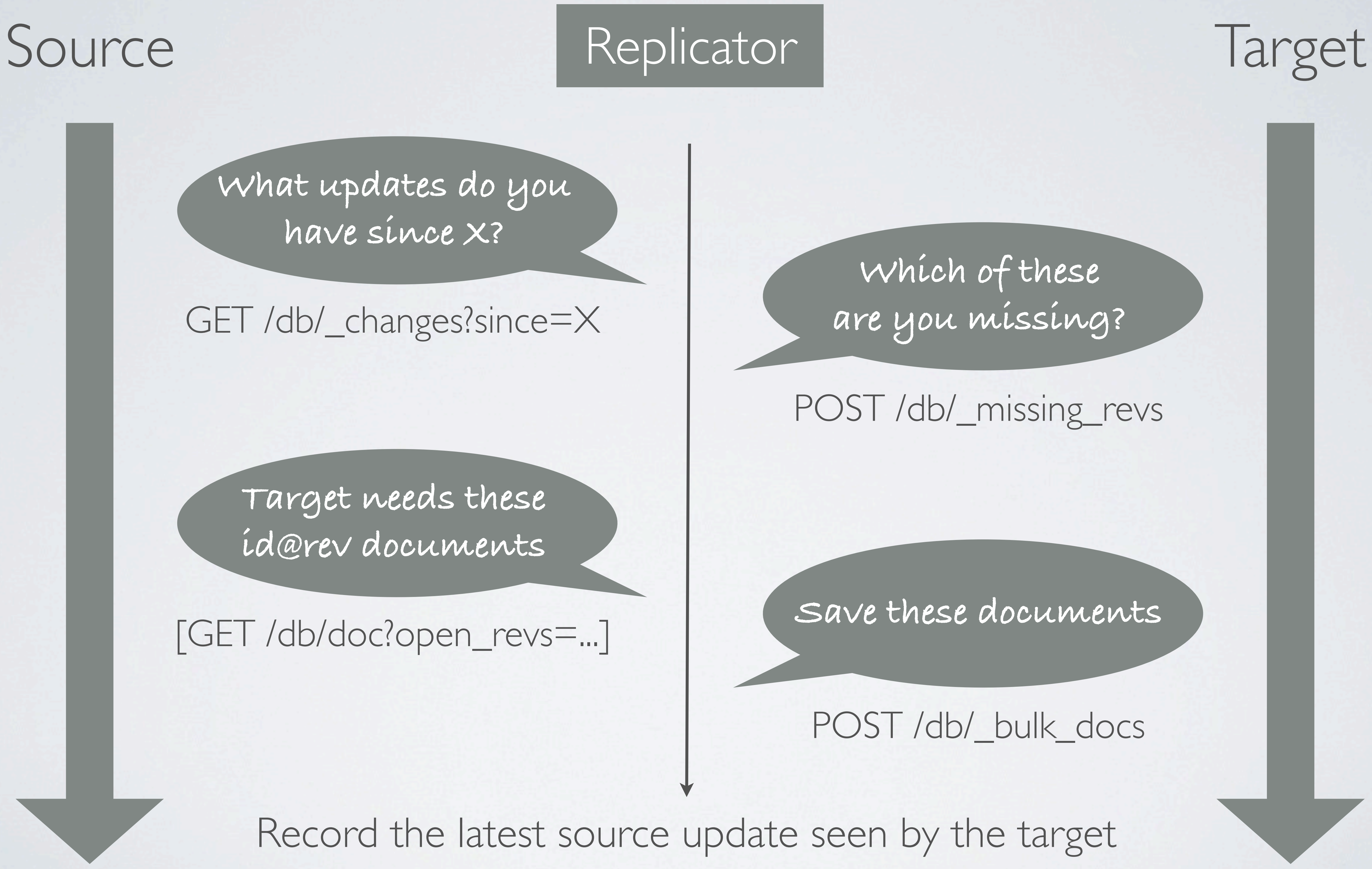


http://horicky.blogspot.com/2008/10/couchdb-implementation.html

REPLICATION

- Peer-based, bi-directional replication using normal HTTP calls
- Mediated by a replicator process which can live on the source, target, or somewhere else entirely
- Replicate a subset of documents in a DB meeting criteria defined in a custom filter function (coming soon)
- Applications (_design documents) replicate along with the data
- Ideal for offline applications -- “ground computing”

REPLICATION


Source
Replicator
Target
What updates do you have since X?
Which of these are you missing?
Target needs these id@rev documents
Save these documents
Record the latest source update seen by the target

GET /db/_changes?since=X

POST /db/_missing_revs

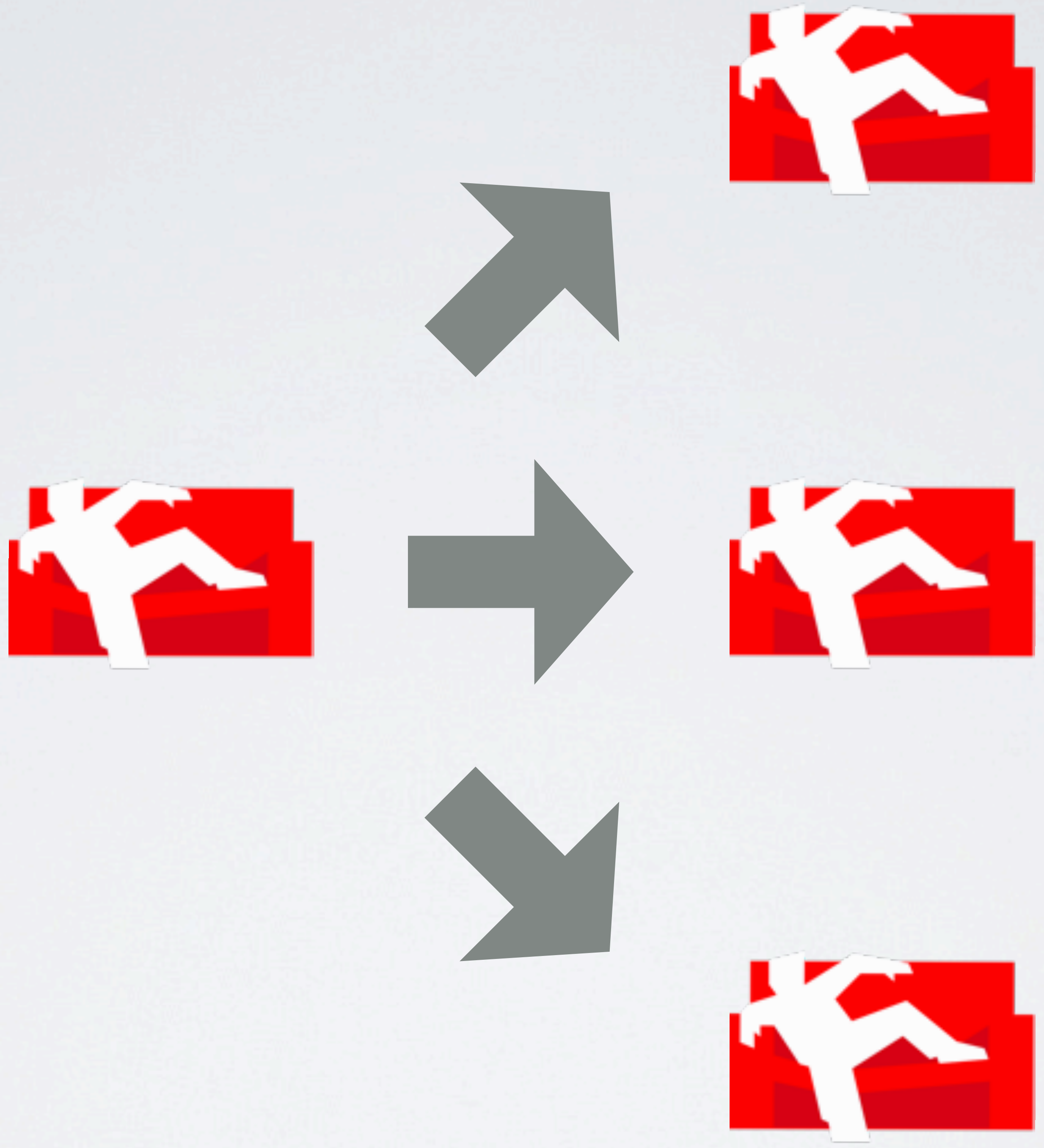
[GET /db/doc?open_revs=...]

POST /db/_bulk_docs

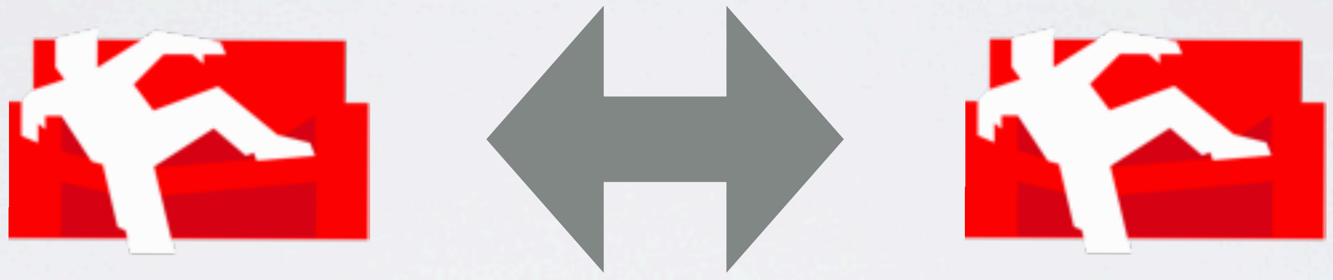
CONFLICT RESOLUTION

- Replication can introduce conflicts in a multi-master setup
- CouchDB deterministically chooses a winner; the loser is saved as a conflict revision
- Conflict revisions are replicated; if you replicate A→B and B→A, both will agree on the winning and losing revisions

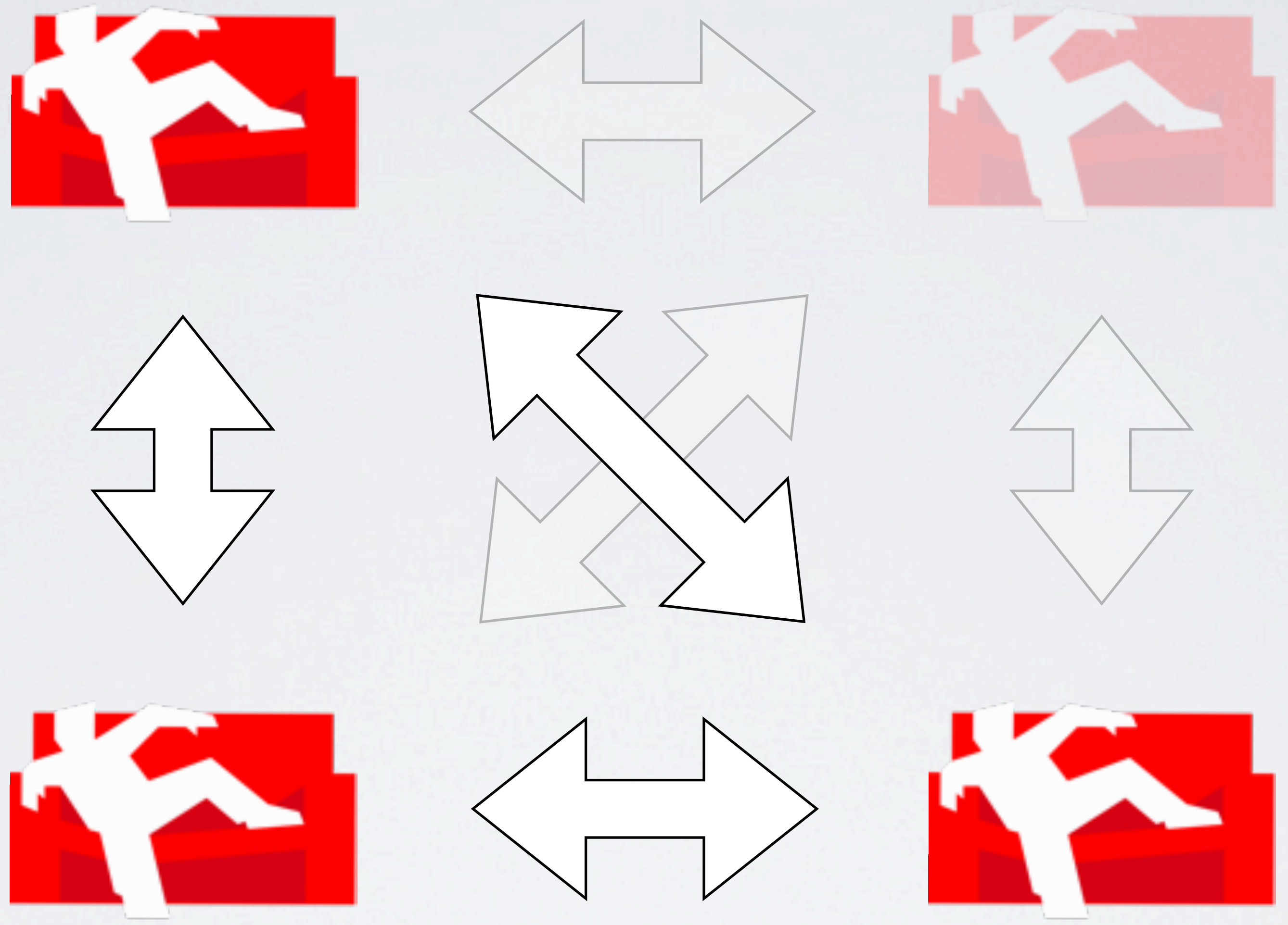
MASTER-SLAVE



MASTER-MASTER



ROBUST MULTI-MASTER



CASE STUDIES

CASE #1: BBC

- Priorities: schema flexibility, robustness, grace under concurrent load
- Incorporated CouchDB as a key/value store into existing application infrastructure
- Chose CouchDB from pool of competitors
- Robust in production for years, continues to scale to meet demands of data, concurrency

CASE #2: MEEBO

- Why CouchDB? "Flexible, fast, query-able datastore that can scale to hundreds of millions of documents. Replication makes failover seamless... It's gotten so popular that many internal dashboards are written as couch-apps"
- Loves? schema-less design, query flexibility, replication, JavaScript mapreduce
- Wishes? Speed, stability, incorporate Lounge (custom nginx/Twisted sharding layer) into couch
- Was it a success? "Absolutely."

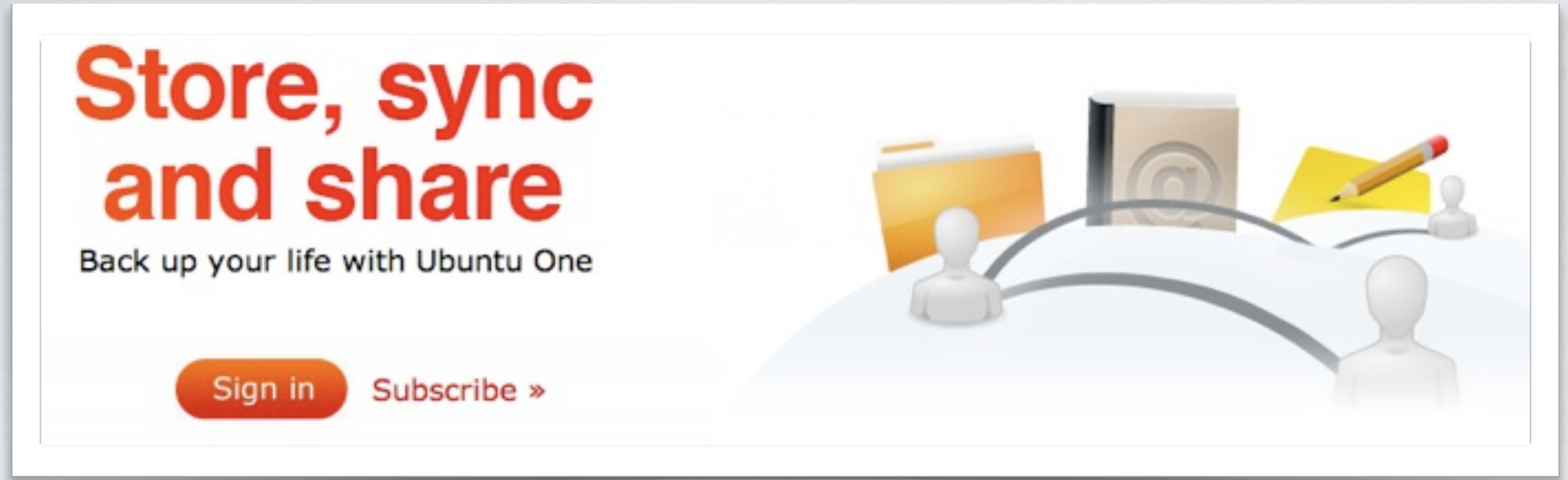
STUDY #3: SCOOPLER

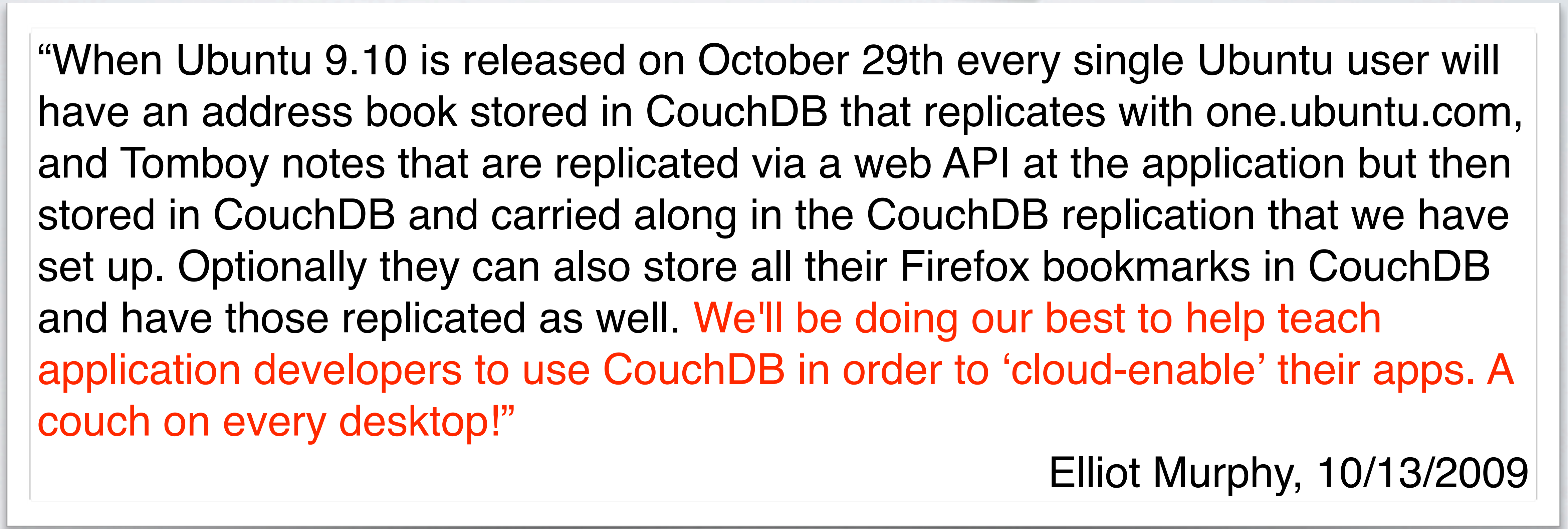
- Realtime aggregation service, large/growing data volumes
- Aggregation, analysis, presentation of realtime feeds
- Schema flexibility crucial, scalability, concurrency
- Well suited to eventual consistency model
- mapreduce excellent fit for ranking/sorting problems

CASE #4: REALTIME ANALYTICS

- High rate advertising analytics.
- Existing 24-hour ETL analysis workflow too slow.
- Complicated SQL stored procedures for social graph construction, traversal, analysis, required 40+ postgres tables
- Replaced with single CouchDB document type and two views:
  - group_level collation to bin data at multiple granularities (ala. RRD) and present to customers in seconds, not hours
  - single mapreduce view (30 lines of JS) for social graph analysis. Paradigm shift from iteration to sorting

CASE #5: UBUNTU 9.10




"When Ubuntu 9.10 is released on October 29th every single Ubuntu user will have an address book stored in CouchDB that replicates with one.ubuntu.com, and Tomboy notes that are replicated via a web API at the application but then stored in CouchDB and carried along in the CouchDB replication that we have set up. Optionally they can also store all their Firefox bookmarks in CouchDB and have those replicated as well. We'll be doing our best to help teach application developers to use CouchDB in order to 'cloud-enable' their apps. A couch on every desktop!"

Elliot Murphy, 10/13/2009

replication is a potential game-changer

WRAPPING UP

- CouchDB: rapidly maturing product with disruptive potential
- From interest to adoption: in production at scale
- Development and future:
  - one of (only?) NoSQL's not developed internally for a specific problem
  - Initial focus on API, robustness, concurrency, replication
  - Accelerating emphasis on features: performance, clustering, iterative mapreduce, ...

relax

BUILDING A BIG COUCH

- Use all your familiar tools for HTTP infrastructure
- Cache-friendly: document _rev is an ETag
- Sharding with CouchDB-Lounge
  - nginx upstream_hash "dumbproxy" for CRUD operations
  - Twisted "smartproxy" for merging view results
- Active development on dynamic sharding using distributed Erlang

STUFF I DIDN'T TALK ABOUT

- Authentication using HTTP Basic, Cookie, OAuth
- couchapp: serve lightweight HTML+JavaScript web apps directly out of CouchDB using _show and _list functions to transform JSON [1]
- _external indexers, including full-text search powered by CouchDB-Lucene [2]
- The many excellent client libraries, view servers, etc. contributed by the community [3]

[1]: http://github.com/couchapp/couchapp

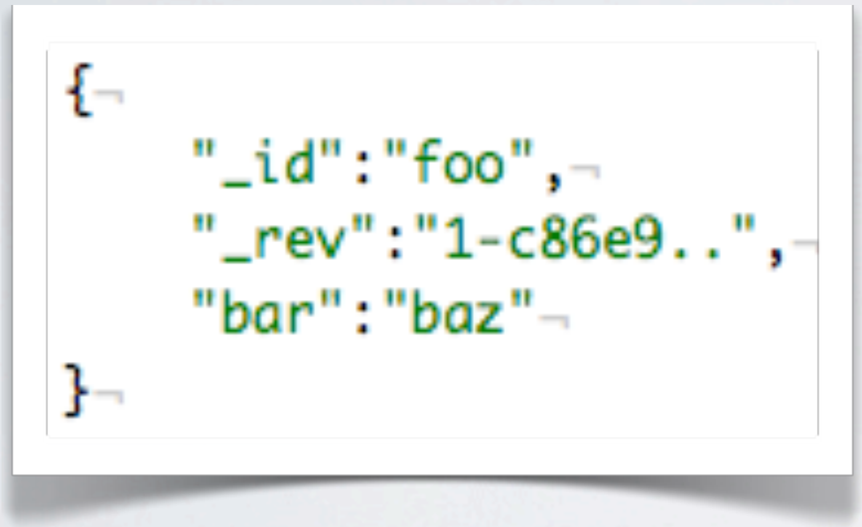
[2]: http://github.com/rnewson/couchdb-lucene

[3]: http://wiki.apache.org/couchdb/Related_Projects

CONFLICT RESOLUTION

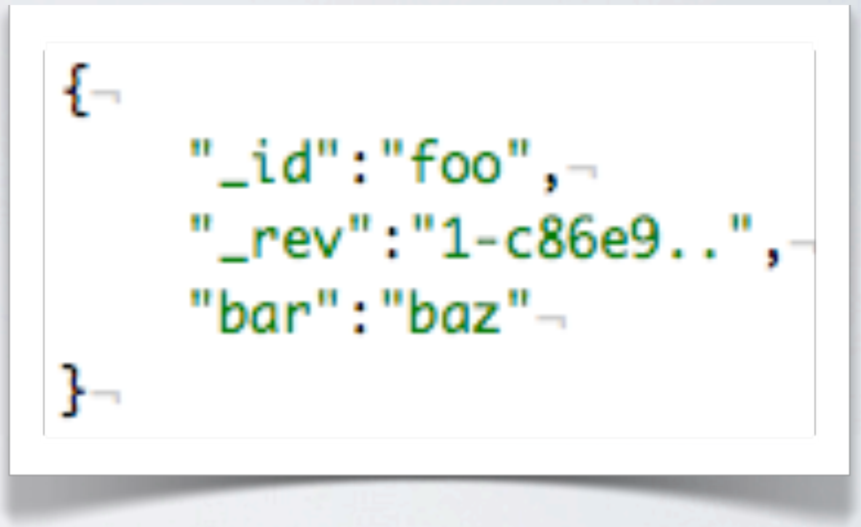
Case 1: save the same update to multiple replication partners

PUT /a/foo

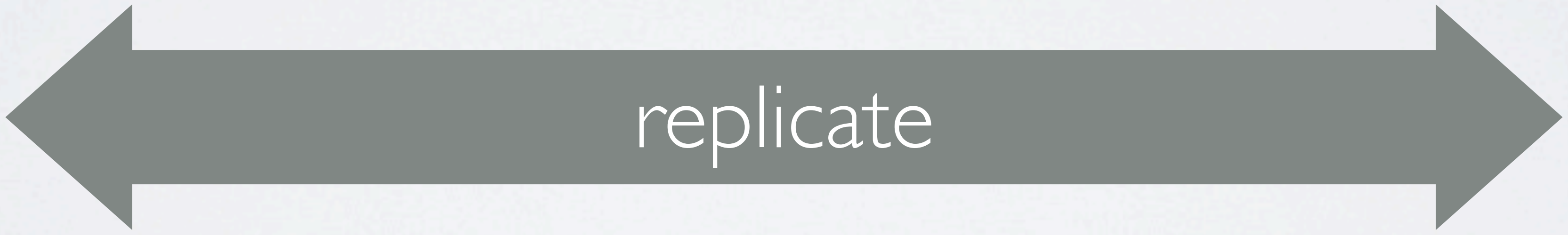


```
{¬
    "_id":"foo",¬
    "_rev":"1-c86e9..",¬
    "bar":"baz"¬
}¬
```

PUT /b/foo



```
{¬
    "_id":"foo",¬
    "_rev":"1-c86e9..",¬
    "bar":"baz"¬
}¬
```

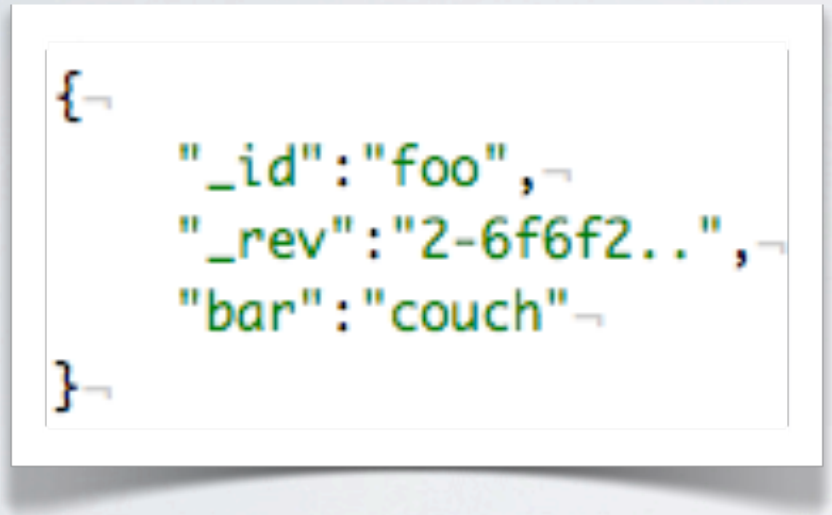


No Conflict

CONFLICT RESOLUTION

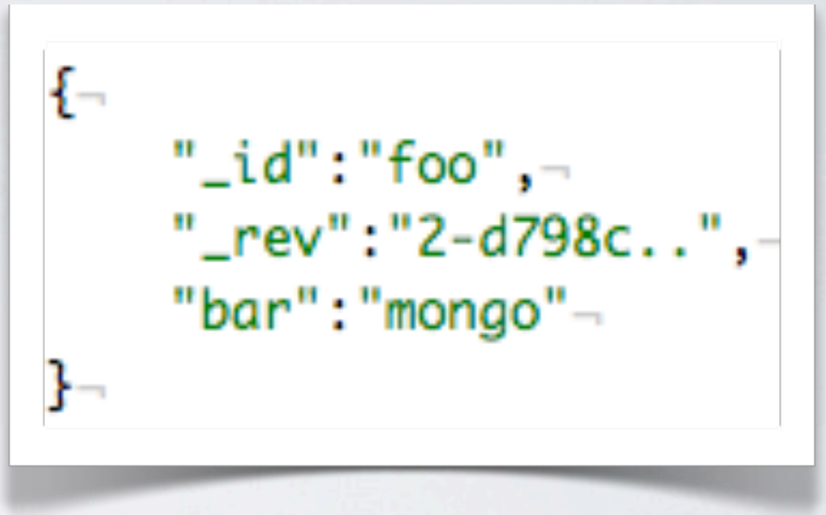
Case 2: save different updates to the same document

PUT /a/foo

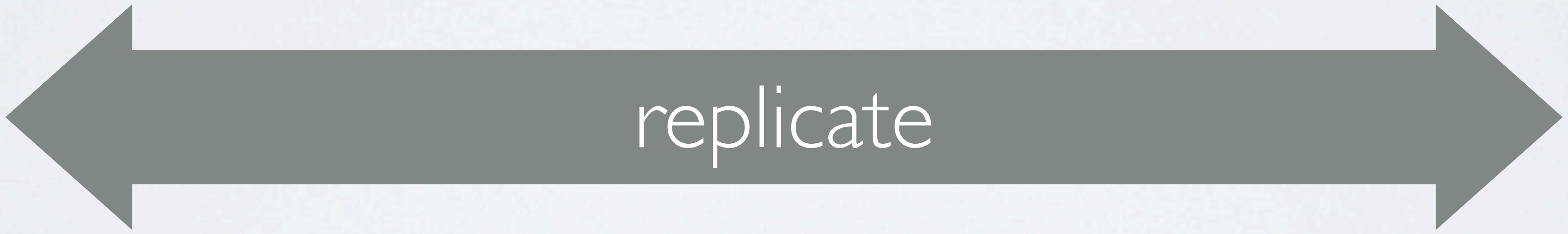


```
{
    "_id":"foo",
    "_rev":"2-6f6f2..",
    "bar":"couch"
}
```

PUT /b/foo

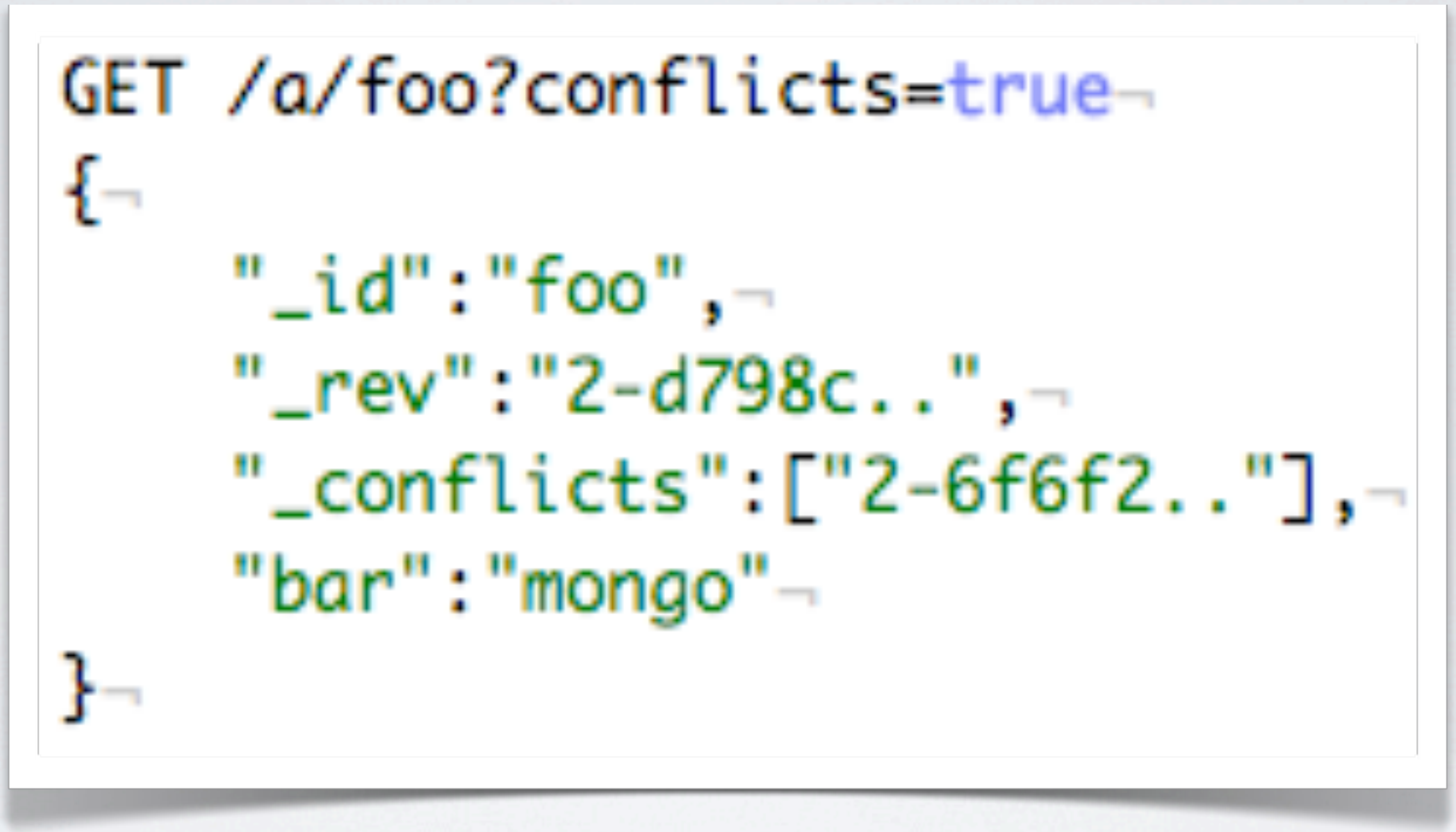


```
{
    "_id":"foo",
    "_rev":"2-d798c..",
    "bar":"mongo"
}
```



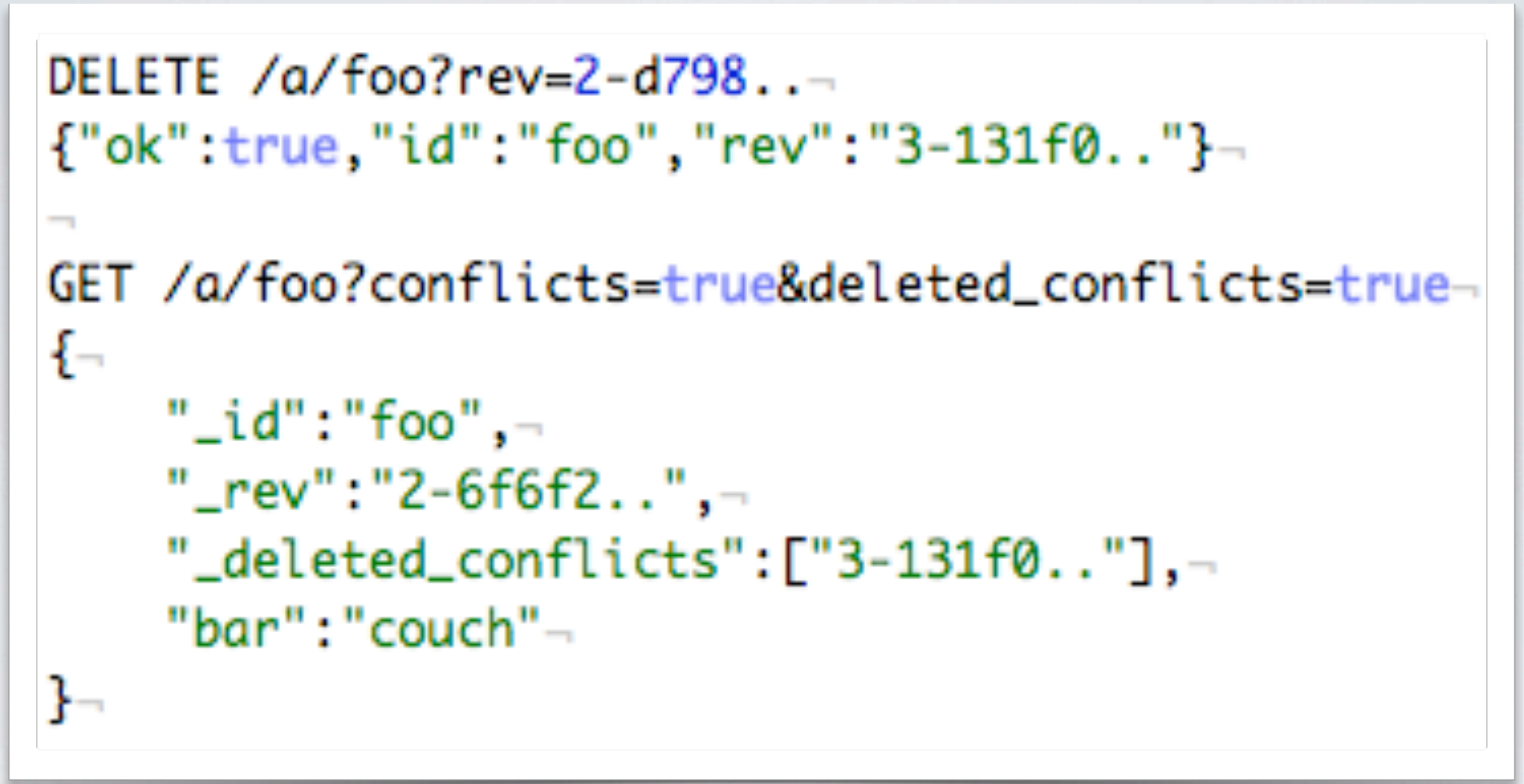
Conflict

CONFLICT RESOLUTION



```
GET /a/foo?conflicts=true¬
{¬
    "_id":"foo",¬
    "_rev":"2-d798c..",¬
    "_conflicts":["2-6f6f2.."],¬
    "bar":"mongo"¬
}¬
```

CONFLICT RESOLUTION

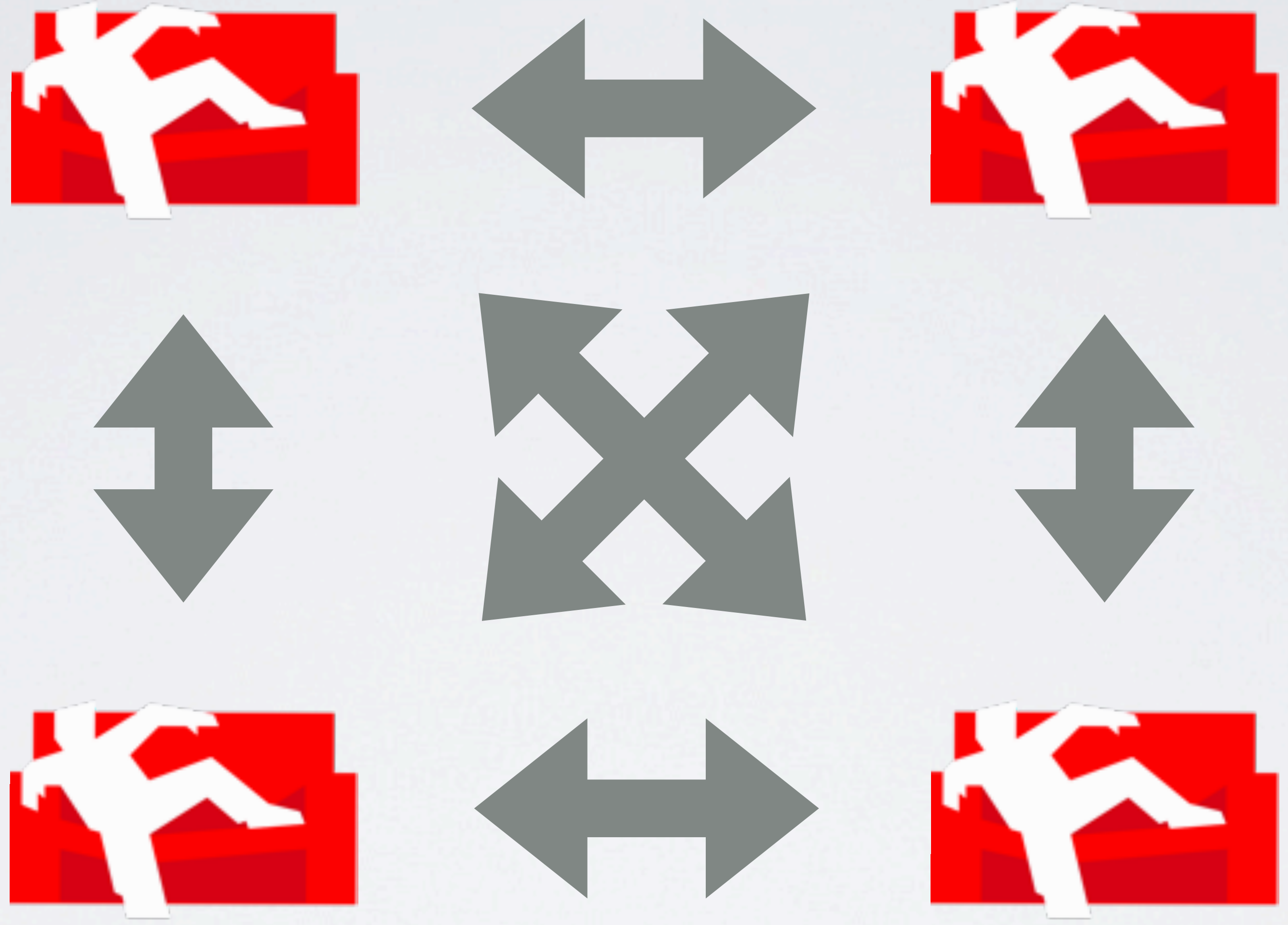


```
DELETE /a/foo?rev=2-d798..
{"ok":true,"id":"foo","rev":"3-131f0.."}

GET /a/foo?conflicts=true&deleted_conflicts=true
{
    "_id":"foo",
    "_rev":"2-6f6f2..",
    "_deleted_conflicts":["3-131f0.."],
    "bar":"couch"
}
```

No Conflict

ROBUST MULTI-MASTER



ROBUST MULTI-MASTER