Extend CouchDB and build apps based on CouchDB

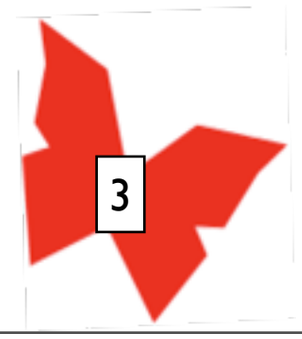
erlang factory lite paris 2011
2011-09-25 - benoît chesneau

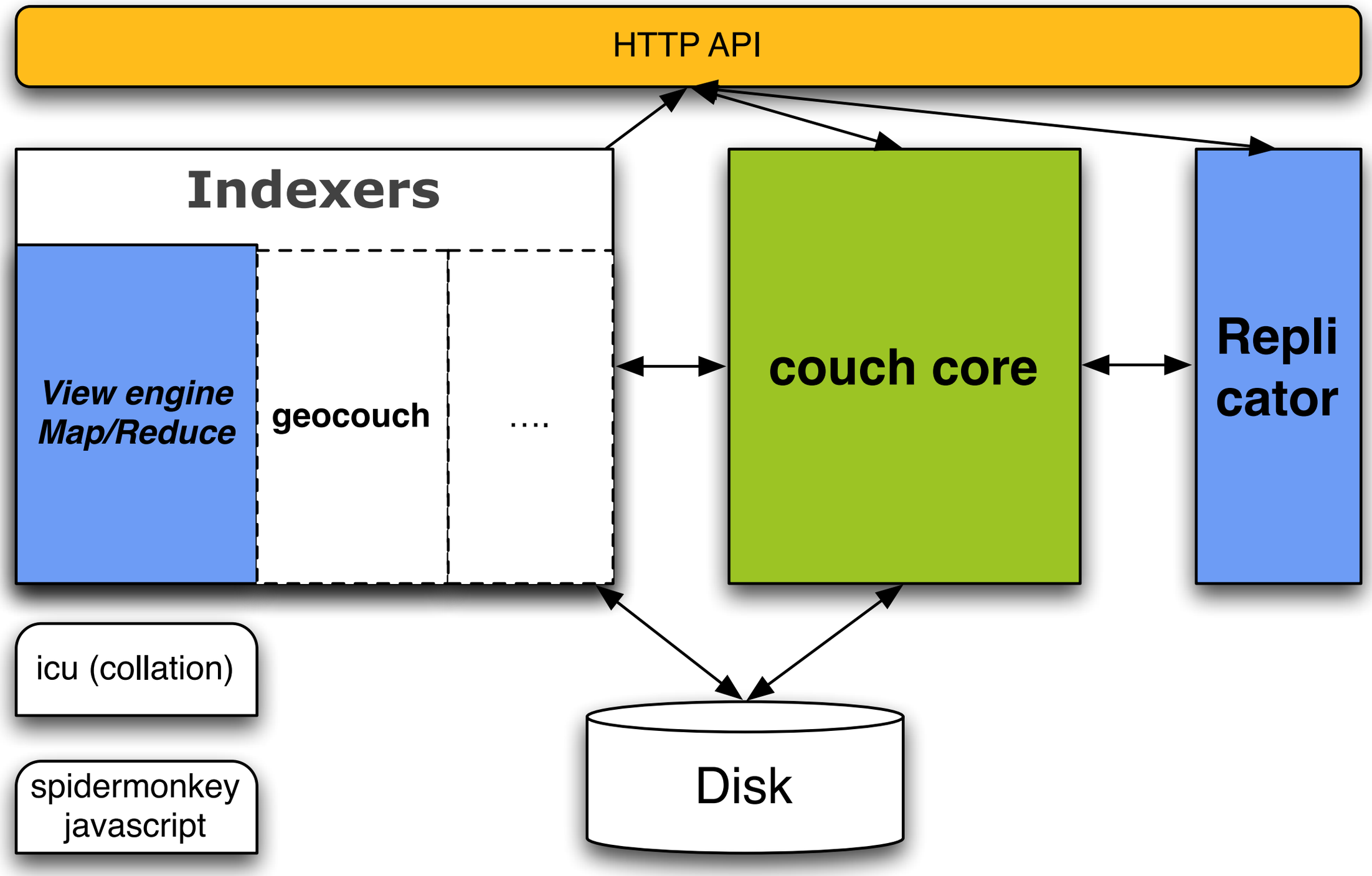
- Web craftsman
- Writing opensource for a living
- benoitc@apache.org
- couchbeam author, gunicorn author ...

Me

CouchDB ?

- Document Oriented
- HTTP
- Erlang
- Javascript
- Append Only File Structure
- B-Tree, R-Tree (geocouch)
- multiplatform




HTTP API
Indexers
View engine
Map/Reduce
geocouch
....
couch core
Repli
cator
icu (collation)
spidermonkey
javascript
Disk

HTTP

$ curl -XGET http://127.0.0.1:5984
{"couchdb":"Welcome","version":"1.1.0"}

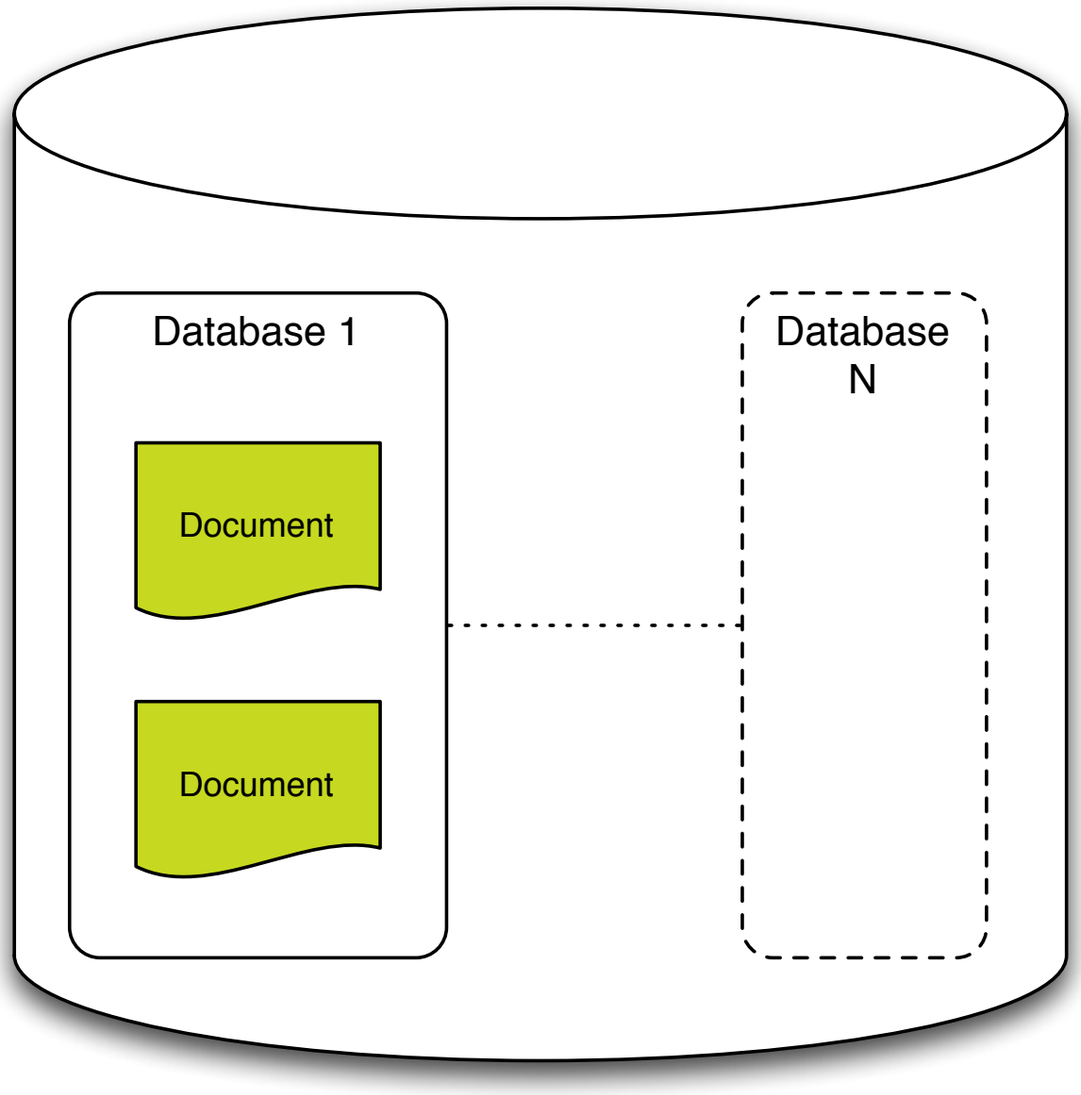
CouchDB urls

/db	Operations in a database
/db/docid	Document operations
/db/_local/docid	Operations on a local document
/db/_design/designid	Design documents provides methods and needed informations to get data from the DB using the M/R engine or transform data. (Used also by some full text query engines)
/_<special>	Any special resources are prefixed by a "_"

What do you manipulate?

- Database
- Document in JSON
  - Design Document
  - Local Document

Couchdb node



Database

- can contains 1 or more documents
- is replicable
- can be protected

Document

- JSON
- can contain attachments
- _design documents: where views & render functions are stored
- _local documents: can't be replicated

CouchDB Document (JSON)

metadata _id & _rev

properties


```
{
  _id: ”098F6BCD4621D373CADE4E832627B4F6”,
  _rev: "4134847",
  “title”: “Tour de France”,
  “author”: “KraftWerk”,
  “length”: {
        “value“: 45,
        “unit“: “mn“
  },
  “description“: “.....“
  “docType”: "MusicAlbum",
  "categories": ["electro", "80's"],
  “created“": “2008-05-15T23:31:19Z“
}
```

CouchDB Document

revisions

- CouchDB doesn't update a doc in place. Instead it creates a new one at the end of the database file with a new revisions
- Need to be passed on update (& delete)
- Revisions are used internally to check conflicts
- Used also in the replication

Don't use the revisions in your application for a different usage.

CouchDB Document - attachment

attachment

```
{

"_id":"098F6BCD4621D373CADE4E832629B4F6",
"_rev":"3-6453069c8677dfbc6543000687d0a6a2",
    "title":"Silence",
    "author":"portishead",
    ...

    "_attachments":{
        "05 Plastic.m4a":{
            "content_type":"audio/x-m4a",
            "revpos":2,"digest":
            "md5-mkQc/p2SWnEHug6/+zJqWw==",
            "length":7580553,"stub":true
        }
    }
}
```

CouchDB Document - Miscellaneous properties

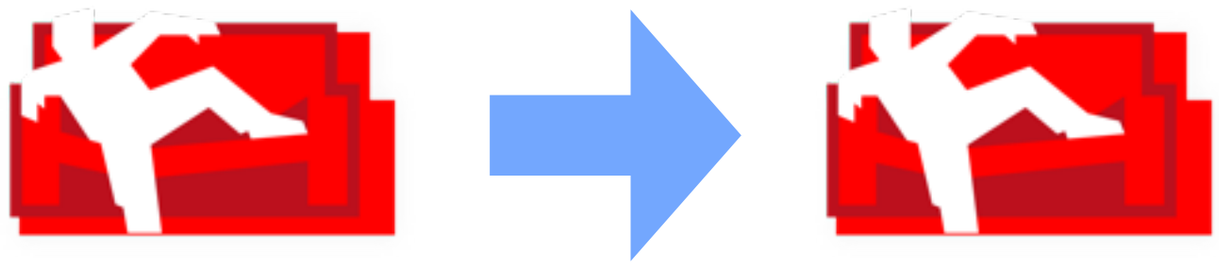
- _deleted : when a doc has been deleted
- _revs_info : list of current revisions
- _conflicts
- _deleted_conflicts

replication

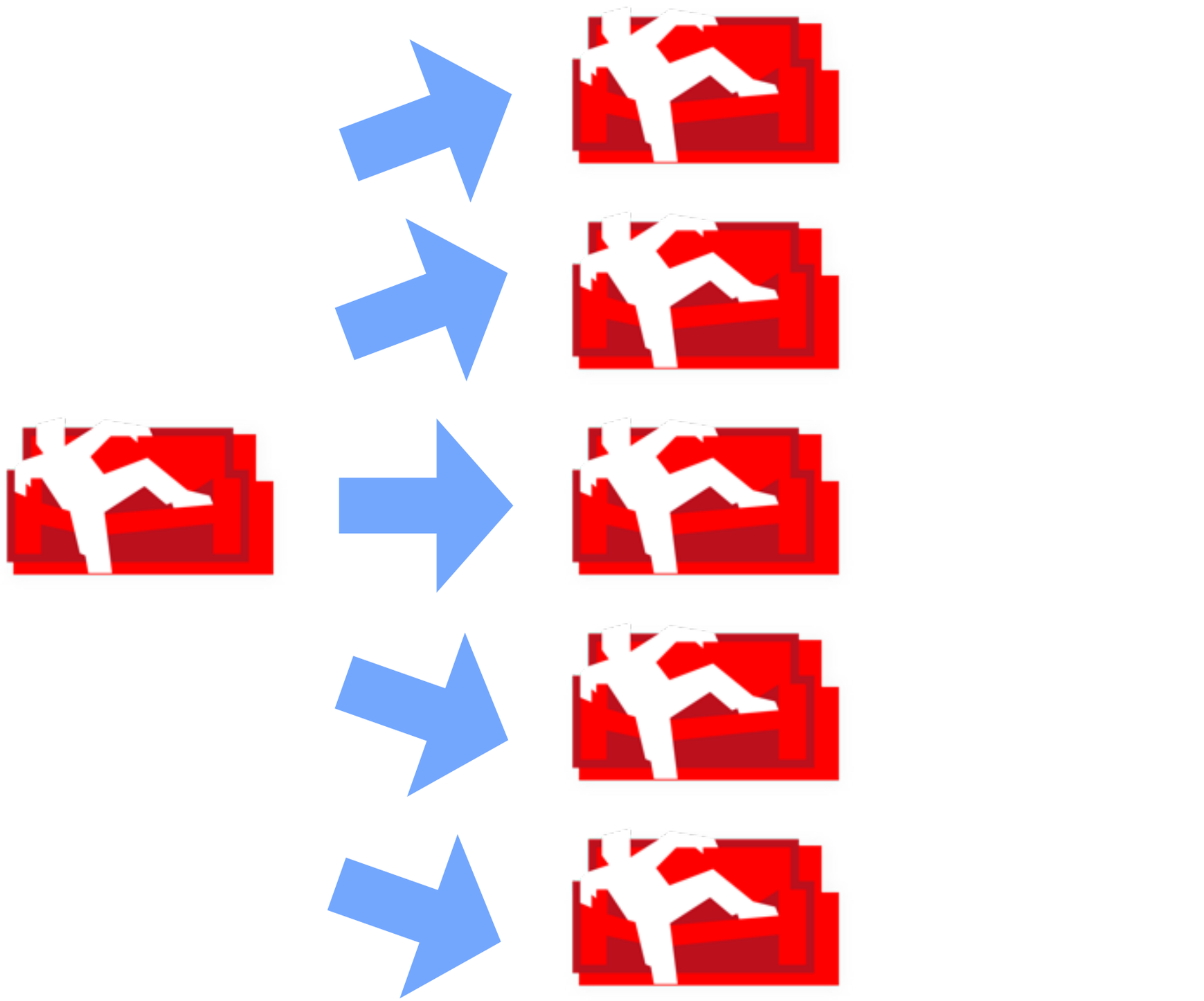
- master-master
- incremental
- over HTTP
- Continuous feed

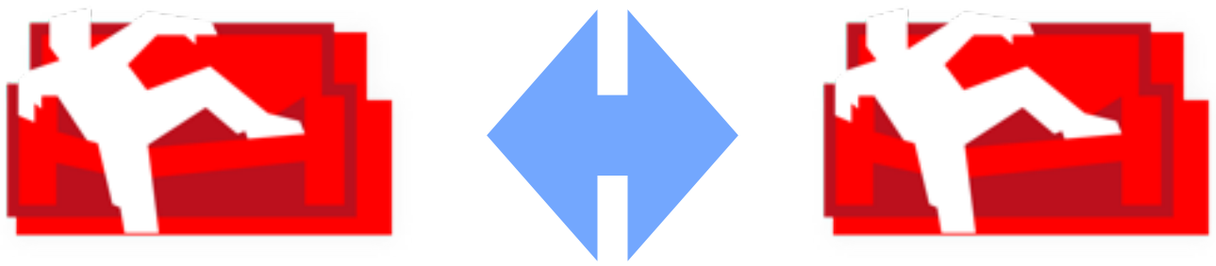
INCREMENTAL REPLICATION

ENKI

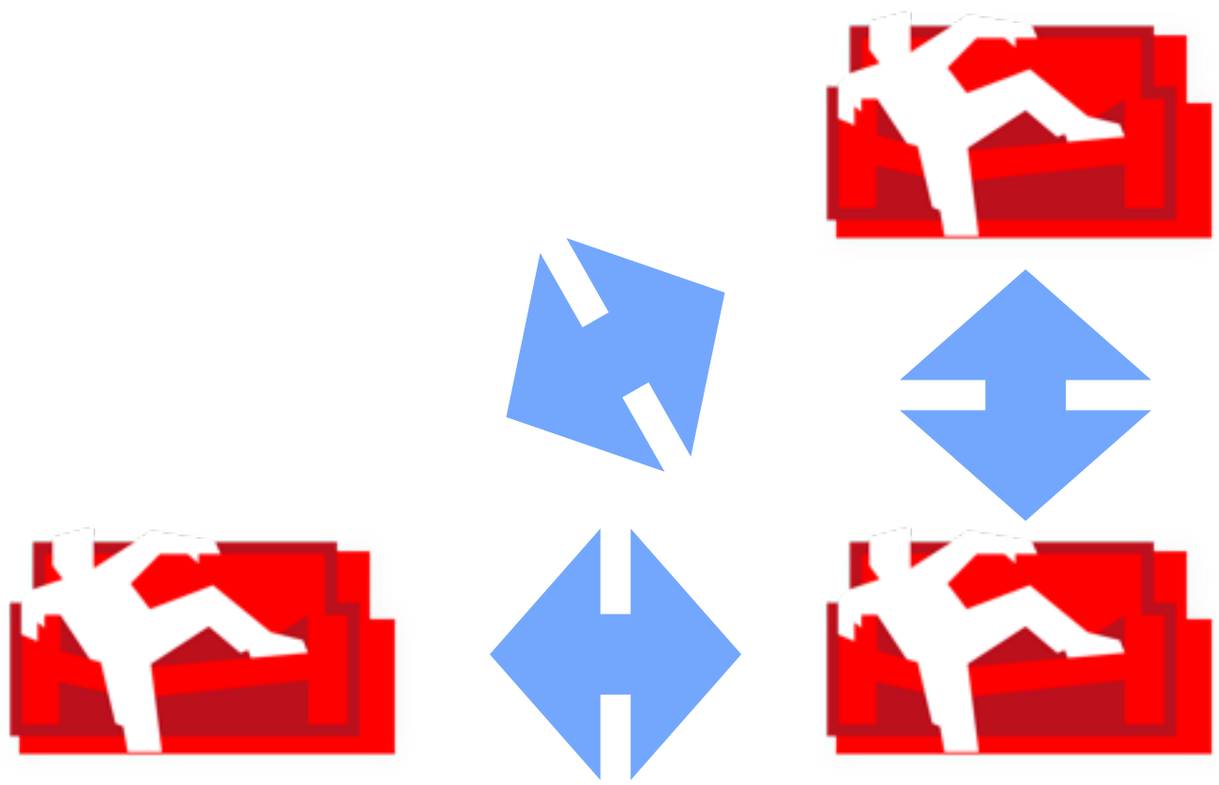



ENKI

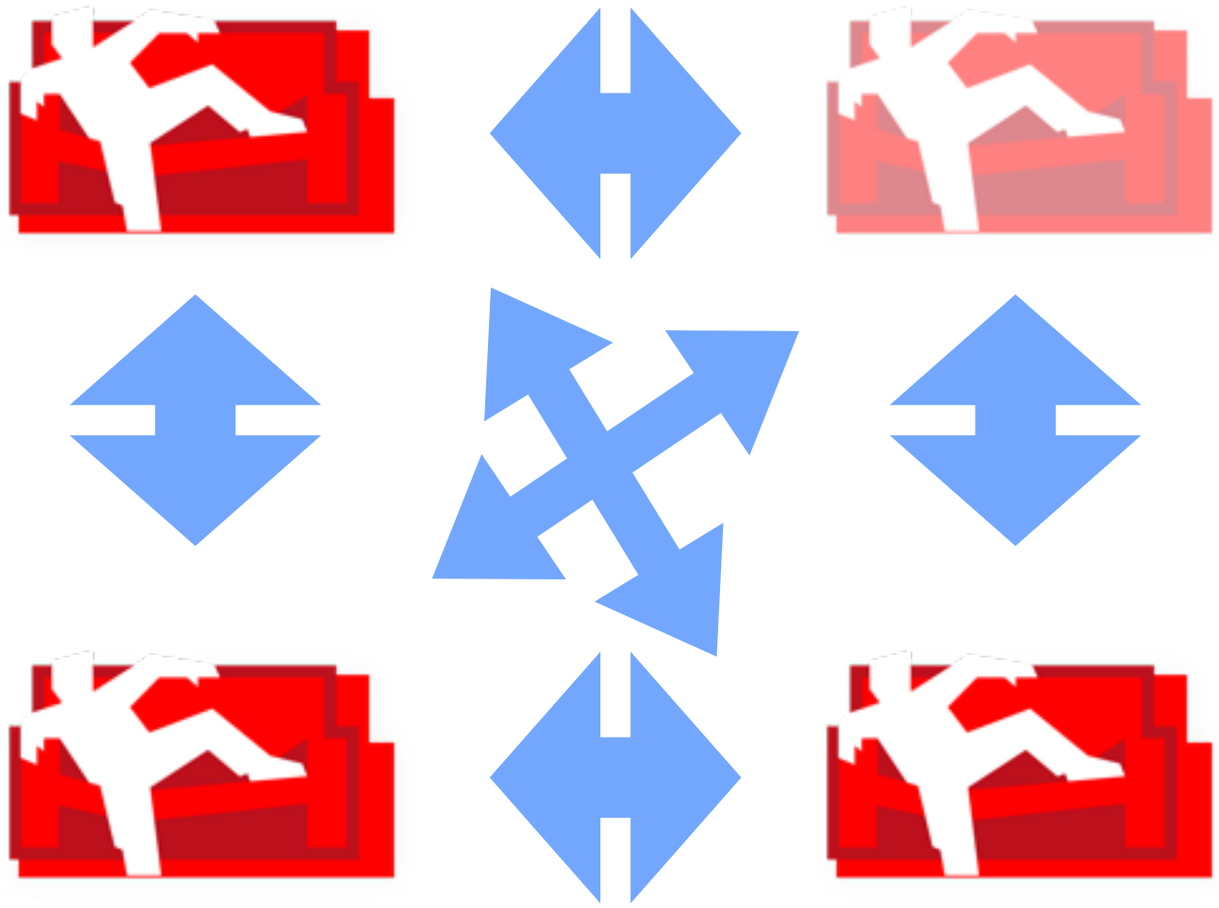




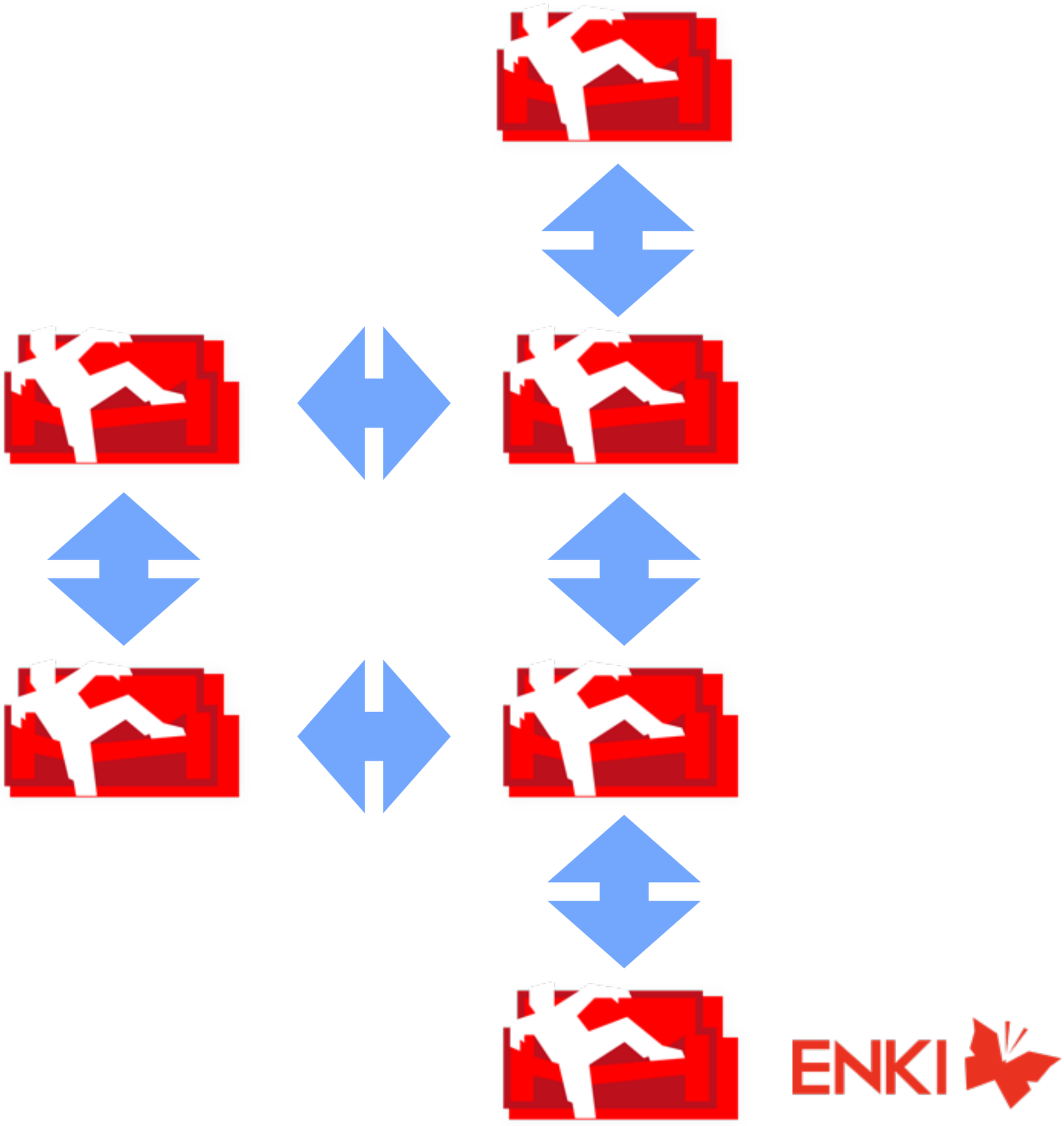

ENKI

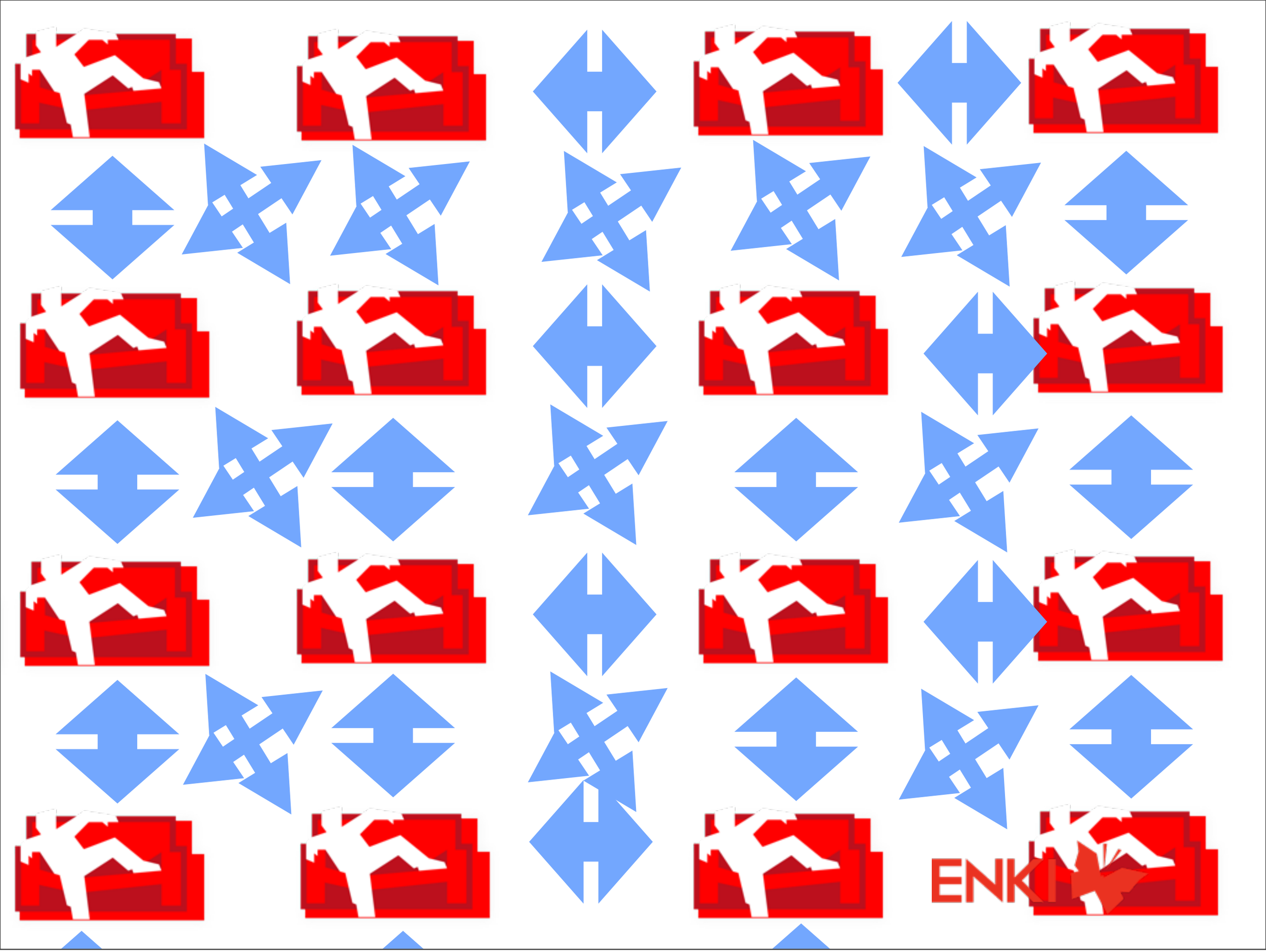



ENKI



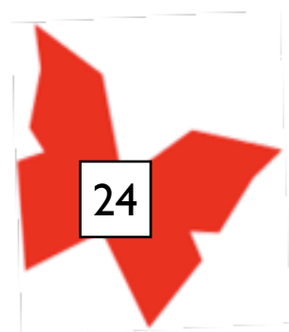

ENKI


ENKI


ENKI

Réplication : Use cases

- Inbox - Mailing lists
- Shard data
- Synchronize machine (icloud but multiplateform?)



API

2 clients

- couchbeam: https://github.com/benoitc/couchbeam . use HTTP API
- couchc https://github.com/benoitc/couchc full erlang counterpart to couchbeam

HTTP API - DB

- Fetch Documents

```
$ curl -XGET localhost:5984/chest/_all_docs
{"total_rows":0,"offset":0,"rows":[]}
```

- Delete

```
$ curl -XDELETE localhost:5984/chest
{"ok":true}
```

- List of databases

```
$ curl -XGET localhost:5984/_all_dbs
["_replicator","_users","chest","testdb"]
```

HTTP API - DB

couchbeam

```
% create a database
Options = [],
{ok, Db} = couchbeam:create_db(Server, "testdb",
Options).

%% open a database
{ok, Db} = couchbeam:open_db(Server, "testdb",
Options).
```

couchc

```
%% create a database
Options = [{user_ctx,
#user_ctx{roles=[<<"_admin">>]}}],
{ok, Db} = couchc:create_db("couchc_testdb", Options).

%% open a database
{ok, Db} = couchc:open_db("couchc_testdb", Options).
```

HTTP API - Get all docs

/db/**_all_docs**.

```
curl -XGET localhost:5984/chest/_all_docs
```


```
{"total_rows":3,"offset":0,"rows":[
{"id":"20110725-1","key":"20110725-1","value":{"rev":"1-002c67027dc67c1ec62fcd2c7f5a7582"}},
{"id":"t20110725-1","key":"t20110725-1","value":{"rev":"1-77724440b82c743ebae8d44482a184f1"}},
{"id":"t20110912-1","key":"t20110912-1","value":{"rev":"1-77724440b82c743ebae8d44482a184f1"}}
]}
```


_all_docs est a **specialized view** of the document Which means you can use the same query parameters you use in a view. Ex, to get docs in inversed order, use the parameter `descending=true` :


```
{"total_rows":3,"offset":0,"rows":[
{"id":"t20110912-1","key":"t20110912-1","value":{"rev":"1-77724440b82c743ebae8d44482a184f1"}},
{"id":"t20110725-1","key":"t20110725-1","value":{"rev":"1-77724440b82c743ebae8d44482a184f1"}},
{"id":"20110725-1","key":"20110725-1","value":{"rev":"1-002c67027dc67c1ec62fcd2c7f5a7582"}}
]}
```

couchbeam

```
{ok, AllDocs} = couchbeam_view:fetch(Db, 'all_docs', []).
{ok,[{[{<<"id">>,<<"7a0ce91d0d0c5e5b51e904d1ee3266a3">>},
          {<<"key">>,<<"7a0ce91d0d0c5e5b51e904d1ee3266a3">>},
          {<<"value">>,
           {[{<<"rev">>,<<"15-15c0b3c4efa74f9a80d28ac040f18bdb">>}]}}},
        ]}.
```

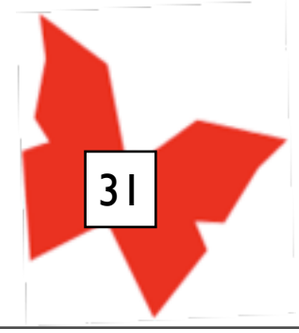
couchc

```
{ok, {TotalRowsCount, Offset, Results1}} = couchc:all(Db).
{ok,{1,1,
     [{[{id,<<"86d2a33eed8e6555a506cacace1f39b4">>},
        {key,<<"86d2a33eed8e6555a506cacace1f39b4">>},
        {value,{[{rev,<<"1-967a00dff5e02add41819138abb3284d">>}]}}]}]}}
```

HTTP Api - Document

PUT: update

- _id & _rev properties are required.
- Option batch=ok -> speed insertion by delaying it
- _ensure_full_commits
- **POST** can be used to create a document and gerenerate automatically an _id. But it's better to use PUT when you face to a firewall (duplication errors)



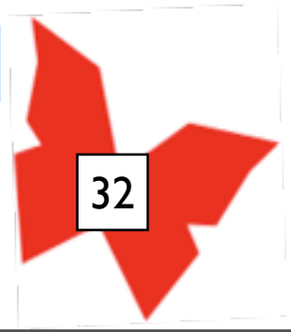
API HTTP - Document

PUT: update

```
$ curl -XPUT -d '{
      "_id": "i20110602",
      "_rev": "1-e30af3c8895165d153317ae5e98f88a4",
      "title": "Invoice D100",
      "tags": [ "invoice", "development" ],
      "description": "some description",
      "client": "client1",
      "type": "invoice",
      "date": "2011-06-02T09:01:12Z+01:00"
}' -H "Content-Type: application/json" localhost:5984/
coffre/20110602-1

      {
          "ok":true,
          "id":"i20110602",
          "rev":"2-a1052d78554433927ee0a644e7d083c6"
      }
```

dernière révision

nouvelle révision



HTTP Api- Document

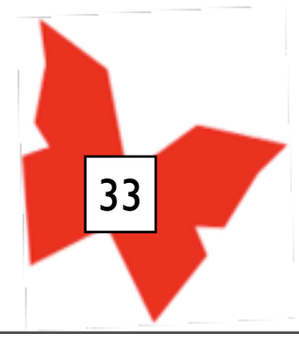
DELETE: Delete a document

last revision

```
$ curl -XDELETE localhost:5984/chest/20110602-1?rev=2-
a1052d78554433927ee0a644e7d083c6
```

new revision

```
{
    "ok":true,
    "id":"i20110602",
    "rev":"3-4ecc45338fe5c4a2ac8ca1893a5ef708"
}
```



couchbeam

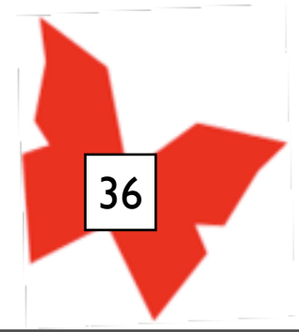
```
Doc = {[
{<<"_id">>, <<"test">>},
{<<"content">>, <<"some text">>}
]},
%% save a document
{ok, Doc1} = couchbeam:save_doc(Db, Doc).

%% open a document
{ok, Doc2} = couchbeam:open_doc(Db, "test").
```

couchc

```
%% save a document
{ok, DocID, DocRev} =  couchc:save_doc(Db, Doc).

%% open a document
{ok, Doc1} = couchc:open_doc(Db, DocId).

%% delete a document
{ok, DocID, NewDocRev} = couchc:delete_doc(Doc, {DocId, DocRev}).
```

HTTP Api - Attachments

- Like an email, attach binaries to your doc
- 2 APIs :
  - **inline** : added to the document object before to be sent
  - **standalone**: send attachments alone.



HTTP Api- Attachments

Get an attachment

```
$ curl -XGET localhost:5984/chest/note-20110725-0000/hello.txt
```

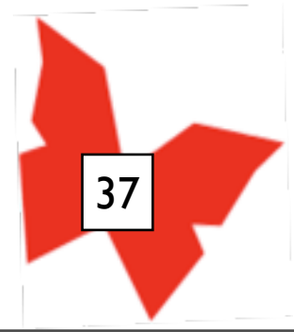
docid

attachement “hello.txt”

return :

```
hello world
```

Note : You can create a directory structure by adding “/” in the attachment name.



couchbeam

```
DocId = "test",
AttName = "test.txt",
Att = "some content I want to attach",
Options = []

%% send attachment
{ok, _Result} = couchbeam:put_attachment(Db, DocId, AttName, Att, Options).

%% fetch attachment
{ok, Att1} = couchbeam:fetch_attachment(Db, DocId, AttName).
```

couchc

```
%% save attachment
{ok, DocId, DocRev} = couchc:save_attachment(Db, DocId, AttName,
Att, Options).

%% fetch attachment
{ok, Att1} = couchc:open_attachment(Db, DocId, AttName)
```

- async api
- couchbeam:stream_fetch_attachment/6
- couchc:open_attachment_async/5

API HTTP - Bulk Documents

Edit or Create multiple docs in one request

method

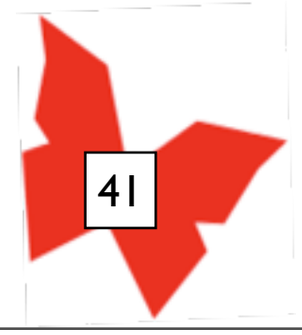
docs

```
curl -XPOST -d'{
    "docs": [
        { "type": "note", "title": "test", "date":"2011-07-25T00:00:00Z+01:00"},
        { "type": "note", "title": "test 2", "date":"2011-07-25T00:00:00+01:00"}
    ]
}' -H"Content-Type: application/json" localhost:5984/chest/_bulk_docs
```

_bulk_docs handler.

Résultat

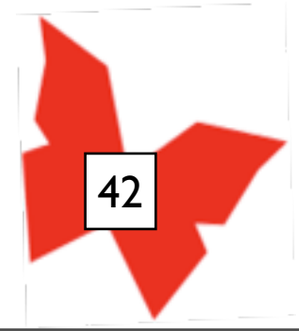
```
[
    {"id":"201ccc0bc2ceee00dae47299c1001fc6","rev":"1-4d9a53af70fdf8d59a71c25b857e708a"},
    {"id":"201ccc0bc2ceee00dae47299c10029d6","rev":"1-76306d88cb8594f65421b9cbad9a610d"}
]
```



Api HTTP - Bulk updates

Edit multiple docs in 1 request

- _id can be generated if you don't put it in the doc during the **creation**.
- To remove a document, add a `_deleted` property to the document.
- 2 models
- non-atomic: soem docs will be saved, those who can't be saved are also returnerd.
- all-or-nothing :`all_or_nothing: true` . If any conflict happen, 0 docs will be saved.



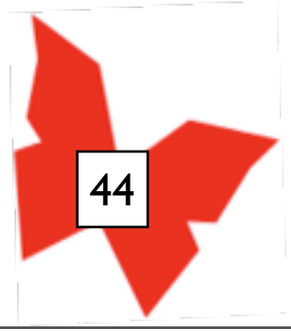
- couchbeam:save_docs/3, couchbeam:delete_docs/3

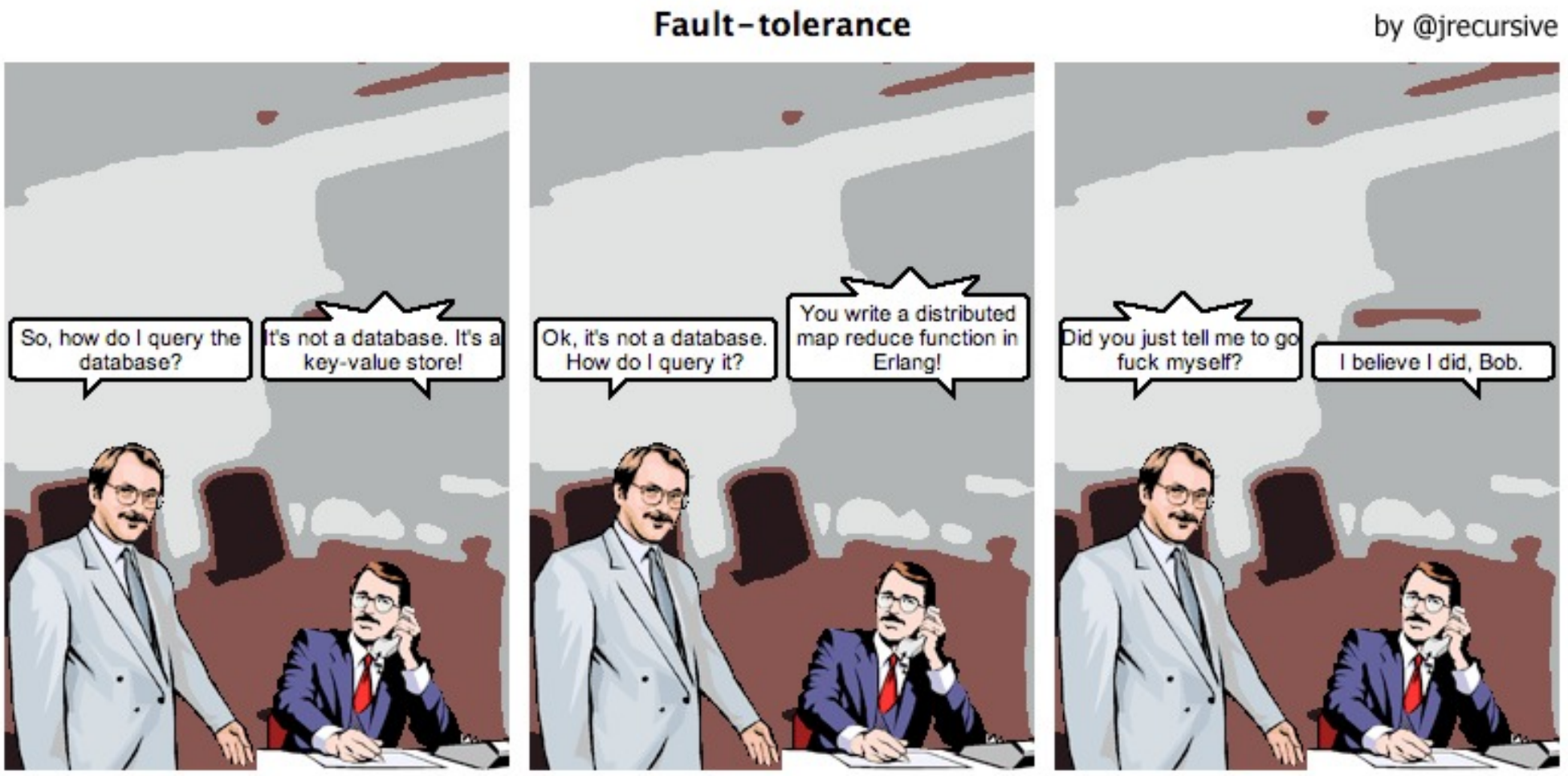
- couchc:save_docs/3, couchc:delete_docs/3

Views

http://wiki.apache.org/couchdb/Using_Views

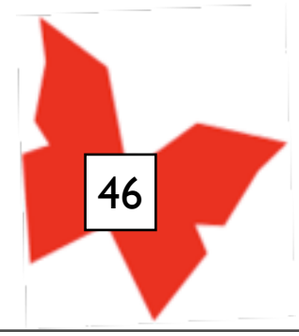
- Map/Reduce to extract informations
- By default javascript is used to write functions but virtually any language can be used/sQS
- functions are put in a document design : **_design/docid**
- Results are incremented
- Calculated when you call them. (remember to do it often)
- Rows are sent line by line to the HTTP client (really efficient).




Fault-tolerance
by @jrecursive
So, how do I query the database?
It's not a database. It's a key-value store!
Ok, it's not a database. How do I query it?
You write a distributed map reduce function in Erlang!
Did you just tell me to go fuck myself?
I believe I did, Bob.

Map / Reduce ?

- Map (M) : collect a list of value associated to a key
- Return a list of key/values (K/V)
- Reduce (R) : reduce a list of key/value to a list of values.
- Rereduce : Reduce intermediary step of the reduce function.



Map

Extract results from a view

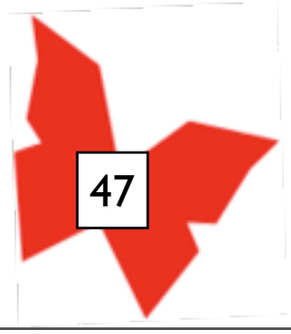
view name

```
$ curl -XGET localhost:5984/chest/_design/main/_view/by_type?key=%22training%22
```

Document Design Id

_view resource used
to query views

query parameter



Map

map function

```
1 function(doc) {
2     emit(doc.type, doc);
3 }
```

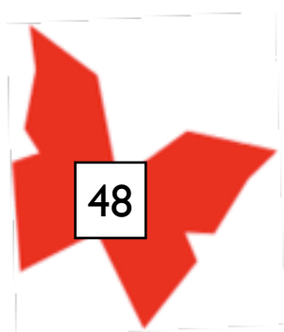
Create a document design

views object: all views functions are saved here

```
$ curl -XPUT -d '{
    "views": {
        "by_type": {
            "map": "function(doc) { emit(doc.type, doc); }",
        }
    }
}' localhost:5984/chest/_design/main

{"ok":true,"id":"_design/main","rev":"1-7da64646ac19f8d9d5da9d14a689f3ec"}
```

view name

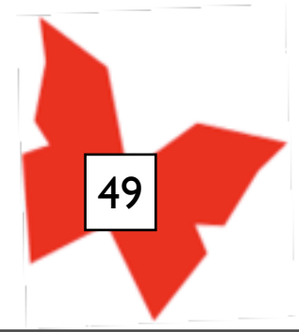
map function



Map

Results


```
{
    "total_rows":4,
    "offset":2,
    "rows":[
        {"id":"t20110725-1","key":"training","value":
{"_id":"t20110725-1","_rev":"1-77724440b82c743ebae8d44482a184f1","title":"tr
aining CouchDB ES","tags":["training","couchdb"],"description":"some
description","client":"clientXXXX","type":"training","date":"2011-07-25T00:0
0:00Z+01:00"}},
        {"id":"t20110912-1","key":"training","value":
{"_id":"t20110912-1","_rev":"1-77724440b82c743ebae8d44482a184f1","title":"tr
aining CouchDB ES","tags":["training","couchdb"],"description":"some
description","client":"clientXXXX","type":"training","date":"2011-07-25T00:0
0:00Z+01:00"}}

    ]
}
```

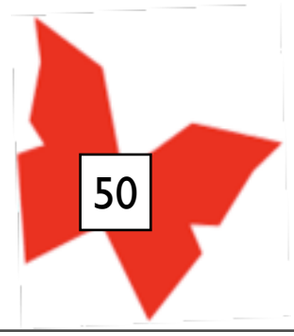
reduce

- CouchDB use heuristics methods to detect reduce fucntions that won't perform.
- 3 arguments: keys values, rereduce
- reduce functions need to manage 2 cases:
- rereduce = false

```
reduce([ [key1,id1], [key2,id2], [key3,id3] ],
       [value1,value2,value3], false)
```

- rereduce = true

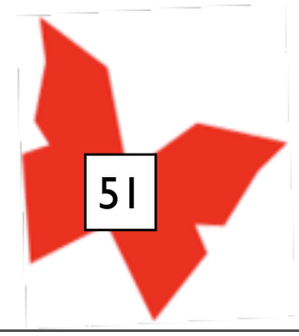
```
reduce(null, [intermediate1,intermediate2,intermediat
e3], true)
```



Map / Reduce ?

- emit: [{key1, 1}, {Key2, 2}] ,
  reduce: 1+ 2
  resultat: 3

- emit: [{key3, 1}],
  reduce: 1
  rereduce: 3 + 1
  resultat : 4



Reduce

map function

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

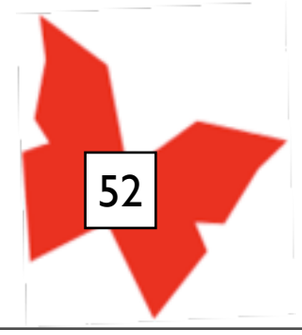
Update the document desihn

views object

```
curl -XPUT -d '{
    "_id": "_design/main",
    "_rev": "1-7da64646ac19f8d9d5da9d14a689f3ec",
    "views": {
        "by_type": {
            "map": "function(doc) { emit(doc.type, doc); }",
            "reduce": "function(keys, values, rereduce) { if (rereduce)
{return sum(values); } else { return values.length } }"
        }
    }
```

view name

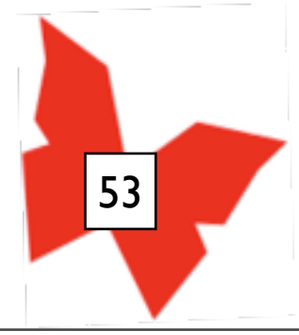
reduce function



Reduce: native functions

- _sum: count values
- _count: count distinct values
- _stats:

```
{
    "sum": 100,
    "count": 10,
    "min": 0,
    "max": 100,
    "sumsqr": 10000
}
```



fetch

```
Options = [],
DesignNam = "designname",
ViewName = "viewname",
{ok, ViewResults} = couchbeam_view:fetch(Db, {DesignNam, ViewName}, Options).
```

stream results

```
ViewFun = fun(Ref, F) ->
    receive
        {row, Ref, done} ->
            io:format("done", []),
            done;
        {row, Ref, Row} ->
            io:format("got ~p~n", [Row]),
            F(Ref, F);
        {error, Ref, Error} ->
            io:format("error: ~p~n", [Error])
    end
end,

{ok, StartRef3, _ViewPid3} = couchbeam_view:stream(Db, {DesignNam, ViewName}, self(), []),
ViewFun(StartRef3, ViewFun).
```

fetch

```
{ok, {TotalRowsCount, Offset, Results1}} = couchc:all(Db, {<<"test">>, <<"v1">>});
```

stream results

```
ViewName = {<<"test">>, <<"v1">>},

collect_results(Row, Acc) ->
    {ok, [Row | Acc]}.

all(Db, ViewName, Options) ->
    couchc:fold(Db, ViewName, fun collect_results/2, Options).
```

_changes

```
$ curl -XGET localhost:5984/chest/_changes
```

_changes handler.

sequence

document id

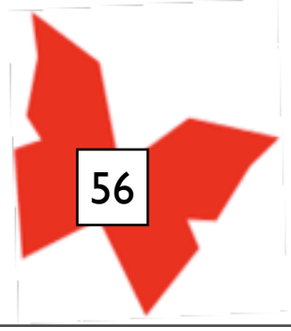
changes list

when a doc is deleted


```
{"results":[
    {"seq":4,"id":"i20110602","changes":[{"rev":"4-470476e8bd2cee80e02a5e4826d2ee8a"}],"deleted":true},
    {"seq":7,"id":"20110602-1","changes":[{"rev":"3-4ecc45338fe5c4a2ac8ca1893a5ef708"}],"deleted":true},
    {"seq":8,"id":"20110725-1","changes":[{"rev":"1-002c67027dc67c1ec62fcd2c7f5a7582"}]},
    {"seq":9,"id":"t20110725-1","changes":[{"rev":"1-77724440b82c743ebae8d44482a184f1"}]},
    {"seq":10,"id":"t20110912-1","changes":[{"rev":"1-77724440b82c743ebae8d44482a184f1"}]},
    {"seq":14,"id":"note-20110725-0000","changes":[{"rev":"4-c6c21d9c217e807d73135b681c4297fc"}]},

    ...

    {"seq":29,"id":"201ccc0bc2ceee00dae47299c10030e2","changes":[{"rev":"1-76306d88cb8594f65421b9cbad9a610d"}]}
],
"last_seq":29}
```


change row

Last sequence of the feed

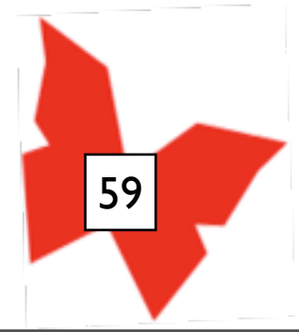


- changes can be filtered
- couchbeam provides a gen_changes behaviour and a pub/sub system
- couchc just use couch_changes module

```erlang
-module(test_gen_changes).

-behaviour(gen_changes).
-export([start_link/2]

-export([init/1, handle_change/2, handle_call/3, handle_cast/2,
         handle_info/2, terminate/2]).

-export([get_changes/1]).
-record(state, {changes=[]}).

start_link(Db, Opts) ->
    gen_changes:start_link(?MODULE, Db, Opts, []).

init([]) ->
    {ok, #state{}}.

get_changes(Pid) ->
    gen_changes:call(Pid, get_changes).

handle_change({done, _LastSeq}, State) ->
    {noreply, State};


handle_change(Change, State=#state{changes=Changes}) ->
    NewChanges = [Change|Changes],
    {noreply, State#state{changes=NewChanges}}.

handle_call(get_changes, _From, State=#state{changes=Changes}) ->
    {reply, lists:reverse(Changes), State}.

....
```

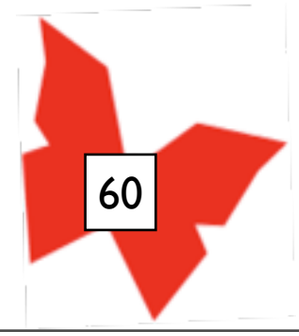
doc & views transformations: couchapp

- shows: functions that transform a doc
- lists: functions that transform view results
- updates: functions that can updated a document and return a results



couchapp

- make embedded and replicable webapps in couch
- or just transform your data
- simple functions in javascript (or your language)
- cached



list - view transformation

```
function(head, req) {
    provides("html", function() {
        send("HTML <ul>");

        var row, num = 0;
        while (row = getRow()) {
          num ++;
          send('\n<li>Key: '
            +row.key+' Value: '+row.value
            +' LineNo: '+num+'</li>');
        }

        // tail
        return '</ul>';
    });
}
```

send a chunk

iterrate view results

stop to send

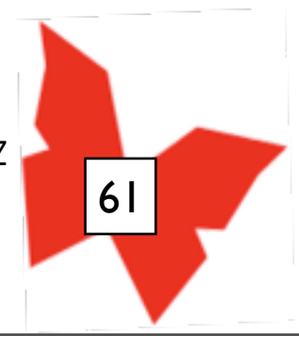
Result :

list handler

function name

view ame

```
$ curl -XGET localhost:5984/coffre/_design/example/_list/test/type
HTML <ul>
...
<li>key:note: Value: {"_id":"note-20110725-0000","_rev":"4-
c6c21d9c217e807d73135b681c4297fc","title":"hello word","tags":
["note","divers"],"description":"note divers","type":"note","date":"2011-07-25T00:00:00Z
+01:00"}LineNo:0</li>
...
</ul>
```

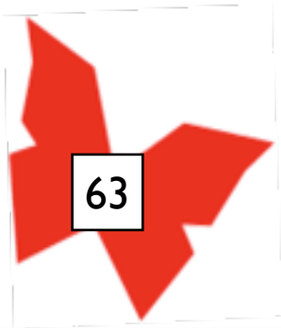


erica

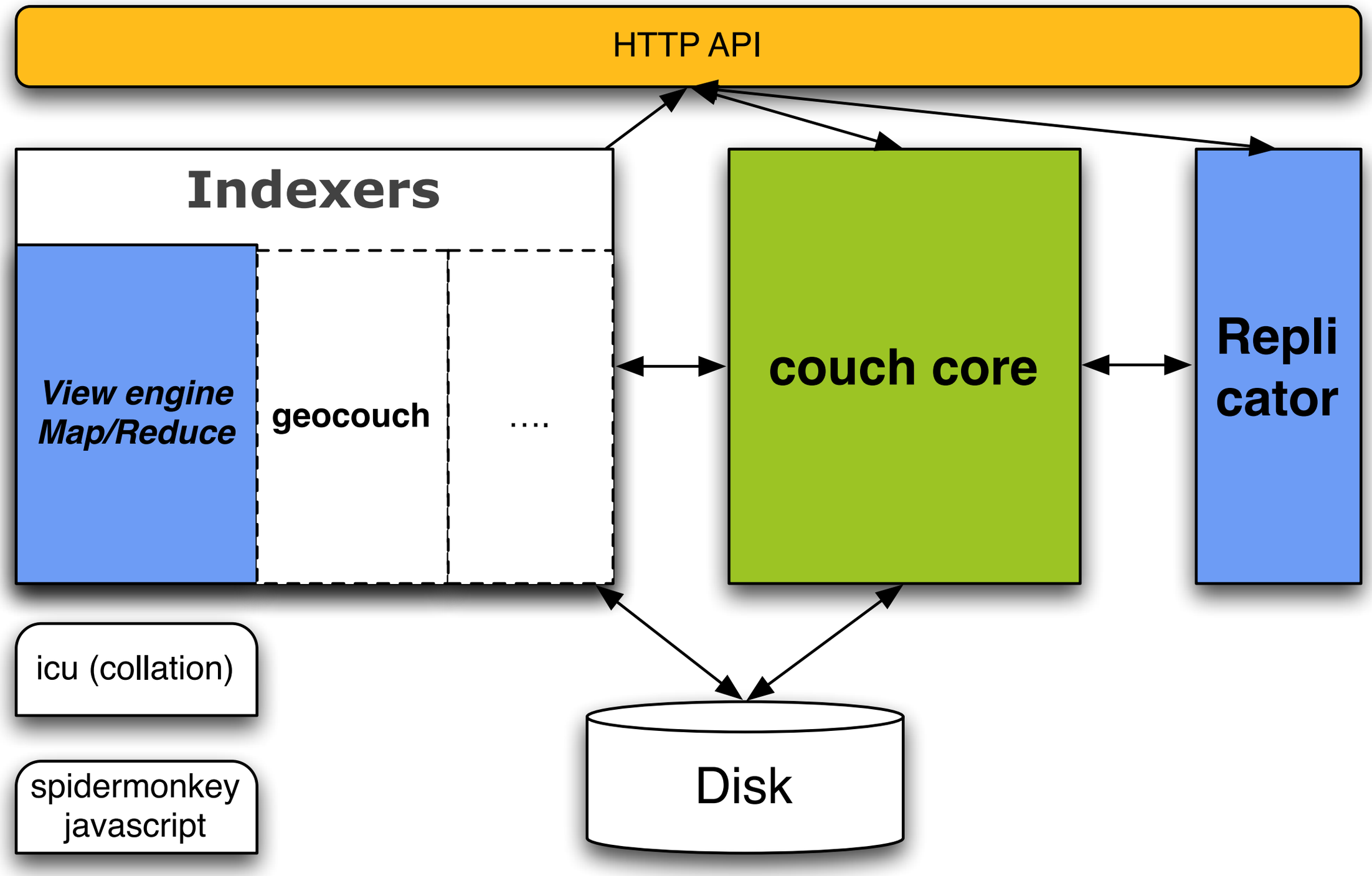
- create and edit your design documents
- couchapp but in erlang. (https://github.com/couchapp/couchapp)
- Sources dispos sur:
  https://github.com/benoitc/erica
- create design doc (couchapps) templates

security

- useCtx object (name & roles)
- database security object (/db/_security) : db admins and readers
- Different HTTPP authentication(basic, cookie auth, oauth, ...)
- per database
- validate functions : validation on write



Extend CouchDB


HTTP API
Indexers
View engine
Map/Reduce
geocouch
....
couch core
Repli
cator
icu (collation)
spidermonkey
javascript
Disk

the couch distribution

- autotools
- src/... -> main applications and dependancies
- some nifs for versions > R13B04 (ejson, less_json)
- ports: couchjs, couch_icu_drv

the couch distribution

- couch_config: manage settings in ini file
- couch_httpd* : http api
- couch_rep* : replicator
- the couch_core
- couch_index, couch_mrview

dependancies

- mochiweb
- erlang_oauth
- ibrowse
- ejson

add your own daemon

```
[daemons]
view_manager={couch_view, start_link, []}
external_manager={couch_external_manager, start_link, []}
.....
```

- gen_server
- supervised by couch_secondary_sup

add your own daemon

```
[os_daemons]
; For any commands listed here, CouchDB will attempt to ensure
that
; the process remains alive while CouchDB runs as well as shut
them
; down when CouchDB exits.
;foo = /path/to/command -with args
```

- erl ports
- 1 system process
- can be in any languages

add your http handler

```
[httpd_db_handlers]
..
_hello = {helloapp_httpd, handle_hello_req}
..
```

- erlang function
- globlal, db or design handlers

add your http handler

```
handle_hello_req(#httpd{method='GET'}=Req, _Db) ->
    couch_httpd:send_json(Req, {[
        {msg, "hello world"}
    ]});
handle_hello_req(Req, _) ->
    couch_httpd:send_method_not_allowed(Req, "GET,HEAD").
```

Or just proxy

```
[httpd_global_handlers]
_google = {couch_httpd_proxy, handle_proxy_req,
<<"http://www.google.com">>}
```

- ERL_FLAGS environmen
- couch-config (in 1.2)

```
$ couch-config --erl-libs-dir
/Users/benoitc/misc/couchdb/lib/couchdb/erlang/lib
```

example of Makefile : rebar & couch-config

```
COUCHVER=$(shell couch-config --couch-version)
COUCH_PATH=$(ERLIBS)/couch-$(COUCHVER)
CONFDIR=$(shell couch-config --config-dir)
...

compile:
    @ERL_COMPILER_OPTIONS='[{i, "$(COUCH_PATH)/include"}]' ./rebar compile

...

install: all
    @mkdir -p $(ERLIBS)/couch_es
    @cp -r ebin $(ERLIBS)/couch_es
    @cp couch_es.ini $(CONFDIR)
    @echo "\n\
couch_es has been installed in $(ERLIBS)/couch_es\n\
To launch it run the commandline:\n\
\n\
    $ ERL_CFLAGS=\"-pa $(ERLIBS)/couch_es\" couchdb -a $(CONFDIR)/couch_es.ini\n"
```

some examples

- couch_es: http://github.com/refuge/couch_es

- couchapp_ng : http://github.com/benoitc/couchapp_ng

Add your own indexer

- geocouch way: an retree with functions similar to the view query engine (old way)
- couch_es, couchdb-lucene: use an external indexer
- new way: couch_index

it could be easier

the refuge project

- fully decentralized and opensource platform to share, collect and render data
  - refuge the P2P platform
  - upondata.com: curation platform based on couchdb (launch on 2011/09/27), the hub
  - http://github.com/refuge

- couch_core: rebared distribution of couchdb
- rcouch templates: build your own couchdb distribution
- farmer: mdns discovery of couchdb nodes
- ported couch to build on any arm platform

- couch_core: rebared distribution of couchdb
- rcouch templates: build your own couchdb distribution
- farmer: mdns discovery of couchdb nodes
- ported couch to build on any arm platform
- couch_mergeview

- slowly replacing bits that make couchdb hard to distribute: eg. only one binding for icu

- static build
- relocatable release


Vittel

- http://github.com/refuge
- http://github.com/benoitc
- http://refuge.io & http://lists.refuge.io
- @benoitc

Thanks!

?