CouchDB

Presented By

Team 9:

Donita Almeida

Dhiraj Gurnani

George Zachariah

Ritika Shetty

Saina Patel

Introduction

- Why CouchDB ?
  - High availability is important
  - Eventually consistent
  - Powerful data processing using the built-in query engine
  - Modular & Scalable design
  - User friendly

Features

- A NoSQL database
- Uses JSON for documents
- Uses JavaScript for MapReduce indexes
- Uses HTTP for it’s API
- Documents can have attachments
- Incremental peer-peer replication

SQL vs CouchDB

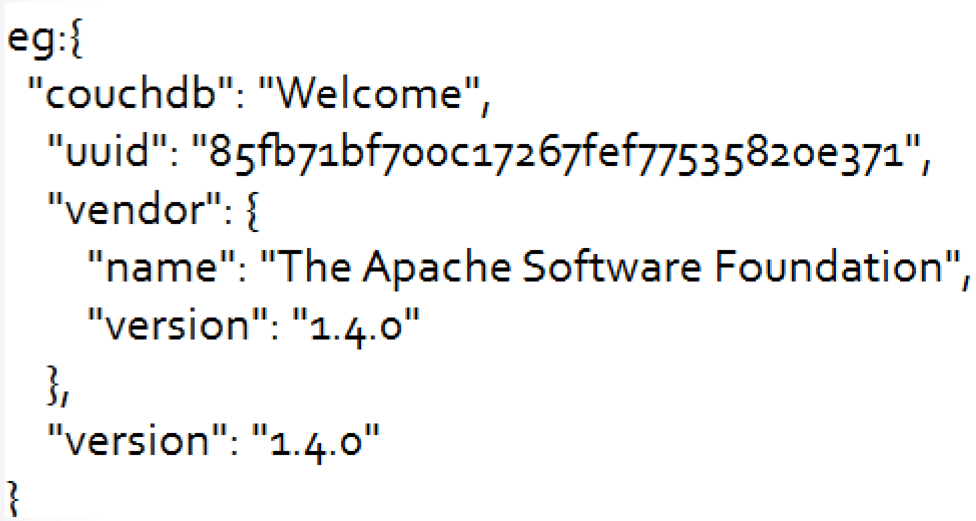
SQL	CouchDB
Relational	Non-Relational
Tables	Documents with types
Rows and Columns	Document Fields
SQL Query Engine	Map / Reduce Engine

CouchDB Core API (Command Line Utility)

- Server API
- Database API
- Document API
- Replication API

Server API (Access via CURL)

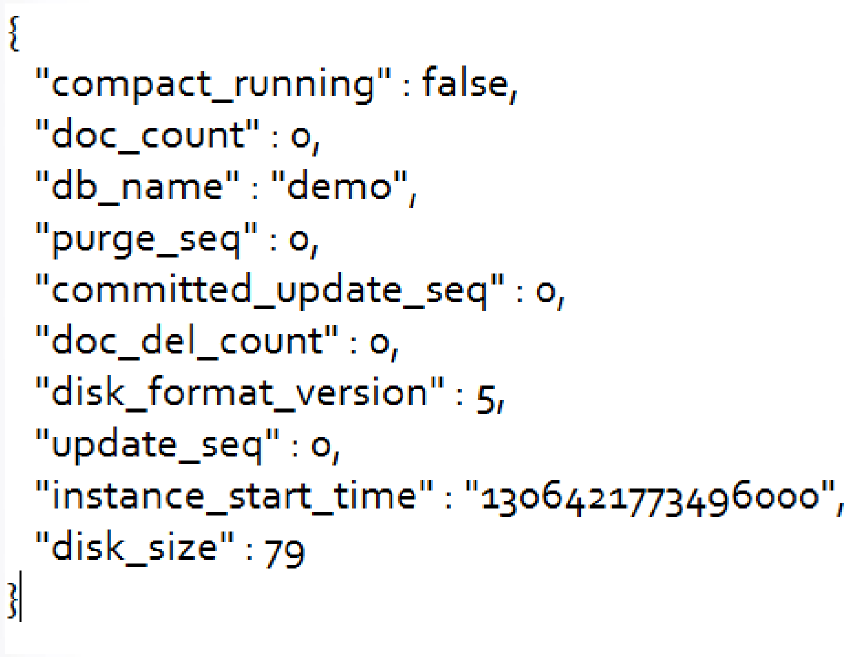
- CouchDB runs on port 5984
- curl -X GET http://127.0.0.1:5984/
- returns the server information



```
eg:{
 "couchdb": "Welcome",
  "uuid": "85fb71bf700c17267fef77535820e371",
  "vendor": {
     "name": "The Apache Software Foundation",
     "version": "1.4.0"
  },
  "version": "1.4.0"
}
```

Database API

- curl -X **GET** http://127.0.0.1:5984/demo
- Returns the information about database demo



```
{
  "compact_running" : false,
  "doc_count" : 0,
  "db_name" : "demo",
  "purge_seq" : 0,
  "committed_update_seq" : 0,
  "doc_del_count" : 0,
  "disk_format_version" : 5,
  "update_seq" : 0,
  "instance_start_time" : "1306421773496000",
  "disk_size" : 79
}
```

Database API (Contd..)

Create a database

- curl -X **PUT** http://127.0.0.1:5984/baseball

Delete a database

- curl -X **DELETE** http://127.0.0.1:5984/baseball

Document API via CURL

- curl -X **PUT** http://127.0.0.1:5984/albums

Create a document

- curl -X **PUT** http://127.0.0.1:5984/albums/1000
  -d '{"title":"Abbey Road","artist":"The Beatles"} `

Retrieve a document

- curl -X **GET** http://127.0.0.1:5984/albums/1000

Document API (Contd..)

- **_rev** - If you want to update or delete a document, CouchDB expects you to include the _rev field of the revision you wish to change

- curl -X **PUT** http://127.0.0.1:5984/albums/1000 -d '{"_rev":"1-42c7396a84eaf1728cdbf08415a09a41","title":"Abbey Road", "artist":"The Beatles","year":"1969"}'

Availability and Partition Tolerance

- Every Node in the network is a Master
- CouchDB is highly available and eventually consistent
- For making the System Highly available, couchdb has used two concepts:

  1. Key to your Data
  2. Multi – version Concurrency Control

Key To Your Data

- B-tree storage engine

- This storage engine allows the data to be stored in sorted order based on the keys assigned to each document.

- Allows to perform search, insertion and deletion in logarithmic time.

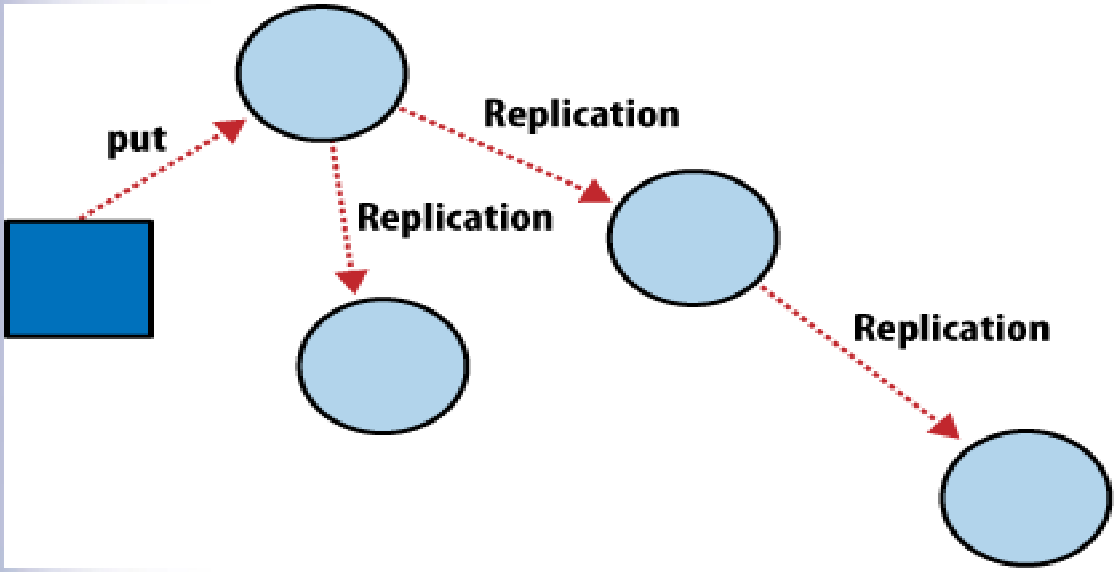
Multi Version Concurrency Control

- In relational database, locks are used to ensure that a row is updated by only 1 user at a time. But this approach waste a lot of time.
- To avoid it CouchDB uses MVCC, where different versions of same document are created.
- i.e. update on documents create a new verson at some other place and old version is also present.

Eventual Consistency

- It uses incremental replication, where document changes is periodically communicated between servers.

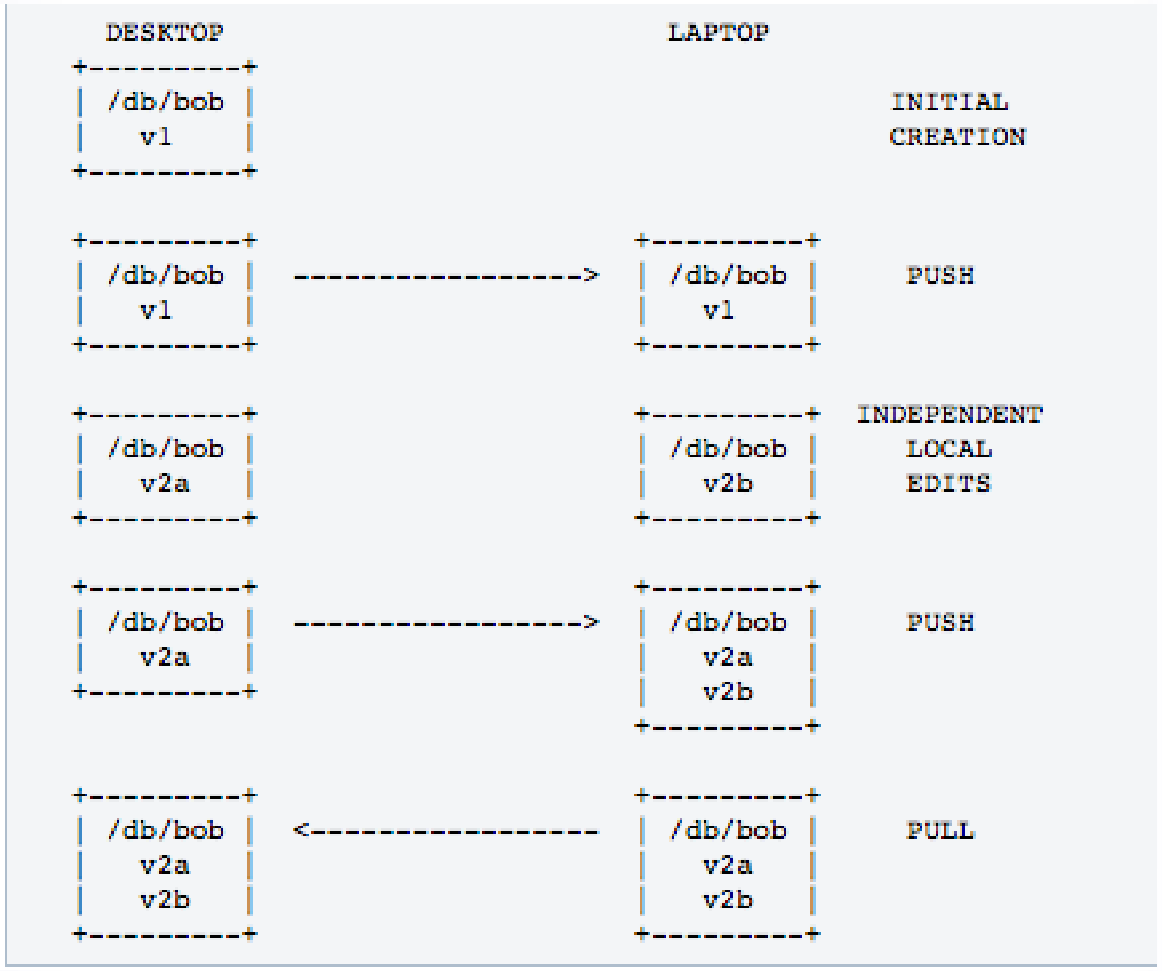
- With this approach, the servers need not be in constant connection.



Replication API via curl

- Command to replicate a Database :
- curl -X POST http://127.0.0.1:5984/_replicate -d '{"source":"http://example.org/database", "target":"http://admin:password@127.0.0.1:5984/database"}' -H "Content-Type: application/json"
- This is unidirectional command. To make the replication bydirectional, we call the same command by swapping source and target values.

Replication



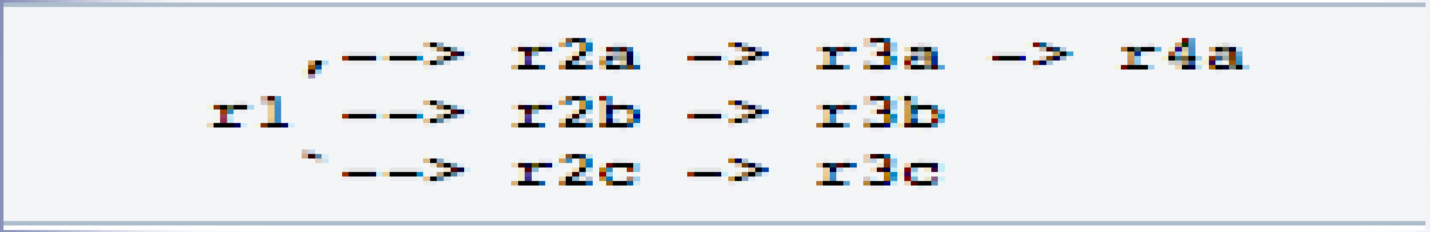
```
      DESKTOP                           LAPTOP
    +----------+
    | /db/bob  |                                       INITIAL
    |   v1     |                                       CREATION
    +----------+

    +----------+                       +----------+
    | /db/bob  |  --------------->     | /db/bob  |     PUSH
    |   v1     |                       |   v1     |
    +----------+                       +----------+

    +----------+                       +----------+  INDEPENDENT
    | /db/bob  |                       | /db/bob  |     LOCAL
    |   v2a    |                       |   v2b    |     EDITS
    +----------+                       +----------+

    +----------+                       +----------+
    | /db/bob  |  --------------->     | /db/bob  |     PUSH
    |   v2a    |                       |   v2a    |
    +----------+                       |   v2b    |
                                       +----------+

    +----------+                       +----------+
    | /db/bob  |  <---------------     | /db/bob  |     PULL
    |   v2a    |                       |   v2a    |
    |   v2b    |                       |   v2b    |
    +----------+                       +----------+
```

Continuous Replication

- curl -X POST http://127.0.0.1:5984/_replicate -d '{"source":"db", "target":"db-replica", **"continuous":true**}' -H "Content-Type: application/json"'

- It will keep listening to '_changes' API and replicate any missing documents to the target.

Revision Tree



```
     ,--> r2a -> r3a -> r4a
r1   --> r2b -> r3b
     `--> r2c -> r3c
```

- To access a particular revision of a document
  - GET /somedatabase/some_doc_id?rev=946B7D1C HTTP/1.

- To get information about which revisions are present
  - GET /somedatabase/some_doc_id?revs_info=true HTTP/1.0

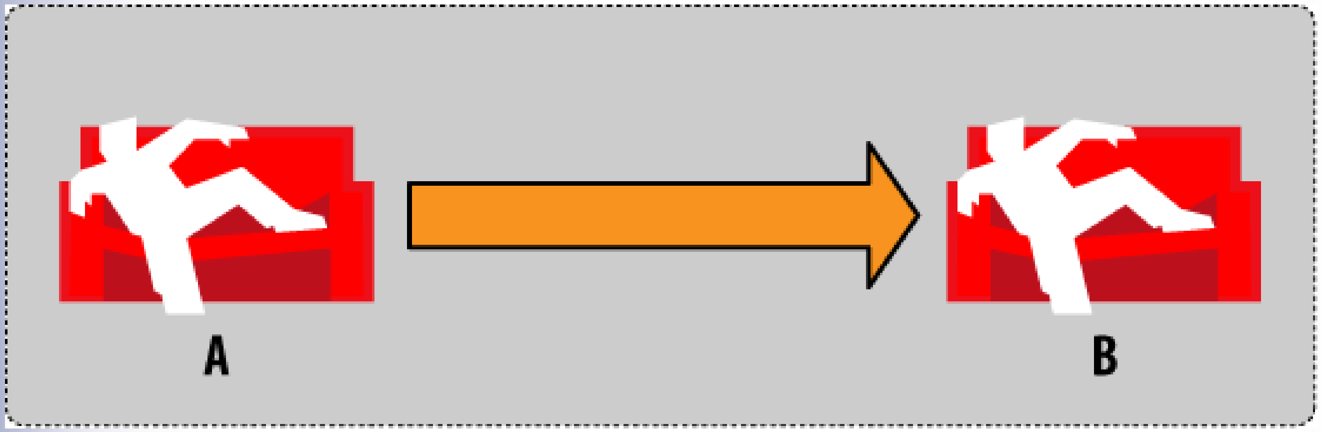
CouchDB Conflict Resolution

- How does CouchDB's replication system deal with conflicts?

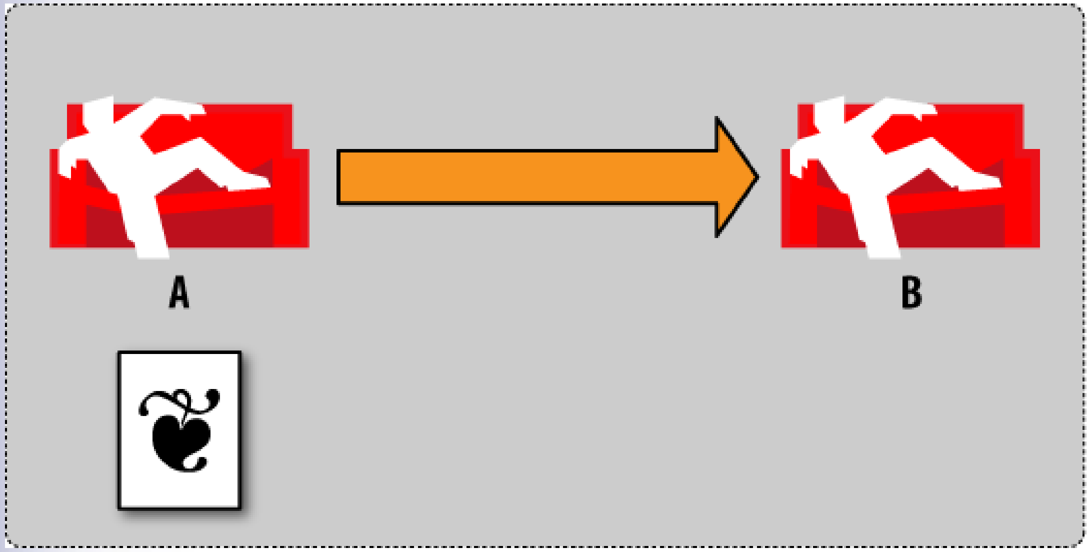
1. Flags the affected document with a special attribute "_conflicts": true
2. Determines which of the changes will be stored as the latest revision (winning revision).
3. The loosing revision gets stored as the previous revision

CouchDB Conflict Resolution

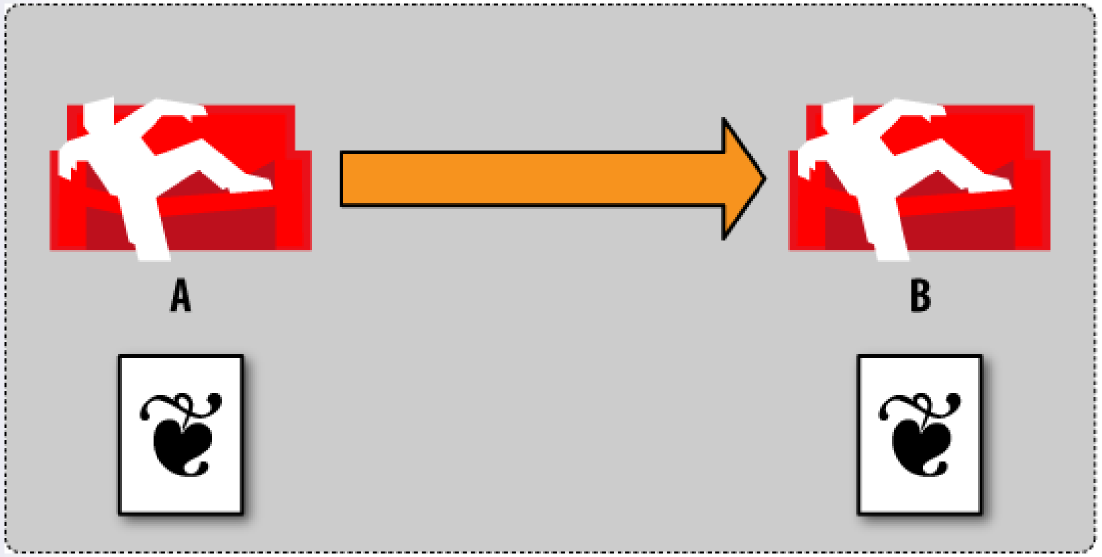
1. We have two CouchDB databases A and B, and we are replicating from A to B.



2. We create a document in database A. Database B won’t know about the new document for now.

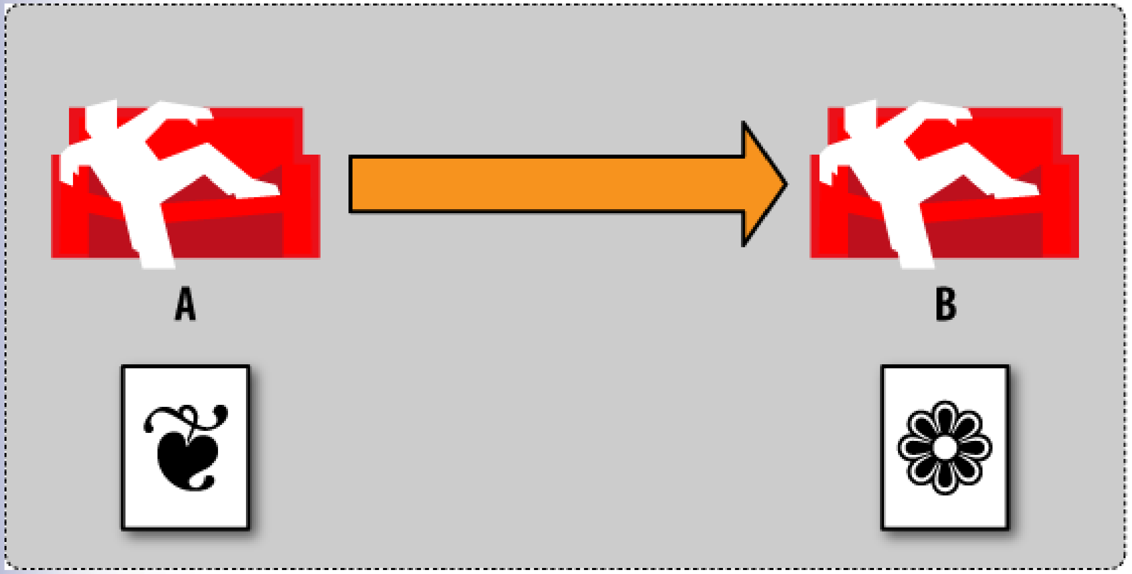


3. We now trigger replication and tell it to use database A as the source and database B as the target



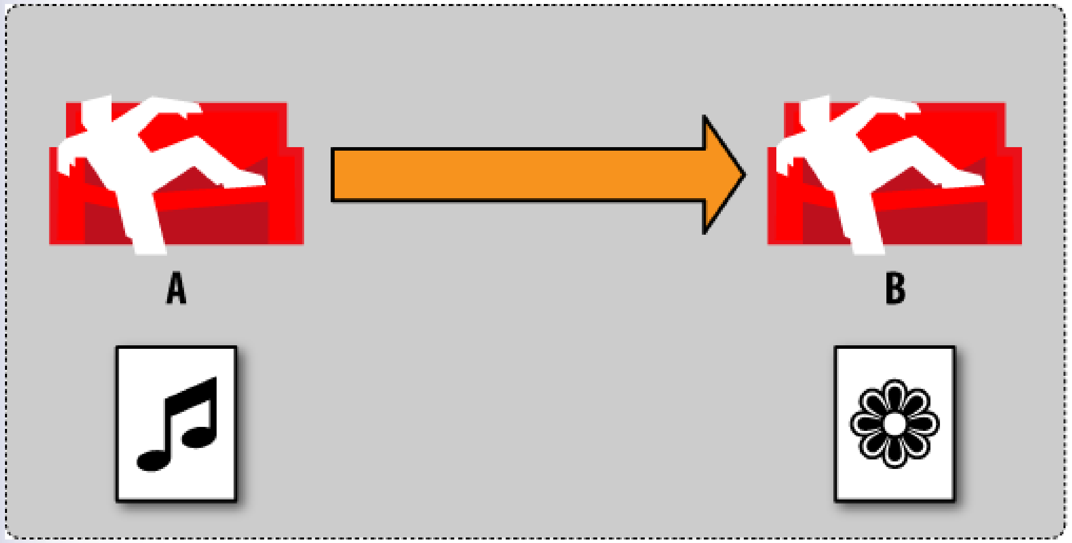
4. Now we go to database B and update the document.

Upon change, CouchDB generates a new revision for us.

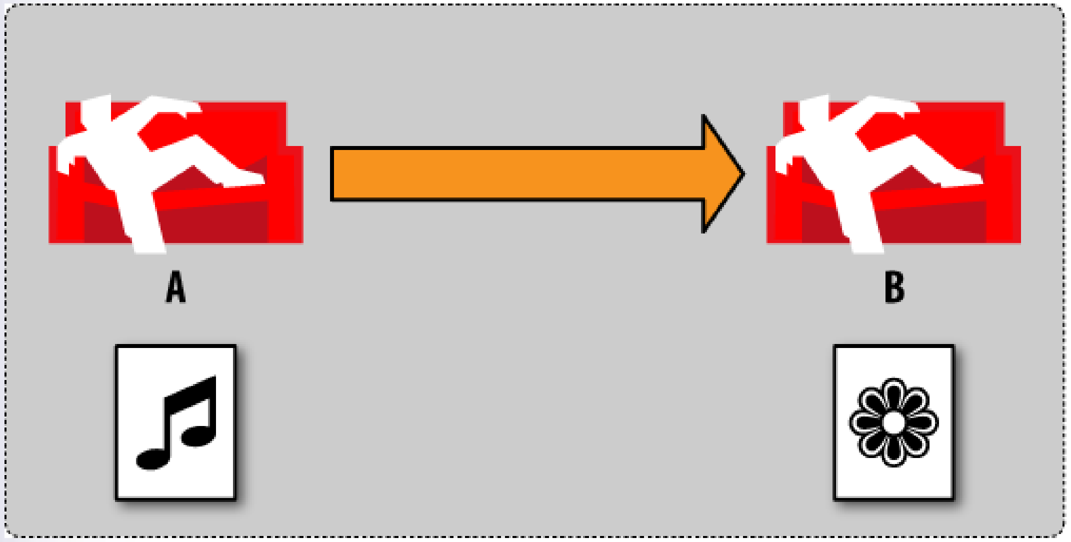


5. Now we make a change to our document in database A by changing some other values.

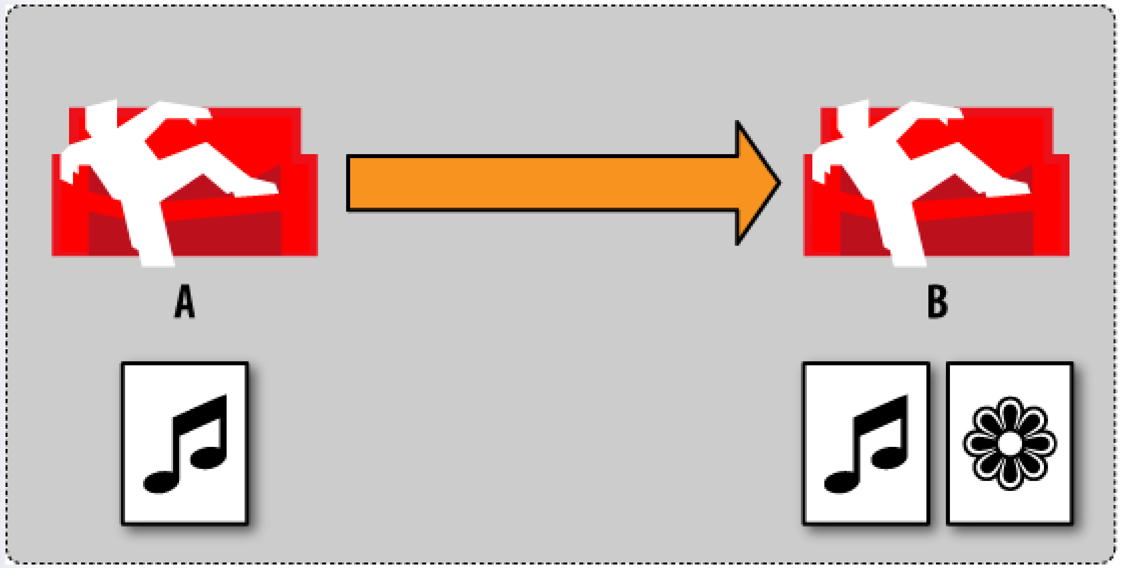
There are two different revisions of that same document in each database.



6. Now we trigger replication again from database A to database B as before

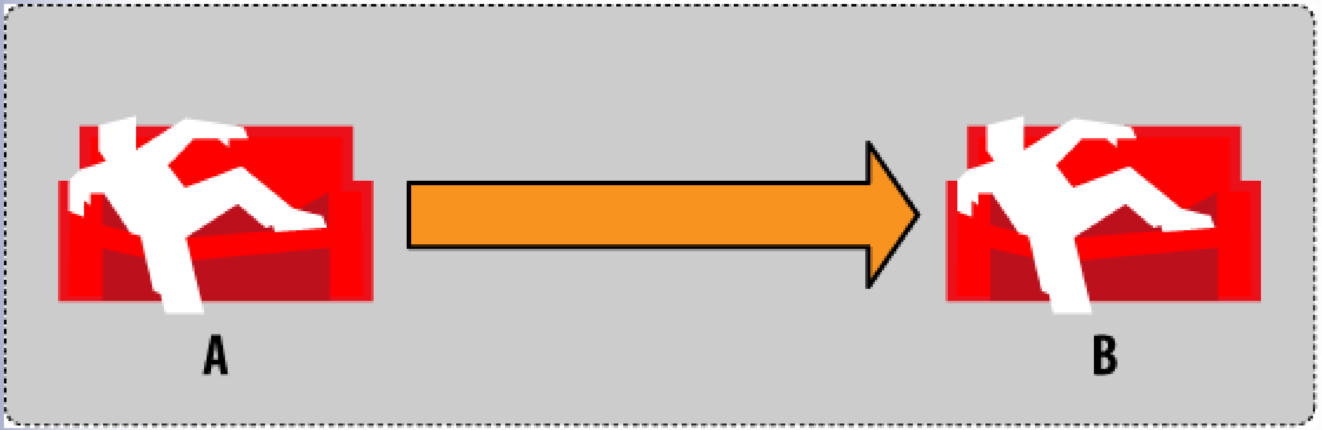


7. When replicating two different revisions are detected for the same document, and it creates a conflict .



8. Finally, we tell CouchDB which version we want as the latest revision by resolving the conflict.

Now both databases have the same data.



Couch DB Views

Why views?

No tables and collection.

So views

- **View server execute functions**

1. Map – used to display a view
2. Reduce(optional) – which is used to create a sorted view

Example Database

```
{
"_id": "album1",
"artist": "Megadeth",
 "title": "Endgame",
 "year": 2010
}
 {
 "_id": "album2",
 "artist": "Slayer",
 "title":"World Painted Blood",
 "year":2009
 }

 {
 "_id": "album3",
 "artist": "Arcturus",
 "title": "Sideshow Symphonies",
 "year": 2005
 }
```

```
{
 "_id": "album4",
 "artist": "Pantera",
 "title": "Reinventing the Steel",
 "year": 2009
}
{
 "_id": "album5",
 "artist": "Slayer",
 "title": "South of Heaven",
 "year": 2009
 }
```

Map and Reduce functions

```
"_id": "_design/foobar",
 "language": "javascript",
 "views": {
                    "by_year": {
                                        "map": "function(doc) {
                                                                                if (doc.year) {
                                                                                                    emit(doc.year, 1);
                                                                                                                       }
                                                                                                        }",
                                       "reduce": "function(keys, values, rereduce) {
                                                                               return sum(values);
                                                                               }"
                                       }
                    }
 }
```

```
$ curl
http://localhost:5984/albums/_design/foobar/_view/by_year

{
    "update_seq": 6,
    "rows": [
        {"key": null, "value": 5}
    ]
}
```

$ curl http://localhost:5984/albums/_design/foobar/_view/by_year?group=true

```
{
      "update_seq": 6,
      "rows": [
            {"key": 2005, "value": 1},
            {"key": 2009, "value": 3},
            {"key": 2010, "value": 1}
      ]
}
```

```
$ curl 'http://localhost:5984/albums/_design/foobar/_view/by_year?
group=true&startkey=2009&endkey=2010'
{
      "update_seq": 6,
      "rows": [
            {"key": 2009, "value": 3},
            {"key": 2010, "value": 1}
      ]
}
```

```
$ curl \
http://localhost:5984/albums/_design/foobar/_view/by_year?reduce=false
```

```
{
        "update_seq":6,
        "rows": [
                {"id": "album3", "key": 2005, "value": 1},
                {"id": "album2", "key": 2009, "value": 1},
                {"id": "album4", "key": 2009, "value": 1},
                {"id": "album5", "key": 2009, "value": 1},
                {"id": "album1", "key": 2010, "value": 1}
                ]
}
```

Limitations

- Temporary views on large datasets are very slow.
- Replication of large databases may fail
- Documents are quite large as the data is represented using “JSON” format
- "Only" eventual consistency.
- Couch maintains a different document for every update you make this fills up your hard disk fast

Compaction

- DB/view files are written in append mode
- Will continue to grow indefinitely
- A DB or View compaction operation can be triggered
- curl -X POST http://127.0.0.1:5984/albums/compact
- curl -X POST http://127.0.0.1:5984/albums/_design/view

What Compaction does?

i. Creates a new file

ii. Traverses the DB or View B-Tree and lookups the most recent data pointed by each node

iii. Writes that most recent data to the new file

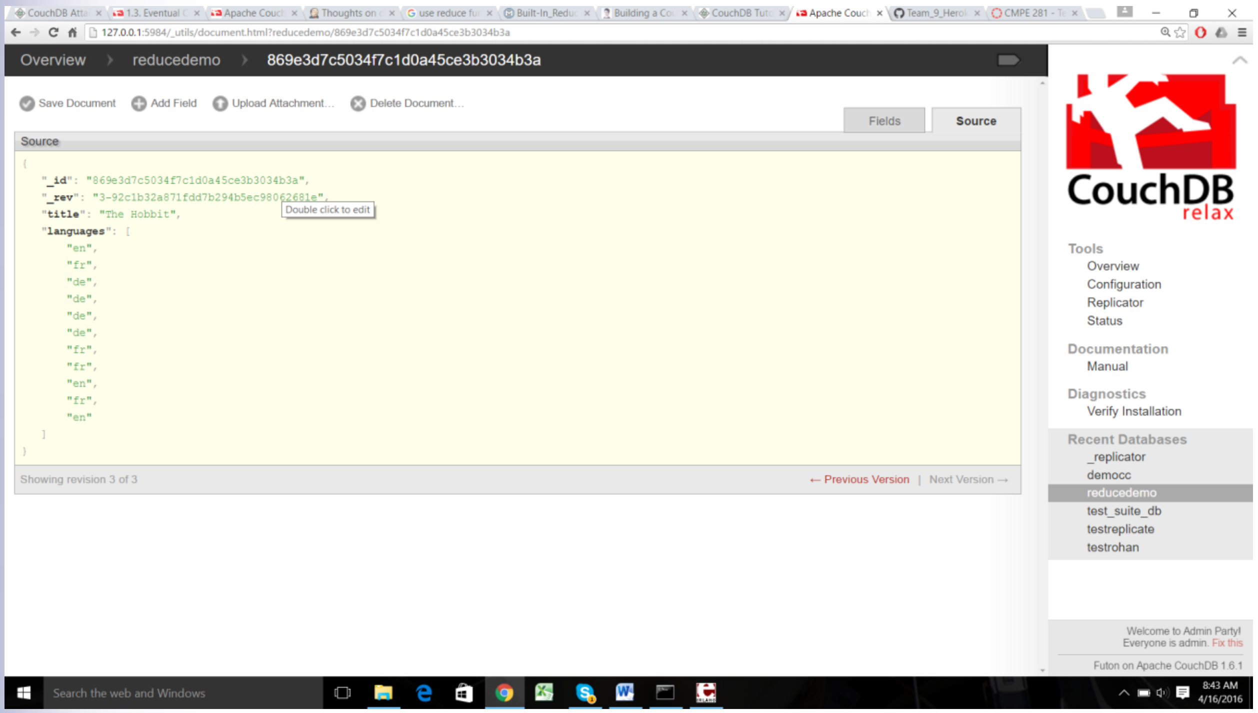
iv. deletes the original file and renames the compacted file to the original DB/View file name.

Who uses CouchDB?

BBC
mozilla FOUNDATION
CREDIT SUISSE
meebo
IBM
ubuntu
CERN European Organization for Nuclear Research
aptela
Engine Yard
memorize.com Quickly learn and remember the things that matter to you.
Magnifeast LOS ANGELES
Florida Tech UNIVERSITY ONLINE
mydochub beta
SOTEL
CLOUDANT
pcapr
ProdCall
OurParents

... and lots of lesser known businesses and early stage startups

Document Creation

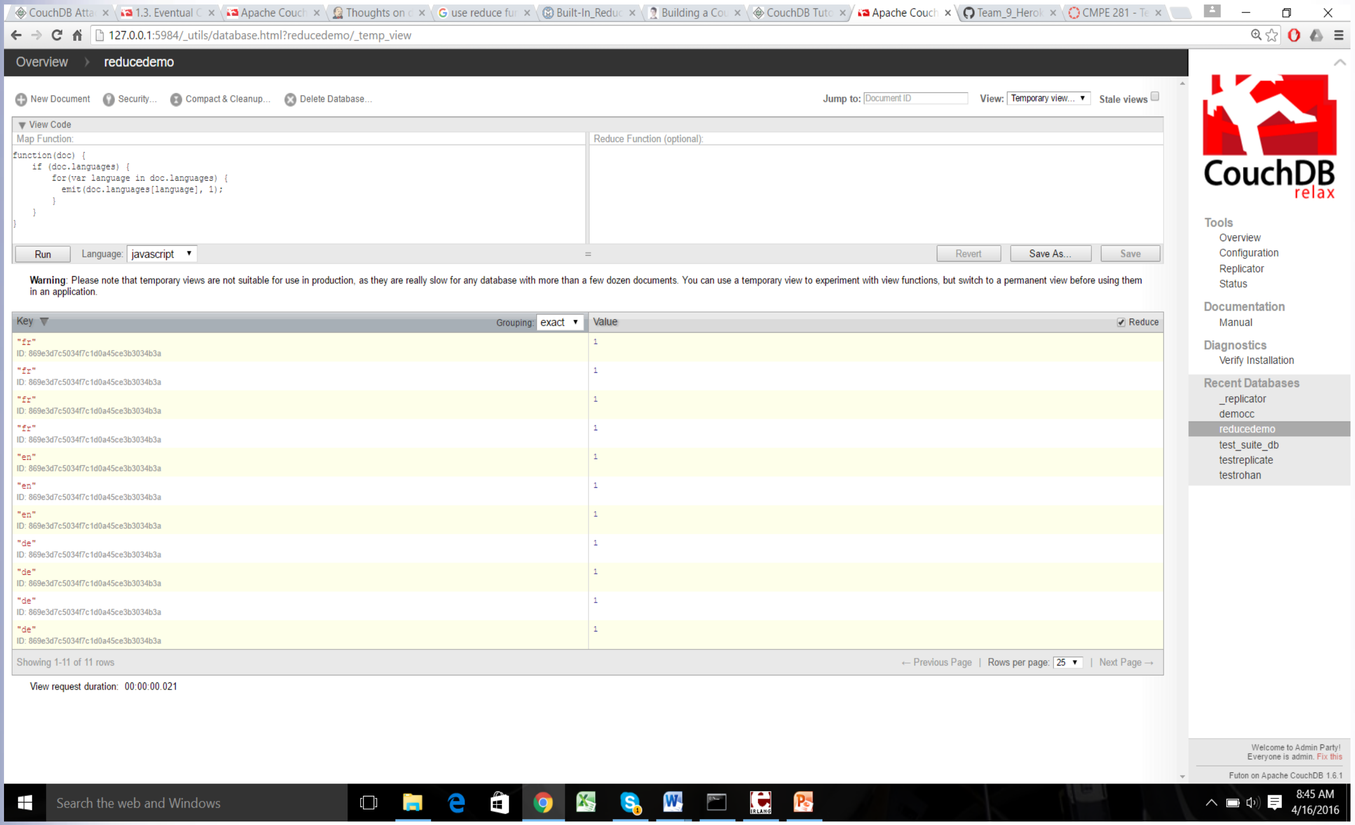


Source

```
{
   "_id": "869e3d7c5034f7c1d0a45ce3b3034b3a",
   "_rev": "3-92c1b32a871fdd7b294b5ec98062681e",
   "title": "The Hobbit",
   "languages": [
       "en",
       "fr",
       "de",
       "de",
       "de",
       "de",
       "fr",
       "fr",
       "en",
       "fr",
       "en"
   ]
}
```

Showing revision 3 of 3

Map View



127.0.0.1:5984/_utils/database.html?reducedemo/_temp_view

Overview › reducedemo

Map Function:

```
function(doc) {
    if (doc.languages) {
        for(var language in doc.languages) {
          emit(doc.languages[language], 1);
        }
    }
}
```

Language: javascript

Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

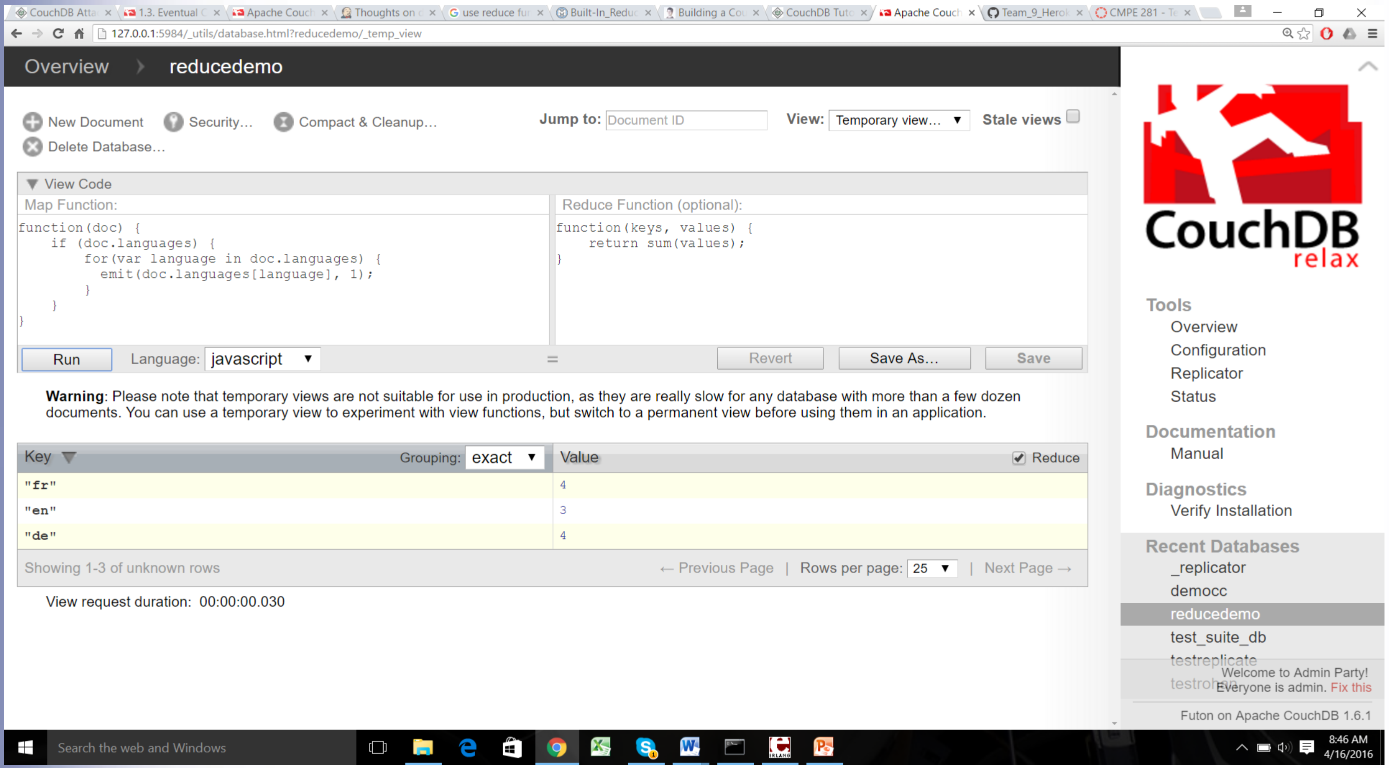
Grouping: exact ☐ Reduce

Key	Value
"fr" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"fr" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"fr" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"fr" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"en" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"en" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"en" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"de" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"de" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"de" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1
"de" ID: 869e3d7c5034f7c1d0a45ce3b3034b3a	1

Showing 1-11 of 11 rows

View request duration: 00:00:00.021

Map/reduce View



127.0.0.1:5984/_utils/database.html?reducedemo/_temp_view

Map Function:

```
function(doc) {
    if (doc.languages) {
        for(var language in doc.languages) {
          emit(doc.languages[language], 1);
        }
    }
}
```

Reduce Function (optional):

```
function(keys, values) {
    return sum(values);
}
```

Language: javascript

Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application.

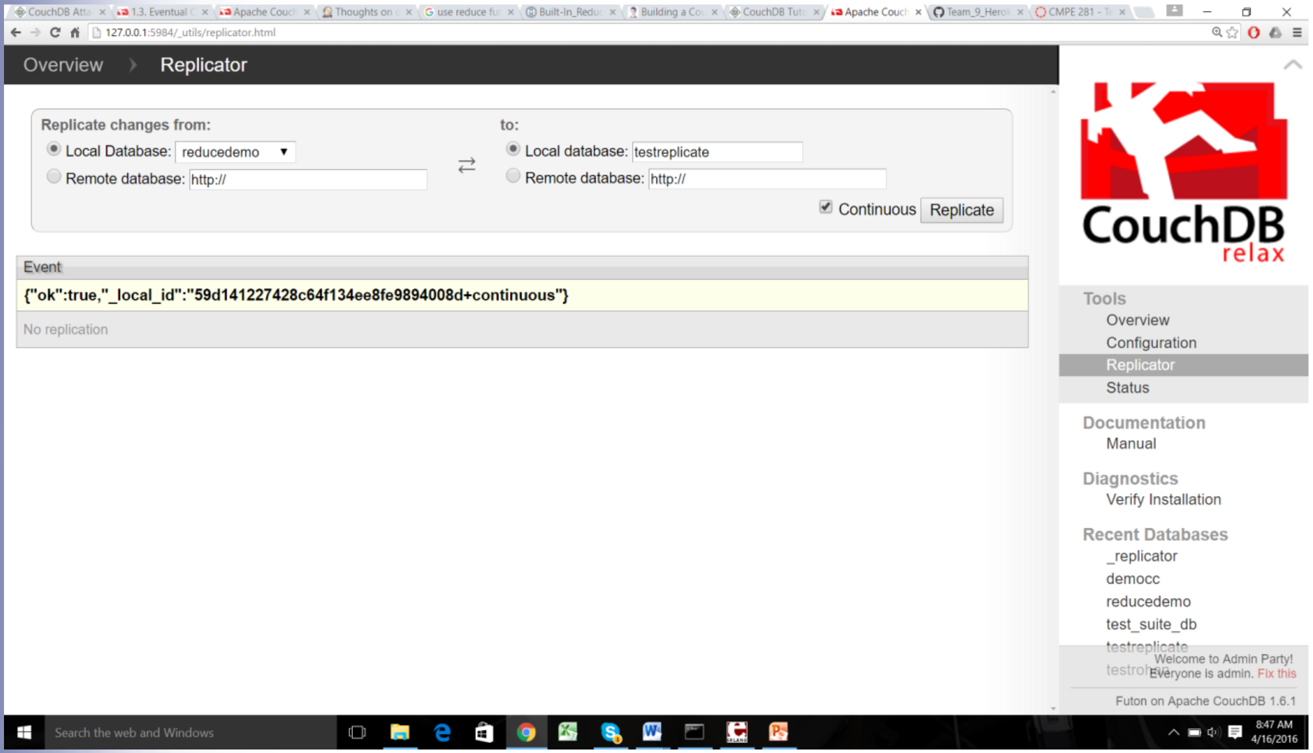
Grouping: exact — Reduce ✓

Key	Value
"fr"	4
"en"	3
"de"	4

Showing 1-3 of unknown rows   ← Previous Page | Rows per page: 25 | Next Page →

View request duration: 00:00:00.030

Replication



Replicate changes from:
- Local Database: reducedemo
- Remote database: http://

to:
- Local database: testreplicate
- Remote database: http://

[✓ Continuous] Replicate

Event

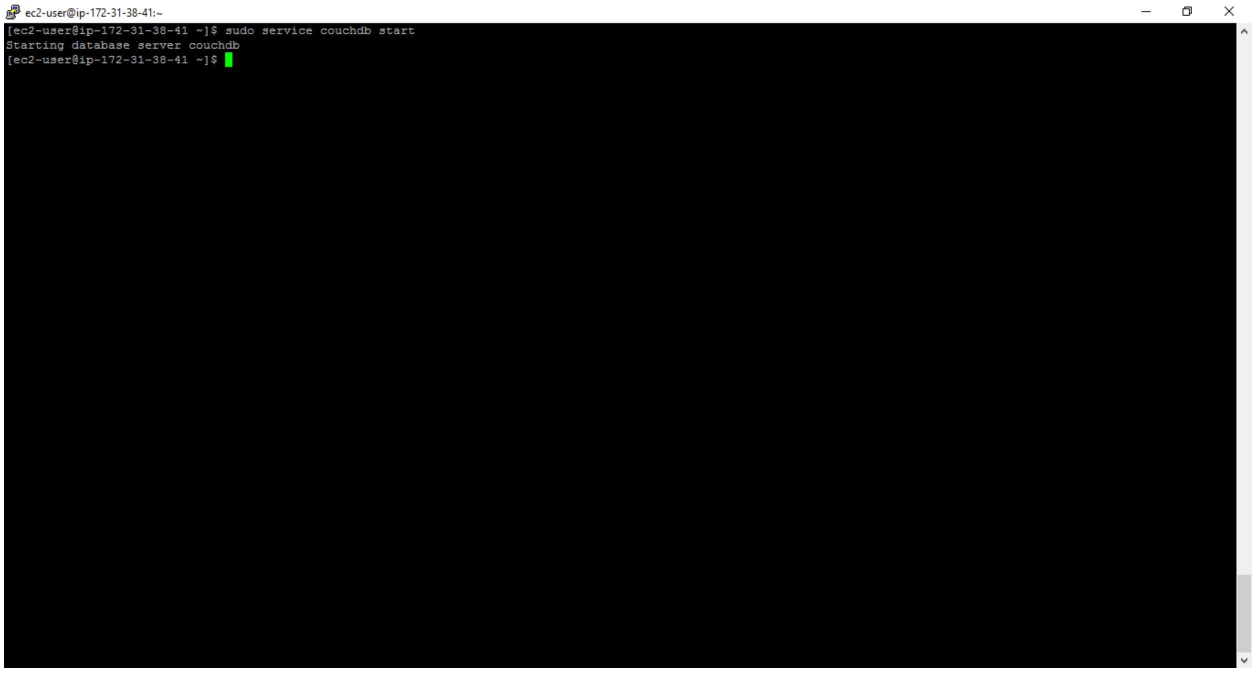
{"ok":true,"_local_id":"59d141227428c64f134ee8fe9894008d+continuous"}

No replication

<u>Demo for CouchDB on EC2</u>

Start couchDB:

sudo service couchdb start

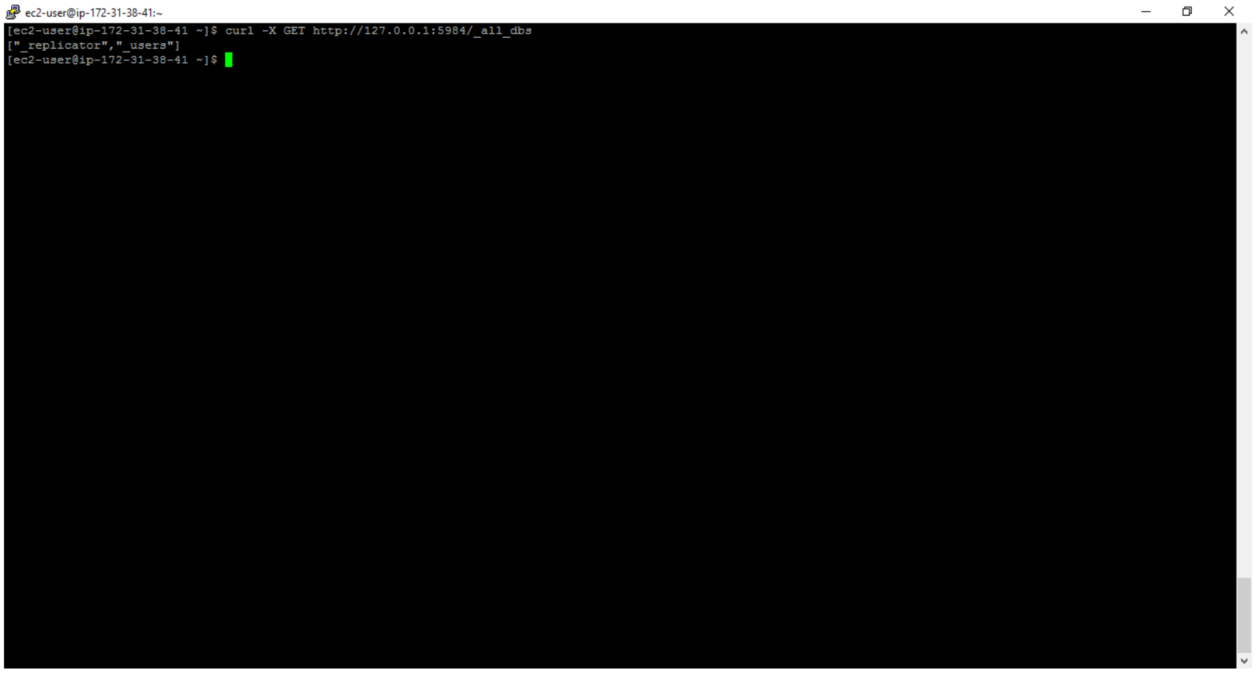


ec2-user@ip-172-31-38-41:~

```
[ec2-user@ip-172-31-38-41 ~]$ sudo service couchdb start
Starting database server couchdb
[ec2-user@ip-172-31-38-41 ~]$
```

View all DBs:

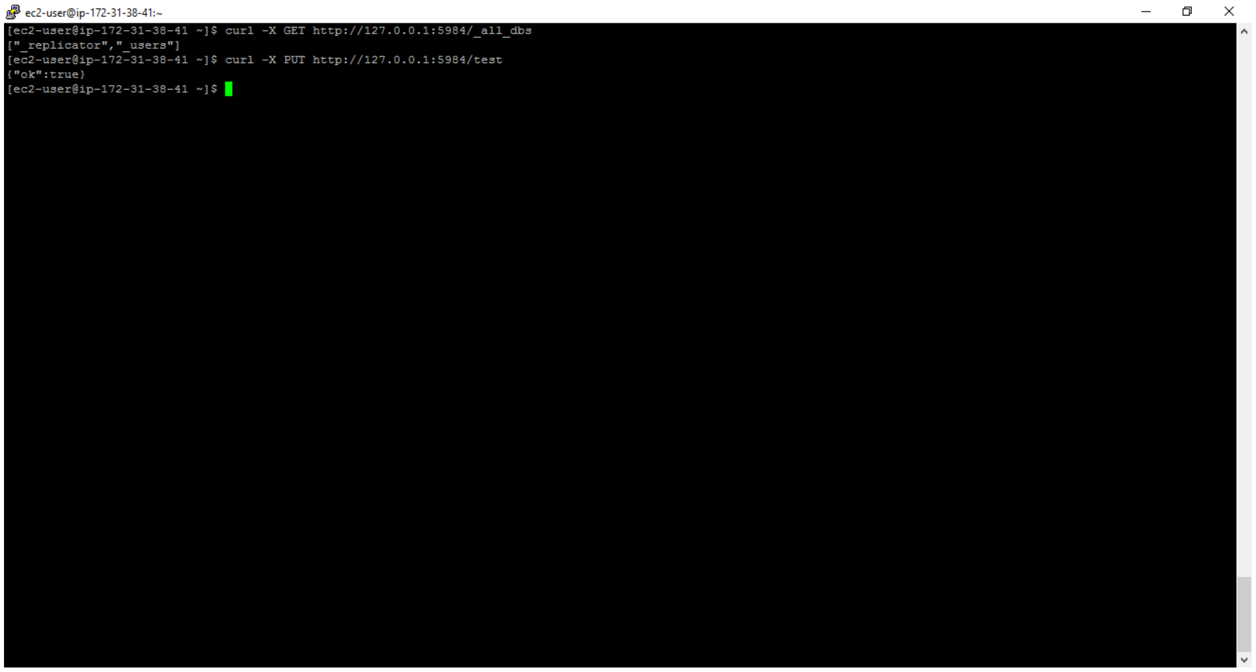
curl -X GET http://127.0.0.1:5984/_all_dbs



ec2-user@ip-172-31-38-41:~

```
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/_all_dbs
["_replicator","_users"]
[ec2-user@ip-172-31-38-41 ~]$
```

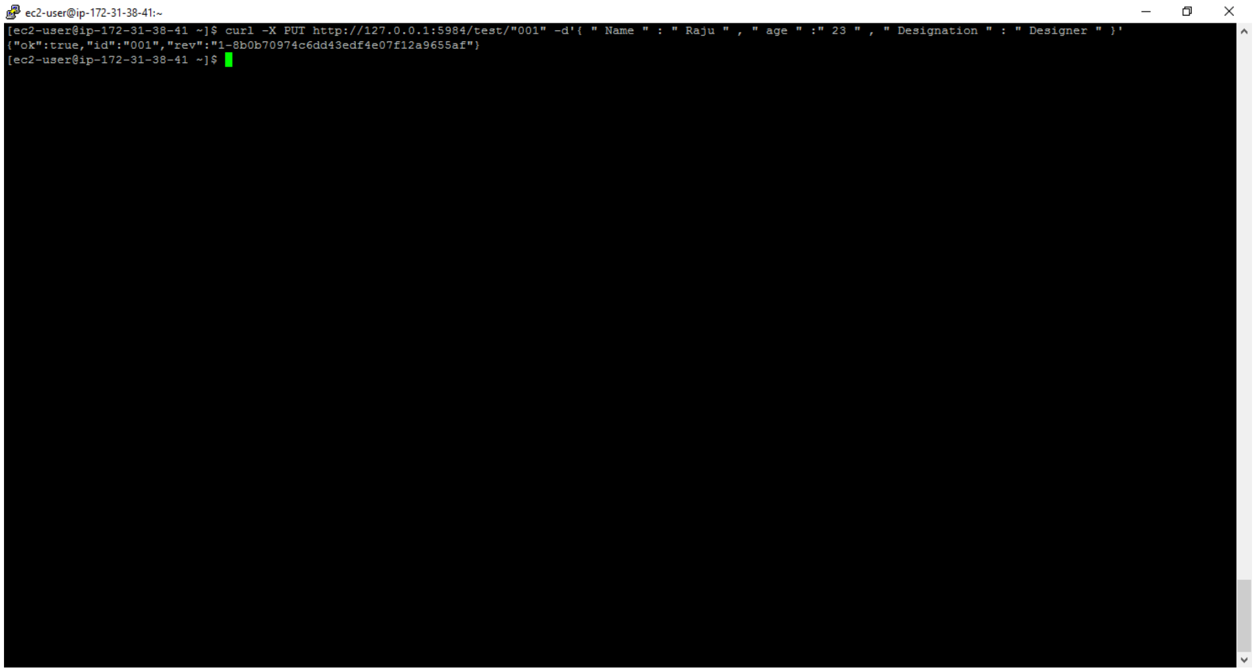
Create a DB:



```
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/_all_dbs
["_replicator","_users"]
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$
```

create a document:

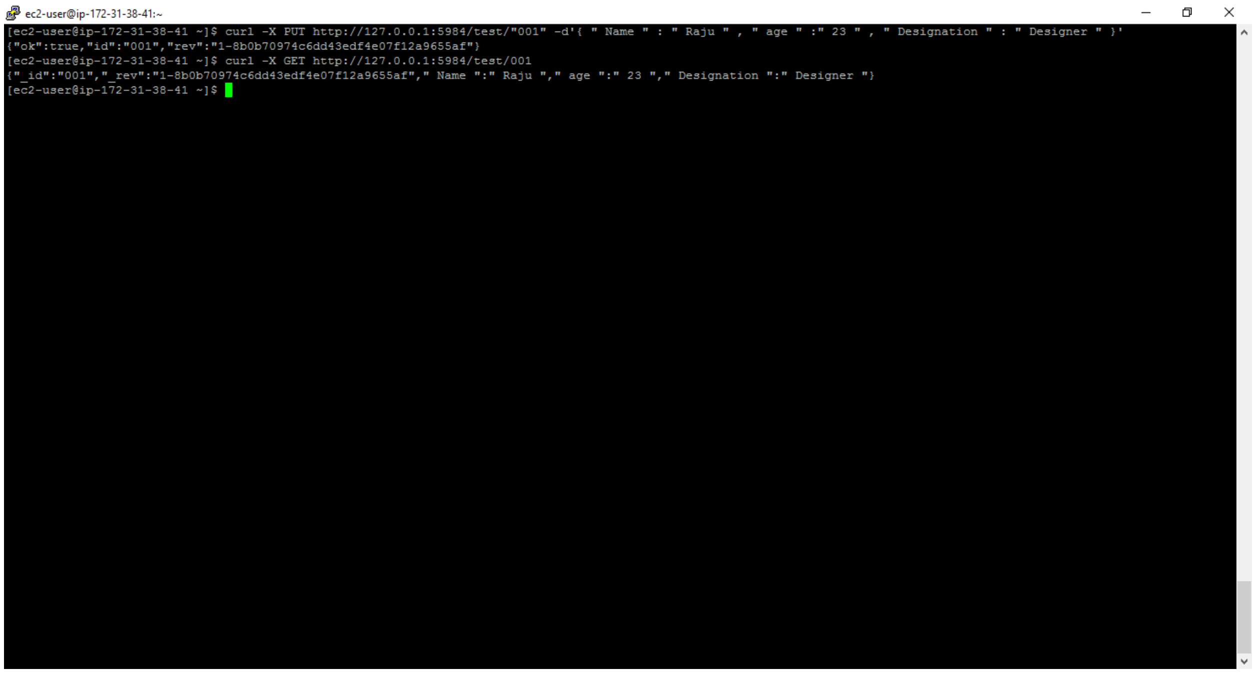
curl -X PUT http://127.0.0.1:5984/my_database/"001" -d'{ " Name " : " Raju " , " age " :" 23 " , " Designation " : " Designer " }'



```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test/"001" -d'{ " Name " : " Raju " , " age " :" 23 " , " Designation " : " Designer " }'
{"ok":true,"id":"001","rev":"1-8b0b70974c6dd43edf4e07f12a9655af"}
[ec2-user@ip-172-31-38-41 ~]$
```

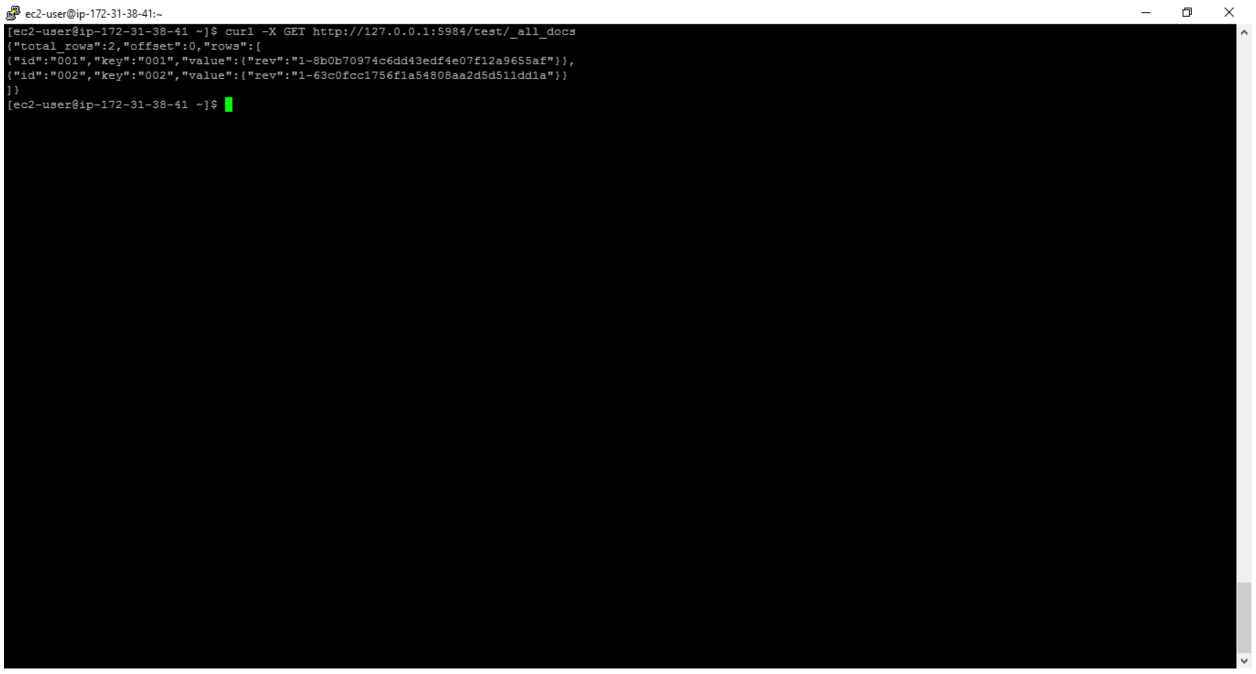
retrieve a document:

curl -X GET http://127.0.0.1:5984/my_database/001



```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test/"001" -d'{ " Name " : " Raju " , " age " :" 23 " , " Designation " : " Designer " }'
{"ok":true,"id":"001","rev":"1-8b0b70974c6dd43edf4e07f12a9655af"}
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test/001
{"_id":"001","_rev":"1-8b0b70974c6dd43edf4e07f12a9655af"," Name ":" Raju "," age ":" 23 "," Designation ":" Designer "}
[ec2-user@ip-172-31-38-41 ~]$
```

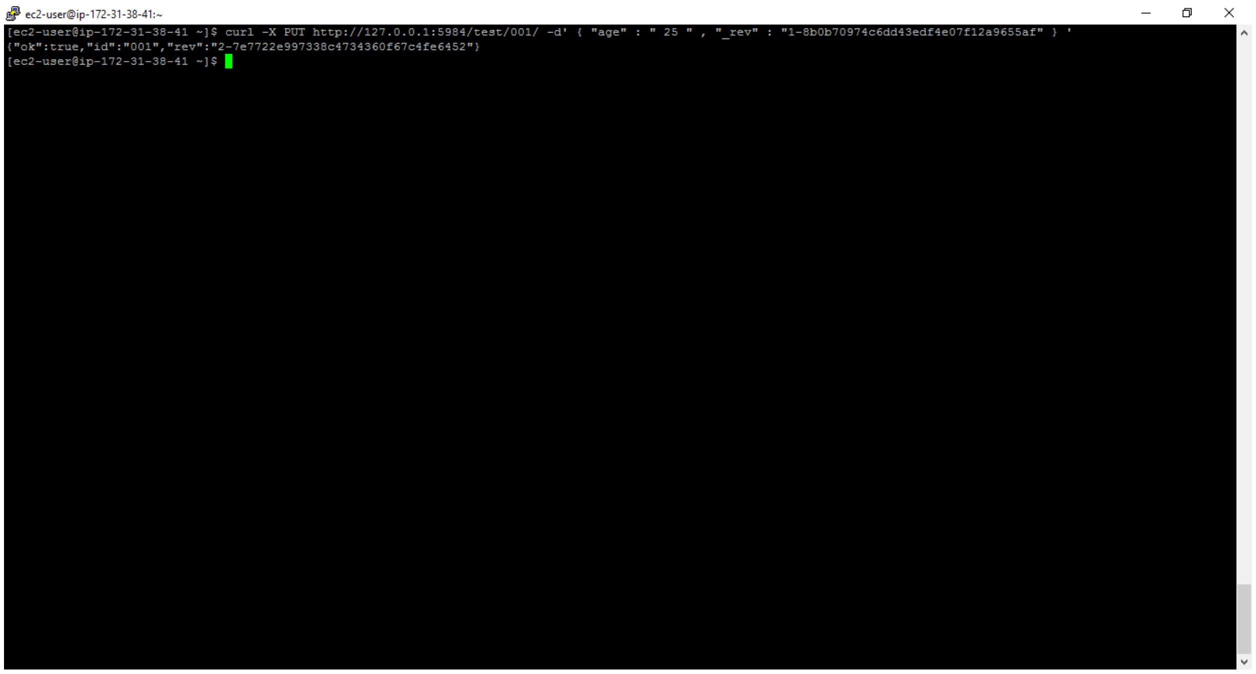
curl -X GET http://127.0.0.1:5984/my_database/_all_docs



```
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test/_all_docs
{"total_rows":2,"offset":0,"rows":[
{"id":"001","key":"001","value":{"rev":"1-8b0b70974c6dd43edf4e07f12a9655af"}},
{"id":"002","key":"002","value":{"rev":"1-63c0fcc1756f1a54808aa2d5d511dd1a"}}
]}
[ec2-user@ip-172-31-38-41 ~]$
```

update a document:

curl -X PUT http://127.0.0.1:5984/test/001/ -d' { " age " : " 24 " , "_rev" : "1-8b0b70974c6dd43edf4e07f12a9655af" } '

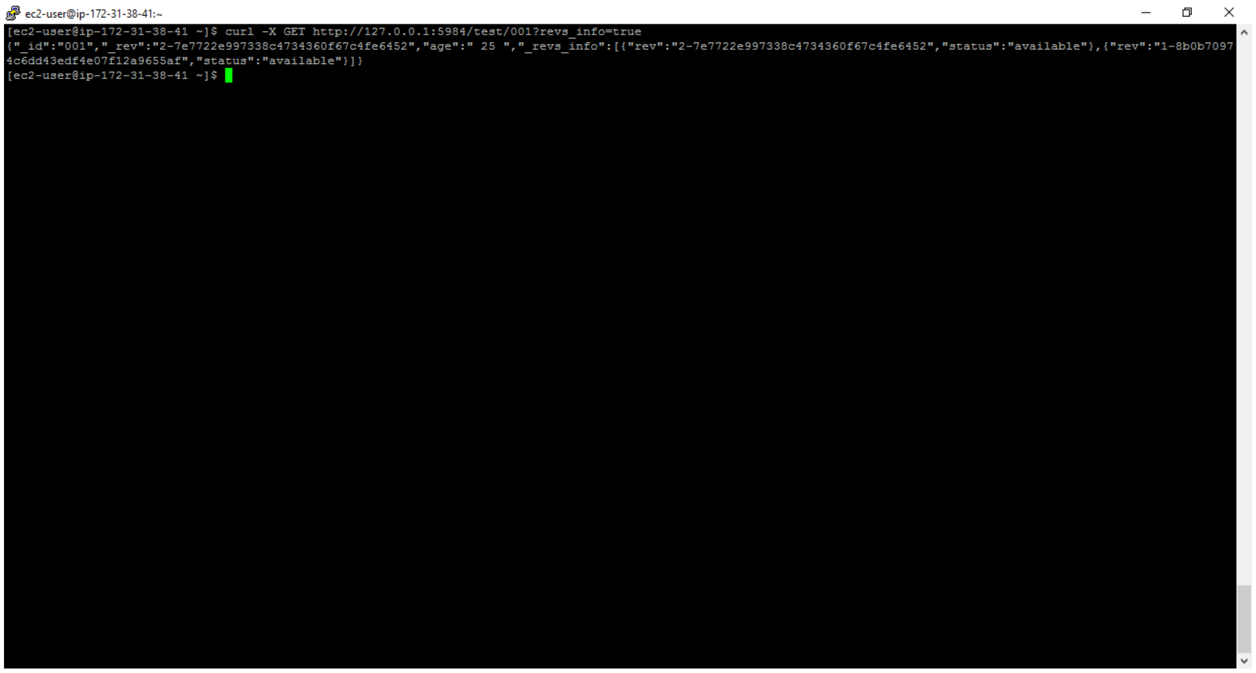


ec2-user@ip-172-31-38-41:~

```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test/001/ -d' { "age" : " 25 " , "_rev" : "1-8b0b70974c6dd43edf4e07f12a9655af" } '
{"ok":true,"id":"001","rev":"2-7e7722e997338c4734360f67c4fe6452"}
[ec2-user@ip-172-31-38-41 ~]$
```

GET revision info:

curl -X GET http://127.0.0.1:5984/my_database/001?revs_info=true

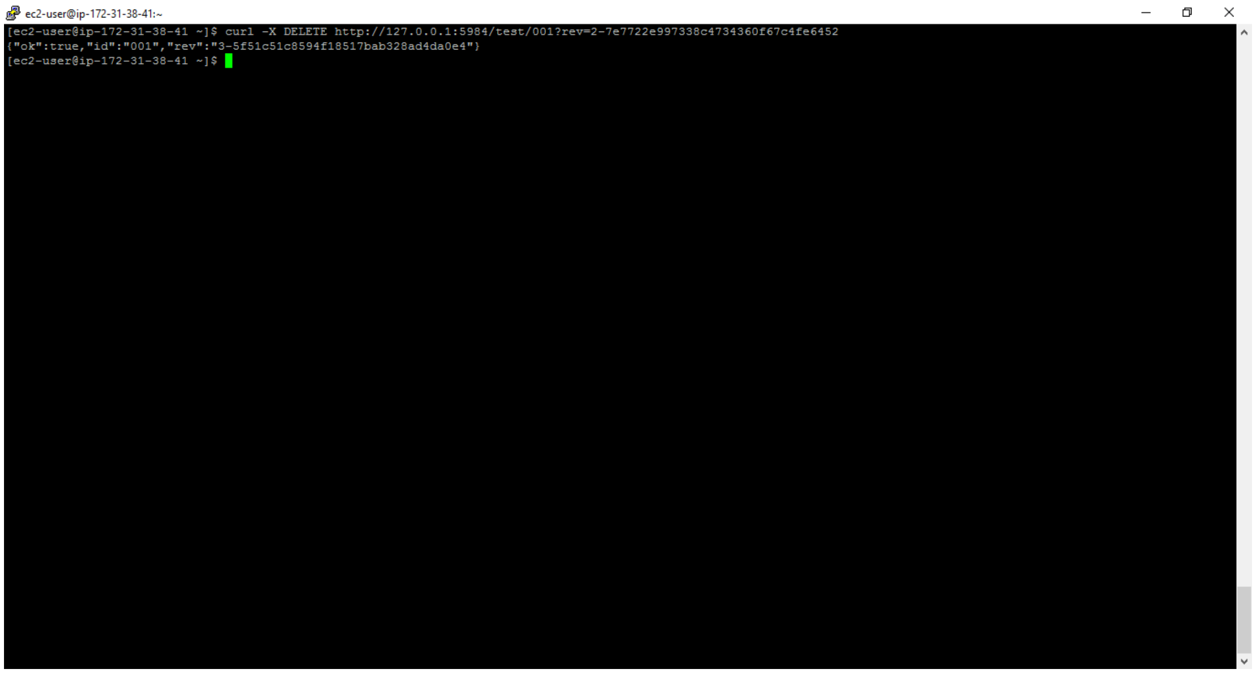


ec2-user@ip-172-31-38-41:~

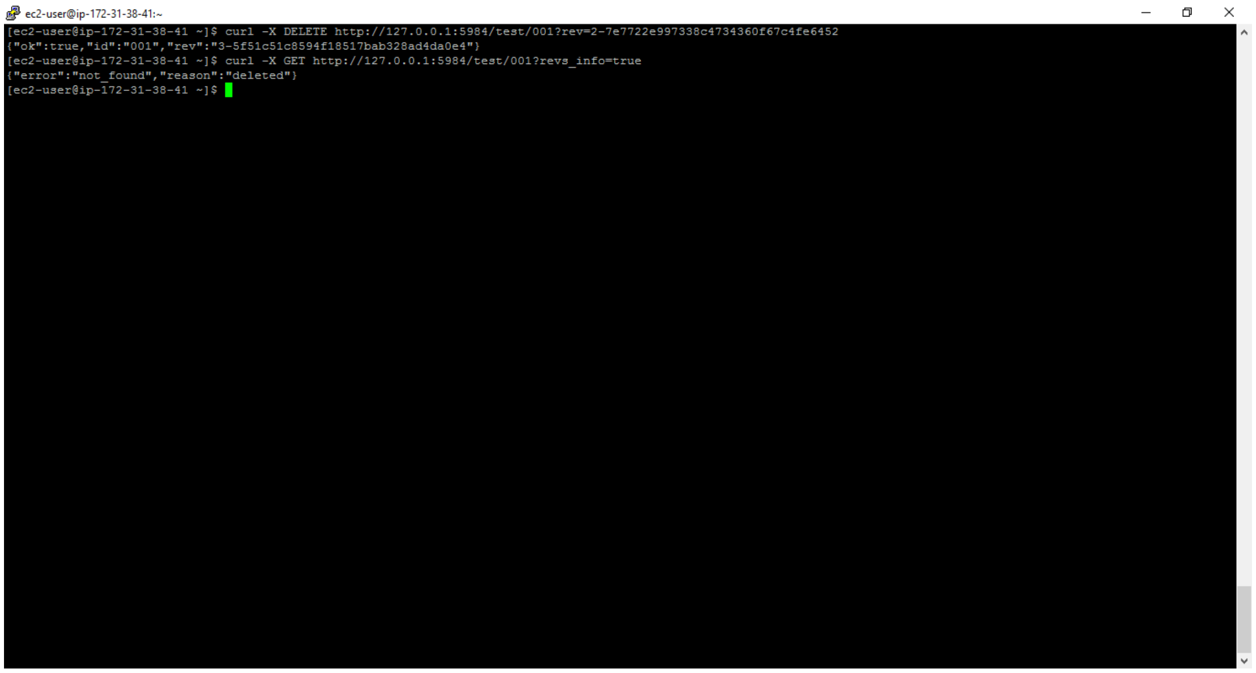
```
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test/001?revs_info=true
{"_id":"001","_rev":"2-7e7722e997338c4734360f67c4fe6452","age":" 25 ","_revs_info":[{"rev":"2-7e7722e997338c4734360f67c4fe6452","status":"available"},{"rev":"1-8b0b7097
4c6dd43edf4e07f12a9655af","status":"available"}]}
[ec2-user@ip-172-31-38-41 ~]$
```

delete a document:

curl -X DELETE http://127.0.0.1:5984/test/001?rev=1-3fcc78daac7a90803f0a5e383f4f1e1e



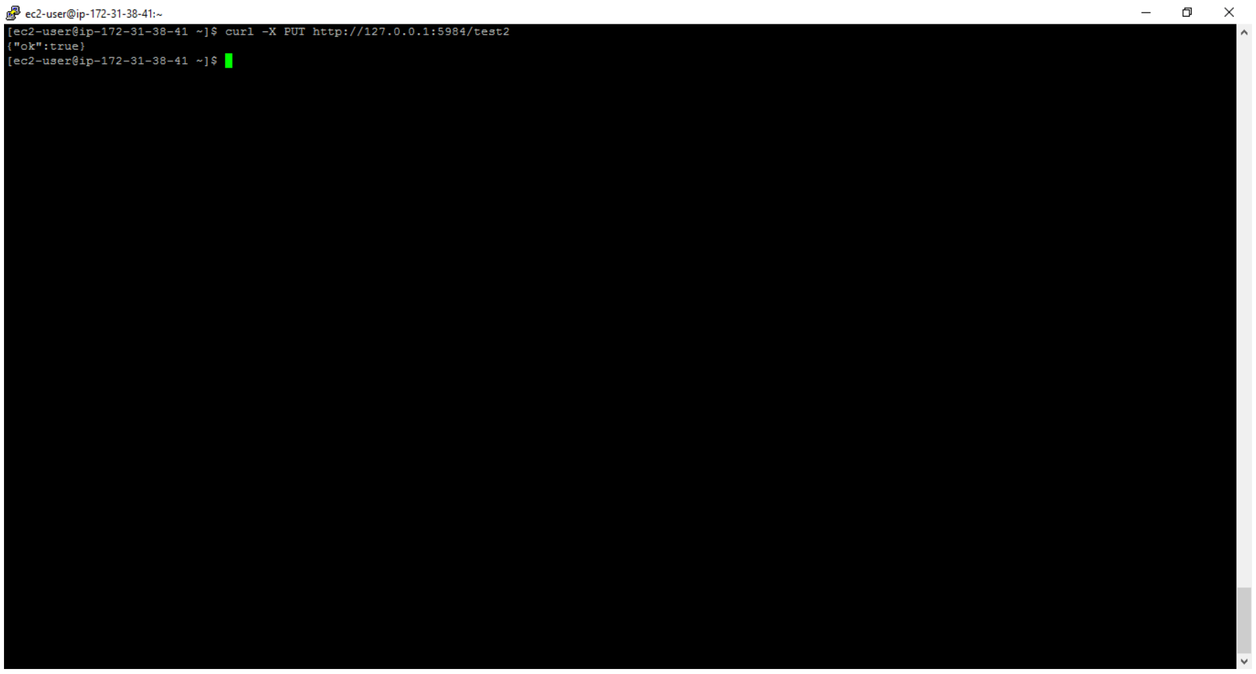
```
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test/001?rev=2-7e7722e997338c4734360f67c4fe6452
{"ok":true,"id":"001","rev":"3-5f51c51c8594f18517bab328ad4da0e4"}
[ec2-user@ip-172-31-38-41 ~]$
```



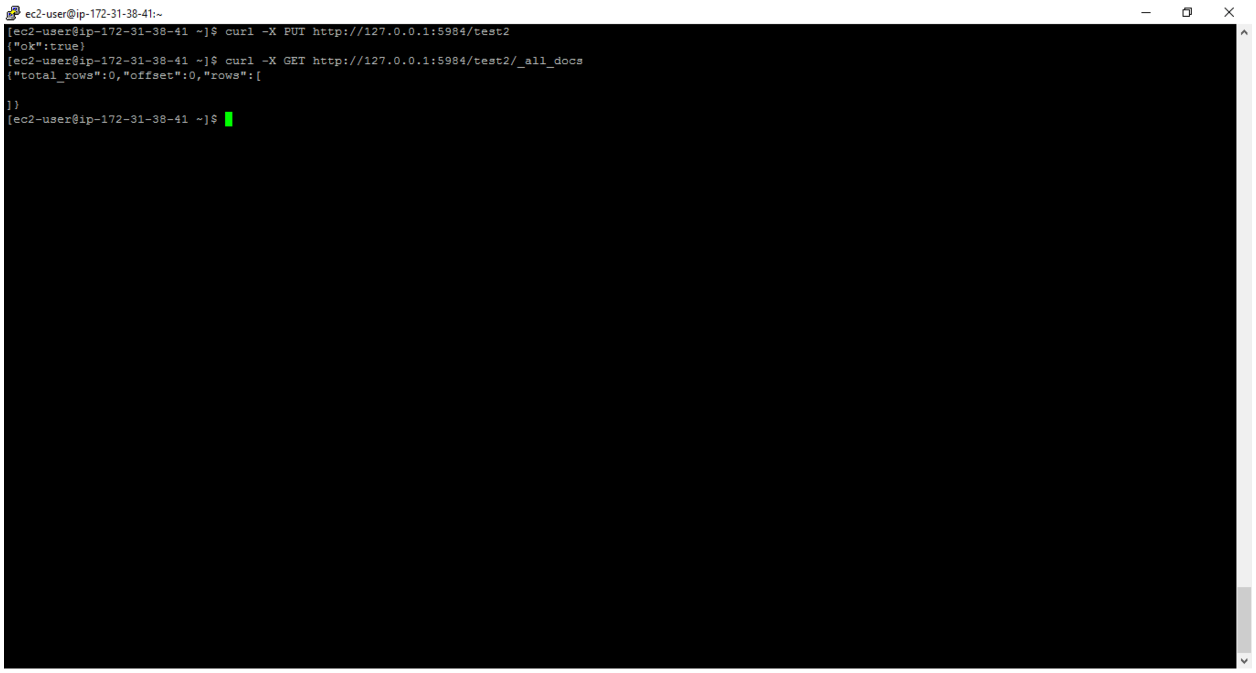
```
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test/001?rev=2-7e7722e997338c4734360f67c4fe6452
{"ok":true,"id":"001","rev":"3-5f51c51c8594f18517bab328ad4da0e4"}
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test/001?revs_info=true
{"error":"not_found","reason":"deleted"}
[ec2-user@ip-172-31-38-41 ~]$
```

Replicate a DB:

create database test2:

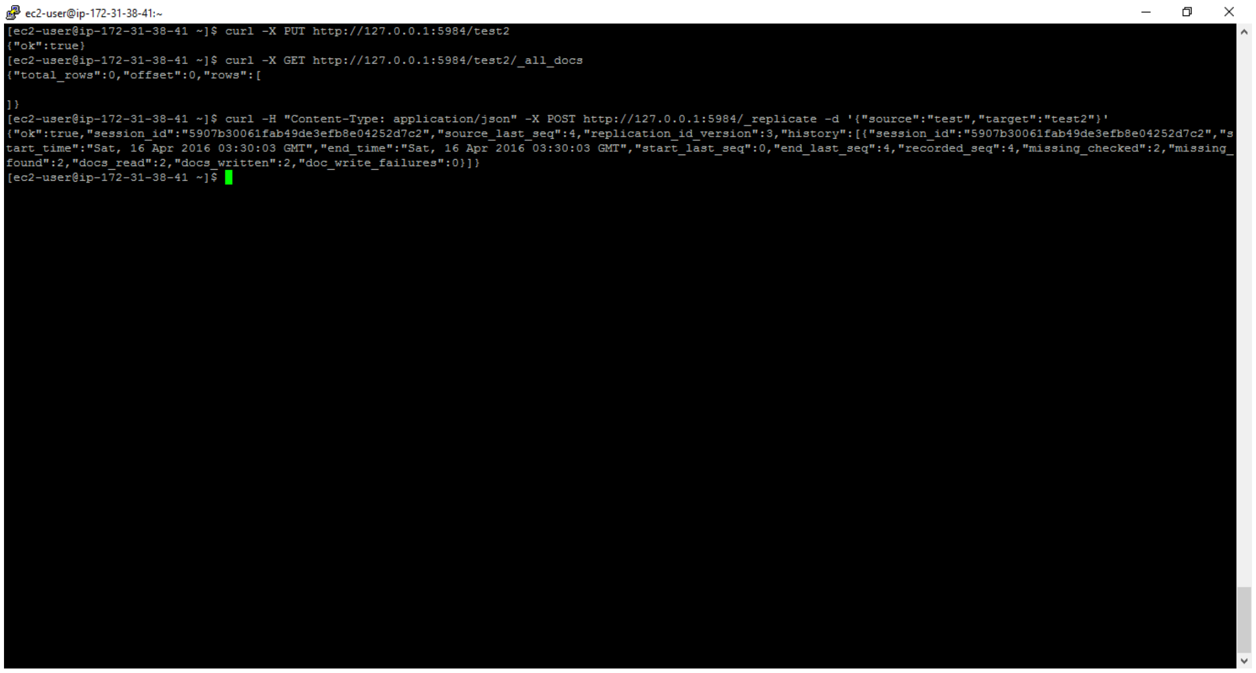


```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test2
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$
```

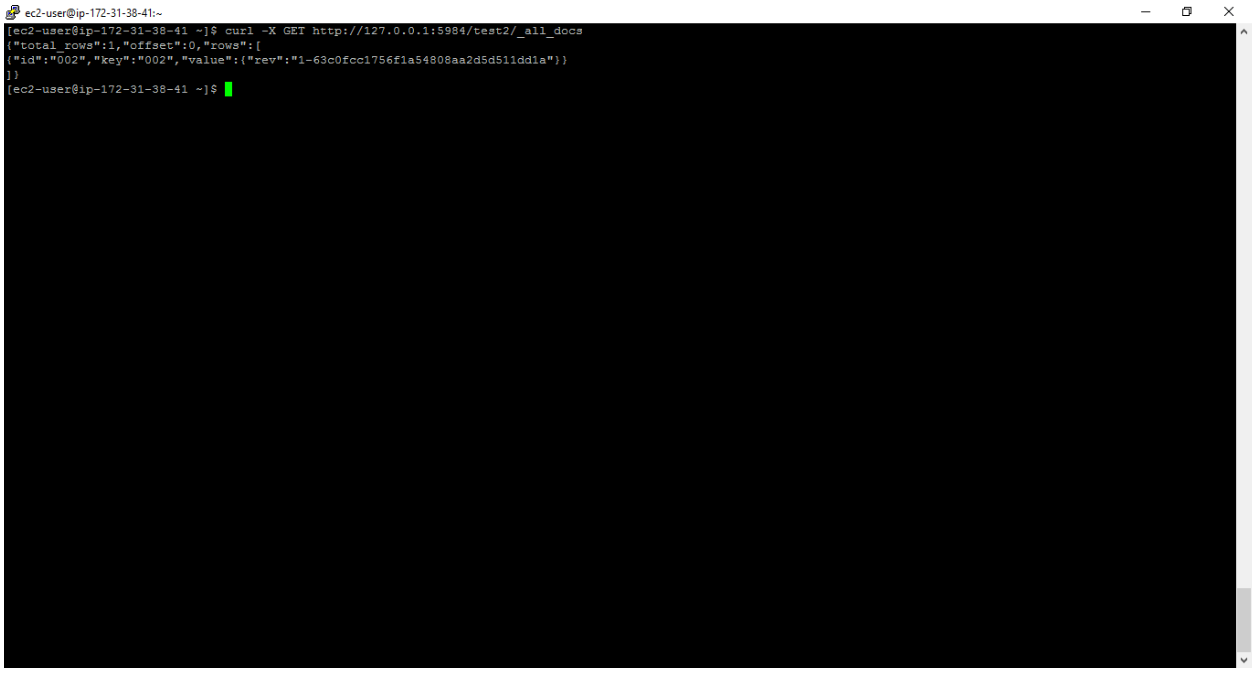


```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test2
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test2/_all_docs
{"total_rows":0,"offset":0,"rows":[

]}
[ec2-user@ip-172-31-38-41 ~]$
```

curl -H "Content-Type: application/json" -X POST http://127.0.0.1:5984/_replicate -d '{"source":"test","target":"test2"}'

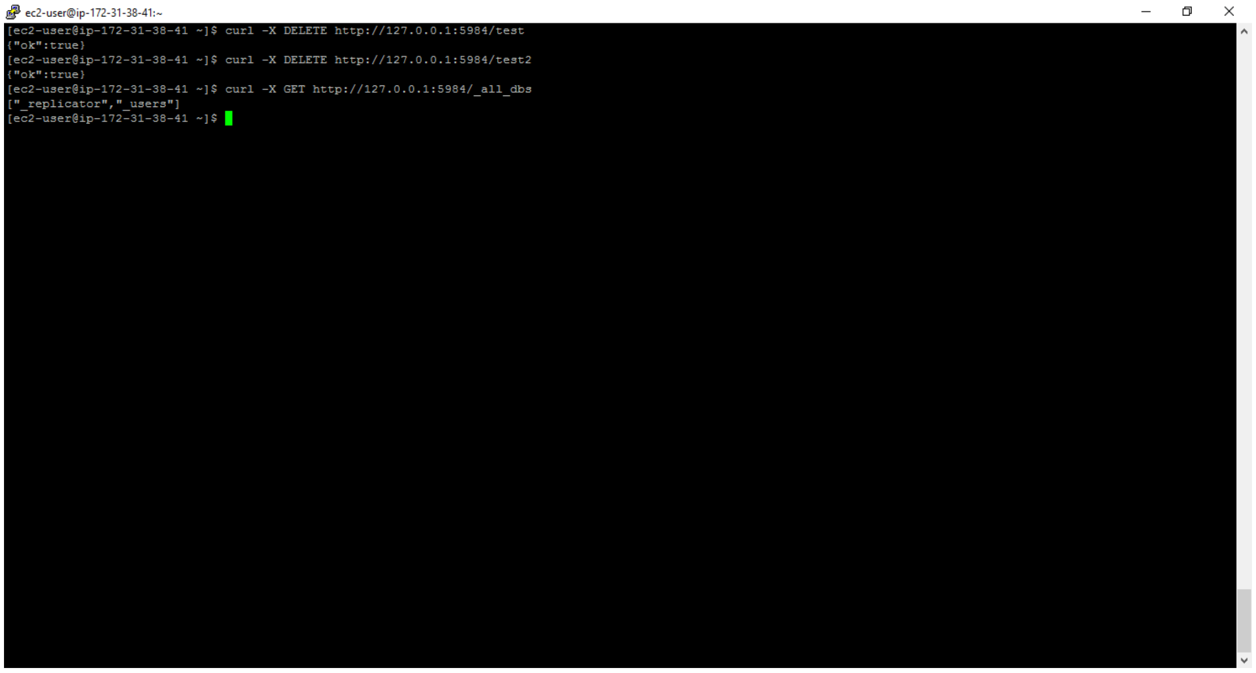


```
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test2
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test2/_all_docs
{"total_rows":0,"offset":0,"rows":[

]}
[ec2-user@ip-172-31-38-41 ~]$ curl -H "Content-Type: application/json" -X POST http://127.0.0.1:5984/_replicate -d '{"source":"test","target":"test2"}'
{"ok":true,"session_id":"5907b30061fab49de3efb8e04252d7c2","source_last_seq":4,"replication_id_version":3,"history":[{"session_id":"5907b30061fab49de3efb8e04252d7c2","s
tart_time":"Sat, 16 Apr 2016 03:30:03 GMT","end_time":"Sat, 16 Apr 2016 03:30:03 GMT","start_last_seq":0,"end_last_seq":4,"recorded_seq":4,"missing_checked":2,"missing_
found":2,"docs_read":2,"docs_written":2,"doc_write_failures":0}]}
[ec2-user@ip-172-31-38-41 ~]$
```



```
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test2/_all_docs
{"total_rows":1,"offset":0,"rows":[
{"id":"002","key":"002","value":{"rev":"1-63c0fcc1756f1a54808aa2d5d511dd1a"}}
]}
[ec2-user@ip-172-31-38-41 ~]$
```

Delete a database:



ec2-user@ip-172-31-38-41:~

```
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test2
{"ok":true}
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/_all_dbs
["_replicator","_users"]
[ec2-user@ip-172-31-38-41 ~]$
```