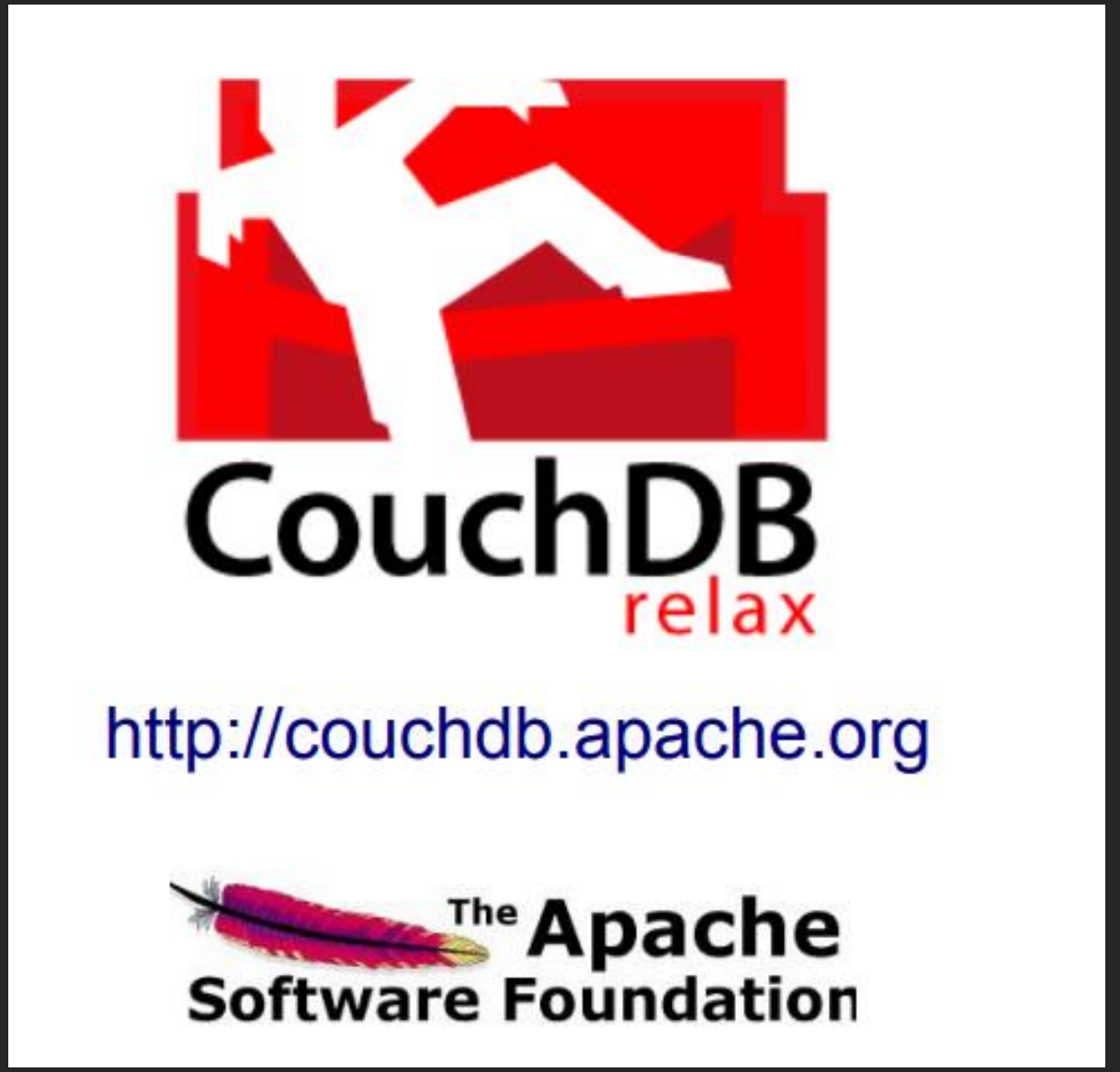
Introduction to CouchDB

Sophia Sandhu



Agenda

- CouchDB History
- CouchDB Characteristics
- Who is using Couch DB
- CouchDB Examples
- CouchDB
  - Documents
  - Storage
  - Views
  - Compaction
  - Replication
- CouchDB Lab

CouchDB's History

- Development started around 2005 by Damien Katz (Lotus Notes, IBM, MySQL)
- Initially written in C++, used XML and a query language similar to Formula (Lotus Notes)
- 2007 - adopted Erlang, JSON and MapReduce with JavaScript, dropping C++, XML and the custom query language
- Drew interest from IBM and got sponsored by IBM
- 2008 - became an Incubator project of the Apache Software Foundation
- 2009 - became a top level Apache project (alongside with httpd, Tomcat, etc)
- 2010 - Proposal to the HTML5 Indexed DB API to make it possible for someone to implement a CouchDB API on top of it (http://www.w3.org/TR/IndexedDB)

CouchDB's Characteristics

- It's a documented-oriented DBMS (i.e. Not a relational DBMS)
- Data is modeled as "documents", resembling real world documents and promoting self-contained data
- A document is a JSON structure containing any kind and number of fields
- Arbitrary binary data (video, audio, images, etc) is supported as document attachments
- Communication with the outside world done exclusively through an HTTP RESTful API
- Has incremental peer to peer replication -very simple and it uses exclusively the public HTTP RESTFul API (no hidden/special APIs or protocols)

CouchDB's Characteristics

- Uses MapReduce for computing views
- ACID properties guaranteed at a single document level
- Doesn't support transactions for updating/adding/removing multiple documents at once -a choice by design
- Instead of locks, it uses Multi Version Concurrency Control (MVCC), i.e. revision numbers, to manage concurrent requests
- Main goals: availability and partition tolerance

CouchDB’ vs RDBMS

- RDBMSs do not scale so gracefully due to their focus on distributed consistency:
- Before writing to a node, synchronization protocols must run between all the nodes and then some distributed commit protocol takes action.
- Client requests are blocked until all nodes reach an agreement and data is committed to all nodes .
- More nodes => more communication overhead => more time for reaching agreements

Who is using Couch DB?

- - Canonical Ltd, Ubuntu One service
- BBC UK
- Mozilla is using it for Raindrop.
- Facebook
- Assay Depot, a big marketplace in USA for pharmaceutical research services.

HTTP RESTful API examples


```
$ curl -X PUT http://localhost:5984/albums
{"ok":true}

$ curl -X PUT http://localhost:5984/albums/album1 -d @-
       { "artist": "Megadeth",
        "title": "Endgame",
        "year":     2009 }
{"ok":true,"id":"album1","rev":"1-35cc06dfb32b482b2e9031a4e3ebaef0"}

$ curl -X GET http://localhost:5984/albums/album1
 {
"_id":"album1","_rev":"1-35cc06dfb32b482b2e9031a4e3ebaef0", "artist":"Megadeth",
"title":"Endgame",
"year":2009
}
```

HTTP RESTful API examples

To update a document, we need to supply its latest revision, otherwise we get an HTTP response with error code 409

```
$ curl -X PUT http://localhost:5984/albums/album1 -d @-{
 "artist": "Megadeth",
"title": "Endgame",
 "year": 2010
}
{"error":"conflict","reason":"Documment update conflict.“}
```

An attribute named “_rev” specifies the target revison:

```
$ curl -X PUT http://localhost:5984/albums/album1 -d @-{
 “_rev": "1-35cc06dfb32b482b2e9031a4e3ebaef0",
 "artist": "Megadeth",
 "title": "Endgame",
 "year": 2010
}
```

HTTP RESTful API examples


```
$ curl -X DELETE http://localhost:5984/albums/album1?rev=\2-
a568258ebba8a79c2ebe9fb5c3700ac2

{"ok":true,"id":"album1","rev":"3-ea701815806524a3bdf7c81e35276801"}

$ curl -X GET http://localhost:5984/albums/album1
{"error":"not_found","reason":"deleted"}

$ curl -X GET http://localhost:5984/albums/album1?rev=\3
ea701815806524a3bdf7c81e35276801
{"_id":"album1","_rev":"3-ea701815806524a3bdf7c81e35276801","_deleted":true}

$ curl -X POST http://localhost:5984/albums/_purge \-d
'{ "album1": ["3-ea701815806524a3bdf7c81e35276801"]}'
{"purge_seq":1,"purged":{"album1":["3-ea701815806524a3bdf7c81e35276801"]}}
```

Document Attachments

- Special attribute “_attachments” in a document with attachments:

```
$ curl -X GET http://localhost:5984/albums/album1
{
      "_id": "album1",
      "_rev": "2-2e0e230f5a3c44aa1dac72e5506ae213",
      "artist": "Megadeth",
      "title": "Endgame",
      "year": 2010,
      "_attachments":
            {"cover.png":
             {"content_type": "image/png",
              "length": 2157,
              "stub": true}}}
```

CouchDB Storage

- Each DB corresponds to a single file
- All data is written in file append mode – existing data is never ever overwritten => no need for data integrity check and recovery on server startup
- Uses B-Trees – a data structure for efficient operations on very large data sets stored in disk. For each tree operation (search, insert, delete) only 2 nodes at most need to be in RAM memory for each step
- Each DB file contains 2 B-Trees:
  - by_doc_id_btree – keys are document IDs, values contain pointers (offsets) to where the document is located within the DB file and other metadata
  - by_db_seq_btree – keys are DB sequence numbers, values are information about what happened (document added or updated) for a particular DB sequence number

CouchDB Views

- To some extent, they have the same end result as SQL in RDBMSs.
- Views can be defined using a Map and (optionally) a Reduce function. JavaScript is the preferred language, but there is 3rd party support for others (except for Erlang which is natively supported as well)
- Views are mapped to individual files as well => they can be stored on different disks for higher IO performance.
- Like a DB file, a view file contains a B-Tree (a single one)
- View B-Tree keys are the ones emitted by a Map function.
- Values emitted by a Map function are stored in the leaf nodes of the B-Tree.
- A non-leaf node of the B-Tree stores the result of a Reduce operation over the values of its children => Reduce functions are used as well for doing a “Re-Reduce”

CouchDB Views

- When a document is updated/added/deleted, Map and Reduce computations are minimized:
  - The Map value for the affected document is re-calculated
  - The Reduce value of the parent node is re-computed
  - Finally re-computation of Reduce values takes place all the way up the B-Tree(Re-Reduce operations)
- Typically the B-Tree has from 4 to 6 levels for a DB with millions of documents.
- Updating the view after a few documents were updated and/or added is cheap.
- Map and Reduce functions are executed by the View Server (it uses Mozilla's SpiderMonkey JavaScript engine)

Couch View Examples

Example Dataset

```
{
    "_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
}
```

Couch View Examples

- View definitions (Map and Reduce functions) are stored in “design” documents – their ID is prefixed with “_design/” and have a special meaning in CouchDB

```
{
    "_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);
                       }"
        }
    }
}
```

Couch View Examples

Getting the Reduce result for all emitted values, ignoring key values:

```
$ curl http://localhost:5984/albums/_design/foobar/_view/by_year
{
    "update_seq": 6,
    "rows": [
        {"key": null, "value": 5}
    ]
}
```

Getting the Reduce for all emitted values but for each distinct key:

```
$ 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}
    ]
}
```

Compaction

- DB files (as well as View files) are written in append mode only, not overriding any previously written data
- DB files and View files will grow indefinitely!!!Unless no more documents are added or updated...
- A DB or View compaction operation can be triggered.
- A compaction operation does the following
  - Creates a new file
  - Starts traversing the DB or View B-Tree and lookups the most recent data pointed by each node (documents or map or reduce values)
  - Writes that most recent data to the new file
  - When the original DB or View file is not being accessed to serve a request, it deletes it and renames the compacted file to the original DB/View file name.

Replication

- Peer to peer replication, no concept of master or slave roles.
- A replication specifies a source and a target DB (unidirectional)
- Any of the DBs, or both, can be local or remote
- Requesting a replication is very simple

```
$ curl -X POST http://localhost:5984/_replicate
-d @-
{"source": "http://example.com/somedb",
 "target": "somedb_copy"}
```

Replication

- It creates a checkpoint history document (a document with the prefix “_local/” as part of its ID) in the source and target DBs
- It uses the _changes API from the source DB
  - For each received line from _changes, which mentions a source DB sequence number SEQ_NUM, retrieves the corresponding Document from the source DB2
  - Inserts the document into the target DB (if it's a remote DB through the public HTTP RESTful API)
  - Changes the replication checkpoint history of both DBs (source and remote) to mark that the source DB sequence number SEQ_NUM was processed
- After a crash (source or target):
  - The replicator compares the checkpoint history of both DBs and choses the highest common source DB sequence number COMMON_SEQ_NUM
  - Starts doing the above process but tells the _changes API to start from sequence number COMMON_SEQ_NUM

Replication Conflicts

Alice then decides to edit her version of the document:

In Alice's server	In Bob's server
{ "_id": "foo", "_rev": "2-72b1d3f01f345e7aa", "phone": "789123", "zip_code": "999" }	{ "_id": "foo", "_rev": "1-abcdef0123456789", "phone": "12345", "zip_code": "999" }

Not being aware of what Alice did to her copy, Bob decided to edit his copy:

In Alice's server	In Bob's server
{ "_id": "foo", "_rev": "2-72b1d3f01f345e7aa", "phone": "789123888", "zip_code": "999" }	{ "_id": "foo", "_rev": "2-111fff0893856def", "phone": "12345", "zip_code": "5984" }

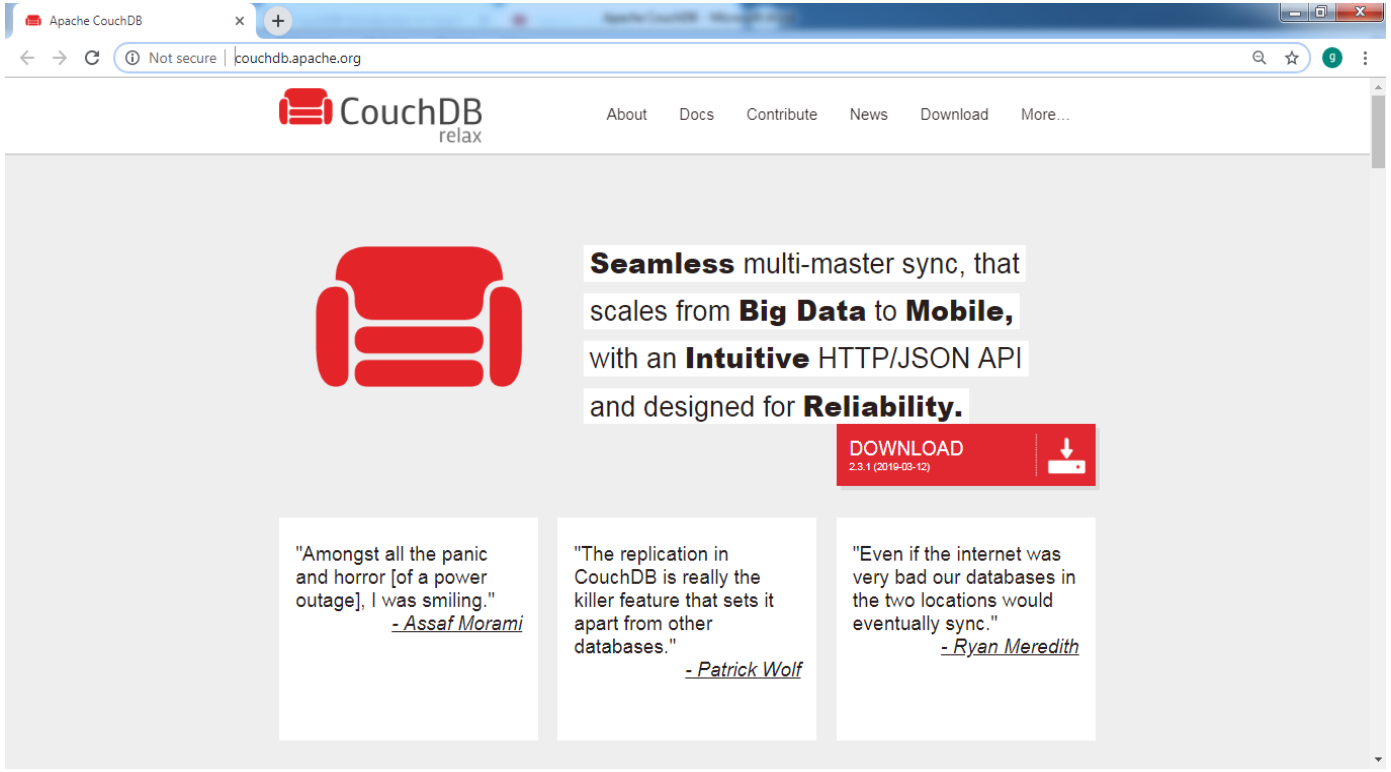
Replication Conflicts

- Later on they decide to synchronize their DB's.
- What happens? Which version remains in DB?
- Both versions will remain. No data is lost on either side.
- But what will happen when :
  - GET http://alice.com/contacts-db/foo
  - GET http://bob.com/contacts-db/foo
- Simple: both request will return exactly the same document. When conflicting versions of a document are detected, CouchDB uses a deterministic algorithm to elect one version as the "winning" version. The elected "winning" version will be the same on all peers.

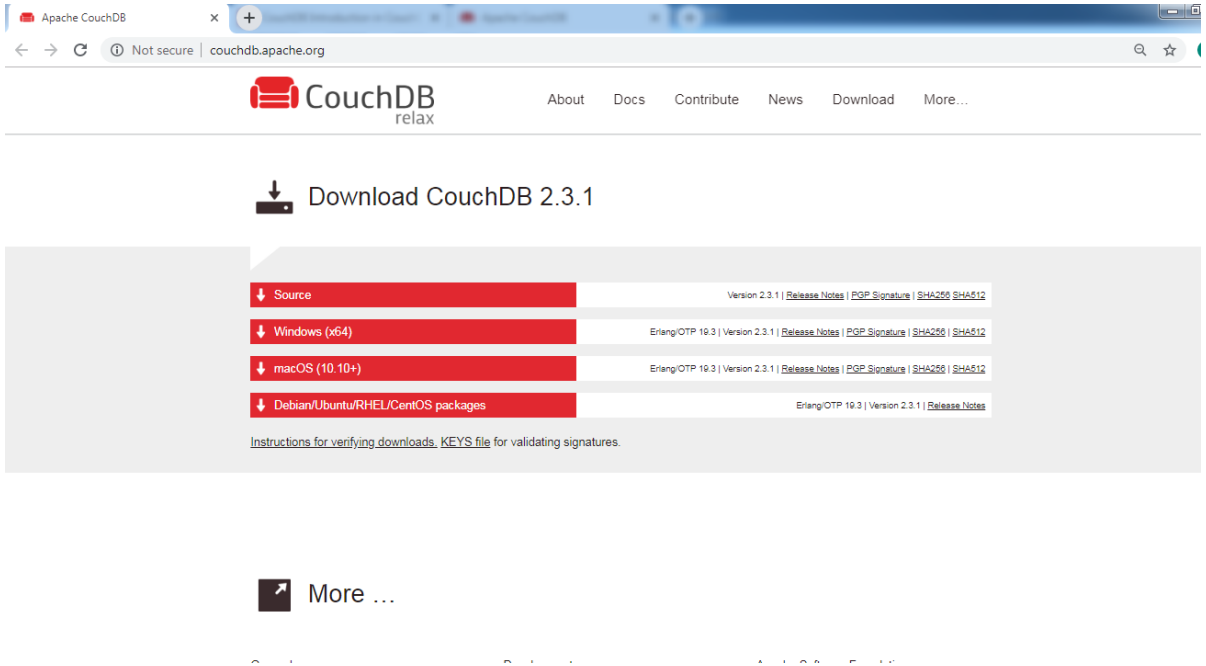
COUCHDB LAB

Installing Couch DB

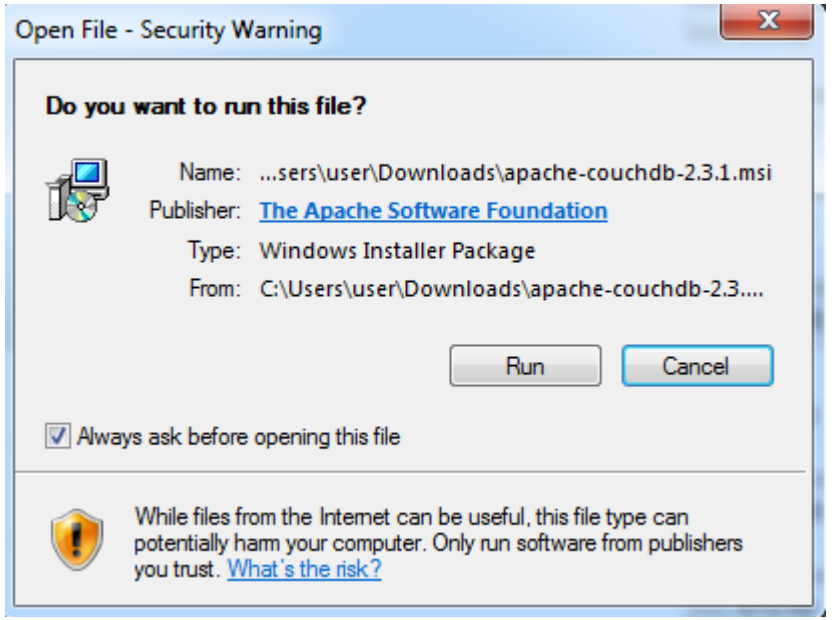
– For downloading the setup file of CouchDB, go to the official website: http://couchdb.apache.org/.



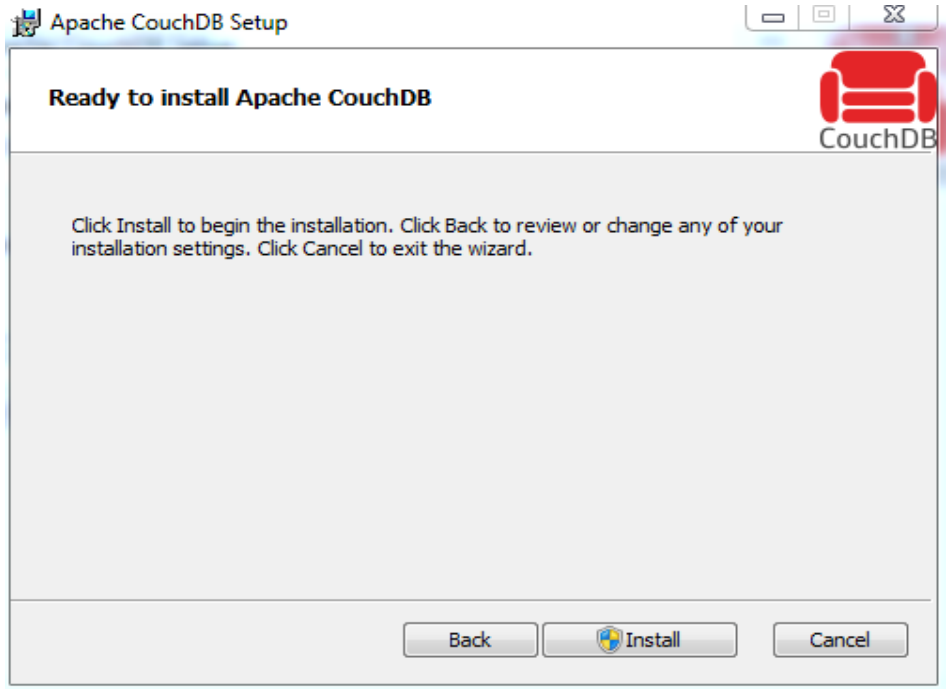
- Now, click on the Download option and then it leads to the page where various download links are provided.



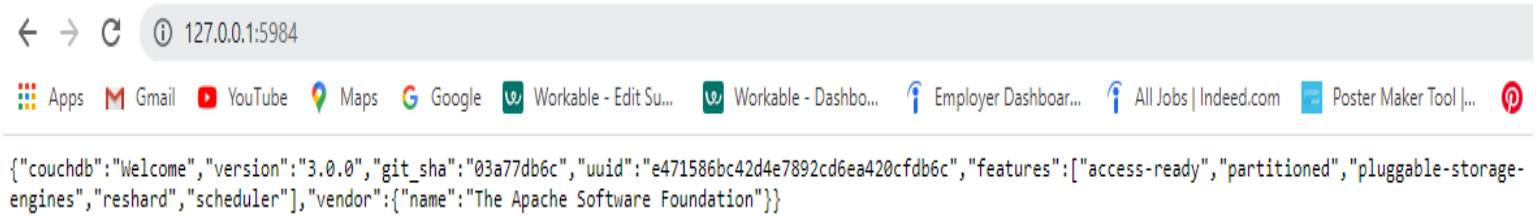
- Now, Click on the windows option. After 2-3 minutes, CouchDB will be download in your system in the form of Setup file.
- Now, run that setup file **apache-Couchdb-2.3.1.**



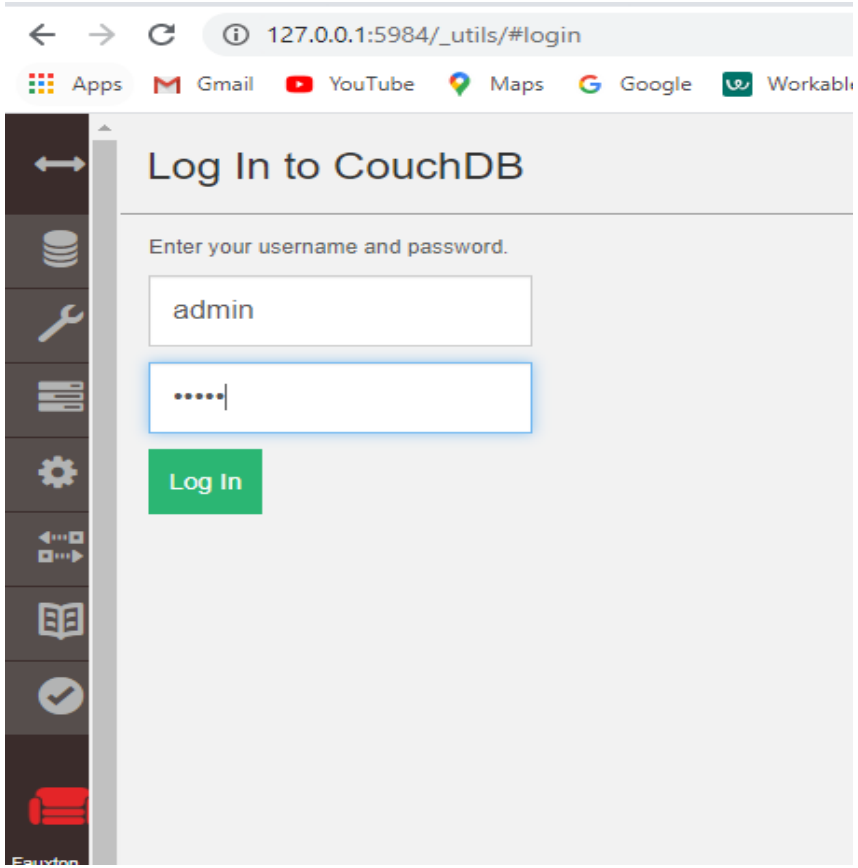
- After clicking on the run button, proceed with the installation. Default Settings are recommended, click on next button.
- Once you reach the Ready to install screen, click on Install. And after that, a confirmation dialog box will appear and in that click on ‘yes’ button. Your installation will complete in some seconds.



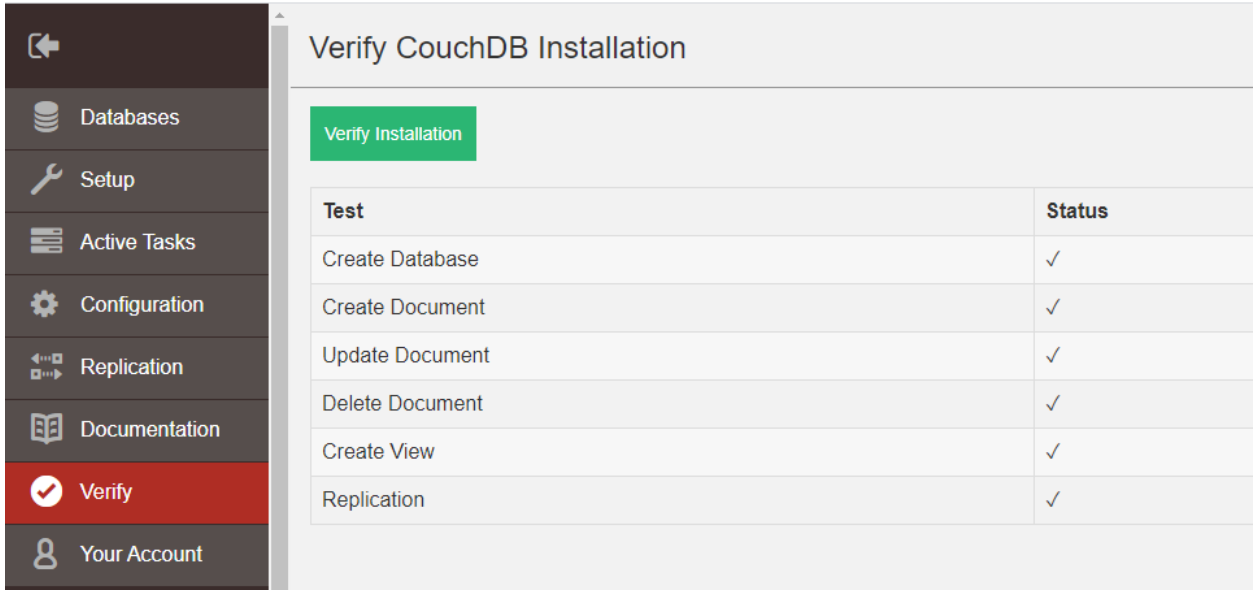
- After the complete installation opens the browser and type the following URL: http://127.0.0.1:5984/ and open the link. If everything goes fine, the resulting output will appear:



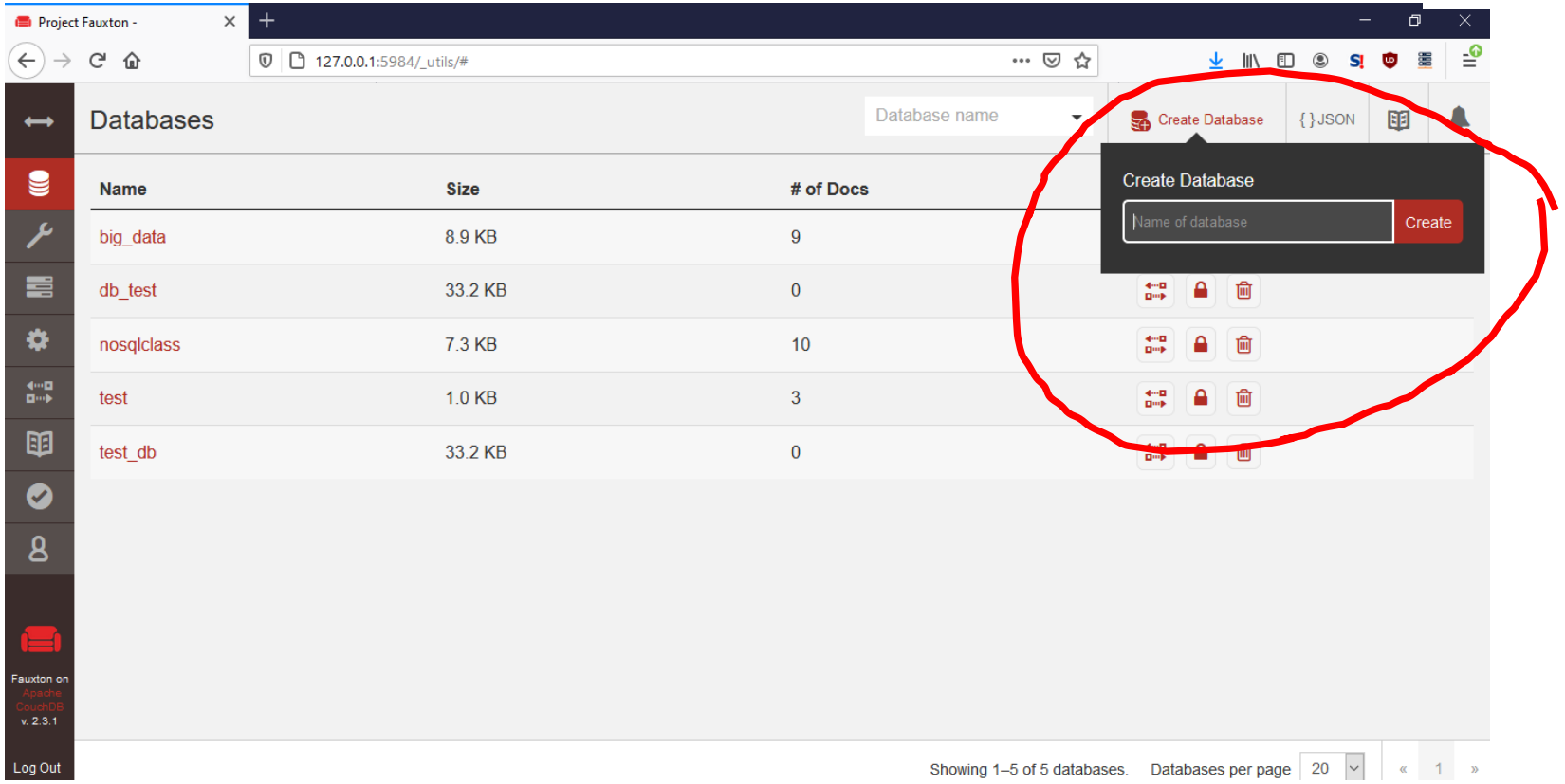
- Now, by typing this URL: http://127.0.0.1:5984/_utils/, you can interact with the CouchDB web interface, which shows the index page of Futon as shown in the figure:



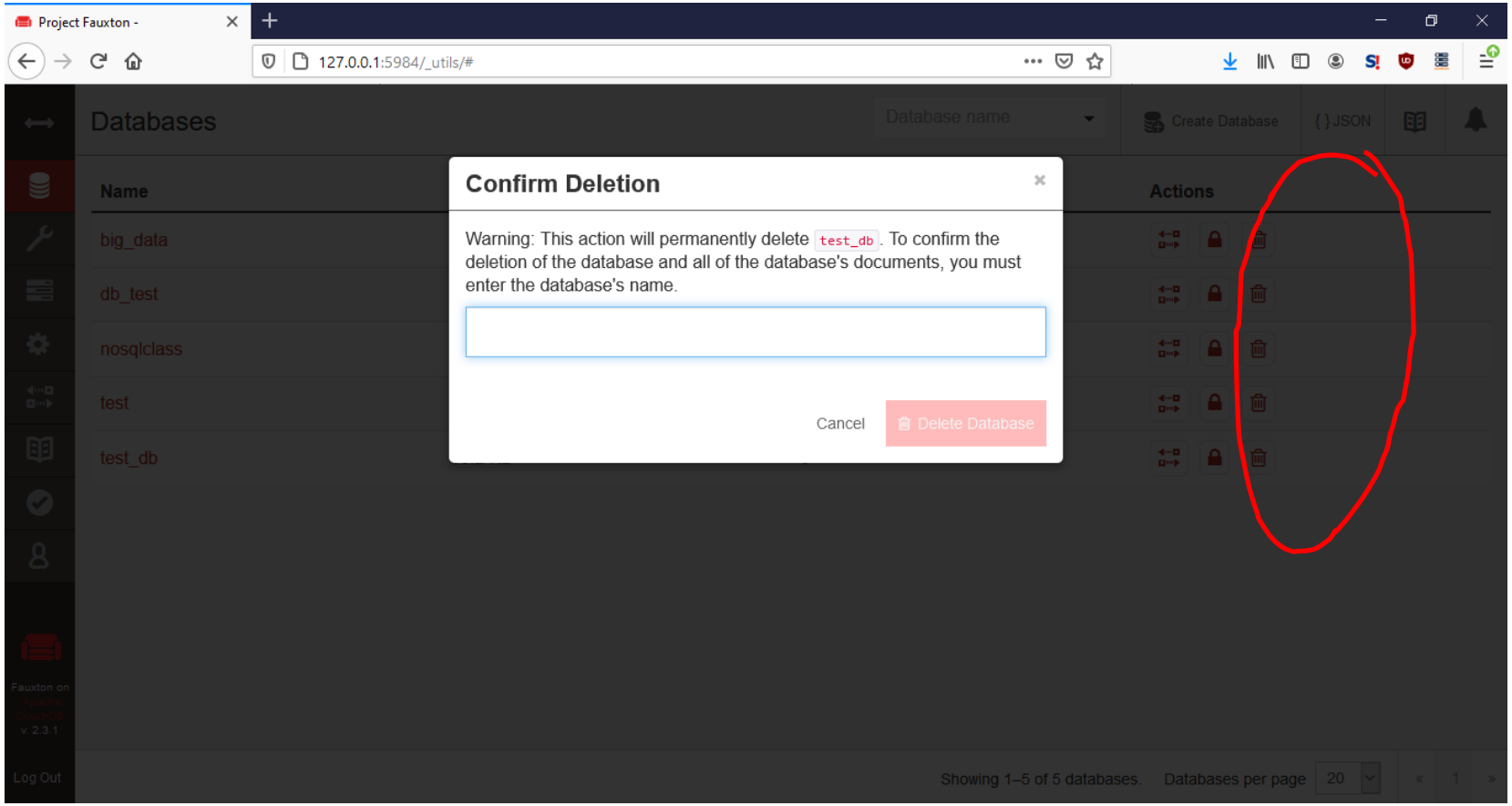
- Verify the CouchDB installation by going to the verify tab that shown in below figure. Now Click on the Verify installation.
- If your CouchDB is installed Successfully, a window will appear as shown below:



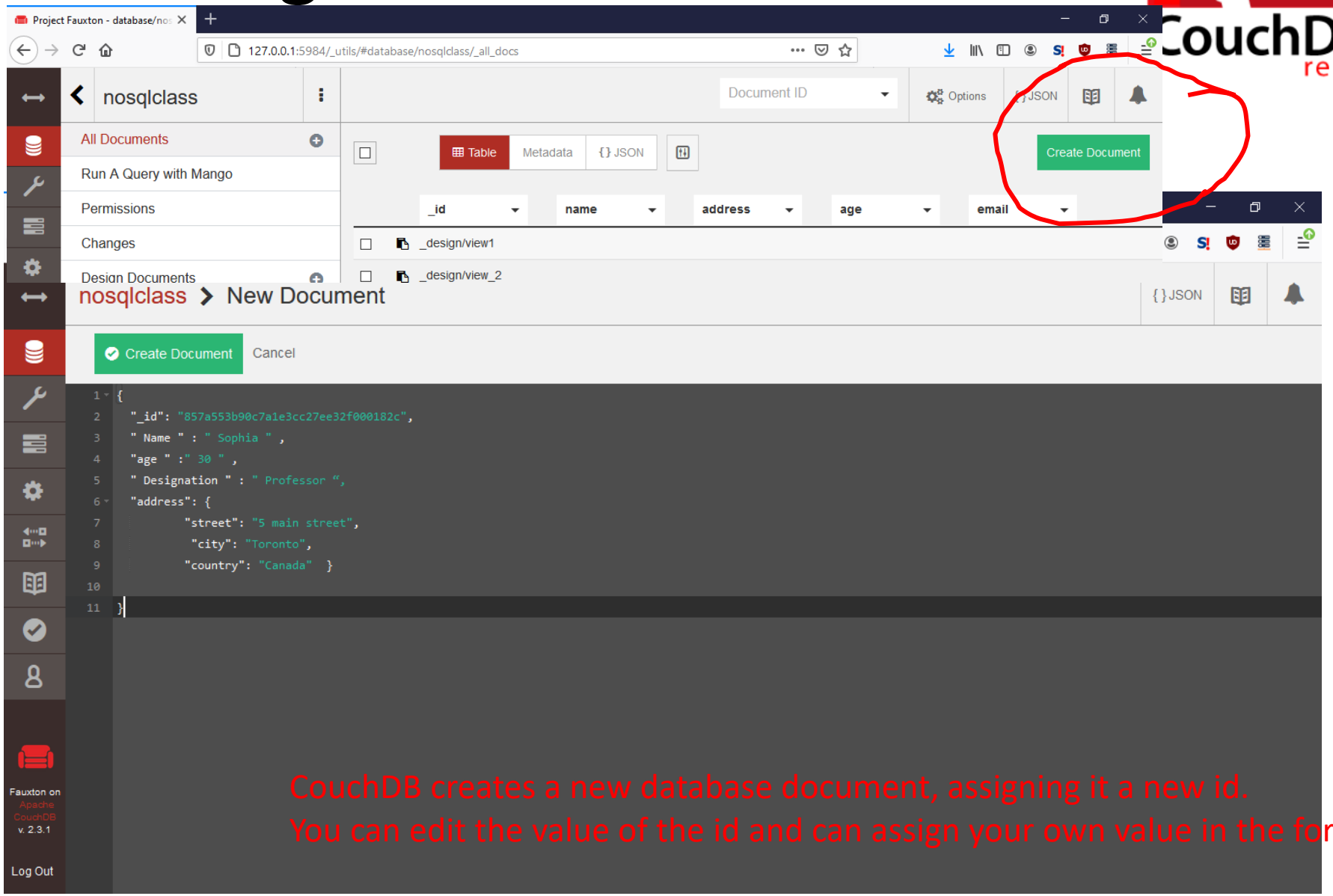
Futon: Create DB



Futon: Delete DB

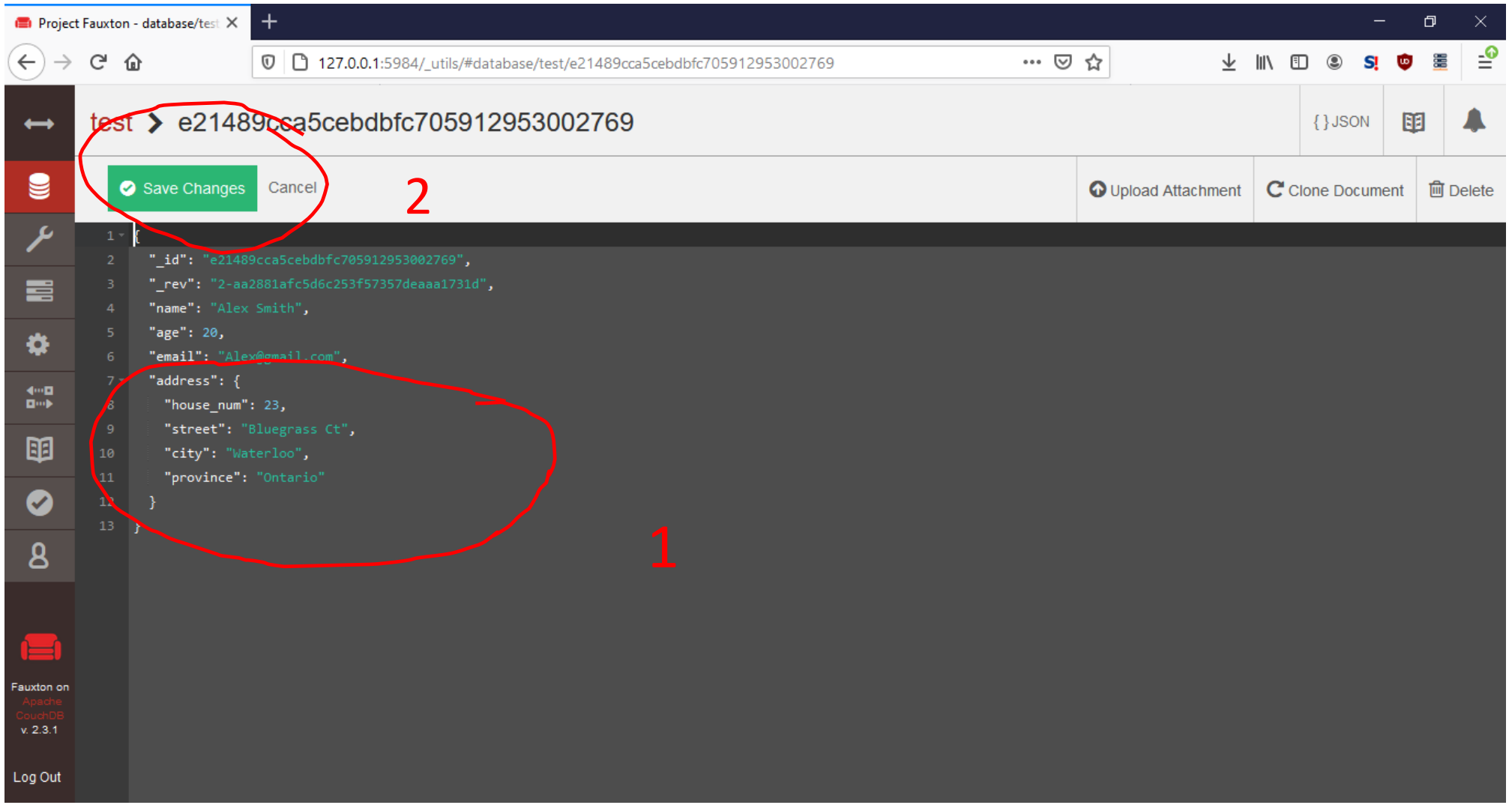


Futon: Creating a document

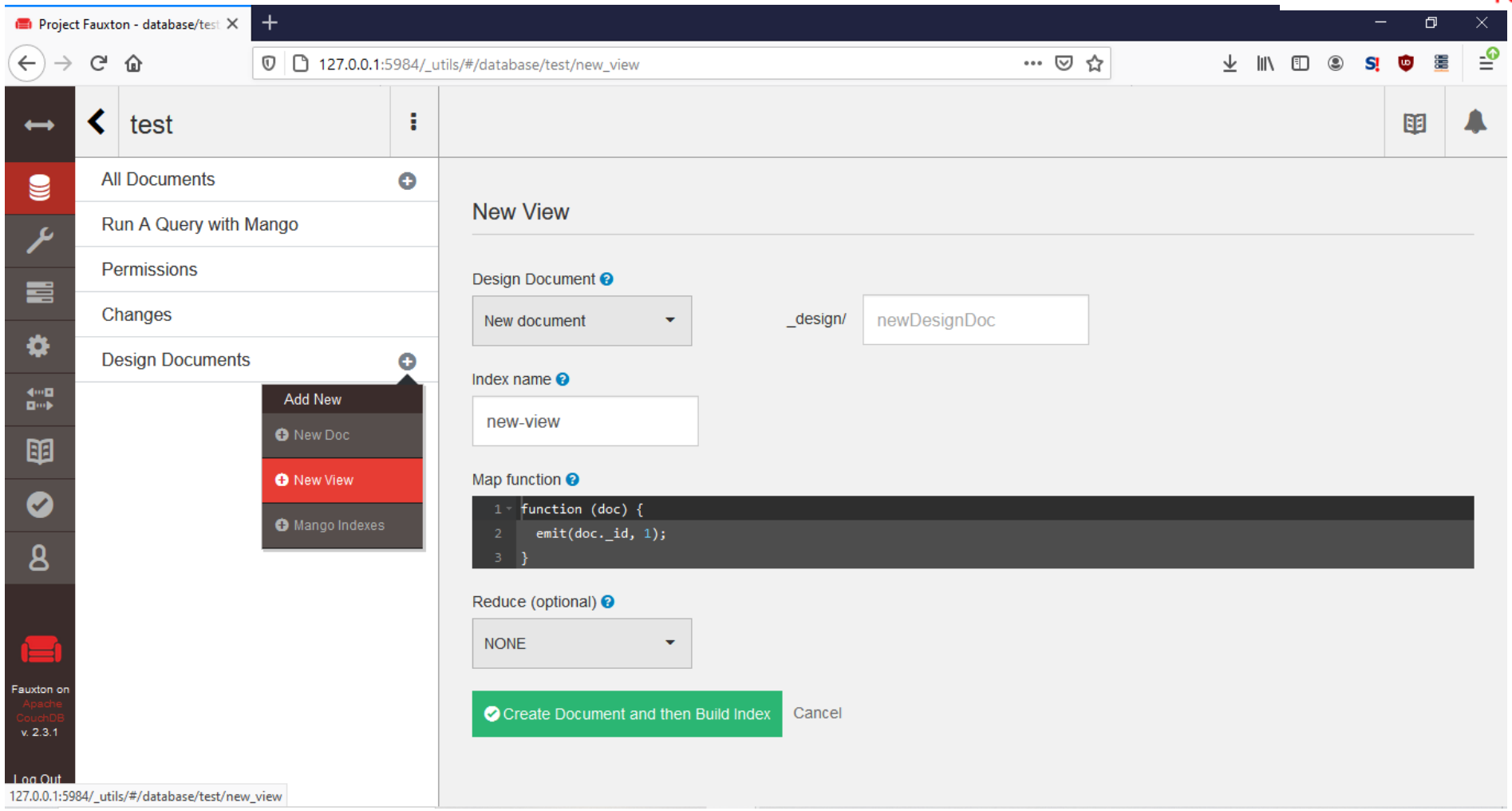


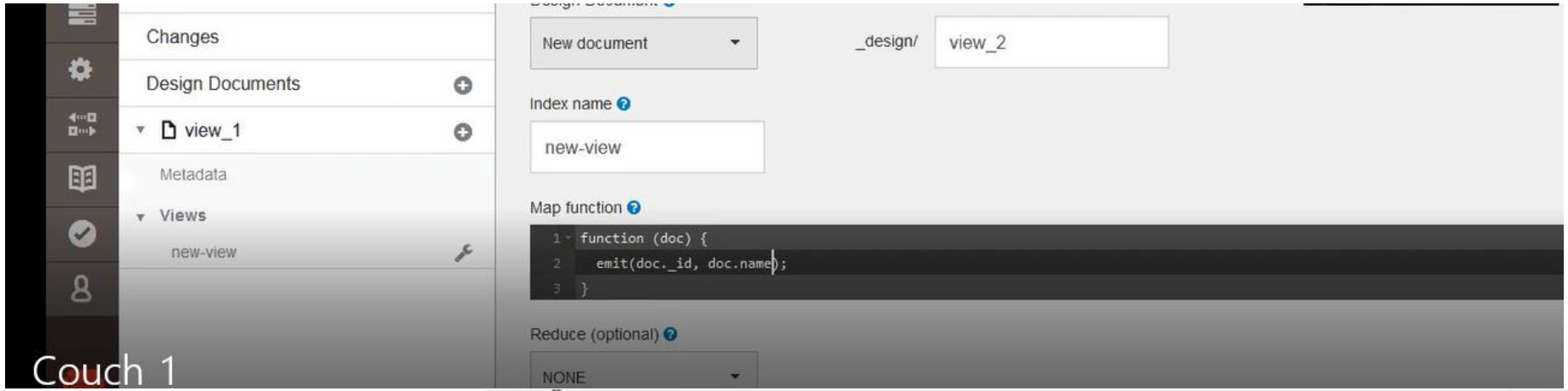
CouchDB creates a new database document, assigning it a new id.
You can edit the value of the id and can assign your own value in the form of a string.

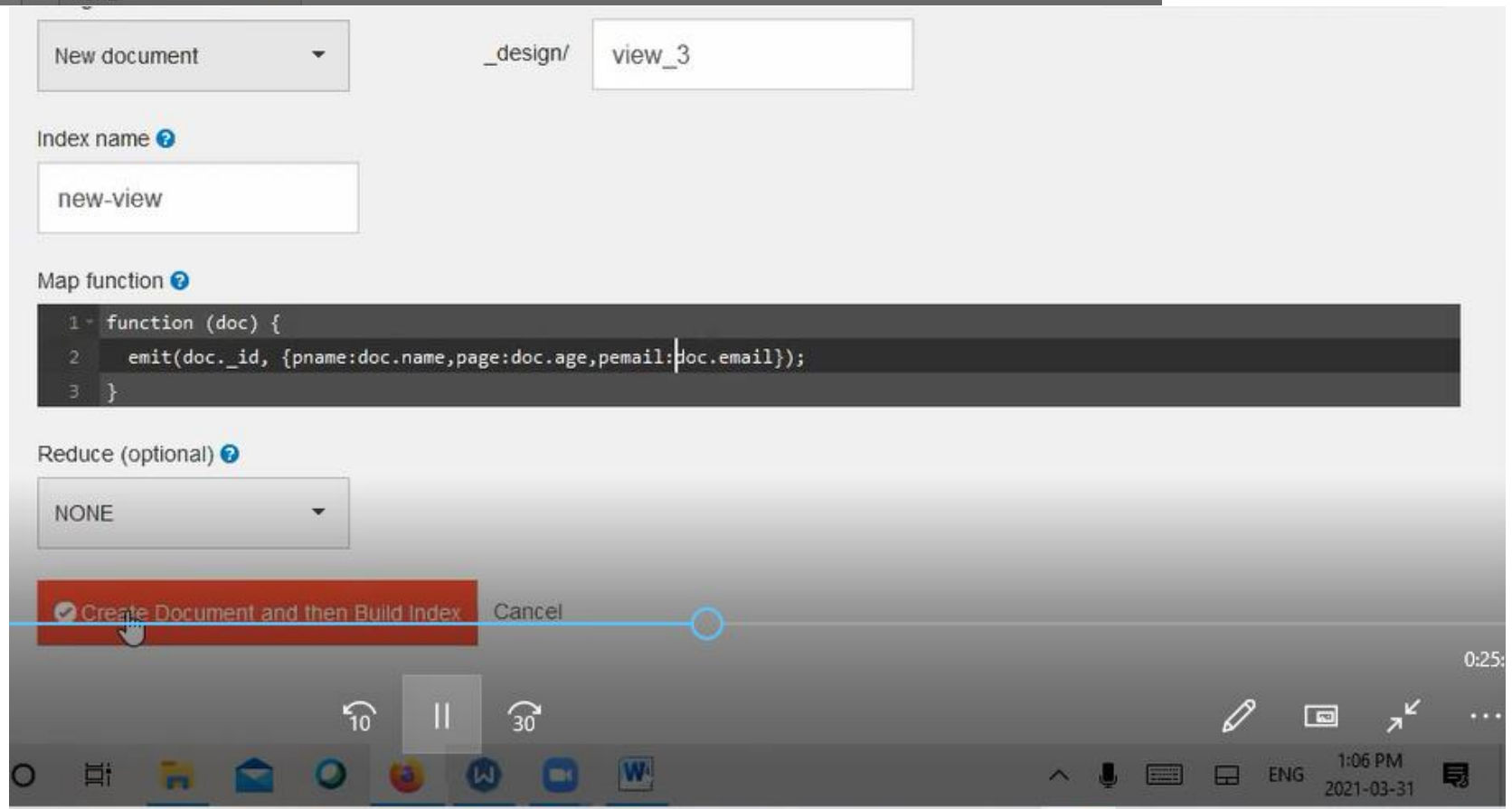
Futon: Updating a document



Creating View




Changes
Design Documents
view_1
Metadata
Views
new-view
New document
_design/
view_2
Index name
new-view
Map function
function (doc) {
emit(doc._id, doc.name);
}
Reduce (optional)
NONE
Couch 1


New document
_design/
view_3
Index name
new-view
Map function
function (doc) {
emit(doc._id, {pname:doc.name,page:doc.age,pemail:doc.email});
}
Reduce (optional)
NONE
Create Document and then Build Index
Cancel
0:25:
10
30
ENG
1:06 PM
2021-03-31

Command Line: Curl

- cURL utility is a way to communicate with CouchDB.
- The cURL utility is available in operating systems such as UNIX, Linux, Mac OS X and Windows. It is a command line utility using which user can access HTTP protocol straight away from the command line.
- To use Curl, goto command prompt , run it as administrator.
- Type `curl`
- You can access the homepage of the CouchDB by sending a GET request to the CouchDB instance installed.

```
curl http://127.0.0.1:5984/
```

Curl: Commands

- List databases : `curl -X GET http://127.0.0.1:5984/_all_dbs`
- Create database : `curl -X PUT http://127.0.0.1:5984/database_name`
- Verify database creation : `curl -X GET http://127.0.0.1:5984/_all_dbs`
- Deleting database : `curl -X DELETE http://127.0.0.1:5984/database name`
- Creating a document :

```
curl -X PUT http://127.0.0.1:5984/Nosql/"001" -d `
       { " Name " : " Sophia " ,
        " age " :" 30 " ,
       " Designation " : " Professor ",
        "address": {
              "street": "5 main street",
               "city": "Toronto",
              "country": "Canada"  }
        }'
```

Curl: Commands

- Updating Documents:
  - First of all, get the revision id of the document that is to be updated. You can find the **_rev** of the document in the document itself.
  - Use revision id _rev from the document to update the document. Here we are updating the name from “Raju” to “Sophia Sandhu”.

```
curl -X GET http://127.0.0.1:5984/test/001 {"_id":"001",
"_rev":"1-d75274b6a12d34cfc956b9a284f78f3b",
" Name ":" Raju "}
```

```
curl -X PUT http:/127.0.0.1:5984/test/001 -d"{
\"_rev\":\"1-d75274b6a12d34cfc956b9a284f78f3b\",
\" Name \":\“Sophia Sandhu\"}

{"ok":true,"id":"001","rev":"2-e28dfa41ac7493013449479db95444a5"}
```

Curl: Commands

- Deleting a Documents:
  - First of all, get the revision id of the document that is to be updated. You can find the **_rev** of the document in the document itself.
  - Use revision id _rev from the document to delete the document.

```
curl -X GET http://127.0.0.1:5984/test/001 {"_id":"001",
"_rev":"1-d75274b6a12d34cfc956b9a284f78f3b",
" Name ":" Raju "}
```

```
curl -X DELETE http:/127.0.0.1:5984/test/001?_rev=1 d75274b6a12d34cfc956b9a284f78f3b
```

Future Reading

- Books:
  - CouchDB: The Definitive Guide, 2010, O'Reilly
    Free online HTML version: http://books.couchdb.org/relax
  - CouchDB in Action, 2010, Manning
  - Beginning CouchDB, 2009, Apress

- Web:
  - Official site - http://apache.couchdb.org
  - Planet CouchDB - http://planet.couchdb.org

- Videos:
  - “CouchDB and me”, Damien Katz, http://www.infoq.com/presentations/katz-couchdb-and-me
  - Google Tech Talk, Chris Anderson, http://www.youtube.com/watch?v=ESDBM9-U804
  - SAPO Tech Talk, Jan Lehnardt, http://developers.blogs.sapo.pt/14565.html36

References

– Filipe David Manana ‘s Presentation.

Thanks