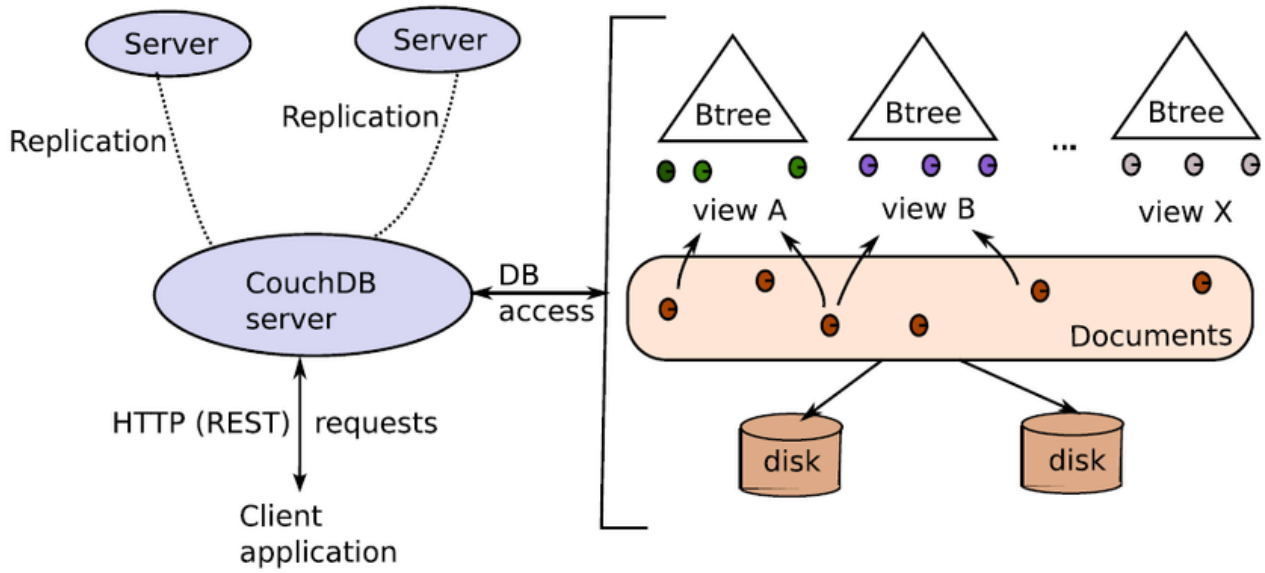
CouchDB

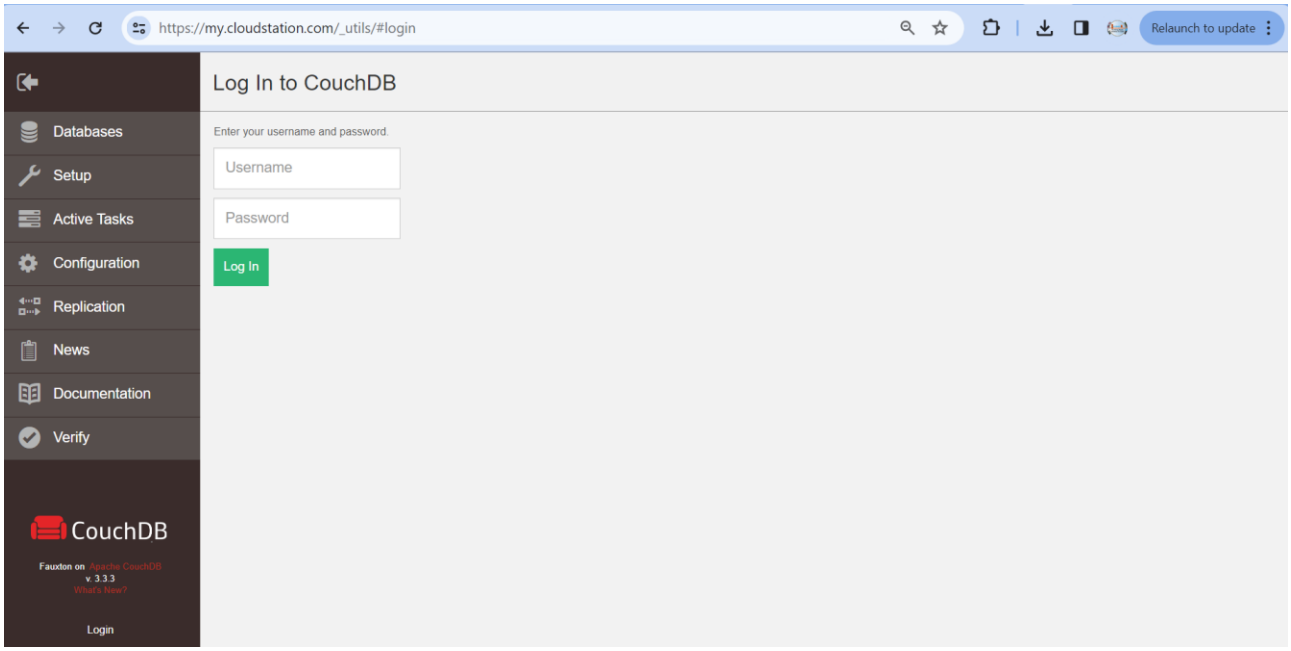
- **Overview**
  - It is an open-source NoSQL document database that collects and stores data in JSON-based document formats.
  - CouchDB: **C**luster **O**f **U**nreliable **C**ommodity **H**ardware
  - CouchDB was created by Damien Katz in April 2005, it was initially written in C++ and later became an Apache Software Foundation project in 2008.
  - Document Store for JSON documents.
  - Apache Foundation project
  - Can act as web-application back-end server.
  - Interactive browsing using Fauxton.
  - CouchDB (Use admin:admin to login)
  - UniProt database, Protein document
  - SQL vs CouchDB

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

- **CouchDB Architecture**
  - It is a 2-tier architecture.
  - CouchDB used HTTP as its main programming interface and JSON as data storage.
  - CouchDB is more suitable for client applications such as web applications.

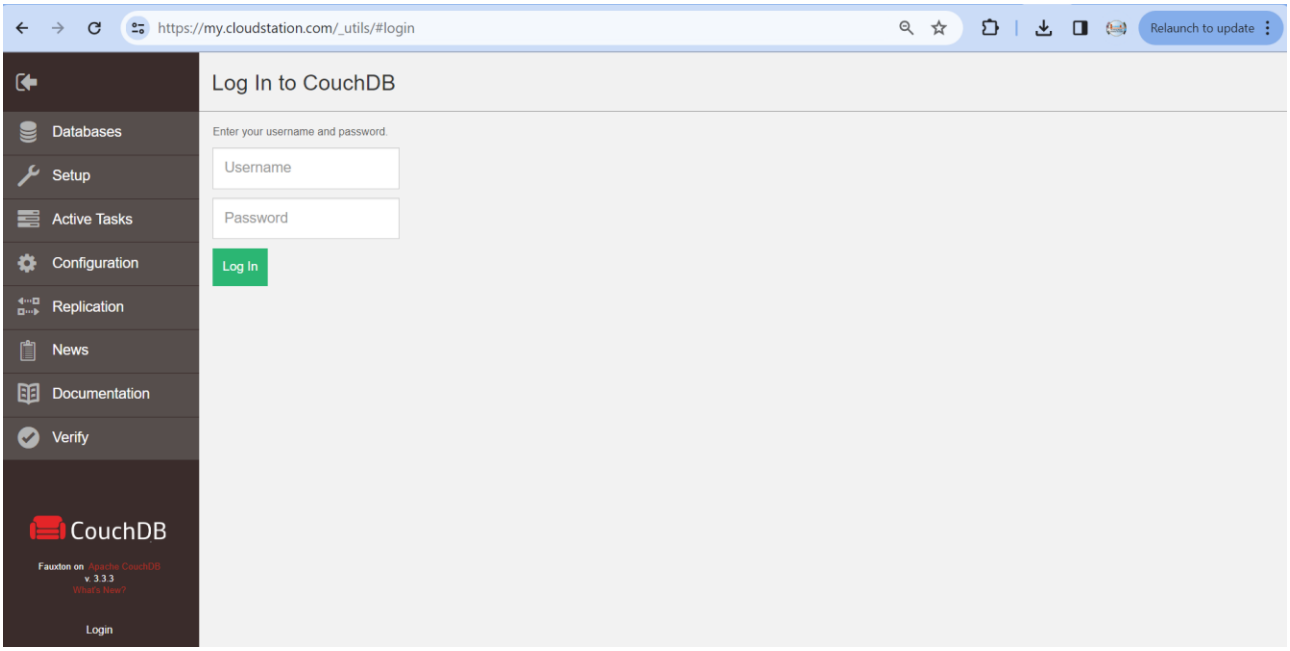


- **CRUD Operations in CouchDB**
  - Two methods to execute each CRUD operation:
    - Fauxton
    - Using curl API
  - Fauxton:
    - It is a web-based interface.
    - Easy to use.
    - It provides full access to all CouchDB features.
    - Create and Destroy databases.
    - Create, View and Edit Documents
    - Compose and run Map / Reduce Views
    - Replicate a Database

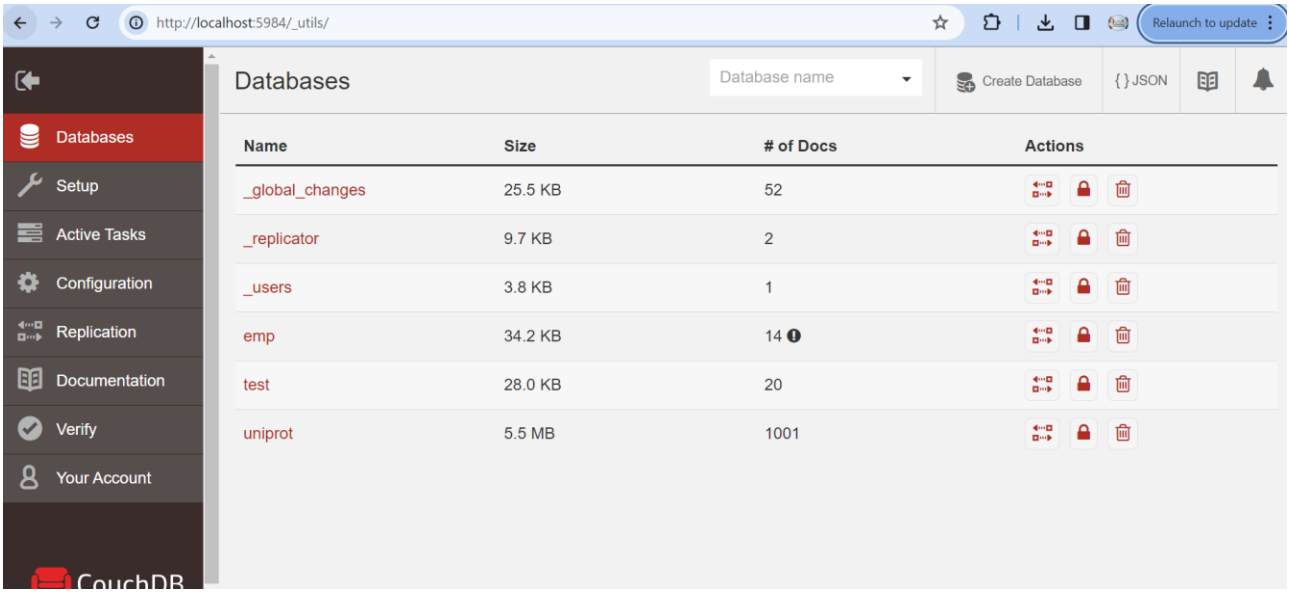


- Using curl API:
  - It is a command line tool available on Unix, Linux, Mac OS X, Windows, and many other platforms.
  - It provides easy access to the HTTP protocol (among others) directly from the command line.
  - It is used to interact with CouchDB over the HTTP REST API.

- **Fauxton**
  - Built-in admin interface



- Fauxton Features:



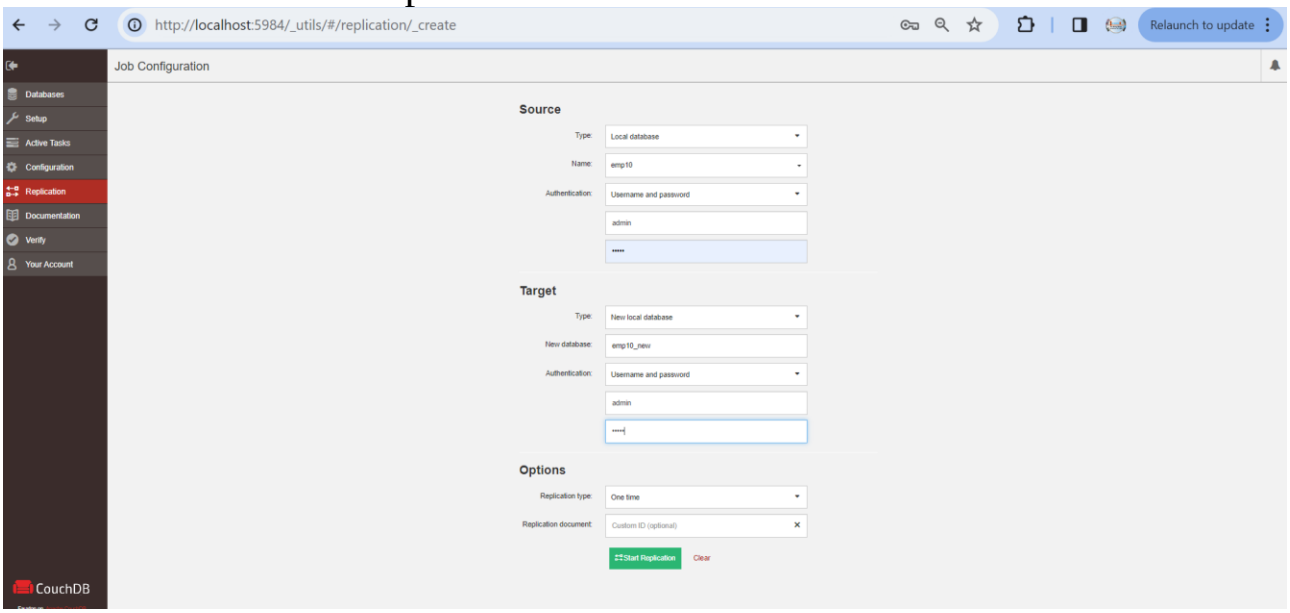
- Databases:
  - It shows a list of all your databases, their size, number of documents, and a list of actions.
- Setup:

- It is a wizard to set up and replicate configure CouchDB clusters or a single node.

- Active Tasks:
  - It displays a list of the running background tasks on the server:
    - view index building, compaction (reduce disk space usage by removing unused and old data from database or view index files), and replication.

- Config:
  - It is used to edit different configurable parameters. For more details on configuration, see Configuring CouchDB.

- Replication:
  - It allows you to replicate your system, enabling you to initiate replication between local and remote databases.



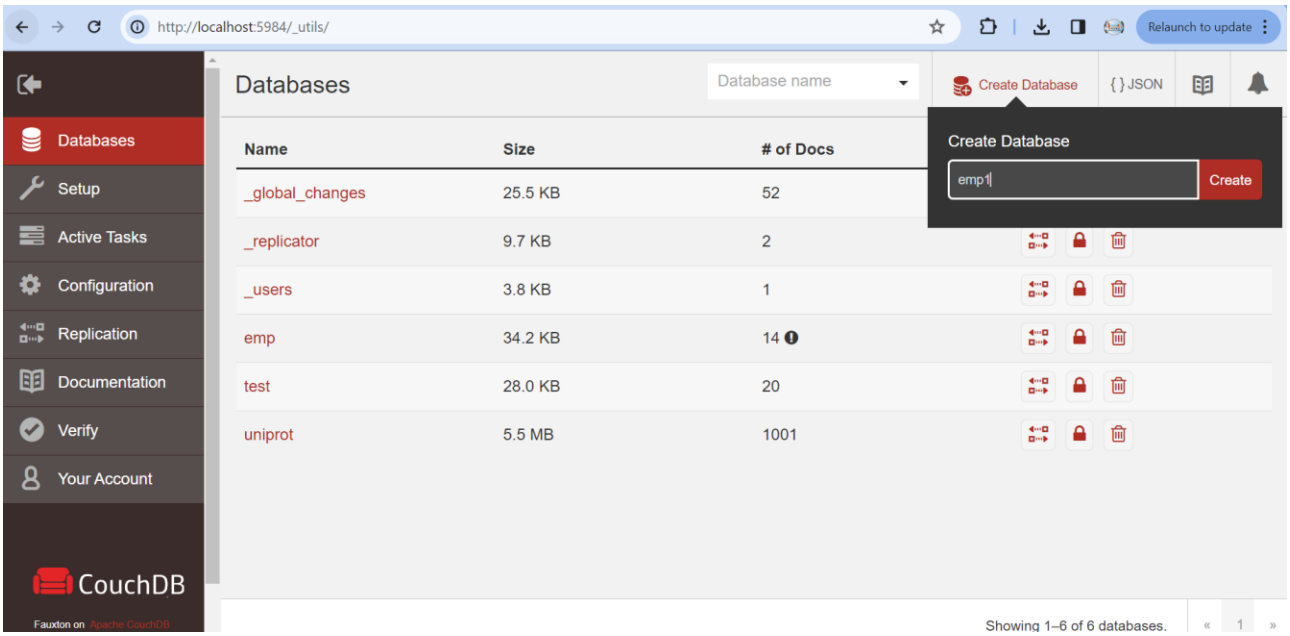
- Documentation:
  - It provides access to your local copy of the documentation.

- Login/User Management:
  - It allows you to change your password or add administrator to your CouchDB instance.

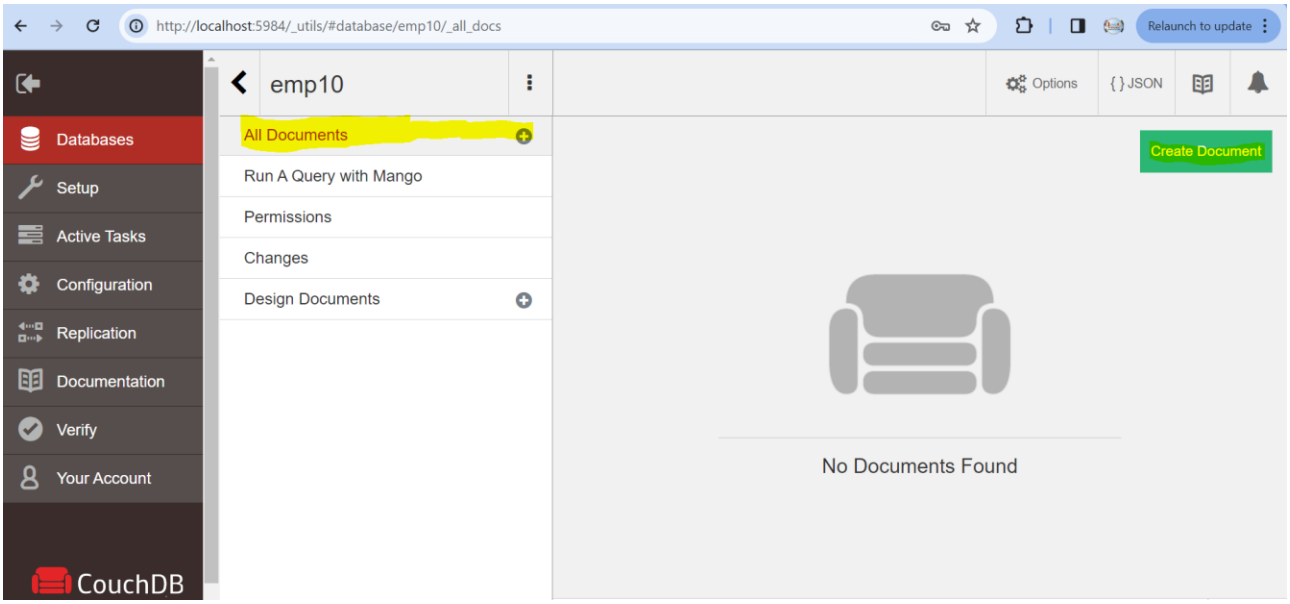
- Verify
  - It allows to verify if your CouchDB installation is correctly installed.

- Create:

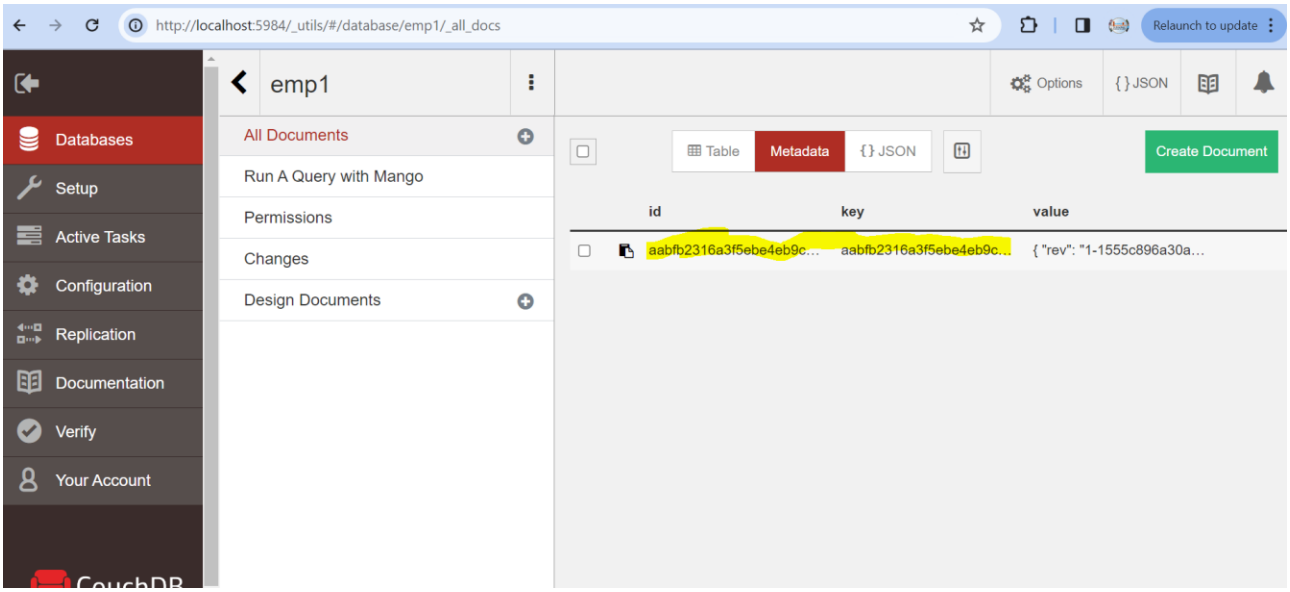
  - Create Databases:

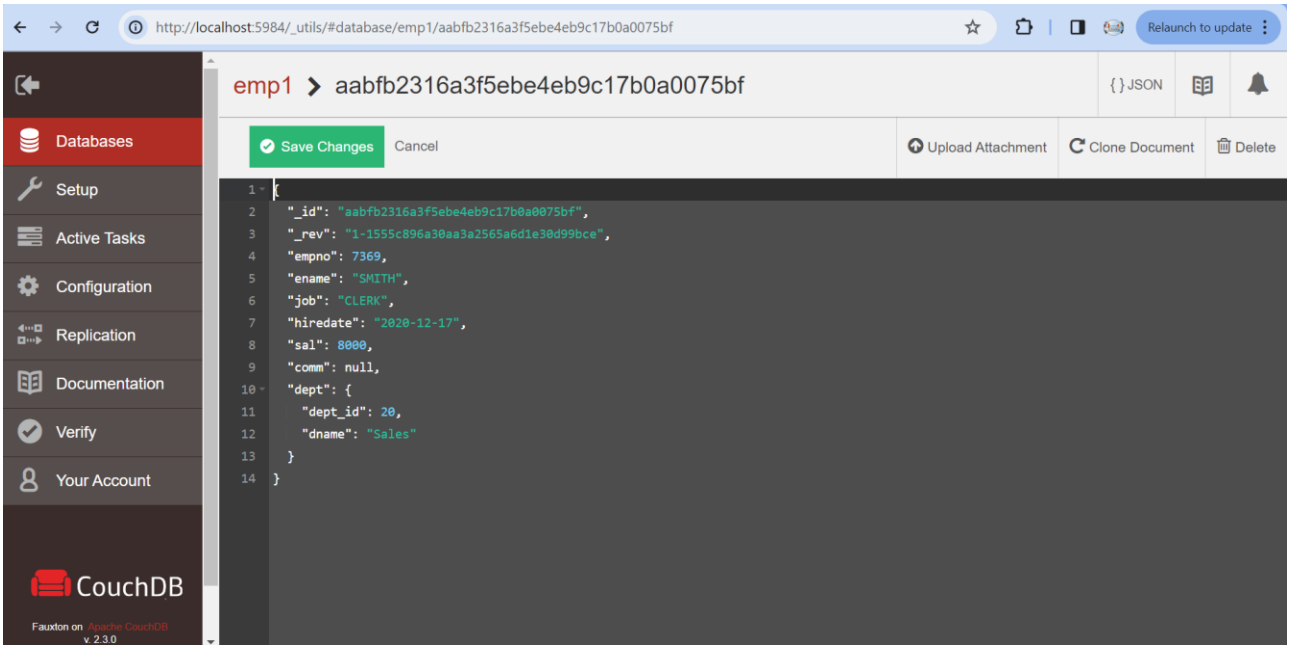


  - Create Documents:
    - Copy a document from emp database and insert it into emp1:



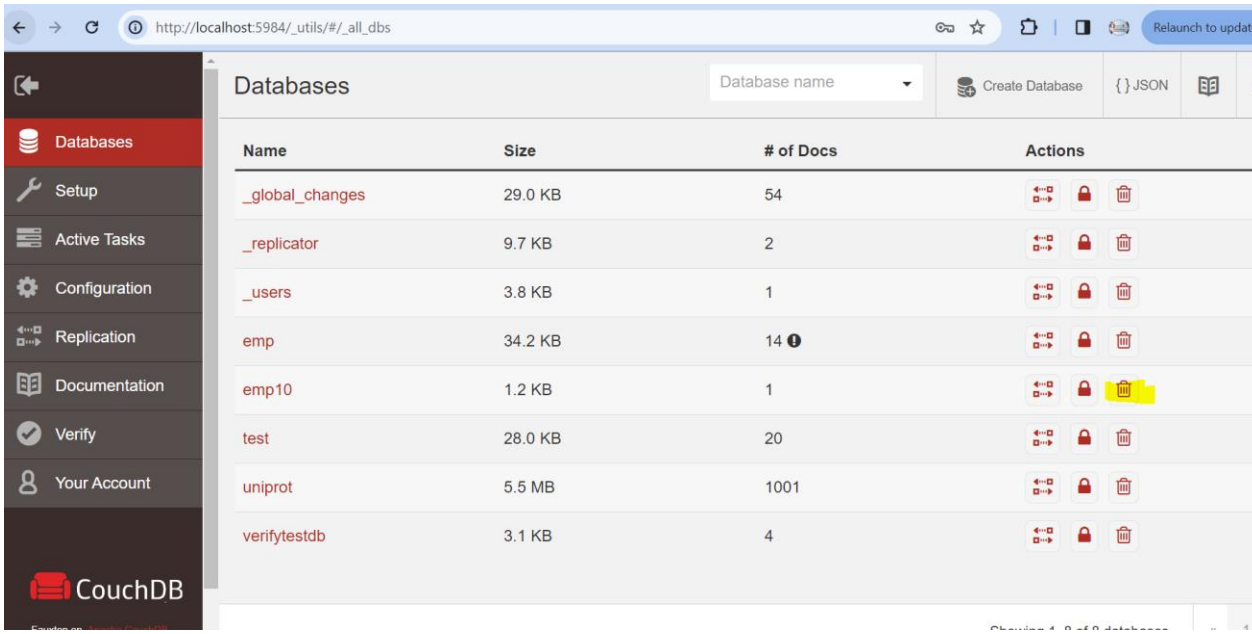
- Read Documents:



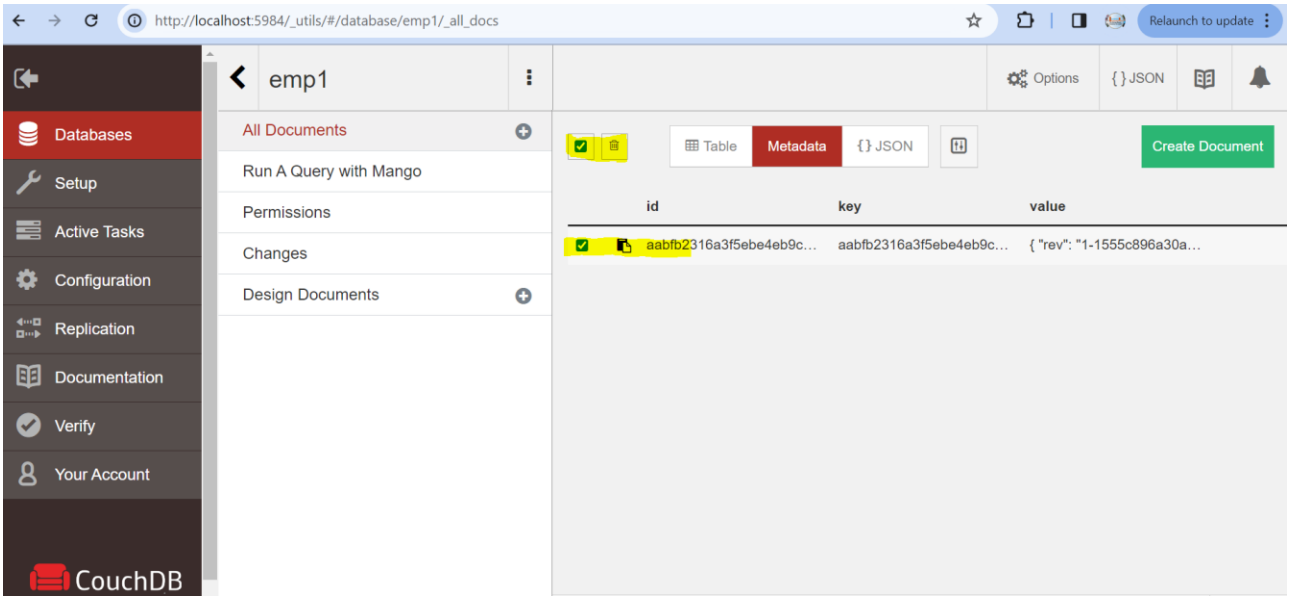


- Update Documents:
  - Read the document by clicking on the document id and update any fields.
  - Update uses the values of _id and _rev to update or delete documents.

- Delete:
  - Delete Databases:

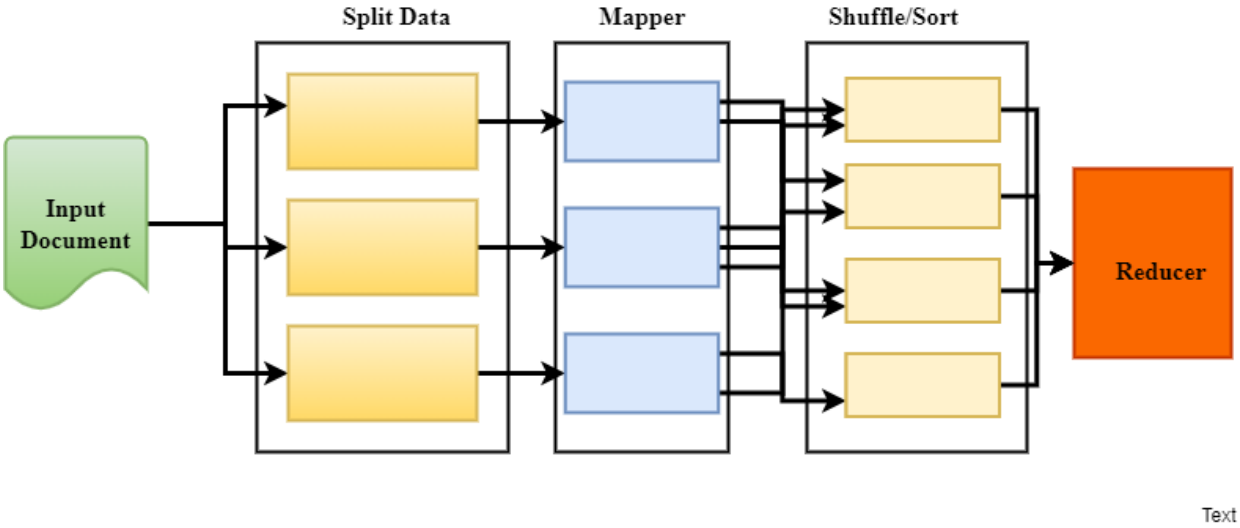


  - Delete Documents:



- **CouchDB Views**
  - They are the primary tool to query CouchDB.
  - Views are stored in design documents which are similar to any other CouchDB document.
  - Views create an index and store it in a B-tree.
  - They use the MapReduce programming paradigm.
  - MapReduce Programming Paradigm:

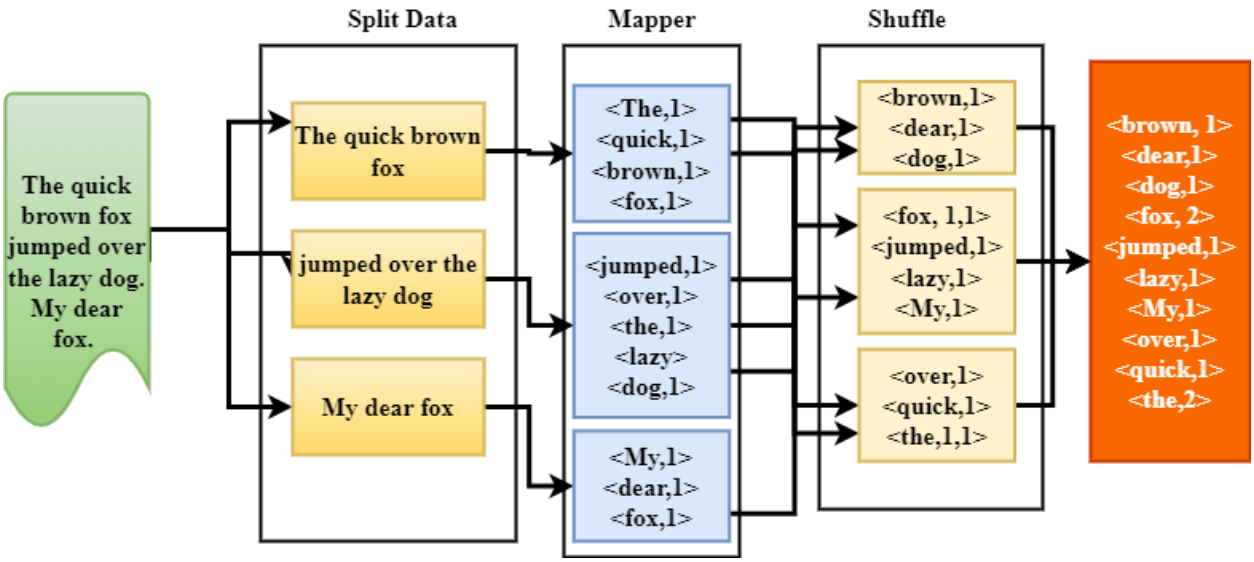
- Simple computational model for large scale parallel data processing
- It is mainly good for partitioned document store queries.
- MapReduce is a parallel and distributed programming model used to process big data.
- The entire MapReduce program can be fundamentally divided into three parts:



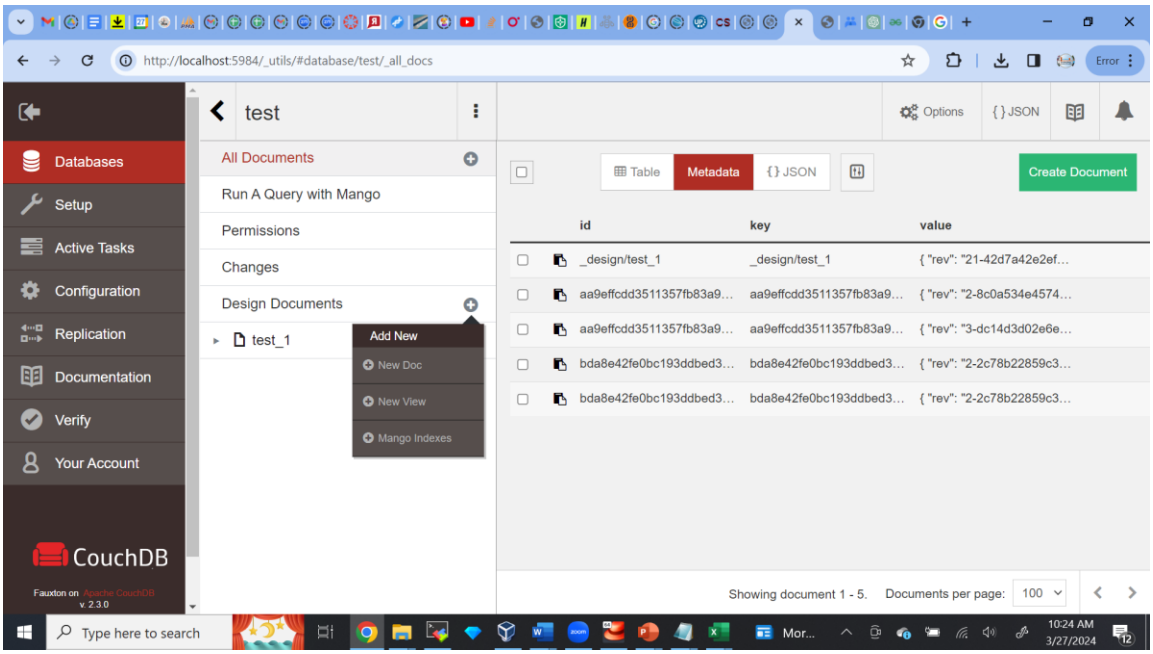
- Mapper:
  - The code to perform the mapping function.
- Reducer:
  - The code to perform the reducer logic.
- Shuffle/Combine/Sort:
  - Shuffle is a build in logic that transfers the map output from Mapper to a Reducer in MapReduce.
  - Data from the mapper are grouped by the key, split among reducers, and sorted by the key.
  - Every reducer obtains all values associated with the same key.

- Example: Word Count:

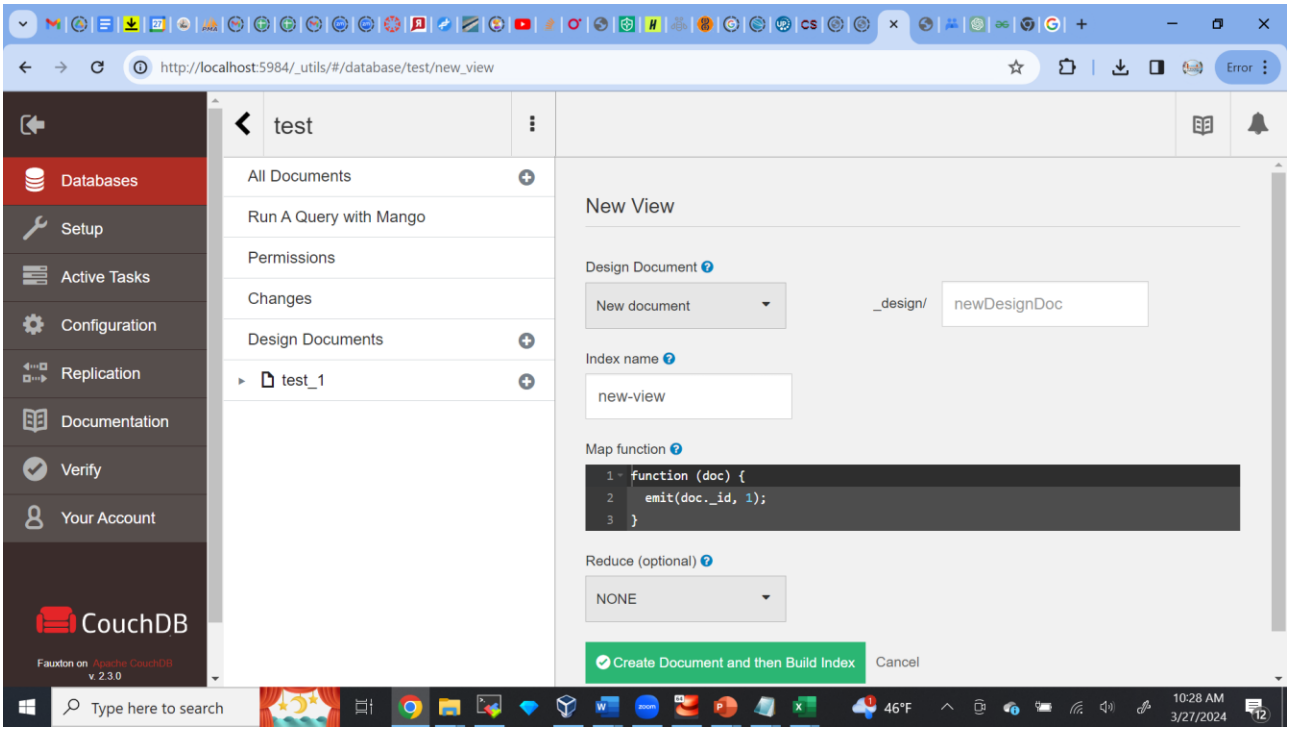
- Given a large dataset that cannot fit in main memory.
- List the count for each word in the dataset:
  - This is one Unix command line if everything fits in memory.
  - For large data, If the total distinct words fit in memory:
  - Use a hash function to map each keyword and keep count.
  - If the data cannot fit in the memory and the total distinct words fits in the memory



- Creating View:
  - Open the database.
  - Click on the "+" next to design document and select "New View."



- You can add to an existing design document or create a new one.
- Creating a new design document
- Examples:



- Viewing View Results using URL

  http://localhost:5984/uniprot/_design/test_view/_view/new_view_source

- Query Parameters
  - A list of options can be found at: https://docs.couchdb.org/en/stable/api/ddoc/views.html
  - doc_includes (Boolean):
    - To view the content of each document, use ?doc_includes=true.
  - group (Boolean):
    - Group the results using the reduce function to a group or single row.
    - descending (Boolean):
      - Return the documents in descending order by key.
  - Etc.

- **Practice**
  - Query Emp Database
    - Number of employees by department (empByDept)
      - http://localhost:5984/emp/_design/emp_views/_view/empByDept
      - http://localhost:5984/emp/_design/emp_views/_view/empByDept?group=true
    - List the cells of all employees – Solution 1
      - http://localhost:5984/emp/_design/emp_views/_view/emp_cells
      - http://localhost:5984/emp/_design/emp_views/_view/emp_cells?include_docs=true
    - List the cells of all employees – Solution 2
      - http://localhost:5984/emp/_design/emp_views/_view/emp_cells_2
      - http://localhost:5984/emp/_design/emp_views/_view/emp_cells_2?include_docs=true
    - Statistics about the salary of all employees:
      - http://localhost:5984/emp/_design/emp_views/_view/highest_sal
    - How many employees are in the "Sales" department?
      - http://localhost:5984/emp/_design/emp_views/_view/emp_in_sales
    - List the employees in the "Sales" department.
      - http://localhost:5984/emp/_design/emp_views/_view/list_sales_emp
      - To view the information of each employee: http://localhost:5984/emp/_design/emp_views/_view/emp_in_sales_list?include_docs=true
  - UniProt:
    - Get the number of accession numbers:

- http://localhost:5984/uniprot/_design/test_view/_view/get_accessions

- List of types of organizations:
  - http://localhost:5984/uniprot/_design/test_view/_view/get_organism_types?group=true
- Number of proteins without genes
  - http://localhost:5984/uniprot/_design/test_view/_view/proteins_no_genes_num
- List of proteins without genes
  - http://localhost:5984/uniprot/_design/test_view/_view/proteins_no_genes_list?include_docs=true
- Get the list of titles of a protein:
  - http://localhost:5984/uniprot/_design/test_view/_view/get_num_references
- Get the list of types of names of an organism:
  - http://localhost:5984/uniprot/_design/test_view/_view/get_organism_types
- Get sequence length:
  - http://localhost:5984/uniprot/_design/test_view/_view/length_sequence

- **Using Curl APIs**
  - **Curl Command - Server API**
    - Check if CouchDB is available:
      - Command to check if CouchDB is working at all using terminal:
        curl http://127.0.0.1:5984/
      - Response:
        {"CouchDB":"Welcome","version":"2.3.0","git_sha":"07ea0c7","uuid":"8a9f0bec88e99619146f15d90d0e30d3","features":["pluggable-storage-engines","scheduler"],"vendor":{"name":"The Apache Software Foundation"}}
      - Command to check if CouchDB is working at all using browser:
        http://127.0.0.1:5984/
      - Response:


```
{
 "CouchDB": "Welcome",
 "version": "2.3.0",
 "git_sha": "07ea0c7",
 "uuid": "8a9f0bec88e99619146f15d90d0e30d3",
 "features": [
  "pluggable-storage-engines",
  "scheduler"
 ],
 "vendor": {
  "name": "The Apache Software Foundation"
 }
}
```


  - **Curl Command - Database API**
    - List of databases:
      - Command to get a list of Databases using terminal:
        curl -X GET http://127.0.0.1:5984/_all_dbs

- Response:
  ["_global_changes","_replicator","_users","emp","emp1","test","uniprot"]
- Command to get a list of Databases using browser:
  http://127.0.0.1:5984/_all_dbs
- Response:

```
[
  "_global_changes",
  "_replicator",
  "_users",
  "emp",
  "emp1",
  "test",
  "uniprot"
]
```

- **Command to create a Database**:
  - Command to create a database:
    - Using terminal:
      curl -X PUT http://127.0.0.1:5984/DB_name
    - Response:
      {"**error**":"unauthorized","reason":"You are not a server admin."}
    - Include credentials:
      curl -X PUT http://admin:admin@127.0.0.1:5984/emp3
    - Respone:
      {"ok":true}

    - Curl's -v option:
      curl -vX PUT http://admin:admin@127.0.0.1:5984/emp5
    - Response:

```
* About to connect() to 127.0.0.1 port 5984 (#0)
*   Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 5984 (#0)
```

```
* Server auth using Basic with user 'admin'
> PUT /emp5 HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 127.0.0.1:5984
> Accept: */*
>
< HTTP/1.1 201 Created
< Cache-Control: must-revalidate.
< Content-Length: 12
< Content-Type: application/Json
< Date: Sat, 30 Mar 2024 21:27:27 GMT
< Location: http://127.0.0.1:5984/emp5
< Server: CouchDB/2.3.0 (Erlang OTP/20)
< X-Couch-Request-ID: 5267856604
< X-CouchDB-Body-Time: 0
<
{"ok":true}
* Connection #0 to host 127.0.0.1 left intact
```

- Check if the database is created:
  http://127.0.0.1:5984/_all_dbs

```
[
  "_global_changes",
  "_replicator",
  "_users",
  "emp",
  "emp1",
  "emp3",
  "emp4",
  "emp5",
  "test",
  "uniprot"
]
```

- Check the database using the browser:
  http://127.0.0.1:5984/emp5


```
{
  "db_name": "emp5",
  "purge_seq": "0-
g1AAAAFTeJzLYWBg4MhgTmEQTM4vTc5ISXI
wNDLXMwBCwxygFFMeC5BkOACk_v__fz8rkY
Gg2gcQtf-
JUbsAonY_frVJDkAyKZ5IMxsgZs4nYGYCyMx6
gmYmKYDU2RNUl8iQJA9RlAUAPoRejg",
  "update_seq": "0-
g1AAAAFTeJzLYWBg4MhgTmEQTM4vTc5ISXI
wNDLXMwBCwxygFFMiQ5L8____sxIZ8ChKUg
CSSfaE1TmA1MUTVpcAUldPUF0eC5BkaABSQ
KXziVG7AKJ2PzFqD0DU3idG7QOIWpB7swBegl
6O",
  "sizes": {
    "file": 33992,
    "external": 0,
    "active": 0
  },
  "other": {
    "data_size": 0
  },
  "doc_del_count": 0,
  "doc_count": 0,
  "disk_size": 33992,
  "disk_format_version": 7,
  "data_size": 0,
  "compact_running": false,
  "cluster": {
    "q": 8,
    "n": 1,
    "w": 1,
    "r": 1
```

```
        },
  "instance_start_time": "0"
}
```

- Create a database and importing Json file:
    1. First create the database: emp6
       http://admin:admin@127.0.0.1:5984/emp6
    2. Import the file:
       curl -X POST
       http://admin:admin@localhost:5984/emp6/_bulk_docs
       -H "Content-Type: application/Json" -d @emp.json

- **Reading Documents:**
    - We need to know the document _id
    - Using terminal:

      curl -X GET
      http://admin:admin@localhost:5984/test/aa9effcdd3511357fb83a9816e000d89

      You can omit "-X GET"

      curl http://admin:admin@localhost:5984/test/aa9effcdd3511357fb83a9816e000d89

      {"_id":"bda8e42fe0bc193ddbed360c4c0469cf","_rev":"1-cabd8464c563f1210f2d3a55dd7c5178","empno":7369,"ename":"SMITH","job":"CLERK","hiredate":"2020-12-17","sal":800,"comm":null,"dept":{"dept_id":20,"dname":"Sales"}}

    - Using web browser:
        - URL:
          http://localhost:5984/emp/bda8e42fe0bc193ddbed360c4c0469cf

- Respone:


```
{
 "_id": "bda8e42fe0bc193ddbed360c4c0469cf",
 "_rev": "1-cabd8464c563f1210f2d3a55dd7c5178",
 "empno": 7369,
 "ename": "SMITH",
 "job": "CLERK",
 "hiredate": "2020-12-17",
 "sal": 800,
 "comm": null,
 "dept": {
  "dept_id": 20,
  "dname": "Sales"
 }
}
```


- Get All docs:
  - Using terminal:
    Curl http://localhost:5984/emp/_all_docs
  - Using url:
    http://localhost:5984/emp/_all_docs

- **Update:**
  - To update a document, you must provide both the _id and _rev for the document.
  - You cannot update a single field; the entire document will be replaced.
  - Example:
    - Using Curl:

```
curl -
X PUT http://127.0.0.1:5984/emp/bda8e42fe0bc193ddbed360c4c0469cf/  -H 'Content-Type: application/json' -d'{"sal": "8000",  "_rev":"1-cabd8464c563f1210f2d3a55dd7c5178"}'
```

- Response:

{"ok":true,"id":"bda8e42fe0bc193ddbed360c4c0469cf","rev":"2-8b2c0ea0f4f11479d5e4aaa1fb0c6d36"}

- Check the document:

http://localhost:5984/emp/bda8e42fe0bc193ddbed360c4c0469cf

- Response:

```
{
 "_id": "bda8e42fe0bc193ddbed360c4c0469cf",
 "_rev": "2-8b2c0ea0f4f11479d5e4aaa1fb0c6d36",
 "sal": "8000"
}
```

Note everything is gone!!!!!!

- Here is the complete command:

```
curl -X PUT
http://127.0.0.1:5984/emp/bda8e42fe0bc193ddbed360c4c0469cf/  -H 'Content-Type: application/json' -d'{
"_id": "bda8e42fe0bc193ddbed360c4c0469cf",
 "_rev": "5-03acfe1b0d585f484e559e3f8b913d4c",
 "empno": 7369,
 "ename": "SMITH",
 "job": "CLERK",
 "hiredate": "2020-12-17",
 "sal": 8000,
 "comm": null,
 "dept": {
  "dept_id": 20,
  "dname": "Sales"
 }}'
```

- Response:

{"ok":true,"id":"bda8e42fe0bc193ddbed360c4c0469cf","rev":"6-ad6926c0aec2352567217ca05d106af5"}

- Note:
  - You will get an error if you try to run the same query without changing the _rev.

- Delete:
  - Delete a database:

    curl -X DELETE http://admin:admin@localhost:5984/new_db_1

    {"ok":true}

  - Delete a document in a database:

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