

CouchDB

• Overview	2
• CouchDB Architecture.....	2
• CRUD Operations in CouchDB	3
• Fauxton	4
• CouchDB Views.....	10
• Practice	15
• Using Curl APIs	17

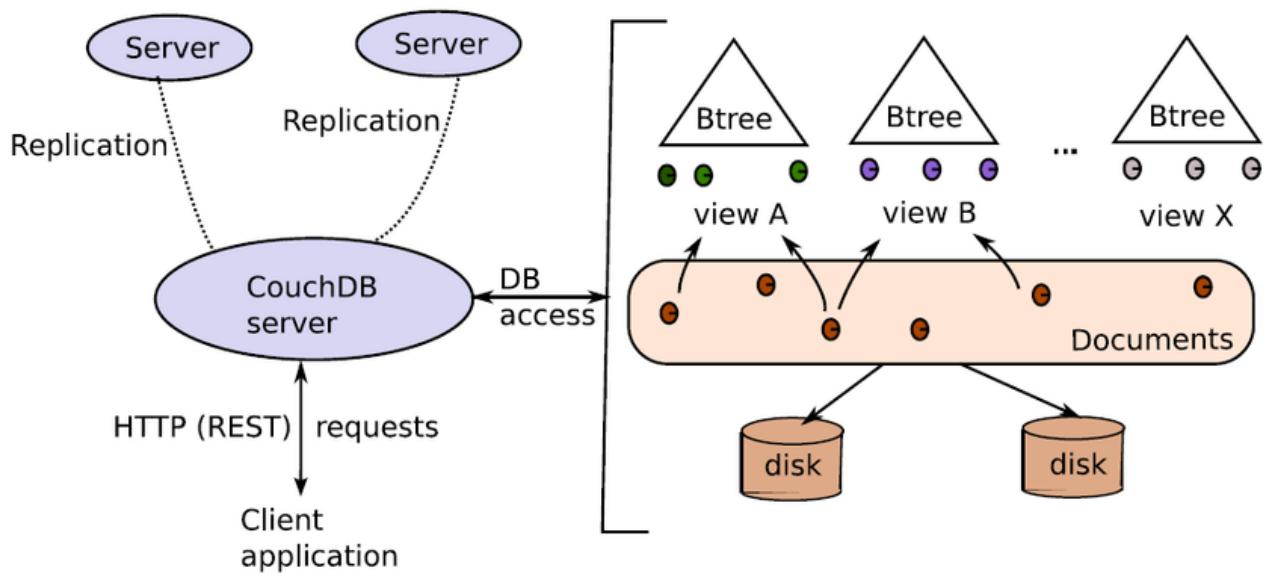
- **Overview**

- 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

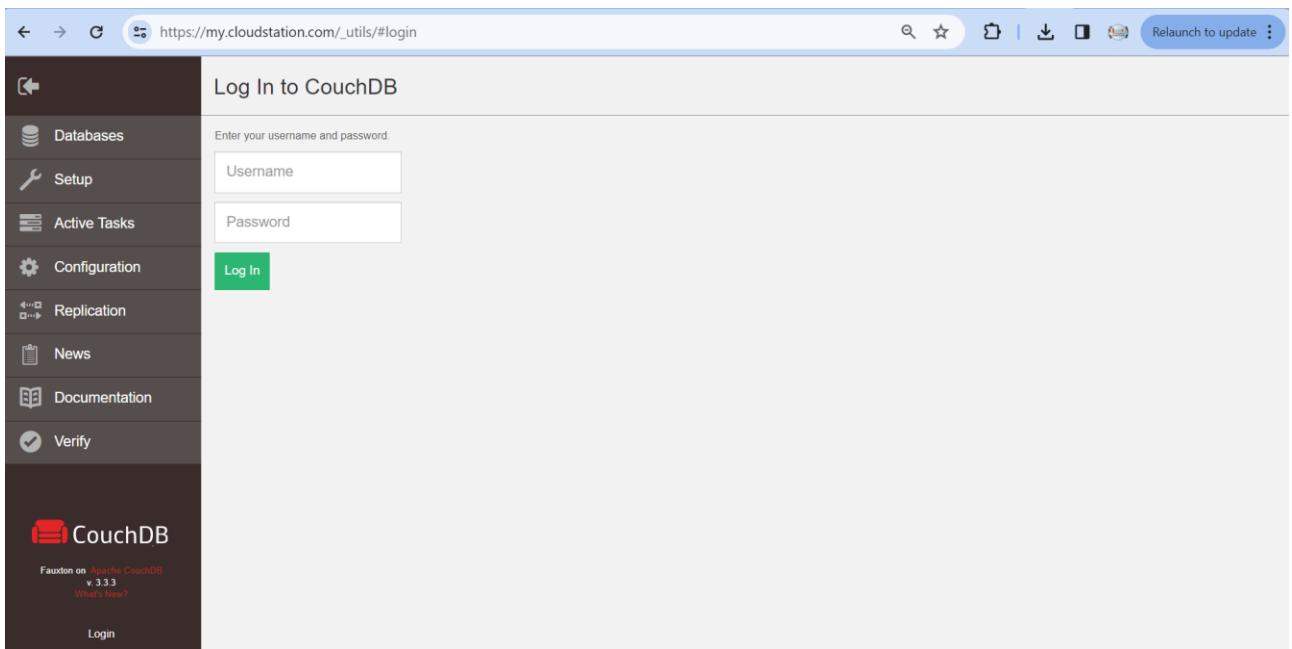
SQL	CouchDB
Relational	Non-Relational
Tables	Documents with types
Rows and Columns	Document Fields
SQL Query Engine	Map / Reduce Engine

- **CouchDB Architecture**

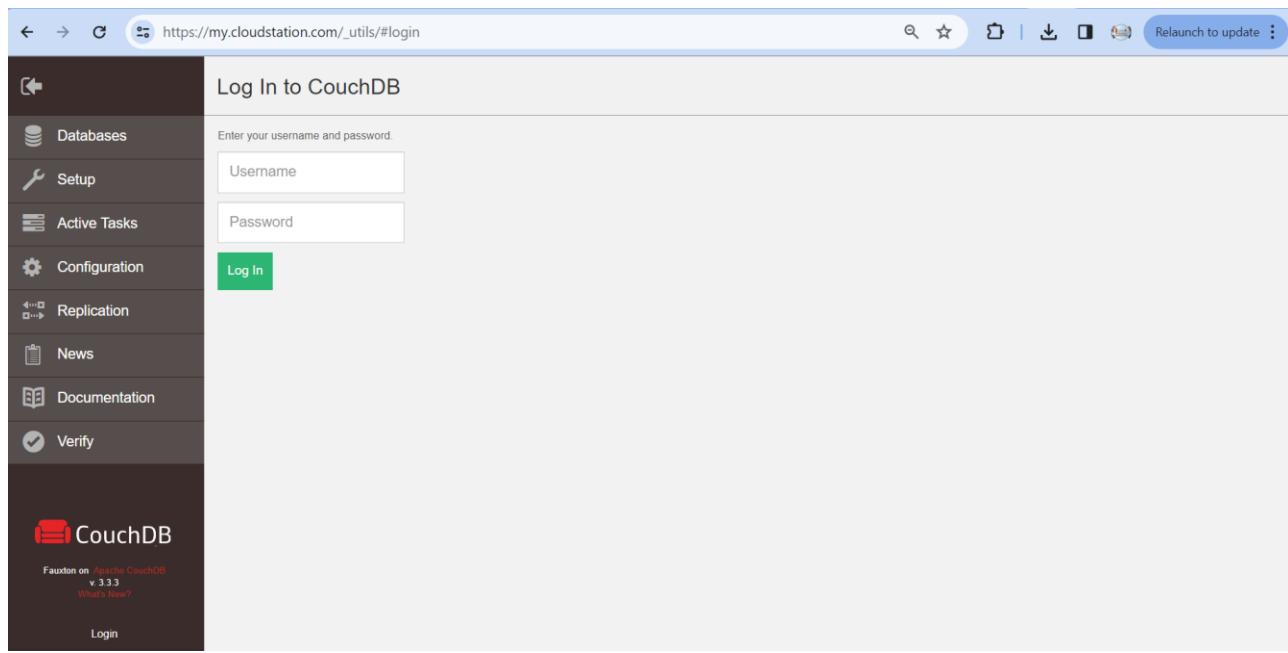
- It is a 2-tier architecture.
- CouchDB used HTTP as its main programming interface and JSON as data storage.
- CouchDB is more suitable for client applications such as web applications.



- **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:

The screenshot shows the Fauxton Databases list at http://localhost:5984/_utils/. The left sidebar includes 'Databases' (which is red), 'Setup', 'Active Tasks', 'Configuration', 'Replication', 'Documentation', 'Verify', and 'Your Account'. The main table lists databases with columns for Name, Size, # of Docs, and Actions. The 'Actions' column contains icons for replication, locking, and deleting.

Name	Size	# of Docs	Actions
_global_changes	25.5 KB	52	
_replicator	9.7 KB	2	
_users	3.8 KB	1	
emp	34.2 KB	14	
test	28.0 KB	20	
uniprot	5.5 MB	1001	

- 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.

Job Configuration

Source

Type: Local database
Name: emp10
Authentication: Username and password
admin

Target

Type: New local database
New database: emp10_new
Authentication: Username and password
admin

Options

Replication type: One time
Replication document: Custom ID (optional)

- 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:

Name	Size	# of Docs
_global_changes	25.5 KB	52
_replicator	9.7 KB	2
_users	3.8 KB	1
emp	34.2 KB	14 1
test	28.0 KB	20
uniprot	5.5 MB	1001

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

All Documents

Create Document

No Documents Found

○ Read Documents:

id	key	value
aabfb2316a3f5ebe4eb9c...	aabfb2316a3f5ebe4eb9c...	{ "rev": "1-1555c896a30a..."}

Create Document

The screenshot shows the Fauxton interface for Apache CouchDB. On the left is a sidebar with icons for Databases, Setup, Active Tasks, Configuration, Replication, Documentation, Verify, Your Account, and CouchDB. The main area shows a document in the 'emp1' database with the ID 'aabfb2316a3f5ebe4eb9c17b0a0075bf'. The document's JSON content is displayed:

```

1 [
2   "_id": "aabfb2316a3f5ebe4eb9c17b0a0075bf",
3   "_rev": "1-1555c896a30aa3a2565a6d1e30d99bce",
4   "empno": 7369,
5   "ename": "SMITH",
6   "job": "CLERK",
7   "hiredate": "2020-12-17",
8   "sal": 8000,
9   "comm": null,
10  "dept": {
11    "dept_id": 20,
12    "dname": "Sales"
13  }
14 ]

```

Buttons at the top right include 'Save Changes', 'Cancel', 'Upload Attachment', 'Clone Document', and 'Delete'.

- 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:

The screenshot shows the Fauxton interface displaying a list of databases. The sidebar is identical to the previous screenshot. The main area lists databases with their names, sizes, and document counts. The 'emp' database has 14 documents, while 'verifytestdb' has 4. Actions for each database include viewing, locking, and deleting, with a yellow trash can icon indicating it can be deleted.

Name	Size	# of Docs	Actions
<code>_global_changes</code>	29.0 KB	54	
<code>_replicator</code>	9.7 KB	2	
<code>_users</code>	3.8 KB	1	
<code>emp</code>	34.2 KB	14	
<code>emp10</code>	1.2 KB	1	
<code>test</code>	28.0 KB	20	
<code>uniprot</code>	5.5 MB	1001	
<code>verifytestdb</code>	3.1 KB	4	

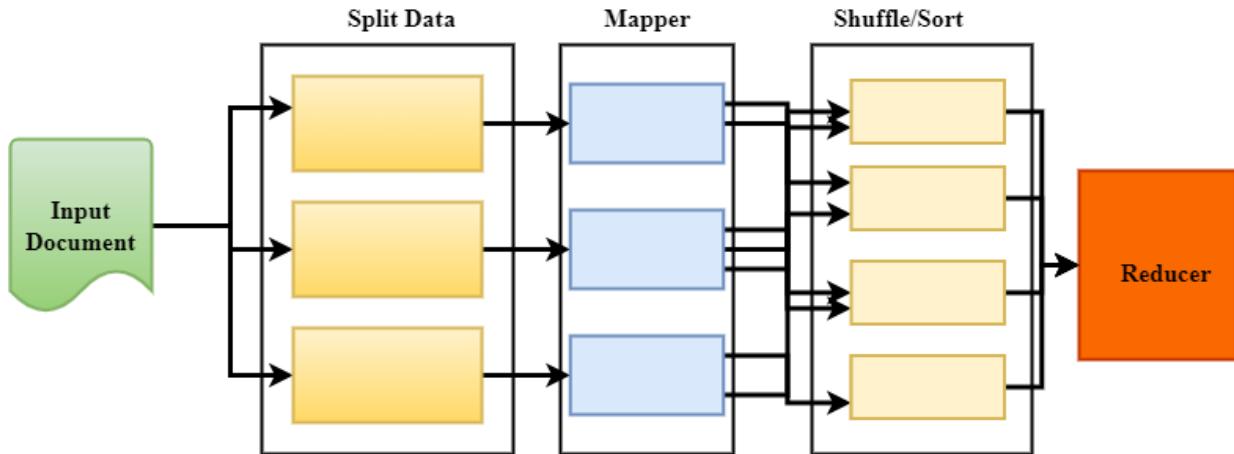
- Delete Documents:

The screenshot shows the CouchDB management interface at http://localhost:5984/_utils/#/database/emp1/_all_docs. The left sidebar has a red header 'Databases' containing 'emp1'. Other options include 'Setup', 'Active Tasks', 'Configuration', 'Replication', 'Documentation', 'Verify', 'Your Account', and a 'CouchDB' logo. The main area shows 'emp1' with tabs for 'All Documents', 'Run A Query with Mango', 'Permissions', 'Changes', and 'Design Documents'. The 'All Documents' tab is selected, showing a table with columns 'id', 'key', and 'value'. One document is listed with id 'aabfb2316a3f5ebe4eb9c...', key 'aabfb2316a3f5ebe4eb9c...', and value '{ "rev": "1-1555c896a30a...}'. There are buttons for 'Table', 'Metadata', 'JSON', and 'Create Document'.

- **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:

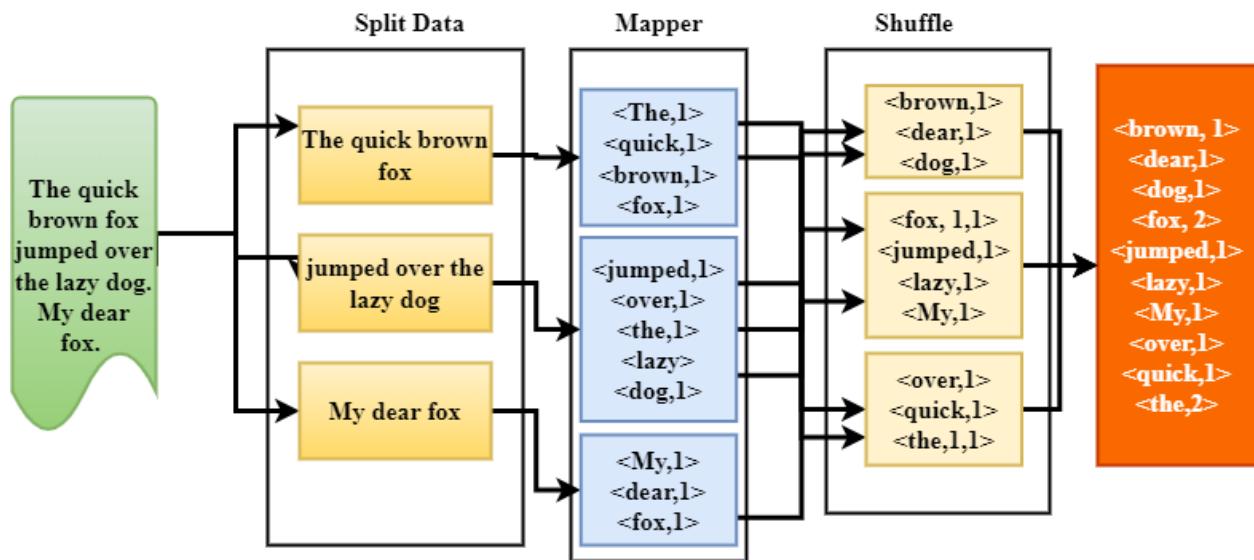
- Simple computational model for large scale parallel data processing
- It is mainly good for partitioned document store queries.
- MapReduce is a parallel and distributed programming model used to process big data.
- The entire MapReduce program can be fundamentally divided into three parts:



Text

- 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.”

The screenshot shows the Fauxton interface for the 'test' database. On the left, there's a sidebar with various navigation links like Databases, Setup, Active Tasks, Configuration, Replication, Documentation, Verify, and Your Account. The main area shows a table of documents with columns for id, key, and value. A context menu is open over a document named 'test_1', with options to 'Add New', 'New Doc', 'New View', and 'Mango Indexes'.

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

The screenshot shows the Fauxton interface for creating a new view. The 'test' database is selected. In the 'Design Documents' section, a new view is being created under 'test_1'. The 'New View' dialog is open, showing fields for 'Design Document' (set to '_design/newDesignDoc'), 'Index name' (set to 'new-view'), and 'Map function' (containing the code: 'function (doc) { emit(doc._id, 1); }'). Below these, there's a 'Reduce (optional)' field set to 'NONE'. At the bottom of the dialog is a green 'Create Document and then Build Index' button.

- 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_acessions
- 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_numb_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](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`
    - Response:  
`{"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](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-
g1AAAAAFTeJzLYWBg4MhgTmEQTM4vTc5ISXI
wNDLXMwBCwxygFFMeC5BkOACK_v__fz8rkY
Gg2gcQtf-
JUbsAonY_frVJDkAyKZ5IMxsgZs4nYGYCyMx6
gmYmKYDU2RNUI8iQJA9R1AUAPoRejg",
 "update_seq": "0-
g1AAAAAFTeJzLYWBg4MhgTmEQTM4vTc5ISXI
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](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>

- Response:

```
{
 "_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](http://localhost:5984/emp/_all_docs)

- Using url:

- [http://localhost:5984/emp/\\_all\\_docs](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/bda8e42fe0bc193ddbe
d360c4c0469cf/ -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.
- o 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`