

# CouchDB - Quick Guide

Advertisements



Secure Your Wife & Child's  
Buy ₹ 1 Crore Term Insurance  
@ just ₹490 p.m\*

⬅ Previous Page

Next Page ➡

## CouchDB - Introduction

Database management system provides mechanism for storage and retrieval of data. There are three main types of database management systems namely RDBMS (Relational Database management Systems), OLAP (Online Analytical Processing Systems) and NoSQL.

### RDBMS

RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

The data in RDBMS is stored in database objects called **tables**. The table is a collection of related data entries and it consists of columns and rows. It stores only structured data.

### OLAP

Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers and analysts to get an insight of the information through fast, consistent, and interactive access to information.

### NoSQL Databases

A NoSQL database (sometimes called as Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational

databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data (big data).

The primary objective of a NoSQL database is to have the following –

- Simplicity of design,
- Horizontal scaling, and
- Finer control over availability.

NoSQL databases use different data structures compared to relational databases. It makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it must solve. These databases store both structured data and unstructured data like audio files, video files, documents, etc. These NoSQL databases are classified into three types and they are explained below.

**Key-value Store** – These databases are designed for storing data in key-value pairs and these databases will not have any schema. In these databases, each data value consists of an indexed key and a value for that key.

Examples – BerkeleyDB, Cassandra, DynamoDB, Riak.

**Column Store** – In these databases, data is stored in cells grouped in columns of data, and these columns are further grouped into Column families. These column families can contain any number of columns.

Examples – BigTable, HBase, and HyperTable.

**Document Store** – These are the databases developed on the basic idea of key-value stores where "documents" contain more complex data. Here, each document is assigned a unique key, which is used to retrieve the document. These are designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data.

Examples – CouchDB and MongoDB.

## What is CouchDB?

CouchDB is an open source database developed by Apache software foundation. The focus is on the ease of use, embracing the web. It is a NoSQL document store database.

It uses JSON, to store data (documents), java script as its query language to transform the documents, http protocol for api to access the documents, query the indices with the web browser. It is a multi master application released in 2005 and it became an apache project in 2008.

## Why CouchDB?

CouchDB have an HTTP-based REST API, which helps to communicate with the database easily. And the simple structure of HTTP resources and methods (GET, PUT, DELETE) are easy to understand and use.

As we store data in the flexible document-based structure, there is no need to worry about the structure of the data.

Users are provided with powerful data mapping, which allows querying, combining, and filtering the information.

CouchDB provides easy-to-use replication, using which you can copy, share, and synchronize the data between databases and machines.

## Data Model

Database is the outermost data structure/container in CouchDB.

Each database is a collection of independent documents.

Each document maintains its own data and self-contained schema.

Document metadata contains revision information, which makes it possible to merge the differences occurred while the databases were disconnected.

CouchDB implements multi version concurrency control, to avoid the need to lock the database field during writes.

## Features of CouchDB:Reduce the Content

### Document Storage

CouchDB is a document storage NoSQL database. It provides the facility of storing documents with unique names, and it also provides an API called RESTful HTTP API for reading and updating (add, edit, delete) database documents.

In CouchDB, documents are the primary unit of data and they also include metadata. Document fields are uniquely named and contain values of varying types (text, number, Boolean, lists, etc.), and there is no set limit to text size or element count.

Document updates (add, edit, delete) follow Atomicity, i.e., they will be saved completely or not saved at all. The database will not have any partially saved or edited documents.

### Json Document Structure

```
{
  "field" : "value",
  "field" : "value",
  "field" : "value",
}
```

## ACID Properties

CouchDB contains ACID properties as one of its features.

**Consistency** – When the data in CouchDB was once committed, then this data will not be modified or overwritten. Thus, CouchDB ensures that the database file will always be in a consistent state.

A multi-Version Concurrency Control (MVCC) model is used by CouchDB reads, because of which the client will see a consistent snapshot of the database from the beginning to the end of the read operation.

Whenever a documents is updated, CouchDB flushes the data into the disk, and the updated database header is written in two consecutive and identical chunks to make up the first 4k of the file, and then synchronously flushed to disk. Partial updates during the flush will be discarded.

If the failure occurred while committing the header, a surviving copy of the previous identical headers will remain, ensuring coherency of all previously committed data. Except the header area, consistency checks or fix-ups after a crash or a power failure are never necessary.

## Compaction

Whenever the space in the database file got wasted above certain extent, all the active data will be copied (cloned) to a new file. When the copying process is entirely done, then the old file will be discarded. All this is done by compaction process. The database remains online during the compaction and all updates and reads are allowed to complete successfully.

## Views

Data in CouchDB is stored in semi-structured documents that are flexible with individual implicit structures, but it is a simple document model for data storage and sharing. If we want see our data in many different ways, we need a way to filter, organize and report on data that hasn't been decomposed into tables.

To solve this problem, CouchDB provides a view model. Views are the method of aggregating and reporting on the documents in a database, and are built on-demand to aggregate, join and report on database documents. Because views are built dynamically and don't affect the underlying document, you can have as many different view representations of the same data as you like.

## History

CouchDB was written in Erlang programming language.

It was started by Damien Katz in 2005.

CouchDB became an Apache project in 2008.

The current version of CouchDB is 1.6.1.

## CouchDB - Installation

This chapter teaches you how to install CouchDB in windows as well as Linux systems.

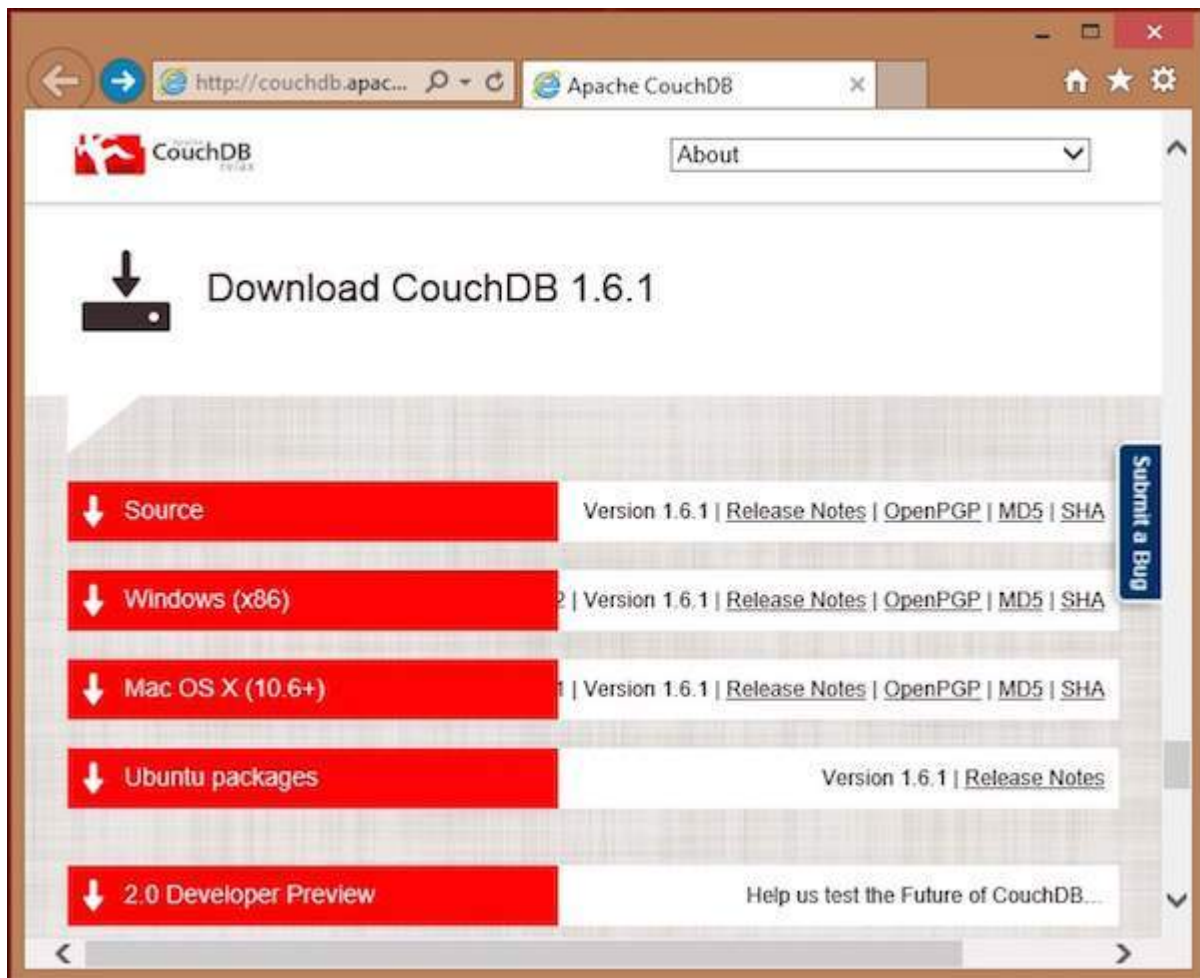
### Installing CouchDB in Windows

#### Download CouchDB

The official website for CouchDB is <https://couchdb.apache.org> . If you click the given link, you can get the home page of the CouchDB official website as shown below.



If you click on the download button that will lead to a page where download links of CouchDB in various formats are provided. The following snapshot illustrates the same.



Choose the download link for windows systems and select one of the provided mirrors to start your download.

## Installing CouchDB

CouchDB will be downloaded to your system in the form of setup file named **setup-couchdb-1.6.1\_R16B02.exe**. Run the setup file and proceed with the installation.

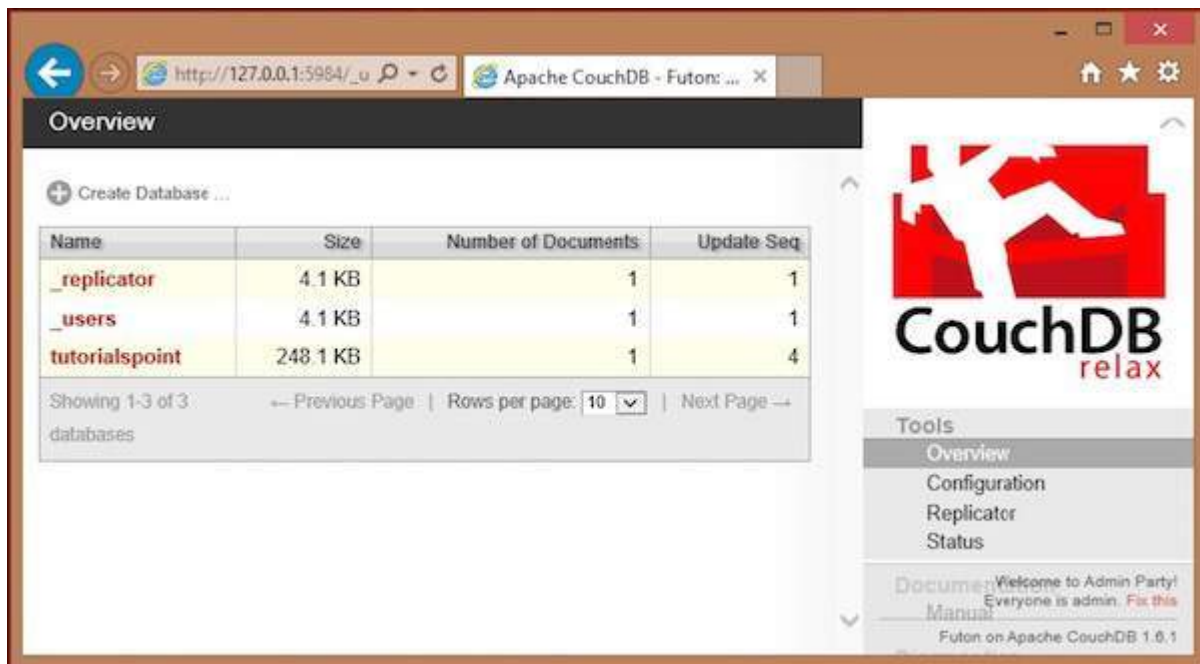
After installation, open built-in web interface of CouchDB by visiting the following **link**: **http://127.0.0.1:5984/**. If everything goes fine, this will give you a web page, which have the following output.

```
{
  "couchdb": "Welcome", "uuid": "c8d48ac61bb497f4692b346e0f400d60",
  "version": "1.6.1",
  "vendor": {
    "version": "1.6.1", "name": "The Apache Software Foundation"
  }
}
```

You can interact with the CouchDB web interface by using the following url –

```
http://127.0.0.1:5984/_utils/
```

This shows you the index page of Futon, which is the web interface of CouchDB.



## Installing CouchDB in Linux Systems

For many of the Linux flavored systems, they provide CouchDB internally. To install this CouchDB follow the instructions.

On Ubuntu and Debian you can use –

```
sudo aptitude install couchdb
```

On Gentoo Linux there is a CouchDB ebuild available –

```
sudo emerge couchdb
```

If your Linux system does not have CouchDB, follow the next section to install CouchDB and its dependencies.

### Installing CouchDB Dependencies

Following is the list of dependencies that are to be installed to get CouchDB in your system–

Erlang OTP

ICU

OpenSSL

Mozilla SpiderMonkey

GNU Make

GNU Compiler Collection

libcurl

help2man



Python for docs

Python Sphinx

To install these dependencies, type the following commands in the terminal. Here we are using Centos 6.5 and the following commands will install the required softwares compatible to Centos 6.5.

```
$sudo yum install autoconf
$sudo yum install autoconf-archive
$sudo yum install automake
$sudo yum install curl-devel
$sudo yum install erlang-asn1
$sudo yum install erlang-erts
$sudo yum install erlang-eunit
$sudo yum install erlang-os_mon
$sudo yum install erlang-xmerl
$sudo yum install help2man
$sudo yum install js-devel
$sudo yum install libicu-devel
$sudo yum install libtool
$sudo yum install perl-Test-Harness
```

**Note** – For all these commands you need to use sudo. The following procedure converts a normal user to a sudoer.

Login as root in admin user

Open **sudo** file using the following command –

```
visudo
```

Then edit as shown below to give your existing user the sudoer privileges –

```
Hadoop All=(All) All , and press esc : x to write the changes to the file.
```

After downloading all the dependencies in your system, download CouchDB following the given instructions.

## Downloading CouchDB

Apache software foundation will not provide the complete .tar file for CouchDB, so you have to install it from the source. Download the source file of CouchDB from the following link – [click here](http://www.google.com/url?q=http%3A%2F%2Fwww.apache.org%2Fdist%2Fcouchdb%2Fsource%2F1.6.1%2Fapache-couchdb-1.6.1.tar.gz) .

Create a new directory to install CouchDB, browse to such created directory and download CouchDB source by executing the following commands –

```
$ cd
$ mkdir CouchDB
$ cd CouchDB/
$ wget
http://www.google.com/url?q=http%3A%2F%2Fwww.apache.org%2Fdist%2Fcouchdb%2Fsource%2F1.6.1%2Fapache-couchdb-1.6.1.tar.gz
```



This will download CouchDB source file into your system. Now unzip the **apache-couchdb-1.6.1.tar.gz** as shown below.

```
$ tar zxvf apache-couchdb-1.6.1.tar.gz
```

## Configuring CouchDB

To configure CouchDB, do the following –

Browse to the home folder of CouchDB.

Login as superuser.

Configure using `./configure` prompt as shown below –

```
$ cd apache-couchdb-1.6.1
$ su
Password:
# ./configure --with-erlang=/usr/lib64/erlang/usr/include/
```

It gives you the following output similar to that shown below with a concluding line saying – **You have configured Apache CouchDB, time to relax.**

```
# ./configure --with-erlang=/usr/lib64/erlang/usr/include/

checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking how to create a ustar tar archive... gnutar
..... *
..... *
config.status: creating var/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: creating src/snappy/google-snappy/config.h
config.status: src/snappy/google-snappy/config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands

You have configured Apache CouchDB, time to relax.

Run `make && sudo make install' to install.
```

## Installing CouchDB

Now type the following command to install CouchDB in your system.

```
# make && sudo make install
```

It installs CouchDB in your system with a concluding line saying – **You have installed Apache CouchDB, time to relax.**

## Starting CouchDB

To start CouchDB, browse to the CouchDB home folder and use the following command –

```
$ cd apache-couchdb-1.6.1
$ cd etc
$ couchdb start
```

It starts CouchDB giving the following output: –

```
Apache CouchDB 1.6.1 (LogLevel=info) is starting.
Apache CouchDB has started. Time to relax.
[info] [lt;0.31.0gt;] Apache CouchDB has started on http://127.0.0.1:5984/
[info] [lt;0.112.0gt;] 127.0.0.1 - - GET / 200
[info] [lt;0.112.0gt;] 127.0.0.1 - - GET /favicon.ico 200
```

## Verification

Since CouchDB is a web interface, try to type the following homepage url in the browser.

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

It produces the following output –

```
{
  "couchdb": "Welcome",
  "uuid": "8f0d59acd0e179f5e9f0075fa1f5e804",
  "version": "1.6.1",
  "vendor": {
    "name": "The Apache Software Foundation",
    "version": "1.6.1"
  }
}
```

# CouchDB - Curl & Futon

## cURL Utility

cURL utility is a way to communicate with CouchDB.

It is a tool to transfer data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, FTPS, TFTP, DICT, TELNET, LDAP or FILE). The command is designed to work without user interaction. cURL offers a busload of useful tricks like proxy support, user authentication, ftp upload, HTTP post, SSL (https:) connections, cookies, file transfer resume and more.

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. This chapter teaches you how to use cURL utility.

## Using cURL Utility

You can access any website using cURL utility by simply typing cURL followed by the website address as shown below –

```
curl www.tutorialspoint.com/
```

By default, the cURL utility returns the source code of the requested page. It displays this code on the terminal window.

## cURL Utility Options

cURL utility provides various options to work with, and you can see them in cURL utility help.

The following code shows some portion of cURL help.

```
$ curl --help
Usage: curl [options...] <url>
Options: (H) means HTTP/HTTPS only, (F) means FTP only
  --anyauth Pick "any" authentication method (H)
  -a/--append Append to target file when uploading (F/SFTP)
  --basic Use HTTP Basic Authentication (H)
  --cacert <file> CA certificate to verify peer against (SSL)
-d/--data <data> HTTP POST data (H)
  --data-ascii <data> HTTP POST ASCII data (H)
  --data-binary <data> HTTP POST binary data (H)
  --data-urlencode <name=data/name@filename> HTTP POST data
urlencoded (H)
  --delegation STRING GSS-API delegation permission
  --digest Use HTTP Digest Authentication (H)
  --disable-eprt Inhibit using EPRT or LPRT (F)
  --disable-epsv Inhibit using EPSV (F)

-F/--form <name=content> Specify HTTP multipart POST data (H)
  --form-string <name=string> Specify HTTP multipart POST data (H)
  --ftp-account <data> Account data to send when requested by server
(F)
  --ftp-alternative-to-user <cmd> String to replace "USER [name]" (F)
  --ftp-create-dirs Create the remote dirs if not present (F)
  --ftp-method [multi cwd/no cwd/single cwd] Control CWD usage (F)
  --ftp-pasv Use PASV/EPSV instead of PORT (F)

-G/--get Send the -d data with a HTTP GET (H)

-H/--header <line> Custom header to pass to server (H)
-I/--head Show document info only
-h/--help This help text
  --hostpubmd5 <md5> Hex encoded MD5 string of the host public key.
(SSH)
  -0/--http1.0 Use HTTP 1.0 (H)
  --ignore-content-length Ignore the HTTP Content-Length header
  -i/--include Include protocol headers in the output (H/F)
```

```

-M/--manual Display the full manual

-o/--output <file> Write output to <file> instead of stdout
  --pass <pass> Pass phrase for the private key (SSL/SSH)
  --post301 Do not switch to GET after following a 301
redirect (H)
  --post302 Do not switch to GET after following a 302
redirect (H)
-O/--remote-name Write output to a file named as the remote file
  --remote-name-all Use the remote file name for all URLs
-R/--remote-time Set the remote file's time on the local output
-X/--request <command> Specify request command to use
  --retry <num> Retry request <num> times if transient problems
occur
  --retry-delay <seconds> When retrying, wait this many seconds
between each
  --retry-max-time <seconds> Retry only within this period
-T/--upload-file <file> Transfer <file> to remote site
  --url <URL> Set URL to work with
-B/--use-ascii Use ASCII/text transfer

```

While communicating with CouchDB, certain options of cURL utility were extensively used. Following are the brief descriptions of some important options of cURL utility including those used by CouchDB.

## **-X flag**

(HTTP) Specifies a custom request method used when communicating with the HTTP server. The specified request is used instead of the method otherwise used (which defaults to GET). Read the HTTP 1.1 specification for details and explanations.

(FTP) Specifies a custom FTP command to use instead of LIST when doing file lists with ftp.

## **-H**

(HTTP) Extra header is used when getting a web page. Note that if you add a custom header that has the same name as one of the internal ones cURL would use, your externally set header will be used instead of the internal one. This allows you to make even trickier work than cURL would normally do. You should not replace internally set headers without perfectly knowing what you're doing. Replacing an internal header with the one without content on the right side of the colon, will prevent that header from appearing.

cURL assures that each header you add/replace get sent with the proper end of line marker. Neither you should add that as a part of the header content nor add newlines or carriage returns to disorder things.

See also the -A/--user-agent and -e/--referer options.

This option can be used multiple times to add/replace/remove multiple headers.

## -d flag

Using this flag of cURL, you can send data along with the HTTP POST request to the server, as if it was filled by the user in the form and submitted.

### Example

Suppose there is a website and you want to login into it or send some data to the website using -d flag of cURL utility as shown below.

```
curl -X PUT http://mywebsite.com/Login.html -d userid=001 -d password=tutorialspoint
```

It sends a post chunk that looks like **"userid=001&password=tutorialspoint"**. Likewise you can also send documents (JSON ) using -d flag.

## -o flag

Using this flag, cURL writes the output of the request to a file.

### Example

The following example shows the use of **-o** flag of cURL utility.

```
$ curl -o example.html www.tutorialspoint.com/index.htm
% Total % Received % Xferd Average Speed Time Time Time Current
   Dload Upload Total Spent Left Speed
100 81193 0 81193 0 0 48168 0 --:--:-- 0:00:01 --:--:--
58077
```

This gets the source code of the homepage of tutorialspoint.com, creates a file named example.com and saves the output in the file named example.html.

Following is the snapshot of the **example.html**.



-O

This flag is similar to **-o**, the only difference is with this flag, a new file with the same name as the requested url was created, and the source code of the requested url will be copied to it.

### Example

The following example shows the use of **-O** flag of cURL utility.

```
$ curl -O www.tutorialspoint.com/index.htm
% Total % Received % Xferd Average Speed Time Time Time Current
   Dload Upload Total Spent Left
Speed
100 81285 0 81285 0 0 49794 0 --:--:-- 0:00:01 --:--:--
60077
```

It creates a new file with the name index.htm and saves the source code of the index page of tutorialspoint.com in it.

## Hello CouchDB

You can access the homepage of the CouchDB by sending a GET request to the CouchDB instance installed. First of all make sure you have installed CouchDB in your Linux environment and it is running successfully, and then use the following syntax to send a get request to the CouchDB instance.

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

This gives you a JSON document as shown below where CouchDB specifies the details such as version number, name of the vendor, and version of the software.

```
$ curl http://127.0.0.1:5984/
{
  "couchdb" : "Welcome",
  "uuid" : "8f0d59acd0e179f5e9f0075fa1f5e804",
  "version" : "1.6.1",
  "vendor" : {
    "name":"The Apache Software Foundation",
    "version":"1.6.1"
  }
}
```

## List of All Databases

You can get the list of all the databases created, by sending a get request along with the string **"\_all\_dbs string "**. Following is the syntax to get the list of all databases in CouchDB.

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

It gives you the list of all databases in CouchDB as shown below.

```
$ curl -X GET http://127.0.0.1:5984/_all_dbs
[ "_replicator" , "_users" ]
```

## Creating a Database

You can create a database in CouchDB using cURL with PUT header using the following syntax –

```
$ curl -X PUT http://127.0.0.1:5984/database_name
```

### Example

As an example, using the above given syntax create a database with name **my\_database** as shown below.

```
$ curl -X PUT http://127.0.0.1:5984/my_database
{"ok":true}
```

### Verification

Verify whether the database is created, by listing out all the databases as shown below. Here you can observe the name of newly created database, **"my\_database"** in the list



```
$ curl -X GET http://127.0.0.1:5984/_all_dbs  
[ "_replicator" , "_users" , "my_database" ]
```

## Getting Database Info

You can get the information about database using the get request along with the database name. Following is the syntax to get the database information.

### Example

As an example let us get the information of the database named **my\_database** as shown below. Here you can get the information about your database as a response.

```
$ curl -X GET http://127.0.0.1:5984/my_database  
  
{  
  "db_name" : "my_database",  
  "doc_count" : 0,  
  "doc_del_count" : 0,  
  "update_seq" : 0,  
  "purge_seq" : 0,  
  "compact_running" : false,  
  "disk_size" : 79,  
  "data_size" : 0,  
  "instance_start_time" : "1423628520835029",  
  "disk_format_version" : 6,  
  "committed_update_seq" : 0  
}
```

## Futon

Futon is the built-in, web based, administration interface of CouchDB. It provides a simple graphical interface using which you can interact with CouchDB. It is a naive interface and it provides full access to all CouchDB features. Following is the list of those features –

### Databases –

- Creates databases.
- Destroys databases.

### Documents –

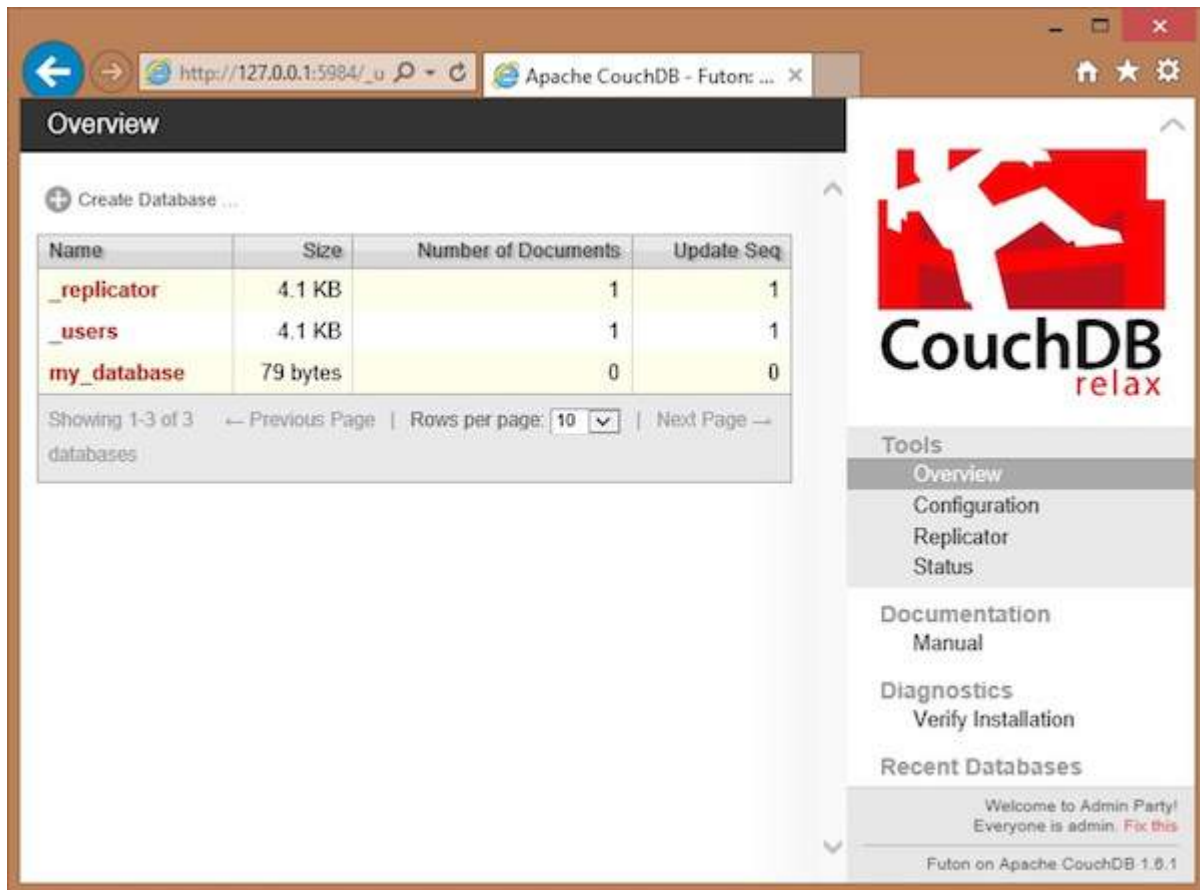
- Creates documents.
- Updates documents.
- Edits documents.
- Deletes documents.

## Starting Futon

Make sure CouchDB is running and then open the following url in browser –

[http://127.0.0.1:5984/\\_utils/](http://127.0.0.1:5984/_utils/)

If you open this url, it displays the Futon home page as shown below –



On the left hand side of this page you can observe the list of all the current databases of CouchDB. In this illustration, we have a database named **my\_database**, along with system defined databases **\_replicator** and **\_user**.

On the right hand side you can see the following –

**Tools** – In this section you can find **Configuration** to configure CouchDB, **Replicator** to perform replications, and **Status** to verify status of CouchDB and recent modifications done on CouchDB.

**Documentation** – This section contains the complete documentation for the recent version of CouchDB.

**Diagnostics** – Under this you can verify the installation of CouchDB.

**Recent Databases** – Under this you can find the names of recently added databases.

## CouchDB - HTTP API

Using HTTP request headers, you can communicate with CouchDB. Through these requests we can retrieve data from the database, store data in to the database in the form of documents, and we can view as well as format the documents stored in a database.

## HTTP Request Formats

While communicating with the database we will use different request formats like get, head, post, put, delete, and copy. For all operations in CouchDB, the input data and the output data structures will be in the form of JavaScript Object Notation (JSON) object.

Following are the different request formats of HTTP Protocol used to communicate with CouchDB.

**GET** – This format is used to get a specific item. To get different items, you have to send specific url patterns. In CouchDB using this GET request, we can get static items, database documents and configuration, and statistical information in the form of JSON documents (in most cases).

**HEAD** – The HEAD method is used to get the HTTP header of a GET request without the body of the response.

**POST** – Post request is used to upload data. In CouchDB using POST request, you can set values, upload documents, set document values, and can also start certain administration commands.

**PUT** – Using PUT request, you can create new objects, databases, documents, views and design documents.

**DELETE** – Using DELETE request, you can delete documents, views, and design documents.

**COPY** – Using COPY method, you can copy documents and objects.

## HTTP Request Headers

HTTP headers should be supplied to get the right format and encoding. While sending the request to the CouchDB server, you can send Http request headers along with the request. Following are the different Http request headers.

**Content-type** – This Header is used to specify the content type of the data that we supply to the server along with the request. Mostly the type of the content we send along with the request will be MIME type or JSON (application/json). Using Content-type on a request is highly recommended.

**Accept** – This header is used to specify the server, the list of data types that client can understand, so that the server will send its response using those data types.

Generally here, you can send the list of MIME data types the client accepts, separated by colons.

Though, using Accept in queries of CouchDB is not required, it is highly recommended to ensure that the data returned can be processed by the client.

## Response Headers

These are the headers of the response sent by the server. These headers give information about the content sent by the server as response.

**Content-type** – This header specifies the MIME type of the data returned by the server. For most request, the returned MIME type is text/plain.

**Cache-control** – This header suggests the client about treating the information sent by the server. CouchDB mostly returns the must-revalidate, which indicates that the information should be revalidated if possible.

**Content-length** – This header returns the length of the content sent by the server, in bytes.

**Etag** – This header is used to show the revision for a document, or a view.

## Status Codes

Following is the tabular form of the status code sent by the http header and the description of it.

<b>200 – OK</b>	This status will be issued when a request completed successfully.
<b>201 – Created</b>	This status will be issued when a document is created.
<b>202 – Accepted</b>	This status will be issued when a request is accepted.
<b>404 – Not Found</b>	This status will be issued when the server is unable to find the requested content.
<b>405 – Resource Not Allowed</b>	This status is issued when the HTTP request type used is invalid.
<b>409 – Conflict</b>	This status is issued whenever there is any update conflict.
<b>415 – Bad Content Type</b>	This status indicated that the requested content type is not supported by the server.
<b>500 – Internal Server Error</b>	This status is issued whenever the data sent in the request is invalid.

# HTTP URL Paths

There are certain url paths using which, you can interact with the database directly. Following is the tabular format of such url paths.

URL	Operation
<b>PUT /db</b>	This url is used to create a new database.
<b>GET /db</b>	This url is used to get the information about the existing database.
<b>PUT /db/document</b>	This url is used to create a document/update an existing document.
<b>GET /db/document</b>	This url is used to get the document.
<b>DELETE /db/document</b>	This url is used to delete the specified document from the specified database.
<b>GET /db/_design/design-doc</b>	This url is used to get the definition of a design document.
<b>GET /db/_design/designdoc/_view/view-name</b>	This url is used to access the view, view-name from the design document from the specified database.

## CouchDB - Creating a Database

Database is the outermost data structure in CouchDB where your documents are stored. You can create these databases using cURL utility provided by CouchDB, as well as Futon the web interface of CouchDB.

### Creating a Database using cURL Utility

You can create a database in CouchDB by sending an HTTP request to the server using PUT method through cURL utility. Following is the syntax to create a database –

```
$ curl -X PUT http://127.0.0.1:5984/database_name
```

Using **-X** we can specify HTTP custom request method to be used. In this case, we are using PUT method. When we use the PUT operation/method, the content of the url specifies the object name we are creating using HTTP request. Here we have to send the name of the database using put request in the url to create a database.

### Example

Using the above given syntax if you want to create a database with name **my\_database**, you can create it as follows

```
curl -X PUT http://127.0.0.1:5984/my_database
{
  "ok":true
}
```

As a response the server will return you a JSON document with content **"ok" – true** indicating the operation was successful.

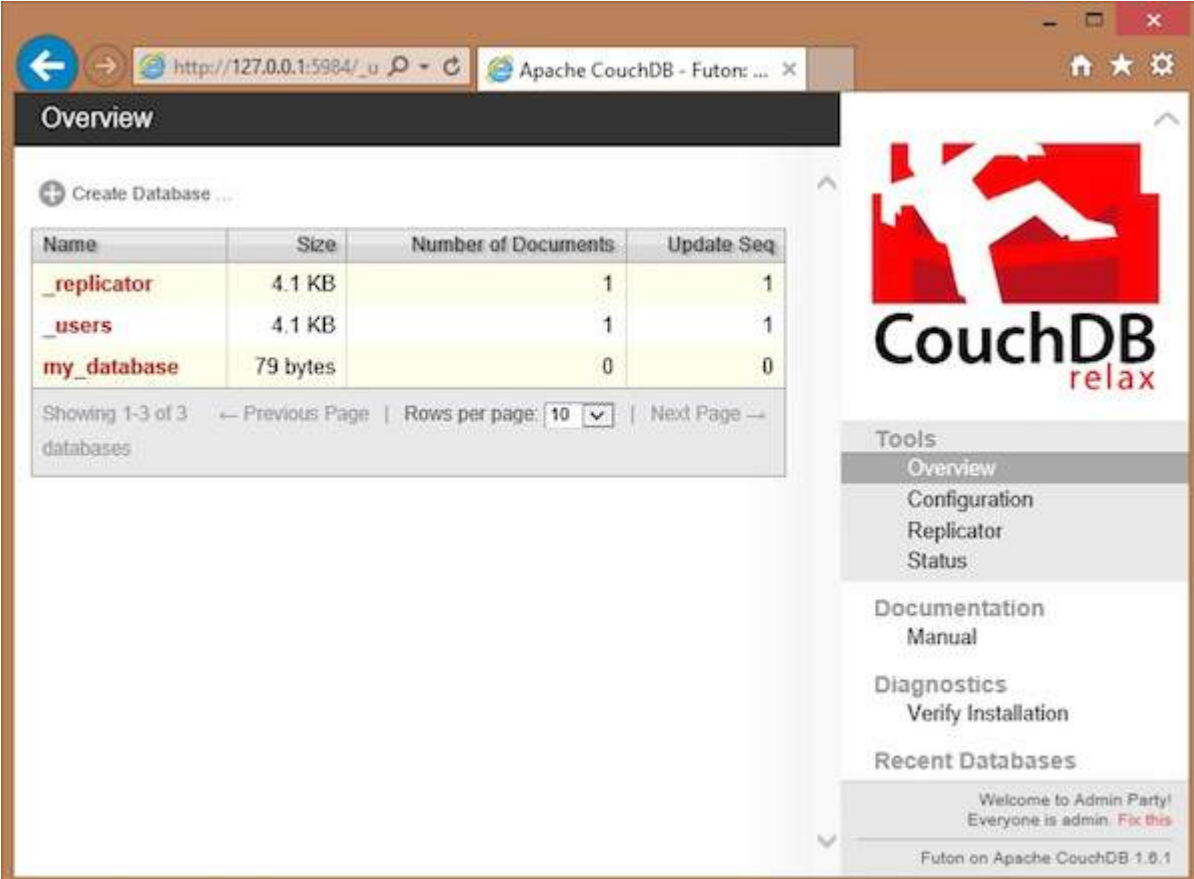
## Verification

Verify whether the database is created, by listing out all the databases as shown below. Here you can observe the name of a newly created database, **" my\_database "** in the list.

```
$ curl -X GET http://127.0.0.1:5984/_all_dbs
[ "_replicator " , " _users " , " my_database " ]
```

## Creating a Database using Futon

To create a database open the **http://127.0.0.1:5984/\_utils/**. You will get an Overview/index page of CouchDB as shown below.



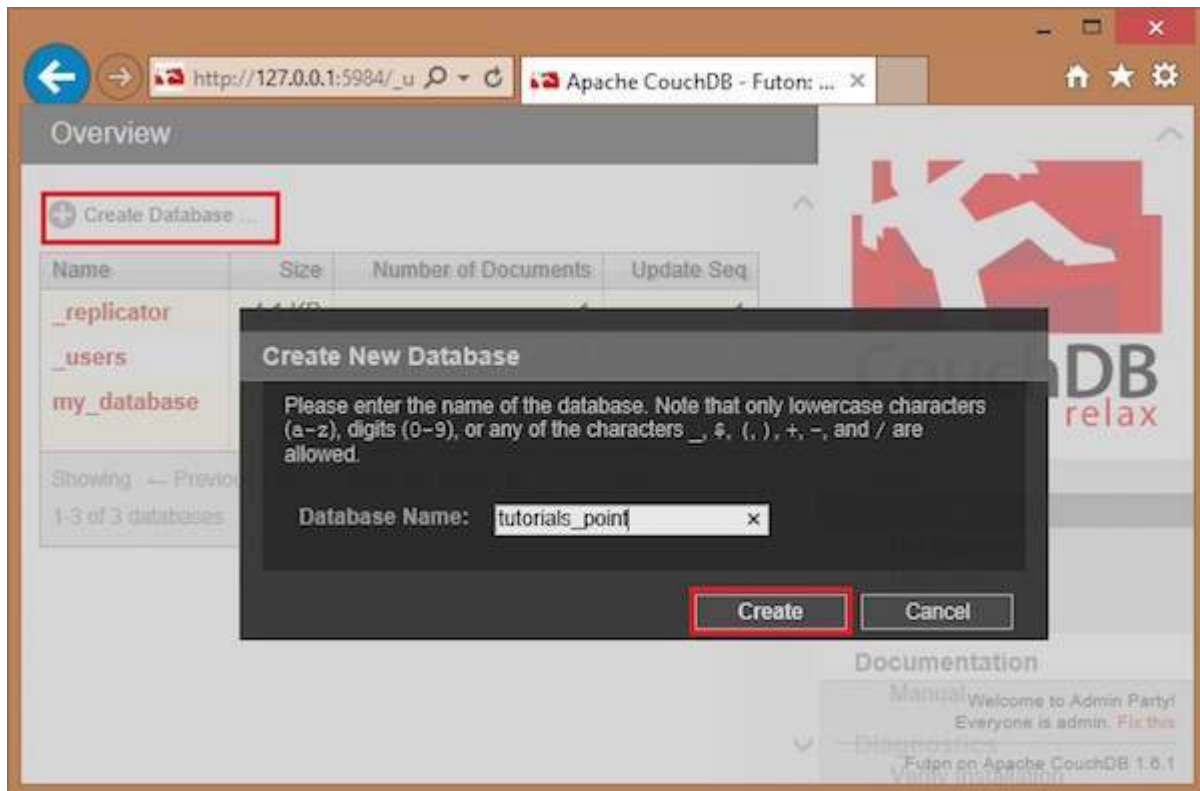
The screenshot shows the Apache CouchDB Futon interface. The main content area displays the 'Overview' page with a table of databases. The table has four columns: Name, Size, Number of Documents, and Update Seq. It lists three databases: \_replicator (4.1 KB, 1 document, update seq 1), \_users (4.1 KB, 1 document, update seq 1), and my\_database (79 bytes, 0 documents, update seq 0). Below the table, it indicates 'Showing 1-3 of 3' and provides navigation links for 'Previous Page' and 'Next Page'. The right sidebar contains a CouchDB logo with the tagline 'relax', a 'Tools' menu with links to Overview, Configuration, Replicator, and Status, a 'Documentation' section with a link to the Manual, a 'Diagnostics' section with a link to 'Verify Installation', and a 'Recent Databases' section. At the bottom of the sidebar, there is a message: 'Welcome to Admin Party! Everyone is admin. Fix this' and a footer indicating 'Futon on Apache CouchDB 1.6.1'.

Name	Size	Number of Documents	Update Seq
<b>_replicator</b>	4.1 KB	1	1
<b>_users</b>	4.1 KB	1	1
<b>my_database</b>	79 bytes	0	0

Showing 1-3 of 3 | Previous Page | Rows per page: 10 | Next Page →  
databases

In this page, you can see the list of databases in CouchDB, an option button Create Database on the left hand side.

Now click on the create database link. You can see a popup window **Create New Databases** asking for the database name for the new database. Choose any name following the mentioned criteria. Here we are creating another database with name tutorials\_point. Click on the create button as shown in the following screenshot.



## CouchDB - Deleting a Database

### Deleting a Database using cURL Utility

You can delete a database in CouchDB by sending a request to the server using DELETE method through cURL utility. Following is the syntax to create a database –

```
$ curl -X DELETE http://127.0.0.1:5984/database_name
```

Using **-X** we can specify a custom request method of HTTP we are using, while communicating with the HTTP server. In this case, we are using the DELETE method. Send the url to the server by specifying the database to be deleted in it.

### Example

Assume there is a database named my\_database2 in CouchDB. Using the above given syntax if you want to delete it, you can do it as follows –

```
$ curl -X DELETE http://127.0.0.1:5984/my_database2
{
```



```
"ok" : true
}
```

As a response, the server will return you a JSON document with content **"ok" – true** indicating the operation was successful.

## Verification

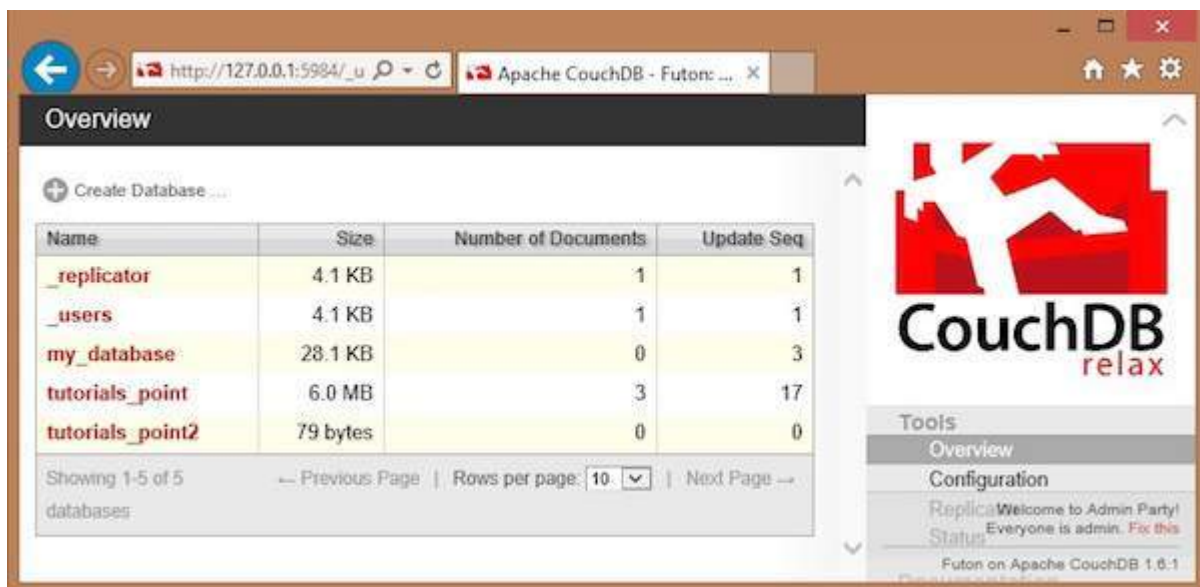
Verify whether the database is deleted by listing out all the databases as shown below. Here you can observe the name of the deleted database, **"my\_database"** is not there in the list.

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

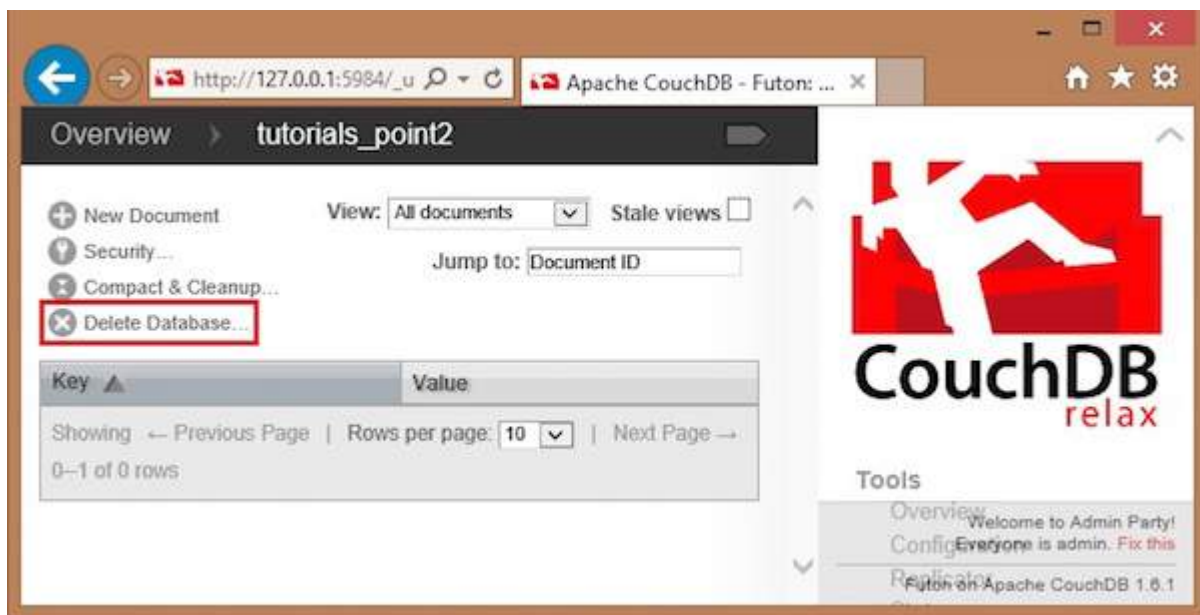
[ "_replicator " , " _users " ]
```

## Deleting a Database using Futon

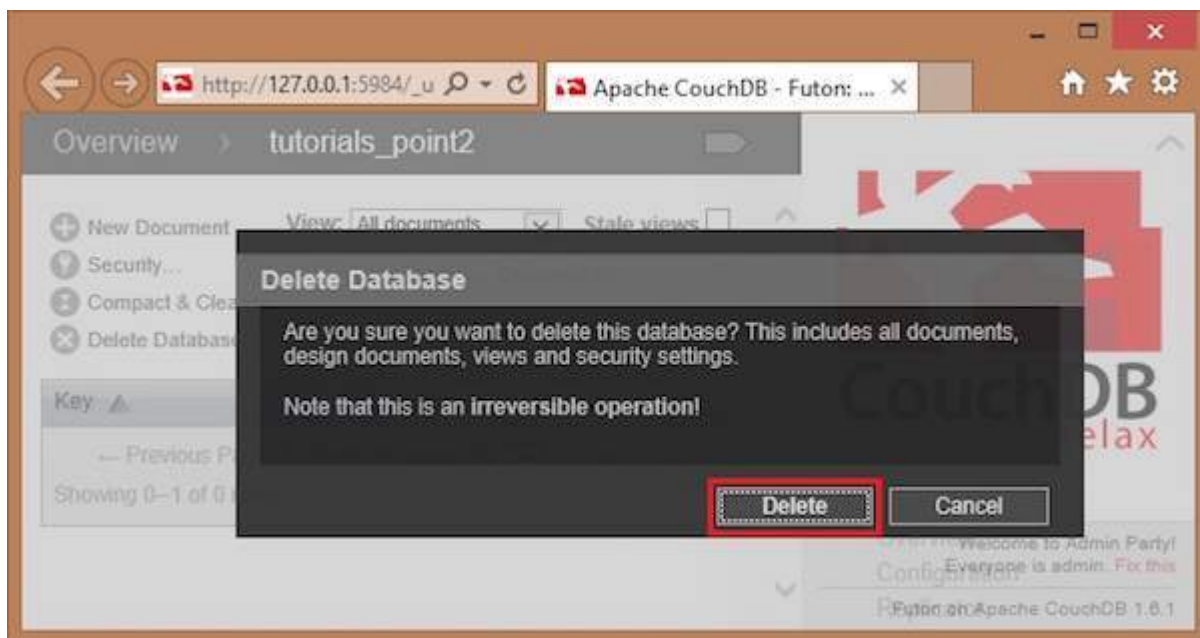
To delete a database, open the **http://127.0.0.1:5984/\_utils/** url where you will get an Overview/index page of CouchDB as shown below.



Here you can see three user created databases. Let us delete the database named **tutorials\_point2**. To delete a database, select one from the list of databases, and click on it, which will lead to the overview page of the selected database where you can see the various operations on databases. The following screenshot shows the same –



Among them you can find **Delete Database** option. By clicking on it you will get a popup window, asking whether you are sure! Click on delete, to delete the selected database.



## CouchDB - Creating a Document

Documents are CouchDB's central data structure. Contents of the database will be stored in the form of Documents instead of tables. You can create these documents using cURL utility provided by CouchDB, as well as Futon. This chapter covers the ways to create a document in a database.

Each document in CouchDB has a unique ID. You can choose your own ID that should be in the form of a string. Generally, UUID (Universally Unique Identifier) is used, which are random numbers that have least chance of creating a duplicate. These are preferred to avoid collisions.

# Creating a Document using cURL Utility

You can create a document in CouchDB by sending an HTTP request to the server using PUT method through cURL utility. Following is the syntax to create a document.

```
$ curl -X PUT http://127.0.0.1:5984/database name/"id" -d ' { document} '
```

Using **-X**, we can specify a custom request method of HTTP we are using, while communicating with the HTTP server. In this case, we are using PUT method. When we use the PUT method, the content of the url specifies the object name we are creating using the HTTP request. Here we have to send the following –

The name of the database name in which we are creating the document.

The document id.

The data of the document. **-d** option is used to send the data/document through HTTP request. While writing a document simply enter your Field-Value pairs separated by colon, within flower brackets as shown below –

```
{  
  Name : Raju  
  age : 23  
  Designation : Designer  
}
```

## Example

Using the above given syntax if you want to create a document with id **001** in a database with name **my\_database**, you can create it as shown below.

```
$ curl -X PUT http://127.0.0.1:5984/my_database/"001" -d  
'{ " Name " : " Raju " , " age " : " 23 " , " Designation " : " Designer " }'  
  
{"ok":true,"id":"001","rev":"1-1c2fae390fa5475d9b809301bbf3f25e"}
```

The response of CouchDB to this request contains three fields –

**"ok"**, indicating the operation was successful.

**"id"**, which stores the id of the document and

**"rev"**, this indicates the revision id. Every time you revise (update or modify) a document a **\_rev** value will be generated by CouchDB. If you want to update or delete a document, CouchDB expects you to include the **\_rev** field of the revision you wish to change. When CouchDB accepts the change, it will generate a new revision number. This mechanism ensures concurrency control.

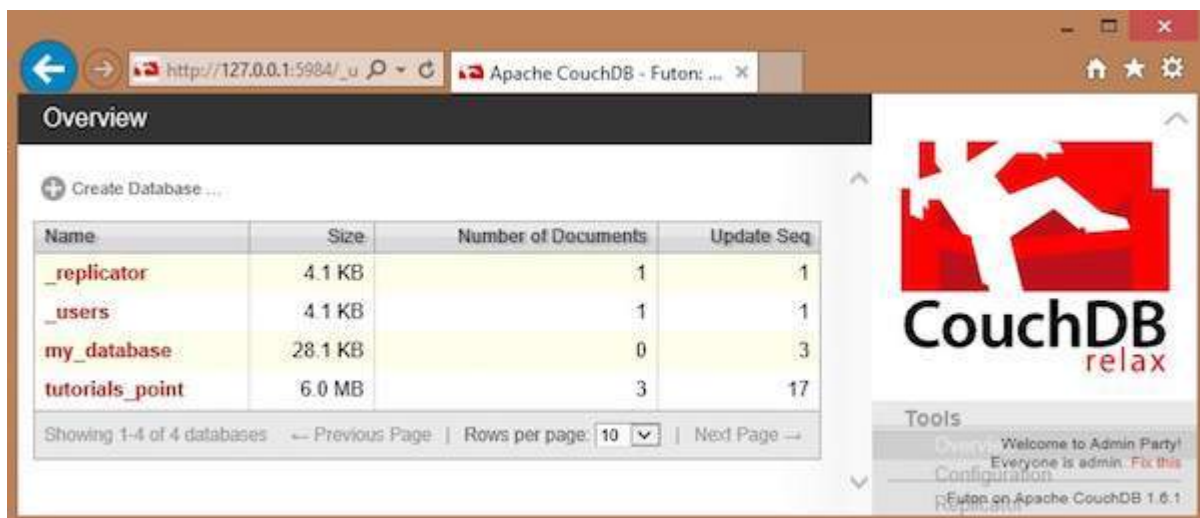
## Verification

If you want to view the created document you can get it using the document as shown below.

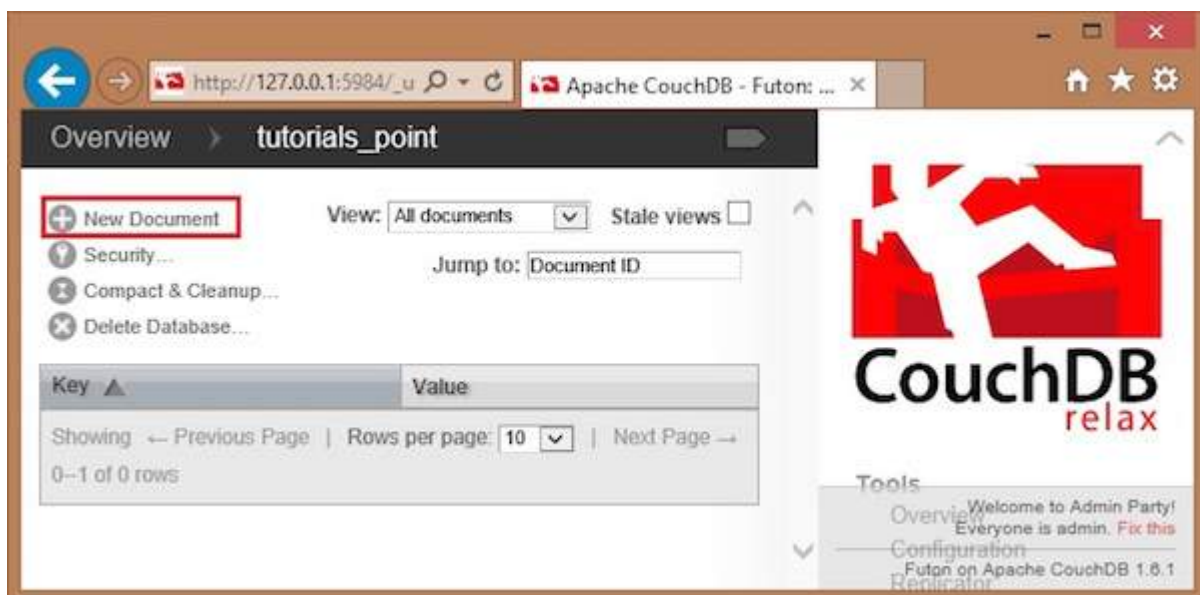
```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id": "001",
  "_rev": "1-3fcc78daac7a90803f0a5e383f4f1e1e",
  "Name": "Raju",
  "age": 23,
  "Designation": "Designer"
}
```

## Creating a Document using Futon

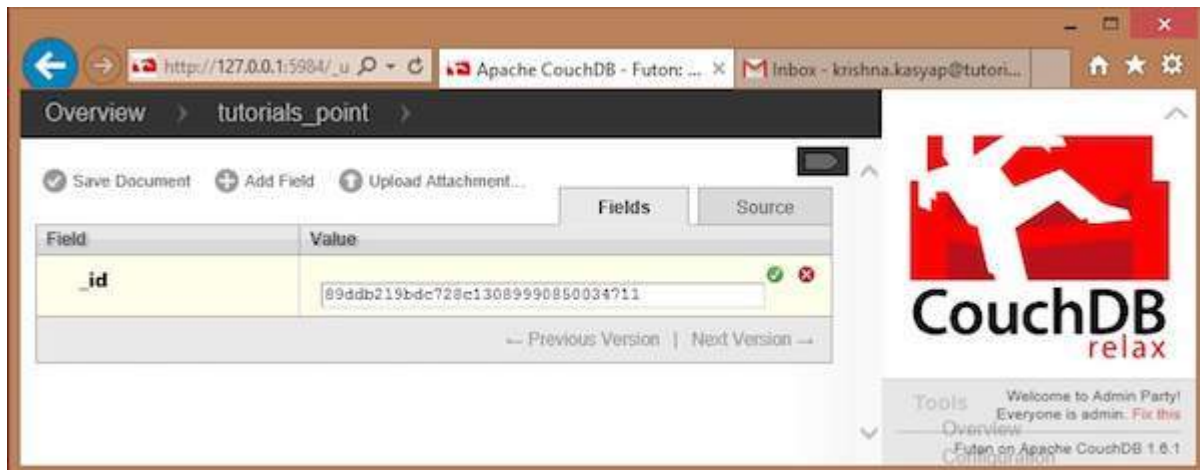
To Create a document open the **[http://127.0.0.1:5984/\\_utils/](http://127.0.0.1:5984/_utils/)** url to get an Overview/index page of CouchDB as shown below.



Select the database in which you want to create the document. Open the Overview page of the database and select **New Document** option as shown below.



When you select the **New Document** option, 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. In the following illustration, we have created a new document with an id 001.

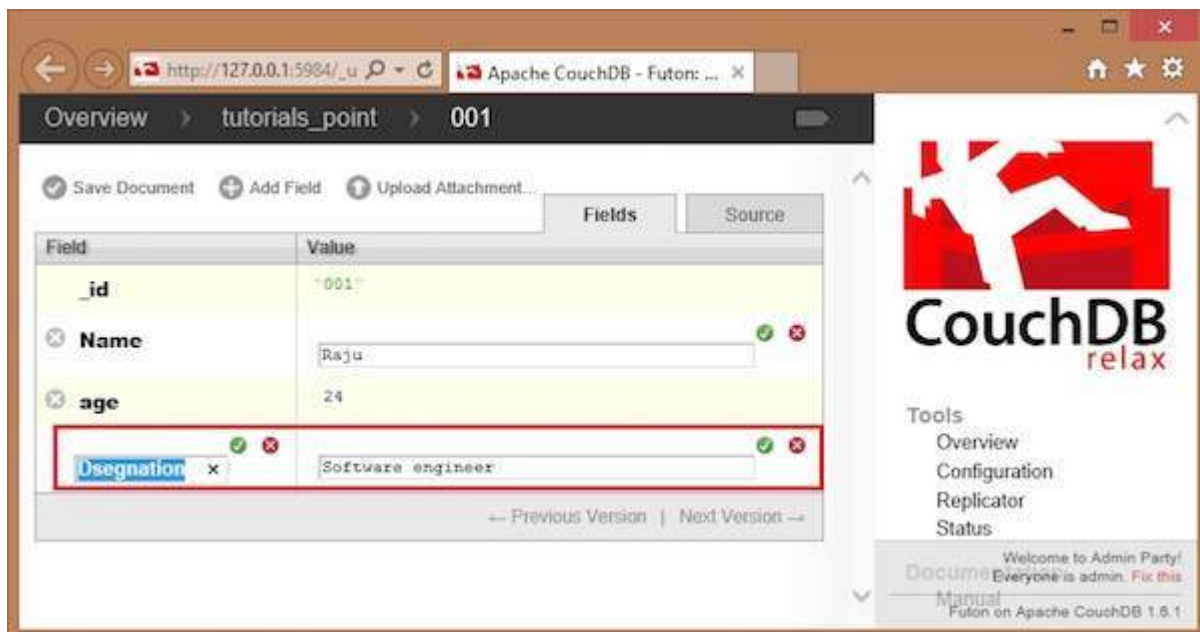


In this page, you can observe three options – save Document, Add Field and Upload Attachment.

## Add Field to the Document

To add field to the document click on **Add Field** option. After creating a database, you can add a field to it using this option. Clicking on it will get you a pair of text boxes, namely, **Field**, **value**. You can edit these values by clicking on them. Edit those values and type your desired Field-Value pair. Click on the green button to save these values.

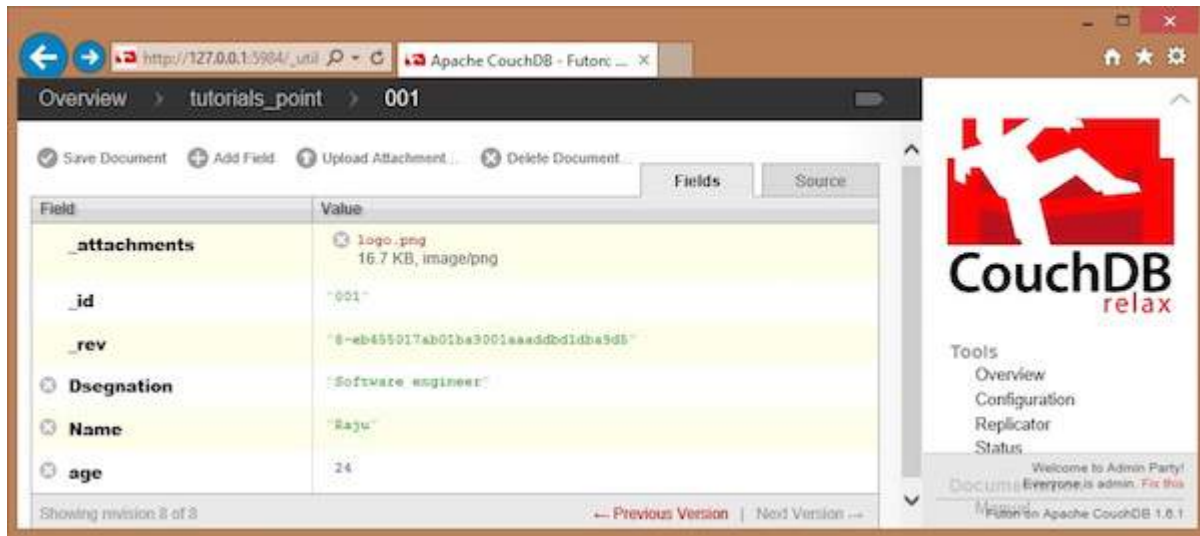
In the following illustration, we have created three fields Name, age and, Designation of the employee.



## Save Document



You can save the changes made to the document by clicking on this option. After saving, a new id **\_rev** will be generated as shown below.



## CouchDB - Updating a Document

### Updating Documents using cURL

You can update a document in CouchDB by sending an HTTP request to the server using PUT method through cURL utility. Following is the syntax to update a document.

```
curl -X PUT http://127.0.0.1:5984/database_name/document_id/ -d '{ "field" : "value", "_rev" : "r
```

### Example

Suppose there is a document with id 001 in the database named my\_database. You can delete this as shown below.

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, therefore get the document as shown below.

```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id" : "001",
  "_rev" : "2-04d8eac1680d237ca25b68b36b8899d3 " ,
  "age" : "23"
}
```

Use revision id \_rev from the document to update the document. Here we are updating the age from 23 to 24.

```
$ curl -X PUT http://127.0.0.1:5984/my_database/001/ -d
' { " age " : " 24 " , " _rev " : " 1-1c2fae390fa5475d9b809301bbf3f25e " } '

{ " ok " : true , " id " : " 001 " , " rev " : " 2-04d8eac1680d237ca25b68b36b8899d3 " }
```

## Verification

To verify the document, get the document again using GET request as shown below.

```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id" : " 001 ",
  "_rev" : " 2-04d8eac1680d237ca25b68b36b8899d3 " ,
  "age" : " 23 "
}
```

## Note

Following are some important points to be noted while updating a document.

The URL we send in the request containing the database name and the document id.

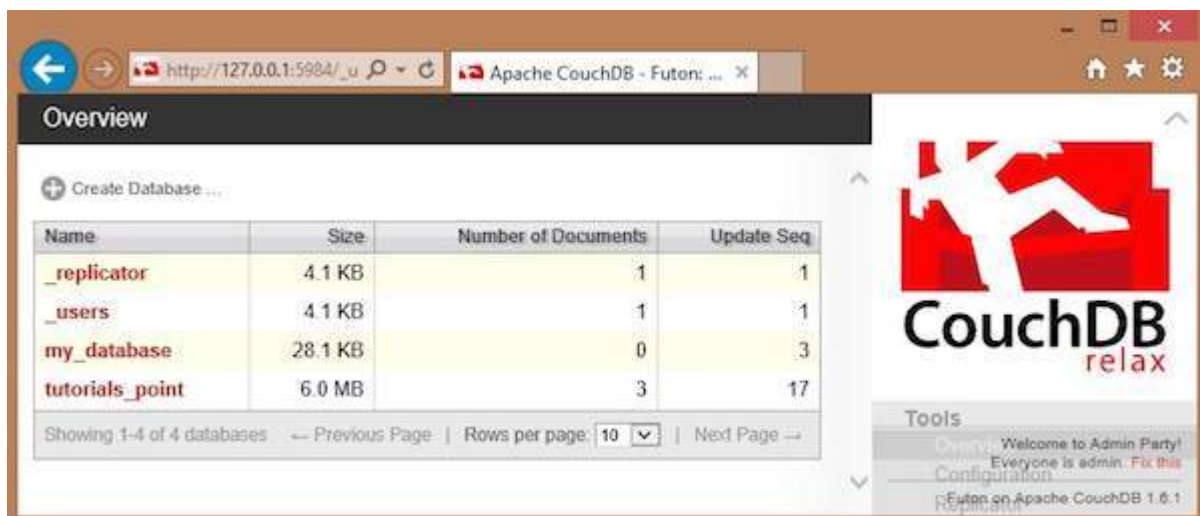
Updating an existing document is same as updating the entire document. You cannot add a field to an existing document. You can only write an entirely new version of the document into the database with the same document ID.

We have to supply the revision number as a part of the JSON request.

In return JSON contains the success message, the ID of the document being updated, and the new revision information. If you want to update the new version of the document, you have to quote this latest revision number.

## Updating Documents using Futon

To delete a document open the **[http://127.0.0.1:5984/\\_utils/](http://127.0.0.1:5984/_utils/)** url to get an Overview/index page of CouchDB as shown below.



Select the database in which the document to be updated exists and click it. Here we are updating a document in the database named **tutorials\_point**. You will get the list of documents in the database as shown below.



The screenshot shows the Apache CouchDB Futon interface. The browser address bar displays `http://127.0.0.1:5984/_u`. The page title is "Overview > tutorials\_point". On the left, there are navigation links: "New Document", "Security...", "Compact & Cleanup...", and "Delete Database...". The "View" dropdown is set to "All documents", and the "Stale views" checkbox is unchecked. A "Jump to: Document ID" input field is present. The main content area displays a table with the following data:

Key	Value
"001" ID: 001	{rev: "1-967a00dff5e02add41819138abb3284d"}
"002" ID: 002	{rev: "2-bc27b6930ca514527d8954c7c43e6a09"}
"003" ID: 003	{rev: "2-bc27b6930ca514527d8954c7c43e6a09"}

At the bottom of the table, it says "Showing 1-3 | Previous Page | Rows per page: 10 | Next Page | of 3 rows". On the right, there is a sidebar with the CouchDB logo and "relax" text. Below the logo, there are links for "Tools" (Overview, Configuration, Replicator, Status) and "Documentation" (Manual, Welcome to Admin Party! Everyone is admin. Fix this, Futon on Apache CouchDB 1.6.1).

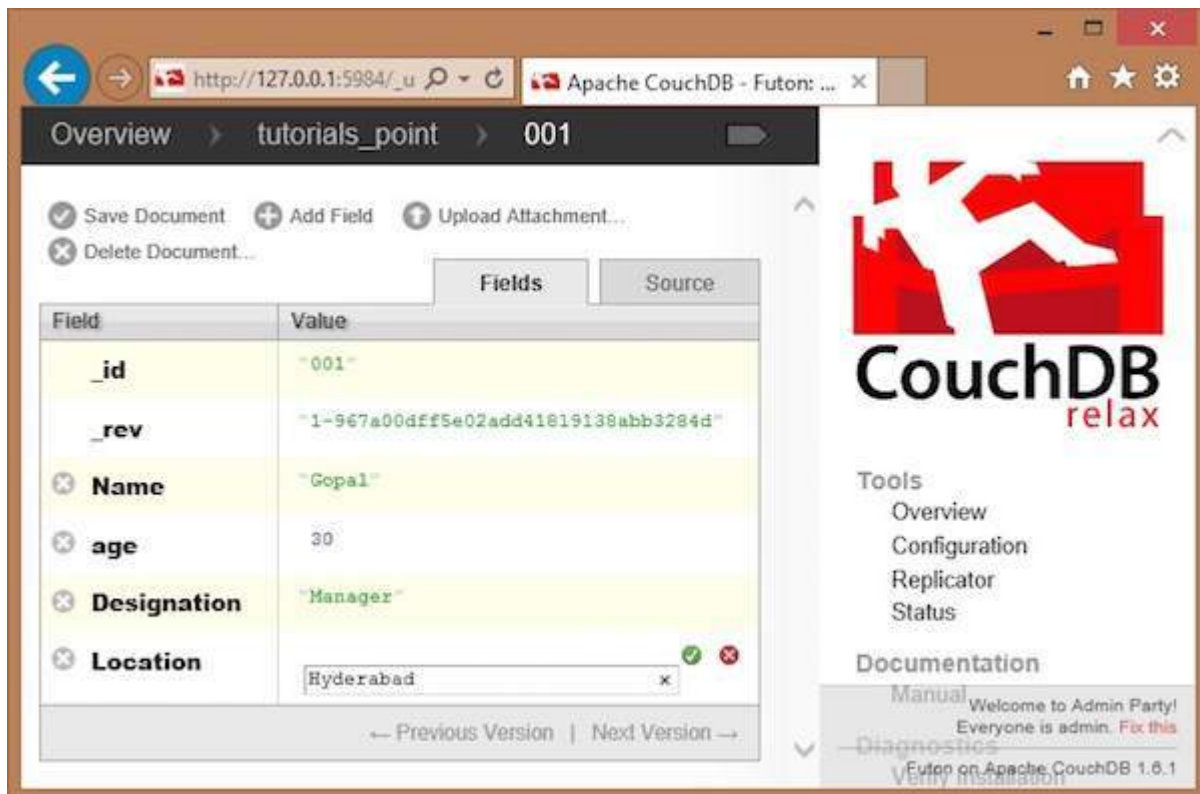
Select a document that you want to update and click on it. You will get the contents of the documents as shown below.

The screenshot shows the Apache CouchDB Futon interface for a specific document. The browser address bar displays `http://127.0.0.1:5984/_u`. The page title is "Overview > tutorials\_point > 001". On the left, there are navigation links: "Save Document", "Add Field", "Upload Attachment...", and "Delete Document...". The "Fields" tab is selected, showing a table with the following data:

Field	Value
<b>_id</b>	"001"
<b>_rev</b>	"1-967a00dff5e02add41819138abb3284d"
<b>Name</b>	"Gopal"
<b>age</b>	30
<b>Designation</b>	"Manager"
<b>Location</b>	"Delhi"

At the bottom of the table, there are links for "Previous Version" and "Next Version". On the right, there is a sidebar with the CouchDB logo and "relax" text. Below the logo, there are links for "Tools" (Overview, Configuration, Replicator, Status) and "Documentation" (Manual, Welcome to Admin Party! Everyone is admin. Fix this, Futon on Apache CouchDB 1.6.1).

Here, to update the location from Delhi to Hyderabad, click on the text box, edit the field, and click the green button to save the changes as shown below.



## CouchDB - Deleting a Document

### Deleting a Document using cURL Utility

You can delete a document in CouchDB by sending an HTTP request to the server using DELETE method through cURL utility. Following is the syntax to delete a document.

```
curl -X DELETE http : // 127.0.0.1:5984 / database name/database id?_rev id
```

Using **-X**, we can specify a custom request method of HTTP we are using, while communicating with the HTTP server. In this case, we are using Delete method. To delete a database /database\_name/database\_id/ is not enough. You have to pass the recent revision id through the url. To mention attributes of any data structure "?" is used.

### Example

Suppose there is a document in database named **my\_database** with document id 001. To delete this document, you have to get the rev id of the document. Get the document data as shown below.

```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id" : " 001 ",
  "_rev" : " 2-04d8eac1680d237ca25b68b36b8899d3 " ,
  "age" : " 23 "
}
```

Now specify the revision id of the document to be deleted, id of the document, and database name the document belongs to, as shown below –

```
$ curl -X DELETE http://127.0.0.1:5984/my_database/001?rev=1-3fcc78daac7a90803f0a5e383f4f1e1e  
  
{"ok":true,"id":"001","rev":"2-3a561d56de1ce3305d693bd15630bf96"}
```

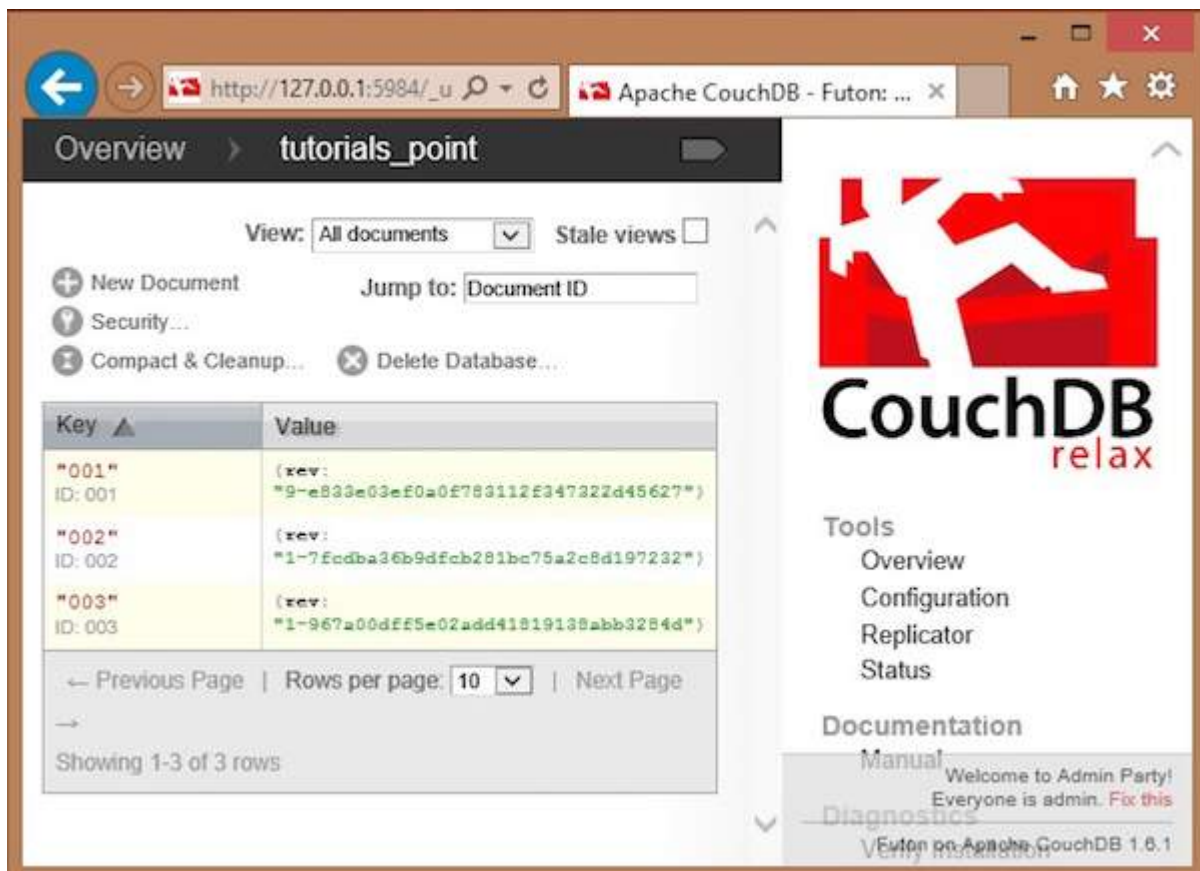
## Verification

To verify whether the document is deleted, try to fetch the document by using the **GET** method. Since you are fetching a deleted document, this will give you an error message as shown below –

```
$ curl -X GET http://127.0.0.1:5984/my_database/001  
  
{"error":"not_found","reason":"deleted"}
```

## Deleting a Document using Futon

First of all, verify the documents in the database. Following is the snapshot of the database named **tutorials\_point**.



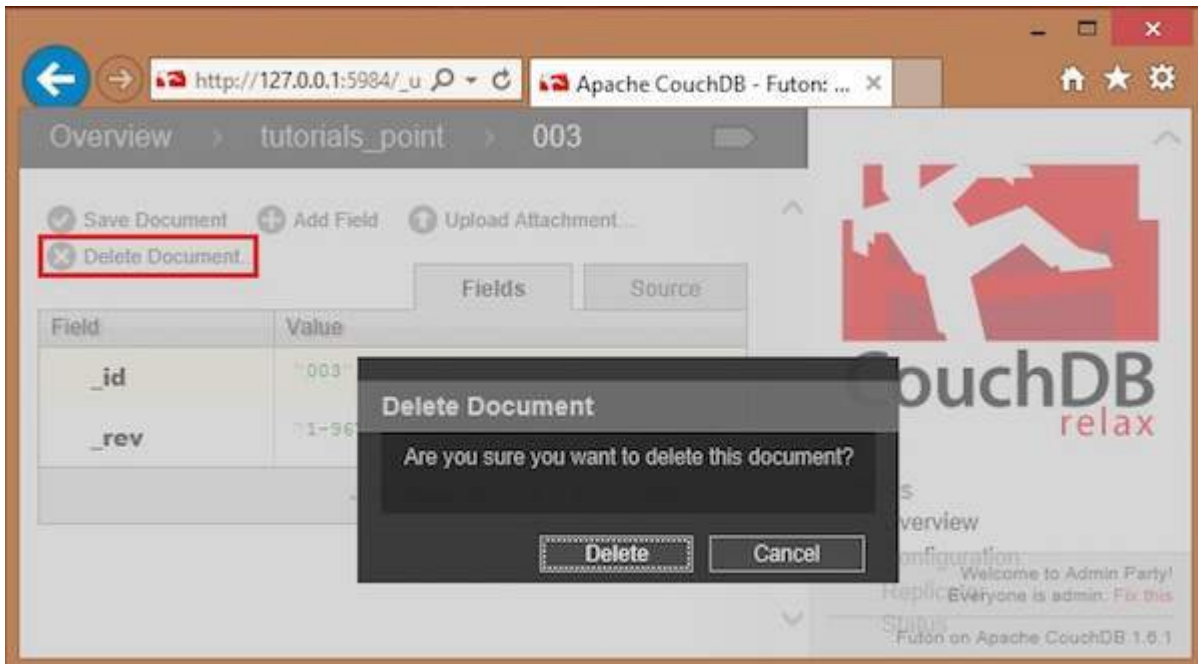
Here you can observe, the database consists of three documents. To delete any of the documents say **003**, do the following –

Click on the document, you will get a page showing the contents of selected document in the form of field-value pairs.

This page also contains four options namely **Save Document, Add Field, Upload Attachment, Delete Document.**

Click on **Delete Document** option.

You will get a dialog box saying "**Are you sure you want to delete this document?**" Click on delete, to delete the document.



## CouchDB - Attaching Files

### Attaching Files using cURL

You can attach files to CouchDB just like email. The file contains metadata like name and includes its MIME type, and the number of bytes the attachment contains. To attach files to a document you have to send PUT request to the server. Following is the syntax to attach files to the document –

```
$ curl -vX PUT http://127.0.0.1:5984/database_name/database_id  
/filename?rev=document_rev_id --data-binary @filename -H "Content-Type:  
type of the content"
```

The request has various options that are explained below.

**--data-binary@** – This option tells cURL to read a file's contents into the HTTP request body.

**-H** – This option is used to mention the content type of the file we are going to upload.

## Example

Let us attach a file named **boy.jpg**, to the document with id **001**, in the database named **my\_database** by sending PUT request to CouchDB. Before that, you have to fetch the data of the document with id **001** to get its current **rev** id as shown below.

```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id": "001",
  "_rev": "1-967a00dff5e02add41819138abb3284d"
}
```

Now using the **\_rev** value, send the PUT request to the CouchDB server as shown below.

```
$ curl -vX PUT http://127.0.0.1:5984/my_database/001/boy.jpg?rev=1-967a00dff5e02add41819138abb3284d --data-binary @boy.jpg -H "ContentType: image/jpg"
```

## Verification

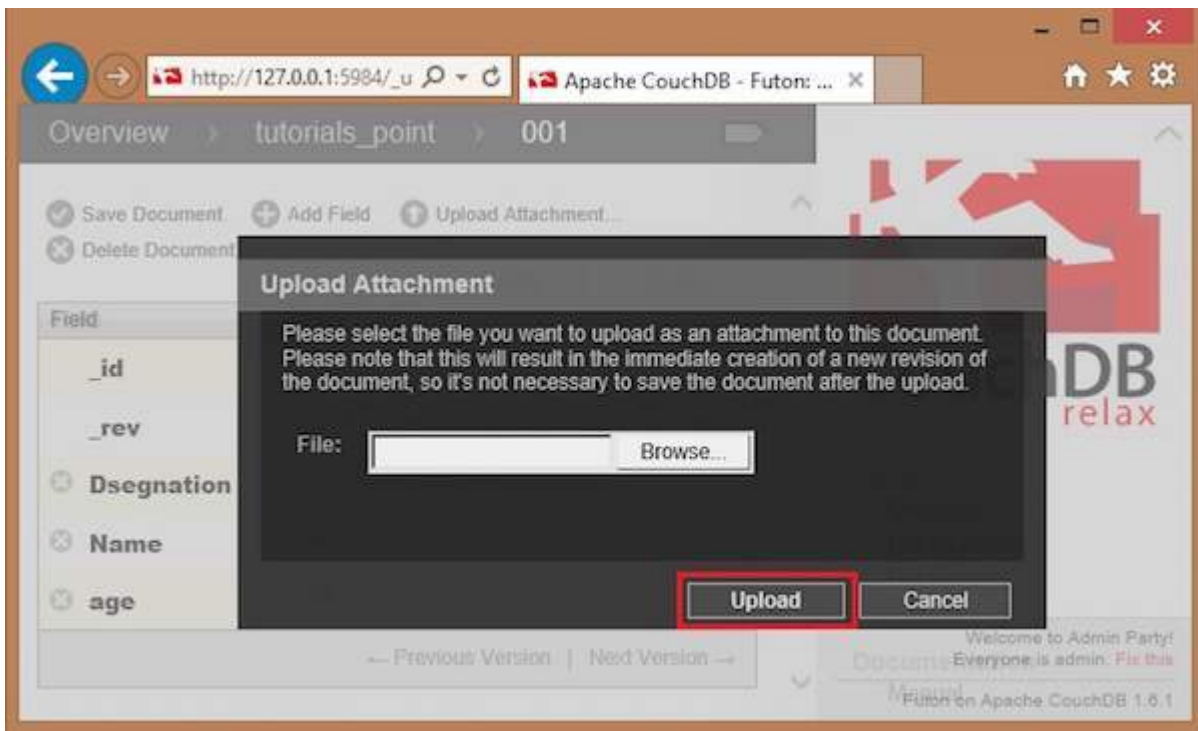
To verify whether the attachment is uploaded, fetch the document content as shown below–

```
$ curl -X GET http://127.0.0.1:5984/my_database/001
{
  "_id": "001",
  "_rev": "2-4705a219cdcca7c72aac4f623f5c46a8",
  "_attachments": {
    "boy.jpg": {
      "content_type": "image/jpg",
      "revpos": 2,
      "digest": "md5-9Swz8jvmga5mfBIsmCxQtQ==",
      "length": 91408,
      "stub": true
    }
  }
}
```

## Attaching Files using Futon

### Upload Attachment

Using this option, you can upload a new attachment such as a file, image, or document, to the database. To do so, click on the **Upload Attachment** button. A dialog box will appear where you can choose the file to be uploaded. Select the file and click on the **Upload** button.



The file uploaded will be displayed under `_attachments` field. Later you can see the file by clicking on it.



[FAQ's](#) [Cookies Policy](#) [Contact](#)

© Copyright 2018. All Rights Reserved.