Couchdb report

Jeremi Szlapka

Advanced Databases

For Peter Given

# Summary

This report was meant to examine CouchDB in its appropriate uses, usability with Python and in ease of migration to the cloud. CouchDB is a database was found to be well suited for data store requirements which focus on accessibility above accuracy. As such, storing references to papers would be appropriate use of it. A prototype was made in Python. Drivers for CouchDB in Python were compared, and it was found many of Python’s popular options have become deprecated. New libraries have been created since then which easy the use of this library by incorporating python syntax and integration. However, the prototype used here forgoes the convenience in favour of a RESTful API library. The ease of migration to the cloud was shown via steps to migrate to Azure cloud.

Table of Contents

[Summary 1](#_Toc116641921)

[Introduction 3](#_Toc116641922)

[Solution description for CouchDB 3](#_Toc116641923)

[API 4](#_Toc116641924)

[Drivers for CouchDB 8](#_Toc116641925)

[Requests 8](#_Toc116641926)

[Flask CouchDB 8](#_Toc116641927)

[CouchDB-python 8](#_Toc116641928)

[Cloudant/Cloudant-SDK 8](#_Toc116641929)

[AIOCouch 9](#_Toc116641930)

[CouchDB on the cloud 9](#_Toc116641931)

[Conclusion 10](#_Toc116641932)

[References 11](#_Toc116641933)

# Introduction

This report aims to examine CouchDB by creating a prototype project using the database and then migrating it to the cloud. It aims to focus on basic NoSQL database functionality, create, read, update, redestroy documents and on views, both normal and map-reduce views. The process of creating this vertical slice began with choosing an appropriate solution to produce.

# Solution description for CouchDB

Academic paper storage system would be appropriate for CouchDB. In such a system, the need to be able to interact with the database is more important than having consistency between databases. A user would prefer to still be able to add a new book they found in the library to the system on their phone even if their desktop is not online at that moment.

The data stored for such a system would be mostly text and numeric so CouchDB’s keeping of redundant data would not have too much of an impact. In turn, this redundant data can benefit the system by allowing users to undo later if they delete a reference they once thought would be required.

CouchDB’s flexible document storage can allow for this system to add new structures of reference data and support different referencing standards more easily. Changes would only have to be done on the client related to inputting and displaying the data.

# API

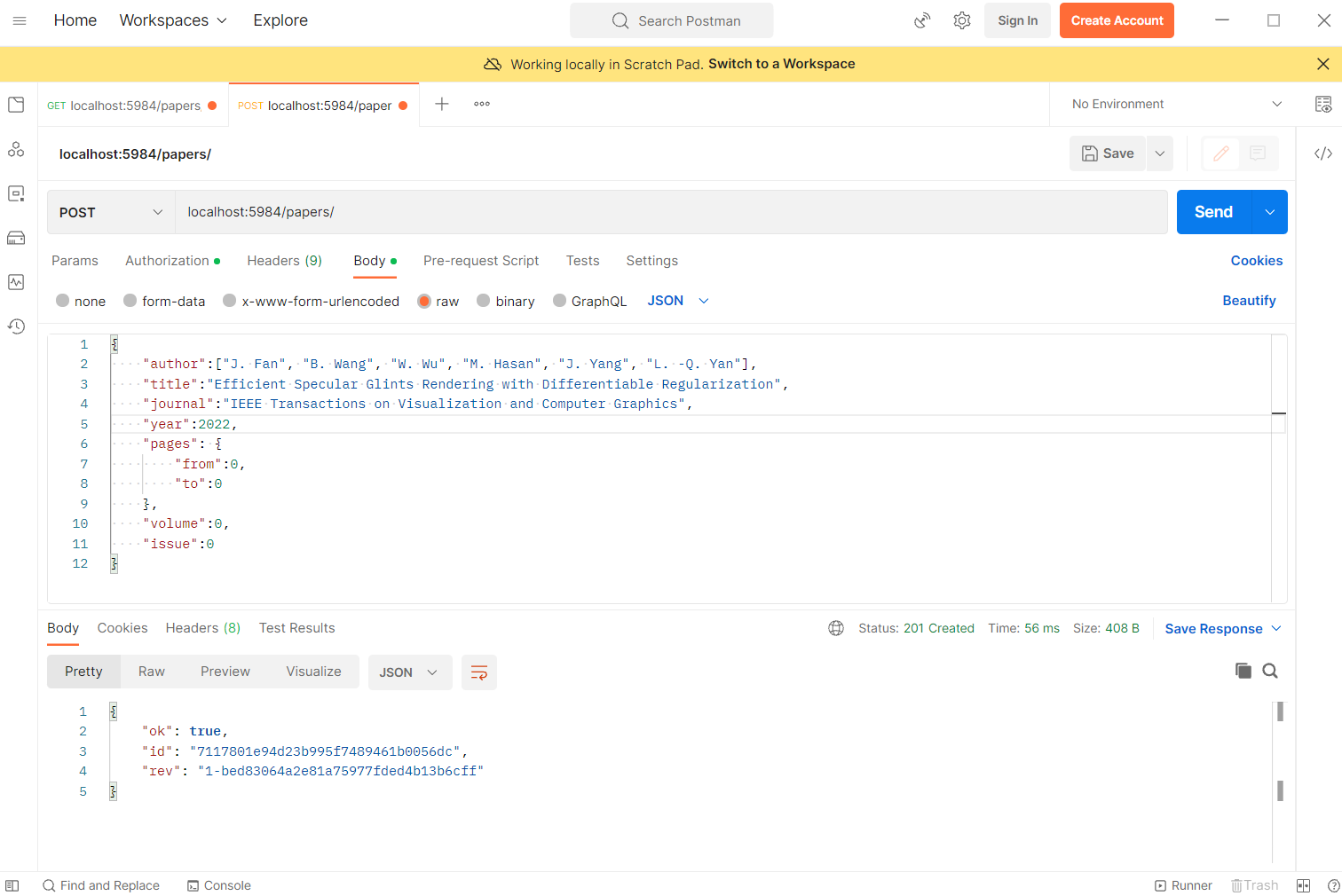


Figure 1 Inserting new document into the papers database

Figure 1 shows a Post call to the papers database with a document in the body and a user and password in the authentication create a new document in the database. The confirmation below shows the document has been added and a id and revision number have been assigned.

Graphical user interface, text, application, email

Description automatically generated

Figure 2 Getting a document with a Get request

Figure 2 contains a GET API call to the id of the previously added document under the Paper database returns the data of the document in the response section.

Graphical user interface, text, application, email

Description automatically generated

Figure 3 Updating a document with a put request

Figure 3 has a put request updating the document created in figure 1 with -1 for the invalid 0s. The response shows both the update occurred with a true in the ok property and the changed revision code.

Graphical user interface, text, application, email

Description automatically generated

Figure 4 Removing a document from the paper database with a delete account

Figure 4 contains a delete request to the paper database with both the id in the URL and the revision number as a parameter.

Graphical user interface, text, application, email

Description automatically generated

Figure 5 Getting the document ids with a key of the journal name

In figure 5, the view returned the ids of all documents with a key of the journal they belong to.

Graphical user interface, text, application, email

Description automatically generatedIn figure 6, the count-year view is called with a key of 2021. The value returned is the amount of documents which had that year in their year field.

Figure 6 Getting the count of the times a year appeared in a document

A picture containing text

Description automatically generated

Figure 7 First screen of application

The opening screen of the application, as seen in figure 7,displays the title of all documents in the data base. These are grouped by journal by calling the by-journal view, shown in figure 5. The Refresh button on this page gets the documents from the database again to get more up to date data. All database calls in the application have been isolated to the db-driver.py file.

Pressing the Remove button on any document will delete it from the database.

Graphical user interface, application

Description automatically generated

Figure 8 Add document screen of application

The Add Document button opens the Add Document screen as pictured in Figure 8. Here all data is entered. The Add button will validate the data. The API is called in the db\_driver file.

Graphical user interface, application

Description automatically generated

Figure 9 Detail view screen of document

Once added and refreshed, the document will appear in the main screen. Then the details button can be clicked. This brings up the details page, pictured in figure 7, with the details of the document in question filled out. Altering any field and pressing the Update button will update the document on the database via the db\_driver.

Graphical user interface, application, Teams

Description automatically generated

Figure 10 Which Year pop up menu

If the Year Count button is pressed, the pop up shown in figure 10 will appear requesting a year. The user must type a year in. An empty field will be treated as a wildcard. The input will be placed as a parameter into a call to the year-count map-reduce view.

Graphical user interface, application, chat or text message, Teams

Description automatically generated

Figure 11 Result pop up from the Which Year pop up

Once the call response is received it will display the number of documents that included the year selected.

# Drivers for CouchDB

## Requests

This library is only a REST API library providing the basic functionality for sending and receiving RESTFUL calls. It does have support for post, put, get, and delete but does not abstract away the basic data as it has no unique functionality for CouchDB. It is not limited however as any function exposed to the CouchDB API can be accessed. It can convert responses to JSON, but to send data as JSON, the python JSON module must be used. The JSON module can accept any dictionary. All python objects can have a dictionary of their attributes accessed so all python objects can be converted to JSON in this manner. This library is synchronous and will block execution. (Reitz, 2022)

## Flask CouchDB

Flask is a web framework. Flask CouchDB is an addon for Flask which is a driver for CouchDB. It does not require JSON objects as it implicitly converts python objects to JSON. It uses the python indexing syntax. It can create, update, remove and retrieve documents based on their key. It can also create views. By inheriting from Flask’s document object, database calls can be obfuscated with just a load and store function on any created object. The major limitation of this driver is it needs to be used in tandem to the Flask framework. It also has not been updated since July 2010. (Doraemon, 2010)

## CouchDB-python

CouchDB-python was a library for using CouchDB in Python. It is now archived with the last commit having been in 2018. Forks have been made but none have the same level of community around it. Officially, the maintainer has pointed people at python-Cloudant. (Ochtman, 2018) It used the python index syntax for accessing databases on the service and documents in the databases. As with others that used this syntax. It works of the keys: database name for database and ids for documents. However, this indexing can only be used to update, get, or delete. To create a new document, you must use the save function on the database object. The same index syntax can be used to access fields in the document. The python del keyword is used to remove documents. This library does not require JSON conversion as it only takes in and produces python objects. View creation and access are available in the driver. View results are stored in the module’s ViewResults object where rows can be access with keys or slices. (Ochtaman, 2010)

## Cloudant/Cloudant-SDK

Cloudant is IBM’s scalable cloud document storage. Python-Cloudant was a library to communicate with Cloudant service or CouchDB. Unlike other drivers, this one does not generate ids for documents automatically as it supports Cloudant’s key specifications. Yet it still uses the indexing syntax used in many other drivers in python. Views can be created but these must be grouped in design documents. This driver supports MongoDB like querying in addition to views and mapReduce normally available in CouchDB. Documents can be created from python objects however they must be converted and added to the database with a function as opposed to just inserting the object at an index in the database. Updates and gets do use the indexing syntax. Deletes require a function performed on the driver’s document object from the database object. (IBM, 2015)

This library was deprecated and archived with the lasted release being dated 26th of the 8th 2021. IBM Cloudant Python SDK replaced it as a client for working with Cloudant’s APIs on Python. Both can communicate with CouchDB as well. Documentation for IBM’s Cloudant does focus more on how to port a CouchDB database to Cloudant. (IBM, 2021)

## AIOCouch

This module allows for asynchronous access to a database. Once the connection object is step up, the CouchDB object can be used like a python dictionary. This first object has all the databases indexed by their name. These can be retrieved with the square brackets, typical python syntax. Similarly, the documents in a database object can be accessed via their index in the same way. Unlike requests, this UX is more designed to work in python, integrating well with common python syntax. It also works with python’s asynchronous system. Therefore, it will not lock resource while waiting for the response from the database. (Metriq, 2022)

# CouchDB on the cloudGraphical user interface, application Description automatically generated

To spin up CouchDB on Azure cloud, first an account must be created. Then a new resource must be made. Azure has 6 options for CouchDB as of the 13th of October 2022. Bitnami has a container or virtual machine. The virtual machine is free while the container has no price shown. AskForCloud LLC. has 2 options. Both options are virtual machines. A 1c/hr option running on Ubuntu 18.04 and a 2c/hr option running on Ubuntu 20.04. Apps4Rent LLC. also, virtual machine running Ubuntu 18.04 but their option only costs 0.5c/hr. VMLAB Inc. offers a virtual machine running Ubuntu 20.04 running CouchDB 3.2 for 3.3c/hr. (Microsoft, 2022)

Figure 12 Azure marketplace results for the query CouchDB

This guide will continue with the free Bitnami virtual machine.

Graphical user interface, text, application, email

Description automatically generated

Figure 13 Project details configuration page to the CouchDB virtual machine

A subscription and a region must be selected. Depending on whether the resource group selected has a virtual network or not, a virtual network may need to be created in the networking menu. Other details can be configured if the defaults do not suit the target use. Then the instance can be created and spun up.

Once the instance is ready, a public IP address is given by Azure to access the virtual machine externally. Security rules must be configured to allow inbound traffic on the database and Fauxton ports. (Olarewaju, 2022) The public IP address can be inserted into the connection string in a client or the client’s driver. From there, the cloud database can be treated like a local one.

CouchDB recommend using replication to transfer data from between CouchDB instances. (Apache, 2022) For a one-off transfer from an old instance to the new cloud instance, copying the data files may be more applicable. This only requires copying the .couch, and .shard if secondary indexes exist, files from the old machine to the new. This can be done via any file transfer protocol supported by the cloud provider such as azcopy command line utility for Azure. (Microsoft, 2022) The .ini file from the etc folder can be copied as well to maintain the configuration of the CouchDB server.

Authors note: Images of the database online, the data being transferred to the hosted database, or the client operating with the hosted database do not appear in this paper as the account used had been suspended. Those section relied on other sources to show accurate data and so may not be as up to date.

# Conclusion

This report found CouchDB to be suited for document storage in situations where accessibility is valuable. It can integrate with Python whether through convenient, well integrated drivers or just API requests. However, drivers could pose risks to long running projects due to how often they become no longer supported. Migrating these servers can be done just by coping files, to local or cloud hosted databases. There are multiple options for hosting CouchDB online are available meaning migration to the cloud is an opportunity for users of this database.

# References

Apache, 2022. *5.3. Backing up CouchDB — Apache CouchDB® 3.2 Documentation.* [Online]   
Available at: https://docs.couchdb.org/en/3.2.2-docs/maintenance/backups.html  
[Accessed 13 October 2022].

Doraemon, L. R., 2010. *Flask-CouchDB — Flask-CouchDB v0.1 documentation.* [Online]   
Available at: https://pythonhosted.org/Flask-CouchDB/  
[Accessed 13 October 2022].

IBM, 2015. *Getting started — python-cloudant 2.15.0 documentation.* [Online]   
Available at: https://python-cloudant.readthedocs.io/en/stable/index.html  
[Accessed 14 October 2022].

IBM, 2021. *cloudant/python-cloudant: A Python library for Cloudant and CouchDB.* [Online]   
Available at: https://github.com/cloudant/python-cloudant  
[Accessed 14 October 2022].

Metriq, 2022. *metricq/aiocouch: 🛋 An asynchronous client library for CouchDB 2.x and 3.x.* [Online]   
Available at: https://github.com/metricq/aiocouch  
[Accessed 12 October 2022].

Microsoft, 2022. *Marketplace - Microsoft Azure.* [Online]   
Available at: https://portal.azure.com/#view/Microsoft\_Azure\_Marketplace/MarketplaceOffersBlade/selectedMenuItemId/home/searchQuery/couchDB/searchInitiatedFrom/plusNewBladeSearchContext  
[Accessed 13 October 2022].

Microsoft, 2022. *Transfer data to or from Azure Files by using AzCopy v10 | Microsoft Learn.* [Online]   
Available at: https://learn.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-files?toc=%2Fazure%2Fstorage%2Ffiles%2Ftoc.json  
[Accessed 13 October 2022].

Ochtaman, D., 2010. *Introduction — couchdb-python 1.0 documentation.* [Online]   
Available at: https://couchdb-python.readthedocs.io/en/latest/  
[Accessed 14 October 2022].

Ochtman, D., 2018. *djc/couchdb-python: Python library for working with CouchDB.* [Online]   
Available at: https://github.com/djc/couchdb-python  
[Accessed 14 October 2022].

Olarewaju, D., 2022. *How to setup CouchDb on Azure: A step by step guide | Codementor.* [Online]   
Available at: https://www.codementor.io/@doyinolarewaju/how-to-setup-couchdb-on-azure-a-step-by-step-guide-nohmm1q2s  
[Accessed 13 October 2022].

Reitz, K., 2022. *Requests: HTTP for Humans™ — Requests 2.28.1 documentation.* [Online]   
Available at: https://requests.readthedocs.io/en/latest/  
[Accessed 12 October 2022].