A Guide to CouchDB Installation Configuration and Monitoring

https://www.monitis.com/blog/a-guide-to-couchdb-installation-configuration-and-monitoring/

Introduction

There is huge amount of documentation about CouchDB on the web. This article is not a complete guide to CouchDB. It's just a very brief description of CouchDB with examples which could be useful if you decide to try cluster building.
A little about CouchDB:
Basically, CouchDB is a schema-less document oriented database with a lot of features:

- *Document Storage*
- *ACID Semantics*
- *Map/Reduce Views and Indexes* (JavaScript based)
- *Distributed Architecture with Replication*
- *REST API*
- *Eventual Consistency*
- *Mobile access*

It also has a great web based management console called Futon.

Installation

The easiest way to install CouchDB is to use apt-get: *apt-get install couchdb.*

Notice that the packages in the repositories are not always the latest version. So, when we install some packages with apt-get, we risk losing new features, bug fixes and improvements.

Installation from sources is not the best way to install, however in this case we'll install CouchDB in this way.

The latest stable version for Linux is available at: http://www.apache.org/dyn/closer.cgi?path=/couchdb/releases/1.2.0/apache-couchdb-1.2.0.tar.gz

Select your mirror and proceed!

Before installation, we have to install some dependency packages to make CouchDB work properly.

```
add-apt-repository ppa:launchpad/ppa  #add the proper repository for libmozjs-dev and other packages
#install the dependencies
apt-get install libmozjs-dev
apt-get install libicu-dev
apt-get install libcurl4-openssl-dev
apt-get install build-essential erlang
# After all the dependencies successfully install, it is time to define where we will install from.
# There are 2 options
# The SVN repository or the last stable version from the official mirror.
# However, if you want to try installation from the svn repo, relax and type the commands below, otherwise do
wget –chttp://couchmirrorpath/<couch-version>.tar.gz
# cd <couch version>
# ./configure && make && sudo make install

svn co http://svn.apache.org/repos/asf/couchdb/trunk couchdb
cd couchdb
./bootstrap
./configure
make
sudo make install
make clean
```

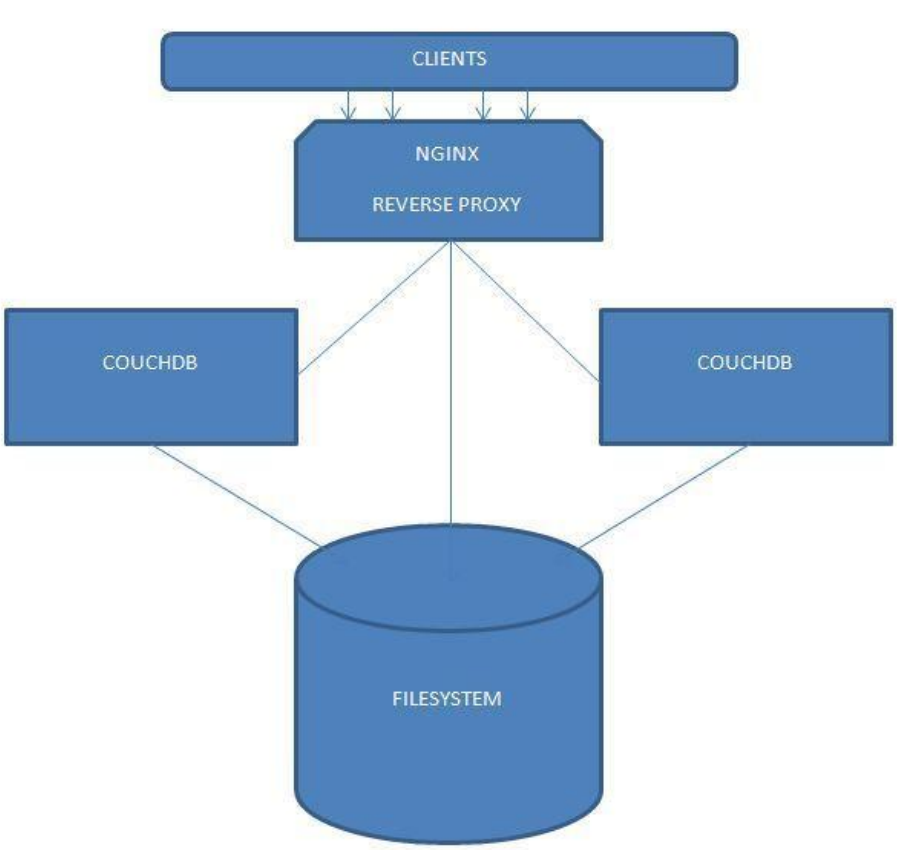
```
make distclean
sudo –i
# adding a user for couchdb
adduser --system --home /usr/local/var/lib/couchdb --
no-create-home --shell /bin/bash --group --gecos
"CouchDB Administrator" couchdb
# changing the owner to couchdb for couchdb’s
directories
chown -R couchdb:couchdb /usr/local/var/lib/couchdb
chown -R couchdb:couchdb /usr/local/var/log/couchdb
chown -R couchdb:couchdb /usr/local/var/run
chown -R couchdb:couchdb /usr/local/etc/couchdb
# changing permissions for couchdb’s directories
chmod -R 0770 /usr/local/var/lib/couchdb
chmod -R 0770 /usr/local/var/log/couchdb
chmod -R 0770 /usr/local/var/run
chmod -R 0770 /usr/local/etc/couchdb
# copying startup script to init.d/ to be available
within system boot process
cp /usr/local/etc/init.d/couchdb /etc/init.d/
update-rc.d couchdb defaults
# starting
/etc/init.d/couchdb start
```

Configuring CouchDB

After installation, CouchDB is ready to use. The default configuration is usually sufficient. But what if you want to make your database available from an external network? Or have to change the listen port because another instance of CouchDB is already using the default one? It’s possible, when more than one instance of CouchDB is running on the same machine.

Let discuss an example.

Assume we need to create a simple CouchDB failover cluster of 2 nodes.



Nginx (a very popular lightweight web server) will act as a reverse proxy. It's the front end for our small cluster, that is, it will handle user requests and redirect them to one of the CouchDB servers, depending on settings or an algorithm defined in the Nginx configuration file.

We have to create a configuration file for each instance of CouchDB:

Copy the *default.ini* to *couch_alpha.ini* and *couch_beta.ini*

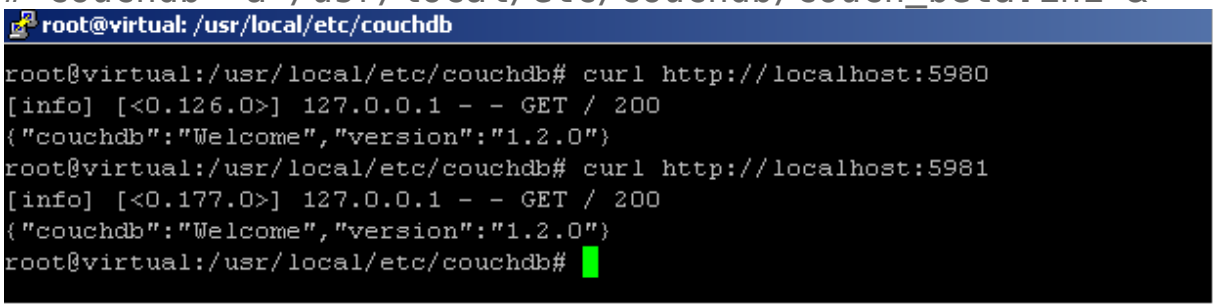
```
cp  /usr/local/etc/couchdb/default.ini  /usr/local/etc/couchdb/couch_alpha.ini
cp  /usr/local/etc/couchdb/default.ini  /usr/local/etc/couchdb/couch_beta.ini
```

Replace the port and ssl variable values in both files.

couch_alpha.ini must look like this (you can set other values for port and/or database_dir, this is just an example):

```
[httpd]
port = 5980
[ssl]
port = 6984
```

```
[couchdb]
database_dir = /usr/local/var/lib/couchdb/alpha
```
And *couch_beta.ini* must look like this:
```
[httpd]
port = 5981
[ssl]
port = 6985
[couchdb]
database_dir = /usr/local/var/lib/couchdb/beta
```
Create database directories for alpha and beta:
```
mkdir /usr/local/var/lib/couchdb/{alpha,beta}
```
Change the owner:
```
chown -R couchdb:couchdb /usr/local/var/lib/couchdb/
```
Start servers from the command line manually or modify the startup script in */etc/init.d/* to do it automatically every time the system boots
```
# Starting couchdb from command line
# couchdb -a /usr/local/etc/couchdb/couch_alpha.ini &
# couchdb -a /usr/local/etc/couchdb/couch_beta.ini &
```



Okay, both servers are up and running.
Now, it's time to configure Nginx.
Edit the nginx.conf file (usually it's located in */etc/nginx*) and add the following in the *http*section:
```
 upstream couchdb_cluster {
   server 127.0.0.1:5980;
   server 127.0.0.1:5981;
 }
 server {
   listen 80;
  server_name master.couch.local;
   location / {
     proxy_pass http://couchdb_cluster;
     break;
```

```
    }
  }
```

Add to your /etc/hosts file: *'127.0.0.1 master.couch.local'*

```
 echo '127.0.0.1 master.couch.local'  >> /etc/hosts
```

Restart Nginx.

Now let's create databases on both servers:

```
 curl -X PUThttp://localhost:5980/netangels
 curl -X PUThttp://localhost:5981/netangels
 Start  continuous replication between databases:
 curl -X POST -H 'Content-Type: application/json'
http://localhost:5980/_replicate -d
'{"source":"netangels", "target":"http://
localhost:5981/netangels", "continuous":true}'
 curl -X POST -H 'Content-Type: application/json'
http://localhost:5981/_replicate -d
'{"source":"netangels", "target":"http://
localhost:5980/netangels", "continuous":true}'
```

We have completed basic installation.

Let's check it!

Create a document on alpha server:

```
 #curl -X POST -H 'Content-Type: application/json'
http://localhost:5980/netangels/ -d '{"color":"red"}

{"ok":true,"id":"06b419bd3cdfdcda2472718756000ffb","rev
":"1-5f9b733004332774907f800eae6fd321d"}
```

Get the same document from beta server:

```
@virtual:/usr/local/etc/couchdb# curl http://localhost:5981/netangels/06b419bd3cdfdcda2472718756000ffb
fo] [<0.1603.0>] 127.0.0.1 - - GET /netangels/06b419bd3cdfdcda2472718756000ffb 200
id":"06b419bd3cdfdcda2472718756000ffb","_rev":"1-5f9b733004332774907f800eae6fd321d","color":"red"}
@virtual:/usr/local/etc/couchdb#
```

Replication is working.

Notice that continuous replication currently does not survive server restar t. Maybe in a future release this will be fixed.

Monitoring CouchDB

Your server is ready to serve! What now?

Now we are going to monitor our server.

There are many monitoring tools for monitoring CouchDB, but in this article we will look at only Monitis tools, as they are easy to use and offer good functionality.

Basic metrics
CouchDB itself provides necessary statistical data via the REST interface, enough to make an educated guess about the server's health. By requesting http://server/_stats, we get information about the open databases count, request failures, etc., and all in JSON format!
Following is a list of the metrics available via the REST: http_codes and http_methods are not actually metrics, but they provide useful information such as invalid queries count, internal errors count, non-existent documents count and detailed statistics for each HTTP method.
open_databases – number of open databases since last restart
open_os_files – number of file descriptors CouchDB has open at the moment
database_writes – number of times a database was changed since last restart
database_reads – number of times a document was read from a database since last restart
request_time – the time elapsed from the start of the request, afterMochiWeb has passed it to CouchDB
clients_requesting_changes – number of clients for continuous changes since last restart
requests – number of HTTP requests since last restart
view_reads – number of view reads since last restart
temporary_view_reads – number of temporary view reads since last restart
bulk_requests – number of bulk requests since last restart
We will use Monitis M3 to process all of these metrics.

Installing Monitis M3

First of all clone the github repository:

```
git clone git@github.com:monitisexchange/Monitis-Linux-Scripts.git
```

This will clone the whole repository. We need only M3v3 part. Install MonitisMonitormanager according to the installation recommendations.

```
cd Monitis-Linux-Scripts/M3v3/MonitisMonitorManager
perl Makefile.PL
```

```
make
sudo make install
```

Edit M3Templates.pm and add your APIKEY and SECRETKEY

Check your template by running:

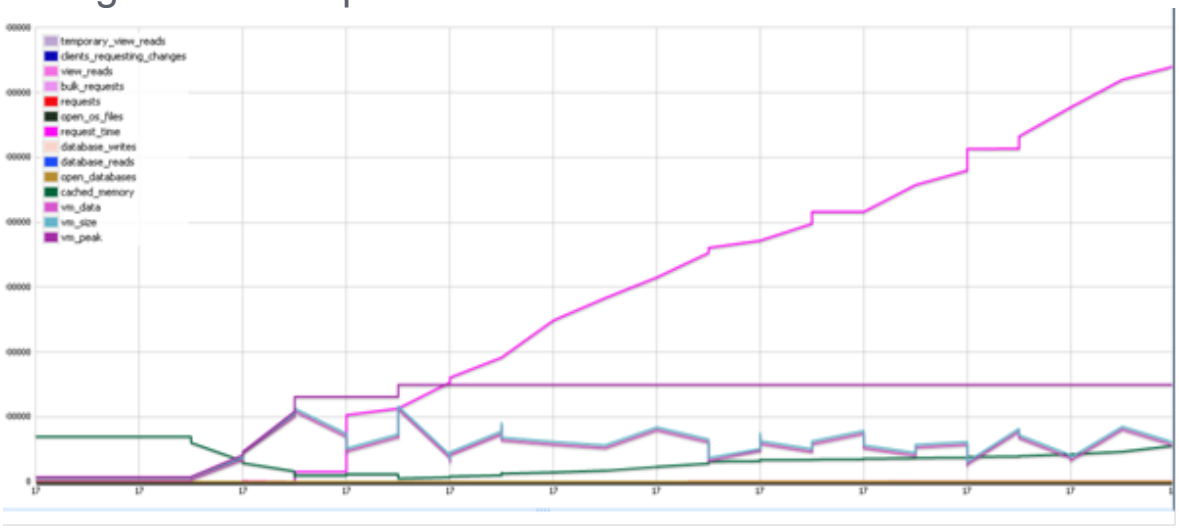
```
monitis-m3 --dry-run --once /path/to/monitis/couchdb_monitor.xml
```

Ensure all is okay, then do:

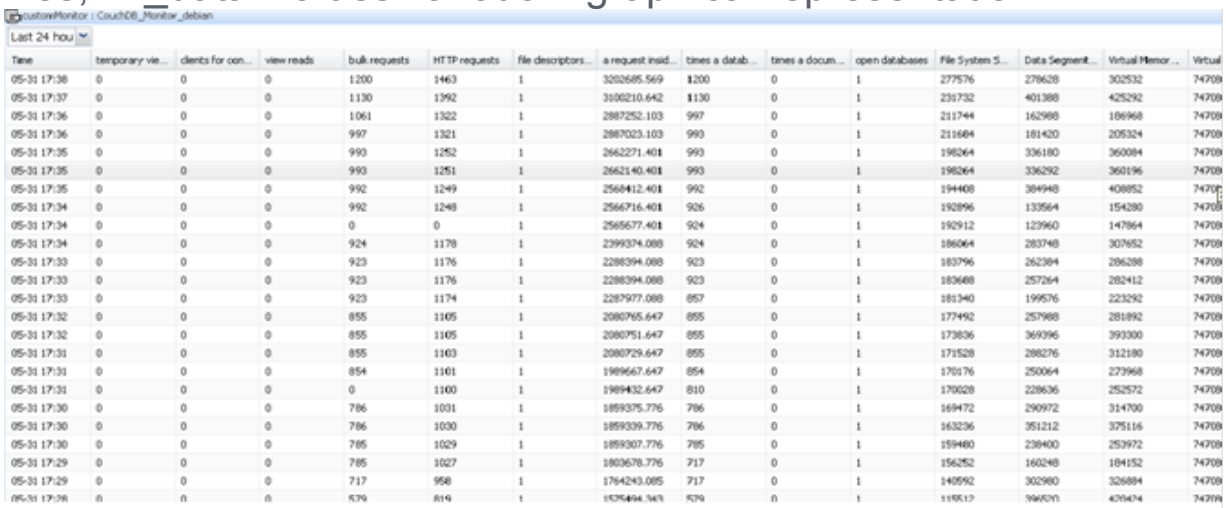
```
monitis-m3 /path/to/monitis/couchd_template.xml
```

Now you should get data in your Monitis.com dashboard.

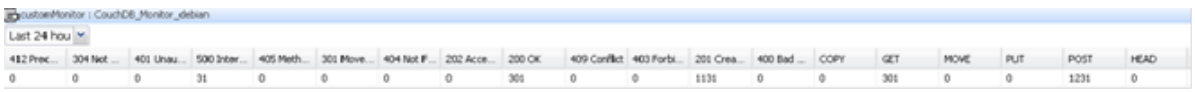
The graphics below show the variation in several metric's values during a one hour period:



virtual memory peak, view reads, cached_memory, open os files,vm_data values variation graphical representation



all metric's values variation table view



additional data set for a single row.