CartoDB Installation for RHEL 6

This was based of doing a fresh GUI install of RHEL6 and selecting the “Web Development” server option during the install. The necessary files are located in CartoDBInstall.tar.gz

1. Linux Preinstalls
	1. $ sudo yum remove boost\*
	2. $ sudo yum remove postgresql\*
	3. $ sudo yum update
	4. $ yum install libtiff\*
	5. copy unp file from installation packages to /usr/local/bin as root
	6. install libyaml and libyaml-devel from libyaml folder.
	$ sudo rpm –i libyaml-0.1.4-1.el6.rf.(version).rpm
	$ sudo rpm –i libyaml-devel-0.1.4-1.el6.rf.(version).rpm
	Use x86\_64 for 64 bit installations and i686 for 32 bit.
	7. install libicu-devel from libicu-devel folder. For some reason this package no longer comes with RHEL and is not available through yum
	$ sudo rpm –i libicu-devel-4.2.1-9.1.el6\_2.(version).rpm
	Use x86\_64 for 64 bit installations and i686 for 32 bit.
	8. install libtool-ltdl-devel from libtool-ltdl-devel folder. For some reason this package no longer comes with RHEL and is not available through yum
	$ sudo rpm –i libtool-ltdl-devel-2.2.6-15.5.el6.(version).rpm
	Use x86\_64 for 64 bit installations and i686 for 32 bit.
	9. as root - $ echo ‘usr/local/lib’ >> /etc/ld.so.conf.d/local.conf
2. Ruby
	1. $ cp ruby-1.9.3-p194.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf ruby-1.9.3-p194.tar.gz
	4. $ cd ruby-1.9.3-p194
	5. $ ./configure
	6. $ make
	7. $ make test
	8. $ sudo make install
	9. as root ->$ gem install bundler
3. Node.js
	1. $ cp node-v0.4.11.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf node-v0.4.11.tar.gz
	4. $ cd node-v0.4.11
	5. $./configure
	6. $ make
	7. $ make test
	8. $ sudo make install
	9. as root – $ echo ‘export NODE\_PATH=/usr/local/lib/node’ >> /etc/profile.d/local.sh
	10. source /etc/profile.d/local.sh in all open terminals
4. npm
	1. as root, run install.sh from npm folder
5. PostgreSQL
	1. install pgdg rpm from pgdg folder
	$ sudo rpm –i pgdg-redhat91-9.1-5.noarch.(version).rpm
	Use x86\_64 for 64 bit installations and i686 for 32 bit.
	2. $ sudo yum install libpq\*
	3. $ sudo yum install postgresql91\*
	4. $ sudo service postgresql-9.1 initdb
	5. $ sudo service postgresql-9.1 start
	6. $ sudo chkconfig postgresql-9.1 on
	7. as root $ echo ‘PATH=/usr/pgsql-9.1/bin:$PATH’ >> /etc/profile.d/local.sh
	8. source /etc/profile.d/local.sh in all open terminals
	9. $ sudo su – postgres
	10. $ cd /usr/pgsql-9.1/bin/
	11. $ ./createdb (user) <- username you’re currently using (don’t use root)
	12. $ ./createuser (user) <- same user as above. Superuser? Y
	13. as root $ echo ‘/usr/pgsql-9.1/lib’ >> /etc/ld.so.conf.d/local.conf
	14. $ sudo ldconfig –v <- make sure postgres libs are found in /usr/pgsql-9.1/lib
	15. $ sudo vi /var/lib/pgsql/9.1/data/pg\_hba.conf
		1. Change these lines
			1. local all all peer -> change peer to trust
			2. host all all 127.0.0.1/32 ident ->change ident to trust
			3. host all all ::1/128 ident ->change ident to trust
		2. This is just for testing purposes while we set the server up. You will want to lock down the security settings on postgres once the server is ready for production.
6. GEOS
	1. $ cp geos-3.3.5-tar.bz2 to /tmp
	2. $ cd /tmp
	3. $ bunzip2 geos-3.3.5.tar.bz2
	4. $ tar –xvf geos-3.3.5.tar
	5. $ cd geos-3.3.5
	6. $ ./configure CFLAGS=”-O1” CXXFLAGS=”-O1” <- (letter O’s)
	7. $ make
	8. $ make check
	9. $ sudo make install
	10. $ sudo ldconfig –v <- make sure geos libraries made it into /usr/local/lib
7. GDAL
	1. $ cp gdal-1.9.1.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf gdal-1.9.1.tar.gz
	4. $ cd gdal-1.9.1
	5. $ ./configure
	6. $ make
	7. $ make install
	8. $ sudo ldconfig –v <- make sure gdal libraries made it into /usr/local/lib
8. JSON
	1. $ cp json-c-0.9.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf json-c-0.9.tar.gz
	4. $ cd json-c-0.9
	5. $ ./configure
	6. $ make
	7. $ make check
	8. $ sudo make install
	9. $ sudo ldconfig –v <- make sure json libraries made it into /usr/local/lib
9. PROJ4
	1. $ cp proj-4.8.0.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf proj-4.8.0.tar.gz
	4. $ cd proj-4.8.0
	5. $ ./configure
	6. $ make
	7. $ make check
	8. $ sudo make install
	9. $ sudo ldconfig –v <- make sure proj libraries made it into /usr/local/lib
10. PostGIS
	1. $ cp postgis-2.0.1.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf postgis-2.0.1.tar.gz
	4. $ cd postgis-2.0.1
	5. $ ./configure <- make sure all necessary dependencies are found.
	6. $ make
	7. $ make check
	8. $ sudo make install
	9. run postgis\_template\_setup.sh as the postgres user you created earlier
11. Redis
	1. $ cp redis-2.4.16.tar.gz /tmp
	2. $ tar –xzvf redis-2.4.16.tar.gz
	3. $ cd redis-2.4.16
	4. $ make
	5. $ make test
	6. $ sudo make install
	7. $ sudo mkdir /var/log/redis
	8. $ sudo chmod a+w /var/log/redis
	9. $ sudo mkdir /var/lib/redis
	10. $ sudo chmod a+w /var/lib/redis
12. Python Requirements
	1. $ sudo easy\_install pip
	2. $ sudo pip install –r python\_requirements.txt
13. Boost
	1. $ cp boost\_1\_50\_0.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf boost\_1\_50\_0.tar.gz
	4. $ cd boost\_1­\_50\_0
	5. $ ./bootstrap.sh
	6. $ ./b2
	7. $ sudo ./b2 install
	8. $ sudo ldconfig –v <- make sure boost libs are found in /usr/local/lib
14. Mapnik
	1. $ cp mapnik-mapnik-v2.0.0-0-g5b4c20e.tar.gz /tmp
	2. $ cd /tmp
	3. $ tar –xzvf mapnik-mapnik-v2.0.0-0-g5b4c20e.tar.gz
	4. $ cd mapnik-mapnik-745fcb5
	5. $ ./configure <- Make sure all necessary dependencies are found.
	6. $ python scons/scons.py
	7. $ sudo python scons/scons.py install
	8. $ python test/run\_tests.py –q <- Some errors will be normal (out of 148 tests, I had 9 errors). This will be related to some plugins of gdal/geos that may not be there.
15. CartoDB Directory Setup
	1. Create a directory you want to run CartoDB from. I used /opt/cartodb using the steps below.
	2. $ sudo mkdir /opt/cartodb
	3. $ sudo chown (user):(user) /opt/cartodb
	4. $ cd /opt/cartodb
16. CartoDB SQL API
	1. as root -> npm install –g mocha@1.2.1
	2. as root -> npm install -g expresso
	3. $ cp CartoDB-SQL-API.tar.gz /opt/cartodb
	4. $ cd /opt/cartodb
	5. $ tar –xzvf CartoDB-SQL-API.tar.gz
	6. $ cd CartoDB-SQL-API
	7. $ npm install
	8. start redis server on new terminal -> $ redis-server
	9. $ cd test
	10. $ ./prepare\_db.sh
	11. $ expresso acceptance/app.test.js <- When output stops, hit ctrl-c for results
	12. $ expresso acceptance/app.auth.test.js <- When output stops, hit ctrl-c for results
	13. $ expresso test/unit\*.js <- When output stops, hit ctrl-c for results
	14. kill redis using ctrl-c on that terminal
17. CartoDB Windshaft
	1. $ cp Windshaft-cartodb.tar.gz /opt/cartodb
	2. $ cd /opt/cartodb
	3. $ tar –xzvf Windshaft-cartodb.tar.gz
	4. $ cd Windshaft-cartodb
	5. $ npm install
	6. $ make check
18. CartoDB
	1. $ cp cartodb.tar.gz /opt/cartodb
	2. $ cd /opt/cartodb
	3. $ tar –zxvf cartodb.tar.gz
	4. $ cd cartodb
	5. $ bundle install –binstubs
	6. $ redis-server <- start redis on another terminal.
	7. add entries to /etc/hosts needed in development
	8. $ echo "127.0.0.1 admin.localhost.lan" | sudo tee -a /etc/hosts
	9. $ echo "127.0.0.1 admin.testhost.lan" | sudo tee -a /etc/hosts
	10. $ echo "127.0.0.1 [mysubdomain].localhost.lan" | sudo tee -a /etc/hosts <- where mysubdomain is the user account you’ll create for cartodb.
	11. $ sh script/create\_dev\_user [mysubdomain] <- where mysubdomain is the user account you used in the last step. Create a password and admin password for the account.
	12. kill redis using ctrl-c on that terminal.
	13. $ vi Procfile
	14. Comment out redis line if it’s not already. For some reason bundle won’t start redis properly with the rest of the stack.
19. Running CartoDB
	1. open a terminal. $ redis-server /opt/cartodb/cartodb/config/redis.conf
	2. open another terminal.
	3. $ cd /opt/cartodb/cartodb
	4. $ bundle exec foreman start –p 3000
	5. open browser and go to [mysubdomain].localhost.lan:3000 <- where mysubdomain is the user account you used in the last step.
	6. If it runs…break down in tears of joy….if it fails…break down in tears of anguish.