# Installation and maintenance manual

[Installation and maintenance manual 1](#_Toc448907992)

[Software initial installation 1](#_Toc448907993)

[Prerequisites installation. 1](#_Toc448907994)

[Source code acquisition and setup. 4](#_Toc448907995)

[Software maintenance 7](#_Toc448907996)

[Typical routines 7](#_Toc448907997)

[Configuration file options 9](#_Toc448907998)

## Software initial installation

Smart Analytics solution designed to run on the IBM Softlayer cloud. For demonstration purpose the solution can be installed to virtual environment like VirtualBox.

IBM Softlayer cloud provides RHEL environment with IBM BigInsights components. The components include:

* Ambari
* HDFS
* Kafka
* Spark
* PostgreSQL

Installation to VirtualBox requires pre-installed CentOS 7.

Nuances of installation specific for cloud or virtual environment versions are describes in text.

Commands executed as root are marked with ‘# ’

Commands that should be run from ordinal user (‘smartanalytics’) are marked with ‘$ ’

### Prerequisites installation.

1. Install git

# yum install git

1. Create file /etc/yum.repos.d/mongodb-org-3.2.repo with following content

[mongodb-org-3.2]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86\_64/

gpgcheck=1

enabled=1

1. Install mongodb

# yum install mongodb-org

1. Install PostgreSQL (only for VirtualBox installation)

# yum install python-psycopg2 postgresql-server postgresql

1. Install Python tools

# rpm -iUvh http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-5.noarch.rpm

# yum -y install python-pip

1. Install required python libraries

# yum install numpy

# yum install scipy

# yum install python-pandas

# pip install lifelines

# yum install -y gcc

# yum install -y gcc-c++

# pip install notebook

# pip install py2neo

# pip install twitter

# pip install pymongo

# pip install flask flask\_sockets

# pip install requests

1. Do initial configuration for PostgreSQL

# postgresql-setup initdb

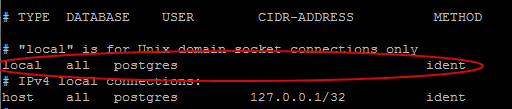
# /etc/rc.d/init.d/postgresql start

# sudo -u postgres psql -U postgres -c "CREATE DATABASE twitter"

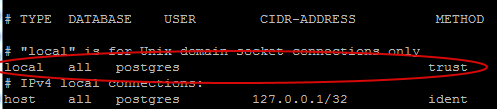
# sudo -u postgres psql -U postgres -c "CREATE DATABASE pci"

1. Edit file /var/lib/pgsql/data/pg\_hba.conf

Find this line:



And edit with replacing word “ident” to “trust”. After edit line should look like on this screenshot:



Save file and restart PostgreSQL with command:

# /etc/rc.d/init.d/postgresql restart

1. Install jdbc library for postgresql

# mkdir /usr/share/java/

# curl -o /usr/share/java/postgresql-9.4.1208.jre6.jar https://jdbc.postgresql.org/download/postgresql-9.4.1208.jre6.jar

1. Create file /etc/yum.repos.d/neo4j.repo with following content

[neo4j]

name=Neo4j Yum Repo

baseurl=http://yum.neo4j.org

enabled=1

gpgcheck=1

1. Install Neo4j

# rpm --import http://debian.neo4j.org/neotechnology.gpg.key

# yum install neo4j-2.3.2

# mkdir /var/lib/neo4j

1. Install supervisor daemon

# yum install supervisor

1. Install Predictive intelligence prerequisites and libs:

# yum install java-1.8.0-openjdk java-1.8.0-openjdk-headless

# pip install pycorenlp nltk

# yum install scipy

# pip install -U scikit-learn

### Source code acquisition and setup.

1. Create SSH keypair and register public key as described in “Setting up local git repository for Windows.docx”. After registering public key, clone repository from IBM cluster:

$ git clone git@10.170.9.216:smartanalytics

git will create folder “smartanalytics” in current directory. This directory will contain all sources and data files needed to launch application.

1. Install Stanford’s CoreNLP library:

$ cd smartanalytics/data\_analysis/PCI

$ wget http://nlp.stanford.edu/software/stanford-corenlp-full-2015-12-09.zip

$ unzip stanford-corenlp-full-2015-12-09.zip

$ rm stanford-corenlp-full-2015-12-09.zip

$ cd ../../..

1. Copy Neo4j database with twitter user graph to /var/lib/neo4j

$ sudo -i

# cp smartanalytics/data\_marts/neo4j\_graph/neo4j\_db.zip /var/lib/neo4j/

# cd /var/lib/neo4j

# unzip neo4j\_db.zip

# chown -R neo4j:neo4j /var/lib/neo4j

1. Now we can start neo4j

# chkconfig --add neo4j

# chkconfig neo4j --level 345 on

# /etc/rc.d/init.d/neo4j start

1. Neo4j requires password change before first use. Change password with following commands

$ curl -H "Content-Type: application/json" -X POST -d '{"password":"my\_neo4j"}' -u neo4j:neo4j http://localhost:7474/user/neo4j/password

$ curl -H "Content-Type: application/json" -X POST -d {"password":"neo4j"}' -u neo4j:my\_neo4j http://localhost:7474/user/neo4j/password

1. Run application initialization scripts:

$ cd smartanalytics/app/clf

$ sh ./init\_pyfiles.sh

$ sh ./init\_database.sh

1. Configure supervisor daemon

Create file /etc/ supervisord.conf with following content:

[unix\_http\_server]

file=/tmp/supervisor.sock ; (the path to the socket file)

[supervisord]

logfile=/tmp/supervisord.log ; (main log file;default $CWD/supervisord.log)

logfile\_maxbytes=50MB ; (max main logfile bytes b4 rotation;default 50MB)

logfile\_backups=10 ; (num of main logfile rotation backups;default 10)

loglevel=debug ; (log level;default info; others: debug,warn,trace)

pidfile=/tmp/supervisord.pid ; (supervisord pidfile;default supervisord.pid)

nodaemon=false ; (start in foreground if true;default false)

minfds=1024 ; (min. avail startup file descriptors;default 1024)

minprocs=200 ; (min. avail process descriptors;default 200)

user=smartanalytics ; (default is current user, required if root)

[rpcinterface:supervisor]

supervisor.rpcinterface\_factory = supervisor.rpcinterface:make\_main\_rpcinterface

[supervisorctl]

serverurl=unix:///tmp/supervisor.sock ; use a unix:// URL for a unix socket

[include]

files = supervisord.d/\*.ini

1. Configure supervisor daemon to handle application startup

Create file /etc/supervisord.d/clf\_web.ini

[program:clf\_web]

directory=/home/user/smartanalytics/app/clf

command=python2.7 \_\_init\_\_.py

autostart=true

autorestart=true

Create file /etc/supervisord.d/data\_ing.ini

[program:data\_ing]

directory=/home/user/smartanalytics/local\_version

command=python2.7 data\_ingestion.py

autostart=true

autorestart=true

Create file /etc/supervisord.d/corenlp.ini

[program:corenlp]

directory=/home/user/smartanalytics/data\_analysis/PCI/stanford-corenlp-full-2015-12-09

command=java -mx4g -cp "\*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer

autostart=true

autorestart=true

1. Enable supervisor daemon

# chkconfig --add supervisord

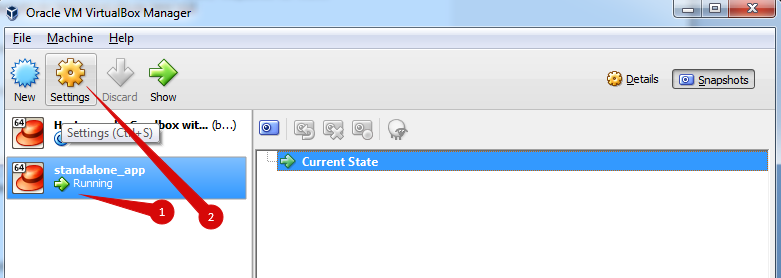
# chkconfig supervisord --level 345 on

# /etc/rc.d/init.d/supervisord start

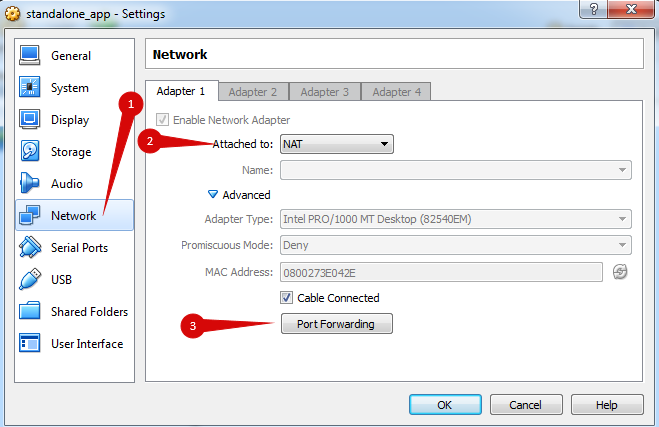
1. (Only for VirtualBox version) Configure port forwarding for virtual machine

To connect to services that is ran inside VirtualBox you should configure port forwarding.

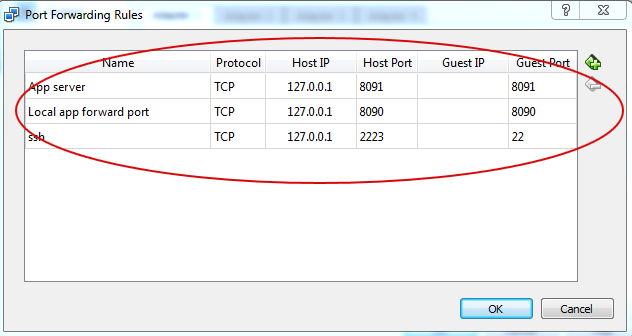
In VirtualBox select right image and press “Settings” button



In “Settings” window, select “Network” and check that “Attached to” is set to “NAT”. Then press “Port forwarding button”



In “Port Forwarding” window configure port forwarding as shown on picture:



Port 8091 is used for web application services.

Port 8090 is used for accessing IPython notebook (do not started by default).

Port 22 is used for SSH connectivity.

Congratulations! You now have the solution installed and ran.

You can connect to web service on your local machine with <http://127.0.0.1:8091/>

For cluster version use http://<cluster\_address>:8091/

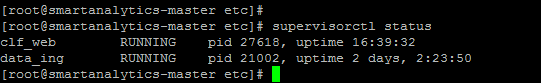
## Software maintenance

### Typical routines

1. Checking that software services are up:

# supervisorctl status

Example:

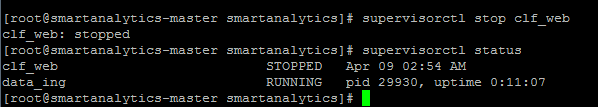


1. Manually stopping & starting service:

# supervisorctl stop clf\_web

# supervisorctl start clf\_web

Example:

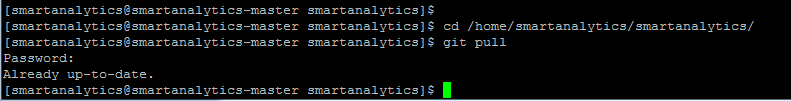


1. Update software from github repository:

$ cd /home/smartanalytics/smartanalytics/

$ git pull

Example:

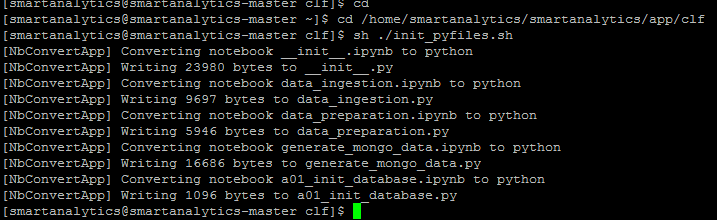


When you receive updates to core scripts such as “app/clf/\_\_init\_\_.ipynb” or “local\_version/data\_ingestion.py”, you should update locally created scripts and restart services

$ cd /home/smartanalytics/smartanalytics/app/clf

$ sh ./init\_pyfiles.sh

Example:



Solution services should be restarted after regeneration of “.py” files

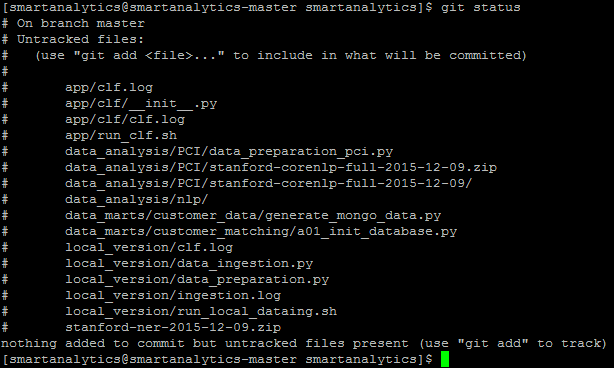
# supervisorctl restart clf\_web

# supervisorctl restart data\_ing

1. Checking for local changes in repository

$ git status

Example:



1. Checking postgresql database for saved records

$ psql -U postgres -d twitter

twitter=# select \* from stream;

1. Checking mongodb for customer records

$ mongo

MongoDB shell version: 3.2.4

connecting to: test

> show dbs

customer\_database 0.008GB

local 0.000GB

> use customer\_database

switched to db customer\_database

> db.customers.find({'customerId' : 1}).limit(1).pretty()

… JSON output returned …

### Configuration file options

Solution has configuration file in JSON format. All possible parameters described in table below.

|  |  |  |
| --- | --- | --- |
| № | Option | Description |
| 1 | token | This four parameters are used for OAuth authentication with twitter. They can be found at <https://apps.twitter.com/> |
| 2 | token\_secret |
| 3 | consumer\_key |
| 4 | consumer\_secret |
| 5 | twitter\_db | Name of database with twitter data (tweet stream, users) |
| 6 | neo4j | URL for accessing neo4j database |
| 7 | gmail\_user | Gmail user name |
| 8 | gmail\_pass | Gmail password |
| 9 | gmail\_recipient | List of recipients for emails from solution |