

Table of Contents

Table of Contents	1
ASPARC STARTER DEVOPS GUIDE	2
1. WAY OF WORKING	2
1.1. Issues handling	2
1.2. Documentation	2
2. NAMING CONVENTIONS	2
2.1. Bash scripts	2
2.1.1. Dirs naming conventions	2
2.1.2. Attempt to not mix code from different languages / run-times in the same dirs	2
2.1.3. Root Dirs naming conventions	2
2.1.4. Bash scripts naming conventions	2
2.2. Scala code capitalization styles and naming conventions	2
2.2.1. Pascal case usage	2
2.2.2. Camel Case usage	3
3. INSTALLATIONS AND CONFIGURATIONS	3
3.1. Install Java Development Kit 1.8	3
3.1.1. Configure java_home	3
3.1.2. Verify the JDK installation and configuration	3
3.2. Install Scala	3
3.3. Install sbt	3
3.4. Install apache spark	3
3.4.1. Download the latest stable Apache Spak package	3
3.4.2. Unpack and deploy	4
3.4.3. Add env vars	4
3.4.4. Verify the installation	4
3.5. Configure the Ubuntu repositories	4
3.6. Add the media keys	5
3.7. Install the postgre package with apt	5
3.8. Change the postgre user password	5
3.8.1. start the postgresSQL	5
3.8.2. Start the psql client as the postgres shell user	5
3.8.3. Create the pgsq user	6
3.8.4. add the uuid generation capability enabling extension	6
3.8.5. Install the dblink extension as follows	6
3.9. Install the perl modules (optional)	6
3.10. Install and configure Hadoop	6
3.10.1. Add the hadoo group	7
3.10.2. Add the hadoo user	7
3.10.3. Configure the ssh keys for the hduser	7
3.10.4. Fetch and install hadoop	7
3.10.5. Edit the hadoop-env.sh file	8
3.10.6. create the hadoop data dir	8
3.10.7. copy the map-reg.site.template file	8
3.10.8. Edit the hdfs-site.xml	8
3.10.9. format the hadoop file system	9
3.10.10. start the single node cluster	9
3.10.11. Verify that the everything is up-and-running	10
4. OPERATIONS	10
5. RUN THE EXAMPLES	10
6. INFORMATION SOURCES	10
6.1. Overall tutorials	10

ASPARC STARTER DEVOPS GUIDE

1. WAY OF WORKING

1.1. Issues handling

Each proper time spent on time saves 10 times more in execution, thus the tasks and activities related to this tool are tracked via the issue-tracker tool:

<https://github.com/YordanGeorgiev/issue-tracker>

and could be found @:

https://docs.google.com/spreadsheets/d/1-oYPtBM8PG_FUogk40RDmcM_Xzq91Tb81Zlyi0cMwYQ/edit#gid=135774576

1.2. Documentation

The purpose of the tool is to "grasp the concept of apache spark", thus a proper documentation set is created as well.

2. NAMING CONVENTIONS

2.1. Bash scripts

2.1.1. Dirs naming conventions

The dir structure should be logical and a person navigating to a dir should almost understand what is to be find in thre by its name. While creating new dirs the main principle is that the more general the subject of the dir the upper in the dir hierarchy it should stay.

2.1.2. Attempt to not mix code from different languages / run-times in the same dirs

Avoid as much as possible the mixing code from different languages / run-times in the same dirs

2.1.3. Root Dirs naming conventions

The root dirs and named as follows:

bin - contains the produced binaries for th project

cnf - for the configuration

dat - for the data of the app

lib - for any external libraries used

src - for the source code of the actual projects and subprojects

2.1.4. Bash scripts naming conventions

Do not use capital letters - they are too noisy.

2.2. Scala code capitalization styles and naming conventions

2.2.1. Pascal case usage

Use in application wide global variables

```
val ProductInstanceDir
```

2.2.2. Camel Case usage

Use in local class variables

3. INSTALLATIONS AND CONFIGURATIONS

3.1. Install Java Development Kit 1.8

Install Java Development Kit 1.8 as follows:

```
# update your Ubuntu repositories
sudo apt-get update
# install the openjdk
sudo apt-get install -y openjdk-8-jdk
```

3.1.1. Configure java_home

Configure java_home env var to the the java_opts file.

```
echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64' >> ~/.java_opts.host-name
```

3.1.2. Verify the JDK installation and configuration

Verify the JDK installation and configuration as follows:

```
# and verify
java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

3.2. Install Scala

The scala libs will be installed with the sbt build tool.

3.3. Install sbt

Install sbt scala build tool by following the instructions in the following url:

<http://www.scala-sbt.org/0.13/docs/Installing-sbt-on-Linux.html>

```
echo "deb https://dl.bintray.com/sbt/debian/" | sudo tee -a /etc/apt/sources.list.d/sbt.list
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2EE0EA64E40A89B84B2DF73499E82A75642AC823
sudo apt-get update
sudo apt-get install sbt
which sbt
```

3.4. Install apache spark

3.4.1. Download the latest stable Apache Spak package

Download the spak package with curl as follows:

```
export dir=/vagrant/pckgs/apache
```

```
mkdir -p $dir ; cd $dir
curl -O https://d3kbcqa49mib13.cloudfront.net/spark-2.2.0-bin-hadoop2.7.tgz
```

3.4.2. Unpack and deploy

Download the spak package with curl as follows:

```
mv -v spark-2.2.0-bin-hadoop2.7/ spark
mv -v spark /usr/lib/
sudo mv -v spark /usr/lib/
```

3.4.3. Add env vars

Add the following env vars

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export SPARK_HOME=/usr/lib/spark
export PATH=$PATH:$SPARK_HOME

# reload the env vars
source ~/.profile_opts
```

3.4.4. Verify the installation

Verify the installation by startin the spark shell

```
spark-shell

Spark context available as 'sc' (master = local[*], app id = local-1505242880618).

Spark session available as 'spark'.

Welcome to

      ____      _
     / __ \    / \  _ __| |_ / __ \
    _ \ / __ \  / _ \| '__| __| / __ \
   ___ \ \_/ \| |_) | | | | |_\ \_/ \|
  / ___ \___ \_/___|_|_|_|_|_\ \___ \
 /___ \___ \_/___|_|_|_|_|_\ \___ \
      /___ \_/___|_|_|_|_|_\ \___ \
          /___ \_/___|_|_|_|_|_\ \___ \

      version 2.2.0

      /___ \_/___|_|_|_|_|_\ \___ \

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

3.5. Configure the Ubuntu repositories

Configure the Ubuntu repositories

```
sudo add-apt-repository "deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main"

sudo apt-get update

sudo apt-get install postgresql-9.6
```

3.6. Add the media keys

Add the media keys as follows:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

3.7. Install the postgres package with apt

Install the postgres package with apt

```
# update your repos
sudo apt-get update

# install the postgresql binary
sudo apt-get install postgresql postgresql-contrib

# enable postgres
sudo update-rc.d postgresql enable
```

3.8. Change the postgres user password

Configure the Ubuntu repositories

```
sudo passwd postgres
# Type a pw - add to your password manager !!!

# and verify
su - postgres
```

3.8.1. start the postgresSQL

Start the postgresSQL by issuing the following command

```
sudo /etc/init.d/postgresql start
```

3.8.2. Start the psql client as the postgres shell user

Start the psql client as the postgres shell user

source:

<http://dba.stackexchange.com/a/54253/1245>

```
sudo su - postgres
# start the psql client
psql

# the psql prompt should appear as
# postgres=#

# list the databases
\l

#and quit
\q
```

3.8.3. Create the pgsq user

Create the pgsq user and grant him the privileges to create dbs and to connect to the postgres db. You could alternatively configure different way of authentication according to the options provided in this [stackoverflow](http://stackoverflow.com/a/9736231/65706) answer:

<http://stackoverflow.com/a/9736231/65706>

```
# create the pgsq user to be the same as the shell
# user you are going to execute the scripts with
sudo su - postgres -c "psql -c 'CREATE USER '$USER' ;'"

# grant him the privileges
sudo su - postgres -c "psql -c 'grant all privileges on database postgres to '$USER' ;'"

# grant him the privilege to create db's
sudo su - postgres -c "psql -c 'ALTER USER '$USER' CREATEDB;'"

sudo su - postgres -c "psql -c 'select * from information_schema.role_table_grants
where grantee='''$USER''';'"
```

3.8.4. add the uuid generation capability enabling extension

add the uuid generation capability enabling extension

```
sudo su - postgres -c "psql template1 -c 'CREATE EXTENSION IF NOT EXISTS \"uuid-osspl\";'"

sudo su - postgres -c "psql template1 -c 'CREATE EXTENSION IF NOT EXISTS \"pgcrypto\";'"
```

3.8.5. Install the dblink extension as follows

Install the dblink extension as follows

```
sudo su - postgres -c "psql template1 -c 'CREATE EXTENSION IF NOT EXISTS \"dblink\";'"
```

3.9. Install the perl modules (optional)

Install the perl module by first installing the server development package

```
# check which server development packages are available
sudo apt-cache search postgres | grep -i server-dev | sort

# install it
sudo apt-get install -y postgresql-server-dev-9.6

# install the DBD::Pg module
sudo perl -MCPAN -e 'install DBD::Pg'

sudo perl -MCPAN -e 'Tie::Hash::DBD'
```

3.10. Install and configure Hadoop

This section has been created largely by following this tutorial:

http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_16_04_single_node_cluster.php

3.10.1. Add the hadoop group

Add the hadoop Linux group as follows:

```
export group=hadoop
export gid=1001
sudo groupadd -g "$gid" "$group"
sudo cat /etc/group | grep --color "$group"
```

3.10.2. Add the hadoop user

Add the hduser Linux user as follows:

```
export user=hduser
export uid=1001
export home_dir=/home/$user
export desc="the hadoop group"
#how-to add an user
sudo useradd --uid "$uid" --home-dir "$home_dir" --gid "$group" \
--create-home --shell /bin/bash "$user" \
--comment "$desc"
sudo cat /etc/passwd | grep --color "$user"
groups "$user"
```

3.10.3. Configure the ssh keys for the hduser

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus this local machine.

```
sudo su - hduser

# create the ssh keys of rsa type
ssh-keygen -t rsa

# add localhost to the trusted hosts
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

# verify that ssh works
ssh localhost
```

3.10.4. Fetch and install hadoop

Fetch and install hadoop as follows:

```
pckgs_dir=/var/pckgs/apache
sudo mkdir -p "$pckgs_dir" ; cd "$pckgs_dir"

install_dir=/usr/local/hadoop
sudo mkdir -p "$install_dir" ; cd "$install_dir"
```

```
# download the hadoop package
curl -O 'http://mirror.netinch.com/pub/apache/hadoop/common/hadoop-2.7.4/hadoop-2.7.4.tar.gz'

# uncompress it
sudo gzip -dc hadoop-2.7.4.tar.gz | tar -C "$install_dir" -xvf -

# go back to the starting dir
cd -

echo "created the following dir"
sudo find $install_dir/hadoop-2.7.4/ -type d -maxdepth 2

sudo chown -Rv hduser:hadoop /usr/local/hadoop/
```

3.10.5. Edit the `hadoop-env.sh` file

Edit the `hadoop-env.sh` file

```
cd $product_instance_dir

# edit the hadoop-env.sh file as follows:
diff /usr/local/hadoop/hadoop-2.7.4/etc/hadoop/hadoop-env.sh cnf/hosts/host-name/usr/local/hadoop/hadoop-2.7.4/etc/hadoop/hadoop-env.sh

sudo vim -o `find /usr/local/hadoop -name hadoop-env.sh`
```

3.10.6. create the hadoop data dir

create the hadoop data dir

```
hadoop_tmp_data_dir=/var/hadoop/tmp
sudo mkdir -p "$hadoop_tmp_data_dir"
sudo chown -Rv hduser:hadoop $hadoop_tmp_data_dir
ls -al $hadoop_tmp_data_dir
```

3.10.7. copy the `map-reg.site.template` file

copy the `mapred-site.xml.template` file

```
sudo cp -v /usr/local/hadoop/hadoop-2.7.4/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/hadoop-2.7.4/etc/hadoop/mapred-site.xml
```

3.10.8. Edit the `hdfs-site.xml`

edit the `hdfs-site.xml` as follows

```
sudo mkdir -p /var/hadoop/dat/hdfs/name_node
sudo mkdir -p /var/hadoop/dat/hdfs/data_node
sudo chown -Rv hduser:hadoop /var/hadoop/
```



```
find /var/hadoop
```

```
sudo diff /usr/local/hadoop/hadoop-2.7.4/etc/hadoop/hdfs-site.xml cnf/hosts/host-name/usr/local/hadoop/hadoop-2.7.4/etc/hadoop/hdfs-site.xml
```

3.10.9. format the hadoop file system

format the hadoop file system as follows.

Note that hadoop namenode -format command should be executed once before we start using Hadoop, otherwise all the data will be destroyed ...

```
sudo cp -v /home/ysg/.hadoop_opts.host-name /home/hduser/
sudo cp -v /home/ysg/.profile_opts.host-name /home/hduser/
sudo su - root -c "echo 'source ~/.profile_opts.host-name ' >> /home/hduser/.bashrc"

sudo su - hduser -c "/usr/local/hadoop/hadoop-2.7.4/bin/hdfs namenode -format"

# SDOUT
# 17/09/18 14:46:48 INFO util.ExitUtil: Exiting with status 0
# 17/09/18 14:46:48 INFO namenode.NameNode: SHUTDOWN_MSG:
# /******
# SHUTDOWN_MSG: Shutting down NameNode at host-name/127.0.1.1
# *****/
```

3.10.10. start the single node cluster

After running the commands bellow point to the following url:

<http://host-name:50070/dfshealth.html#tab-overview>

```
cd /usr/local/hadoop/hadoop-2.7.4/sbin
bash start-dfs.sh

# 17/09/18 15:00:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes
# where applicable
# Starting namenodes on [localhost]
# localhost: namenode running as process 23006. Stop it first.
# localhost: datanode running as process 23169. Stop it first.
# Starting secondary namenodes [0.0.0.0]
# The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
# ECDSA key fingerprint is SHA256:2FhGsEKZO07xN4MPVcSVmSW0Clx9PKDtFANEqo3iHqg.
# Are you sure you want to continue connecting (yes/no)? yes
# 0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
# 0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/hadoop-2.7.4/logs/hadoop-hduser-secondarynamenode-host-
name.out
# 17/09/18 15:00:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes
# where applicable
#
```

```
start-yarn.sh
```

3.10.11. Verify that the everything is up-and-running

We can check if it's really up and running:

```
hduser@laptop:/usr/local/hadoop/sbin$
```

```
jps
23169 DataNode
24194 SecondaryNameNode
25306 ResourceManager
26429 NodeManager
23006 NameNode
26543 Jps
```

4. OPERATIONS

5. RUN THE EXAMPLES

You can run all the examples as follows:

```
# check the actions to run
cat src/bash/aspark-starter/tests/run-aspark-starter-tests.lst

# STDUOT
# sbt-compile-verbose
# sbt-clean-compile
# sbt-compile
# sbt-stage
# sbt-run

bash src/bash/aspark-starter/test-aspark-starter.sh

# now the tool will start producing output

# 2017-09-14 08:26:11    START test-aspark-starter test run report
#
# result start-time stop-time action-name
# ok 08:26:11 08:26:59 sbt-compile-verbose
# ok 08:27:00 08:27:25 sbt-clean-compile
# ok 08:27:25 08:27:34 sbt-compile
# ok 08:27:35 08:27:49 sbt-stage
# ok 08:27:49 08:27:59 sbt-run
```

6. INFORMATION SOURCES

This section contains good information sources

6.1. Overall tutorials

<https://www.edureka.co/blog/spark-tutorial/>

```
# check the actions to run
cat src/bash/aspark-starter/tests/run-aspark-starter-tests.lst

# STDUOT
# sbt-compile-verbose
# sbt-clean-compile
# sbt-compile
# sbt-stage
# sbt-run

bash src/bash/aspark-starter/test-aspark-starter.sh

# now the tool will start producing output

# 2017-09-14 08:26:11    START test-aspark-starter test run report
#
# result start-time stop-time action-name
# ok 08:26:11 08:26:59 sbt-compile-verbose
# ok 08:27:00 08:27:25 sbt-clean-compile
# ok 08:27:25 08:27:34 sbt-compile
# ok 08:27:35 08:27:49 sbt-stage
# ok 08:27:49 08:27:59 sbt-run
```