//////////////////Installing Spark

1. download spark from <http://spark.apache.org/downloads.html>

If spark already downloaded and we want to copy it from remote machine then use:

scp mabidm@remote-machine-ip:~/spark\* ./

2. tar -xvzf spark-1.5.2-bin-hadoop2.6.tgz -C /opt

here -C specifies the directory where spark will be decompressed

////checklist before start making cluster

1. make sure same ubuntu version is installed (not required but it's better to have because all pre-installed tools would be of the same versiov, e.g.python 2.7, openjdk 1.7 and so on. More it would be easy to manage OS)

2. make sure same jdk version is installed (not necessary but it would be easy to mention jdk paths in spark-env.sh)

3. Assign static IP addresses to machines.

4. change hostname if required using: “sudo gedit /etc/hostname” and then restart your system so that changes can take effect

5. Register machine names against IP address in DNS

6. comment out if hostname is mapped against ip address 127.0.1.1 in the /etc/hosts

 if running in standalone mode with DNS facility (same host mapped twice)

7. If running spark in local mode and no DNS facility then in /etc/hosts remove comments from

 127.0.1.1 spark-worker01

//////////////////////////Manual Standalone spark deployment

**-Master script**

Usage: ./sbin/start-master.sh [options]

Options:

-i HOST, --ip HOST Hostname to listen on (deprecated, please use

 --host or -h)

-h HOST, --host HOST Hostname to listen on

-p PORT, --port PORT Port to listen on (default: 7077)

--webui-port PORT Port for web UI (default: 8080)

--properties-file FILE Path to a custom Spark properties file.

 Default is conf/spark-defaults.conf.

Examples:

-./sbin/start-msater.sh --host 192.168.1.6

going to start spark master at machine 192.168.1.6

**-Slave script**

Usage: Worker [options] <master>

Master must be a URL of the form spark://hostname:port

Options:

 -c CORES, --cores CORES Number of cores to use for this work

 -m MEM, --memory MEM Amount of memory to use (e.g. 1000M, 2G)

 for this worker

 -d DIR, --work-dir DIR Directory to run apps in (default:

 SPARK\_HOME/work)

 -i HOST, --ip IP Hostname to listen on (deprecated, please

 use --host or -h)

 -h HOST, --host HOST Hostname to listen on

 -p PORT, --port PORT Port to listen on (default: random)

 --webui-port PORT Port for web UI (default: 8081)

 --properties-file FILE Path to a custom Spark properties file.

 Default is conf/spark-defaults.conf.

-./sbin/start-slave.sh --cores 2 spark://192.168.1.6:7077

Default Worker memory = (tot. memory -1 )GB

Worker will have 2 processing cores to run it tasks. Default: Use all cores of a machine to process tasks

**NOTE: Parameters precedence: highest to lowest**

**Spark Driver Program ->spark running shell scripts comand line param(e.g. spark-submit, start-slave,etc.) ->spark configuration files (e.g. spark-env)->spark source code**

Example:

./sbin/start-msater.sh --host 192.168.1.6

Spark Command: /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java -cp /home/mabidm/spark-1.5.2-bin-hadoop2.6/sbin/../conf/:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/spark-assembly-1.5.2-hadoop2.6.0.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-api-jdo-3.2.6.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-rdbms-3.2.9.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-core-3.2.10.jar -Xms1g -Xmx1g -XX:MaxPermSize=256m org.apache.spark.deploy.master.Master --host 172.30.13.54 --port 7077 --webui-port 8080 --host 192.168.1.6

“--host 172.30.13.54 “ specified in the ./conf/spark-env.sh using SPARK\_MASTER\_IP="172.30.13.54". However, value given at command prompt “--host 192.168.1.6” overriding value specified in the ./conf/spark-env.sh.

//////////////////////////////////////./sbin/start-all.sh ////////////////

./sbin/start-all.sh : it does not read any command line parameters

///////////////////////////////////////error////////////

when I used following command I was getting an error:

./sbin/start-slave.sh spark://spark-master:7077

**Error**

**+++++++++++++++**

Spark Command: /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java -cp /home/mabidm/spark-1.5.2-bin-hadoop2.6/sbin/../conf/:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/spark-assembly-1.5.2-hadoop2.6.0.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-api-jdo-3.2.6.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-rdbms-3.2.9.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-core-3.2.10.jar -Xms1g -Xmx1g -XX:MaxPermSize=256m org.apache.spark.deploy.worker.Worker --webui-port 8081 spark://172.30.13.246:7077

========================================

16/09/05 14:24:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

**16/09/05 14:24:43 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport**

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

16/09/05 14:24:44 WARN Utils: Service 'sparkWorker' could not bind on port 0. Attempting port 1.

16/09/05 14:24:44 ERROR Remoting: Remoting system has been terminated abrubtly. Attempting to shut down transports

16/09/05 14:24:44 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport

Exception in thread "main" java.net.BindException: Failed to bind to: /192.168.1.6:0: Service 'sparkWorker' failed after 16 retries!

 at org.jboss.netty.bootstrap.ServerBootstrap.bind(ServerBootstrap.java:272)

 at akka.remote.transport.netty.NettyTransport$$anonfun$listen$1.apply(NettyTransport.scala:393)

 at akka.remote.transport.netty.NettyTransport$$anonfun$listen$1.apply(NettyTransport.scala:389)

 at scala.util.Success$$anonfun$map$1.apply(Try.scala:206)

 at scala.util.Try$.apply(Try.scala:161)

 at scala.util.Success.map(Try.scala:206)

 at scala.concurrent.Future$$anonfun$map$1.apply(Future.scala:235)

 at scala.concurrent.Future$$anonfun$map$1.apply(Future.scala:235)

 at scala.concurrent.impl.CallbackRunnable.run(Promise.scala:32)

 at akka.dispatch.BatchingExecutor$AbstractBatch.processBatch(BatchingExecutor.scala:55)

 at akka.dispatch.BatchingExecutor$BlockableBatch$$anonfun$run$1.apply$mcV$sp(BatchingExecutor.scala:91)

 at akka.dispatch.BatchingExecutor$BlockableBatch$$anonfun$run$1.apply(BatchingExecutor.scala:91)

 at akka.dispatch.BatchingExecutor$BlockableBatch$$anonfun$run$1.apply(BatchingExecutor.scala:91)

 at scala.concurrent.BlockContext$.withBlockContext(BlockContext.scala:72)

 at akka.dispatch.BatchingExecutor$BlockableBatch.run(BatchingExecutor.scala:90)

 at akka.dispatch.TaskInvocation.run(AbstractDispatcher.scala:40)

 at akka.dispatch.ForkJoinExecutorConfigurator$AkkaForkJoinTask.exec(AbstractDispatcher.scala:397)

 at scala.concurrent.forkjoin.ForkJoinTask.doExec(ForkJoinTask.java:260)

 at scala.concurrent.forkjoin.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1339)

 at scala.concurrent.forkjoin.ForkJoinPool.runWorker(ForkJoinPool.java:1979)

 at scala.concurrent.forkjoin.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:107)

++++++++++++++++++++++++++++++++++++++

Error: **16/09/05 14:24:43 ERROR NettyTransport: failed to bind to /192.168.1.6:0, shutting down Netty transport**

**Sol: the ip address shown is from DSL switch installed at my home. When I restarts the computer the problem is gone.**

**///////////////////////Passwordless ssh //////////////////**

Execute these two commands on a master machine:

**ssh-keygen**

ssh-copy-id user@spark-worker01 (repeat this command for each worker on the master machine)

## or if your server uses custom port no:

ssh-copy-id "user@spark-worker01 -p 1234"

After the key is copied, ssh into the machine as normal

ssh user@spark-worker01

You can now login without entering a password from the particular machine you executed the commands at.

\*\*\*\*\*\*\*\*\*\*\*\*\*\* sample run

mabidm@spark-master:~/spark-1.5.2-bin-hadoop2.6/conf$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/mabidm/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/mabidm/.ssh/id\_rsa.

Your public key has been saved in /home/mabidm/.ssh/id\_rsa.pub.

The key fingerprint is:

15:90:9d:71:57:fc:12:44:1d:c5:81:0d:84:55:ee:35 mabidm@spark-master

The key's randomart image is:

+--[ RSA 2048]----+

| .+oo=\*OO\*|

| . o+ oo.+|

| . Eo|

| . o +|

| S o |

| |

| |

| |

| |

+-----------------+

mabidm@spark-master:~/spark-1.5.2-bin-hadoop2.6/conf$ ssh-copy-id mabidm@spark-worker01

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

mabidm@spark-worker01's password:

Permission denied, please try again.

mabidm@spark-worker01's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'mabidm@spark-worker01'"

and check to make sure that only the key(s) you wanted were added.

mabidm@spark-master:~/spark-1.5.2-bin-hadoop2.6/conf$ ssh mabidm@spark-worker01

Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.16.0-30-generic x86\_64)

 \* Documentation: https://help.ubuntu.com/

167 packages can be updated.

95 updates are security updates.

Last login: Fri Jun 3 17:39:46 2016 from localhost

it's /etc/bash.bashrc

its /etc/profile

it's /home/mabidm/.bashrc

it's /home/mabidm/.profile

mabidm@spark-worker01:~$

\*\*\*\*\*\*\*\*\*\*\*\*

////////////Getting info about processor and memory

1. lscpu

2. cat /proc/cpuinfo

3. cat /proc/meminfo

**Using GUI**

gnome-system-monitor

/////////////////////////////////////Automatic Standalone spark deployment OR Cluster Launch Scripts

1. specify all workers in ./conf/slaves [just for master machine]. No need to copy it on each worker machine

2. specify all parameters in ./conf/spark-env.sh. e.g master, worker parameters,etc. Copy it on all worker machines. Worker needs values specified in the said file.

3. run ./sbin/start-all.sh on a master machine

4. open browser and type [http://spark-master:8080](http://spark-master:8080/) (make sure in your browser that spark-master is specified in list of “no proxy for” if using proxy server). We should find all workers in running state

5. [http://spark-master:8080](http://spark-master:8080/)can be used from remote browser

**Important:**

-use ps -aux | grep java to see if java processes running on spark master and worker

-if hostname resolves to loopback address 127.0.1.1 then use following command to remove this association (comment out line showing 127.0.1.1):

sudo gedit /ect/hosts

-I was getting an error message at all workers: “failed to register to master”. This message was because of a master hostname mapping to loopback.

**Amount of parallelism: Number of simultaneous executors running in all worker nodes and master**

**-for each partition of data, we have one task that can be scheduled on a executor**

-SPARK\_WORKER\_CORES=2 and SPARK\_WORKER\_MEMORY=3G: These resources are allocated to each executor. For example:

 01: Assume --executor-cores 2 and --executor-memory 1gb then each worker can only create one executor that can run two parallel taks, one on each core (assuming one thread/core)

 02: Assume --executor-cores 1 and --executor-memory 1gb then each worker can create two executors that can run two parallel tasks, one per executor per core (assuming one thread/core)

**////////////////////////////////Connecting SPARK shell to the Cluster**

**1.** ./bin/spark-shell –master spark://spark-master:7077 --name “practice” --total-executor-cores 8 --executor-cores 2 --executor-memory 1gb

**If we just use “**./bin/spark-shell –master spark://spark-master:7077 --name “practice”**” then each executor (One executor launched on each worker machine)will occupy all cores on the worker and 1024MB memory (This can be confirmed from master URL, e.g. spark-master:8080 by looking at “Running Applications”). So, if we launch application like this it's going to occupy all cores of the whole cluster specified using spark://spark-master:7077**

./bin/pyspark --help

Usage: ./bin/pyspark [options]

Options:

 --master MASTER\_URL spark://host:port, mesos://host:port, yarn, or local.

 --deploy-mode DEPLOY\_MODE Whether to launch the driver program locally ("client") or

 on one of the worker machines inside the cluster ("cluster") (Default: client).

 --class CLASS\_NAME Your application's main class (for Java / Scala apps).

 --name NAME A name of your application.

 --jars JARS Comma-separated list of local jars to include on the driver and executor classpaths.

 --packages Comma-separated list of maven coordinates of jars to include

 on the driver and executor classpaths. Will search the local

 maven repo, then maven central and any additional remote

 repositories given by --repositories. The format for the

 coordinates should be groupId:artifactId:version.

 --exclude-packages Comma-separated list of groupId:artifactId, to exclude while

 resolving the dependencies provided in --packages to avoid

 dependency conflicts.

 --repositories Comma-separated list of additional remote repositories to

 search for the maven coordinates given with --packages.

 --py-files PY\_FILES Comma-separated list of .zip, .egg, or .py files to place

 on the PYTHONPATH for Python apps.

 --files FILES Comma-separated list of files to be placed in the working

 directory of each executor.

 --conf PROP=VALUE Arbitrary Spark configuration property.

 --properties-file FILE Path to a file from which to load extra properties. If not

 specified, this will look for conf/spark-defaults.conf.

 --driver-memory MEM Memory for driver (e.g. 1000M, 2G) (Default: 1024M).

 --driver-java-options Extra Java options to pass to the driver.

 --driver-library-path Extra library path entries to pass to the driver.

 --driver-class-path Extra class path entries to pass to the driver. Note that

 jars added with --jars are automatically included in the

 classpath.

 --executor-memory MEM Memory per executor (e.g. 1000M, 2G) (Default: 1G).

 --proxy-user NAME User to impersonate when submitting the application.

 --help, -h Show this help message and exit

 --verbose, -v Print additional debug output

 --version, Print the version of current Spark

 Spark standalone with cluster deploy mode only:

 --driver-cores NUM Cores for driver (Default: 1).

 Spark standalone or Mesos with cluster deploy mode only:

 --supervise If given, restarts the driver on failure.

 --kill SUBMISSION\_ID If given, kills the driver specified.

 --status SUBMISSION\_ID If given, requests the status of the driver specified.

 Spark standalone and Mesos only:

 --total-executor-cores NUM Total cores for all executors.

 Spark standalone and YARN only:

 --executor-cores NUM Number of cores per executor. (Default: 1 in YARN mode,

 or all available cores on the worker in standalone mode)

 YARN-only:

 --driver-cores NUM Number of cores used by the driver, only in cluster mode

 (Default: 1).

 --queue QUEUE\_NAME The YARN queue to submit to (Default: "default").

 --num-executors NUM Number of executors to launch (Default: 2).

 --archives ARCHIVES Comma separated list of archives to be extracted into the

 working directory of each executor.

 --principal PRINCIPAL Principal to be used to login to KDC, while running on

 secure HDFS.

 --keytab KEYTAB The full path to the file that contains the keytab for the

 principal specified above. This keytab will be copied to

 the node running the Application Master via the Secure

 Distributed Cache, for renewing the login tickets and the

 delegation tokens periodically.

**////////////////////////////////Running standalone python application on the Cluster**

-In the driver program create SparkContext Object:

**Example program:**

from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local[\*]").setAppName("file.py")

sc = SparkContext(conf=conf)

text = sc.textFile("/home/mabidm/readme.md")

ftext = text.filter(lambda line: "Spark" in line)

print ( ftext.count() )

save above code in a file called python-file.py. now use the following command to submit application to spark cluster:

**./bin/spark-submit** python-file.py

python-file.py specifies that this application is going to run on a local machine. Only one JVM process is created that includes driver (i.e. python-file.py) and executors occupying all cores of a machine. No master and worker jvm process running. To confirm this, execute following command on a local machine:

mabidm@spark-worker01:~$ ps -aux|grep java

mabidm 16475 139 1.2 2385520 104656 pts/11 Sl+ 13:32 0:01 /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java -cp /home/mabidm/spark-1.5.2-bin-hadoop2.6/conf/:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/**spark-assembly-1.5.2-hadoop2.6.0.jar**:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/**datanucleus-api-jdo-3.2.6.jar**:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/**datanucleus-rdbms-3.2.9.jar**:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/**datanucleus-core-3.2.10.jar** -Xms1g -Xmx1g -XX:MaxPermSize=256m org.apache.spark.deploy.SparkSubmit --master local[\*] --class X **./target/scala-2.10/learning-spark-mini-example\_2.10-0.0.1.jar** /home/mabidm/readme.md

**////////////////////////////////// spark-submit help**

mabidm@spark-master:~/spark-1.5.2-bin-hadoop2.6$ ./bin/spark-submit --help

**Usage: spark-submit [options] <app jar | python file> [app arguments]**

**Usage: spark-submit --kill [submission ID] --master [spark://...]**

**Usage: spark-submit --status [submission ID] --master [spark://...]**

Options:

 **--master MASTER\_URL**  spark://host:port, mesos://host:port, yarn, or local.

 **--deploy-mode** DEPLOY\_MODE Whether to launch the driver program locally ("client") or

 on one of the worker machines inside the cluster ("cluster") (Default: client).

 **--class CLASS\_NAME** Your application's main class (for Java / Scala apps).

 **--name NAME** A name of your application.

  **--jars JARS** Comma-separated list of local jars to include on the driver

 and executor classpaths.

  **--packages** Comma-separated list of maven coordinates of jars to include

 on the driver and executor classpaths. Will search the local

 maven repo, then maven central and any additional remote

 repositories given by --repositories. The format for the

 coordinates should be groupId:artifactId:version.

 --exclude-packages Comma-separated list of groupId:artifactId, to exclude while

 resolving the dependencies provided in --packages to avoid

 dependency conflicts.

 --repositories Comma-separated list of additional remote repositories to

 search for the maven coordinates given with --packages.

  **--py-files PY\_FILES** Comma-separated list of .zip, .egg, or .py files to place

 on the PYTHONPATH for Python apps.

 **--files FILES** Comma-separated list of files to be placed in the working

 directory of each executor.

 --conf PROP=VALUE Arbitrary Spark configuration property.

 **--properties-file FILE** Path to a file from which to load extra properties. If not

 specified, this will look for conf/spark-defaults.conf.

 --driver-memory MEM Memory for driver (e.g. 1000M, 2G) (Default: 1024M).

 --driver-java-options Extra Java options to pass to the driver.

 --driver-library-path Extra library path entries to pass to the driver.

 --driver-class-path Extra class path entries to pass to the driver. Note that

 jars added with --jars are automatically included in the

 classpath.

  **--executor-memory MEM** Memory per executor (e.g. 1000M, 2G) (Default: 1G).

  **--proxy-user NAME** User to impersonate when submitting the application.

 --help, -h Show this help message and exit

 --verbose, -v Print additional debug output

 --version, Print the version of current Spark

 Spark standalone with cluster deploy mode only:

 --driver-cores NUM Cores for driver (Default: 1).

 Spark standalone or Mesos with cluster deploy mode only:

 **--supervise** If given, restarts the driver on failure.

 --kill SUBMISSION\_ID If given, kills the driver specified.

 --status SUBMISSION\_ID If given, requests the status of the driver specified.

 Spark standalone and Mesos only:

 **--total-executor-cores** NUM Total cores for all executors.

 Spark standalone and YARN only:

 **--executor-cores** NUM Number of cores per executor. (Default: 1 in YARN mode,

 or all available cores on the worker in standalone mode)

 YARN-only:

 --driver-cores NUM Number of cores used by the driver, only in cluster mode

 (Default: 1).

 --queue QUEUE\_NAME The YARN queue to submit to (Default: "default").

 --num-executors NUM Number of executors to launch (Default: 2).

 --archives ARCHIVES Comma separated list of archives to be extracted into the

 working directory of each executor.

 --principal PRINCIPAL Principal to be used to login to KDC, while running on

 secure HDFS.

 --keytab KEYTAB The full path to the file that contains the keytab for the

 principal specified above. This keytab will be copied to

 the node running the Application Master via the Secure

 Distributed Cache, for renewing the login tickets and the

 delegation tokens periodically.

**Note: -pyspark, spark-shell, spark-sql and spark-submit use the same command line options**

 **-If some input file is being read in the spark program then that file must exist on all worker and**

 **master machines. Otherwise, we will get an error.**

- /bin/spark-submit --master spark://spark-master:7077 --name “try” --total-executor-cores 8 --executor-cores 2 --executor-memory 1gb /home/mabidm/Spark-practice/error-warn.py :

This command will launch driver locally i.e. client mode = separate java process will be created on the machine where driver program is launched. For example:

/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/mabidm/spark-1.5.2-bin-hadoop2.6/conf/:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/spark-assembly-1.5.2-hadoop2.6.0.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-api-jdo-3.2.6.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-rdbms-3.2.9.jar:/home/mabidm/spark-1.5.2-bin-hadoop2.6/lib/datanucleus-core-3.2.10.jar -Xms1g -Xmx1g org.apache.spark.deploy.SparkSubmit --master spark://spark-master:7077 --name “try” --total-executor-cores 8 --executor-cores 2 --executor-memory 1gb /home/mabidm/Spark-practice/error-warn.py

**Right now for python application, cluster mode in standalone cluster is not supported. Check spark version > 1.5.2**

/////////////////////////////////Running scala shell

mabidm@spark-worker01:~/spark-1.5.2-bin-hadoop2.6$ ./bin/spark-shell

16/09/01 10:57:26 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Welcome to

 \_\_\_\_ \_\_

 / \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

 \_\ \/ \_ \/ \_ `/ \_\_/ '\_/

 /\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 1.5.2

 /\_/

**Using Scala version 2.10.4 (OpenJDK 64-Bit Server VM, Java 1.7.0\_101)**

Type in expressions to have them evaluated.

Type :help for more information.

16/09/01 10:57:32 WARN SparkConf:

**SPARK\_WORKER\_INSTANCES was detected (set to '2').**

This is deprecated in Spark 1.0+.

**Please instead use:**

 **- ./spark-submit with --num-executors to specify the number of executors**

 **- Or set SPARK\_EXECUTOR\_INSTANCES**

 **- spark.executor.instances to configure the number of instances in the spark config.**

16/09/01 10:57:34 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

Spark context available as sc.

16/09/01 10:57:37 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)

16/09/01 10:57:38 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)

16/09/01 10:57:57 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0

16/09/01 10:57:57 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException

16/09/01 10:58:01 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/09/01 10:58:01 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)

16/09/01 10:58:01 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)

16/09/01 10:58:19 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0

16/09/01 10:58:20 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException

SQL context available as sqlContext.

/////////////////Running Scala application on Spark Cluster

step 01. Create sbt project:

1. mkdir newproject

2. cd newproject

3. gedit build.sbt

name := "learning-spark-mini-example"

version := "0.0.1"

scalaVersion := "2.10.4"

// additional libraries

libraryDependencies ++= Seq(

 "org.apache.spark" %% "spark-core" % "1.5.2" % "provided"

)

4. mkdir -p src/{main,test}/{java,resources,scala}

5. mkdir lib project target

6. Write WordCount example and save it in src/main/scala/WordCount.scala

package com.oreilly.learningsparkexamples.mini.scala

import org.apache.spark.\_

import org.apache.spark.SparkContext.\_

object WordCount {

 def main(args: Array[String]) {

 val inputFile = args(0)

 val outputFile = args(1)

 val conf = new SparkConf().setAppName("wordCount")

 // Create a Scala Spark Context.

 val sc = new SparkContext(conf)

 // Load our input data.

 val input = sc.textFile(inputFile)

 // Split up into words.

 val words = input.flatMap(line => line.split(" "))

 // Transform into word and count.

 val counts = words.map(word => (word, 1)).reduceByKey{case (x, y) => x + y}

 // Save the word count back out to a text file, causing evaluation.

 counts.saveAsTextFile(outputFile)

 }

}

7. sbt clean && sbt package

8. /home/mabidm/spark-1.5.2-bin-hadoop2.6/bin/spark-submit –class com.oreilly.learningsparkexamples.mini.scala .WordCount

./target/scala-2.10/learning-spark-mini-example\_2.10-0.0.1.jar

./README.md ./WordCount

////////////////////////////////////////////

**Verifying that pyspakr shell (i.e. driver) and all its tasks are run in a single thread**

**Step1. run pyspark**

**$ pyspark**

**this will launch pyspark in web browser as following variables are set in /etc/profile**

**export** **PYSPARK\_DRIVER\_PYTHON\_OPTS='notebook'**

**export PYSPARK\_DRIVER\_PYTHON=ipython**

**Step2: $ps -aux | grep java**

**No JVM is running as this output shows. This implies that just running pyspark does not start any jvm (Exception: If not running pyspark in a notebook i.e. without setting PYSPARK\_DRIVER\_PYTHON\_OPTS='notebook' and by issuing command PYSPARK\_DRIVER\_PYTHON=ipython pyspark then it launches jvm even before running any application)**

**Step 3: Create some notebook**

**This will launch single executor jvm that will run both the driver and tasks. Right now no task is running as we have not yet run any code in the notebook. Only driver is running now.**

**Note: Running multiple notebooks will launch multiple jvms but built-in SparkContext 'sc' is available in the first notebook started.**

**Step5: Confirming single jvm is running**

**$ps -aux | grep java**

**mabidm 7031 1.1 5.3 4656612 433836 ? Sl 19:17 0:38 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /opt/spark-2.1.0-bin-hadoop2.7/conf/:/opt/spark-2.1.0-bin-hadoop2.7/jars/\* -Xmx1g org.apache.spark.deploy.SparkSubmit --name PySparkShell pyspark-shell**

**here**

**-cp <class search path of directories and zip/jar files> A : separated list of directories, JAR archives, and ZIP archives to search for class files.**

**org.apache.spark.deploy.SparkSubmit is the name of class being executed**

**--name PySparkShell pyspark-shell are the args to class Spark.Submit**

**Spark.Submit class:**

|  |
| --- |
| This program handles setting up the classpath with relevant Spark dependencies and provides |
| \* a layer over the different cluster managers and deploy modes that Spark supports. |

**See the code of “org.apache.spark.deploy.SparkSubmit” more insights**

**Eight tasks are running for the following code as eight partitions are created:**

**text = sc.textFile("/opt/spark-2.1.0-bin-hadoop2.7/README.md",8)**

**ftext = text.filter(lambda line: ("Spark" in line) or ("Python" in line))**

**print (ftext.count())**

****

//////////

**pyspark /usr/bin/env: bash: No such file or directory**

**When I started pyspark I wa getting /usr/bin/env: bash: No such file or directory error.**

**Sol: include /usr/bin in the PATH variable**

**export PATH= /usr/bin:$PATH**

**//////////////////////////// Spark Options**

[**https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/deploy/SparkSubmit.scala**](https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/deploy/SparkSubmit.scala)

|  |  |
| --- | --- |
| /\*\* |  |
|  | \* Main gateway of launching a Spark application. |
|  | \* |
|  | \* This program handles setting up the classpath with relevant Spark dependencies and provides |
|  | \* a layer over the different cluster managers and deploy modes that Spark supports. |
|  | \*/ |

|  |
| --- |
| // All cluster managers |
| OptionAssigner(args.master, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, sysProp = "spark.master"), |
| OptionAssigner(args.deployMode, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.submit.deployMode"), |
| OptionAssigner(args.name, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, sysProp = "spark.app.name"), |
| OptionAssigner(args.ivyRepoPath, ALL\_CLUSTER\_MGRS, CLIENT, sysProp = "spark.jars.ivy"), |
| OptionAssigner(args.driverMemory, ALL\_CLUSTER\_MGRS, CLIENT, |
| sysProp = "spark.driver.memory"), |
| OptionAssigner(args.driverExtraClassPath, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.driver.extraClassPath"), |
| OptionAssigner(args.driverExtraJavaOptions, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.driver.extraJavaOptions"), |
| OptionAssigner(args.driverExtraLibraryPath, ALL\_CLUSTER\_MGRS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.driver.extraLibraryPath"), |
|  |
| // Yarn only |
| OptionAssigner(args.queue, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.queue"), |
| OptionAssigner(args.numExecutors, YARN, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.executor.instances"), |
| OptionAssigner(args.jars, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.dist.jars"), |
| OptionAssigner(args.files, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.dist.files"), |
| OptionAssigner(args.archives, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.dist.archives"), |
| OptionAssigner(args.principal, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.principal"), |
| OptionAssigner(args.keytab, YARN, ALL\_DEPLOY\_MODES, sysProp = "spark.yarn.keytab"), |
|  |
| // Other options |
| OptionAssigner(args.executorCores, STANDALONE | YARN, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.executor.cores"), |
| OptionAssigner(args.executorMemory, STANDALONE | MESOS | YARN, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.executor.memory"), |
| OptionAssigner(args.totalExecutorCores, STANDALONE | MESOS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.cores.max"), |
| OptionAssigner(args.files, LOCAL | STANDALONE | MESOS, ALL\_DEPLOY\_MODES, |
| sysProp = "spark.files"), |
| OptionAssigner(args.jars, LOCAL, CLIENT, sysProp = "spark.jars"), |
| OptionAssigner(args.jars, STANDALONE | MESOS, ALL\_DEPLOY\_MODES, sysProp = "spark.jars"), |
| OptionAssigner(args.driverMemory, STANDALONE | MESOS | YARN, CLUSTER, |
| sysProp = "spark.driver.memory"), |
| OptionAssigner(args.driverCores, STANDALONE | MESOS | YARN, CLUSTER, |
| sysProp = "spark.driver.cores"), |
| OptionAssigner(args.supervise.toString, STANDALONE | MESOS, CLUSTER, |
| sysProp = "spark.driver.supervise"), |
| OptionAssigner(args.ivyRepoPath, STANDALONE, CLUSTER, sysProp = "spark.jars.ivy") |

/////////////////proxy settings

1. bin/pysparkl --conf "spark.driver.extraJavaOptions=-Dhttp.proxyHost=<proxyHost> -Dhttp.proxyPort=<proxyPort> -Dhttps.proxyHost=<proxyHost> -Dhttps.proxyPort=<proxyPort> -Dhttp.proxyUser=<name> -Dhttp.proxyPassword=<passwd> -Dhttps.proxyUser=<name> -Dhttps.proxyPassword=<passwd> " --packages <somePackage>

2. bin/pysparkl –driver-java-options "-Dhttp.proxyHost=<proxyHost> -Dhttp.proxyPort=<proxyPort> -Dhttps.proxyHost=<proxyHost> -Dhttps.proxyPort=<proxyPort> -Dhttp.proxyUser=<name> -Dhttp.proxyPassword=<passwd> -Dhttps.proxyUser=<name> -Dhttps.proxyPassword=<passwd> " --packages <somePackage>

3. gedit conf/spark-defaults.conf

spark.driver.extraJavaOptions "-Dhttp.proxyHost=<proxyHost> -Dhttp.proxyPort=<proxyPort> -Dhttps.proxyHost=<proxyHost> -Dhttps.proxyPort=<proxyPort> -Dhttp.proxyUser=<name> -Dhttp.proxyPassword=<passwd> -Dhttps.proxyUser=<name> -Dhttps.proxyPassword=<passwd> "

////

**object** SQLProgGuide {

 **def** main(args: Array[*String*]) {

 //assuming SparkSession named spark is created

 **import** spark.implicits.\_

 **case** **class** **Person**(name: *String*, age: Long)

// Encoders are created for case classes

 **val** caseClassDS = Seq(**Person**("Andy", 32)).toDS()

 caseClassDS.show()

 //

 **val** path = "/opt/spark-2.1.0-bin-hadoop2.7/examples/src/main/resources/people.json"

 **val** peopleDS = spark.read.json(path).as[**Person**]

 peopleDS.show()

}

This is not working in Spark 2.1. However, if we move case class out of main method then its working

**object** SQLProgGuide {

 **case** **class** **Person**(name: *String*, age: Long)

 **def** main(args: Array[*String*]) {

 //assuming SparkSession named spark is created

 **import** spark.implicits.\_

.

.

}

//

SparkContext

Main entry point for Spark functionality. A SparkContext represents the connection to a Spark cluster, and can be used to create RDDs, accumulators and broadcast variables on that cluster.

Only one SparkContext may be active per JVM. You must stop() the active SparkContext before creating a new one. This limitation may eventually be removed; see SPARK-2243 for more details.

SparkConf

Once a SparkConf object is passed to Spark, it is cloned and can no longer be modified by the user. Spark does not support modifying the configuration at runtime.

Changing Configurations

**var** spark = SparkSession

 .builder()

 .master("local[\*]")

 .appName("Spark SQL basic example")

 .config("key1", "value1")

 .getOrCreate()

 //verifying that same value exists in SparkContext

 println(spark.conf.get("key1") == spark.sparkContext.getConf.get("key1"))

 //adding anothor configuration

**val** spark2 = SparkSession.builder.config("key2", "value2").getOrCreate()

println(spark2.conf.get("key2") == spark2.sparkContext.getConf.get("key2"))

We will receive a warning:

WARN SparkSession$Builder: Using an existing SparkSession; some configuration may

not take effect

**The problem is that Spark does not support modifying the configuration at runtime (with exception SQLContext configuration can be modified on runtime). Once a SparkConf object is passed to Spark, it is cloned and can no longer be modified by the user**

**Solution: First stop the SparkContext using SparkSession object**

//first stop existing SparkContext

spark.stop()

//new configurations

//If we want old configurations from previous SparkSession object then we have to add them manually

**val** conf = **new** SparkConf().setAppName("Testing").setMaster("local[\*]")

**val** spark2 = SparkSession.builder.config(conf).getOrCreate()

println(spark2.sparkContext.getConf.toDebugString)

//Sets the default SparkSession that is returned by the builder

SparkSession.setDefaultSession(spark2)

//setting old spark to new SparkSession

spark = SparkSession.builder.getOrCreate()

//verifying keys

**println(spark.sparkContext.getConf.toDebugString)**

println(spark2.sparkContext.getConf.toDebugString)